А К О Е Н Е О Б Х О Д И М О

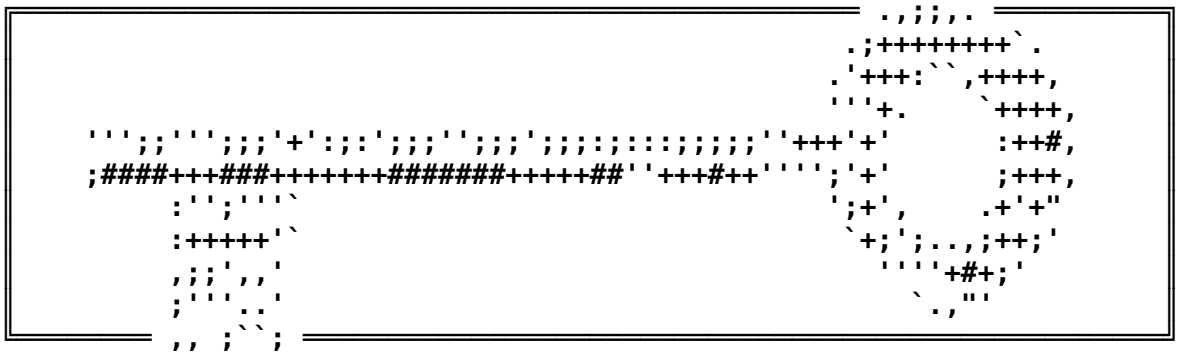
ШИФРОВАЙТЕ

Следват напътствия за надеждно шифроване на вашата поверителна информация. Това е важно, ако мислите да обработвате и съхранявате такава информация в електронна среда или да я споделяте по електронен път. Въпреки, че на места напътствията са трудни за прилагане от обикновения компютърен потребител (поне докато се преодолеят известни предразсъдъци и задръжки), изпълнението им е постижимо „в домашни условия“, изцяло с достъпни за гражданска употреба средства. Нищо от предложените напътствия не зависи от един единствен производител или доставчик, а правилното функциониране на всеки от компонентите може да се потвърди от поне няколко независими и алтернативни източника. Това е ключов момент в нашата концепция, чието цялостно и точно прилагане ще осигури респектиращи нива на информационна сигурност. Но още тук честно ще отбележим, че „абсолютна сигурност“ няма. Помислите отново, преди да въведете поверителна информация в електронна среда и просто не го правете, ако не е наистина необходимо!

Това ръководство е резултат от работата на **www.Advocati.org**
и Института за Свободни технологии **www.LibTec.org**
със специални благодарности към **Richard M. Stallman** и **Denis 'GNUtoo' Carikli**

https://www.advocati.org/
https://www.libtec.org/

Прилагането на това ръководство е изцяло на ваша отговорност. Никой от авторите не може да бъде държан отговорен за каквито и да било вреди, произтекли от цялостно или частично прилагане на представените технологии и методи.



Официалната версия на това ръководство е в електронно „подписан“ **in-need-encrypt-bg.asc** файл с обикновен текстови формат, ползваем включително от система без графична среда и достъпен на този адрес:

https://advocati.org/consultation/security/in-need-encrypt/
https://libtec.org/in_need_encrypt/

ВЪВЕДЕНИЕ

Адвокатската тайна е призната за ваше основно право по чл. **30**, ал. **5** от **Конституцията** и е вменена като основно задължение на адвокатите по чл. **45** от **Закона за адвокатурата** (скрепено дори с адвокатската клетва). Въпреки това не са изолирани случаите на грубо погазване на това право (задължение), включително и под натиск от официалните власти. Като единствена действителна гаранция за опазване на тайната остава прилагането на адекватни организационни и технически мерки за информационна сигурност. Това включва комплекс от взаимосвързани условия, които трябва да се приложат точно и цялостно. По-долу прилагаме нашите напътствия за информационна сигурност, за изпълнението на които не се нуждаете от друго, освен от базова компютърна грамотност, приятелско отношение към електрониката и внимателно следване на изложението. Ако се нуждаете от допълнително съдействие, можете да се свържете с нас или с ваш доверен съветник за по-нататъшни указания.

СЪДЪРЖАНИЕ

Цялостната организация на изложението е ориентирана преди всичко към Обикновения потребител, който се нуждае от основни познания и последователно въвеждане в темата. Препоръчваме да се запознаете с ръководството изцяло, преди да пристъпите към прилагането на която и да било негова част. Това е така, защото изложението се отклонява от практическата последователност на предложените стъпки – поради някои дидактически съображения. Например защитата на софтуера от „ниско“ ниво (**III**) предхожда цялостното шифроване на системата (**IV**), а цялостното шифроване от своя страна предхожда самото инсталиране на вашата операционна система (**II**). Въпреки това допускаме определена (не)последователност, защото чрез нея ще предадем знания и умения, при които всяка следваща стъпка базира върху предходните. Така, докато разглеждаме една напълно свободна операционна система (**I**), ще отделим специално внимание на работата чрез директни текстови команди към компютъра – което е фундаментално за установяването на истински контрол над системата и за обезпечаване на сигурността. Докато разглеждаме процеса по инсталиране на една свободна операционната система (**II**), ще навлезем в архитектурата на системата, в протичането на цялостния инсталационен процес и в последващото настройване чрез текстово редактиране на някои от основните системни файлове. Тези знания и умения от своя страна ще бъдат необходими за защита на софтуера от „ниско“ ниво (**III**), без което не можем да приемем за надеждно последващото цялостното шифроване на системата (**IV**). На следващо място – без да сме свикнали с текстовите команди към компютъра и без да сме изяснили как се обезпечава сигурността на паролите (**V**), би било напълно безсмислено да пристъпваме към каквито и да било опити за извършването на надеждно **GPG**-шифроване (**VIII**). Без да сме взели мерки за защита на свързването с internet (**VI**) и за организиране на физическата сигурност (**VII**), никое от досегашните ни усилия няма особен смисъл за обезпечаване на сигурността. Ето защо ви насърчаваме да разглеждате изложението като едно неделимо цяло, да не „прескачате“ през неговите части и да не бързате да прилагате отделни елементи от него, преди да сте взели предвид неговата интегрална свързаност. А на онези от вас, за които пространните ни обяснения относно „елементарни“ и „очевидни“ неща са излишни, поднасяме искрени извинения и молим да проявите търпение – като не забравяте, че преди време именно подобни обяснения и напътствия са ви превърнали в това, което сте днес.

I. НАПЪЛНО СВОБОДНА ОПЕРАЦИОННА СИСТЕМА

Дори и най-надеждното шифроване загубва своя смисъл, ако се осъществява от несигурен компютър. Такъв компютър може да бъде компрометиран, ако се допусне проникване в неговата система и се прегледа съдържанието преди да бъде шифровано; ако системата бъде компрометирана по начин, че сама да изпрати към трети страни поверителната информация; ако се проследят действията по самото шифроване и се извлекат шифровъчните ключове и пароли; или ако се компрометират средствата, с които се извършва шифроването и се отслаби неговата надеждност. За да се ограничат подобни рискове, трябва да се ползва надеждно компютърно оборудване, инсталирано с надежден софтуер. И разбира се – оборудването и софтуерът да се ползват разумно.

1. Защо Свободният софтуер е от решаващо значение

Само софтуер, чийто изходен код (**Source Code**) е публично достъпен и може да се усъвършенства от цялата световна общност от програмисти, може да бъде надежден. Ако изходният код на софтуера е „затворен“ (скрит), не означава, че пази някаква тайна; нито означава, че е по-сигурен. Публичната достъпност на изходния код гарантира, че всички (а не само специалистите в определена компания-собственик) могат да изучават детайлно начина на функциониране на софтуера и да посочват слабости в него. В противен случай рискуваме компанията-собственик, която единствена има достъп до изходния код, да пропусне определени слабости или дори да се насочи към политики, които не винаги гарантират сигурността, а дори тъкмо обратното. Публичната достъпност на изходния код гарантира, че в софтуера не могат да бъдат вграждани скрити „задни врати“ (**Backdoors**) за неговото компрометиране – защото това не би могло да остане незабелязано от програмистите и поне някои от тях ще огласят публично този факт. Респективно, всеки ще може да вземе мерки срещу установения пробив в сигурността. И не на последно място – възможността всички програмисти по света да се включат в работата (а не само одобрените от компанията-собственик) позволява максимален брой умове да дадат своя принос за усъвършенстване на софтуера. Следователно – вие не сте ограничени до предложения кръг от специалисти на компанията-собственик, а можете да се доверите за проучване, коригиране и адаптиране на софтуера към вашите нужди на предпочитани от вас специалисти или дори да извършите това сами (ако положите усилия да овладеете необходимите знания и умения).

https://en.wikipedia.org/wiki/Source_code
https://en.wikipedia.org/wiki/Backdoor_(computing)

На горните изисквания отговаря единствено Свободният софтуер. Под „свободен“ софтуер не разбираме непременно „безплатен“ софтуер – а такъв, при който се уважава свободата на потребителите. Истински свободен (като във философското понятие „свобода“) е например софтуерът, който се разпространява съгласно правилата на лиценза **GPL** (**General Public License**), утвърдени в проекта за Свободен софтуер **GNU** (рекурсивен акроним от „**GNU is Not UNIX**“, където '**UNIX**' е основната операционна система от **80**$^{\text{-те}}$ години на **XX** в., с несвободен код). Проектът **GNU** се поддържа от **Фондацията за Свободен софтуер** (**Free Software Foundation**). Специфичното при такъв вид софтуер е, че всеки, включително и вие, без да се нуждаете от предварително разрешение и без да дължите каквото и да било плащане на някого, получавате следните четири свободи:

(**0**) **да ползвате софтуера за вашите нужди**;
(**1**) **да изследвате и променяте неговия изходен код**;
(**2**) **да споделяте софтуера в първоначалния му вид**; и
(**3**) **да споделяте софтуера в променения му вид**.

Забранява се единствено „да затваряте“ кода – Свободният софтуер трябва да продължи да бъде свободен за всеки желаещ и занапред. Така бизнесът се измества от даването под наем на срочни лицензи за ползване на вече създаден софтуер към предоставянето на частни софтуерни услуги. Тези услуги обаче са в интерес на цялата общност (не само на клиенти и собственици), тъй като са насочени към подобряване на софтуера и към решаване на конкретни частни проблеми в него; и всичките подобрения и нови технологични решения, за чието изготвяне някой е платил или ги е изготвил с доброволния си труд (както например ние сме работили по това ръководство) стават свободно достъпни и облагодетелстват цялата общност. И всеки е поканен да се включи в по-нататъшното подобряване и решаване на

проблеми – което трябва да продължи да бъде свободно достъпно и да облагодетелства цялата общност. Не е за учудване в този контекст, че Свободният софтуер се развива много по-бързо от несвободния и предоставя много повече възможности от него.

https://bg.wikipedia.org/wiki/GNU_General_Public_License
https://en.wikipedia.org/wiki/Unix
https://www.gnu.org/gnu/thegnuproject.en.html
https://www.fsf.org/

Повечето софтуерни продукти на компании като **Microsoft** и **Apple** очевидно не са свободни. Вие нямате право да ползвате тези продукти по начин, различен от посочения в техните лицензи. Това включва забрана да ги променяте и адаптирате към вашите нужди (включително към нуждите ви, свързани със сигурността). Забранено е дори да узнаете как точно работят такива продукти (включително дали компрометират по някакъв начин вашата сигурност). Следователно, вашето доверяване на компютър с **Windows** или **MacOS**, или на телефон с **iOS** или дори **Android** (предлаган уж като свободен, но зависим от напълно несвободни ключови компоненти за функционирането му), е доверяване „на сляпо“ – с надеждата (лишена от каквито и да било обективни доказателства или легални възможности да се докаже), че компанията-собственик е честна към вас и уважава сигурността ви; въпреки изначалния ѝ отказ да бъде честна към вас, като скрива от полезрението ви изходния код, забранява ви да знаете и ви лишава от свободата да променяте и адаптирате софтуера към ваши специфични нужди.

Безусловно свободни са преди всичко базираните на **GNU/Linux-libre** операционни системи, които не притежават нито един несвободен (и скрит от погледа и контрола на потребителите) компонент.

https://en.wikipedia.org/wiki/Linux-libre

Такива са например **Parabola**, **Trisquel**, **Dragora**, **PureOS** (за компютри) или '**Replicant**' (за телефони – с уговорката, че към настоящия момент все още не са решени всичките технически проблеми за постигането на наистина напълно свободен телефон).

https://www.parabola.nu/
https://trisquel.info/
https://www.dragora.org/
https://pureos.net/
https://www.replicant.us/

Към настоящия момент на етап **β**-тестване (основните функции работят, но несигурно и се отстраняват съществени слабости) е проектът за разработване на Свободната операционна система **Heads**, която е ориентирана специално към обезпечаване на информационната сигурност. Вероятно след излизането на версия **1.0** (първата стабилно и надеждно работеща версия) ще можем да препоръчаме **Heads** като подходяща Свободна операционна система за целите на нашето изложение. Към настоящия момент обаче това все още не е факт и се надяваме в следващо време да бъде променено. Стабилна (далеч отвъд версия **1.0**) е операционната система **TAILS**, която също е ориентирана към обезпечаване на информационната сигурност, но за съжаление не е изцяло и безусловно свободна – поради което няма как да ѝ се доверим напълно и да ви препоръчваме да я ползвате.

https://heads.dyne.org/
https://tails.boum.org/

Като всеки Свободен софтуер, Свободните операционни системи могат да бъдат свалени, инсталирани и ползвани от вас свободно, без да е необходимо да искате разрешение или да плащате за това. Ако имате нужда от специфични настройки или от допълнително адаптиране на софтуера, можете да проучвате безпрепятствено неговия изходен код и да го промените сами или да поръчате да бъде променен за вас от избрани от вас специалисти – без това да нарушава нечии патентни претенции или лицензионни режими. В това се изразява свободата при Свободния софтуер. Следователно, този вид софтуер може да работи точно така, както желаете вие, а не както е определил някой друг. И ако се съмнявате в начина, по който работи софтуерът, можете свободно да направите своя независима проверка на

изходния код и да нанесете корекции (или да поръчате това да бъде направено за вас от специалисти, на които имате доверие). И не на последно място – можете да разпространявате свободно направените от вас подобрения – което позволява на идеите да се надграждат и допълват свободно и децентрализирано (т.е. не зависят от конкретна компания-собственик, чието евентуално компрометиране да води задължително до компрометирането и на вашия софтуер). Последното е ключово за информационната сигурност.

Без да предопределяме избора ви, по-долу ще се съсредоточим на работа със Свободната операционна система **Parabola**. Спираме се именно на нея, защото тя е добре поддържана и предпочитана от по-опитните потребители, макар да изисква малко повече познания, за да бъде инсталирана и ползвана. Друга предпочитана операционна система е **Trisquel** (тъй като е съобразена специално със затрудненията на новите потребители на **GNU/Linux**, но не се поддържа толкова активно и не може да гарантира в достатъчна степен търсената от нас стабилност и надеждност. Принципът на работа при всичките Свободни операционни системи е сходен. Независимо от това на коя от тях ще изберете да се доверите, не представлява проблем на база дадените от нас напътствия за **Parabola** и огромната по обем свободно достъпна техническа документация да се ориентирате и да се справите. Ако се нуждаете от допълнително съдействие, можете да се свържете с нас или с ваш доверен съветник за по-нататъшни указания.

2. Най-общо за командния ред, терминала и системата като цяло

В свободна софтуерна среда е застъпено широко използването на командния ред (**Command Line**). Преди няколко десетилетия това е бил единственият механизъм за взаимодействие между човек и компютър. В действителност графичната среда е твърде ограничена откъм своите възможности в сравнение с командния ред и в крайна сметка не се препоръчва при сериозно ползване на компютър. За съжаление при широкото проникване на компютрите сред хората през миналия век, **Microsoft** комерсиализирали графичния интерфейс като „по-удобен“ за потребителите-неспециалисти и така на практика осакатили работата с компютър, която работа в същността си остава въвеждане на директни команди към компютъра във вид на прост текст. Компютърът изпълнява командите и когато е необходимо, извежда резултати и/или указания за следващи възможни действия (отново във вид на прост текст). Предимството на този подход е, че между вас и компютъра не посредничи никакъв графичен интерфейс (чиято сигурност също да трябва да се гарантира по някакъв начин; който интерфейс (както ще ви убедим) е много по-ограничен и неудобен от командния ред; и който интерфейс в някои случаи (например при технически срив) може да се окаже недостъпен и това да наложи поемане на контрола над системата именно от командния ред).

https://www.linuxcommand.org/lc3_learning_the_shell.php
https://www.tldp.org/LDP/GNU-Linux-Tools-Summary/html/GNU-Linux-Tools-Summary.html

Тъй като Свободните операционни системи третират потребителя като свободен човек (който трябва да има действителен и неограничен контрол спрямо своята система, вместо да стои „затворен“ в лимитираните възможности на потребителския интерфейс), базираният на **GNU/Linux-libre** софтуер недвусмислено насърчава потребителите да ползват възможностите на командния ред. Това позволява на потребителя да се държи със системата като програмист (вместо като незнаещ ползвател, за когото са достъпни единствено бутоните от графичното меню). Освен технологичната гаранция, че Свободният софтуер работи наистина по официално обявения начин и възможността да променяте начина му на работа, следващото ключово достойнство на Свободния софтуер, което го прави по-добър от други софтуерни алтернативи, е именно широкото използване на командния ред и произтичащата от това свобода. Без да навлизаме в подробности – само ще отбележим, че докато командният ред на пръв поглед изглежда по-сложен от графичния интерфейс, той в действителност позволява осъществяването на доста сложни процеси само с въвеждането на няколко символа и това е още една причина да го предпочитаме.

Командния ред можете да ползвате посредством т.нар. виртуален терминал (**Terminal**). Терминалът е базово приложение във всички Свободни операционни системи. Той представлява екран с възможност за въвеждане на прости текстови команди, който можете да отворите от графичния интерфейс или с клавишни комбинации. От графичния интерфейс можете да изберете менюто **Applications** и оттам да потърсите в **Accessories**, **System Tools** или **Utilities** (в зависимост от особеностите на вашето графично меню). Можете да отворите терминала и просто като натиснете едновременно **[Ctrl]**+**[Alt]** и заедно с тях – някой от бутоните **[F1]**, **[F2]**, **[F3]**, **[F4]**, **[F5]**

или **[F6]**. С всяка от предложените комбинации ще се отвори (на целия екран!) един от възможните терминални екрани, които ви позволяват да осъществявате паралелно различни процеси в системата (като се прехвърляте със съответните клавишни комбинации). За да се върнете отново в графичния интерес, трябва да натиснете **[Ctrl]**+**[Alt]** и заедно с тях – **[F7]** (или в някои Свободни операционни системи – **[F2]**).

В интерес на вашата информационна сигурност е препоръчително да подавате своите инструкции към системата по възможно най-простия и директен начин – чрез командния ред. Поради това ви насърчаваме още сега да отворите терминала от графичния интерфейс или чрез предложените клавишни комбинации и да усвоите някои от основните принципи за работа с командния ред. Ще се убедите, че това е изключително мощен метод за работа с компютър, в сравнение с който графичният интерфейс (макар да е привидно по-удобен) ви лишава от много възможности.

В терминала можете да въвеждате прости текстови команди към системата и тя да ги изпълнява, като извежда резултати и указания. За да изпълни каквато и да било ваша команда обаче, системата трябва да съобрази дали имате изискуемите права за изпълнение на заявената команда. Поради тази причина (и за ваше улеснение) в терминала винаги се извежда обозначение за това как ви възприема системата и съответно с какви права разполагате.

Ако отворите терминала с някоя от предложените по-горе клавишни комбинации (например **[Ctrl]**+**[Alt]**+**[F6]**), в горния ляв ъгъл ще бъде изведено обозначението '**Компютър login:**' (където под '**Компютър**' разбираме наименованието на вашето устройство, а '**login:**' е указание да въведете наименованието на потребителя, от чието име желаете да получите достъп). Когато въведете наименованието на съществуващ във вашата система потребителски акаунт и натиснете **[Enter]**, системата ще изведе указание '**Password:**' в очакване да въведете вашата парола.

Имайте предвид, че докато въвеждате пароли в терминала, от съображения за сигурност няма да се извеждат никакви индикации за броя въведени символи (например '****************' или '**••••••••••••**'). Това се прави с цел ограничаване на риска да бъде установена (приблизителната) дължината на вашата парола от трета страна (която например наднича в екрана ви, докато въвеждате). Ако бъде установен приблизителният брой символи на паролата (както ще проследим в Част **V**), нейното разбиване чрез прилагането на **bruteforce** методи (последователно автоматизирано опитване на предполагаеми комбинации, докато паролата накрая бъде уцелена) ще стане много лесно. Ето защо приблизителният брой символи в паролата не трябва да се разкрива пред трети страни.

След като въведете вашата парола (и отново натиснете **[Enter]**), системата (при правилна парола) ще изведе обозначение от вида на '**[Потребител@Компютър ~]$**' (където под '**Потребител**' разбираме наименованието на потребителския акаунт, от чието име сте получили достъп; под '**Компютър**' разбираме наименованието, с което е озаглавено вашето устройство; символът '**~**' обозначава директорията на вашия потребител, където се намирате в момента; и символът '**$**' обозначава това, че системата ви разпознава като обикновен потребител, без администраторски права). Това означава, че за вас са в сила определени ограничения по отношение изменянето на системата или достъпването и променянето на съдържание, което е асоциирано с други потребители (ако в системата има създадени акаунти на такива). Ако вместо наименованието на потребител въведете '**root**' (от англоезичната дума за „корен"; коренов, основен потребител; системен администратор), системата отново ще изведе указание '**Password**:' в очакване този път да въведете парола за достъп като администратор на системата. Когато въведете паролата и натиснете **[Enter]**, системата ще изведе обозначение '**[root@Компютър ~]#**' (където под '**root**' разбираме администратора на вашата система; а под '**Потребител**' – наименованието на текущия потребител, от чиято сесия сте осъществили административния достъп; и символът '**#**' обозначава това, че системата вече ви възприема като неин администратор).

В случай, че отворите терминала от менютата в графичния интерфейс, системата директно ще изведе обозначение от вида на '**[Потребител@Компютър ~]$**' (тъй като извършените действия от графичния интерфейс по подразбиране се интерпретират като такива на потребителя, от чиято сесия достъпвате устройството. При това положение можете да получите администраторски права по два начина.

В някои операционни системи (като например **Trisquel**) по подразбиране е инсталирана програмата **Sudo**, която ви позволява да изпълнявате действия с необходимост от администраторски права, като в сесията на обикновения потребител (обозначавана със символа '**$**') въведете такава команда:

```
$ sudo Команда
```

(където символът '**$**' не е част от командата и не трябва да го въвеждате; а под '**Команда**' разбираме това указание към системата, което желаете да бъде изпълнено и за което са необходими администраторски права). След като въведете горната команда и натиснете **[Enter]**, системата ще изведе указание '**Password:**' в очакване да въведете вашата парола като обикновен потребител. Когато въведете паролата и натиснете **[Enter]**, системата ще изпълни въведената команда (включително и ако за нейното изпълнение се изискват администраторски права – независимо от факта, че сте в сесията на обикновен потребител).

В операционни системи като **Parabola** програмата **Sudo** не е инсталирана по подразбиране. Ако желаете, можете да я инсталирате допълнително, но ако все още не сте сторили това, за изпълнението на действия с необходимост от администраторски права трябва първо да стартирате администраторска сесия. За тази цел в сесията на обикновения потребител (обозначавана със символа '**$**') можете да въведете тази команда:

```
$ su
```

(където символът '**$**' не е част от командата и не трябва да го въвеждате; а под '**su**' разбираме указание към системата, че желаете временно да смените текущия потребител; **Switch User** и да бъдете възприети като '**root**' потребителя с администраторски права. След като въведете горната команда и натиснете **[Enter]**, системата ще изведе указание '**Password:**' в очакване да въведете вашата парола като администратор. Когато направите това и натиснете **[Enter]**, системата ще изведе обозначение от вида на '**[root@Компютър Директория]#**' (където, както посочихме по-горе, '**#**' се явява обозначение за това, че системата вече ви възприема като администратор).

Бъдете особено внимателни, докато работите в режим с администраторски права. Това ви позволява да въздействате с пълна власт над системата, включително да я увредите. Не оставяйте компютъра без надзор, докато имате отворен терминал с администраторски права, дори и „само за малко“! За да се върнете в режим на обикновен потребител (на когото не се разрешава достъп до по-сериозните функции), натиснете **[Ctrl]**+**[D]** едновременно (при което отново ще бъде изведено обозначение от типа на '**[Потребител@Компютър ~]$**') или просто затворете терминала от графичния интерфейс. Препоръчваме да влизате в режим с администраторски права само тогава, когато това е необходимо и веднага след приключване на съответните действия да излизате от този режим.

Като предимство на разграничаването между сесиите на обикновения потребител (**$**) и тези на администратора (**#**) се изтъква обстоятелството, че при този режим на работа се налага да въведете администраторска парола само веднъж (и в резултат от това се открива сесия на администратора, в която можете да въвеждате неограничен брой команди, независимо от това какви права се изискват за тяхното изпълнение). Както ще проследим по-надолу обаче, въвеждането на паролата е един от най-критичните моменти за нейната сигурност и възможността това да се извърши само веднъж, е безспорно предимство. Друго предимство при този режим на работа е, че при него системата разграничава две отделни пароли за достъп – на обикновения потребител и на администратора. Това ви позволява да влизате в системата без да е необходимо всеки път да въвеждате паролата, с която би се разрешило изпълнението на чувствителни команди. Недостатък обаче е това, че веднъж стартирана, сесията с администраторски достъп остава отворена, докато не бъде прекратена (било с изрична команда, било със затваряне на терминала или с изключване на устройството). С други думи – ако по някаква причина изоставите компютъра (или внезапно бъдете отстранени насилствено от него), без да съумеете преди това да прекратите сесията, тя може да попадне под чужд контрол. Тук е основното предимство на програмата **Sudo** (и нейната едноименна команда). При този режим на работа изобщо не съществува администраторска сесия, а паролата се въвежда инцидентно всеки път, когато указвате да се изпълни чувствителна команда. След това можете да продължите да въвеждате команди с администраторски права още няколко минути, но след пауза от няколко минути сесията се прекратява от само себе си. Друго предимство на програмата **Sudo** е, че при нея избягвате поначало стартирането на сесия с администраторски права и от там – инцидентното стартиране на програми или извършването на действия като администратор, вместо като обикновен

потребител. Недостатък на **Sudo** е, че позволява осъществяването на административен достъп с парола, която се ползва от потребителя без администраторски права – т.е. всеки път, когато стартирате системата. А както посочихме, въвеждането на паролата е най-критичният момент от нейната сигурност. Очевидно всеки от двата режима на работа има своите предимства и недостатъци. Ето защо трябва да се съобразите преди всичко с вашия начин на работа и със спецификите на заобикалящата ви среда, за да изберете правилния за вас подход.

По-нататък в изложението ще се натъкнем на т.нар. системни конфигурационни файлове. В Свободните операционни системи тези файлове съдържат текстови настройки, които се използват за управление на системата. Понякога обаче в системните файлове се налага да бъде оставена ориентираща бележка за програмистите или някои настройки да бъдат изключени. Тогава преди началото на съответния ред или текстови абзац (бележка или изключена настройка) се поставя символът '**#**', който в този случай указва на компютъра да не интерпретира следващия текст като команда за изпълнение, а като текст, предназначен за хората; т.нар. „коментар“. На този етап отбелязваме това второ значение на символа '**#**' само за пълнота; в командния ред символът '**#**' обозначава само и единствено сесия с администраторски права, за разлика от сесията без такива права, обозначавана със символ '**$**'.

Терминалът ви позволява да се премествате навсякъде в системата, като преглеждате намиращото се на съответните места съдържание, стартирате програми, въвеждате настройки, копирате, местите, променяте и изтривате файлове или текстово съдържание вътре в тях (стига да имате права за това). За нуждите на нашето изложение обаче ще се ограничим само до по-простите функции, свързани с информационната сигурност – като започнем от най-базовите команди.

Така например, терминалът ви позволява да се премествате в системата – например можете да отидете на вашия Работен плот (**Desktop**):

```
$ cd Desktop
```

Понякога, за да се преместите в системата, може да ви се наложи да въведете и т.нар. „пълен адрес“ на избраното място – в примера с Работния плот трябва да въведете командата така:

```
$ cd /home/Потребител/Desktop
```

(където под '**home**' разбираме директорията с файловете и настройките на отделните потребители в системата; под '**Потребител**' – наименованието на текущия потребител, с което е озаглавена неговата лична директория; и под '**Desktop**' – директорията с Работния плот на този потребител, която се намира в неговата лична директория). Ако има проблем, системата ще изведе резултат за липса на такъв файл или директория (**`bash: cd: Desktop: No such file or directory`**). Ако всичко е наред, системата ще изведе резултат '**`[Потребител@Компютър Desktop]$`**' в потвърждение на това, че вече се намирате на Работния плот (като се запазва символът '**$**' за това, че системата ви възприема като потребител без администраторски права).

Тук за пълнота трябва да отбележим, че **GNU/Linux** в основата си представлява многопотребителска сървърна система (където различни потребители могат да работят паралелно и всеки от тях може да достъпва своето съдържание, но не и съдържанието на други потребители). Едва на по-късен етап е създаден графичният интерфейс и многопотребителската сървърна система е започнала да функционира и като графична Desktop-среда за обикновени потребители (като каквато е замислена още от самото си начало например концепцията за персоналния компютър (**Personal Computer**) на **IBM**, възприета по-късно и от **Microsoft**).

Всеки потребител с акаунт в **GNU/Linux** притежава своя лична директория, разположена в директорията **`/home`** и озаглавена със собственото му наименование. Личната директория на потребителя изпълнява функции като тези на **C:\Users\Потребител** при **Windows** например (там потребителят може да съхранява своето лично съдържание и персонални настройки). В директорията на потребителя обикновено съществува и директория **`Desktop`** – това е Работният плот на съответния потребител (който е различен от Работните плотове на други потребители в системата). В тази директория се съхраняват всичките файлове, които съответният потребител държи графично на своя Работен плот.

Можете да се ориентирате за разположението и функциите на отделните директории в една типична **GNU/Linux** системата, като се запознаете с цялостната файлова структура на системата (която започва с Кореновата директория (обозначава се със символа '**/**') и се разгръща по следния начин (с някои незначителни вариации в отделните системи):

```
/bin            (основни програми и приложения на системата)
/boot           (ядро на системата и настройки на стартираща програма)
/dev            (закачени към системата компоненти и устройства)
/etc            (текстови файлове с базовите настройки на системата)
/home           (файлове и настройки на отделните потребители)
   ├── /Текущ_потребител  (терминалът се отваря тук по подразбиране)
   │       ├── /Desktop  (тук е Работният плот на потребителя)
   │       └── /Още_директории_на_текущия_потребител...
   └── /Друг_потребител  (ако има такъв с акаунт в системата)
           ├── /Desktop  (Работен плот на другия потребител)
           └── /Още_директории_на_другия_потребител...
/lib            (основни програмни библиотеки, ползвани от системата)
/lost+found     (аварийно спасени файлове – например при токов удар)
/mnt            (закачени към системата дялове от твърдия диск)
/opt            (допълнителни потребителски приложения и настройки)
/proc           (описания на процесите и състоянията в системата)
/root           (файлове и настройки на администратора на системата)
/run            (ресурси, които взаимодействат активно със системата)
   └── /media   (закачени към системата носители с външна памет)
         ├── /Текущ_потребител  (директория на съответния потребител)
         │     ├── /USB_памет  (появява се при закачане към системата)
         │     ├── /CD/DVD  (появява се при закачане към системата)
         │     └── /Още_устройства_с_външна_памет...
         └── /Друг_потребител  (ако има такъв с акаунт в системата)
               ├── /USB_памет  (появява се при закачане към системата)
               ├── /CD/DVD  (появява се при закачане към системата)
               └── /Още_устройства_с_външна_памет...
/sbin           (специални приложения на администратора на системата)
/srv            (специфични данни, относими към настоящата система)
/sys            (специални файлове за управление на системата)
/tmp            (временни файлове на потребителските приложения)
/usr            (инсталирани в системата допълнителни приложения)
/var            (временни файлове и текущо състояние на системата)
```

Принципна файлова структура на една GNU/Linux система

Важно е да отбележим, че при някои системи директорията '**media**' (където се позиционират закачените от съответния потребител устройства с памет – външни твърди дискове, **SD**-карти, **USB**-памет и т.н.), е възможно да бъде разположена и направо в Кореновата директория '**/**', а не както по-горе – в директорията '**/run**'.

Когато отворите терминала, ще се озовете по подразбиране в директорията на текущия потребител '**/home/Текущ_потребител**' (а ако сте осъществили административен достъп, ще се озовете в директорията на администратора '**/root**').

Можете да премествате позицията си във всяка от горните директории и намиращите се в тях под-директории (стига да имате право на достъп до тях) чрез командата '**cd**' (**Change Directory**), последвана от адреса на съответната директория (или оставащата част от адреса спрямо настоящата ви позиция, ако искате да се преместите в поддиректория на тази, в която се намирате към момента).

Независимо къде сте се намирали преди това, можете да се преместите в директорията на текущия потребител чрез тази команда:

```
$ cd
```

... или съответно

```
# cd
```

Можете по всяко време да проверявате къде в системата се намирате:

```
$ pwd
```

Системата ще изведе пълният адрес на вашето местоположение.

Можете да се преместват в различните директории на системата:

```
$ cd /Адрес
```

(където под '**/Адрес**' разбираме пълната последователност от директории, докато стигнете до тази, в която искате да се преместите). Адресът в този случай започва от Кореновата директория (като въведете всички останали директории до желаната от вас – това е т.нар. „абсолютен“ адрес, който не зависи от вашето настоящо местоположение в системата.

Можете да ползвате и съкратени адреси (т.нар. „релативни“ адреси), които обаче зависят от вашето настоящо местоположение. Така например, като съкратен адрес можете да въведете само наименованието на под-директория, намираща се във вашата настояща директория (вместо целия „път“ от Кореновата директория до нея):

```
$ cd Директория
```

(където под '**Директория**' разбираме наименованието на съответната под-директория). Системата ще ви премести в тази под-директория. Можете по този начин да се преместите и през няколко директории наведнъж:

```
$ cd Директория/Следваща_директория
```

Разбира се, ако в момента се намирате в Кореновата директория, всяка друга ще се окаже в нея и при това положение „абсолютният“ и „релативният“ адрес ще съвпаднат.

При всички случаи трябва да изписвате точно наименованията на директориите (с правилни малки и големи букви) и да разделяте със символ '**/**' имената на директориите, като в началото на адреса ще поставите символ '**/**', само ако адресът започва от Кореновата директория.

Ако имате директория с интервали в наименованието, трябва да имате предвид, че те ще бъдат интерпретирани от командния ред като край на наименованието и втората част от това наименование ще се интерпретира като следващо наименование. За да избегнете тази грешка, трябва да обозначите интервалите в наименованието със символ '****' и интервал след него – например при '**Наименование на директория**' трябва да изпишете '**Наименование\ на\ директория**' в командния ред. Независимо от тази възможност обаче, не препоръчваме да поставяте в наименованията на директориите и файловете каквито и да било интервали, специални символи (освен '**-**' и '**_**') или букви, различни от обикновената латинска **US**-стандартизация). Символите, различни препоръчаните, могат да бъдат интерпретирани от системата по особен начин, който да доведе до технически грешки или до повреждане на информацията, записана с подобни наименования.

Със символа '****' обозначаваме и такива команди, които са твърде дълги и не се побират на един ред на екрана (или на страницата). Такива команди би следвало да се въвеждат в терминала на един единствен ред (без пренасяне и без

натискане на **[Enter]** преди въвеждането на цялата команда). Поради своята голяма дължина обаче, командите се изобразяват на два или повече последователни реда (въпреки, че от гледна точка на командния ред са въведени на един единствен по-дълъг ред). Когато липсата на достатъчно място не ни позволява да представим съответната команда на един ред, можем със символа '****' да обозначим това, че следващият текст (макар да е поместен на долния ред) е част от горния текст и би следвало да се изпише без пренасяне на нов ред. Ако заедно с това има и интервал, същият следва да се постави преди символа '****', по подобен начин:

```
# pacman -S linux-libre-lts linux-libre-firmware mkinitcpio \
bash-completion networkmanager nano lvm2 cryptsetup
```

(където имаме една дълга команда, изписвана при въвеждане в терминала на един ред, но поради липсата на достатъчно място по-горе сме изобразили на **2** последователни реда).

Можете да се преместите в директорията, съдържаща вашата настояща директория:

```
$ cd ..
```

Ако повторите горната команда достатъчно пъти, накрая ще се озовете в Кореновата директория, която съдържа всичките останали директории в система.

Можете да се върнете в предходната директория, от която сте се преместили в настоящата:

```
$ cd -
```

Можете да отидете в директорията на съответен потребител:

```
$ cd ~Потребител
```

(където под '**Потребител**' разбираме наименованието на потребителя, в чиято директория желаете да се преместите, а символът '**~**' обозначава останалата част от адреса на съответната директория). Системата ще ви премести там, без значение къде се намирате в момента.

Следователно, можете да се преместите в директорията на администратора (за което е необходимо да сте стартирали сесия с администраторски права) чрез тази команда:

```
# cd ~root
```

Можете да се преместите в директорията на текущия потребител и без да въвеждате неговото наименование:

```
$ cd ~
```

или съответно:

```
# cd ~
```

Терминалът ви позволява да преглеждате заглавията на файловете и под-директориите, които се намират в настоящата директория:

```
$ ls
```

Ако желаете да видите и „скритите“ файлове и под-директории (тези, които обикновено не се обработват пряко от потребителя и поради това системата ги „скрива“ от него, за да не бъдат засегнати поради грешка), можете да направите това чрез тази команда:

```
$ ls -a
```

Системата този път ще изведе наименованията и на „скритите“ директории и файлове (които в **GNU/Linux** се обозначават със символ '**.**' в началото на наименованията).

Можете да изискате допълнителни подробности за директориите и файловете:

```
$ ls -l
```

Системата този път ще изведе обобщени подробности за директориите и файловете, които се намират в настоящата директория, от типа на следните:

```
-rw-r--r-- 1 Потребител Група 4836707 Jan 12 12:48 Document.pdf
drwxr-xr-x 4 Потребител Група    4096 Dec 30 03:50 Директория
-rw-r--r-- 1 Потребител Група  140590 Sep 16 22:08 Изображение.png
-rw-r--r-- 1 Потребител Група  584204 Jan 16 12:30 Текстови_файл.txt
lrwxrwxrwx 1 root       root       11 Jan  6 09:08 Link -> ../home
```

(където първи символ '**-**' обозначава файл, първи символ '**d**' – директория (от англоезичното '**D**[**irectory**]'), а първи символ '**l**' – препратка (от англоезичното '**L**[**ink**]') към друго място в системата; следващите **9** символа (примерно '**rwxr-xr-x**') обозначават правата за достъп до съответната директория или файл (по-долу в изложението ще се спрем подробно на този въпрос); следващото след интервала число (примерно '**1**' или '**4**') обозначава броя препратки към дадена директория или файл (една директория или файл може да се достъпва едновременно от няколко различни директории); следващите две колони обозначават наименованията на потребителите и групите, притежаващи съответните директории и файлове (в горния пример препратката '**Link**' от последния ред принадлежи на администратора); числото от следващата колона обозначава размера на директорията или файла в байтове (**B**); следват колони за датата и часа, когато съответните директории или файлове са променяни последно; и в последната колона са наименованията на съответните директории и файлове, като за препратките има и обозначение на мястото, към което водят).

Често към една команда можете да добавяте повече от едно действия. Така например, можете да укажете да се изведе едновременно съдържанието на няколко директории (с подробности, включително за „скритите“ директории и файлове):

```
$ ls -la /home/Потребител /run/media/Устройство
```

(където под '**Потребител**' разбираме наименованието на потребителя, чиято лична директория ви интересува в дадения пример; а под '**Устройство**' – наименованието на устройство (вероятно външен носител с памет), което сте закачили към системата и желаете да прегледате). При това системата ще изведе наименованията на директориите и файловете (включително „скритите“ такива), които се намират и на двете места, с подробности за тях. Когато свикнете да комбинирате по този начин команди, ще можете да указвате извършването на сложни действия в системата с един кратък ред символи, вместо с многократни и продължителни цъкания на мишката из графичния интерфейс; или да съчетаете няколко последователни команди:

```
$ ls -la /Директория /run/media/Устройство > Текстови_файл
```

(където под '**Файл**' разбираме текстови файл, в който желаете да бъде изведено съдържанието на няколкото директории от горния пример).

Освен че можете да се премествате из системата и да преглеждате намиращото се на съответните места съдържание, с терминала можете да стартирате програми и да отваряте файлове с програмите. Така например можете да отворите браузъра **Firefox** (стига да имате инсталиран този браузър във вашата система):

```
$ firefox
```

или дори да отворите конкретен web-сайт с този браузър:

```
$ firefox Адрес_на_web_сайт
```

(където под '**Адрес_на_web_сайт**' разбираме web-адреса на съответния web-сайт, например '**fsf.org**'). Същата команда може да бъде изпълнена и с всеки друг браузър, който е инсталиран във вашата система.

Можете например да отворите текстовия файл '**Текстови_файл.txt**' с популярния свободен текстови редактор **Gedit** (стига да имате инсталирана тази програма на вашата система):

```
$ gedit Текстови_файл.txt
```

За да бъде изпълнена командата, трябва да сте посочили точно наименованието на интересуващия ви файл – с точно посочване на главни и малки букви в наименованието и включено неговото разширение, ако има такова). Ако файлът не се намира там, където се намирате вие в момента, трябва преди неговото наименование да въведете и неговия „абсолютен“ или „релативен“ адрес:

```
$ gedit Адрес/Текстови_файл.txt
```

(където под '**Адрес**' разбираме точното местоположение на файла в системата). Ако вие се намирате например в директорията на потребителя без администраторски права, а файлът се намира на Работния плот, горната команда ще придобие подобен вид:

```
$ gedit /home/Потребител/Desktop/Текстови_файл.txt
```

или:

```
$ gedit Desktop/Текстови_файл.txt
```

Можете да стартирате и „портативна“ (**Portable**) програма (която не е инсталирана в системата, а се намира в някоя от директориите като независим файл):

```
$ Адрес/Програма
```

Ако програмата се намира в настоящата ви директория, можете да я стартирате така:

```
$ ./Програма
```

(където символът '**.**' указва, че командата се отнася за настоящата директория).

Ако нямате инсталирана във вашата система програмата, която желаете да стартирате (с която желаете да отворите интересуващия ви файл), можете да я свалите от хранилищата със Свободен софтуер и да я инсталирате чрез тази команда:

– ако работите с **Parabola** или друга **Arch**-базирана **GNU/Linux** система:

```
# pacman -S Програма
```

– ако работите с **Trisquel** или друга **Debian**-базирана **GNU/Linux** система:

```
# apt-get install Програма
```

Респективно, ако ползвате програмата **Sudo**, командата ще има подобен вид:

```
$ sudo pacman -S Програма
```

... или съответно:

```
$ sudo apt-get install Програма
```

Системата ще се свърже с хранилищата със свободен софтуер (в дадените примери – хранилищата на операционната система **Parabola** или тези на операционната система **Trisquel**), ще свали от копия на интересуващите ви програми, ще им направи криптографска проверка за автентичност и интегритет и след като се убеди, че всичко е наред, ще ги инсталира на компютъра. Преди да стори това обаче, системата ще изведе указание за това какви софтуерни пакети предстои да бъдат инсталирани и ще изиска от вас да потвърдите дали да продължи с инсталацията (**`Proceed with installation? [Y/n]`**), при което от вас се очаква да въведете **`[y]`** (от англоезичното '**Y**[**es**]') и да натиснете **`[Enter]`**, за да се изпълни процесът. Можете да продължите да работите със системата, а когато инсталационният процес приключи, ще можете да стартирате и новата програма.

В случай, че желаете да изтриете вече инсталирана програма от вашата система, можете да сторите това чрез тази команда:

– ако работите с **Parabola** или друга **Arch**-базирана **GNU/Linux** система:

```
# pacman -R Програма
```

– ако работите с **Trisquel** или друга **Debian**-базирана **GNU/Linux** система:

```
# apt-get remove Програма
```

Респективно, ако ползвате програмата **Sudo**, командата ще има подобен вид:

```
$ sudo pacman -R Програма
```

... или съответно:

```
$ sudo apt-get remove Програма
```

Някои програми позволяват да отваряте с тях включително файлове, които не съществуват. В такъв случай съответната програма ще зареди празен файл, в очакване да въведете в него необходимото съдържание и да го запишете. Можете да постъпите така например с програмата **Gedit**):

```
$ gedit Несъществуващ_файл.txt
```

Можете да създадете нов празен файл, дори без да ползвате конкретна програма за това и без да отваряте този файл по какъвто и да било начин:

```
$ touch Празен_файл
```

(където под '**`Празен_файл`**' разбираме точното наименование на празния текстови файла, като можете да добавите и разширение, ако искате да има такова (например '**`.txt`**'; който файл ще бъде създаден в текущата директория, където се намирате в момента – освен ако не укажете друг адрес преди наименованието на файла).

Съществува възможност да създадете обикновен текстови файл, като наберете директно съдържанието му във вид на прост текст в терминала и укажете да бъде създаден с това съдържание:

$ cat > Текстови_файл

(където под '**Текстови_файл**' разбираме наименованието на файла, който искате да създадете). Ако желаете файлът да бъде на място, различно от текущата директория, преди наименованието му трябва да укажете и съответния адрес. Системата ще влезе в режим на очакване, за да въведете в терминала желания текст. Имайте предвид, че след като веднъж натиснете **[Enter]** за преминаване към следващ абзац, повече не можете да се връщате на предходните абзаци. Когато завършите въвеждането, трябва да натиснете едновременно **[Ctrl]**+**[d]** и файлът с въведеното текстово съдържание ще бъде създаден на указаното място. Ако преждевременно сте натиснали **[Enter]** и желаете да редактирате нещо в предходен абзац, няма пречка да отворите файла след създаването му в текстови редактор и да нанесете желаните промени.

В случай, че файл с посоченото от вас наименование и местоположение вече съществува, командата '**cat > Текстови_файл**' (или респективно '**cat > Адрес/Текстови_файл**') ще изтрие стария файл със съдържанието му и ще създадете нов файл със същото наименование и местонахождение, но вече с новото въведено от вас съдържание. Бъдете особено внимателни, когато работите с файлове с важно съдържание (в т.ч. – системни файлове от значение за функционирането на системата).

Посредством символа '**>**' можете да запазвате изведените от съответна команда резултати към файл. Така например можете да препратите резултата от командата '**ls**' за преглеждане заглавията на директориите и файловете в текущата директория към текстови файл с подобна команда (за която споменахме по-горе):

$ ls > Текстови_файл

... при което посоченият списък ще бъде отпечатан в указания текстови файл (или указаният файл със списък ще бъде създаден, ако до момента не е съществувал файл с такова наименование). Няма пречка да сторите това и за места в системата, различни от вашата текуща директория:

$ ls /Адрес/Файл_или_директория > /Адрес/Текстови_файл

Ако желаете да запазите съществуващото във файла съдържание и само да допълните ново съдържание под вече съществуващото, можете да направите това така:

$ cat >> Текстови_файл

(където под '**Текстови_файл**' отново разбираме наименованието на файла, който този път само ще допълвате). С двойната стрелка в горната команда указвате системата да запази заварения файл с неговото съдържание и да добави най-отдолу нововъведеното съдържание. Това е удобно, когато например желаете периодично да добавяте нови записи, без обаче да засягате направените по-рано.

От терминала можете да изтриете файл:

$ rm Файл

В зависимост от вашата система файлът ще бъде направо изтрит или преди това ще бъде изискано потвърждение (**[Y/n]**).

Можете да изтриете и празна директория:

$ rmdir Празна_директория

От терминала можете да изтриете и директория, която не е празна (ведно с нейното съдържание):

```
$ rm -r Директория_със_съдържание
```

Можете да изтриете цялото съдържание в текущата директория (без „скритите“ файлове в тази директория, обозначени със символ '**.**' в началото на наименованията им):

```
$ rm -r *
```

Можете да изтриете всичко (в т.ч. и „скритите“ файлове) в текущата директория:

```
$ rm -r .[^.]* *
```

С горните команди можете да изтриете само съдържание, което е асоциирано с вашия потребител. Ако желаете да изтриете съдържание от съществено значение за системата или такова, което е асоциирано с друг потребител, исканото от вас изтриване ще бъде отказано – освен ако не заявите администраторски права за изпълнението му или ако на вашия текущ потребител не бъдат предоставени съответните права на достъп по отношение на тези директории или файлове (по-долу ще се спрем подробно на този въпрос).

Тъй като командите за изтриване, включващи символ '*****', обозначават да се изтрие „всичко“, за да не допуснете нежелана загуба на съдържание, можете да въведете командата '**echo**' (при което горните команди ще придобият вид '**echo rm -r ***', съответно '**echo rm -r .[^.]* ***'. При това положение триене няма да бъде осъществено, но системата ще изведе резултат – наименованията на директориите и файловете, които биха били изтрити, ако се изпълни съответната команда за изтриване без предшестващата я команда '**echo**'. Препоръчва се да правите такава проверка, преди да указвате изтриване на „всичко“ – защото с подобна команда няма пречка например да изтриете цялата система, ако се намирате в Кореновата директория и действате като администратор.

Терминалът ви позволява да преглеждате и редактирате съдържанието на текстови файлове във вид на прост текст, без да е необходимо да отваряте файловете в отделен текстови редактор. Тъй като под формата на обикновени текстови файлове е изградена цялата система от настройки на **GNU/Linux-libre** (споменатите по-горе системни и конфигурационни файлове), това позволява от командния ред да преглеждате съответните файлове и да нанасяте необходимите ви настройки, като на практика правите всичко, което системата изобщо е способна да ви предостави, в текстови режим. Никой графичен интерфейс (въпреки привидното му удобство за потребителите) не може да предостави подобна мощност на въздействие над системата.

Да прегледате от терминала съдържанието на текстови файл (в т.ч. на системен файл) можете чрез тази команда:

```
$ less Текстови_файл
```

В терминала ще бъде изведено съдържанието на файла във вид на прост текст (който можете да „прелиствате“ надолу-нагоре със стрелките и бутоните **[HOME]**, **[PGUP]**, **[PGDN]** и **[END]**, ако текстът е по-дълъг). Можете да търсите из преглеждания в терминала текст чрез тази команда:

```
/Търсене
```

(където под '**Търсене**' разбираме интересуващата ви дума, израз или части от тях). Системата ще обозначи с цветови индикатор местата в текста, където въведеното '**Търсене**' се намира; или ще изведе указание, че търсенето не е намерено (**Pattern not found**). Ако вашето '**Търсене**' се повтаря на няколко места в текста и искате да прегледате всяко от тях, можете след въвеждането на горната команда да натискате **[n]** (от англоезичното '**N**[**ext**]'), като всеки следващ път с цветови индикатор ще бъде обозначавано следващото място в текста, на което се среща вашето '**Търсене**'. Можете също да натискате **[Shift]**+**[n]**, при което търсенето ще продължи назад, към предходните места в текста с вашето '**Търсене**'. Повече подробности относно функциите и командите, достъпни в режима за преглеждане на текст, можете да видите с натискането на **[h]** (от англоезичното '**H**[**elp**]'). Както ще забележите, този режим позволява и редица манипулации върху преглеждания текст, макар да не е съвсем лесен за

ползване от неопитен потребител. За да излезете от режима за преглеждане на текст и да се върнете в командния ред, натиснете бутона **[q]** (от англоезичното '**Q**[**uit**]'). Имайте предвид, че клавиатурата трябва да е настроена да въвежда на латиница, за да работят командите.

С командата '**less**' можете например да прегледате текстовото съдържание на системния файл, където се архивира историята на последните въведени команди във вашия терминал:

```
$ less ~/.bash_history
```

или респективно:

```
# less ~/.bash_history
```

(където под '**.bash_history**' разбираме системния файл, в който се отразява историята на въвежданите команди, въведени съответно като обикновен потребител или като администратор; а '**~**' служи като обозначение за личната директория на съответния потребител). Системата ще изведе последните команди, които сте въвели в командния ред – ведно с резултатите от тях (в първия случай като потребител без администраторски права, а във втория – като администратор).

Да редактирате удобно текстовото съдържание на файлове в терминала можете например чрез текстовия редактор **Nano**, който не се нуждае от допълнителен графичен интерфейс и може да бъде ползван директно в терминала. Можете да отворите редактора, като просто въведете неговото наименование и натиснете **[Enter]**, а ако искате с него да отворите и файла, който ще редактирате, след наименованието на редактора трябва да въведете и наименованието на съответния файл:

```
$ nano Текстови_файл
```

... или респективно:

```
$ nano Адрес/Текстови_файл
```

Nano ви позволява да работите в терминала почти като в обичаен текстови редактор – като се движите в текста с помощта на стрелките и въвеждате текст от клавиатурата, триете, изрязвате, копирате, поставяте. Ако отворите **Nano**, ще видите в долната част на екрана два реда с няколко от основните команди, които можете да ползвате (като със знака '**Caret**' (**^**) във въпросното менюто се отбелязва клавишът **[Ctrl]**, а със знака '**M**' – клавишът **[Alt]** (който някога е бил наричан '**M**[**eta**]'); следователно, ако искате да изпълните например командата '**^G**', трябва да натиснете едновременно **[Ctrl]**+**[g]**, а ако искате да изпълните например командата '**M-U**', трябва да натиснете едновременно **[Alt]**+**[u]**; имайте предвид, че командите при **Nano** работят само докато клавиатурата въвежда латински символи – внимавайте ако в текстообработката междувременно ползвате и кирилица или други клавиатурни подредби, различни от **US**-стандартизацията на латиницата). Основните команди, които със сигурност ще ви потрябват веднага след като отворите **Nano**, са тези:

^G (**Get help**) или **[Ctrl]**+**[g]** – отваря пояснения за функциите на **Nano**;
^X (**eXit**) или **[Ctrl]**+**[x]** – затваря отворения в момента текстови файл;
^O (**write Out**) или **[Ctrl]**+**[o]** – запомня промените в текущия файл;
N (**No**) или **[n]** (без **[Ctrl]**) – отказ от затварянето или запомнянето;
Y (**Yes**) или **[y]** (без **[Ctrl]**) – потвърждаване затварянето/запомнянето;

^R (**Read file**) или **[Ctrl]**+**[r]** – въвеждане на съдържание в текущия файл;
M-U (**Undo**) или **[Alt]**+**[u]** – отменя последните ви редакции по текста;
M-E (**rEdo**) или **[Alt]**+**[e]** – отменя отмяната на последните ви редакции.

Едно от удобствата при **Nano** е това, че винаги можете да извикате пояснения за ползването му (като натиснете '**^G**' или **[Ctrl]**+**[g]**), а през цялото време за ваше улеснение най-основните команди в конкретния контекст се изобразяват в двата реда в долната част на екрана.

За пълнота ще отбележим и това, че популярните бързи клавишни комбинации **[Ctrl]+[c]** (копиране) и **[Ctrl]+[v]** (поставяне) могат да се ползват спрямо текста в терминала, като за целта обаче задържите натиснат и клавиша **[Shift]**. Тези команди обаче позволяват копиране и поставяне само на текстово съдържание – не можете да ги прилагате и спрямо директории и файлове, както правите това в графичния интерфейс. За тези функции в терминала са предвидени текстови команди.

От терминала можете да създадете нова директория:

```
$ mkdir Директория
```

(където под '**Директория**' разбираме наименованието на новосъздаващата се директория). Ако желаете да създадете директория на място, различно от текущата директория, трябва да въведете преди наименованието на новосъздаващата се директория и нейния адрес.

Можете да изтриете празна директория:

```
$ rmdir Празна_директория
```

Системата ще откаже да изтрие директорията, ако нямате права над нея (например защото тя е от кореново значение за системата, а вие работите като потребител без администраторски права). Системата ще откаже да изтрие директорията и тогава, когато в нея има файлове или други директории. Във втория случай можете да изтриете директория заедно с цялото нейно съдържание (стига да имате право на това) чрез тази команда:

```
$ rm -r Директория_със_съдържание
```

Можете да копирате определен файл:

```
$ cp Файл Нов_Файл
```

Ако указаният от вас **Нов_Файл** вече съществува, системата ще го изтрие и замени с новия фай.

Можете да копирате директорията заедно с цялото намиращо се в нея съдържание:

```
$ cp -r Директория Нова_директория
```

Важно е да имате предвид при копирането на файл или директория, че ако указаният нов път е вече съществуваща директория, то копираният файл или директория ще се копира в заварената директория, запазвайки своето оригинално наименование. Ако пътят завършва с несъществуваща директория, то копирането съдържание ще се именува с указаното от вас наименование.

Можете да преместите определена директория или файл (при което те ще бъдат изтрити от стария им адрес и записани на посочения от вас нов адрес):

```
$ mv Заварен_адрес/Директория_или_Файл Нов_адрес/Директория_или_файл
```

Горната команда можете да ползвате и просто за да смените наименованието на директория или файл (като въведете един и същи адрес, но със завареното и след това – с новото наименование.

Друга важна команда позволява закачане към системата на устройства (например външни носители със съдържание като **USB**-памети и подобни). Следва в тази връзка да се спрем и на понятието „**Точка на закачане**“ (**Mounting Point**) –

това е местоположение в системата, от което съответното устройство става софтуерно достъпно, когато бъде закачено хардуерно към компютъра. Такова софтуерно достъпване не винаги става автоматично, пък и не винаги става именно към точката на закачане, която вие бихте желали.

Какви устройства имате закачени хардуерно към компютъра и с какви наименования системата ги разпознава, можете да се ориентирате чрез тази команда:

```
$ lsblk
```

Ще бъде изведен подобен резултат:

```
NAME      MAJ:MIN RM   SIZE RO TYPE  MOUNTPOINT
sda         8:0    0   100G  0 disk
├─sda1      8:1    0     2G  0 part  [SWAP]
├─sda2      8:2    0     4G  0 part  /boot
└─sda3      8:3    0    94G  0 part  /
sdb         8:32   1     8G  0 disk  /run/media/Потребител/Външен_носител
```

(където от наименованията '**sda**' и '**sdb**' научаваме, че към компютъра има закачени **2** устройства – очевидно твърдия диск (който е структуриран в **3** дяла '**sda1**', '**sda2**' и '**sda3**', действащи съответно като стартираща '**boot**' система, разменен '**swap**' дял и системен '**/**' дял); и **USB**-памет, която след хардуерното ѝ закачена към системата по подразбиране е закачена софтуерно към директорията '**/run/media/Потребител/Външен_носител**', където може да бъде достъпена било от терминала, било от графичния интерфейс). При колебание можете да се ориентирате кое устройство кое е, като обърнете внимание и на размерите на съответните дялове: сумарно дяловете на устройството '**sda**' (твърдия диск) правят **100GB** (разделени съответно на **2GB**, **4GB** и **94GB**); а устройството '**sdb**' (**USB**-паметта, която не е структурирана в дялове) прави **8GB**. Ако въпреки горните индикации все пак се колебаете кое устройство кое е, можете да подавате няколко пъти командата '**lsblk**', като в някои от случаите закачате външния носител, а в други го разкачате от компютъра; редът с наименованието на външния носител (или респективно редовете, ако този носител е структуриран в дялове) ще се появява(т), когато носителят е закачен към компютъра; и ще изчезва(т), когато е разкачен. Имайте предвид, че ако междувременно сте закачили и други устройства, е възможно интересуващото ви устройство да бъде интерпретирано с ново наименование – например '**sdc**', '**sdd**' и т.н.. Да се ориентирате правилно кое устройство кое е, е от критична важност, когато например искате да го форматирате – ако подадете командата към неправилно устройство, ще изтриете записаното в него съдържание. И макар изтриването да не е напълно необратимо, все пак следва да се отбележи, че възстановяването му ще бъде доста трудно (ако изобщо е възможно).

След като вече знаете под какво наименование системата разпознава конкретно закачено към компютъра устройство, можете да укажете неговото софтуерно разкачане:

```
$ umount /dev/Устройство
```

(където под '**Устройство**' разбираме наименованието, под което системата разпознава съответното устройство или дял – например '**sda**', '**sdb**', '**sdb1**' и подобни).

След като сте разкачили софтуерно устройството, можете да укажете системата да преустанови подаването на електричество към него (за да предотвратите риска от токов удар и загуба на данни при хардуерното му разкачане):

```
$ udisksctl power-off -b /dev/Устройство
```

(където под '**Устройство**' отново разбираме наименованието, под което системата разпознава съответното устройство).

На следващо място можете да укажете повторно софтуерно закачане на устройството към системата, но в избрана от вас точка на закачане (директория, от която бихте желали да достъпвате това устройство):

```
$ mount /dev/Устройство /Точка_на_закачане
```

(където под '**Точка_на_закачане**' разбираме местоположението в системата, от което желаете софтуерно да достъпвате устройството (например Работния плот '**/home/Потребител/Desktop**'). Ако след изпълнението на горната команда отново въведете командата '**lsblk**', ще бъде изведен подобен резултат:

```
NAME      MAJ:MIN RM   SIZE RO TYPE  MOUNTPOINT
sda         8:0    0   100G  0 disk
├─sda1      8:1    0     2G  0 part  /boot
├─sda2      8:2    0     4G  0 part  [SWAP]
└─sda3      8:3    0    94G  0 part  /
sdb         8:32   1     8G  0 disk
└─sdb1      8:33   1    10G  0 part  /home/Потребител/Desktop
```

(откъдето е видно, че интересуващата ни **USB**-памет (чрез единствения дял, обособен в нея) е закачена към Работния плот и може да бъде достъпена като директория на това място в системата).

Когато свършите работа с устройството, не забравяйте да го разкачите софтуерно с командата '**umount**' и да преустановите подаването на електричество към него с командата '**udisksctl power-off**', преди да го разкачите и хардуерно от компютъра.

Следва да имате предвид, че когато закачате устройството към дадена директория в системата, ако съответната директория не е празна, нейното съдържание ще стане недостъпно (невидимо) след закачането. Оригиналното съдържание на тази директория ще бъде отново достъпно (видимо) след последващото разкачане на устройството от директорията.

След като вече можете да установите под какви наименования системата разпознава хардуерно закачените към компютъра устройства и обособените в тях физически дялове, можете да подавате различни команди към всяко от тях – като например да ги форматирате:

```
# mkfs.ext4 /dev/Физически_дял -n Наименование
```

(ако ще ползвате устройството само със Свободни операционни системи); или:

```
# mkfs.fat /dev/Физически_дял -n Наименование
```

(ако ще ползвате устройството и с несвободни операционни системи като например **Windows**).

И в двата горни случая под '**Физически_дял**' разбираме наименованието, под което системата разпознава съответния физически дял от закаченото устройство – например '**sdb1**'.

В горните два примера под '**Наименование**' разбираме наименованието, под което желаете вашето устройство да бъде разпознавано при хардуерното му закачане към системата; ако не желаете да указвате наименование, можете да изпълните предложените команди без допълнителния параметър **-n** и съответно без да указвате наименование.

И преди да завършим въпроса за директориите и файловете (независимо дали са разположени в системата или на външни носители), трябва да се спрем на правата за достъп. Както стана дума по-горе, **GNU/Linux** базираните операционни системи в основата си представляват многопотребителски сървърни системи, където всеки потребител е притежател на своите директории и файлове, но няма права спрямо директориите и файловете на други потребители. Така, ако се опитате да работите с директории и файлове, които са създадени на друго устройство, настоящата система може да ги възприеме като притежавани от друг потребител и да ви откаже достъп. Същото може да се случи и с директории и файлове, които сте създали като администратор, а след това се опитвате да достъпите като

потребител без администраторски права; както и с такива, които са от кореново значение за системата и тя по подразбиране отказва достъп до тях, освен при сесия с администраторски права.

Системата разграничава три групи потребители (като пропускаме администратора, който по дефиниция има всички права в системата и може да извършва всичко в нея):

– текущия обикновен потребител – отбелязван с '**u**' (от '**U**[**ser**]');
– групата на текущия потребител – отбелязвана с '**g**' (от '**G**[**roup**]'); и
– всички останали потребители – отбелязвани с '**o**' (от '**O**[**thers**]').

На всеки от горните видове потребители е възможно да се предоставят права за извършването на едно или няколко от следните действия: **да чете** (преглежда) определени директории и файлове; **да пише** (да нанася промени) в тях; **да изпълнява** (да стартира) като програми; или съответно да не му се предоставят **никакви права** за какъвто и да било достъп. Всяко от посочените права съответства на числова стойност (която има и буквено-символно обозначение), както следва:

– четене (преглеждане): '**4**' или '**r**' (от '**R**[**ead**]');
– писане (променяне): '**2**' или '**w**' (от '**W**[**rite**]');
– изпълняване (стартиране): '**1**' или '**x**' (от '[**e**]**X**[**ecute**]');
– никакъв достъп (без права): '**0**' или '**-**'.

Числовите стойности на всяко от горните права са подбрани така, че да могат да се сумират и полученият сбор (едноцифрено число от **0** за „никакви права“ до **7** за „всички права“) да съответства на конкретните права, признати на съответния потребител. Така например, ако потребителят има пълни права спрямо определена директория или файл (да чете, да пише и да изпълнява), това ще съответства на **4+2+1=7**; ако има право само да чете и да пише, но не и да изпълнява, това ще съответства на **4+2=6**; ако има само право да чете, но не може нито да пише, нито да изпълнява, това ще съответства на 4; ако няма право на никакъв достъп, това ще съответства на **0**.

Описаните числови стойности позволяват да въвеждате или премахвате определени права за достъп по отношение на съответния вид потребители, като по този начин управлявате достъпа до конкретни директории и файлове в системата. При това трябва да въведете три поредни едноцифрени числови стойности (съответно за текущия обикновен потребител; за неговата група от потребители; и за всички останали) – чрез тази команда:

```
# chmod ??? /Директория_или_Файл
```

(където под '**???**' разбираме трите едноцифрени числови стойности, всяка от които вариращa от '**0**' (никакви права) до '**7**' (всички права); съответно за текущия потребител, за потребителите от неговата група, и за всички останали потребители).

Така например, ако желаете да дадете право на всички потребители да четат директорията или файла, но промени да може да нанася само текущият потребител (без никой да има право да изпълнява файла като програма), числовата стойност ще бъде '**644**' (в команда от типа на '**chmod 644 /Директория_или_Файл**'; ако искате текущият потребител да има пълни права, потребителите от неговата група да могат само да четат и изпълняват като програма (без да променят), а останалите потребители да нямат никакъв достъп, числовата стойност ще бъде '**750**' (в команда от типа на '**chmod 750 /Директория_или_Файл**'.

Както проследихме по-горе, при изискване на детайли за определени директории и файлове чрез командата '**ls -l**', системата извежда подобен резултат:

```
-rw-r--r-- 1 Потребител Група 584204 Jan 16 12:30 Файл
drwxr-x--- 4 Потребител Група   4096 Dec 30 03:50 Директория
```

(където първият символ '**-**' обозначава '**файл**', а първият символ '**d**' обозначава '**директория**'). Както вече става ясно, следващите **9** символа представляват буквено-символни обозначения за правата на достъп, съответно на

текущия потребител, на неговата група и на всички останали. В горния пример файлът е достъпен за четене (**r**) и писане (**w**), но не и за изпълняване като програма (**-**) от текущия потребител (вероятно файлът изобщо не представлява програма и не може да се изпълнява като такава); потребителите от неговата група и всички останали потребители могат само да четат (**r**) файла, но не могат нито да пишат (**-**) в него, нито да го изпълняват (**-**) като програма. Съответно текущият потребител има пълен достъп до директорията от примера за четене (преглеждане) на нейното съдържание (**r**), за писане (създаване и триене на файлове и под-директории) в нея (**w**), и за изпълняване на (влизане в) директорията (**x**); потребителите от неговата група имат право само на преглеждане на съдържанието (**r**) и изпълняване на (влизане в) директорията (**x**), но не и на писане в (променяне на) нейното съдържание (**-**); всички останали потребители са лишени от всякакъв достъп до тази директория (**---**).

При управление на правата трябва да бъдете особено внимателни с директориите и файловете от кореново значение за системата (чието променяне (**w**), а в някои случаи изпълняване (**x**) и дори четене (**r**) по подразбиране е позволено само на администратора. Ако разширите неразумно достъп до такива директории и файлове на други потребители, може да отслабите сигурността на системата. Другата крайност (да направите всичко в системата абсолютно недостъпно за когото и да било, освен за администратора) също може да отслаби сигурността, тъй като би ви принудило да осъществявате всяко тривиално действие като администратор. Това означава или винаги да сте в администраторска сесия (чрез командата '**su**'), или постоянно да ползвате командата '**sudo**' (и да въвеждате парола за нейното изпълнение). Както ще проследим в Част **V**, моментите на въвеждане на парола са критични за информационната сигурност и не следва да се повтарят твърде често, защото всеки такъв момент крие риск от прихващане на паролата. Както ще проследим в Част **VII**, постоянното пребиваване в сесия с администраторски права също крие рискове, тъй като при внезапно отнемане на физическия достъп до компютъра може да изгубите контрол над системата преди да успеете да прекъснете административната сесия. Поради това препоръчваме да подходите балансирано при решаването на въпроса за правата на достъп до съответните директории и файлове, и да избягвате да въвеждате правила, които не са заложени в **GNU/Linux** по подразбиране (освен ако знаете много добре какво и защо правите).

Ако желаете да предоставите (или отнемете) само определено право на достъп на всички потребители, можете да направите това и съкратено, без да указване целия ред от съществуващи права. За тази цел трябва да използвате символите '**r**' (четене), '**w**' (писане) и '**x**' (изпълняване), които при предоставяне на права са предхождани от '**+**', а при отнемане на права – от '**-**'. Така например можете да предоставите право един изпълним файл да бъде стартиран като програма чрез тази команда:

```
# chmod +x Изпълним_файл
```

За да финализирате процеса по определяне на правата, е необходимо да укажете и това кои потребители и групи системата да разглежда като ползватели на съответната директория или файл – чрез тази команда:

```
# chown Потребител:Група /Директория_или_Файл
```

(където под '**Потребител**' разбираме наименованието на потребителя, а под '**Група**' – съответно групата, която желаете системата да разглежда като ползвател (**User**) на съответната директория или файл). След изпълнението на тази команда съответната директория или файл ще се разглежда от системата като принадлежност на съответния потребител (съответните потребители) с предоставени права на достъп.

Често се случва определена информация да бъде архивирана (което позволява няколко отделни файла или директория с файлове да се интерпретира(т) като един общ файл). Също така архивните файлове често се компресират (което позволява информацията да заеме значително по-ограничен обем от паметта на съответното устройство). И не на последно място – архивираното съдържание в много от случаите е по-устойчиво на случайни грешки при копиране – съответно се препоръчва да архивирате важно съдържание, когато предвиждате неговото преместване.

Да разархивирате преди това архивирана информация можете чрез тази команда:

`$ tar xf Архивен_файл`

(където под '**`Архивен_файл`**' разбираме съответния архив). Системата ще изведе съдържанието (но вече разархивирано) на същото място, където е архивният файл.

Можете да изведете разархивиращото се съдържание и на друго място в системата, чрез тази команда:

`$ tar xf Архивен_файл -C Адрес`

Можете на свой ред да архивирате файл или директория с файлове:

`$ tar cf Нов_архивен_файл.tar Съществуваща_директория_или_файл`

(където под '**`Нов-архивен-файл.tar`**' разбираме архивния **`.tar`** файл, който желаете да създадете).

Можете да компресирате определен файл (или вече архивиран файл или директория с файлове), с цел намаляване на пространството, което заема в паметта на вашето устройство:

`$ xz Файл_или_архивирана_директория`

Системата ще изведе съдържанието (но вече компресирано и заемащо по-малко място в паметта) на същото място, където е архивът.

Компресирани файлове и директории могат да се разкомпресират чрез тази команда:

`$ unxz Компресиран_файл_или_архивирана_директория`

Системата ще изведе съдържанието (но вече разкомпресирано) на същото място, където е архивният файл (или на друго място, ако укажете това с проследения по-горе параметър '**`-C Адрес`**').

Можете да сваляте файлови ресурси от internet чрез тази команда:

`$ wget Адрес_в_internet/Ресурс`

(където под '**`Адрес_в_internet`**' разбираме web-адреса, от който съответният ресурс е достъпен, а под '**`Ресурс`**' – наименованието на файла, в който се помещава интересуващият ви ресурс). Системата ще се свърже с internet и ще копира указания ресурс. Тази команда е полезна за бързо сваляне на файлове, без нуждата от браузър.

Можете да сваляте файловете с изходния код за компилиране на софтуерни програми от internet (нататък в изложението ще проследим подобни процеси по компилиране):

`# git clone Адрес_в_internet`

(където под '**`Адрес_в_internet`**' отново разбираме web-адреса, от който съответният ресурс с интересуващия ви изходен код е достъпен. Системата ще се свърже с internet и ще копира интересуващата ви директория ведно с всички директории и файлове, в които е разпределен изходният код. Тази команда е полезна за придобиване на изходния (програмен) код от програми, които възнамерявате да проучвате или да компилирате до получаването на машинен (изпълним) код за изпълняване като софтуер от вашия компютър.

Особено ценна при работа в терминал е функцията за разпознаване и довършване на текстовите указания (**Bash Completion**). Тази функция позволява да въведете първите няколко символа от адрес, наименование или команда и с натискане на **`[Tab]`** да изведете списък с възможности, които започват с въведените от вас символи. Това ви

позволява да се ориентирате, когато не сте сигурни за точния адрес, наименование или команда, или желаете да ускорите работата си, като като въвеждате само първите няколко символа и с натискате на **[Tab]** указвате на системата да изведе останалата част. Можете да инсталирате функцията за разпознаване и довършване на текстовите указания във вашата система:

– ако работите с **Parabola** или друга **Arch**-базирана **GNU/Linux** система:

```
# pacman -S bash-completion
```

– ако работите с **Trisquel** или друга **Debian**-базирана **GNU/Linux** система:

```
# apt-get install bash-completion
```

Респективно, ако ползвате програмата **Sudo**, командата ще има подобен вид:

```
$ sudo pacman -S bash-completion
```

... или съответно:

```
$ sudo apt-get install bash-completion
```

В някои случаи трябва да излезете от системата (**Logout**) и да влезете обратно (**Login**), за да може функцията **Bash Completion** да заработи. Оттук нататък функцията **Bash Completion** ще ви бъде на помощ винаги, когато се колебаете за точната формулировка на дадена команда; за точния адрес в системата на определена директория или файл; и за точното наименование на интересуващата ви директория или файл: просто натиснете два пъти **[Tab]** и ще видите възможните довършвания на започнатото от вас, ако изобщо има такива.

Възможно е да искате да копирате и поставяте текст в терминала (например като копирате от някъде команди, които вече са набрани, вместо да ги набирате повторно). За да правите това с бързите клавиши **[Ctrl]**+**[c]** и **[Ctrl]**+**[v]**, в терминала е необходимо (както отбелязвахме, докато ви запознавахме с текстовия редактор **Nano**) да задържите натиснат и **[Shift]** – командите съответно ще бъдат **[Ctrl]**+**[Shift]**+**[c]** и **[Ctrl]**+**[Shift]**+**[v]**.

Системата позволява и това – просто да маркирате желания текст, където го има въведен и след като отидете в терминала (маркираният текст си стои маркиран), просто да натиснете среден бутон (или Scroll-колелцето на мишката, ако не разполагате със среден бутон). Системата ще въведе в терминала маркирания от вас текст (без да е необходимо преди това да го копирате).

Възможно е и това – да наберете команда, която обаче не желаете да изпълните веднага, а малко по-късно. За да запазите въведената команда за по-късно (и да не се налага да я въвеждате повторно), можете да натиснете **[Ctrl]**+**[a]** (връщане в началото на текущия ред) и след това – **[Ctrl]**+**[k]** (изрязване на текста до края на реда). След като изпълните следващите желани команди и дойде ред на тази, която сте запазили за по-късно, можете да натиснете **[Ctrl]**+**[y]** (поставяне обратно в терминала на последно изрязания текст).

Ако желаете да въведете команда, която скоро сте въвеждали в терминала, можете да я потърсите и чрез натискане на горна стрелка **[↑]** – при всяко следващо натискане на горната стрелка ще се извежда предходна изпълнена от вас команда. Ако натиснете долна стрелка **[↓]**, ще се върнете обратно, към последната изпълнена команда. Това е удобно например при необходимост от многократно въвеждане на няколко еднакви команди. Можете да ползвате тази функция и при повтарянето на една и съща команда няколко пъти – като връщате назад и редактирате само тази част от командата, която искате да се различава.

Можете да търсите и в по-отдавна ползваните команди – като натиснете **[Ctrl]**+**[r]** и въведете поне няколко символа от интересуващата ви команда. Ще бъде изведена последната въведена във вашия терминал команда, съдържаща тези символи. А ако продължите да натискате **[Ctrl]**+**[r]**, ще продължат да бъдат извеждани и по-

старите команди, въвеждани във вашия терминал, които съдържат същите символи. Така можете да достъпвате бързо необходимите ви команди при еднотипни дейности, изискващи същите или сходни команди.

Възможно е в някакъв момент да въведете команда, за която да бъде изведено съобщение, че не може да бъде намерена: 'bash: Команда: command not found'. Това обикновено се дължи на допусната правописна грешка, но може да се дължи и на това, че нямате инсталиран необходимия софтуерен пакет за изпълнението на тази команда. Ако сте сигурни, че не става дума за правописна грешка, можете да потърсите кой софтуерен пакет е свързан с интересуващата ви команда:

 # pacman -F Команда

Ще бъде изведен списък от софтуерни пакети, работещи с търсената команда. Можете да инсталирате тези от тях, които ви трябват, за да изпълните командата. За да сте сигурни обаче, че работите с актуални бази с данни, можете преди изпълнението на горната команда да обновите базите:

 # pacman -Fy

Така ще сте сигурни, че търсите командите в съответствие с актуалното състояние на софтуерните пакети, обновени до техните последни версии.

И накрая – не можем да завършим бележките за командния ред, без да се спрем на основните възможности за достъпване на помощна информация за интересуващите ви програми и команди. Чрез терминала можете да разгледате подробно ръководство за начина, по който функционира коя да е от програмите, инсталирани във вашата система; или конкретна команда, която желаете да ползвате:

$ man Програма_или_команда

(където под '**Програма_или_команда**' разбираме наименованието на интересуващата ви програма или конкретна команда). Системата ще изведе структурирано текстово ръководство (от англоезичното '**Man**[**ual**]') с обяснение за основните функции на съответната програма (команда), синтаксиса (поредността и начина на въвеждане) на свързаните с тази програма (команда) параметри и кратки описания за техните функции и възможности.

Ключова за всяко такова ръководство е частта **Synopsis**, представляваща своеобразна „формула“ за ползването на съответната програма (команда). Синтаксисът на подобни „формули“ обикновено включва наименованието на самата програма (команда), следвано от допустимите за тази програма (команда) аргументи (заграждани в квадратни '**[]**' скоби, когато не са задължителни и разделяни с вертикална '**|**' черта, когато са допустими алтернативно на принципа „или-или“). Например за командата '**mkdir**' (създаване на директория) като **Synopsis** е предложена „формулата“ '**mkdir [OPTION]... DIRECTORY...**' – откъдето научаваме, че за тази команда могат да се въвеждат неопределен брой незадължителни аргументи (възможности; **Option**[**s**]) и задължителен неопределен брой (не)съществуващи директории; за командата '**cd**' (преместване в друга директория) като **Synopsis** е предложена „формулата“ '**cd [-L|-P] [DIR]**' (откъдето научаваме, че за нея са допустими два незадължителни параметъра, като първият от тях има две алтернативно допустими стойности – '**-L**' (преместването да последва препратката, ако има такава) или '**-P**' (преместването да се придържа само към посоченото физическо местоположение, без отношение към каквито и да било препратки), а вторият от тях може да включва една съществуваща директория. Макар „формули“ от този тип да са непълни, все пак дават отправна точка за синтаксиса, на който се подчинява съответната програма (команда).

Както стана дума по-горе, можете да се придвижвате в съответното ръководство (аналогично на представения по-горе режим при '**less**' преглеждането на текстове) със стрелките и бутоните **[HOME]**, **[PGUP]**, **[PGDN]** и **[END]**, да търсите с '**/Търсене**' и **[n]** за следващи съвпадения, и да прекратите четенето с **[q]** (което ви връща обратно в командния ред).

Още информация за начина на функциониране на определена програма или команда можете да достъпите и така:

`$ Програма_или_команда --help`

... като в някои случаи е възможно и съкратено въвеждане:

`$ Програма_или_команда -h`

След изпълнението на горната команда ще останете в командния ред, но над него ще бъде изведена синтезирана информация за посочената от вас програма (команда). Тази информация в повечето случаи е по-ограничена от понякога излишно пространните напътствия, достъпни с командата '**`man`**', но нейната систематичност и пестеливост често се оказва по-подходяща за начинаещия потребител.

Горните базови команди ви позволяват да се „придвижвате“ чрез терминала навсякъде из системата, да преглеждате съдържанието на съответните директории и текстови файлове, да създавате, променяте, стартирате, копирате, премествате и изтривате директории и файлове (в т.ч. инсталираните програми), и да присвоявате определени права на достъп на съответните видове потребители. Възможността да редактирате съдържанието на прости текстови файлове (каквито са практически всичките системни файлове с настройки при **GNU/Linux** системите) е ключово преимущество пред коя да е друга компютърна работна среда. Въпреки, че графичният интерфейс може да изглежда по-удобен, в действителност е много по-ограничен и трудоемък при ползване в сравнение с командния ред. И не на последно място – в някои случаи стартирането на графичната среда може да се окаже изобщо невъзможно (било поради технически проблеми, било поради съображения от значение за информационната сигурност). Тогава терминалът ще се окаже единственият работещ вариант за достъп. Ето още един аргумент за това да свикнете с терминала и да започнете да го ползвате активно; още повече, че голяма част от следващите действия, които предстои да осъществим, изобщо няма как да бъдат изпълнени по друг начин, освен от командния ред.

II. ИНСТАЛИРАНЕ НА ОПЕРАЦИОННАТА СИСТЕМА

Преди да пристъпите практически към инсталирането на ваша Свободна операционна система (**II**), препоръчваме да се запознаете и със следващите три части от изложението, тъй като те са от значение за цялостното обезпечаване на информационната сигурност. Без адекватна защита на софтуера от „ниско“ ниво (**III**) системата би продължила да бъде уязвима на атаки, които поради особеното място на този вид софтуер могат да останат напълно „невидими“ за операционната система. Обезпечаването на паролите (**V**) е ключов момент от сигурността на всички криптографски технологии и именно там се намира „най-слабото звено“ на такива технологии – поради което е добре да се запознаете и с тази тема, преди да започнете да създавате надеждни пароли и да ги ползвате по надежден начин. И накрая – цялостното шифроване на системата (**IV**), което от своя страна предхожда нейното инсталиране, е единственият начин да гарантирате нейната неприкосновеност и от там – неприкосновеността на вашето поверително съдържание.

В изложението не сме се съобразили с тази последователност поради определени дидактически съображения. Веднага, след като разгледахме някои от основните възможности на командния ред, пристъпваме към простото инсталиране на една обикновена (нешифрована) операционна система – защото въведените дотук знания и умения ще ни бъдат необходими в следващите стъпки. Последните, от позицията на вече достигнатото ниво на разбиране, ще бъдат по-лесни за изпълнение и това оправдава допуснатото разместване. Поради тази причина започваме именно с обикновено инсталиране на Свободна операционна система, което на следващ етап ще повторим още веднъж с надграждане на нейното цялостно шифроване, но вече в наистина сигурен хардуер, който работи с надежден софтуер от „ниско“ ниво.

1. Особености на общността при Свободния софтуер

Както стана дума в Част **I**, Свободният софтуер е преди всичко **с в о б о д е н** (съгласно израза „**Free as in Freedom**“). Вие можете свободно да свалите, инсталирате и ползвате този софтуер, без да е необходимо да искате разрешение от някого, да плащате за лицензи или да се съобразявате с патентни ограничения. Единствената забрана е срещу затварянето на софтуера по отношение на по-нататъшното му свободно разпространение и ползване; софтуерът (в първоначалния му вид или след неговите модификации) трябва да продължи да бъде свободен за всички.

https://en.wikipedia.org/wiki/Free_as_in_Freedom

Свободата при такъв софтуер е несъвместима със съществуването на една компания-собственик, която единствена да има достъп до изходния код и единствена да може да модифицира и разпространява софтуера със съответни лицензни условия. Вместо това Свободният софтуер се създава и усъвършенства от всеки желаещ в световната общност от програмисти. Следователно вие можете да придобиете такъв софтуер (или негови модификации) от много различни източници. Това разбира се поставя въпроса за гарантиране автентичността и интегритета на вашата версия. Ако бъде „подпъхната“ манипулирана версия на софтуера или такава, която е модифицирана некомпетентно, е възможно да се компрометира информационната сигурност. Ето защо е важно да се снабдите с версия, за която може да се потвърди, че е щателно проверена, автентична и гарантираща сигурността.

Сред програмистите, които участват активно, подпомагат или дори само наблюдават развитието на софтуера, се формират относително независими едно от друго ядра, които едновременно се подпомагат помежду си и контролират работата на своите колеги. Липсата на единен управляващ център предотвратява възможностите за умишлено вграждане на скрити „задни врати“ в софтуера – защото всеки може да извърши своя независима проверка в достъпния за преглеждане изходен код на софтуера и да направи своята констатация публично достояние. Липсата на единен управляващ център позволява в проверките да се приложи допълнителна експертиза и алтернативни гледни точки, което увеличава вероятността евентуални пропуски да бъдат забелязани, огласени и отстранени.

Но независимо от липсата на единен управляващ център, по-активните програмисти и групи от програмисти все пак се утвърждават като „водещи“ в разработването и усъвършенстването на софтуера. В условията на децентрализация и незадължителност на процеса това няма как да стане на друг принцип, освен на база отдадения принос и признатата компетентност. Така софтуерът (който продължава да бъде под погледа и контрола на неограничен брой независими

проверители) се асоциира с конкретни програмисти и групи от програмисти, които потвърждават официално неговата автентичност и интегритет (под страх от компрометиране на тяхната работа, ако бъдат установени слабости). Така – въпреки липсата на една „акредитирана“ компания-собственик, вие можете да се доверите на потвържденията за качествата на Свободния софтуер от многобройни и независими един от друг източници. Ако това не е достатъчно за вас, можете да организирате и своя собствена проверка чрез ваши доверени специалисти. По-долу ще проследим процеса по инсталиране на една Свободна операционна система, като специално внимание ще обърнем на удостоверяването, че разполагате с версия на софтуера, чиято автентичност и интегритет, а от там – сигурност и надеждност – е потвърдена.

2. Сваляне на автентично копие от инсталационния .iso файл

Базирана на **GNU/Linux-libre** Свободна операционна система можете да инсталирате от предварително подготвена „жива“ **USB**-памет (**Live-USB**) или от „жив“ оптичен диск (**Live-CD**). По време на инсталирането „живият“ носител изпълнява функциите по управление на компютъра, докато същият бъде инсталиран с вашата операционна система, способна сама да поеме управлението. Несвободният софтуер обикновено се доставя на „лицензирани“ носители с памет или се позволява инсталирането му от web-базирани хранилища, които поемат контрола над компютъра докато протича инсталационният процес (а в последно време изобщо се отрича идеята за инсталиране на пълноценна операционна система, вместо което се предлага предоставяне на софтуера като облачна услуга – за да бъде потребителят напълно зависим от съответния облачен сървър). За разлика от това, при Свободния софтуер можете да изтеглите свободно копие от необходимия ви инсталационен **.iso** файл, за да подготвите „жив“ носител, с който след това да инсталирате вашата система. По-нататък в изложението ще проследим и възможност вместо да сваляте инсталационен **.iso** файл, директно да подготвите вашия „жив“ носител като инсталационна система – за която цел обаче е необходимо вече да разполагате с действаща Свободна операционна система.

https://en.wikipedia.org/wiki/Live_USB
https://en.wikipedia.org/wiki/Live_CD

От съществуващите напълно Свободни операционни системи на този етап можем да препоръчаме **Parabola**, **Trisquel**, **Guix** и **PureOS**. Както вече обосновахме по-горе в изложението, ще следваме работата с **Parabola**, но няма пречка да приспособите следващите напътствия и към друга предпочитана от вас Свободна операционна система. Повече подробности и списък с всичките съществуващи Свободни операционни системи можете да видите например в официалния web-сайт на проекта за Свободен софтуер **GNU**.

https://wiki.parabola.nu/Get_Parabola
https://trisquel.info/en/download
https://guix.gnu.org/en/download/
https://pureos.net/download/
https://www.gnu.org/distros/free-distros.html

Независимо коя Свободна операционна система сте избрали, ще забележите, че се предлагат тестови версии „в процес на разработка“ (**Test**) и „стабилни“ (**Stable**) версии, от които препоръчваме да изберете стабилните (тъй като софтуерът при тях е проверен и ако са установени технологични слабости, са отстранени). Предлагат се и стабилни версии с предвидена дългосрочна **LTS**-поддръжка (**Long Term Support**), за които се очаква, че са достатъчно добри и това обуславя решението на съответните разработчици да ги обезпечават в по-продължителен период след тяхното пускане. В зависимост от вида на вашия централен процесор, трябва да изберете версия с **32**- или с **64**-битова архитектура, като свалите копие от инсталационния **.iso** файл (най-добре от официалния web-сайт на избраната операционна система).

След като свалите копие от необходимия ви инсталационен **.iso** файл, ви препоръчваме преди всичко да проверите дали вашето копие наистина съответства на това, което очаквате, че сте свалили. Разработването на една операционна система е значително начинание, което – както отбелязахме по-горе – включва съвместните усилия на голям брой програмисти от целия свят. Това неизбежно трябва да се синхронизира, а получените резултати да се потвърдят от поне няколко независими източника. За да се гарантира, че софтуерът, който се готвите да инсталирате, е

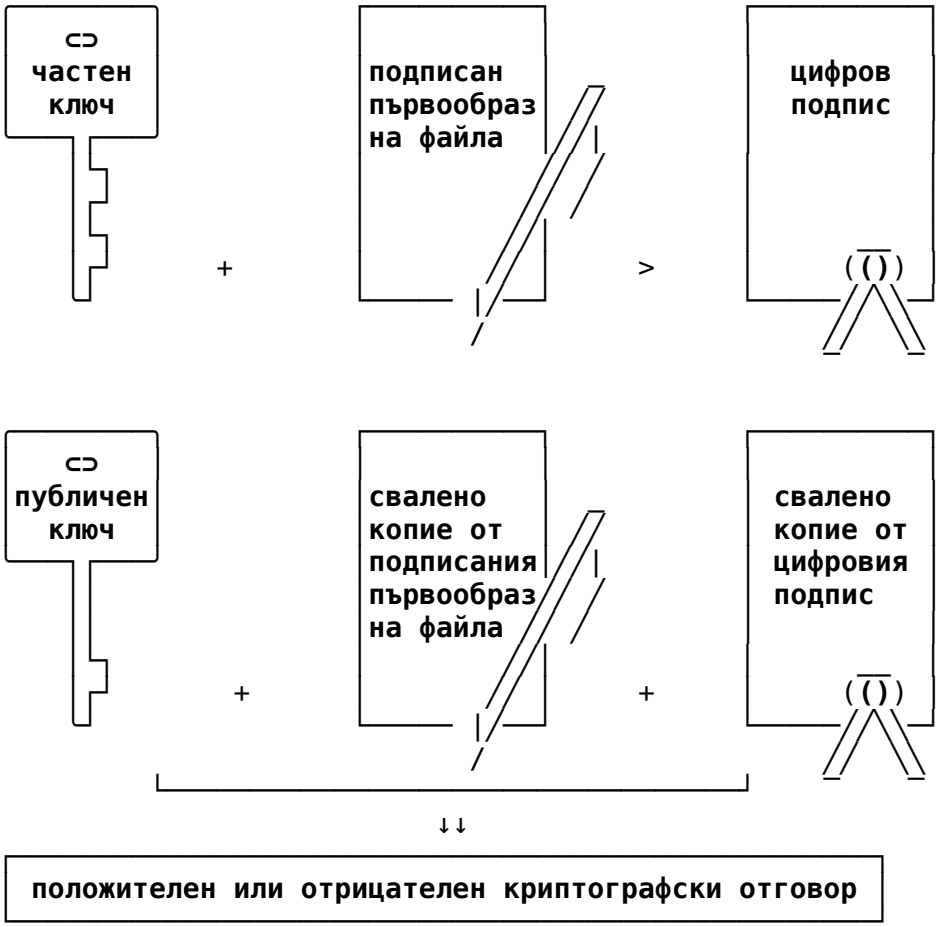
наистина този, който е бил синхронизиран и потвърден (междувременно не е бил компрометиран по някакъв начин), заедно с инсталационния **.iso** файл обикновено се предлага и **`.sig`** файл с цифров подпис (**Digital Signature**), „положен“ върху първообраза на свалените от вас инсталационен **`.iso`** файл, или 'проверовъчна сума' (**Check Sum**), генерирана от символите в проверявания файл и позволяваща да се установи дали не е настъпила някаква промяна в тези символи. Всяка от тези технологии позволяват да установите дали сваленото от вас копие на инсталационния **`.iso`** файл е идентично на „подписания“ с цифров подпис (или проверения с 'проверовъчната сума') първообраз на файла, който е бил синхронизиран и потвърден.

`https://en.wikipedia.org/wiki/Digital_signature`
`https://en.wikipedia.org/wiki/Checksum`

По-надолу ще проследим подробно начина, по който функционират криптографските технологии. Тъй като обаче цифровото подписване и 'проверовъчните суми' разчитат преди всичко на такива технологии, е необходимо да вземете предвид някои основни принципи още тук. На този етап е достатъчно да знаете, че цифровият подпис се „полага“ върху „подписвания“ файл посредством 'частен' ключ (**Secret Key**), известен само на неговия притежател. В момента на създаването на 'частния' ключ е бил създаден и 'публичен' ключ (**Public Key**), който се асоциира еднозначно и недвусмислено с 'частния' ключ, но за разлика от него се разпространява свободно (публично). 'Публичният' ключ може да се ползва от всеки желаещ, за да проверява цифровите подписи, които са „положени“ с асоциирания към него 'частен' ключ.

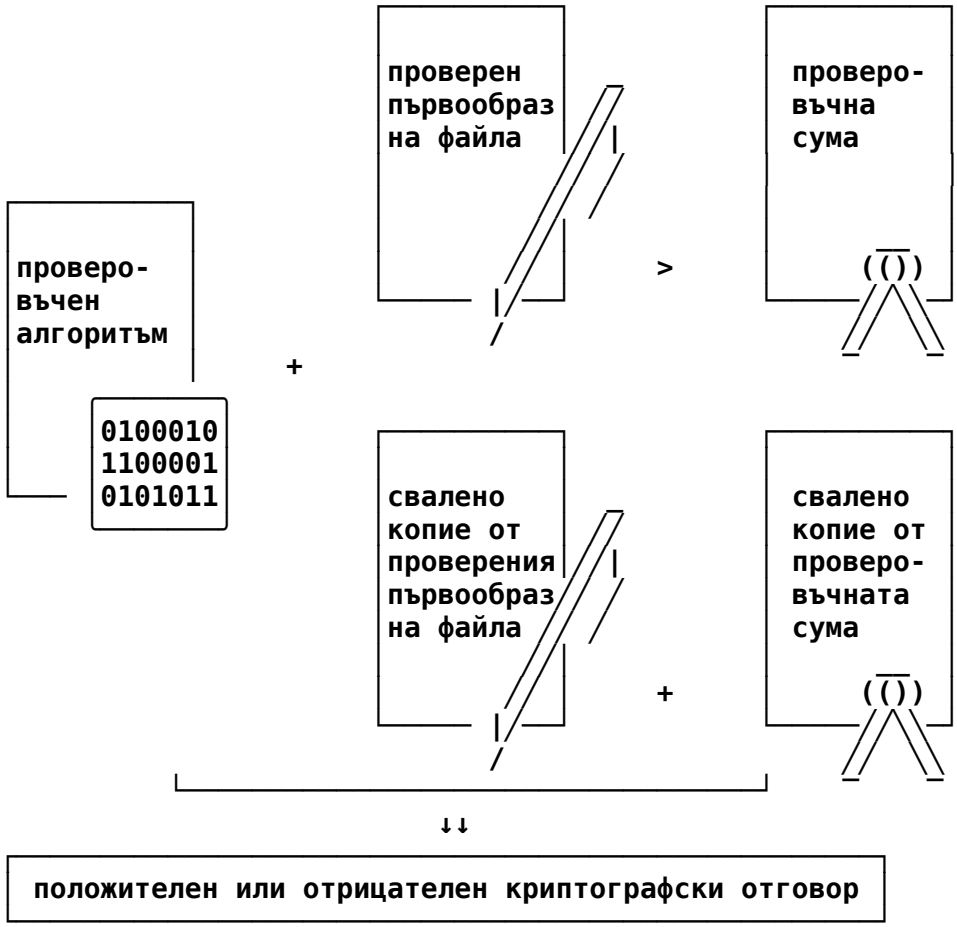
`https://en.wikipedia.org/wiki/Public-key_cryptography`

Самият цифров подпис представлява сложна криптографска производна от всичките символи в „подписания“ файл и от символите в 'частния' ключ, ползван за „полагането“ на цифровия подпис. По време на проверката се образува още една сложна криптографска производна – този път от всичките символи в проверявания файл, от символите в „положения“ цифров подпис и от символите в 'публичния' ключ. Само ако символите в проверявания файл съответстват криптографски на символите в цифровия подпис (следователно проверяваният файл е идентичен с „подписания“ първообраз) и ако символите в цифровия подпис съответстват криптографски на символите в 'публичния' ключ (следователно цифровият подпис е „положен“ чрез 'частния' ключ, съответстващ именно на този 'публичен' ключ), от проверката ще бъде изведен положителен криптографски отговор (**`Good signature`**). Ако дори един символ е бил променен по някаква причина, ще бъде изведен отрицателен криптографски отговор (**`Bad signature`**).



Принципна схема на цифровото „подписване“ на файлове

'Проверовъчните суми' разчитат на съответни криптографски алгоритми (например **MD5**, **SHA512** и др.), които позволяват от всичките символи в проверявания файл да бъде изведена криптографска производна, представляваща точно определена буквено-цифрена комбинация от няколко десетки символа. Конкретният набор от символите в проверявания файл води до извеждането на точно тази буквено-цифрена комбинация, като промяната дори само на един символ води до извеждането на коренно различна буквено-цифрена комбинация. Това свойство на 'проверовъчните суми' позволява да установите лесно (без да преглеждате съдържанието на проверявания файл) дали сваленият от вас инсталационен **.iso** файл е идентичен със своя първообраз.



Принципна схема на използването на 'провероъчни суми'

Евентуален отрицателен криптографски отговор (липса на съответствие между сваленото от вас копие на инсталационния **.iso** файл с „подписания“ първообраз; или липса на съответствие между ползвания за „подписването“ 'частен' ключ и ползвания при проверката 'публичен' ключ; или респективно – липса на съответствие между предоставената 'провероъчна сума' и генерираната от вас) може да се дължи най-общо на две причини. Първо, по време на свалянето на файла може да е възникнала техническа грешка в трансфера на данните и файлът да се е повредил. Второ (по-малко вероятно, но не е напълно невъзможно), срещу вас персонално или срещу потребителите по принцип може да е насочена атака с предварително подготвено „подправено“ копие на инсталационния **.iso** файл (например с компрометирана сигурност), с което е подменен оригиналният файл.

В първия случай е достатъчно да свалите повторно инсталационния **.iso** файл, цифровия подпис и 'провероъчната сума', и да извършите повторна проверка – ако разполагате с прилична internet-свързаност, е малко вероятно отново да възникне техническа грешка. Ако обаче продължавате да получавате отрицателен криптографски отговор, можете например да се преместите на друг компютър, свързан с internet от друга точка (откъдето не сте се свързвали преди време, като в момента не се свързвате с други internet-ресурси, които обичайно ползвате), за да подготвите своя „жив“ носител на спокойствие. Ако въпреки всичко продължавате да получавате отрицателни криптографски отговори, това вече може да се дължи на пропуск в работата на програмистите (което също е малко вероятно, тъй като работата се следи от голям брой независими едно от друго лица) и в такъв случай можете просто да изберете някоя от другите Свободни операционни системи.

Един действителен цифров подпис има подобен вид:

```
-----BEGIN PGP SIGNATURE-----
Version: GnuPG v1.4.11 (GNU/Linux)

iEYEABECAAYFAlRWeJIACgkQtO+5842K6/FFbACcDNoEBlu2/8eNLkzXMbQQ0rTM
rxYAnjT8zU1BtemEofoObJ+6hGqLFGZE
=EbQg Trisquel
-----END PGP SIGNATURE-----
```

В горния пример представяме цифровия подпис на **Trisquel**, „положен“ върху първообраза на инсталационния **`.iso`** файл '**`trisquel_7.0_amd64.iso`**' от екипа, разработващ тази операционна система (но препоръчваме да не се доверявате на примери като горния, а да сваляте сами необходимите ви подписи, например от официалните web-сайтове на предпочитаните от вас операционни системи).

Един 'публичен' ключ има подобен вид:

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v1.4.10 (GNU/Linux)

mQGiBEWqgWsRBACyvlHQxUA6RnHvXLZTowvkja5RlfVea0jZze1y4qjmjJ+FKluu
WKmLr68h0RQaGb6D9drhpDqcGyY/Hcvd4xe/Lbu7rZ2hXXW/7hPdWXLQe9sFnGFZ
B3L3xy+nMZdQ2HvnYuqLK7gm6SWO6aoV7X8w63hj0pC75TWYY2COJ4O86wCgw0xL
OORa0TC549VTbgRiAYq9yKsD/j2Y72cuQSfQlKDFb4AbknMCWdH+mUlEL2vcOP4l
vjTO+0ekMK9QLBeZQQrV8Kbi5r3lWhEZLxEdFMXMrics/Z2DIZ4oOL+zxTatvc2K
SVJoTjGyk8GxThdpa2WCRIe+Uf0Fm9gucuVkfUuk7W46SLyTYWFDRqa7IUtbnC+G
QUFnA/986wzmODkuyipMKvFnVd+4/9ubfUV9DZ76StVr9+nx+sOi2GtK3AiHggLG
yOw1Wnt1HIaopA7Y9olc7S6nnakMq6rpKA+MoFFSUmF18WfTs1ha9T+kTQ8JA00L
N/9GacaAEDQwKqPWWO9FnRUN46m43ukG6/0uDT3G3ItdRhzewrRSVHJpc3F1ZWwg
R05VL0xpbnV4IChUcmlzcXVlbCBHTlUvTGludXggc2lnbmluZyBrZXkpIDx0cmlz
cXVlbC1kZXZlbEB0cmlzcXVlbC5pbmZvPohgBBMRAgAgBQJKqFs1AhsDBgsJCAcD
AgQVAggDBBYCAwECHgECF4AACgkQtO+5842K6/GptQCgot6ILdUO7k7ZgSAPk4rX
dYboGsUAnjLTU+Gyqcpk4aGEH7LlEfcLG1GhuQINBEWqgXEQCACs5IoPtgOMBGHu
iz3Y48caxWNws6XZ2PedsX2GaPyuVTwraaW0pfrxT9pJgsR7NQOabQPUIdvMHAIK
gowrVWzol3EuF9yoY601Zntzk/HAfMLm6QFQBykIpbRXpf5ts4aHCZuHredoxb+y
bEvYbgCzEDIA+v8+2H8YvX1ji+fhFoXk0rbYRgUSoHsNsOR+xAvzYBpYtUeHtFua
I0Yn5A+7AaobzlPRNssOyi5YpPBWKgrnGUi8hZ4S1MxyzEd5IO1uZjX46hP17QSc
Bv5+iOVFBrsIb1jwVHcJ4RWRgCOwJgVXP6Bqurbp4iSX1H1FmEqTKNAeBGj0kCqZ
BuLI4Ke7AAMGB/9igeS4U77kpXvG7+NmNhjnZM9hCKoa4a8ND/YBaj4YElR0eXv9
9ayXVf12mWIKBMRs4qr/6JnuT0EqB3YBcGt2sYFFE/7Oy9/ZmSD1lS9ib9C3LsPa
AgKSpL+2E/NtZtTjwQ77LCGYP7RMmGCSoLVwa2/KQNCFnWe+dfKR8A8ODHLsBK0U
mr8j3uPZBFudaK2VcnBKf74kkaa7hcFJo5nkjXDwQdletkUG0RxlF/6QkzA/gphd
MwRt8LYBs/8CWwJM73NiAB3+qEGJ0khiIct9MpZ/VCc4bKZ/w97qcwQ3zJMtvuCD
rKzMS+Kyea4Wpy+DaGlAs2ZSUHvG3mo41bnfiEkEGBECAAkFAkWqgXECGwwACgkQ
tO+5842K6/EoPACffP0BW9s0UyehOrXR0GhkcTEw5iYAoLjJ8WpzucaKz9w4Y1Sb
7E05kXhA
=zlqX
-----END PGP PUBLIC KEY BLOCK-----
```

В горния пример представяме 'публичния' ключ на **Trisquel** (но отново препоръчваме да не се доверявате на примери като горния, а да сваляте сами необходимите ви 'публични' ключове от официални източници – в Част **VIII** ще обърнем специално внимание на въпроса за проследяване на тяхната автентичност).

Една 'проверовъчна сума' има подобен вид:

79011e1dce0cf6d1f3aa206bf8c135d4 Файл

... или подобен вид (с включен адреса на проверявания файл):

79011e1dce0cf6d1f3aa206bf8c135d4 Адрес/Файл

(където в предложения пример **32**-символната буквено-цифрена комбинация е самата 'проверовъчна сума', следвана от два интервала и наименованието на проверявания файл). Това е 'проверовъчната сума' на произволен файл, извлечена с вече остарелия проверовъчен алгоритъм **MD5**. В момента се предпочитат проверовъчни алгоритми от типа на **SHA512**, където 'проверовъчната сума' е по-голяма, но общата ѝ структура е идентична с горния пример. Една действителна **SHA512** 'проверовъчната сума' има подобен вид:

e4baa49c333c42c9e1b0a60ead6e3ab3312e3c6f0f3b3fcccbad7f3e64b52b712af49e51750025f7c1b45f1eadbbafbcef3c7dcfd08a758960d35c1fcd02c7d6 parabola-systemd-cli-dual-complete-2018.06.02.iso

(където в случая се проверява инсталационният **.iso** файл '**parabola-systemd-cli-dual-complete-2018.06.02.iso**' на дистрибуцията **Parabola** (но отново – намирайте по свои независими канали необходимите ви 'проверовъчни суми'). Следва да отбележим, че 'проверовъчната сума' от горния пример е изписана на един единствен ред, но поради своята голяма дължина може да изглежда разположена на няколко отделни реда. Ако бъде накъсана целостта на 'проверовъчната сума', проверка няма да може да бъде извършена.

Цифровите подписи, 'публичните' ключове и 'проверовъчните суми' могат да бъдат оформяни като прости текстови файлове, чието съдържание е съответният цифров подпис, 'публичен' ключ или 'проверовъчна сума' във вид на прост текст (както показахме по-горе), а разширението на файла може да бъде **.sig** (от '**Sig**[**nature**]' за подписите), съответно **.pub** (от **Pub**[**lic Key**] за 'публичните' ключове). При 'проверовъчните суми' обикновено няма разширение, но в някои случаи би могло да бъде **.sum** (от '**Sum**[**ma**]'). Оформени по такъв начин, тези файлове могат да бъдат ползвани за криптографската проверка, която се готвим да извършим. За тази цел трябва да копирате необходимите ви цифрови подписи, 'публични' ключове и 'проверовъчни суми' от internet (или от ваш надежден източник) и да ги оформите в прости текстови файлове с необходимите разширения и с избрано от вас подходящо местоположение в системата, откъдето ще ви бъде удобно след това да ги приложите. Не е излишно още тук да започнете да свиквате да работите в командния ред – с проследените в Част **I** команди като например '**mkdir Директория**', '**cd Директория**', '**wget Адрес_в_internet/Ресурс**', '**cat > Файл**' и пр.

'Публичният' ключ, както стана дума, се асоциира еднозначно и недвусмислено със съответния 'частен' ключ (известен само на своя притежател). Това означава, че не съществува и теоретично не може да съществува друг 'частен' ключ, който да се асоциира със свалените от вас 'публичен' ключ; и обратно – не съществува и теоретично не може да съществува друг 'публичен' ключ, който да се асоциира с известния само на неговия притежател 'частен' ключ. Стана дума и за това, че 'публичният' ключ може да се ползва за проверка на цифровите подписи, „положени“ чрез асоциирания с него 'частен' ключ. Следователно – може да се удостоверят две неща: първо, че цифровият подпис е „положен“ наистина от притежателя на 'частния' ключ, с който е асоцииран предоставеният 'публичен' ключ; и второ, че първообразът на проверявания файл (който е бил „подписан“ от притежателя на 'частния' ключ) е идентичен със свареното от вас копие на този файл. Само при изпълнени и двете условия ще бъде изведен положителен криптографски отговор (**Good signature**).

За да бъде обаче криптографската проверка надеждна, е необходимо да я извършите в свободна софтуерна среда – което ще гарантира, че процесът се осъществява правилно и софтуерът действително извежда коректни криптографски отговори. В противен случай няма никакви доказателства (освен уверенията на компанията-собственик), че ползваният от вас несвободен софтуер работи правилно. Възможно е дори това – процесът на проверката да бъде специално компрометиран, за да получите „положителен“ криптографски отговор по отношение на инсталационен **.iso** файл, който всъщност не е идентичен на „подписания“ първообраз. Поради това ви насърчаваме да работите на помощен компютър, който вече е инсталиран със Свободен софтуер. Ако обаче въпреки нашето

предупреждение решите да подготвите вашия „жив“ носител с несвободен софтуер и се нуждаете от съдействие, ви препоръчваме да се обърнете към вашия системен администратор.

За да извършите криптографска проверка, е необходимо да поставите проверявания инсталационен **.iso** файл и предоставения **.sig** файл с цифровия подпис на едно и също място в системата (можете да си послужите с командата '**mv**'), и да се преместите с терминала на същото място (с командата '**cd**'). Ако не се намирате на същото място, в дадените по-долу примери ще трябва да въвеждате и адресите на съответните файлове.

Можете в крайна сметка да стартирате криптографската проверка така:

```
$ gpg Подпис.sig
```

(където под '**Подпис.sig**' разбираме точното наименование на **.sig** файла с цифровия подпис – като внимавате за големи и малки букви, и въведете включително разширението '**.sig**'; за да сте сигурни, че въвеждате правилно, можете да си служите с **[Tab]** за автоматично довършване на това, което сте започнали да въвеждате).

При горната команда е възможно системата да изведе указание да въведете и наименованието на проверяван файл (**Please enter name of data file:**). За да избегнете такова питане, можете да зададете горната команда и по-пълно, като директно укажете на кой файл желаете да извършите криптографска проверка:

```
$ gpg --verify Подпис.sig Проверяван_файл
```

(където под '**Проверяван_файл**' разбираме наименованието на файла, чийто цифров подпис проверявате).

Когато въведете командата (не забравяйте да въведете и адреса, ако не сте преместили терминала на същото място), системата ще да прочете цифровия подпис, ще потърси проверявания файл и посредством 'публичния' ключ, с който е асоцииран 'частният' ключ на автора на цифровия подпис, ще установи дали проверяваният файл е идентичен на своя първообраз и дали подписът е „положен“ от притежателя на 'частния' ключ.

Вероятно 'публичният' ключ, асоцииран с 'частния' ключ, с който е „положен“ цифровият подпис върху първообраза на проверявания файл, го няма въведен във вашата система. Тогава ще бъде изведен подобен резултат:

```
gpg: Signature made Sun 02 Nov 2014 09:17:57 PM EET
gpg:                using DSA key B4EFB9F38D8AEBF1
gpg: Can't check signature: No public key
```

От дадения пример е видно, че подписът е „положен“ на **02.11.2014** г. малко преди **09:18** ч. посредством шифровъчния алгоритъм **DSA** (**Digital Signature Algorithm**), а ползваният за това 'частен' ключ се асоциира с необходимия ви за проверката 'публичен' ключ с идентификатор (**key ID**) '**B4EFB9F38D8AEBF1**'. Той обаче не е намерен във вашата система, а проверката не може да се осъществи без него (**Can't check signature: No public key**). След като вече знаете идентификатора на необходимия ви 'публичен' ключ, можете да го изискате от системата доверени сървъри (по-долу ще се спрем подробно на начина, по който функционира тази мрежа). Засега е достатъчно да въведете тази команда:

```
$ gpg --search-keys 'Идентификатор'
```

(където под '**Идентификатор**' разбираме буквено-цифрената комбинация, с която се идентифицира интересуващият ви 'публичен' ключ. Ако изискате представения по-горе Идентификатор '**B4EFB9F38D8AEBF1**' на 'публичния' ключ на **Trisquel**, системата ще изведе подобен резултат:

```
gpg: searching for "B4EFB9F38D8AEBF1" from hkp server keys.gnupg.net
(1)  Trisquel GNU/Linux (Trisquel GNU/Linux signing key) <trisquel-devel@tr
     1024 bit DSA key E6C27099CA21965B734AEA31B4EFB9F38D8AEBF1, created:
2007-01-14
Keys 1 of 1 for "B4EFB9F38D8AEBF1". Enter number(s), N)ext, or Q)uit >
```

... и ще влезе в режим на очакване за вашето следващо решение: да въведете намерения 'публичен' ключ (като въведете неговия пореден номер '**1**' и натиснете **[Enter]**); да потърсите следващи възможности, ако има такива (като въведете **[n]** (от '**N**[**ext**]') и натиснете **[Enter]**); или да се откажете и върнете в изходно положение (като въведете **[q]** (от '**Q**[**uite**]')и натиснете **[Enter]**).

В случай, че решите да въведете (**Import**) намерения 'публичен' ключ (сочен като притежаван от екипа на **Trisquel**), трябва да въведете неговия пореден номер (в случая '**1**') и да натиснете **[Enter]**. Ще бъде изведен подобен резултат:

```
gpg: key B4EFB9F38D8AEBF1: public key "Trisquel GNU/Linux (Trisquel
GNU/Linux signing key) <trisquel-devel@trisquel.info>" imported
gpg: Total number processed: 1
gpg:               imported: 1
```

'Публичният' ключ, сочен като притежаван от екипа на **Trisquel**, е въведен във вашата система. Сега вече можете да повторите опита си за извършване на криптографска проверка:

```
$ gpg Подпис.sig
```

След като разполагате с цифровия подпис, 'публичния' ключ, асоцииран с ползвания за „подписването“ 'частен' ключ и копие от файла, чийто първообраз е „подписан“, проверката ще бъде извършена и е възможно да получите ОТРИЦАТЕЛЕН криптографски отговор (**Bad signature**):

```
gpg: Signature made Sun 02 Nov 2014 09:17:57 PM EET
gpg:                using DSA key B4EFB9F38D8AEBF1
gpg: Bad signature from "Trisquel GNU/Linux (Trisquel GNU/Linux signing key)
<trisquel-devel@trisquel.info>" [unknown]
gpg: WARNING: This key is not certified with a trusted signature!
gpg:          There is no indication that the signature belongs to the
owner.
Primary key fingerprint: E6C2 7099 CA21 965B 734A  EA31 B4EF B9F3 8D8A EBF1
```

или ПОЛОЖИТЕЛЕН криптографски отговор (**Good signature**):

```
gpg: Signature made Sun 02 Nov 2014 09:17:57 PM EET
gpg:                using DSA key B4EFB9F38D8AEBF1
gpg: Good signature from "Trisquel GNU/Linux (Trisquel GNU/Linux signing
key) <trisquel-devel@trisquel.info>" [unknown]
gpg: WARNING: This key is not certified with a trusted signature!
gpg:          There is no indication that the signature belongs to the
owner.
Primary key fingerprint: E6C2 7099 CA21 965B 734A  EA31 B4EF B9F3 8D8A EBF1
```

При отрицателен криптографски отговор (**BAD signature**) системата констатира липса на съответствие (или проверяваният от вас файл не е идентичен на „подписания“ първообраз; или не е подписан чрез 'частния' ключ на този,

чийто 'публичен' ключ се сочи в цифровия подпис – т.е. подписан е с друг 'частен' ключ, който по всяка вероятност принадлежи на другиго). При положителен криптографски отговор (**Good signature**) системата установява пълно съответствие между вашето копие от проверявания файл и неговия първообраз, от една страна; и между цифровия подпис и 'частния' ключ, посредством който се твърди да е „положен“ подписът, от друга страна.

И в двата случая обаче системата извежда предупреждение, че този 'публичен' ключ не е потвърден чрез доверени подписи и не съществува потвърждение за това подписът да принадлежи на посочения притежател: (**WARNING: This key is not certified with a trusted signature! There is no indication that the signature belongs to the owner**). Възниква въпрос – дали този 'частен' ключ, чрез който е „положен“ цифровият подпис и съответстващият му 'публичен' ключ, който току що свалихте и ползвахте за криптографска проверка, наистина принадлежат на този, който се сочи като техен притежател и на когото се доверявате?

В Част **VIII** отново ще разгледаме този въпрос, но на този етап е достатъчно да се спрем за малко на понятието 'отпечатък' (**Fingerprint**). Това е **40**-символна буквено-цифрена комбинация, представляваща сложна криптографска производна от шифровъчния ключ, за който се отнася (в горния пример това е комбинацията '**E6C2 7099 CA21 965B 734A EA31 B4EF B9F3 8D8A EBF1**', която системата извежда на последния ред от криптографския отговор като '**Primary key fingerprint**'). 'Отпечатъкът' е уникален за всеки шифровъчен ключ и не може да бъде променен, а теоретично не може и да бъде дублиран. Следователно, ако знаем 'отпечатъка' на определен шифровъчен ключ и намерим начин да получим потвърждение от неговия собственик, че именно това е неговият 'отпечатък', можем да сме сигурни, че въпросният шифровъчен ключ му принадлежи, а от там – че подписът действително е „положен“ от него.

За тази цел трябва да потърсите потвърждение за 'отпечатъка' на интересуващия ви шифровъчен ключ от алтернативен, независим канал, който не е лесно да се компрометира паралелно с канала, от който получавате проверявания подпис. Можете за целта да се обърнете към предоставените ви публично телефони, e-mail адреси или такива за конвенционална поща, или да попитате за потвърждение ваши доверени лица, които от своя страна имат доверен контакт с притежателите на съответния 'частен' ключ. Също така лицата, които ползват криптографски технологии за подписване, обикновено полагат усилия действителните 'отпечатъци' на техните криптографски ключове да са достатъчно добре разпространени в публичното пространство, което затруднява подменянето на техните цифрови подписи с чужди. Следователно – има много възможности да потърсите дори няколко независими един от друг източници, които да потвърдят конкретен 'отпечатък' – особено що се отнася до толкова мащабно и известно в компютърните среди начинание, каквото е създаването и поддържането на една Свободна операционна система.

Както стана дума по-горе, възможно е вместо цифров подпис (или паралелно с него) да бъде предоставена 'проверовъчна сума' (**Check Sum**) за установяване автентичността на инсталационния **.iso** файл (или каквото и да било друго цифрово съдържание). 'Проверовъчните суми' представляват сложни криптографски производни, които са резултат от символите в проверяваните файлове и позволяват установяване на това дали тези файлове са променяни. За разлика от цифровите подписи обаче, 'проверовъчните суми' не изискват при осъществяването на проверка да се прилагат 'публични' ключове – което поставя въпроса за автентичността на конкретна 'проверовъчна сума'. Дали тя наистина произхожда от този, на когото се доверяваме? Този проблем трябва да бъде решаван по начини, до голяма степен аналогични на описаното по-горе във връзка с цифровите подписи. Така например може да се потърси копие от 'проверовъчната сума' чрез независим канал, който да е трудно да се компрометира едновременно с канала, от който получавате самото проверявано цифрово съдържание. Често 'проверовъчните суми' биват „подписани“ цифрово, за да се удостовери тяхната автентичност посредством двойката 'публичен' и 'частен' ключ на техния собственик.

Най-често срещаните стандартизации за 'проверовъчни суми', които се ползват днес, са **MD5** и **SHA512** (втората от които по-сложна и от там – по-надеждна и предпочитана). Да извлечете 'проверовъчната сума' на конкретен файл (в т.ч. на интересуващия ви инсталационен **.iso** файл) можете чрез тази команда:

```
$ Стандартизацияsum Проверяван_файл
```

(където под '**Стандартизация**' разбираме избраната стандартизация за 'проверовъчни суми'; под '**sum**' разбираме указание, че става дума за 'проверовъчна сума'; и под '**Проверяван_файл**' разбираме точното наименование на файла с неговото разширение, ако има такова и адрес, ако терминалът не е преместен на същото място в системата).

Ако искате например да изведете 'проверовъчна сума' по стандартизацията **MD5**, конкретният вид на командата ще бъде '**md5sum Проверяван_файл**', а по стандартизацията **SHA512** – '**sha512sum Проверяван_файл**'. Ще бъде изведен подобен резултат:

```
79011e1dce0cf6d1f3aa206bf8c135d4  Проверяван_файл
```

(където **32**-символната буквено-цифрена комбинация е самата 'проверовъчна сума' (в случая **32**-символната **MD5**), следвана от два интервала и от наименованието на файла, за който се отнася; ако е указан файл на друго място в системата (различно от настоящата ви директория), наименованието на файла ще бъде (трябва да бъде) предхождано от съответния адрес, тъй като в противен случай ще бъде констатирана липсата на проверявания файл и няма да бъде изведен никакъв криптографски отговор).

Можете да осъществите проверка с просто сравняване на предоставената 'проверовъчна сума' и изведената от вас чрез горните команди, или да изискате системата да направи автоматична проверка. Вторият подход ограничава риска от случайни пропуски, които бихте могли да допуснете, докато сравнявате символите ръчно (възможно е например в бързината да не забележите разликата между '**c**' и '**e**', между '**b**' и '**6**' или други подобни).

За да извършите автоматична проверка, трябва да се снабдите с интересуващата ви 'проверовъчна сума' и да я обособите в текстови файл, където след самата сума оставяте два интервала и въвеждате точното наименование на проверявания файл (с неговия „адрес“ преди наименованието му, ако той не се намира на същото място в системата). След това трябва да преместите терминала на съответното място в системата (където се намира файлът с 'проверовъчната сума') и да въведете тази команда:

```
$ Стандартизациясum -c Чексума
```

(където под '**Стандартизация**' разбираме ползваната от вас стандартизация за 'проверовъчни суми'; под '**sum**' разбираме указание, че става дума за 'проверовъчна сума'; под '**Чексума**' разбираме наименованието на файла с чексумата; предшествано от неговия адрес, ако не се намира в настоящата директория, където е отворен терминалът). Ще бъде изведен подобен положителен криптографски отговор:

```
Адрес/Файл: OK
```

... или подобен отрицателен криптографски отговор:

```
Адрес/Файл: FAILED open or read
Стандартизациясum: WARNING: 1 computed checksum did NOT match
```

(**ВНИМАНИЕ: 1 изведена 'проверовъчна сума' НЕ съвпада**). Отрицателен криптографски отговор ще бъде изведен тогава, когато няма пълно тъждество меджу 'проверовъчната сума' и символите от проверявания файл. В случай, че някой от файловете не се намира на изискуемото място в системата (или терминалът не е отворен в изискуемата директория), ще бъде изведен подобен резултат:

```
Стандартизация: Адрес/Файл: No such file or directory
Стандартизациясum: WARNING: 1 listed file could not be read
```

(**ВНИМАНИЕ: 1 посочен файл не може да бъде прочетен**). В случай на резултат като горния проверете дали изписвате правилно наименованията и адресите на файловете (можете да си помагате с бутона **[Tab]**), и дали настоящата директория, в която е отворен терминалът, съответства на въведените адреси.

3. Подготвяне на вашия „жив“ инсталационен носител (класически метод)

След като вече сте сигурни, че вашият **.iso** файл съответства напълно на своя първообраз (следователно вашата Свободна операционна система ще бъде инсталирана точно по начина, по който се очаква да бъде инсталирана), идва

ред да подготвите самия „жив“ носител (**Live USB** или **Live CD/DVD**), способен да стартира компютъра и да задейства инсталационния процес. Ако работите с несвободна операционна система обаче, има риск създаването на вашия „жив“ носител да бъде компрометирано. Поради това отново препоръчваме да ползвате помощен компютър, който вече е инсталиран със Свободен софтуер. Но ако въпреки това предупреждение решите да подготвите вашия „жив“ носител в несвободна софтуерна среда, оставяме това на ваша отговорност.

Можете да пристъпите към подготвянето на вашия „жив“ носител – чрез тази команда (с изискуеми администраторски права):

```
# dd if=Файл.iso of=/dev/Устройство bs=64M
```

... или респективно, ако ползвате програмата **Sudo**:

```
$ sudo dd if=Файл.iso of=/dev/Устройство bs=64M
```

(където под '**Файл.iso**' разбираме сваления от вас инсталационен **.iso** файл, а под '**Устройство**' – **USB**-паметта, която възнамерявате да подготвите като „жив“ носител). За да можете да въведете командата по актуалния за вашия случай начин, трябва да замените '**Устройство**' с наименованието, под което компютърът разпознава вашата **USB**-памет (чрез проследената по-горе команда '**lsblk**'), да откачите нейните дялове (ако има такива) от точките им на закачане (чрез командата '**umount**') и да пристъпите към подготвянето на вашия „жив“ носител. Следва много внимателно да проследите към кое устройство насочвате командата – цялата налична информация на това устройство ще бъде изтрита.

Ако трябва да придадем по-реалистичен вид на командата за създаване на „жив“ носител чрез избраната от вас **USB**-памет (приемаме, че тя е разпозната от системата като '**sdb**'; терминалът е отворен на същото място в системата, където е вашият инсталационен **.iso** файл; и работите в сесия с администраторски права, без командата '**sudo**'), то командата ще придобие подобен вид:

– ако решите да инсталирате **Parabola**:

```
# dd if=parabola-x86_64-systemd-cli-2022.04-netinstall.iso of=/dev/sdb
bs=64M
```

– ако решите да инсталирате **Trisquel**:

```
# dd if=trisquel_11.0_amd64.iso of=/dev/sdb bs=64M
```

– ако решите да инсталирате **Guix**:

```
# dd if=guix-system-install-1.4.0.x86_64-linux.iso of=/dev/sdb bs=64M
```

– ако решите да инсталирате **PureOS**:

```
# dd if=pureos-10.3-gnome-live-20230614_amd64.iso of=/dev/sdb bs=64M
```

Когато процесът по подготовката на „живия“ носител приключи, ще бъде изведен подобен резултат:

```
188+1 records in
188+1 records out
1584398336 bytes (1.6 GB, 1.5 GiB) copied, 358.945 s, 4.4 MB/s
```

Преди да завършите с подготовката на „живия“ носител, е необходимо да гарантирате, че процесът наистина е финализиран успешно. За целта трябва да въведете и тази команда:

```
# sync
```

След като отново бъде изведен индикатор '**#**', можете да сте сигурни, че създаването на „живия“ носител е завършено и вашата **USB**-памет може да се ползва за инсталиране на избраната Свободна операционна система.

Тъй като ползването на „жив“ оптичен диск (**Live CD/DVD**) до голяма степен през последните години намаля значително, ще спестим даването на отделни напътствия по този въпрос. Отбелязваме само, че можете да ползвате софтуер за запис на оптични дискове, при което трябва да укажете, че желаете да създадете инсталационен **`.iso`** диск – останалите стъпки от процеса не разкриват особеност, която да заслужава отделно обсъждане.

Независимо от това коя Свободна операционна система ще изберете, начинът на нейното инсталиране се подчинява до голяма степен на общи принципи. По-долу ще проследим инсталирането на **Parabola**, тъй като е много добре поддържана и това осигурява необходимата стабилност и надеждност на системата. Въпреки трудностите, които създава на начинаещия потребител (при **Parabola** не се предлага „готов пакет“, а се очаква вие сами да настроите системата според вашите предпочитания и нужди), тази Свободна операционна система се отличава с надеждност и бързо отразяване на възникващите проблеми – което е важно за обезпечаване на информационната сигурност. Нашият избор обаче не трябва да предопределя решенията ви. Можете да се запознаете с документацията на която и да било предпочитана от вас Свободна операционна система и на база на следващите напътствия да я инсталирате успешно, адаптирайки казаното от нас където се налага (например при наименованията на асоциираните със съответната Свободна операционна система софтуерни хранилища и съответните софтуерни пакети).

4. Подготвяне на вашия „жив“ инсталационен носител (напредничав метод)

Класическият метод на подготвяне на „жив“ инсталационен носител работи безупречно, но крие някои трудни за отстраняване от неопитен потребител проблеми – свързани например с включването на все още неинсталираната система към internet, за да бъдат изтеглени необходимите за инсталацията софтуерни пакети. Поради тази причина предлагаме и по-напредничав метод, при който няма да сваляме инсталационен **`.iso`** файл, а директно ще създадем инсталационна система на нашата **USB**-памет. Следва обаче да отбележим, че напредничавият метод е достъпен само при ползването на помощен компютър със Свободна операционна система, включен към internet. Следващите напътствия са адаптирани за работа с компютър, инсталиран с операционна система **Parabola** – ако предпочитате друга операционна система, ще трябва да адаптирате напътствията или да прибегнете към класическия метод, който проследихме по-горе в изложението.

Както вече беше проследено, първо и основно трябва да установите под какво наименование компютърът разпознава вашата **USB**-памет и я закача към системата – чрез проследената по-горе команда '**`lsblk`**'; и след това да откачите **USB**-паметта от системата – чрез командата '**`umount`**':

Сега вече можете да започнете създаването на инсталационната система – чрез тази команда:

```
# fdisk /dev/Устройство
```

(където под '**`Устройство`**' разбираме наименованието, под което компютърът разпознава вашата **USB**-памет и я закача към вашата система – например '**`sdb`**', '**`sdc`**' или подобни). Уверете се, че това не е например твърдият диск на компютъра (най-често (но не винаги) разпознаван като '**`sda`**'), за да не изтриете например системата, от която работите!

След като стартирате горната команда по отношение на вашата **USB**-памет, програмата **Fdisk** ще изведе опростен интерфейс, с който можете да си взаимодействате посредством въвеждането на еднобуквени символи (въведете **`[m]`** за допълнителни разяснения). Обособяването на необходимия ви единствен дял за бъдещата инсталационна система ще започне чрез тази команда:

```
# o
```

(при което системата ще изведе резултат за създаването на ново обозначение '**Created a new DOS disklabel**' с обозначен идентификатор).

Самото създаване на новия дял трябва да осъществите така:

```
# n
```

(при което системата ще изведе възможност да изберете дали новосъздаващият се дял да бъде основен ([**P**]**rimary**) или продължаващ ([**E**]**xtended**); трябва да изберете първата възможност:

```
# p
```

(при което системата ще изведе възможност да изберете номер (от **1** до **4**) за новосъздаващия се дял). Въведете **[1]** за първия новосъздаван дял и натиснете **[Enter]**.

След като сторите горното, системата ще изведе подобен резултат:

```
First sector (2048-31250000, default 2048):
```

(където двете числа (разделени със символ '**-**') обозначават съответно първия и последния сектор на **USB**-паметта). Тъй като всеки сектор е **512B**, излиза, че общият размер на вашата **USB**-памет е **6'250'000'000B** (или **16GB**).

От вас се очаква в този момент просто да определите с кой сектор от **USB**-паметта желаете да започва новосъздаващият се дял. Натиснете **[Enter]** и системата ще обозначи по подразбиране като „първи“ сектор първия възможен.

След като сторите горното, системата ще изведе възможност да определите с кой сектор от **USB**-паметта да приключва новосъздаващият се дял. Тъй като имате нужда от един-единствен дял, натиснете отново **[Enter]** и системата ще обозначи по подразбиране като „последен“ сектор последния възможен в края на **USB**-паметта. Ще бъде изведен подобен резултат:

```
Created a new partition 1 of type 'Linux' and of size 16 GiB.
```

Накрая остава да стартирате форматирането на вашата **USB**-памет:

```
# w
```

Имайте предвид, че след въвеждане на горната команда **USB**-паметта ще бъде изтрита. Трябва да сте сигурни, че насочвате командите към правилното устройство и сте задали точно параметрите, които желаете, преди да натиснете **[Enter]**.

Системата ще изведе подобен резултат:

```
The partition table has been altered.
Calling ioct() to re-read partition table.
Syncing disks.
```

След като вашата **USB**-памет вече е организирана по необходимия начин, трябва да пристъпите към форматиране на единствения дял съобразно функциите, които ще изпълнява. Можете да повторите описаната в Част **I** команда '**lsblk**', за да се ориентирате под какво наименование системата разпознава новообразувания дял. Ако **USB**-паметта се разпознава като '**sdb**', най-вероятно новосъзданеният дял се разпознава като '**sdb1**' (но все пак сверете изведените размери, за да сте сигурни, че насочвате следващите команди правилно към **USB**-паметта.

Сега идва ред да форматирате единствения дял на вашата **USB**-памет:

```
# mkfs.ext4 -O^metadata_csum_seed /dev/Физически_дял
```

(където под '**ext4**' разбираме една от основните файлови системи при съвременните **GNU/Linux-libre** системи; под '**-O^metadata_csum_seed**' – новите функционалности, присъщи на **ext4**-файловите системи, които все още не се поддържат от Libreboot софтуера от „ниско“ ниво, но са необходими за стартиране на системата; а под '**Физически_дял**' отново разбираме единствения физически дял на вашата **USB**-памет – например '**sdb1**' или '**sdc1**').

След като приключи форматирането, можете да монтирате новосъздадения единствен дял на **USB**-паметта към директорията **/mnt** в системата на помощния компютър:

```
# mount /dev/Физически_дял /mnt
```

Най-после идва ред да пристъпите и към самото създаване на инсталационната система:

```
# pacman -Sy arch-install-scripts
```

Възможно е докато се изпълнява горната команда, системата да изведе грешка и да откаже да завърши инсталационния процес, поради неактуализирани 'публични' ключове, с които се удостоверява автентичността и интегритетът на необходимите софтуерни пакети – можете да отстраните грешката с проследената по-горе команда '**pacman-key --refresh-keys Адрес_на_неактуализиран_ключ**'.

Когато първоначалната инсталация приключи успешно, идва ред да инсталирате необходимите софтуерни пакети в бъдетата инсталационна система на вашата **USB**-памет – чрез тази команда:

```
# pacstrap /mnt
```

Във връзка с тази команда трябва да напомним командата '**mount /dev/Физически_дял /mnt**', с която преди малко монтирахте единствения дял на вашата **USB**-памет към директорията **/mnt** в системата на помощния компютър. Така сега с командата '**pacstrap /mnt**' ще укажете необходимите софтуерни пакети да бъдат инсталирани именно там – в единствения дял на вашата **USB**-памет, който е монтиран на посоченото място в системата на помощния компютър.

Следват базови настройки на файловата система, първа сред които е генерирането на системния файл **fstab**:

```
# genfstab -U /mnt >> /mnt/etc/fstab
```

В системния файл **/mnt/etc/fstab** е възможно да бъде вписан поради грешка разменният '**swap**' дял от твърдия диск на помощния компютър. Ако това се случи, е необходимо да премахнете този дял. За да се убедите дали има такова погрешно вписване, можете да отворите системния файл **fstab** (например чрез програмата **Nano**):

```
# nano /mnt/etc/fstab
```

... и ако намерите в записите вписан подобен ред (обърнете внимание на израза '**swap**'):

```
UUID=f11e4960-7db9-8be0-49ce-bb67d63f1fb6 none swap defaults 0 0
```

... е необходимо да го премахнете (например като преместите със стрелките **[↓]** и **[↑]** курсора на необходимия ред и премахнете този ред с натискане на **[Ctrl]+[k]**. След като сторите това, можете да запишете така редактирания файл и да излезете от **Nano** с проследените по-горе команди: **[Ctrl]+[o]**, **[Enter]** и след това – **[Ctrl]+[x]**.

Необходимо е след това да укажете ползване по подразбиране на определен базов език от системата. На този етап следва да изберете американската стандартизация на английския език:

```
# echo en_US.UTF-8 UTF-8 >> /mnt/etc/locale.gen
```

... и след това:

```
# echo LANG=en_US.UTF-8 > /mnt/etc/locale.conf
```

По този начин ще укажете на системата да включва по подразбиране англоезичната (**US**) клавиатурна подредба и стандартизацията **UTF-8** (**Unicode Transformation Format** – **8B**). Независимо от езиковите ви предпочитания, на този етап не ви съветваме да включвате други клавиатурни подредби. След като приключите с инсталацията, няма да бъде проблем да въведете каквито допълнителни клавиатурни подредби са необходими.

Остава още една езикова настройка, която трябва да въведете:

```
# echo LANG=en_US.UTF-8 > /etc/locale.conf
```

За да ускорите предстоящите инсталационни процеси, можете да отворите още един системен файл (може би с **Nano**):

```
# nano /mnt/etc/pacman.conf
```

... и да премахнете символа '**#**' от началото на този ред надолу в текста:

```
#ParallelDownloads = 5
```

... който след промяната придобива такъв вид:

```
ParallelDownloads = 5
```

Така ще ускорите инсталационния процес, тъй като ще можете да изтегляте паралелно по няколко софтуерни пакета от множество софтуерни хранилища, вместо само последователно един по един.

Идва ред да се впишете като администратор на директорията, към която в настоящия момент е закачена вашата **USB**-памет – чрез тази команда:

```
# arch-chroot /mnt/
```

Във връзка с тази команда трябва да напомним командата '**`mount /dev/Физически_дял /mnt`**', с която монтирахте единствения дял на вашата **USB**-памет към директорията **`/mnt`** в системата на помощния компютър и командата '**`pacstrap /mnt`**', с която указахте необходимите софтуерни пакети да бъдат инсталирани именно там. С въведената току що команда ще се впишете като администратор на новосъздаващата се инсталационна система на вашата **USB**-памет и ще можете да подавате команди към нея като администратор на тази именно система (вместо на системата на помощния компютър, от който работите).

След като вече сте вписани като администратор на новосъздаващата се инсталационна система на вашата **USB**-памет, идва ред да укажете да бъдат създадени основните компоненти на системното ядро на тази система:

```
# locale-gen
```

Ако няма грешки, обозначени с '**ERROR**', можете да считате, че инсталационната система е създадена успешно.

За да може инсталационната система да работи, включително на компютри с все още несвободен софтуер от „ниско“ ниво, е необходимо да инсталирате един основен набор от софтуерни пакети. Следва да инсталирате самото системно ядро и няколко базови софтуерни модула, които ще ви позволят да работите с вашата инсталационна система:

```
# pacman -S linux-libre-lts linux-libre-firmware grub \
cryptsetup lvm2 networkmanager nano bash-completion \
arch-install-scripts
```

(където със символа '****' указваме, че следващият ред от командата продължава на същия ред в терминала; можете да пропуснете този символ и да въведете горните софтуерни пакети на един ред, преди да натиснете **[Enter]**; но ако въведете командата по дадения по-горе начин, благодарение на символа '****' ще бъде интерпретирана като въведена на един единствен ред).

Възможно е в процеса на инсталирането системата да изведе въпрос за това коя от няколкото достъпни и взаимозаменяеми версии на определен софтуерен пакет предпочитате да бъде инсталирана. В такъв случай отделните версии ще бъдат номерирани и от вас се очаква да въведете предпочитания номер, след което да натиснете **[Enter]**. В повечето случаи първата предложена алтернатива е за предпочитане.

След като основните софтуерни пакети бъдат инсталирани, трябва да пристъпите към стартиращата програма **GRUB** (**GRand Unified Bootloader**), която позволява да стартирате операционната система:

```
# grub-install --recheck /dev/Устройство
```

(където под '**Устройство**' разбираме наименованието, под което системата разпознава вашата **USB**-памет, примерно '**sdb**'); след това:

```
# mkdir /boot/grub
```

... и накрая:

```
# grub-mkconfig -o /boot/grub/grub.cfg
```

Непременно преди да продължите трябва да въведете и администраторска парола – чрез тази команда:

```
# passwd
```

Ако не сторите това, след рестартиране на **USB**-паметта няма да можете да се впишете като потребител с администраторски права и новосъздадената инсталационна система ще остане неизползваема.

След като инсталирането на посочените софтуерни пакети приключи и сте въвели администраторска парола, можете да излезете от административния режим:

```
# exit
```

... и да разкачите вашата **USB**-памет от системата с описаните по-горе команди '**umount**' и '**udisksctl power-off -b /dev/Устройство**', и да откачите физически външния „жив“ носител.

5. Инсталиране на Свободна операционна система

Следващите напътствия са за инсталиране от „жив“ и са подходящ при системи, които все още не са програмирани със Свободен софтуер от „ниско“ ниво (все още работят с **BIOS/UEFI**). Следва в тази връзка да отбележим, че информационната сигурност изисква инсталирането на една Свободна операционна система да бъде предшествано от цялостно шифроване на вашето устройство. Тъй като това изисква допълнителни знания и умения, сме „разместили“

реда, по който ви запознаваме с отделните операции. Цялостното шифроване ще разгледаме в Част **IV**, след като вече сме проследили един обикновен инсталационен процес и сме усвоили някои основни моменти, които ще ни бъдат полезни за следващия етап. Нещо повече – никое цялостно шифроване не може да се изпълни по сигурен начин, ако преди това контролът върху компютъра не е поверен на Свободна операционна система, обезпечена със Свободен софтуер от „ниско“ ниво (последното ще проследим в Част **III**). Подобно обезпечаване от своя страна не може да се осъществи надеждно, ако контролът върху вашето устройство не е „в ръцете“ на Свободна операционна система, стартирана от „жив“ носител. Ето защо, след като проследим как става това, ще можем да преминем към изграждането на сигурна софтуерна среда, в която да осъществим надеждно цялостно шифроване и едва след това ще бъдем готови да инсталираме нашата напълно шифрована операционна система. Въпреки това „разместване“ обаче, придобитите знания и умения ще ни помогнат за успешно справяне със следващите стъпки от процеса – затова настояваме именно на този ред при представяне на интересуващите ни въпроси.

За да стартирате инсталирането на Свободна операционна система, е необходимо да включите физически вашия „жив“ носител към компютъра и да рестартирате. Ако всичко е наред, ще започне процес по зареждане от „живия“ носител (това може да отнеме малко повече време, тъй като прехвърлянето на данни от **USB**-носител води до известно забавяне). В зависимост от операционната система и вида на „живия“ носител, тук може бъде изведен интерфейс за инсталиране на новата операционна система (или тя направо ще се зареди на компютъра и да позволи да я ползвате директно от „живия“ носител, още преди да е инсталирана – макар и малко по-бавно от обичайното).

Не винаги обаче нещата са толкова прости – особено при по-новите дистрибуции несвободни операционни системи и при по-новите модели компютри. При тях все по-често се въвеждат технически ограничения срещу евентуалните ви опити да стартирате софтуер, различен от посочения от производителя; или поради усложняването на софтуера се допускат несъвместимости и конфликти, препятстващи ползването на желания от вас софтуер.

При по-свободните компютри в повечето случаи можете да стартирате „живия“ носител, след като влезете в т.нар. стартово меню (**Boot Menu**) на **BIOS/UEFI** софтуера от „ниско“ ниво и укажете, че желаете да стартирате компютъра от **USB**-портовете (или от **CD/DVD**-устройството). Обикновено в стартовото меню се влиза, като в самото начало при включването на компютъра (в зависимост от неговата марка и модел) започнете да натискате през съвсем кратки интервали бутона **[Esc]**, или бутона **[Del]**, или някой от функционалните бутони **[F1]**, **[F2]**, **[F3]** и т.н. Ако не уцелите от първия път, рестартирайте отново (можете да го направите бързо, като натиснете едновременно **[Ctrl]**+**[Alt]**+**[Del]**) и пробвайте с някой друг от функционалните бутони.

Съответният функционален бутон прекъсва обичайния процес по зареждане на системата и вместо това на монитора се появява ограничен графичен интерфейс, в който можете да оперирате посредством стрелките на клавиатурата **[↓]** и **[↑]**, и още няколко клавиша, за които ще намерите указания в самия интерфейс (най-често това са **[Enter]**, **[Backspace]**, **[+]**, **[-]** и **[Space]**). Там трябва да посочите, че желаете системата да стартира от **USB**-портовете (или от **CD/DVD**-устройството). Ако в **USB**-портовете има закачена **USB**-памет, която е подготвена като „жив“ носител (или респективно – в **CD/DVD**-устройството има поставен „жив“ оптичен диск), системата ще зареди намиращия се в този „жив“ носител стартиращ файл и ще започне да зарежда вашата Свободна операционна система.

Понякога обаче няма стартиращо меню (или достъпът до него е забранен), или въпреки вашите указания системата отново стартира по обичайния начин и отказва да зареди „живия“ носител. В такъв случай е необходимо да влезете в **BIOS/UEFI** (програмното осигуряване на компютъра от „ниско“ ниво, което управлява хардуера и „посредничи“ между него и вашата операционна система), за да укажете на системата да разреши стартиране (**Boot**) на компютъра от **USB**-портовете (или от **CD/DVD**-устройството). Обикновено **BIOS/UEFI** се достига, като в самото начало при включването на компютъра (в зависимост от неговата марка и модел) започнете да натискате през съвсем кратки интервали или бутона **[Esc]**, или бутона **[Del]**, или някой от функционалните бутони **[F1]**, **[F2]**, **[F3]** и т.н. – подобно на начина, по който преди малко достъпвахме стартовото меню. **BIOS/UEFI** също има ограничен графичен интерфейс, в който можете да оперирате посредством стрелките на клавиатурата и още няколко клавиша (за които ще намерите указания в самия интерфейс).

BIOS/UEFI програмното осигуряване от „ниско“ ниво може да бъде различно в зависимост от марката и модела на компютъра ви, поради което не можем да дадем конкретни напътствия. Но така или иначе трябва да разрешите (**Enable**) стартирането (**Boot**) от вашите **USB**-портове (или от **CD/DVD**-устройство) и да изключите (**Disable**) функцията

Secure Boot (за „сигурно“ стартиране), която ограничава възможностите за стартиране от външни устройства (ако има включена такава функция във вашия случай). За да не позволите несвободната ви операционна система (ако имате инсталирана такава) да се стартира преди всичко останало и да забрани стартирането на каквото и да било друго след себе си (**Windows** например прави това винаги, когато има подобна възможност), трябва да промените и приоритета на стартиране (потърсете функция от рода на '**Change order of booting devices**' или '**Boot devices priority**'), като заложите компютърът да стартира първо от **USB**-портовете (или от **CD/DVD**-устройството). Ако тази настройка не помага, можете дори да изключите (**Disable**) възможността за стартиране от твърдия диск (**HDD**) или дори от всяко друго устройство с изключение на **USB**-портовете (или **CD/DVD**-устройството). Завършете работата си в **BIOS/UEFI**, като потърсите функция за запомняне на промените и излизане (**Save changes and exit**) и отново пробвайте да стартирате „живия“ носител. Ако всичко е наред, при стартиране на системата вашето **BIOS/UEFI** програмно осигуряване ще потърси стартираща програма (**Boot-loader**) най-напред в закачените устройства на вашите **USB**-портовете (или в **CD/DVD**-устройството). По този начин ще бъде стартиран „живият“ носител и той ще поеме контрола върху компютъра, позволявайки ви да пристъпите към инсталиране на вашата Свободна операционна система.

Някои компютри са проектирани специално по начин, който изобщо да не позволи стартирането на софтуер, различен от определения от производителя. Това се прави най-често поради картелни споразумения с компаниите-собственици на несвободен софтуер, насочени към това потребителите да се задължат да ползват само и единствено техните продукти. До неотдавна поради същата причина компютрите на **Apple** изобщо не можеха да бъдат стартирани по какъвто и да било друг начин, освен със софтуерно обезпечение от **Apple** (каквото е положението и до днес например с телефоните **iPhone**). Възможно е в този ред на мисли вашият компютър изобщо да откаже да стартира „живия“ носител или да го стартира по ограничен начин, който не позволява включване на пълната му функционалност (най-често проблем възниква с компонентите, отговарящи за свързването с internet и в по-малка степен за обработката на звук и видео). В някои от случаите това може да се преодолее от опитен компютърен специалист, но в други случаи до момента нямаме технологично решение за преодоляване на ограниченията и би следвало да се откажем от подобно хардуерно оборудване.

При всички случаи гарантирането на сигурността е несъвместимо с инсталирането на какъвто и да било несвободен софтуер (**Firmware**) за хардуерните компоненти, за които няма свободни софтуерни решения. Подобни несвободни хардуерни компоненти (ведно с несвободния софтуер, необходим за тяхното функциониране) представляват риск за вашата информационна сигурност, тъй като за тях нито можете да знаете какво точно извършват (това е „търговска тайна“ на компанията-собственик), нито можете да промените начина, по който функционират (ако този начин е неприемлив за вас и желаете да го подобрите). Подобен риск не е оправдано да се поема. Ако някой хардуерен компонент не работи със свободен софтуер, е препоръчително да се откажете от него или дори ако се наложи да замените целия компютър с друг, съвместим със свободни технологии. За много потребители това може да изглежда радикално, но в действителност няма как да гарантирате вашата информационна сигурност, ако разчитате на оборудване, което ограничава свободата и ви задължава да ползвате несвободни технологии – за които дори нямате право да знаете какво точно извършват във вашата система! Следователно няма как с подобно оборудване да си гарантирате каквато и да било сигурност (освен може би сигурността на компанията-собственик, че нещата се случват по начин, определен от нея и укриван от вас като „търговска тайна“).

След като „живият“ носител най-после бъде стартиран на вашия компютър, можете да започнете инсталационния процес – от графичния интерфейс или от командния ред. При положение, че сте проверили автентичността на „живия“ носител чрез криптографски способи и на този етап е твърде трудно да се реализира пробив във вашата система, няма проблем да осъществите инсталацията през графичния интерфейс (ако има предложен такъв). Това обаче ограничава вашите възможности и въпреки, че засега няма риск за сигурността, ви насърчаваме все пак да изберете командния ред (ако е предложена такава възможност). Това ще позволи да направите някои по-фини настройки, които изобщо не се предвиждат в графичния интерфейс; още повече, че когато пристъпите в Част **`IV`** към цялостно шифроване на системата, ще трябва да проведете инсталационния процес изключително в командния ред.

След като сте стартирали „живия“ носител, трябва да влезете в терминал. При някои инсталационни **`.iso`** файлове изобщо не се предлага друга възможност (например при **Parabola** се зарежда терминал и за вас остава сами да инсталирате графична среда, ако желаете да ползвате такава). При други **`.iso`** файлове (например при **Trisquel**) по подразбиране се зарежда графична среда и в нея трябва да потърсите терминала, ако желаете да ползвате командния ред.

– при някои системи по подразбиране се зарежда терминал – в такава ситуация достъпът се осъществява по подразбиране с администраторски права (като в някои случаи трябва да се впишете като потребител '**root**' и да въведете парола '**root**'; или да се впишете с потребител и парола, въведени от вас на по-ранен етап);

– при други системи по подразбиране се зарежда графичната среда – при тях често може да достъпите терминал през менютата на графичната среда; в терминала, ако не сте в сесия с администраторски права на '**root**' потребителя, може да се наложи да го активирате чрез командата '**su**' и въвеждане на парола '**root**';

Когато вече сте отворили терминал (или той се е заредил по подразбиране), най-напред трябва да подсигурите internet-свързаност, която ще ви бъде необходима за следващите стъпки. Най-сигурният и безпроблемен начин е да свържете вашия компютър към рутер чрез помоща на **LAN**-кабел при което свързването към internet ще бъде установено автоматично.

За безжично свърване с internet чрез **WiFi**, тази възможност е силно зависима от избраната от вас „жива“ система – като при някои това може да се окаже твърде трудно. Ако компютърът работи с несвободно безжично **WiFi** устройство, което изисква за функционирането си несвободен софтуер, може би ще откаже да работи при стартиране от „жив“ носител със Свободна операционна система.

Може да пробвате да свържете компютъра безжично към internet чрез тази команда:

```
# nmcli dev wifi connect Мрежа password Парола
```

(където под '**Мрежа**' разбираме наименованието на предпочитаната от вас безжична мрежа, а под '**Парола**' – паролата за достъпване на мрежата, ако има въведена такава). Ако безжичното свързване към internet не проработи от няколко опита, ще се наложи да свържете компютъра към internet чрез **LAN**-кабел.

Можете да проверите дали компютърът е свързан към internet:

```
# ping Сървър
```

(където под '**Сървър**' разбираме адреса на предпочитан от вас сървър – например този на Фондацията за Свободен софтуер '**fsf.org**'). При наличие на internet-свързаност, системата ще започне да извежда подобни резултати:

```
PING fsf.org (209.51.188.174) 56(84) bytes of data.
64 bytes from www.fsf.org (209.51.188.174): icmp_seq=1 ttl=54 time=121 ms
64 bytes from www.fsf.org (209.51.188.174): icmp_seq=2 ttl=54 time=119 ms
64 bytes from www.fsf.org (209.51.188.174): icmp_seq=3 ttl=54 time=119 ms
```

Прекъснете процеса (като натиснете **[Ctrl]+[c]**) и продължете нататък с инсталацията.

След като сте подсигурили internet-свързаност, следва да разделите твърдия диск на обособени дялове (**Partitions**), в които ще бъде разпределена операционната система. В Част **I** представихме файловата структура на една Свободна операционна система. Тази файлова структура може да бъде разпределена по различни начини върху твърдия диск и това зависи преди всичко от функциите, за които ще ползвате системата. Едно класическо разпределение, подходящо за нашите цели, включва обособяването на разменен '**swap**' дял, системен '**/**' дял и още един дял с потребителското съдържание върху останалата част от твърдия диск, който засега ще наречем '**data**' (няма пречка да дадете на този дял и друго наименование, стига да не го дублирате някое от съществуващите наименования във файловата структура на **GNU/Linux-libre**, разположена в Кореновата директория на системата).

Разменният '**swap**' дял представлява буферно пространство за подпомагане на изчислителните процеси, осъществявани от системата. Неговият размер зависи от общия размер на **RAM**-чиповете на компютъра и е оптимално да бъде двойно по-голям. Ако вашият компютър разполага например с **4GB RAM**, за разменния '**swap**' дял е добре да отделите **8GB** пространство. Ако обаче разполагате с повече **RAM** (например **8GB** и повече), не е необходимо да

отделяте двоен размер пространство – тогава би било достатъчно да отделите толкова, колкото е вашата **RAM**. Благодарение на разменния '**swap**' дял **GNU/Linux-libre** системите се отличават с особена стабилност, което ги прави предпочитани не само за обезпечаване на сигурността, но така също – за поддържане на непрекъсваеми и критично важни процеси; не случайно почти всичките сървъри в света и най-мощните супер-компютри са **GNU/Linux**-базирани.

Системният дял съдържа самата система (нейното ядро, софтуерни библиотеки, потребителски приложения) и се обозначава със символа '**/**'. За разлика от други операционни системи, при **GNU/Linux** цялата система започва именно от тук. Поради критичната важност на този дял, препоръчваме да го поставите след разменния '**swap**' дял – така при инцидент с твърдия диск намалява вероятността да бъде засегната системата. В зависимост от това какъв софтуер възнамерявате да ползвате, за системния '**/**' дял в повечето случаи са достатъчни до **20-30GB** пространство. Ако работите със сложен и специализиран софтуер, можете да предвидите и повече пространство, но в повечето случаи посоченото е достатъчно. Обособяването на отделен дял, предназначен за системата, дава възможност компютърът да бъде преинсталиран, без това да засегне потребителското съдържание. Това е особено полезно, ако се наложи инцидентно изтриване на целия софтуер (например защото системата е компрометирана). Обикновено това се случва без предупреждение и за потребителя не остава възможност да изтегли своето съдържание, ако преди това не го е разположил в обособен за целта дял от твърдия диск.

И накрая – потребителският '**data**' дял може да заеме останалото свободно пространство на твърдия диск. Както стана дума, този дял ще служи за съхраняване на вашето потребителско съдържание, отделно от цялата система. Обособяването на такъв дял позволява инцидентно преинсталиране на системата (например при технически срив или компрометиране), без това да засегне по какъвто и да било начин вашето съдържание. След като приключите с преинсталирането и стартирате компютъра, вашите файлове ще стоят непокътнати на своите местата в обособения потребителски дял. Отделно – обособяването на потребителски '**data**' дял позволява обезпечаване на неговата сигурност по начин, който не е свързан пряко с останалата част от вашата система. Това е още една причина да препоръчаме вашето потребителско съдържание да бъде разположено в обособен дял, който е извън самата система.

Във връзка с горното не препоръчваме да държите потребителско съдържание в директорията '**/home**' (не съвсем сполучливо сравнявана например с директорията '**\My Documents**' при **Windows**), защото евентуална инцидентна необходимост да преинсталирате системата може да засегне съдържанието в тази директория. В тази връзка съществува принципна възможност да обособите на твърдия диск отделен дял '**home**' за директорията '**/home**', но дори при това не препоръчваме да ползвате тази директория като хранилище за вашето съдържание. При компрометиране на системата е възможно запазването на директорията '**/home**' да доведе до запазване и на някои от компрометираните файлове (в т.ч. огромен брой системни и трудно проследими файлове, които няма как да познавате и проверите изцяло – защото в '**/home**' освен всичко друго, се съхраняват повечето от вашите настройки и някои от допълнително инсталираните програми). Ето защо препоръчваме да отделите недвусмислено вашето съдържание от дяловете, в които се разполагат каквито и да било системни файлове – и съответно да изтриете цялата система, ако се наложи, без това да доведе до засягане на съдържанието ви. Ако желаете да запазите някои специфични настройки, можете да направите резервни копия от техните файлове в директорията '**/home**' и в случай на необходимост да ги копирате от предварително създаденото резервно копие обратно в '**/home**', след като преинсталирате системата.

В Част **IV** ще разгледаме възможността да обособите още един дял, който да шифровате допълнително и да ползвате за съхраняване на поверително съдържание. Това ще позволи да разполагате с пространство на вашия компютър, което продължава да стои шифровано, дори и когато системата е отворена и вие работите с нея. Това шифровано пространство можете да отваряте само в случай на необходимост, като преди това сте се оттеглили на сигурно място и сте преустановили всякаква свързаност с internet – за да ограничите рисковете от случайно или преднамерено проникване в поверителното съдържание. На този етап обаче няма да създаваме такъв допълнителен дял и ще се задоволим с горната опростена структура, която ни е необходима за едно обикновено инсталиране на Свободна операционна система. Тук е достатъчно да отбележим, че такива дялове е възможно да се създадат – с оглед организирането на вашата информационна сигурност като слоеве със сегментиран достъп.

По-горе (докато разглеждахме особеностите на командния ред в Част **I**) проследихме две основни понятия във връзка със закачането на устройства към системата: файлова система (**File System**) и точка на закачане (**Mounting Point**). Както стана дума – файлова система е начинът, по който указваме да се форматира съдържанието в определено

устройство (включително в обособен дял от твърдия диск); а точка на закачане е мястото, на което указваме съответното устройство (или обособен дял) да се закача към системата и да се достъпва софтуерно от нея.

В нашия случай са ни необходими следните точки на закачане и следните файлови системи:

– за разменния '**swap**' дял – точка на закачане '**swap**' (при това положение файловата система задължително е **swap** (което позволява подпомагане на изчислителните процеси) и обикновено няма възможност за отделното ѝ избиране);

– за системния '**/**' дял – точка на закачане '**/**' (което определя този дял като Коренова директория) и файлова система '**ext4**' (което позволява записване и достъпване на съдържание от **GNU/Linux-libre** базирани системи);

– за потребителския '**data**' дял – точка на закачане '**/data**' (или друго предпочитано от вас наименование, което ще въведете ръчно и което не се дублира с някое от съществуващите наименования във файловата структура на Кореновата директория) на **GNU/Linux-libre**) и файлова система '**ext4**'.

	/	**/data**
swap	**Ext4**	**Ext4**
8GB	**30GB**	**останалото пространство на твърдия диск**

Твърд диск с обособени основни дялове на системата

Организирането на необходимите ни дялове върху твърдия диск може да осъществите чрез програмата **Fdisk** (ако не е вградена по подразбиране във вашия „жив“ носител, инсталирайте с командата '**pacman -S util-linux**'). Можете да започнете да организирате необходимите ви дялове чрез тази команда:

```
# fdisk /dev/Устройство
```

(където под '**Устройство**' разбираме наименованието, под което вашата система разпознава твърдия диск – в повечето случаи нещо от типа на '**sda**'). В случай, че не сте сигурни под какво наименование системата разпознава вашия твърд диск, можете с проследената в Част **I** команда '**lsblk**' да се ориентирате по размерите на отчетените устройства. Внимавайте какво правите, тъй като следващите команди ще форматират указаното от вас устройство, изтривайки наличната на него информация!

Докато разглеждахме напредничавия метод за създаване на инсталационна система на външен „жив“ носител, стана дума за програмата **Fdisk**. При въвеждане на горната команда ще бъде изведен опростения интерфейс, в който можете да въвеждате еднобуквени символи (въведете **[m]** за допълнителни разяснения).

Както стана дума, необходимо е да обособите три дяла (разменен дял '**swap**', коренов дял '**/**' и потребителски дял '**data**'). Обособяването ще започне чрез тази команда:

```
# o
```

(при което системата ще изведе резултат за създаването на ново обозначение '**Created a new DOS disklabel**' с обозначен идентификатор).

Самото създаване на всеки от новите дялове трябва да осъществите чрез тази команда:

`# n`

(при което системата ще изведе възможност да изберете дали новосъздаващият се дял да бъде основен (**Primary**) или продължаващ (**Extended**); вие трябва да изберете първата възможност:

`# p`

(при което системата ще изведе възможност да изберете номер (от **1** до **4**) на новосъздаващия се дял). Въведете **`[1]`** за първия новосъздаван дял и натиснете **`[Enter]`**.

Системата ще изведе подобен резултат:

`First sector (2048-195314548, default 2048):`

(където двете числа (разделени със символ '**`-`**') обозначават съответно първия и последния сектор на твърд диск). Тъй като всеки сектор е **512B**, излиза, че общият размер на твърдия диск е **100'000'000'000B** (или **100GB**).

Можете да се ориентирате на колко **GB** съответстват изведените по-горе стойности чрез тази формула:

`( Последен_сектор – Първи_сектор ) * 512 / 1000000000 = GB`

(където под '**`Последен_сектор`**' разбираме второто число, а под '**`Първи_сектор`**' – първото число от изведения резултат; съгласно което горната формула би придобила следните числови стойности:

`( 195314548 – 2048 ) * 512 / 1000000000 = 100 [GB]`

От вас се очаква в този момент просто да определите с кой сектор от твърдия диск желаете да започва новосъздаващият се дял. Натиснете **`[Enter]`** и системата ще обозначи по подразбиране като „първи“ първия възможен сектор.

Системата ще изведе възможност да определите с кой сектор от твърдия диск да приключва новосъздаващият се дял. Тук трябва да въведете такова число, с което да обособите необходимото ви пространство за новосъздаващия се дял.

– ако създавате разменния '**`swap`**' дял и сте решили той да бъде **8GB**, можете да укажете това чрез тази команда:

`# +8G`

– ако създавате системния '**`/`**' дял и сте решили той да бъде **30GB**, можете да укажете това чрез тази команда:

`# +30G`

(където под '**`8G`**' или '**`30G`**' разбираме съкратено съответно '**`8GB`**' или '**`30GB`**'). Остава да натиснете повторно **`[Enter]`** и указанието ви ще бъде изпълнено, като системата автоматично ще определи за последен сектор на новосъздаващия се дял този, който съответства най-точно на указания размер (с възможни незначителни отклонения). Ще бъде изведен подобен резултат:

`Created a new partition 1 of type 'Linux' and of size 8 GiB.`

... или респективно:

`Created a new partition 1 of type 'Linux' and of size 30 GiB.`

Трябва да повторите горната процедура, като започнете от командите '**n**' и '**p**', зададете следващи номера на новосъздаващите се дялове и определите начални и крайни сектори от твърдия диск, върху които да бъдат разположени и останалите дялове на вашата система – разменния '**swap**' дял, системния '**/**' дял и потребителския '**data**' дял (в случай, че възприемете предложеното от нас разпределение).

Накрая, когато желаете потребителският '**data**' дял да запълни цялото останало пространство (без значение какъв е точният му размер), можете просто да натиснете два пъти **[Enter]** при задаване като „първи“ първия възможен сектор и като „последен“ последния възможен сектор от оставащото пространство на твърдия диск. Така ще обхванете цялото останало свободно пространство.

Когато параметрите на всички необходими дялове са зададени, можете да стартирате форматирането на твърдия диск:

```
# w
```

Имайте предвид, че след въвеждането на горната команда твърдият диск ще бъде изтрит и след това разпределен в указаните дялове. Трябва да сте сигурни, че насочвате командите си към правилното устройство и сте задали точно параметрите, които желаете, преди да натиснете **[Enter]**!

Системата ще изведе подобен резултат:

```
The partition table has been altered.
Calling ioct() to re-read partition table.
Syncing disks.
```

След като вашият твърд диск вече е разпределен в необходимите ви дялове, трябва да пристъпите към форматиране със съответния вид файлова система и към закачане в съответната точка на закачане към системата за всеки един от тези дялове (в зависимост от неговите специфични функции). За да се ориентирате правилно при насочването на следващите команди, можете чрез проследената в Част **I** команда '**lsblk**' да проследите под какви наименования системата разпознава новообразуваните дялове (ако твърдият диск се разпознава като '**sda**', най-вероятно новосъздадените дялове се разпознават като '**sda1**', '**sda2**' и '**sda3**'). Ще бъде изведен подобен резултат:

```
NAME      MAJ:MIN RM   SIZE RO TYPE  MOUNTPOINT
sda         8:0    0   100G  0 disk
├─sda1      8:1    0     8G  0 part
├─sda2      8:2    0    30G  0 part
└─sda3      8:3    0    62G  0 part
```

В дадения пример разполагаме с твърд диск с капацитет **100GB**, разпределен в три дяла. Първият е предвиден за разменния '**swap**' дял с **8GB**; вторият – за системния '**/**' дял с **30GB**; и третият – за потребителския '**data**' дял с останалото пространство на твърдия диск (в случая – **62GB**).

На първо място трябва да укажете на системата кой дял да интерпретира и ползва като разменен '**swap**' дял:

```
# mkswap /dev/Разменен_дял
```

(където под '**Разменен_дял**' разбираме наименованието, под което системата разпознава съответния дял – примерно '**sda1**'). За разлика от останалите дялове на системата, разменните '**swap**' дялове не се форматират с определена файлова система и не се ориентират към конкретна точка на закачане. Вместо това трябва да активирате съответния дял за осъществяване на неговите помощни функции:

```
# swapon /dev/Разменен_дял
```

Една от основните файлови системи, която се ползва в съвременните **GNU/Linux** системи, е **Ext4** (за разлика например на ползваната от **Microsoft** файлова система **NTFS** или всеобщо признатата файлова система **FAT**).

Дяловете, предназначени за съхраняване на файлове, е препоръчително да бъдат форматирани именно с файловата система **Ext4**. Можете да осъществите такова форматиране за системния '**/**' дял така:

```
# mkfs.ext4 -O^metadata_csum_seed /dev/Системен_дял
```

(където под '**Системнен_дял**' разбираме наименованието, под което системата разпознава съответния дял – примерно '**sda2**').

След като системният дял бъде форматиран (ще отнеме няколко секунди), следва да определите неговата точка на закачане към системата, стартирана от външния „жив“ носител:

```
# mount /dev/Системен_дял /mnt
```

С изпълнението на тази команда позволявате последващите инсталационни дейности да бъдат извършени в кореновата '**/**' директория на инсталиращата се система, която докато тече инсталационният процес, ще стои закачена към директорията '**/mnt**' на помощната система (стартирана от външния „жив“ носител).

Накрая идва ред да форматирате и потребителския '**data**' дял, който по-горе обособихме за съхраняване на потребителско съдържание. Тъй като този дял е създаден допълнително и в оригиналната файлова система на **GNU/Linux-libre** системите не е предвидено по подразбиране място за него, трябва да създадете директория, която след това ще определите като точка на закачане за този дял. Можете да създадете необходимата директория чрез тази команда:

```
# mkdir /mnt/Потребителска_директория
```

(където под '**Потребителска_директория**' разбираме наименованието, избрано от вас за съответната директория (примерно '**data**'); в зависимост от предпочитанията ви можете да изберете и друго наименование, стига то да не дублира наименованията на други директории от Кореновата директория на системата, като препоръчваме да не въвеждате никаква специфична информация, интервали или други специални символи (освен '**-**' и '**_**') и букви, различни от **US**-стандартизацията на латиницата).

Обърнете внимание и на това, че адресираме новосъздаващата се потребителска директория в директорията '**/mnt**', а не в Кореновата '**/**' директория, където определихме, че ще желаем да бъде. Това е така, защото в момента работим от външен „жив“ носител, към чиято директория '**/mnt**' преди малко закачихме Кореновата '**/**' директория на новоинсталиращата се система. След като приключим с инсталационния процес и премахнем външния „жив“ носител, всичко извършено в директорията '**/mnt**' на този външен „жив“ носител ще изглежда извършено в Кореновата '**/**' директория на новоинсталираната система.

След като директорията на вашия потребителски '**data**' дял вече е създадена (в директорията '**/mnt**' на външния „жив“ носител), трябва да форматирате този дял с файлова система, подходяща за съхраняване на файлове. Аналогично на проследеното за системния '**/**' дял, тук също приложение намира файловата система **Ext4** – чрез тази команда:

```
# mkfs.ext4 /dev/Потребителски_дял
```

(където под '**Потребителски_дял**' разбираме наименованието, под което системата разпознава съответния дял – примерно '**sda3**').

След като директорията на потребителския '**data**' дял е създадена и той самият вече е форматиран, остава да укажете точката му на закачане към системата в новосъздадената директория – чрез тази команда:

```
# mount /dev/Потребителски_дял /mnt/Потребителска_директория
```

(което с дадените по-горе примери ще приеме подобен вид: '**mount /dev/sda3 /mnt/data**').

Ако повторите проследената в Част **I** команда '**lsblk**', ще бъде изведен подобен резултат:

```
NAME      MAJ:MIN RM   SIZE RO TYPE  MOUNTPOINT
sda         8:0    0   100G  0 disk
├─sda1      8:1    0     8G  0 part  [SWAP]
├─sda2      8:2    0    30G  0 part  /
└─sda3      8:3    0    62G  0 part  /data
```

След като архитектурата на твърдия диск е изградена, идва ред да инсталирате основните софтуерни пакети, които ще подготвят основите на системата – чрез тази команда:

```
# pacstrap /mnt
```

Отново следва да обърнете внимание на адресирането на горната команда към директорията '**/mnt**' на външния „жив“ носител: докато водим инсталационния процес с указана точка на закачане за Кореновата '**/**' директория на новоинсталиращата се система към директорията '**/mnt**' на „живия“ носител, тази директория ще бъде разпознавана като Коренова за новоинсталиращата се система.

Ако в процеса на инсталирането се установи необновен шифровъчен ключ, с който е удостоверена автентичността и интегритета на някой от софтуерните модули, обновете този ключ с проследената в Част **II** команда '**pacman-key --refresh-keys Електронен_пощенски_адрес**' след това опитайте повторно с командата '**pacstrap**'.

След като основният набор от софтуерни пакети бъде инсталиран, трябва да укажете системата да се зарежда при всяко следващо стартиране на компютъра съгласно файловите системи и точките на закачане, които настроихте по-горе за всеки един от логическите дялове. Първо трябва да генерирате системния файл **fstab**:

```
# genfstab -U /mnt >> /mnt/etc/fstab
```

Необходимо е след това да укажете ползване по подразбиране на определен базов език от системата. На този етап следва да изберете американската стандартизация на английския език:

```
# echo en_US.UTF-8 UTF-8 >> /mnt/etc/locale.gen
```

... и след това:

```
# echo LANG=en_US.UTF-8 > /mnt/etc/locale.conf
```

Независимо от езиковите ви предпочитания, на системно ниво не препоръчваме да избирате други езикови стандартизации. Няма пречка на по-късен етап да добавите желаните от вас езици (клавиатурни подредби, превод на графичния интерфейс и пр.), но сега подобни настройки могат да объркат инсталационния процес.

Следват настройки на основните параметри на системата. Най-напред трябва да се впишете като администратор в директорията '**/mnt**' (където са разположени основните софтуерни пакети):

```
# arch-chroot /mnt
```

Необходимо е да генерирате базовите езикови настройки на системата:

```
# locale-gen
```

Тъй като Свободните операционни системи уважават вашата свобода, **GNU/Linux-libre** не налага нито инсталирането на конкретни софтуерни пакети, нито съобразяването с решения на определена компания-собственик. Вместо това

можете да посочите какво точно желаете да се инсталира в зависимост от функциите, за които ще ползвате вашата система; и по всяко време можете да изтриете излишните ви пакети и да инсталирате други, съобразно вашите нужди и предпочитания.

При операционни системи като **Trisquel** например ви се предполага да последвате инсталационния процес от графичен интерфейс, като на първо време можете да се доверите на предложения набор от софтуерни пакети по подразбиране. При системи като **Parabola** обаче се разчита на това вие сами да организирате своята система по предпочитан от вас начин и в зависимост от функциите, за които мислите да я ползвате, като указвате от командния ред какви пакети желае да бъдат инсталирани след като бъде инсталирано системното ядро. За осъществяване на самия инсталационен процес можем да предложим един основен набор от софтуерни пакети за **Parabola**:

```
# pacman -S linux-libre-lts linux-libre-firmware \
grub bash-completion networkmanager nano
```

(където със символа '****' указваме, че вторият ред от командата продължава на същия ред в терминала; можете да пропуснете този символ и да въведете горните софтуерни пакети на един ред, преди да натиснете '**Enter**'; но ако поставите по начина, по който е дадена по-горе, командата ще бъде интерпретирана благодарение на символа '****' като такава на един единствен ред).

Цитираните софтуерни пакети са основните, необходими за осигуряване базовата функционалност на системата. Няма пречки на по-късен етап да инсталирате и всякакъв друг софтуер, който без съмнение ще ви бъде необходим за работа – но на този етап препоръчваме да се концентрирате върху самата инсталация. Препоръчаните базови софтуерни пакети са следните:

linux-libre-lts
системно ядро с базовите софтуерни пакети на системата

linux-libre-firmware
софтуерен пакет за функциониране на хардуерните компоненти

grub
спомагателна програма за стартиране на операционната система

bash-completion
помощен инструмент за „довършване“ на възможните команди при въвеждане в терминал – въведете първите няколко символа от командата и натиснете два пъти **[Tab]**

networkmanager
помощен инструмент за свързване към internet

nano
програма за текстообработка директно в терминал

Както вече стана дума, възможно е в процеса на инсталирането системата да изведе въпрос за това коя от няколко достъпни и взаимозаменяеми версии на определен софтуерен пакет предпочитате да бъде инсталирана. В такъв случай отделните версии ще бъдат номерирани и от вас се очаква да въведете предпочитания номер, след което да натиснете **[Enter]**. В повечето случаи първата предложена алтернатива е за предпочитане.

След като основните софтуерни пакети бъдат инсталирани, трябва да пристъпите към стартиращата програма **GRUB** (**GRand Unified Bootloader**), която позволява да стартирате операционната система или да изберете коя от достъпните операционни системи да заредите, респективно – с коя от достъпните конфигурации на системното ядро да сторите това. Благодарение на **GRUB** можете да разположите на един и същи компютър няколко операционни системи с по няколко различни конфигурационни състояния и да стартирате тези от тях, които са ви необходими в конкретния случай (за разлика от това например **Windows** след инсталирането си премахва всякакви други възможности за

стартирането на каквото и да било друго освен **Windows** – без това да е технологично необходимо). Можете да инсталирате **GRUB** чрез тези команди:

```
# grub-install --recheck /dev/Устройство
```

(където под '**Устройство**' разбираме наименованието, под което системата разпознава вашия твърд диск, примерно '**sda**'); след това:

```
# mkdir /boot/grub
```

... и накрая:

```
# grub-mkconfig -o /boot/grub/grub.cfg
```

Следва съществения за вашата информационна сигурност момент, в който трябва да въведете парола за достъп до системата като администратор. Тази парола трябва да бъде наистина надеждна – защото позволява пълен достъп до системата и дава възможност за нейното цялостно настройване (включително и с цел да се компрометира сигурността). Една парола е толкова по-надеждна, колкото е по-дълга и по-сложна (в Част **V** ще се спрем подробно върху този въпрос). Тук само ще отбележим, че съществуват широко известни bruteforce технологии, позволяващи последователно автоматизирано опитване на предполагаеми комбинации, докато паролата бъде най-после уцелена. Ако вашата парола за достъп като администратор не е достатъчно дълга и достатъчно сложна, с такива технологии може да бъде разбита за секунди. Затова препоръчваме да въвеждате дълги и сложни пароли (от **15** и повече символа, съдържащи всякакви безразборни символи и изрази, избягващи очаквани клавишни последователности или съществуващи думи и номера). Можете да въведете паролата чрез тази команда:

```
# passwd
```

Системата ще изведе указание '**New password:**' в очакване да въведете своята парола и след потвърждаването ѝ (няма да бъде изведен индикатор за броя въвеждани символи) ще я запише по криптографски hash-начин, който не позволява нейното извличане. При всяко следващо въвеждане на парола системата ще я преобразува по същия криптографски hash-начин, за да установи дали резултатът е същият. Това дава възможност паролата да бъде проверявана, без да съществува записана в „чист“ вид в системата.

Следващият съществен за вашата информационна сигурност момент е създаването на допълнителен потребителски акаунт, чрез който ще осъществявате достъп до системата без администраторски права. Това е необходимо при текущо ползване на системата, когато не осъществявате никакви специални настройки, инсталиране или премахване на програми и т.н. В интерес на информационната сигурност е да достъпвате вашата система само чрез този акаунт и да осъществявате достъп като администратор само тогава, когато това е наистина необходимо.

Можете да създадете акаунт като потребител без администраторски права така:

```
# useradd -m Потребител
```

(където под '**Потребител**' разбираме наименованието, под което желаете системата да разпознава съответния потребител). Препоръчваме наименованието да е кратко, да не включва никаква специфична информация, интервали или други специални символи (освен '**-**' и '**_**') и букви, различни от **US**-стандартизацията на латиницата. Често срещани наименования са например '**user**', '**owner**' или '**me**'.

Акаунтът на потребителя без администраторски права също трябва да бъде защитен чрез парола. Имайте предвид, че естеството на тази парола е такова, че тя ще бъде въвеждана често, докато работите с устройството (например всеки път, когато желаете да отключите Работния плот след оставяне на компютъра за малко и влизане в режим на очакване). Затова тази парола може да бъде по-лесна и удобна за многократно въвеждане, включително „пред погледите“ на хората наоколо. Все пак обаче трябва да имате предвид, че евентуално проникване в потребителския акаунт чрез тази парола може да доведе например до монтирането на Keylogger-програми за проследяване на

въведените пароли – и оттам да доведе до прихващане на паролата за достъп с администраторски права. Ето защо не бива да я подценявате. Можете да въведете парола на потребителския акаунт за достъп без администраторски права чрез тази команда:

```
# passwd Потребител
```

(където под '**Потребител**' разбираме наименованието, под което сте указали системата да разпознава съответния потребител). Ако в момента сте в сесия от акаунта на този потребител, можете да смените неговата парола и без да въвеждате наименованието му – просто чрез тази команда:

```
$ passwd
```

Системата ще изведе указание '**Current password:**' в очакване да въведете настоящата парола на текущия потребител и след като направите това, ще можете да въведете и потвърдите неговата нова парола, която също ще бъде записана по криптографски hash-начин.

Както вече на няколко пъти стана дума, **GNU/Linux-libre** системите са замислени като сървъри с много потребители и са много чувствителни към личната неприкосновеност. Ето защо един потребител без администраторски права не може да достъпва директории и файлове на други потребители, освен ако те не са разрешили това. По време на инсталационния процес (когато създадохме потребителската директория) действахме като администратор, поради което сега нямаме достъп до други директории от системата, освен до нашата потребителска директория и до намиращия се в нея наш Работен плот. Това е така, защото в графичната среда по подразбиране действаме като потребител без администраторски права, а всичко извън потребителската директория и намиращия се в нея Работен плот на съответния потребител е притежание на други потребители или е от кореново значение за системата. Следователно, за да позволите на потребителя без администраторски права да ползва вашия потребителски '**data**' дял (който по-горе настроихме да се закача към директорията '**/data**'), трябва да укажете на системата да го възприема като принадлежност на този потребител – чрез тази команда:

```
# chown Потребител /Потребителска_директория
```

След като приключите с горните настройки, е време да излезете от **chroot**-режима:

```
# exit
```

... да рестартирате компютъра:

```
# reboot
```

Когато захранването на компютъра бъде изключено, трябва да откачите хардуерно вашия „жив“ носител (да извадите **USB**-паметта от порта; да извадите оптичния диск от **CD/DVD**-устройството) и да изчакате повторното зареждане, но вече от инсталираната система на компютъра. Ако всичко е наред, ще се зареди програмата **GRUB**, от която можете да укажете желаната за стартиране операционна система (ако не сторите това за няколко секунди, по подразбиране ще се зареди първата предложена възможност) и ще се появи терминал с указание да въведете потребител:

```
Parabola GNU/Linux-libre Идентификатор (tty1)
parabola login:
```

За да влезете в системата с администраторски права (ще ви бъдат необходими за следващите настройки), от вас се очаква да въведете потребител '**root**' и паролата, която преди малко въведохте за достъпване на системата с такива права.

На следващо място е необходимо да зададете наименование, под което системата ще разпознава вашето компютърно устройство:

```
# hostnamectl hostname Компютър
```

(където под '**Компютър**' разбираме наименованието, под което желаете да бъде разпознавано устройството). Препоръчваме наименованието на вашия компютър да е кратко, да не включва никаква специфична информация, интервали или други специални символи (освен '**-**' и '**_**') и букви, различни от **US**-стандартизацията на латиницата. Често срещани наименования са например '**computer**' или '**comp**'.

Следва да ориентирате системата относно времето и пространството, където се намирате. Това е важно особено за софтуерните хранилища и цифровите подписи, които често се обуславят от периоди на валидност. Ако системата не е правилно ориентирана, е възможно да признае като „валиден“ остарял цифров подпис, вкл. такъв, който на по-късен етап е бил анулиран от притежателя му. Можете да ориентирате системата относно времето и пространството така:

```
# timedatectl set-timezone Континент/град
```

(където под '**Континент/град**' разбираме вашия континент и град, например '**Europe/Sofia**').

Остава да въведете автоматична синхронизация на времето:

```
# systemctl enable --now systemd-timesyncd
```

На следващо място трябва да активирате функционалностите на системата за свързване към internet:

```
# systemctl enable --now NetworkManager
```

Вече можете да се свържете безжично към internet:

```
# nmcli dev wifi connect Мрежа password Парола
```

(където под '**Мрежа**' разбираме наименованието на вашата **WiFi** мрежа, а под '**Парола**' – паролата за нейното достъпване). Можете да проверите дали имате internet-свързаност така:

```
# ping Сървър
```

(където под '**Сървър**' разбираме адреса на предпочитан от вас сървър – например този на Фондацията за Свободен софтуер '**fsf.org**'). При наличие на internet-свързаност, системата ще започне да извежда подобни резултати:

```
PING fsf.org (209.51.188.174) 56(84) bytes of data.
64 bytes from www.fsf.org (209.51.188.174): icmp_seq=1 ttl=54 time=121 ms
64 bytes from www.fsf.org (209.51.188.174): icmp_seq=2 ttl=54 time=119 ms
64 bytes from www.fsf.org (209.51.188.174): icmp_seq=3 ttl=54 time=119 ms
```

Прекъснете процеса (като натиснете **[Ctrl]**+**[c]**) и продължете нататък с инсталацията.

След като вече имате internet-свързаност, следва да инсталирате следващите базови софтуерни пакети, осигуряващи функциониране на системата от гледна точка на обикновения потребител:

```
# pacman -S gdm gnome-shell gnome-control-center \
ttf-dejavu lxterminal nautilus
```

Препоръчаните базови софтуерни пакети са следните:

gdm
програма за графично влизане в системата;

linux-libre-lts
системен графичен интерфейс;

gnome-control-center
интерфейс за потребителски настройки;

ttf-dejavu
основни шрифтове за графичния интерфейс;

lxterminal
терминал за графичната среда;
където под 'Пакет_1' и 'версия_1' разбираме версията

nautilus
програма за графично управление на файловете.

Впоследствие можете да инсталирате и други софтуерни пакети, в зависимост от това какво правите на компютъра и от какъв софтуер се нуждаете – но на този етап препоръчваме да се съсредоточите само върху най-основното.

Имайте предвид, че инсталационният процес не може да бъде осъществен, ако бъде констатиран проблем дори само с един от указаните софтуерни пакети (липсва необходимият за удостоверяване на автентичността му цифров подпис или същият е анулиран от притежателя му; съответното софтуерно хранилище не работи и в момента няма други достъпни хранилища; инсталационният файл е повреден или изключен от програмистката общност като ненадежден; и подобни). Ако това се случи, системата ще изведе съобщение за грешка с наименованието на проблемния пакет и ще откаже да осъществи инсталацията. В такъв случай трябва да изключите този пакет от командата и да опитате отново с останалите пакети. Възможно е да се наложи да подмените някои софтуерни пакети с други, за да довършите успешно процеса, ако тяхното инсталиране се окаже по някаква причина невъзможно.

Можете да видите инсталираните на вашата система софтуерни пакети:

```
# pacman -Qs
```

Системата ще изведе списък с пакети, обозначението на всеки от които ще има подобен вид:

```
core/nano 7.2-1 [installed]
    Pico editor clone with enhancements
```

От горния запис научаваме, че пакетът '**nano**' (чиято последна версия към момента е '**7.2-1**' се намира в софтуерното хранилище '**/core**' и в конкретния случай го имаме инсталиран (**installed**) в нашата система.

Можете да видите всички налични софтуерни пакети, свързани с определена ключова дума (без значение дали са инсталирани на вашата система или не) чрез тази команда:

```
# pacman -Ss Ключова_дума
```

(където под '**Ключова_дума**' разбираме част от наименованието, обозначение за предназначението или друг идентификатор на интересуващия ви софтуерен пакет или на област от приложения, за които ви е необходим).

Когато горните софтуерни пакети бъдат инсталирани, можете да стартирате графичната среда:

```
# systemctl enable --now gdm
```

Добре дошли във вашата Свободна операционна система!

Дотук усвоихме знания и умения за инсталирането на една Свободна операционна система, които по-долу ще трябва да приложим още веднъж в модифициран вид, за да инсталираме нейна изцяло шифрована версия. За да обезпечим обаче сигурна среда за едно надеждно цялостно шифроване, трябва първо да обезпечим софтуера от „ниско“ ниво – на този въпрос ще се спрем подробно в Част **III** по-долу.

6. За обновяването на системата и за възможните усложнения във връзка с това

Съществен аспект от информационната сигурност е да поддържате системата обновена до последни версии на използвания софтуер – защото това означава да приложите всички допълнително направени поправки на забелязани слабости в софтуера (включително такива във връзка със сигурността). Следва да напомним, че дори по-простите операционни системи са твърде комплексни проекти, ангажиращи труда на хиляди и десетки хиляди програмисти от всички краища на света; практически няма човек, който да познава всичките софтуерни компоненти на операционната система и това позволява често да бъдат допускани слабости, които на по-късен етап едни или други разработчици забелязват и поправят. Също с напредъка на технологиите се появяват нови изисквания, а така също – нови рискове за сигурността, които трябва да бъдат отразявани. Ето защо обновяването на софтуера е от първостепенна важност.

Добре е на първо място да си създадете навик за регулярно обновяване на софтуерните пакети, които използвате на вашата система – чрез тази команда:

```
# pacman -Syu
```

При изпълнението на командата ще бъде изведено съобщение за стартирането на пълно обновяване на системата (**Starting full system upgrade...**), ще бъдат посочени софтуерните пакети, подлежащи на обновяване и ще бъде изведено указание да посочите дали посоченото обновяване да бъде осъществено. След въпроса ще видите указание '**[Y/n]**' (от англоезичните '[**Y**]**es**' и '[**n**]**o**'), където главната буква обозначава възможност, заложена в системата като подразбираща се (логично, предвид въведената преди това команда да бъде осъществено обновяване). В случай, че бъде изведено подобно указание, ако просто натиснете **[Enter]**, системата ще изпълни подразбиращата се възможност, отбелязана с главна буква (в случая – '[**Y**]**es**').

Възможно е дадени шифровъчни ключове да са с изтекъл срок на валидност или да са анулирани от техните притежатели. Тогава ще бъдете попитани дали желаете да бъдат изтрити от системата – тогава възможните отговори ще бъдат '**[y/N]**' и ако просто натиснете **[Enter]**, системата ще изпълни подразбиращата се възможност, отбелязана с главна буква (в случая – '[**N**]**o**').

Възможно е в процеса на инсталирането системата да изведе въпрос за това коя от няколко достъпни и взаимозаменяеми версии на определен софтуерен пакет предпочитате да бъде инсталирана – като всяка от предложентие версии бъде обозначена с номер. Ако в такъв случай просто натиснете **[Enter]**, системата по подразбиране ще инсталира първата предложена версия. В останалите случаи е необходимо да въведете предпочитания от вас номер преди натискането на **[Enter]** (като при повече версии (когато няма достатъчно място да бъдат изобразени всичките на екрана) с натискане на **[n]** (от англоезичното [**N**]**ext**) можете да преминете към следващите, докато изчерпате списъка и отново се върнете на първите).

Възможно е докато се изпълнява горната команда, системата да изведе грешка и да откаже да завърши инсталационния процес, поради неактуализирани 'публични' ключове, с които се удостоверява автентичността и интегритетът на необходимите софтуерни пакети. Можете да отстраните грешката с тази команда:

```
# pacman-key --refresh-keys Адрес_на_неактуализиран_ключ
```

(където под '**Адрес_на_неактуализиран_ключ**' разбираме електронната поща, с която се обозначава неактуализираният 'публичен' ключ, станал причина за грешката при удостоверяването. Горната команда ще укаже на системата да потърси съответния 'публичен' ключ в достъпните сървъри с публични ключове (в Част **VIII** ще се спрем по-подробно на този въпрос), за да обезпечи надеждна криптографска проверка за автентичност и интегрите на интересуващите ни софтуерни пакети. След като актуализирате необходимите 'публични' ключове (могат да бъдат няколко наведнъж), трябва да повторите по-горната инсталационна команда.

За да избегнете грешки с неактуализирани 'публични' ключове, е добре да обновявате списъците с криптографските ключове, посредством които се проверява автентичността и интегритетът на софтуера – чрез тези команди:

```
# pacman -Sy parabola-keyring archlinux-keyring
```

... след което:

```
# pacman-key --refresh-keys
```

Възможно е в някакъв момент системата да изведе съобщение, че не може да осъществи обновяването, защото няма достатъчно памет. Ако сте спазили препоръчителните размери на системния '**/**' дял, това най-вероятно се дължи на препълване с архиви от стари файлови пакети, които са били сваляни и използвани за предходни обновявания, но все още стоят в паметта. Можете да ги прочистите чрез тази команда:

```
# rm /var/cache/pacman/pkg/*
```

След като необходимата актуализация бъде извършена, можете да пристъпите и към самото инсталиране на необходимите софтуерни пакети – чрез командата '**pacman -S Пакети**' (където под '**Пакети**' разбираме конкретните софтуерни пакети, които желаете да бъдат инсталирани на вашата система).

Възможно е при опита за обновяване да възникне конфликт между взаимно свързани софтуерни пакети, при което исканото обновяване ще бъде отказано и ще бъде изведен подобен резултат:

```
resolving dependencies...
looking for conflicting packages...
error: failed to prepare transaction (could not satisfy dependencies)
:: installing Пакет_1 (версия_1) breaks
dependency 'версия_2' required by Пакет_2
```

(където под '**Пакет_1**' и '**версия_1**' разбираме версията на инсталиран в системата софтуерен пакет, която би влязла в конфликт с '**версия_2**' на '**Пакет_2**', ако бъде осъществено обновяването и се инсталира тази последна налична версия. За да не бъде допуснат срив на системата от подобен конфликт, се извежда отказ и исканото обновяване изобщо не се осъществява. Възможно е с последващи версии на софтуерния пакет конфликтът да бъде отстранен, но това (ако изобщо бъде направено) отнема време и системата може да остане без обновления за опасно дълъг период. За да ускорите процеса, се препоръчва да докладвате конфликта пред разработващите и поддържащите вашата операционна система. Ако ползвате **Parabola**, можете да докладвате на този адрес:

```
https://labs.parabola.nu/
```

(необходимо е да си създадете потребителски акаунт, за да можете да подадете доклад и да проследите развитието му).

Докато очаквате конфликтът да бъде отстранен, можете да обновите с игнориране на конфликта:

```
# pacman -Sud
```

... но това не се препоръчва (освен при крайна необходимост), тъй като може да доведе до грешки и срив в системата.

Възможно е процесът по обновяването да бъде прекъснат внезапно (например поради изключване на захранването). Такова прекъсване при последващ опит обновяването да бъде довършено може да доведе до невъзможност системата да синхронизира инсталационната база данни – извежда се подобен резултат:

```
:: Synchronizing package databases...
error: failed to update libre (unable to lock database)
error: failed to update core (unable to lock database)
error: failed to update extra (unable to lock database)
error: failed to update community (unable to lock database)
error: failed to update pcr (unable to lock database)
error: failed to synchronize all databases
```

В такъв случай преди да подновите обновяването с командата '**pacman -Syu**', е необходимо за изтриете сгрешената (внезапно прекъсната) процедура по обновяване, преди да започнете процеса наново:

```
# rm /var/lib/pacman/db.lck
```

(където файлът '**db.lck**' отговаря за обновяването и при внезапно прекъснат процес е възможно да бъде повреден).

Възможно е също така внезапното прекъсване на процеса по обновяване да доведе до изтриване на стари софтуерни пакети и системни файлове, без обаче да се довърши записването на новите пакети и файлове – извежда се подобен резултат:

```
ldconfig: File Празен_системен_файл is empty, not checked.
```

... или при опит да стартирате определена програма – такъв резултат:

```
Програма: error while loading shared libraries: Липсващ_системен_файл:
cannot open shared object file: No such file or directory
```

В горните два случая са възникнали проблеми със софтуерните пакети, свързани с '**Празен_системен_файл**' (наличен като заглавие, но празен), респективно – с '**Липсващ_системен_файл**' (напълно липсващ). За да бъдат поправени проблемите, е необходимо да установите с кой софтуерен пакет са свързани тези празни / липсващи системни файлове и съответният пакет да бъде преинсталиран. За да сторите това, трябва първо да обновените наличните бази с данни:

```
# pacman -Fy
```

... и след това да потърсите интересуващия ви софтуерен пакет, свързан с празния / липсващия системен файл:

```
# pacman -F Системен_файл
```

(където под '**Системен_файл**' разбираме съответния системен файл).

Ще бъде изведен подобен резултат:

```
community/Проблемен_пакет
```

(където '**Проблемен_пакет**' е търсеният проблемен софтуерен пакет за преинсталиране, предхождан от обозначение на софтуерното направление, в което бива разработван).

Веднъж като установите проблемния пакет, можете да го преинсталирате така:

```
# pacman -S --overwrite '*' Проблемен_пакет
```

(където под '**Проблемен_пакет**' разбираме наименованието на проблемния софтуерен пакет).

За да установите по-прецизно наличните проблеми със софтуерни пакети в системата (ако има такива) можете да използвате тази команда:

```
# paccheck --quiet --file-properties
```

(за която цел е необходимо да имате инсталирана програмата **Paccheck**). Ще бъде изведен подобен резултат:

```
shadow: 'адрес/Системен_файл' permission mismatch (expected 4755)
warning: Системен_файл: mtree data not available (No such file or directory)
warning: Системен_файл: mtree data not available (Success)
```

Следва да обърнете внимание на обозначените с '**warning**' редове, с които системата предупреждава за системни файлове, свързани с проблемни софтуерни пакети. Остава да установите свързаните с всеки от тези системни файлове проблемни софтуерни пакети (чрез проследената по-горе команда '**pacman -F Системен_файл**') и след това да преинсталирате всеки от проблемните софтуерни пакети (с командата '**pacman -S --overwrite '*' Проблемен_пакет**'). Добра практика е да поддържате системата си в кондиция, като проверявате от време на време за подобни проблеми и ги отстранявате по собствена инициатива – без да е възникнал конкретен повод за това; който конкретен повод в някои случаи би могъл да се окаже твърде труден за отстраняване проблем.

В случай на срив е добре да прегледате журналния файл **/var/log/pacman.log** (ще се ориентирате по датата и часа на срива) и като проследите последните (неуспешни) опити за инсталиране / обновяване на софтуерни пакети, да преинсталирате същите чрез командата '**pacman -S --overwrite '*' Проблемни_пакети**'.

7. Създаване на външни хранилища за съдържание

И преди да завършим темата за инсталиране на Свободна операционна система (макар и все още нешифрована), за пълнота ще обърнем внимание на създаването на външни хранилища за потребителско съдържание (външни твърди дискове, **USB**-памети, **SD**-карти и подобни). Общото за такива хранилища е, че те са предназначени за ползване от вече стартирана операционна система, към която се закачат и чрез приложенията ѝ се достъпва съхраняваното в тях съдържание.

За да подготвите вашето външно хранилище, трябва да закачите устройството, което ще използвате за тази цел към системата и с командата '**lsblk**' да се ориентирате под какво наименование бива разпознавано от системата. Ще бъде изведен подобен резултат:

```
NAME    MAJ:MIN RM   SIZE RO TYPE  MOUNTPOINT
sda       8:0    0 100.0G  0 disk
├─sda1    8:1    0     8G  0 part  [SWAP]
├─sda2    8:1    0    30G  0 part  /
└─sda3    8:3    0    62G  0 part  /data
sdb       8:32   1    10G  0 disk
└─sdb1    8:33   1    10G  0 part  /run/media/Потребител/USB-памет
```

(където под '**USB-памет**' разбираме закаченото от вас устройство (в конкретния пример – **USB**-памет, която системата разпознава като '**sdb**' с един единствен дял '**sdb1**'); а под '**/run/media/Потребител/USB-памет**' разбираме адреса на точката на закачане, от която устройството е свързано със системата).

За да можете да подготвите устройството като външно хранилище, трябва най-напред да откачите от системата неговите физически дялове (ако има такива), с тази команда:

```
# umount /dev/Физически_дял
```

(където под '**Физически_дял**' разбираме наименованието, под което системата разпознава интересуващия ви физически дял на устройството – в случая единствения такъв дял '**sdb1**'; ако има и други физически дялове, трябва да откачите всеки от тях).

Ако във вашето устройство няма обособен физически дял, може да се наложи да го създадете чрез проследената по-горе програма **Fdisk**. След като физическите дялове (ако има такива) са откачени от системата, можете да ги форматирате:

– ако ще ползвате устройството изключително от **GNU/Linux** системи:

```
# mkfs.ext4 /dev/Физически_дял
```

– ако ще ползвате устройството и с несвободни **Windows**-базирани системи (не препоръчваме такава съвместимост, ако мислите да съхранявате поверителна информация в това хранилище):

```
# mkfs.fat /dev/Физически_дял
```

(където под '**Физически_дял**' разбираме наименованието, под което системата разпознава съответния дял от интересуващото ви устройство – в горния случай '**sdb1**'; под '**ext4**' – файловата система **Ext4**, подходяща за **GNU/Linux** базирани системи; и под '**fat**' – файловата система **FAT**, подходяща и за несвободни системи като **Windows**).

Ако форматирате с файловата система **FAT** и във вашето устройство няма обособен физически дял, може да се наложи да го създадете чрез проследената по-горе програма **Fdisk**.

Докато форматирате, можете да укажете на системата и това да го обозначи с определено наименование – като въведете към горната команда и допълнителния параметър '**`-L Наименование`**' (където под '**Наименование**' разбираме наименованието, под което желаете физическият дял да бъде разпознаван) – командата ще придобие такъв вид:

```
# mkfs.ext4 /dev/Физически_дял -L Наименование
```

... или респективно такъв вид:

```
# mkfs.fat /dev/Физически_дял -n Наименование
```

След като вече сте форматирали, можете да закачите към системата:

```
# mount /dev/Физически_дял /Точка_на_закачане
```

(където под '**`/Точка_на_закачане`**' разбираме директория в системата, от която желаете да достъпите съдържанието на устройството или съответния негов физически дял). За да направите тази директория достъпна от графичния интерфейс (от потребителя без администраторски права) на вашата система, трябва да укажете системата да възприема съответната директория като принадлежност на този потребител – чрез проследената по-горе команда '**`chown`**'. Ако желаете да въведете по-детайлни права, можете да ползвате командата '**`chmod`**'. В последствие, ако закачате устройството към системата, тя ще го възприема автоматично като такова с присвоени съответните видове права.

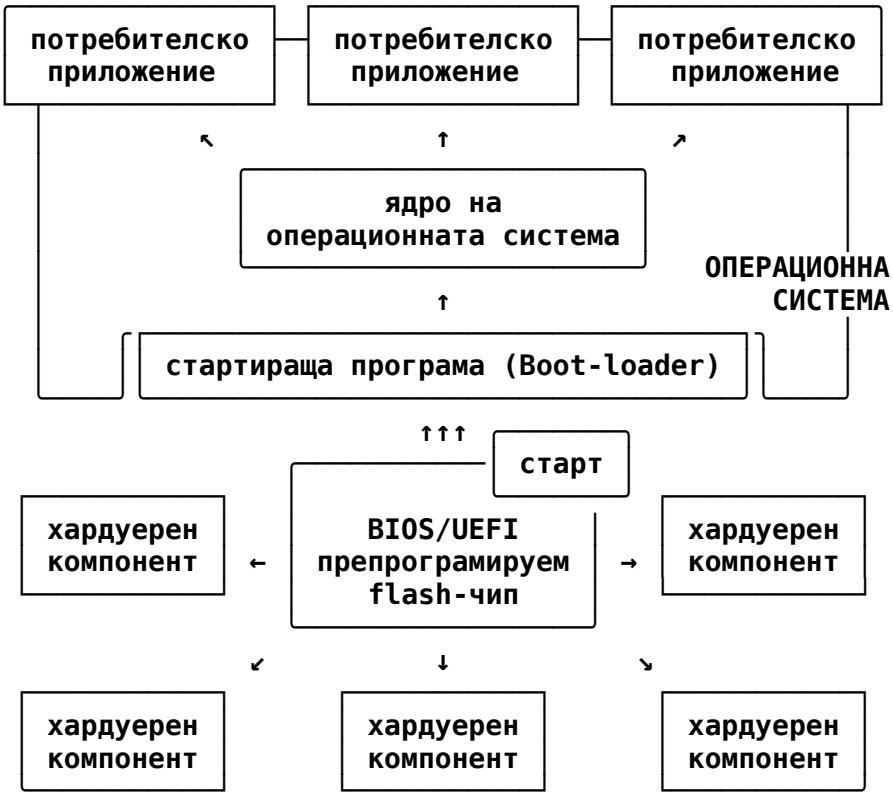
В Част **IV** по-долу ще се върнем отново на въпроса за създаването на външни хранилища за съдържание, но вече в напълно шифрован вид.

III. ЗАЩИТАВАНЕ НА СОФТУЕРА ОТ НИСКО НИВО

Обикновено в компютрите има множество енергонезависими препрограмируеми flash-чипове, известни като **EEPROM** (**Electrically Erasable Programmable Read-Only Memory**). Характерно за такива чипове е, че те съхраняват информацията без необходимост от електрозахранване и позволяват електрично изтриване и презаписване. При компютрите **x200** например на такъв **EEPROM** чип се съхранява т.нар. софтуер от „ниско“ ниво. При по-старите компютри това е **BIOS** (**Basic Input / Output System**), а при по-новите това е надграден до **UEFI** (**Unified Extensible Firmware Interface**). И в двата случая софтуерът от „ниско“ ниво се явява „посредник“ между хардуера на компютъра и вашата операционна система.

https://en.wikipedia.org/wiki/EEPROM
https://en.wikipedia.org/wiki/Flash_memory
https://en.wikipedia.org/wiki/BIOS
https://en.wikipedia.org/wiki/UEFI

Когато електричеството проникне в централния процесор при включване на компютъра, негова първа задача е да намери и прочете стартиращата програма, за да изпълни заложените в нея инструкции за стартиране. В резултат от това **BIOS/UEFI** се свързва с останалите компоненти от хардуера (процесор, **RAM**-чипове, твърд диск, клавиатура, екран и т.н.), проверява тяхната наличност и изправност, и след като се убеди, че системата е способна да работи, активира стартиращата програма (**Boot-loader**) на операционната система (обикновено при **GNU/Linux** това е програмата **GRUB**), разположена на твърдия диск или на „живия“ носител, от който стартираме компютъра. Веднъж активирана, стартиращата програма пуска в действие операционната система и едва след като тя се зареди, виждате стартиращия екран и вашия Работен плот. За пълнота ще добавим и това, че операционната система е съвкупност най-общо от системно ядро (базова програма, обезпечаваща функционирането на системата) и потребителски приложения (на първо място графичната среда и достъпните чрез нея файлови мениджъри, текстообработващи програми, плеъри, браузъри, комуникатори и т.н.), които позволяват на потребителя да ползва своя компютър по удобен начин, без да е необходимо да програмира.



Принципна структура на едно компютърно устройство

Обикновеният потребител може да работи с потребителските приложения, без изобщо да подозира, че те са напълно зависими от базовото обезпечение на ядрото на операционната система; и без изобщо да подозира, че операционната система на свой ред е напълно зависима от софтуера от „ниско“ ниво, без който компютърът нито може да бъде стартиран, нито може да изпълни коя да е от функциите на потребителските приложения.

Нещо повече – **BIOS/UEFI** софтуерът от „ниско“ ниво управлява пряко хардуерните компоненти и практически взема „окончателните решения“ за това какво, доколко и как да се случва в системата – в т.ч. какво да може и какво да не може да правят потребителските приложения и операционната система като цяло. Възможно е софтуерът от „ниско“ ниво да се окаже програмиран за изпълняването на инструкции, които операционната система изобщо не може да следи и контролира. В някои случаи (при по-новите версии на **BIOS** и **UEFI**) е възможно и дистанционно достъпване и препрограмиране на софтуера от „ниско“ ниво, което открива вектори за атака специално срещу вашата система или изобщо срещу системи от типа на вашата.

https://en.wikipedia.org/wiki/Intel_Management_Engine

Ето защо е от критична важност софтуерът от „ниско“ ниво също да бъде напълно свободен (което би позволило да изследваме как точно функционира и да го променяме, ако счетем това за необходимо). Само така можем да гарантираме сигурност на начина, по който функционира хардуерът и операционната система, инсталирана на него. И тъй като стандартното **BIOS/UEFI** софтуерно обезпечение от „ниско“ ниво не е свободно, се налага да препрограмираме енергонезависимия flash-чип, като заменим **BIOS/UEFI** със Свободна софтуерна алтернатива (например **Libreboot**).

За пълнота ще отбележим, че алтернативата **Libreboot** е резултат от създаването на **Coreboot** (известна преди това и като **LinuxBIOS**). Целта на **Coreboot** е да бъде създадена опростена алтернатива на софтуера от „ниско“ ниво **BIOS/UEFI**, която да е достъпна за редактиране съгласно нуждите на потребителя. Въпреки, че се разпространява съгласно свободния софтуерен лиценз **GNU GPL** (**GNU General Public License**) обаче, **Coreboot** не е „**libre**“ в строгия

смисъл на думата (допуска прилагането на несвободни компоненти). Наличието на достатъчно подробна техническа документация и възможността за редактиране обаче е позволила всичките несвободни компоненти да бъдат премахнати и заменени от програмисти, разработващи Свободен софтуер, като така се достига до напълно свободния софтуер от „ниско“ ниво **Libreboot**, който препоръчваме. По принцип няма пречка да приложите във вашата система и **Coreboot** (който все пак е значително по-надежден от **BIOS/UEFI**), но това означава да се доверите на софтуер, предполагащ ползването на несвободни компоненти – което няма как да препоръчаме да направите.

Пак за пълнота ще отбележим и това, че първоначалните идеи на **Libreboot** впоследствие бяха изоставени и днес проектът се развива в значително отстъпление от Свободата – предлага се поддръжката включително и на хардуер, който не е напълно свободен; във връзка с което в софтуера от „ниско“ ниво се допускат и несвободни компоненти. Благодарение на Свободните софтуерни лицензи обаче, част от разработващите проекта програмисти, които държат на Свободата, клонираха проекта и възобновиха неговия първоначален web-адрес (**`libreboot.at`**), където понастоящем продължават да се развиват само напълно свободни версии на **Libreboot**. Именно тези версии препоръчваме.

`https://www.coreboot.org/`
`https://libreboot.org/`
`https://libreboot.at/`

Трябва също да се има предвид, че някои компютърни компоненти (особено такива от по-ново поколение) се проектират така, че да не се позволи функционирането им с друг софтуер или с други хардуерни компоненти, освен с определените от разработващата компания. Това ограничава свободата на потребителите и ги принуждава да ползват своето компютърно оборудване по начин, наложен от другиго. Такива компютри (или отделни компоненти в тях) са несвободни и трябва да бъдат подменени, ако желаете да сте сигурни в надеждността на вашата система. Тъй като свободните технологии се налага да „догонват“ новосъздаващите се компютърни компоненти (които по правило се произвеждат от големи собственически компании), обикновено по-старото компютърно оборудване е по-добре проучено и документирано, и може да бъде инсталирано по-успешно със Свободен софтуер.

Най-вероятно вашият компютър ще успее да се справи със стартирането на избраната от вас Свободна операционна система, но е напълно възможно да не може да заработи със свободна алтернатива на **BIOS/UEFI**. Поради тази причина препоръчваме да избягвате най-новите модели хардуер, ако планирате да базирате своята сигурност на свободни технологични решения. Можете в тази връзка да се обърнете към съществуващите списъци с марки и модели компютърно оборудване, за които е установено, че са съвместими с **Libreboot** и има утвърдени подходи за препрограмиране на техния **BIOS/UEFI** flash-чип. И ако вашият хардуер не попада сред тези марки и модели, е най-добре да го замените с друг.

`https://libreboot.at/docs/hardware/`

Подходящо подбрано компютърно оборудване, инсталирано с **Libreboot** и с напълно свободна **GNU/Linux-libre** операционна система, готова за ползване от потребителя, можете да поръчате например от **Technoethical** и **Libiquity**. Въпреки тези удобни възможности обаче ви насърчаваме да се интересувате активно от вашия компютър и по възможност да придобиете колкото се може повече знания и умения, които да ви позволяват да се справите сами с предизвикателствата пред вашата информационна сигурност (или най-малкото лично да участвате във всеки елемент от нейното обезпечаване). Само така можете да гарантирате за себе си, че сигурността на вашата поверителна информация не се поставя в зависимост от благоволението на трети страни, а зависи доколкото е възможно от самите вас (именно това беше и първоначалният мотив за започване на работата по това ръководство).

`https://technoethical.com/`
`https://libiquity.com/`

1. Необходими устройства, инструменти и консумативи

Препрограмирането на **BIOS/UEFI** flash-чипа за обикновените потребители изглежда да е най-трудният момент от това ръководство. Ако обаче бъдат надмогнати известни предразсъдъци и задръжки, такова препрограмиране може да се осъществи успешно в „домашни условия“ – като просто следвате напътствията, без да е необходимо да прилагате някакви по-специфични знания и умения.

Необходимо ви е следното оборудване:

– обикновен помощен компютър, инсталиран със Свободна операционна система;

– компютърно устройство с входно-изходен интерфейс тип **SPI** (**Serial Peripheral Interface**) за комуникация с flash-чипа и **USB**-кабел за свързването на това устройство към помощния компютър (по-долу ще предложим достъпни варианти);

– няколко къси съединителни кабелчета с щифтове за свързване и **SOIC**-щипка за захващане към flash-чипа с **8** или **16** пина (в зависимост от вида на вашия **BIOS/UEFI** flash-чип, който ще уточним малко по-долу в изложението);

– или (вместо съединителни кабелчета и **SOIC**-щипка – ако например пиновете на вашия flash-чип се окажат твърде малки, за да бъдат захванати от **SOIC**-щипка) – поялник с регулатор на температурата (до около **375°C**) с плосък или сходен коничен накрайник с големина **0,1 ÷ 0,3** mm); тънък емайлиран проводник с дебелина **0,1 ÷ 0,3** mm; малко безоловен тинол с дебелина **0,5 ÷ 1,0** mm, за предпочитане с флюс; и отделно течен или гелообразен флюс за запояване.

Всичко това (освен помощения компютър и устройството с интерфейс за директни инструкции) е достъпно във всеки средно зареден магазин за електронни компоненти. Както стана дума – надолу ще предложим различни варианти за препрограмиране на вашия **BIOS/UEFI** flash-чип и в зависимост от избраното устройство за комуникация с flash-чипа ще добавим още няколко елемента към оборудването – всичките лесно достъпни от търговската мрежа. Преди да продължим нататък, обаче, трябва да разглобим компютъра, чийто flash-чип се готвим да препрограмираме, за да установим с колко пина е и можем ли да го захванем със съответната **SOIC**-щипка.

Всичко това е достъпно във всеки средно зареден магазин за електронни компоненти. Както стана дума – надолу ще предложим различни варианти за препрограмиране на вашия **BIOS/UEFI** flash-чип и в зависимост от избраното устройство с интерфейс за подаване на директни инструкции ще добавим още няколко елемента към оборудването – всичките лесно достъпни от търговската мрежа.

Преди да продължим нататък, обаче, трябва да разглобим компютъра, чийто flash-чип се готвим да препрограмираме, за да установим с колко пина е и можем ли да го захванем със съответната **SOIC**-щипка.

Както стана дума, за препрограмиране с **Libreboot** е необходимо устройство с интерфейс тип **SPI** (**Serial Peripheral Interface**) за комуникация с вашия **BIOS/UEFI** flash-чип. Съществуват множество такива устройства, които са изпитани в практиката и възможностите им са добре документирани – съответно няма пречка да ползвате кое да е от тях, стига да ги проучите и да адаптирате следващите напътствия към особеностите на техния интерфейс.

Arduino Nano например е такова устройство, което е предпочитано от **Libreboot**-общността. Това е микроконтролер, който работи доста лесно. В допълнение за препрограмирането с това устройство ще ви бъдат необходими: късичък **USB**-кабел за свързване между помощния компютър и микроконтролера; прототипна монтажна платка (**Breadboard**); четириканален преобразувател на логическите нива (**Logic Level Shifter**) за двупосочно конвертиране на сигналите от микроконтролера към flash-чипа (от **5V** към **3,3V**) – какъвто е например преобразувателят **Pololu 2595**; също **12** бр. късички съединителни кабели с „мъжки“ накрайници; и още **6** бр. късички съединителни кабели с „женски“ накрайници от едната страна и „мъжки“ от другата (ако ползвате **SOIC**-щипка), или вместо тях – **6** бр. емайлирани проводници за запояване към съответните пинове на чипа (ако не ползвате **SOIC**-щипка).

BeagleBone Black Rev C е алтернативен едноплатков компютър, който също позволява подаването на директни инструкции към интересуващите ни хардуерни компоненти. Интерфейсът му включва вход за електрозахранване (**5V**), **LAN**, **MiniB USB** и **USB 2.0**, **HDMI**-изход и слот за **MicroSD**-карта (която служи на едноплатковия компютър като „твърд диск“). Заедно с това **Beagle Bone Black** е снабден с две гнездови рейки (всяка от които с по **46** „женски“ щифта) и една щифтова рейка (с **6** „мъжки“ пина). Това е повече от достатъчно за нашите цели. Ако изберете това устройство, ще ви бъдат необходими: **MicroSD**-карта с капацитет **4GB** или повече (и **SD**-слот или **USB**-преходник за нейното закачане към помощния компютър, от който да я подготвите); **USB-UART**-кабел, работещ с **3,3V** логически нива, завършващи с „женски“ накрайници – **GND**, **TXD** и **RXD** (ако има четвърто **VCC** разклонение, просто го изолирайте и оставите свободно); захранващ адаптер (тип **DC Jack** с **5V** изход) за захранване на **BeagleBone** устройството от електрическата мрежа или **miniUSB** към **USB** кабел за захранването му от помощния компютър; още **6** къси съединителни кабели с „женски“ накрайници от една страна и „мъжки“ от другата (ако ползвате **SOIC**-щипка), или вместо тях – такива с „мъжки“ накрайници от двете страни и **6** бр. емайлирани проводници за запояване към пиновете на чипа (ако не ползвате **SOIC**-щипка).

По-долу ще предложим напътствия за препрограмиране на flash-чипа с всяко от посочените устройства, но няма пречка да изберете и друго подобно устройство (например **Arduino Uno**) – стига да проучите документацията на съответните устройства и да адаптирате напътствията към особеностите на съответния интерфейс.

https://store.arduino.cc/products/arduino-nano
https://www.pololu.com/product/2595
https://beagleboard.org/black
https://store.arduino.cc/products/arduino-uno-rev3

Независимо от това кое устройство ще изберете, препрограмирането в най-общи линии ще включи следните етапи: частично разглобяване на компютъра, чийто flash-чип ще препрограмирате, за да осигурите физически достъп до последния; подготовка на избраното устройство за комуникация с flash-чипа; подготовка на необходимия софтуер; свързване на избраното устройство към чипа; и препрограмиране на flash-чипа със Свободен софтуер.

2. Разглобяване на компютъра, чийто flash-чип ще бъде препрограмиран

По-горе споменахме списъка с компютри, които са проучени от **Libreboot**-общността и вече съществуват разработени подходи за тяхното успешно препрограмиране. Този списък с времето (надяваме се) ще се допълва, но не може да се очаква, че ще покрие всичките потребителски очаквания (въпреки, че по наше мнение включва много добри компютърни конфигурации в бизнес-класа, които се държат прилично, въпреки относително по-старите модели). Тук е мястото да отбележим, че разработването на подход за заменяне на **BIOS/UEFI** софтуерното обезпечение със свободен софтуер изобщо не е лека задача и поради тази причина списъкът с разработени модели се допълва бавно. Също така – в случай, че желаете да препрограмирате компютър от друга марка или модел – това може да се окаже напълно невъзможно (ако компютърът е защитен срещу подобно „своеволие“). В други случаи това може да бъде опасно за компютъра (в някои случаи подобни експерименти могат да завършат неуспешно, а пътят назад към старото и работещо състояние на системата да се окаже невъзможен). Ето защо не препоръчваме да се опитвате да препрограмирате компютри, за които няма утвърден и проверен в практиката подход за справяне с изникващите задачи – освен ако не разполагате със задълбочени познания в областта или просто желаете да експериментирате, съгласявайки се с напълно възможния фатален и необратим изход за опитния компютър от вашия експеримент.

Преносимият компютър **Lenovo ThinkPad x200** е може би най-предпочитаният модел за препрограмиране с **Libreboot** – поради няколко причини. На първо място той е проучен подробно и в **Libreboot**-общността съществува богат опит с обслужването на такива компютри. На второ място – препрограмирането на **BIOS/UEFI** flash-чипа на **Lenovo ThinkPad x200** почти винаги се осъществява успешно, без да се сблъсква с допълнителни усложнения. И не на последно място – хардуерът на този модел позволява сравнително лесното му разглобяване и осигуряване на физически достъп до **BIOS/UEFI** flash-чипа. При други марки и модели може да се наложи да разглобите целия компютър на съставните му части, за да достигнете до **BIOS/UEFI** flash-чипа, което понякога е наистина трудоемка задача. При **ThinkPad x200** е достатъчно да отвиете няколко винта, да отстраните клавиатурата с едно просто движение и да откачите предната горна част от корпуса на машината. (За пълнота ще отбележим, че моделът **Lenovo ThinkPad x200** е почти идентичен

на модела **x200s** – поради което всичко, което ще проследим по-долу (освен ако изрично не е отбелязано), е валидно и за двата модела).

Когато пристъпите към разглобяване на компютъра, трябва на първо място да го изключите от електрозахранването и да откачите батерията. В противен случай протичащото в системата електричество може да доведе до причиняване на късо съединение (макар и слаботоково) и това да повреди някои компоненти. Специално моделът **x200** (както и **x200s**) от долната страна на корпуса си (където са винтовете за неговото разглобяване) има обозначения за функциите на тези от винтовете, които ни интересуват – съответно можете да се ориентирате кои от тях трябва да развиете. Направете разграничение между винтовете, които държат клавиатурата (**4** бр., един от които – в гнездото на батерията и достъпен след отстраняването й); и тези, които държат горна част от корпуса (още **5** бр.). Както ще забележите, от долната страна на корпуса има още **3** бр. отвори със символа на клавиатурата, предназначени за оттичането на течности, ако такива залеят случайно компютъра; **2** бр. винтове без обозначения в горните два ъгъла (предназначени за пантите на монитор); **2** бр. винтове на капака в средата на корпуса (откъдето са достъпни слотовете на **RAM**-чиповете); **1** бр. по-голям винт вляво за капака към твърдия диск (който можете лесно да откачите и с едно просто издърпване да извадите от компютъра); и още **2** бр. винтове към средата вдясно, без обозначения, които държат хардуерни компоненти, нямащи нищо общо с предстоящите манипулации).

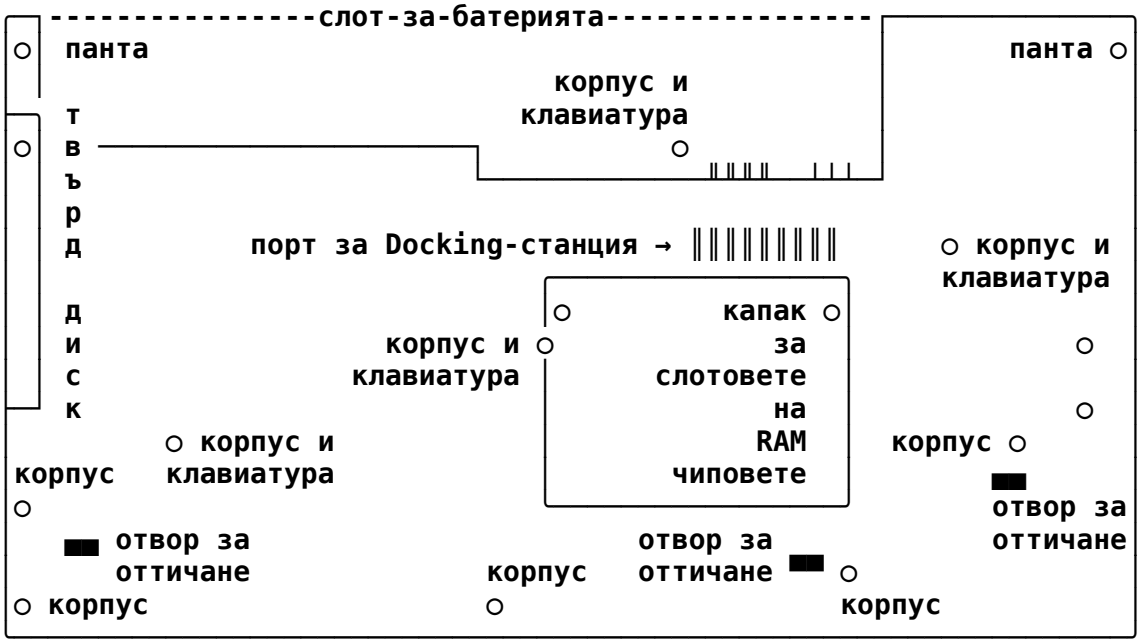


Схема на винтовете в корпуса на компютри Lenovo ThinkPad x200 и x200s

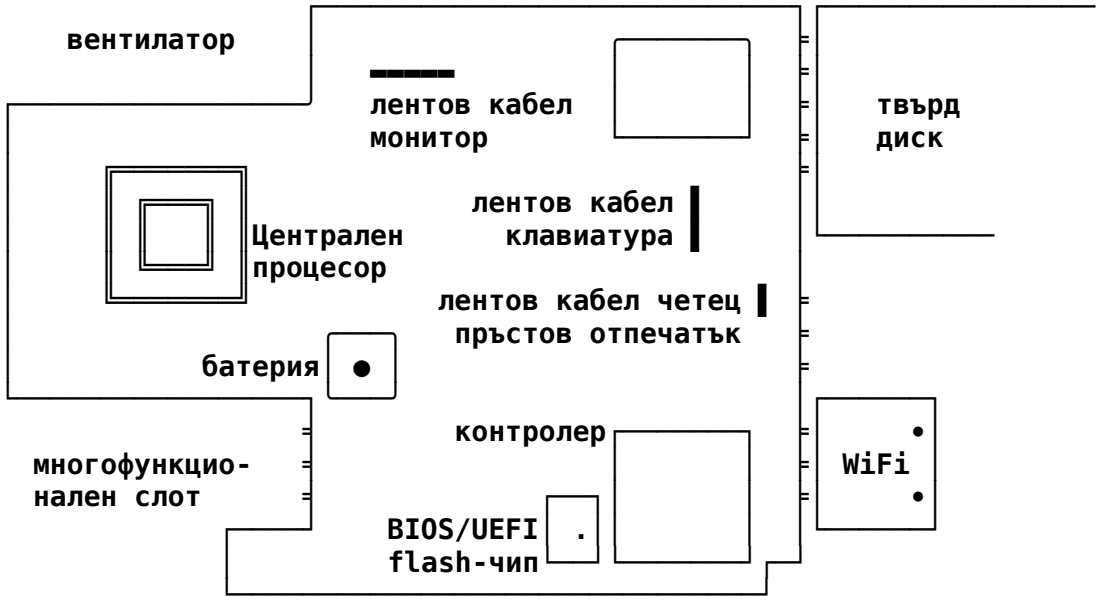
Използвайте наистина подходяща и качествена кръстата отвертка (не по-голяма и не по-малка – дори и да е възможно да се справите с нея), защото в противен случай ще допуснете прескачане на върха на отвертката върху главите на винтовете и това може да ги увреди. Желателно е отвертката да е с намагнетизиран връх, за да изваждате лесно винтовете от гнездата им. Дръжте отвертката изправена и добре натисната към винта, за да не допуснете прескачане. Поставяйте винтовете някъде настрана, с върхчетата нагоре, където няма да ги бутнете случайно – като ги подреждате във форма, съответстваща на разположението им в компютъра – това ще ви позволи да си спомните кой винт откъде беше и да избегнете случайното им разместване (понякога винтовете се различават (макар и минимално) и това може да увреди гнездата им в пластмасовия корпус, докато се опитвате да ги завиете). Преди да приключите, огледайте внимателно долния корпус на компютъра, за да се уверите, че сте развили наистина всички винтове, които е необходимо (включително и този под капака на батерията).

Можете още сега да отстраните от слотовете им двойката **RAM**-чипове и да извадите от гнездото му твърдия диск, тъй като по време на препрограмирането е добре дънната платка да бъде освободена то всички излишни хардуерни компоненти.

След като сте отвили всичките необходими винтове от долния корпус и сте отстранили ненужните на този етап хардуерни компоненти, трябва да обърнете отново компютъра и да отворите екрана му. Поставете ръцете си плътно с целите длани и пръстите върху клавиатурата и докато натискате надолу, плъзнете към екрана. При моделите **x200** и **x200s** клавиатурата се отмества няколко милиметра към екрана и това позволява да я извадите, като я подкачите с нокти (не с остри и метални предмети) отдолу и повдигнете нагоре. Не подкачайте клавиатурата за клавишите (тъй като може да ги откъртите), а се опитайте да я подкачите за носещата шина (която държи всичките клавиши отдолу). След като извадите клавиатурата, не я отстранявайте рязко – преди това трябва да откачете лентовия кабел, с който е свързана към дънната платка. Откачането на подобни лентови кабели става от предвидената за това дръжчица, а ако няма такава – с палец и показалец, като хванете здраво от двете страни щифта и дръпнете право нагоре. Не дърпайте самия лентов кабел, защото това усилие може да го прекъсне. Работете внимателно с такива щифтове, тъй като увреждането им е лесно и това може да направи клавиатурата неизползваема.

След клавиатурата идва ред на предния горен панел от корпуса (където обикновено лягат дланите при ползване на клавиатурата). Този панел се откача с внимателен натиск нагоре по страничните ръбове (като внимавате да не го счупите); по ръбовете има миниатюрни щипки, които при определено усилие се откачат една по една, докато накрая откачите панела от долната част на корпуса. Тук също обикновено има лентов кабел, свързващ четеца за пръстови отпечатъци (ако вашият компютър разполага с такъв) към дънната платка. Преди да отстраните горния панел от корпуса, трябва да откачите и този лентов кабел, като внимавате да не го прекъснете – въпреки, че за нуждите на информационната сигурност този четец няма да ви бъде необходим (и не се препоръчва да го ползвате).

При модела **x200** препрограмируемият **BIOS/UEFI** flash-чип е достъпен веднага, след като отстраните предната горна част от корпуса.



Дънна платка с основни компоненти на компютър Lenovo ThinkPad x200

Дънната платка при модела **x200s** изглежда почти като показаната по-горе за модела **x200**, с малката (но съществена за нашите цели) разлика – **BIOS/UEFI** flash-чипът е разположен от обратната страна на платката. Това означава, че за да получите физически достъп до flash-чипа при модела **x200s**, трябва да продължите с разглобяването, като отстраните твърдия диск, обрамчващата предната част на корпуса горна рамка (откъм екрана) и металната пластина над дънната платка; отлепите нарочните лепенки, които придържат кабелите към местата им и отместите кабелите настрани; накрая отстраните винтовете, които държат самата дънна платка и извадите последната от корпуса на компютъра, за да достигнете до разположения от обратната ѝ страна flash-чип. Докато вършите това, подреждайте

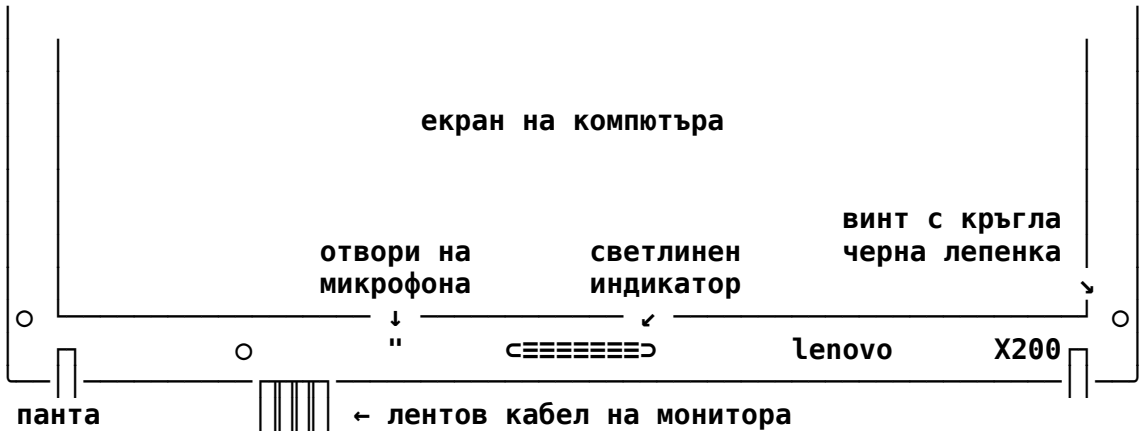
внимателно разглобените компоненти и винтовете към тях, за да можете след това да върнете всяко едно нещо на правилното му място.

При всички случаи трябва да освободите дънната платка от хардуерните компоненти, които биха могли да дадат смущение по време на препрограмирането на flash-чипа. Трябва да извадите твърдия диск от гнездото му. За целта е достатъчно да отстраните капака отстрани на корпуса на компютъра и да изтеглите твърдия диск навън. Трябва също така да освободите **RAM**-чиповете от слотовете, разположени на гърба на дънната платка. Ако работите с **x200** и не се налага да изваждате дънната платка от корпуса на компютъра, ще достигнете до слотовете, като отстраните малкия капак от дъното на корпуса. Всеки от **RAM**-чиповете се освобождава от съответния слот, като натиснете двойките придържащи го щифтове в двете срещуположни посоки.

Междувременно, сега е удобен момент да подмените фабрично вложената wifi-карта за безжично internet-свързване, тъй като моделите **x200** и **x200s** идват с несвободна такава, която отказва да работи с **GNU/Linux-libre**. Това е единственият несвободен хардуерен компонент, който трябва да подмените при моделите **x200** и **x200s**, ако желаете системата ви да работи с напълно Свободен софтуер. За тази цел трябва първо да откачите двата кабела, с които wifi-картата се свързва към вътрешната антена на компютъра (като запомните кой кабел от кой пин откачате; обикновено светлият кабел е закачен на пин **1**, а тъмният – на пин **2**). Пиновете на тези кабели се откачат, като хванете здраво с палец и показалец, и дърпате право нагоре (като внимавате самият щифт да не се накланя, тъй като може лесно да бъде изкривен). След това трябва да отвиете винтовете, с които wifi-картата е фиксирана към мястото си, да я извадите от нейния слот и да я замените със свободен еквивалент. Почти всички модели на **Atheros** са свободни – като на вас ви е необходим модел със стандарта на окачване **mPCIe-full** или **mPCIe-half**. Подходящ е например моделът **AR5BHB116** (който е със стандарт на окачване **mPCIe-half**). Още свободни wifi-карти (с възможност за търсене на конкретен модел) можете да видите на този адрес:

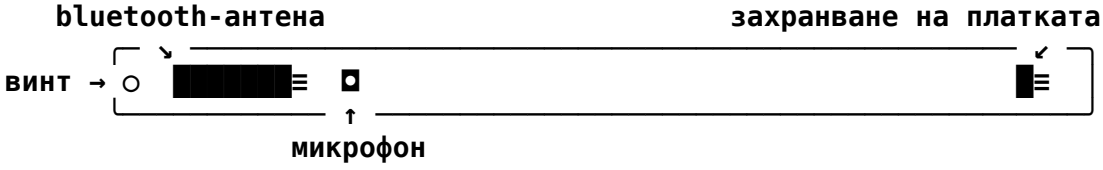
`https://h-node.org/wifi/catalogue/en`

Сега е удобен момент и за това да отстраните вградения микрофон, bluetooth-антената и web-камерата на компютъра (ако конкретният компютър има web-камера). Тази манипулация е важна за потребители, които водят поверителни разговори „в присъствието“ на своя компютър и не желаят да има каквато и да било възможност за проследяване на тези разговори чрез техния хардуер. Така, ако въпреки всичките им усилия се получи пробив в софтуерната сигурност и това позволи на трети страни да получат неразрежен контрол над тези компоненти от хардуера, осъщественият достъп ще остане безполезен. При моделите **x200** и **x200s** микрофонът и bluetooth-антената са вградени в средата и наляво в долната рамка на монитора, а web-камерата (ако има такава) е вградена в средата в горната рамка. За да достигнете до тези компоненти, трябва да отстраните кръглите черни лепенки, с които са покрити трите винта в долната рамка на монитора (по един в двата ъгъла и един към средата на лявата половина). След като отвиете винтовете, трябва с нокти или с нещо тънко и остро (като например ламинирана или пластмасова карта; но не метално острие, тъй като то би надрало повърхностите) да откачите предната част от корпуса, обрамчващ монитора. При тази манипулация трябва много да внимавате да не натискате самия монитор и да не прилагате по-големи усилия, от които рамката може да се счупи.



Място на микрофона в рамката на монитора при x200 и x200s

След като постепенно освободите рамката от нейните щипки (имайте предвид, че в горната част е възможно да има нанесено лепило и освобождаването да се окаже по-трудно), можете просто да откачите щифта със захранването на web-камерата и този със захранването на bluetooth-антената (те повече няма да могат да работят, каквото и да се случи със софтуера).



Място на микрофона в платката на монитора при x200 и x200s

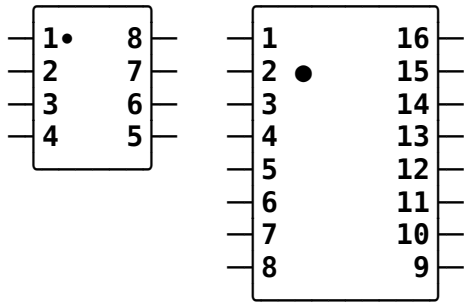
Микрофонът представлява малък правоъгълен компонент с едва забележим отвор на лицевата страна, разположен тъкмо зад двата отвора вляво от светлинния индикатор в долната рамка на монитора. Той е запоен към хоризонтално разположената в долната рамка на монитора продълговата платка. Трябва да отвиете винта в левия край на платката и да я свалите (като откачите разположения вдясно щифт с нейното захранване). Хванете микрофона с чифт здрави метални пинсети с остър връх и го нагрейте добре с поялника при температура около **375°C** (като го натиснете отгоре с върха на нагревателя, без да допирате други компоненти от платката). Нагряването на микрофона ще придаде топлина на тинола, с който е запоен компонентът и след няколко секунди ще се отпои от платката (без да прилагате големи усилия с пинсетите, за да не счупите нещо). След като отстраните микрофона, трябва просто да върнете платката на мястото ѝ (като не забравяте да я свържете със захранването ѝ отдясно и върнете винта от левия ѝ край на мястото му) и да затворите внимателно рамката на монитора. По-нататък, ако ви е необходим микрофон или web-камера, винаги можете да закачите външни устройства към портовете на компютъра (при моделите **x200** и **x200s** разполагате с **3 USB**-порта и аналогови стерео вход и изход за звук).

3. Идентифициране на BIOS/UEFI препрограмируемия flash-чип

След като отстраните клавиатурата и предната горна част на корпуса (при модела **x200**) или след като извадите дънната платка от корпуса на компютъра (при модела **x200s**), препрограмируемия **BIOS/UEFI** flash-чип вече е пред погледа ви. Той е малък и правоъгълен по форма. С прост поглед можете да установите дали е с **8** или с **16** пина (респективно дали ще ползвате **SOIC**-щипка за захващане с **8** пина или такава **16** пина; или пиновете са твърде малки и това прави **SOIC**-щипката неприложима.

Независимо дали вашият **BIOS/UEFI** flash-чип е с **8** или с **16** пина, съществува строга последователност в номерирането на пиновете (която съответства на функциите, изпълнявани от всеки един от тях). Номерацията винаги започва от единия ъгъл на чипа и продължава „кръгообразно“ до края на чипа. Кой от ъглите на чипа е негово „начало“

обозначава малка точица, вдълбана релефно в тялото на чипа (или обозначена с някакво цветно вещество, нанесено отгоре). Имайте предвид, че не винаги обозначаващата точка е точно в „началния“ ъгъл на чипа – понякога е позиционирана не чак толкова прецизно – но винаги обозначава отчетливо кой е „началният“ ъгъл. Който пин е най-близо до „началния“ ъгъл, е пин **1** – след него следва пин **2** и т.н., до края на редицата, като броенето трябва да продължи от отсрещната страна с най-близко разположения пин и така – до последния пин, разположен до „крайния“ ъгъл на чипа.



Номера на пиновете в BIOS/UEFI flash-чипа (8 или 16 пина)

Следва да имате предвид, че при някои компютри (включително **x200**) интересуващият ни **BIOS/UEFI** flash-чип изглежда запоен „наобратно“ – точката, обозначаваща „начало“ на чипа е долу вдясно – съответно от там започва номерирането на пиновете.

След като вече знаете как да определите номерата на пиновете, можете да свържете flash-чипа към избраното устройството за комуникация, за да започнете да подавате желаните от вас сигнали към съответните пинове.

В случай, че не разполагате с правилната **SOIC**-щипка или малкият размер на пиновете на flash-чипа не позволява захващането с такава щипка (както най-често се случва при модела **x200s**), можете да запоите тънки емайлирани проводници към съответните пинове (по-долу ще посочим конкретните необходими пинове) и да ги свържете към устройството чрез съединителни кабели. След като приключите с препрограмирането, ще разпоите проводниците от пиновете на flash-чипа. Ако обаче нямате опит със запояване, препоръчваме преди да посегнете към flash-чипа на компютъра, да поработите с поялника, тънките емайлирани проводници, тинола и флюс-пастата върху някакви стари електронни компоненти, които няма да ви трябват – за да придобиете известно усещане и рутина.

За да използвате емайлирани проводници (вместо **SOIC**-щипка), първо трябва да премахнете изолацията в краищата на всеки от тях (около **5** mm от всяка страна). Нужни са ви **6** бр. емайлирани проводници, всеки с дължина коло **10** cm. При някои такива емайлирани проводници изолацията обикновено е много тънка и дори трудно забележима (важно е да отбележим, че не е уместно да ползвате проводници без изолация, понеже може да предизвикате смущения и къси съединения, които може да повредят електрониката). Може да премахнете изолацията от краищата, като я изстържете внимателно с джобно ножче (внимавайте да не нараните повърхността на проводника, тъй като това на по-късен етап може да доведе до пречупването му).

След като сте премахнали изолацията от краищата на емайлираните проводници, нанесете тинол върху краищата му. Включете поялника и го нагрейте до температура около **375°C**. Разположете поялника на стабилна стойка, по начин, че да не го блъснете докато работите и да причините инцидент с нагорещения му накрайник. Вземете един от емайлираните проводници и намажете единия му край с флюс-паста. Когато поялникът вече е достигнал необходимата температура, избършете върха на накрайника му в неутрална повърхност (метални стружки за поялник или няколко слоя обикновен, небоядисан картон), за да почистите наслоеното по повърхността му окисление. Доближете нагорещения и почистен накрайник на поялника до жицата тинол, вземете върху накрайника една разтопена капка и бързо потопете в нея крайчето на емайлирания проводник – така, че тинолът да полепне по крайчето на проводника и да го оцвети в сребристия цвят на тинола – но без да остане капка тинол върху него. Повторете тази процедура за двата края на всеки от **6**-те емайлираните проводници.

Следващата стъпка е да запоите емайлираните проводници със съединителните кабели (тези, при които накрайниците са „мъжки“ от двете страни). Първо нанесете флюс върху единия накрайник на съединителния кабел и подобно на предишната стъпка, използвайте горещия поялник, за да нанесете тънък слой тинол върху металния накрайник. След това вземете един от емайлираните проводници, допрете единия му край към току-що тинования накрайник на съединителния кабел и използвайте горещия поялник, за да споите двете жици една към друга (единия накрайник на съединителния кабел с единия край на емайлирания проводник).

След като подготвите всички проводници и кабели, намажете пиновете на flash-чипа с флюс-паста (за да ги почистите от наслоеното окисление) и доближете свободния връх на емайлирания проводник до съответния пин (съгласно приложената по-долу схема). Натиснете върха на проводника върху пина с нагорещения връх на поялника за **1 ÷ 2** sec. Внимавайте връхчето на емайлирания проводник да стои фиксирано точно върху интересуващия ви пин (не по средата между два съседни пина). Проводникът ще предаде подаваната от поялника топлина на тинола, той ще се разтопи и ще се стече по проводника към пина, запоявайки ги едно към друго. Ако изпълните точно указанията, спояването не бива да отнеме повече от **1 ÷ 2** sec. Ако задържите поялника твърде дълго, рискувате да нагреете прекомерно flash-чипа и платката под него – което може да повреди електрониката.

Въпреки, че пиновете за нетренирано око изглеждат невъзможно близко разположени един до друг, запояването на тънки проводници към тях не е невъзможно – след известна практика ще започнете да го правите прецизно и с лекота.

Веднъж като запоите тънкия емайлиран проводник, можете леко да го раздвижите, за да проверите здравината на спойката. Ако проводникът поддава на мястото на спояването, значи спойката не е здрава и може би ще се счупи. В такъв случай доближете поялника близо до върха на слабо споения проводник и топлината ще разтопи спойката, съответно ще можете да го отстраните, за да опитате да го споите повторно към интересуващия ви пин. Ако обаче при раздвижване проводникът не поддава на мястото на спояването, значи спойката е здрава и ще можете да я ползвате. При по-нататъшно местене на свободния край на проводника придържайте на няколко сантиметра от споения край, за да предотвратите евентуално скъсване на спойката; и местете проводника колкото се може по-малко, като избягвате големи усуквания и напъни.

Ако се случи да разлеете повече тинол (и например споите няколко от съседните пинове на flash-чипа), трябва да почистите накрайника на поялника в неутрална повърхност (стружки за поялник; небоядисан картон) и да го доближите, за да разтопи излишния тинол и да го поеме върху себе си; след което отново да почистите накрайника и отново да съберете останалия излишен тинол, ако има такъв. При тази манипулация може да се наложи да отпоите вече споените емайлиран проводници и да започнете спояването отначало. При това действайте бързо, за да не предадете прекомерно количество топлина на пиновете и платката под чипа.

4. Архитектура и функционалности на микроконтролера Arduino Nano

Arduino Nano (както стана дума) е напълно подходящ микроконтролер с интерфейс за комуникация с вашия **BIOS/UEFI** flash-чип (по много сходен начин може да се ползва и съвсем близката модификация **Arduino Uno**). Входно-изходният интерфейс на **Arduino Nano** включва **USB**-вход за данни и електрозахранване (**5V**), и три щифтови рейки (две с по **15** и една с **6** „мъжки“ пина). Това е напълно достатъчно за нашите цели.

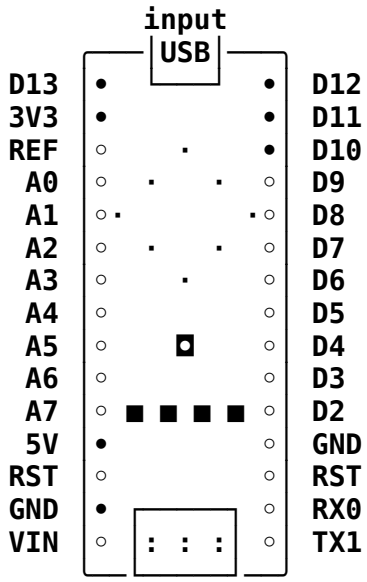


Схема на интерфейса на микроконтролера Arduino Nano

Всеки от пиновете в рейките на микроконтролера отговаря за определена специфична функция. Нас ни интересуват следните: от щифтовите рейки (от които ще подадете необходимите инструкции към препрограмируемия flash-чип) – пин **•D10** – (**Slave Select**; **Cable Select**; **SS**; **CS** – инициализира комуникацията с flash-чипа), пин **•D11** – (**Master Out Slave In**; **MOSI** – изпращане на сигнали към flash-чипа), пин **•D12** – (**Master In Slave Out**; **MISO** – приемане на сигнали от flash-чипа), пин **•D13** – (**Serial Clock**; **SCK** – синхронизиране на сигналите към и от flash-чипа), пин **•3V3** – (захранване на flash-чипа с напрежение **3,3V**), пин **•5V** – (сравнително напрежение от **5V** за преобразуване на сигналите до **3,3V**), и пин **GND** – (затваряне на електрическата верига).

5. Инсталиране на устройството Arduino Nano с помощния компютър

За да можете да укажете на микроконтролера да подаде необходимите инструкции към **BIOS/UEFI** flash-чипа, които ще доведат до премахване на несвободното **BIOS/UEFI** софтуерно обезпечение от „ниско“ ниво и до препрограмиране със свободната алтернатива **Libreboot**, е необходимо най-напред да инсталирате помощния компютър, от който ще подадете необходимите команди към микроконтролера; от който на свой ред подадените команди ще бъдат изпратени като сигнали към съответни пинове на flash-чипа.

Нуждаете се от няколко софтуерни пакета, които сега е моментът да инсталирате:

```
# pacman -S git make avr-gcc avr-libc avrdude flashrom
```

`git`
програма за проследяване на промени и синхронизация, и за изтегляне изходния код на програми

`make`
помощен инструмент за компилиране на софтуер

`avr-gcc`
помощен инструмент за компилиране на софтуер

`avr-libc`
софтуерни библиотеки за програмното осигуряване, което ще компилираме и инсталираме на микроконтролера

`avrdude`
софтуер за трансфериране на компилирания софтуер от помощния компютър към микроконтролера

`flashrom`
софтуер за препрограмиране на **BIOS/UEFI** flash-чипа

След като необходимите софтуерни пакети са инсталирани на помощния компютър, идва ред да свалите изходния код на софтуера, който ще инсталирате на микроконтролера:

```
# git clone https://github.com/noblepepper/serprog-duino.git
```

За да улесните по-нататъшната си работа, препоръчваме да преместите терминала в директорията, която се образува на помощния компютър със свалянето на изходен код:

```
# cd serprog-duino/
```

Отворете файла **`config/user_settings.h`** (например с програмата **Nano**) и променете този ред:

```
// #define S_SPEED        115200    /* Serial speed */
```

... по този начин:

```
#define S_SPEED        115200    /* Serial speed */
```

На следващо място идва ред да компилирате софтуера – с тази команда:

```
# make ftdi
```

След като компилирането завърши, свържете микроконтролера към помощния компютър (за предпочитане е да ползвате по-късичък **USB**-кабел, за да не се забавят (да не се смущават по друг начин) сигналите, протичащи през кабела) и трансферирайте току що компилирания софтуер към микроконтролера:

```
# make flash-ftdi
```

След като сторите това, идва ред да пристъпите към препрограмиране на flash-чипа.

6. Препрограмиране на BIOS/UEFI flash-чипа с Arduino Nano

След като микроконтролерът вече е свързан към помощния компютър и инсталиран с необходимия софтуер, идва ред да свържете микроконтролера към **BIOS/UEFI** flash-чипа, като опосредите тази свързаност с преобразувателя на

логическите нива; и да установите модела и капацитета на flash-чипа. За да сторите това, е необходимо да разглобите компютъра, който се готвите да препрограмирате и да достигнете физически до **BIOS/UEFI** flash-чипа (както проследихме по-горе в изложението).

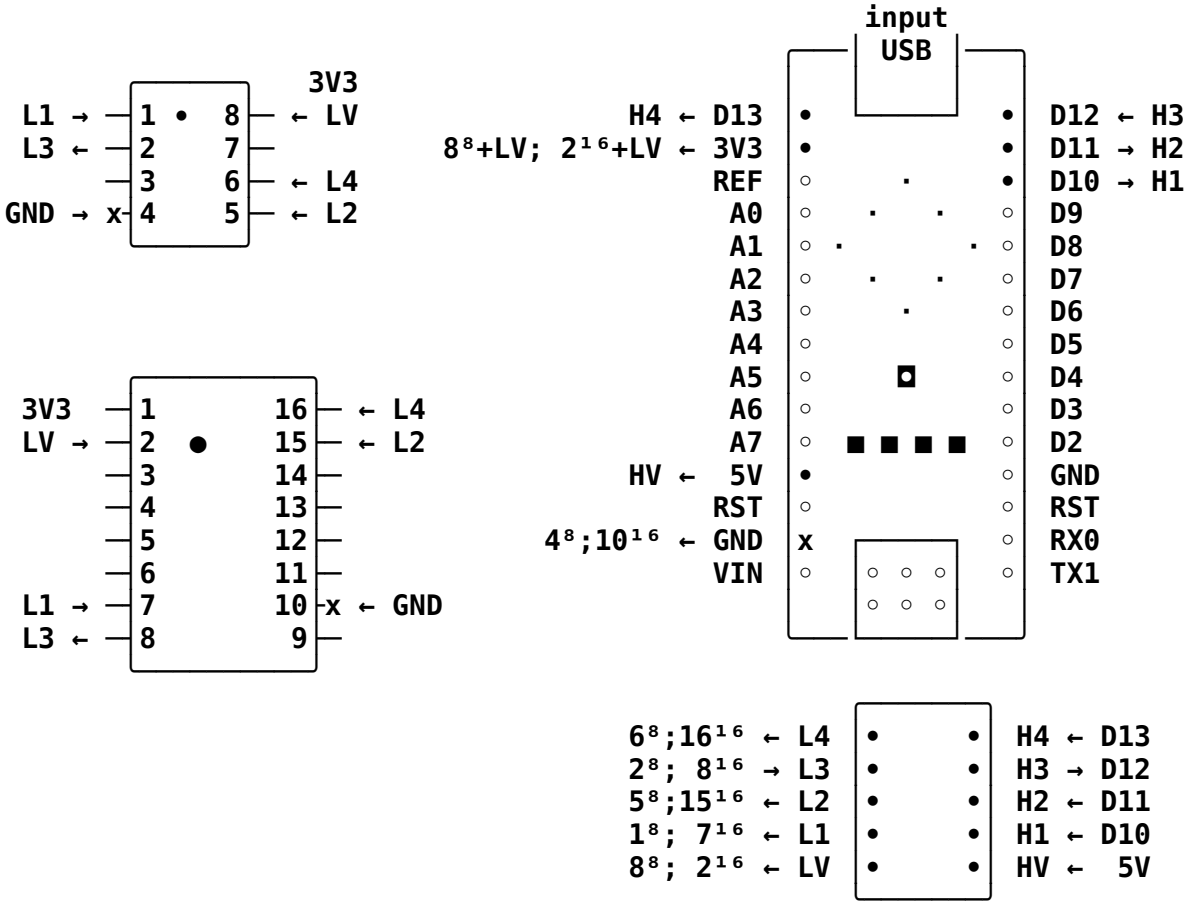
Най-удобно можете да свържете микроконтролера и преобразувателя посредством къси кабелчета (до **10** cm всяко) с „женски“ накрайници, които да закачите към указаните пинове на микроконтролера и обозначената като '**H**' (от англоезичното '[**H**]**igh**'; високо) половина от пиновете на преобразувателя: пин **D10** към пин **•H1**; пин **D11** към пин **•H2**; пин **D12** към пин **•H3**; пин **D13** към пин **•H4**; пин **3V3** към пин **•LV**; и пин **5V** към пин **•HV**.
Най-удобно можете да свържете микроконтролера и преобразувателя посредством прототипна монтажна платка (**Breadboard**) и къси кабелчета (до **10** cm всяко) с „мъжки“ накрайници, които да закачите към указаните пинове на микроконтролера и обозначената като '**H**' (от англоезичното '[**H**]**igh**'; високо) половина от пиновете на преобразувателя: пин **D10** към пин **•H1**; пин **D11** към пин **•H2**; пин **D12** към пин **•H3**; пин **D13** към пин **•H4**; пин **3V3** към пин **•LV**; и пин **5V** към пин **•HV**.

Най-удобно можете да свържете преобразувателя и flash-чипа посредством **SOIC**-щипка (която е за **8** или за **16** пина, в зависимост от конфигурацията на вашия flash-чип) и също така късички кабелчета с „женски“ и „мъжки“ накрайници, които посредством прототипната платка да закачите към другата половина от пиновете на преобразувателя, обозначени като '**L**' (от англоезичното '[**L**]**ow**'; ниско) и към пиновете на **SOIC**-щипката; или ако не разполагате с необходимата ви **SOIC**-щипка или пиновете на flash-чипа са твърде малки, за да бъдат захванати с такава щипка – посредством тънки емайлирани проводници (до **10** cm всеки) с премахната изолация в краища, които да запоите внимателно към необходимите пинове на чипа и да прикачите към съответните пинове на преобразувателя:

– при **BIOS/UEFI** flash-чип с **8** пина: пин **1** (**CS**) от чипа към пин **•L1** на преобразувателя, пин **2** (**MISO**) към пин **•L3** на преобразувателя, пин **4** (**GND**) към пин **GND** на микроконтролера, пин **5** (**MOSI**) към пин **•L2** на преобразувателя, пин **6** (**SCLK**) към пин **•L4** на преобразувателя, и пин **8** (**VCC**) към пин **•LV** на преобразувателя и към пин **•3V3** на микроконтролера;

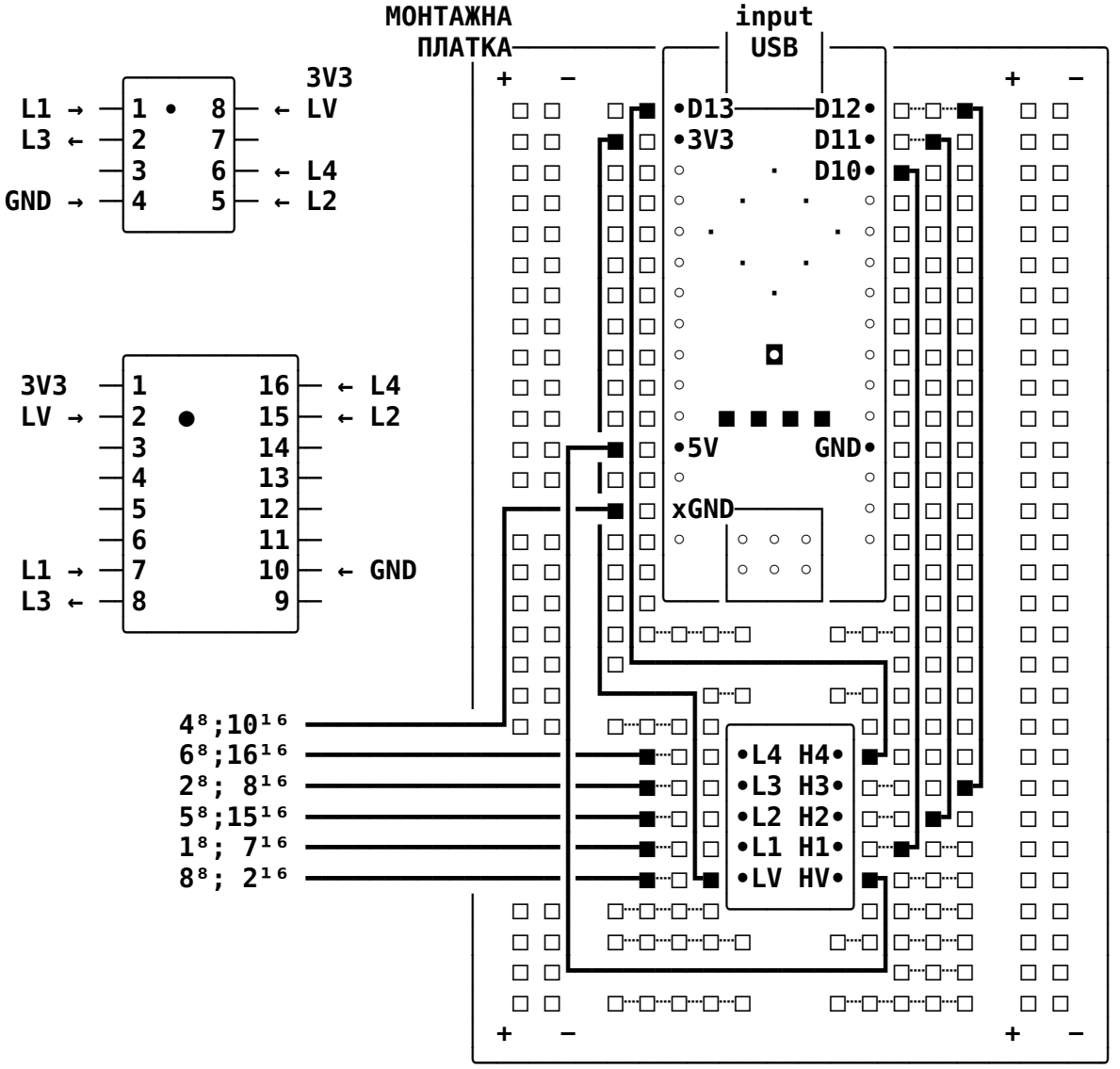
– при **BIOS/UEFI** flash-чип с **16** пина: пин **2** (**VCC**) от чипа към пин **•LV** на преобразувателя и към пин **•3V3** на микроконтролера, пин **7** (**CS**) към пин **•L1** на преобразувателя, пин **8** (**MISO**) към пин **•L3** на преобразувателя, пин **10** (**GND**) към пин **GND** на микроконтролера, пин **15** (**MOSI**) към пин **•L2** на преобразувателя, и пин **16** (**SCLK**) към пин **•L4** на преобразувателя.

При всички положения свързването трябва да стане по следната схема:



Свързване на BIOS/UEFI flash-чипа с микроконтролера Arduino Nano

Ако използвате прототипна монтажна платка (**Breadboard**), може да закрепите микроконтролера и логическия преобразувател за нея и да свържете съединителните кабели по следния начин:



Свързване с Arduino Nano посредством монтажна платка (Breadboard)

След като свържете съгласно горната схема и се убедите, че всичко изглежда правилно, идва ред да инсталирате програмата **Flashrom** (ако още не сте го направили), за достъпите с нея **BIOS/UEFI** flash-чипа, за да го препрограмирате:

```
# pacman -S flashrom
```

След като програмата **Flashrom** е инсталирана, можете да извършите проверка за правилността на свързването с **BIOS/UEFI** flash-чипа:

```
# flashrom -p serprog:dev=/dev/ttyUSB0:115200
```

Възможно е системата постоянно да отчита грешка (липса на открито препрограмируемо устройство). Това най-често се дължи на проблем при свързването. Проверете най-напред дали сте закачили микроконтролера към **BIOS/UEFI** flash-чипа точно съгласно дадените напътствия. Понякога се правят елементарни грешки – като например някой от

кабелите да е забравен откачен, някой от проводниците да е отпоен, **SOIC**-щипката да е поставена накриво и т.н. Ако всичко изглежда наред, изключете напрежението, почистете със спирт и памук (или с мека памучна тъкан) повърхностите, които правят контакт (връхчетата на **SOIC**-щипката, краищата на проводниците, щифтовете на кабелите и микроконтролера, пиновете на чипа), изчакайте всичко да изсъхне добре и опитайте отново. Ако системата продължава да отчита грешка, пробвайте да подменяте един по един свързващите компоненти. Възможно е проводниците да са твърде дълги или твърде дебели и сигналът да не може да преодолее съпротивлението. Възможно е **SOIC**-щипката да е дефектна и някой от сегментите в нея да не дава контакт. Възможно е някой от свързващите кабели да е прекъснат. Възможно е най-накрая да сте пропуснали някоя стъпка от напътствията. Няма причина **BIOS/UEFI** flash-чипът да не може да бъде достъпен софтуерно, ако е в изправност и работи, имате физически достъп до неговите пинове и сте го свързали правилно.

Възможно е също така да бъде открито препрограмируемо устройство, но да не може да бъде установена конкретната марка и модел на flash-чипа – видно от следния ред в изведените резултати:

```
Found Generic flash chip "unknown SPI chip (RDID)" (0 kB, SPI) on serprog.
```

Опитайте горната команда още няколко пъти – понякога системата не сработва първия-втория път, но след това все пак дава търсения резултат. Ако обаче продължава упорито да извежда указания за неразпознат flash-чип или за липса на открито препрограмируемо устройство, изключете микроконтролера и проследете дали всичко е монтирано правилно и дали връзките са достатъчно надеждни.

Ако всичко изглежда наред, но flash-чипът продължава да остава неразпознат, сменете свързващите кабелчета, преинсталирайте микроконтролера и пробвайте отново. Имайте предвид, че понякога се правят елементарни грешки – като например някой от кабелите да е забравен откачен, някой от проводниците да се е отпоил, **SOIC**-щипката да е поставена накриво и т.н. Ако всичко изглежда наред, изключете напрежението, почистете със спирт и памук (или с мека памучна тъкан) повърхностите, които правят контакт (връхчетата на **SOIC**-щипката, краищата на проводниците, щифтовете и пиновете), изчакайте всичко да изсъхне добре и опитайте отново. Ако системата продължава да отчита грешка, пробвайте да подменяте един по един свързващите компоненти. Възможно е проводниците да са твърде дълги или твърде дебели и сигналът да не може да преодолее съпротивлението. Възможно е **SOIC**-щипката да е дефектна и някой от сегментите в нея да не дава контакт. Възможно е някой от свързващите кабели да е прекъснат. Възможно е най-накрая да сте пропуснали някоя стъпка от напътствията. Няма причина **BIOS/UEFI** flash-чипът да не може да бъде достъпен софтуерно, ако иначе е в изправност и работи, имате физически достъп до неговите пинове и сте го свързали правилно.

При успешно разпознаване на flash-чипа, между многото изведени редове ще има и указание за вероятния модел на flash-чипа:

```
Found Macronix flash chip "MX25L8005/MX25L8006E/MX25L8008E/MX25V8005" (1024
kB, SPI) on serprog.
```

(откъдето научаваме вероятния модел чип (дадени са няколко подобни (почти идентични) модификации) на компанията-разработчик **Macronix**).

Можете да подадете горната команда още няколко пъти, за да се убедите, че има поваряемост на извежданите резултати. Ако множество пъти се извежда един и същи резултат, означава, че разпознаването с голяма вероятност е вярно.

Възможно е да бъде изведено указание, че е разпознат flash-чип от типа на няколко марки и модела:

```
Found Macronix flash chip "MX25L6405" (8192 kB, SPI) on serprog.
Found Macronix flash chip "MX25L6405D" (8192 kB, SPI) on serprog.
Found Macronix flash chip "MX25L6406E/MX25L6408E" (8192 kB, SPI) on serprog.
Found Macronix flash chip
"MX25L6436E/MX25L6445E/MX25L6465E/MX25L6473E/MX25L6473F" (8192 kB, SPI) on
serprog.

(...)

Multiple flash chip definitions match the detected chip(s): "MX25L6405",
"MX25L6405D", "MX25L6406E/MX25L6408E",
"MX25L6436E/MX25L6445E/MX25L6465E/MX25L6473E/MX25L6473F"
Please specify which chip definition to use with the -c <chipname> option.
```

Въпреки, че конкретната марка и модел все още не е разпозната, това дава индикация за най-вероятния вид на чипа. В допълнение установяваме, че чипът е с капацитет **8MiB** (записа '**8192 kB**' по-горе). За да установите по-точно марката и модела на чипа, можете да достигнете физически до него, да почистите добре повърхността му (отгоре може да има залепен етикет и цветни наслоения) и да прочетете физически фабрично записаната марка и модел. Внимавайте при почистването да не заличите надписите. Тъй като повърхността на чипа е с графитен цвят, а надписите са много малки по размер, трябва да ползвате увеличение и да променяте ъгъла на светлината, докато пречупването ѝ позволи да видите и разчетете написаното (или поне част от него). Следва да имате предвид, че не винаги изписаното върху корпуса на flash-чипа съответства напълно на действителната марка и модел – трябва да приложите известно старание и усет.

Когато най-накрая извлечете вероятната марка и модел на flash-чипа, можете да потърсите в internet дали наистина има такава (или подобна) марка и модел и чипът с какъв капацитет трябва да е.

Преди да пристъпите към самото препрограмиране, е добре да свалите в помощния компютър резервно копие от заварения запис на **BIOS/UEFI** – който ще бъде полезен, в случай, че нещо се объркa и не можете да стартирате системата с **Libreboot**. В противен случай може да се окаже, че не разполагате с необходимия софтуер от „ниско“ ниво и да направите третирания компютър неизползваем. Можете да свалите резервно копие от **BIOS/UEFI** така:

```
# flashrom -c Чип -p serprog:dev=/dev/ttyUSB0:115200 -r Резервно_копие
```

(където под '**Чип**' разбираме идентификатора на конкретната марка и модел на flash-чипа). Ако при проверката за правилност на свързването не е била изведена марката и моделът на конкретния flash-чип, можете да пробвате един по един предложените варианти; или да почистите внимателно корпуса на flash-чипа и да се опитате да разчетете неговия идентификатор с подходящо увеличение (като внимавате при почистването да не заличите и без това трудно различимия надпис).

Тъй като резервното копие от **BIOS/UEFI** има критично значение, препоръчваме да го свалите няколко пъти (поне **2** на брой) и да проверите идентичността на отделните записи. Така ще установите дали връзката между системите е била достатъчно стабилна и дали е протекъл правилен процес на четене/записване. Можете да направите необходимото сравнение с тази команда:

```
$ md5sum Резервно_копие_1 Резервно_копие_2
```

(където под '**Резервно_копие_1**' и '**Резервно_копие_2**' разбираме отделните свалени записи). В терминала ще бъде изведен резултат – проверовъчните суми от символите в двата файла, явяващи се криптографска производна на алгоритъма **MD5**. Ако проверовъчните суми съвпадат напълно (проверете символ по символ!), значи отделните свалени записи са идентични – твърде малка е вероятността два пъти да е настъпила напълно еднаква грешка, която да се отрази в напълно идентична грешка при копирането! Това гарантира, че в последствие, ако нещо се объркa, ще

можете да върнете свалените запис на **BIOS/UEFI** flash-чипа, за да не остане компютърът неизползваем, ако по някаква причина се окаже, че не може да бъде стартиран с **Libreboot**.

След като се убедите, че разполагате с резервно копие от заварения запис на **BIOS/UEFI**, идва ред да се снабдите с необходимия ви **Libreboot**-софтуер, с който да препрограмирате flash-чипа. **Libreboot** поддържа версии за компютрите **x200** и **x200s** с чипове, чийто капацитет е **4MiB**, **8MiB** или **16MiB**. В приведения по-горе пример установихме, че чипът е с капацитет **8MiB** (записа '**8192 kB**'). За пълнота тук е мястото да отбележим, че съществува известно технологично разминаване между мерните единици (**1KiB** = **1024B**; **1MiB** = **1024KiB**; **1kB** = **1000B**; и **1MB** = **1000kB**) – така **8MiB** (често изписвано неправилно като „**8MB**“) съответства на **8192 kB**). Нашата версия на **Libreboot** трябва да съответства на капацитета на нашия flash-чип и да поддържа необходимата ни клавиатурна подредба (препоръчваме '**usqwerty**' американската стандартизация на английския език). Независимо от разминаването, когато бъде изведен подобен на горния резултат, ще се ориентирате лесно за актуалния капацитет на вашия flash-чип „чрез закръгляне“ и от там ще се насочите към версия на **Libreboot**, подходяща за случая.

След като вече знаете капацитета на **BIOS/UEFI** flash-чипа, можете да изтеглите подходяща версия на софтуера **Libreboot** – от предпочитания от вас огледален сървър. Актуалните адреси на огледални сървъри (възможно е да се променят във времето) вижте на този адрес:

```
https://libreboot.at/download.html#https
```

От предпочитания огледален сървър се препоръчва да изберете директорията със '**Stable**' (стабилни) разработки и да ползвате най-актуалната предложена версия. Там ще намерите директорията с програми за flash-чипове '**roms**' където се съхраняват голям брой **.tar.xz** файлове с архивни масиви, всеки от които съответства на определена марка и модел компютри, респективно – на определен капацитет на **BIOS/UEFI** flash-чипа. В дадения пример с модела **x200** (еквивалентно приложим и при **x200s**) и flash-чип с капацитет **8MiB** трябва да се ориентирате към този от предложените архивни масиви (във времето може да има актуализации):

```
libreboot-20230625_x200_8mb.tar.xz
```

(където под '**20230625**' разбираме, че софтуерът е обновен последно на **25.06.2023** г.; под '**x200**' – разглеждания от нас модел **Lenovo Thinkpad x200**; и под '**8mb**' – неправилно изписване на капацитета на flash-чипа, който в нашия случай е **8MiB**).

Можете да свалите интересуващия ви пакет чрез тази команда:

```
wget web-сайт/директории/stable/20230625/roms/\
libreboot-20230625_x200_8mb.tar.xz
```

(където под '**web-сайт**' и '**директории**' разбираме адреса на съответния огледален сървър и интересуващите ни директории в него, а под '**libreboot_r20160907_grub_x200_8mb.tar.xz**' – наименованието на интересуващия ни софтуерен пакет в съответния огледален сървър и неговите директории. Горната команда в завършен вид може да добие подобен вид:

```
wget https://mirrors.mit.edu/libreboot/stable/20230625/roms/\
libreboot-20230625_x200_8mb.tar.xz
```

(В случай, че имате проблеми с изписването на дългата команда в терминала, можете да въведете командата в обикновен текстови редактор и да я копирате наготово в терминала, като не забравяте да ползвате **[Ctrl]**+**[v]**+**[Shift]** при поставянето.)

След като свалите необходимия ви файл, трябва най-напред да проверите неговата автентичност и интегритет – за която цел можете да използвате от файловете '**libreboot-20230625_x200_8mb.tar.xz.sha512**' и '**libreboot-20230625_x200_8mb.tar.xz.sig**', които се намират в същата директория, откъдето изтеглихте архива (подробно на проверката за автентичност и интегритет се спряхме в Част **II** по-горе).

След като се убедите в автентичността и интегритета на интересуващия ви файл, вече можете да го разархивирате чрез командата '**`tar xf`**'. Ще видите, че в архивния масив за модел **x200** с капацитет на flash-чипа **8MiB** съществуват можество **Libreboot**-пакети. '**`grub_x200_8mb_libgfxinit_corebootfb_usqwerty_noblobs.rom`**' е пакетът, който препоръчваме.

Когато вече разполагате с необходимия ви **Libreboot**-патек, можете да пристъпите към самото препрограмиране:

```
# flashrom -c Чип -p serprog:dev=/dev/ttyUSB0:115200 -w Libreboot_пакет
```

(където под '**`Libreboot_пакет`**' разбираме версията на **Libreboot**, определена за препрограмиране на flash-чипа – например '**`grub_x200_8mb_libgfxinit_corebootfb_usqwerty_noblobs.rom`**').

Ако чипът не бъде намерен, при опита да бъде препрограмиран, ще бъде изведен подобен резултат:

```
flashrom -c MX25L6405D -p serprog:dev=/dev/ttyUSB0:115200 -w
Адрес/Libreboot_пакет
flashrom v1.2 on Linux 5.15.88-gnu-1-lts (x86_64)
flashrom is free software, get the source code at https://flashrom.org

Using clock_gettime for delay loops (clk_id: 1, resolution: 1ns).
serprog: Programmer name is "serprog-duino"
No EEPROM/flash device found.
Note: flashrom can never write if the flash chip isn't found automatically.
```

Ако чипът е намерен и препрограмиран успешно, ще бъде изведен подобен резултат:

```
flashrom -c MX25L6405D -p serprog:dev=/dev/ttyUSB0:115200 -w
Адрес/Libreboot_пакет
flashrom v1.2 on Linux 5.15.88-gnu-1-lts (x86_64)
flashrom is free software, get the source code at https://flashrom.org

Using clock_gettime for delay loops (clk_id: 1, resolution: 1ns).
serprog: Programmer name is "serprog-duino"
Found Macronix flash chip "MX25L6405D" (8192 kB, SPI) on serprog.
Reading old flash chip contents... done.
Erasing and writing flash chip... Erase/write done.
Verifying flash... VERIFIED.
```

Възможно е при препрограмирането да бъде изведена грешка, но въпреки това системата да работи коректно. Желателно е да за пореден път да свалите копие от записа на flash-чипа и да сравните с посочената по-горе команда '**`md5sum`**' дали е идентичен с версията на **Libreboot**, която току що подадохте за записване на flash-чипа. Ако бъде изведен положителен криптографски отговор, това гарантира коректно протичане на процеса; отрицателният криптографски отговор обаче не винаги съответства на истината и понякога въпреки него нещата работят.

Следователно – дори да бъде изведен отрицателен криптографски отговор, все пак опитайте да стартирате препрограмирания компютър. Върнете **RAM**-чиповете в слотовете на дънната платка, наместете самата платка удобно, закачете лентовите кабели на екрана и на клавиатурата към нея, включете електрозахранването (не батерията, а захранващия кабел) и стартирайте.

Ако стартирането премине успешно (ще видите опростено стартово меню на програмата **GRUB**, вградена в **Libreboot** за стартиране на системата), можете да препрограмирате повторно с желаната версия на **Libreboot** – но вече не чрез микроконтролер и помощен компютър, а направо от терминала на избраната от вас Свободна операционна система, с която работи (макар и неправилно) препрограмираният компютър.

Препрограмирайте повторно flash-чипа така:

 # flashrom -p internal -w Libreboot_пакет

(където под '**Libreboot_пакет**' вече разбираме версията на Libreboot, качена в самия препрограмиран компютър, от чийто терминал подавате горната команда). Този директен подход е възможен, след като компютърът вече е бил препрограмиран с **Libreboot** (в много от случаите включително и при отрицателен криптографски отговор за първоначалното препрограмиране на flash-чипа). По този начин се избягват редица усложнения от свръзките между пиновете на микроконтролера и flash-чипа – които биха могли да опорочат процеса по препрограмиране. Специалисти препоръчват при съмнения да се ползва именно този подход, дори и след успешно препрограмиране – за да сме сигурни, че инсталацията е осъществена изрядно и във времето няма да прояви скрити проблеми. Следва да имате предвид, че препрограмиране от терминала можете да извършите, само ако при стартиране на системата натиснете **[e]** (от **E**[**dit**]), за да отворите режима за редактиране на **GRUB**; въведете в края на реда '**linux**' отдолу този параметър:

iomem=relaxed

... и продължите със зареждане на системата, като натиснете **[Ctrl]**+**[x]** или **[F10]** (вижте поясненията на екрана най-отдолу). До следващото си изключване flash-чипът ще бъде в режим, позволяващ неговото препрограмиране. Тук е мястото да отбележим, че при препрограмирането от терминала е удобен момент да смените **MAC**-адреса на системата (който при всички **Libreboot** инсталации по подразбиране е еднакъв – и това освен че разкрива с какъв софтуер от „ниско“ ниво работи вашата система, също създава предпоставки за конфликт, ако в мрежата ви се закачи и друго устройство със същия **MAC**-адрес). На този въпрос обаче ще се спрем по-подробно в Част **VI** във връзка със сигурността в internet.

Тук е мястото да отбележим и това, че докато все още препрограмираният компютър е разглобен и имате физически достъп до flash-чипа, можете да блокирате възможностите за последващото му препрограмиране (с оглед обезпечаване сигурността на софтуера от „ниско“ ниво). На този въпрос обаче ще се спрем едва в Част **VII** – след като в Част **IV** вече сме инсталирали напълно шифрована система (тъй като цялостното шифроване изисква и някои настройки на **Libreboot**); и след като вече сме разгледали въпроса за паролите в Част **V** (тъй като в някои случаи в **Libreboot** се налага да бъдат вграждани определени пароли, които няма да е лесно да променяте след блокирането на възможността за препрограмиране).

Тук трябва да отбележим също, че на този етап **Libreboot** не е настроен да стартира вашата Свободна операционна система. Ето защо трябва да укажете ръчно как да стане това. При стартиране на **Libreboot** натиснете **[c]** (от **C**[**ommand**]), за да влезете в опростения команден ред (ще бъде изведен индикатор '**>**' в началото на реда). При всички случаи ще можете с команда '**ls**' да се ориентирате за наличните възможности. Трябва най-напред да укажете къде се намира стартиращата програма:

> root=ahci0

(символът '**>**' е индикатор за командния ред на **Libreboot**; не е част от командата и не трябва да го въвеждате); под '**ahci0**' разбираме твърдия диск (ако има закачени няколко твърди диска, числото '**0**' в края може да е '**1**', '**2**' и т.н.).

Следва да укажете местоположението и параметрите на системното ядро:

> linux /boot/vmlinuz-linux-libre-lts root=/dev/sda1 rw

... да укажете къде се намира стартиращата система, която да бъде заредена в **RAM** -паметта на компютъра:

> initd /boot/initramfs-linux-libre.img

... и накрая да стартирате зареждането на операционната система с въведените по-горе параметри:

```
> boot
```

По-долу ще конфигурираме **Libreboot** така, че да не се налага да въвеждате тези команди ръчно при всяко стартиране на системата.

7. Архитектура и функционалности на компютъра Beagle Bone Black

BeagleBone Black rev C е алтернативен едноплатков компютър, който също позволява подаването на директни инструкции към интересуващите ни хардуерни компоненти. Интерфейсът му включва вход за електрозахранване (**5V**), **LAN**, **MiniB USB** и **USB 2.0**, **HDMI**-изход и слот за **MicroSD**-карта (която служи на едноплатковия компютър като „твърд диск“). Заедно с това **Beagle Bone Black** е снабден с две гнездови рейки (всяка от които с по **46** „женски“ щифта) и една щифтова рейка (с **6** „мъжки“ пина). Това е повече от достатъчно за нашите цели.

Схема на интерфейса на едноплатковия компютър Beagle Bone Black

Всеки от щифтовете и пиновете в рейките на едноплатковия компютър отговаря за определена специфична функция. Нас ни интересуват следните: от щифтовата рейка (от която ще свържете помощен компютър с едноплатковия компютър) – пин **x1** (**GND**; **ground** – затваря електрическата верига), пин **•4** (**TXD**; **output** – изпраща изходящи данни), и пин **•5** (**RXD**; **input** – приема входящи данни); и от лявата гнездова рейка (от която ще подадете

необходимите инструкции към препрограмируемия **BIOS/UEFI** flash-чип) – щифт **x1** (**dgnd** – затваря електрическата верига), щифт **•3** (**vdv_3v3**; **vcc 3v** – осигурява слаботоково захранване на системата), щифт **•17** (**spi0_cs0 cabel select** – определя с кои устройства да се осъществява връзка), щифт **•18** (**spi0_d1**; **MOSI**; изпраща изходящи данни), щифт **•21** (**spi0_d0**; **MISO** – приема входящи данни), и щифт •22 (**spi0_sclk serial clock** – определя скоростта за обмен на данните).

Тук трябва да направи впечатление, че пин **1** от щифтовата рейка и щифт **1** от лявата гнездова рейка отбелязваме със символ '**x**' (вместо със символ '•', както сме постъпили при останалите отбелязани пинове и щифтове). Това е така, защото пин **1** и щифт **1** затварят електрическата верига – при свързването им в системата започва да тече електричество. Следователно – трябва да бъдат свързвани последни (след като се убедите много внимателно, че едноплатковият компютър е поставен на стабилна основа, чиято повърхност не е електропроводима и всичките други пинове и щифтове са свързани правилно); по време на работа (докато пин **1** и щифт **1** са свързани) не трябва да местите едноплатковия компютър и никой друг от компонентите, участващи в процеса; съответно пин **1** и щифт **1** трябва да бъдат откачени първи, след като процесите приключат (преди да правите каквото и да било друго). В противен случай рискувате да причините токов удар (макар и слаботоков), което може да блокира или повреди някои от компонентите на системата.

8. Инсталиране на устройството Beagle Bone Black с помощния компютър

За да можете да укажете на едноплатковия компютър да подаде необходимите инструкции към **BIOS/UEFI** flash-чипа, които ще доведат до премахване на несвободната **BIOS/UEFI** програма и до препрограмиране със свободната алтернатива **Libreboot**, е необходимо най-напред да инсталирате едноплатковия компютър с подходяща операционна система, за да можете от нея да подадете необходимите команди към микроконтролера; от който на свой ред същите ще бъдат изпратени като сигнали към съответни пинове на flash-чипа.

Може да инсталирате някоя от опростените Свободни операционни системи, препоръчани от екипа на **Beagle Bone Black** на този адрес:

https://beagleboard.org/distros

От предложения списък препоръчваме **GNU/Linux** операционната система **Debian** (софтуерния пакет '**AM3358 Debian 10.3 2020-04-06 1GB SD console**'). Предимство на предложената операционна система е нейната пълна съвместимост с **AM3358**-чиповете на едноплатковия компютър, възможността да я инсталирате на **MicroSD**-памет с капацитет **1GB** и да я ползвате от терминал. Преди да инсталирате, ако изтегленият файл е компресиран (примерно завършва с разширение '**.xz**'), първо разкомпресирайте с командата '**unxz**' (или друга команда според компресията).

След като имате изтеглен и разкомпресиран файла, може да го поставите **MicroSD**-картата в **SD**-слот на помощния компютър (или в подходящ **USB**-преходник), и да инсталирате:

```
# dd if=Файл of=/dev/Устройство bs=64M status=progress oflag=sync
```

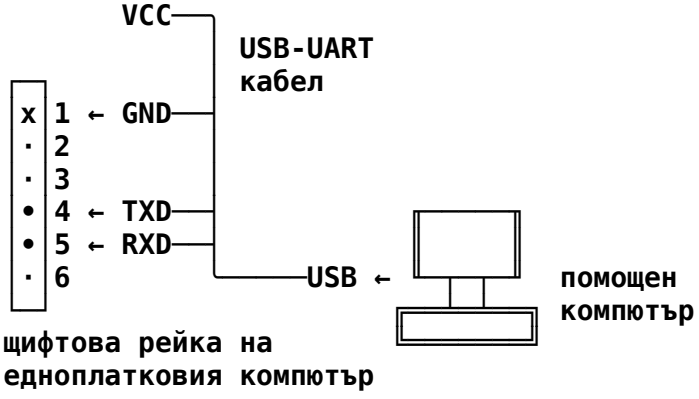
(където под '**Файл**' разбираме сваления инсталационен файл на избраната съвместима операционна система, а под '**Устройство**' – **MicroSD**-картата съгласно наименованието, под което я разпознава вашата система).

Инсталационният процес ще отнеме от няколко до десетина минути. Не забравяйте да завършите с командата '**sync**'.

Контролирането на едноплатковия компютър може да се осъществи посредством помощния компютър. За тази цел можете да ползвате терминала на помощния компютър, като го настроите да показва осъществяваните в едноплатковия компютър процеси и да приема въвеждането на команди към него. Тази задача можете да изпълните с програмата **Screen** (която ви позволява след като свържете двете устройства чрез **USB-UART**-кабел и пуснете електрозахранване към едноплатковия компютър, да следите от терминала на помощния компютър процесите в

едноплатковия компютър и да го управлявате от там). Инсталирайте програмата **Screen** чрез командата '**`pacman -S screen`**'.

След като вече програмата **Screen** е инсталирана на помощния компютър, а инсталираната с операционна система **MicroSD**-карта е поставена в съответния слот на едноплатковия компютър, можете да свържете едноплатковия компютър към помощния компютър чрез **USB-UART** кабела – като закачите **USB**-изхода на кабела в някой от **USB**-портовете на помощния компютър, а „женските“ щифтове на трите **Serial**-разклонения – към щифтовата рейка на едноплатковия компютър, както следва: разклонението **GND** – към пин **x1**; разклонение **TXD** – към пин **•4**; и разклонение **RXD** – към пин **•5** (обикновено наименованията на разклоненията са записани на самите щифтове или са дадени цветови обозначения към документацията на кабела). Ако кабелът има и четвърто разклонение **VCC**, то няма да ви е необходимо и можете да го изолирате и оставите свободно. Както стана дума по-горе, разклонението **GND** трябва да свържете към пин **x1** най-накрая, след като стабилизирате платката на компютъра върху равна и електронепроводима повърхност, и се убедите в правилността на останалите свръзки.



Свързване на помощния компютър с компютъра Beagle Bone Black

След като свържете едноплатковия компютър към помощния компютър, можете да подадете захранване от електромрежата с адаптор от **5V** (или чрез кабел от **USB** към **MiniUSB** между някой от свободните **USB**-портове на помощния компютър и входа **MiniUSB** на едноплатковия компютър). Преди да сторите това, се убедете още веднъж, че едноплатковият компютър е стабилизиран върху равна електроизолираща повърхност, която няма да размествате докато приключите работата си и е свързан правилно с помощния компютър.

Ако подавате захранване към едноплатковия компютър от помощния компютър, включете помощния компютър (ако е преносим, работещ с батерия) към електромрежата – тъй като понякога енергоспестяващите системи на преносимите компютри намаляват разхода енергия, подаван към **USB**-портовете и това може да попречи за достатъчното захранване на едноплатковия компютър).

След като свържете едноплатковия компютър към помощния компютър с **USB-UART** кабела (все още не подавайте ток на едноплатковия компютър), можете да стартирайте програмата **Screen** в терминала на помощния компютър:

```
$ screen /dev/ttyUSB0 115200
```

Ако към помощния компютър сте закачили и други **USB-UART** кабели, е възможно изразът '**`ttyUSB0`**' да се наложи да бъде изписан с друга цифра, съответстваща на поредния номер на съответния закачен кабел (като се започва от **`0`**).

Ако не можете да стартирате програмата **Screen**, причината може да е в липсата на необходимите права на достъп – въдете командата '**`chmod +rw /dev/ttyUSB0`**' (като потребител с администраторски права) и опитайте пак да стартирате.

Следва да подадете захранване към едноплатковия компютър (с адаптор от **5V** от електромрежата или с кабел от **USB** към **MiniUSB** между някой от свободните **USB**-портове на помощния компютър и входа **MiniUSB** на едноплатковия

компютър). Преди да сторите това, се убедете още веднъж, че едноплатковият компютър е стабилизиран върху равна електроизолираща повърхност, която няма да размествате докато приключите работата си и е свързан правилно с помощния компютър.

Ако подавате захранване към едноплатковия компютър от помощния компютър, включете помощния компютър (ако е преносим, работещ с батерия) към електромрежата – тъй като понякога енергоспестяващите системи на преносимите компютри намаляват разхода енергия, подаван към **USB**-портовете и това може да попречи за достатъчното захранване на едноплатковия компютър).

За да може да стартирате от **microSD** картата (където е инсталираната операционната система), трябва преди да подадете захранването да задържите бутона за стартиране от **microSD**-карта на едноплатковия компютър (това бутонът, който е най-близък до **microSD**-слота). Докато задържате бутона, подайте захранване към едноплатковия компютър, след което можете да отпуснете бутона. Ако процедурата е изпълнена правилно, ще забележите четирите най-близки до **MiniUSB** порта светодиодни индикатори да премигват. Ако светят само два от четирите светодиода, това може да е знак, че стартирането не е извършено правилно (изключете захранването и пробвайте отново).

Ако сте изпълнили процедурата за стартиране от **MicroSD**-карта правилно, на помощния компютър в програмата **Screen** ще видите процеса по зареждане на системата на едноплатковия компютър, след което ще бъде изведено указание '**`login:`**', в очакване да въведете потребителско име. Ако едноплатковият компютър работи с препоръчаната версия на **Debian** и последната е с изходните си настройки, можете да въведете '**`root`**' и след като натиснете **`[Enter]`**, системата ще изведе указание '**`password:`**' (където трябва отново да въведете '**`root`**' и да натиснете **`[Enter]`**). Системата ще ви възприеме като администратор на едноплатковия компютър. Възможно е и директно да бъдете възприети като администратор (което ще бъде обозначено с '**`root@beaglebone:~#`**'). Ако сте инсталирали друга предпочитана от вас съвместима операционна система, трябва да се съобразите с нейната документация и настройки.

Друг начин да се свържете с едноплатковия компютър е например чрез софтуера **OpenSSH**, който можете да инсталирате на вашата система чрез командата '**`pacman -S openssh`**'. След като едноплатковият компютър бъде захранен с електричество и разполагате с необходимия софтуер, можете да осъществите **SSH** достъп посредством **USB** към **MiniUSB** кабела:

```
$ ssh debian@IP_адрес
```

(където под '**`IP_адрес`**' разбираме **IP**-адреса на едноплатковия компютър, под който същият е разпознат от помощния компютър. IP-адресът включва четири едноцифрени, двуцифрени и/или трицифрени числа, отделени едно от друго с точки, последното от които '**`1`**' и има подобен вид: '**`192.168.7.1`**'. Можете да извлечете **IP**-адреса чрез командата '**`ip a`**' – ще бъдат изведени всички активни мрежови връзки, сред които (в повечето случаи най-отдолу) и IP-адреса на връзката с едноплатковия компютър (започващ с израза '**`inet`**'). Трябва да замените последното число от интересуващия ви **IP**-адрес с '**`2`**' – командата в крайна сметка ще има подобен вид:

```
$ ssh root@192.168.7.2
```

При успешно свързване с едноплатковия компютър ще бъде изведен подобен резултат:

```
The authenticity of host '192.168.7.2 (192.168.7.2)' can't be established.
ED25519 key fingerprint is
SHA256:vOIQpcq/ASHFGqK03b7YibTquSA2TYAoMJfU7Itapg0.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])?
```

(при което системата указва да въведете '**yes**' и да натиснете **[Enter]**, ако наистина желаете да бъде установена връзка между помощния компютър и едноплатковия компютър). В случай, че работите с едноплатковия компютър **Beagle Bone Black** и ползвате споменатата от нас версия на операционната система **Debian**, при успешно свързване с едноплатковия компютър ще бъде изведен подобен резултат:

```
Warning: Permanently added '192.168.7.2' (ED25519) to the list of known
hosts.
Debian GNU/Linux 10
BeagleBoard.org Debian Buster Console Image 2020-04-06
Support: http://elinux.org/Beagleboard:BeagleBoneBlack_Debian
default username:password is [debian:temppwd]
debian@192.168.7.2's password:
```

(при което системата ще влезе в режим на очакване да въведете парола '**temppwd**' (както е посочено в цитираното съобщение). След като сторите това, ще бъде изведен индикатор, че системата на едноплатковия компютър ви възприема като потребител без администраторски права:

```
debian@beaglebone:~$
```

(където под '**debian**' разбираме потребителя на едноплатковия компютър, а под '**beaglebone**' – наименованието, под което системата разпознава самото устройство).

Можете да инициирате сесия с администраторски права чрез командата '**su**' и парола '**root**'.

В случай, че желаете да прекратите сесията като администратор на едноплатковия компютър, можете да въведете **[Ctrl]+[d]**, при което ще се върнете към потребителския акаунт '**debian**' без администраторски права. Ако натиснете отново **[Ctrl]+[d]**, ще прекратите **SSH**-сесията с едноплатковия компютър. Ако връзката с едноплатковия компютър блокира и командата за прекратяване на сесията не работи, може да прекъснете връзката с **[Enter]**, след което натиснете **[~]** и **[.]**. Ако искате да изключите едноплатковата система, добре е да укажете софтуерно преустановяване на захранването, за да ограничите риска от повреждане на операционната система в едноплатковия компютър – докато все още сте в сесия с администраторски права на едноплатковия компютър, въведете тази команда:

```
# /sbin/poweroff
```

Независимо, че почти веднага ще бъдете върнати в терминала на помощния компютър, все пак изчакайте светлинните индикатори на едноплатковия компютър да угаснат, преди да изключите захранването му.

Следва да имате предвид няколко особености. При свързване с едноплатковия компютър чрез **USB-UART** кабел често се забелязват дефекти в терминала – разместване на редовете, поява на произволни символи и други. Такива дефекти не се проявяват при свързване чрез **OpenSSH**. В замяна на това обаче, при свързване чрез **USB-UART** кабел ще получите достъп веднага, дори преди стартирането на операционната система на едноплатковия компютър – това позволява да проследите процеса по стартирането и да установите евентуални проблеми, ако възникнат такива. При свързване чрез **OpenSSH** няма да получите достъп преди операционната системата на едноплатковия компютър да бъде стартирана успешно.

Но независимо от това кой от двата начина за свързване с едноплатковия компютър ще изберете, сега е моментът да инсталирате програмата '**Flashrom**', която е отговорна за комуникацията с вашия **BIOS/UEFI** flash-чип. За да инсталирате програмата, е нужно да свържете едноплатковия компютър към internet, като най-лесният начин за това е чрез **LAN**-кабел, свързващ едноплатковия компютър към вашия рутер (връзката ще се осъществи автоматично).

След като едноплатковият компютър бъде свързан към internet, може да обновите софтуерните пакети на неговата операционна системата:

```
# apt update
```

... и сетне да инсталирате **Flashrom**:

```
# apt install flashrom
```

След като вече имате инталирана програмата **Flashrom**, може да изключите едноплатковия компютър (командата '**`# poweroff`**' и изключване на захранването след угасване на светодиодните индикатори) и да пристъпите към свързване с flash-чипа, за да бъде той препрограмиран.

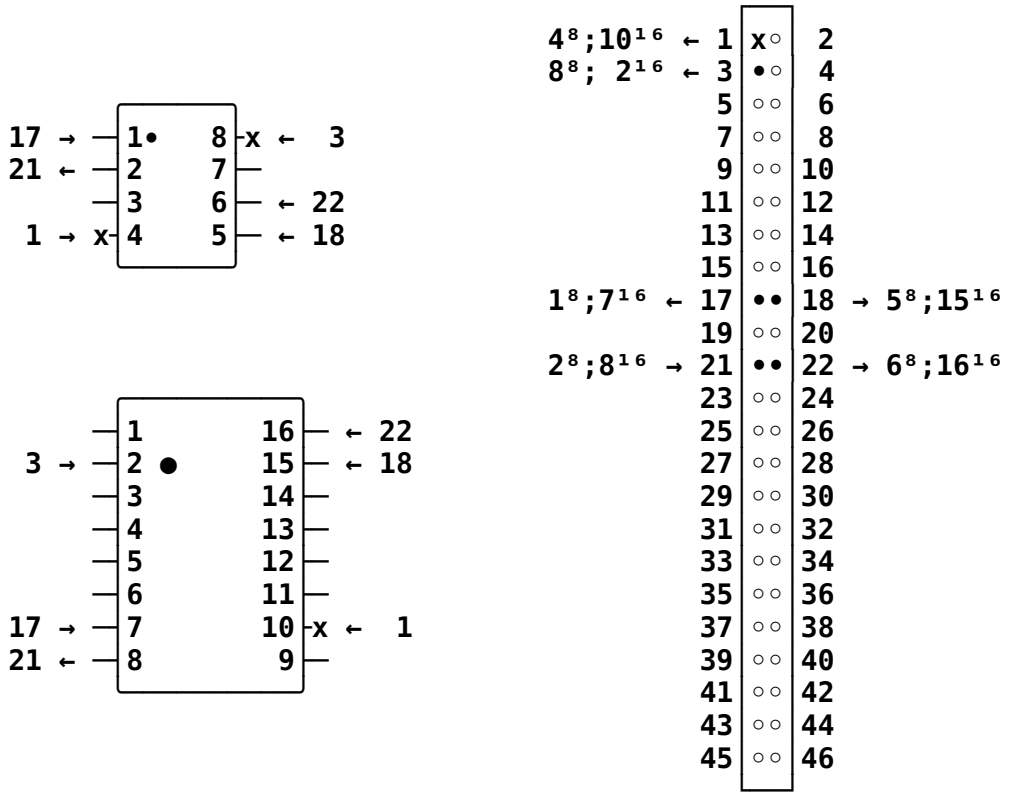
Свръзването между едноплатковия компютър и flash-чипа ще стане посредством част от щифтовете на лявата гнездова рейка, към които ще закачите **6**-**те** късички съединителни кабела (с „мъжки“ и „женски“ щифтове) и вашата **SOIC**-щипка за захващане на flash-чипа (която е с **8** или с **16** пина, в зависимост от конфигурацията на вашия **BIOS/UEFI** flash-чип). „Женските“ щифтове на съединителните кабели ще свържете със съответните пинове на **SOIC**-щипката (която ще захванете плътно и симетрично за пиновете на flash-чипа), а „мъжките“ щифтове на съединителните кабели ще свържете със съответните щифтове в лявата гнездова рейка на едноплатковия компютър.

Ако не разполагате с такива пособия или вашият **BIOS/UEFI** flash-чип е с твърде малки пинове, за да бъдат захванати от **SOIC**-щипка, ще осъществите свързването посредством описаните по-горе **10** cm тънки емайлирани проводници, единия край на които ще запоите към съответните пинове, а другия край към съединителнителни кабели, които ще включите към съответния щифт в лявата гнездова рейка.

При всички положения свързването трябва да стане по следната схема (като не забравяте, че щифт **x1** на гнездовата рейка затваря електрическата верига – поради което трябва да го свържете след всички останали и да го разкачите преди всички останали, за да ограничите риска от токови смущения):

– при **BIOS/UEFI** flash-чип с **8** пина: пин **1** от чипа към щифт **•17** на рейката, пин **2** към щифт **•21**, пин **4** към щифт **x1**, пин **5** към щифт **•18**, пин **6** към щифт **•22**, и пин **8** към щифт **•3**;

– при **BIOS/UEFI** flash-чип с **16** пина: пин **2** от чипа към щифт **•3** на рейката, пин **7** към щифт **•17**, пин **8** към щифт **•21**, пин **10** към щифт **x1**, пин **15** към щифт **•18**, и пин **16** към щифт **•22**.



Свързване на BIOS/UEFI flash-чипа с компютъра Beagle Bone Black

След като **BIOS/UEFI** flash-чипът вече е свързан физически към едноплатковия компютър съгласно приложената схема, идва ред да подадете електрозахранване. Преди да сторите това, проверете още веднъж дали едноплатковият компютър е поставен върху стабилна, равна и електронепроводима повърхност, и дали свързването е правилно (като закачите последен щифт **x1** на гнездовата рейка). От този момент нататък трябва да ограничите всякакви разместванията и сътресения на повърхността, върху която стоят едноплатковият компютър и дънната платка с flash-чипа. Дори най-малки колебания биха могли да доведат до кратковременно прекъсване на връзките и това да провали препрограмирането, а в някои случаи – дори да нанесе поражения на компонентите.

Когато вече всичко е готово, можете да установите достъп до едноплатковия компютър от терминала на помощния компютър:

```
$ screen /dev/ttyUSB0 115200
```

... или съответно:

```
# ssh root@IP_адрес
```

След като **BIOS/UEFI** flash-чипът вече е свързан към вашия едноплатковия компютър, идва ред да тествате дали свързането е правилно и да установите какъв е капацитетът на flash-чипа. Това ще ви позволи да коригирате свързването (ако има проблем) и да подберете подходящата за капацитета на вашия flash-чип версия на **Libreboot**. Първо конфигурирайте **SPI**-връзката, която е нужна за комуникация с flash-чипа – със следната поредица от четири команди (които трябва да въвеждате при всяко следващо стартиране на едноплатковия компютър, за да можете да ползвате програмата **Flashrom**):

```
# config-pin p9_17 spi_cs

# config-pin p9_18 spi

# config-pin p9_21 spi

# config-pin p9_22 spi_sclk
```

Горните четири команди трябва да бъдат въвеждани при всяко следващо стартиране на едноплатковия компютър, за да можете да ползвате програмата **Flashrom**. След като сте изпълнили това, можете да пристъпите към проверка на свързването с **BIOS/UEFI** flash-чипа:

```
# /usr/sbin/flashrom -p linux_spi:dev=/dev/spidev0.0,spispeed=512
```

Ако системата отчете някаква грешка (например проверяваният **BIOS/UEFI** flash-чип не е свързан правилно), ще бъде изведен резултат, че няма открито препрограмируемо устройство (**`No EEPROM/flash device found`**). В противен случай (ако системата работи изправно) ще бъде изведен подобен резултат:

```
Calibrating delay loop... OK.
Found Macronix flash chip "MX25L6405D" (8192 kB, SPI) on linux_spi.
No operations were specified.
```

Вашият flash-чип (в дадения пример това е моделът '**`MX25L6405D`**') би могъл да бъде с капацитет **4MiB**, **8MiB** или **16MiB** (в горния пример под '**8192 kB**' разбираме, че проверяваният flash-чип е с капацитет **8MiB**).

9. Препрограмиране на BIOS/UEFI flash-чипа с Beagle Bone Black

След като вече знаете капацитета на **BIOS/UEFI** flash-чипа, можете да изтеглите подходяща версия на софтуера **Libreboot** – от предпочитания от вас огледален сървър, който можете да изберете от споменатия по-горе адрес:

```
https://libreboot.at/download.html#https
```

Както вече проследихме, от предпочитания огледален сървър трябва да изберете директорията със '**`Stable`**' (стабилни) разработки и да ползвате най-актуалната предложена версия. Там ще намерите директорията с програми за flash-чипове '**`roms`**' и сред **tar.xz** файловете с архивни масиви трябва да изберете вашата марка и модел компютър и вашия капацитет на **BIOS/UEFI** flash-чипа (в дадения пример – модела **x200** (еквивалентно приложим и при **x200s**) и flash-чип с капацитет **8MiB** – на което съответства масивът '**`libreboot-20230625_x200_8mb.tar.xz`**'). Следва да имате предвид, че последно обновеният на **25.06.2023** г. софтуер във времето ще бъде актуализиран с по-нова версия.

Можете да свалите интересуващия ви пакет чрез тази команда:

```
wget web-сайт/директории/stable/20230625/roms/\
libreboot-20230625_x200_8mb.tar.xz
```

... или примерно в завършен вид:

```
wget https://mirrors.mit.edu/libreboot/stable/20230625/roms/\
libreboot-20230625_x200_8mb.tar.xz
```

Следва да проверите автентичността и интегритета на свалените архивен масив (за която цел ще ползвате **`.sha512`** и **`.sig`** файловете в същата директория, както проследихме в Част **`II`** по-горе).

След като се убедите в автентичността и интегритета на архивния масив, можете да разархивирате (командата '**tar xf**' и да намерите необходимата ви версия на Libreboot-пакета (в дадения пример това е версията '**grub_x200_8mb_libgfxinit_corebootfb_usqwerty_noblobs.rom**').

След като вече разполагате с разархивирано копие от необходимия **Libreboot**-пакет, можете най-сетне да стартирате и самото препрограмиране на **BIOS/UEFI** flash-чипа посредством програмата **Flashrom**:

```
# /usr/sbin/flashrom -p linux_spi:dev=/dev/spidev0.0,spispeed=512 -w \
Адрес/Libreboot_пакет
```

(където под '**Адрес**' разбираме местоположението на **Libreboot**-пакета в системата, а под '**Libreboot_пакет**' – наименованието на избрания **Libreboot**-пакет).

Ако системата изведе грешка, означава, че нещо не е свързано правилно или веригата е прекъсната на някое място. В такъв случай системата ще изведе подобен резултат (несигурен изход от процеса):

```
Calibrating delay loop... OK.
Found Winbond flash chip "MX25L6405D" (8192 kB, SPI) on linux_spi.
No operations were specified.
```

или дори подобен резултат (категорично отрицателен изход от процеса):

```
Calibrating delay loop... OK.
Found Winbond flash chip "MX25L6405D" (8192 kB, SPI) on linux_spi.
Reading old flash chip contents... done.
Erasing and writing flash chip... Erase/write done.
Verifying flash... FAILED at 0x00014dc3! Expected=0x00, Found=0x68, failed
byte count from 0x00000000-0x007fffff: 0x74
Your flash chip is in an unknown state.
Please report this on IRC at chat.freenode.net (channel #flashrom) or
mail flashrom@flashrom.org, thanks!
```

Ако всичко е наред, системата след известно време ще изведе подобен резултат (положителен изход от процеса):

```
Calibrating delay loop... OK.
Found Macronix flash chip "MX25L6405D" (8192 kB, SPI) on linux_spi.
Reading old flash chip contents... done.
Erasing and writing flash chip... Erase/write done.
Verifying flash... VERIFIED.
```

Независимо от резултата, не бързайте да откачате flash-чипа от едноплатковия компютър и да сглобявате компютъра. Преди всичко изключете едноплатковия компютър:

```
# /sbin/poweroff
```

След като системата се изключи (ще познаете това по изключването на светодиодния индикатор), можете да изключите програмата **Screen** от помощния компютър, като натиснете едновременно **[Ctrl]+[a]** (системата ще влезе в режим на готовност) и след това натиснете **[k]** (системата ще укаже да потвърдите намерението си за излизане), за което трябва да натиснете **[y]**. Когато сторите това, първо откачете захранването на едноплатковия компютър, след което откачете flash-чипа от едноплатковия компютър. Не бързайте да откачате съединителните кабели (емайлираните проводници) откъм самия flash-чип (тъй като след малко може да се наложи да ги ползвате отново), но се уверете, че никой от кабелите (проводниците) не дава контакт (не се допира) до някаква електропроводима повърхност.

Върнете **RAM**-чиповете в слотовете на дънната платка, наместете самата платка удобно, закачете лентовите кабели на екрана и на клавиатурата към нея, включете електрозахранването (не батерията, а захранващия кабел) и стартирайте току що препрограмирания компютър.

Ако компютърът не стартира (познатото изображение с логото на **Libreboot** и опростено стартово меню на програмата **GRUB**, вградена в **Libreboot** за стартиране на системата), трябва да опитате да осъществите препрограмирането отново. Както вече проследихме по-горе, може да ви се наложи да пробвате правилната свързаност на ползваните от вас кабели (или емайлирани проводници) и тяхната проводимост, да инсталирате повторно съвместимата операционна система **Debian** в едноплатковия компютър, да свалите отново **Flashrom** и необходимата ви версия на **Libreboot**, и да повторите всичките операции отново, преди да получите очаквания положителен резултат.

Ако на вашия екран обаче бъде изведено изображението с логото на **Libreboot** (процесът е протекъл успешно), можете да изключите компютъра (изберете в основното меню на **Libreboot** възможността '**Poweroff**', достъпна и с просто натискане на **[p]**), да го разкачите от едноплатковия компютър и да го сглобите. Тук отново ще напомним, че е добре от терминала на вече препрограмирания компютър да извършите препрограмирането на flash-чипа повторно с командата '**flashrom -p internal -w Libreboot_пакет**', за да сте сигурни, че софтуерът от „ниско“ ниво е инсталиран напълно коректно; както ще напомним и това, че можете да смените **MAC**-адреса на компютъра (на който въпрос ще се спрем подробно в Част **VI**) и да блокирате възможностите за последващо препрограмиране на flash-чипа (на който въпрос ще се спрем подробно в Част **VII**).

Както вече стана дума, на този етап **Libreboot** все още не е настроен да стартира вашата Свободна операционна система. Ето защо трябва да укажете ръчно как да стане това. При стартиране на **Libreboot** натиснете **[c]** (от **C**[**ommand**]), за да влезете в опростения команден ред (ще бъде изведен индикатор '**>**' и можете да се ориентирате за наличните възможности с команда '**ls**').

Трябва най-напред да укажете къде се намира стартиращата програма:

```
> root=ahci0
```

... да укажете местоположението и параметрите на системното ядро:

```
> linux /boot/vmlinuz-linux-libre-lts root=/dev/sda1 rw
```

... да укажете къде се намира стартиращата система, която да бъде заредена в **RAM** -паметта на компютъра:

```
> initd /boot/initramfs-linux-libre.img
```

... и накрая да стартирате зареждането на операционната система:

```
> boot
```

По-долу ще конфигурираме **Libreboot** така, че да не се налага да въвеждате тези команди ръчно при всяко стартиране на системата.

IV. ЦЯЛОСТНО ШИФРОВАНЕ НА ВАШАТА СИСТЕМА

Свободните операционни системи, базирани на **GNU/Linux-libre** предоставят много висока надеждност, която в голяма степен прегражда възможностите за тяхното компрометиране. Това обаче не препятства възможността например твърдият диск с цялата записана на него информация да бъде изваден от вашия компютър и закачен на друго компютърно оборудване, специално предназначено да извлече съхраняваната информация. Когато операционната система не работи (и нейните защити съответно са изключени), е възможно и добавянето на определени софтуерни компоненти на твърдия диск, които при последващо включване на системата могат да се задействат и да я компрометират. За ограничаването на тези рискове се препоръчва цялостно шифроване на твърдия диск. След като в Част **II** проследихме инсталирането на напълно Свободна операционна система, а в Част **III** се спряхме на обезпечаването на системата с напълно Свободен софтуер, замествайки **BIOS/UEFI** програмата, вече разполагаме със сигурна компютърна среда, в която можем да стартираме надежден шифровъчен процес.

В един напълно шифрован диск е невъзможно да се добавят софтуерни компоненти, когато системата е изключена – защото няма свободно място и подобно добавяне върху част от шифрованото пространство би нарушило целостта на шифровката. Освен това извличането на информацията от твърдия диск се обезсмисля, тъй като тя при изключено състояние се съхранява в шифрован вид. Като единствена възможност при това положение остава прилагането на bruteforce технологии, с които да бъде разбита паролата за разшифроване на твърдия диск. Друг вариант е да се осъществи добре организирана атака от internet (например чрез „пробутването“ на фалшиви софтуерни пакети, съдържащи зловреден код). Следващ вариант е да се компрометират някои от хардуерните компоненти на компютъра (например като се добави устройство за „подслушване“ на клавиатурата (**Key Logger**), в т.ч. на въвежданите от нея пароли), за което е необходимо да бъде осъществен физически достъп. И последният вариант е да бъде приложена пряка сила, измама или злоупотреба с доверието на потребителя, който да бъде накаран (принуден) да предостави (въведе) своите пароли (в т.ч. и изненадващо, внезапно отстраняване от компютъра, когато например на него е стартирана сесия с администраторски права).

Трябва да прокараме разграничение между шифроването на само определена част от съдържанието (например определена поверителна информация) и цялостното шифроване на твърдия диск с всичко на него (включително софтуерните компоненти на операционната система). Вторият подход (който не отменя полезността от шифроване на конкретна поверителна информация) позволява да гарантирате неприкосновеност и на самата операционна система. Това е от критично значение за сигурността, тъй като след стартиране на операционната система се очаква именно от нея да достъпвате и обработвате своята информация. Ако системата междувременно е била компрометирана, ще се компрометира и информацията. Ето защо е необходимо да шифровате изцяло твърдия диск с всичко записано на него. Отделно от това по-долу ще предложим и шифроването на един допълнителен дял за поверителна информация, който да превърнете в своеобразна „цитадела“ на вашия компютър, действаща като „втори слой“ на информационната сигурност. Това е особено полезно, ако достъпвате допълнително шифрования дял само тогава, когато се намирате на сигурно място и системата не е свързана с internet. Последното е препоръчително, ако не можете да осигурите отделно компютърно устройство „за всекидневни цели“ и такова за обработка на поверителната информация, което стои далеч от чужди ръце и от всякакво свързване с internet.

1. Теоретична архитектура на цялостното шифроване

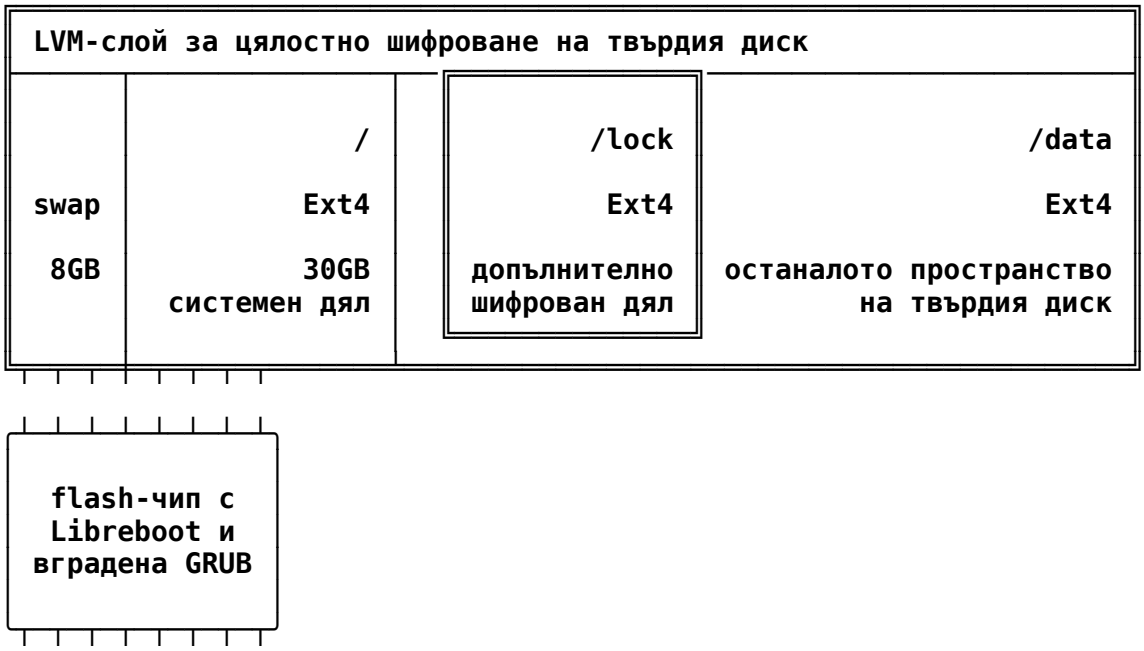
Всяко шифроване зависи от два основни компонента: шифровъчен алгоритъм (който определя механизма на криптографско преобразуване на съдържанието); и шифровъчен ключ и/или парола (които определят параметрите на криптографското преобразуване в рамките на шифровъчния алгоритъм). За да се осъществи (раз)шифроване, в системата трябва да се зареди шифровъчният алгоритъм и да се въведе паролата (и шифровъчният ключ). Това обаче не може да стане без да стартира определена минимална операционна среда, в която да се зареди шифровъчният алгоритъм и интерфейс, в който потребителят да въведе съответната парола (и шифровъчен ключ). Като такава минимална операционна среда при **GNU/Linux-libre** служи стартиращата програма **GRUB**. Тя обаче няма как да бъде заредена, ако е в шифрован вид (освен ако вече нямаме заредена друга минимална операционна среда, чрез която стартиращата програма да бъде разшифрована). На следващо място – ако имаме каквото и да било нешифровано

съдържание на твърдия диск, то може не само да бъде прочетено без разрешение, но и да бъде компрометирано – и след като разшифроваме системата, да компрометира и нея.

Възниква въпросът за това – как да осигурим цялостно шифроване на твърдия диск и все пак да стартираме **GRUB**, за да се зареди шифровъчният алгоритъм и интерфейсът, необходим на потребителя за въвеждане на паролата (и шифровъчния ключ)? Успешен отговор на този въпрос е извеждането на **GRUB** извън твърдия диск. Това позволява твърдият диск да бъде шифрован изцяло, което предотвратява компрометирането на програмното осигуряване и/ли добавянето на какъвто и да било компрометиран софтуерен компонент отвън. Каквото и да е манипулиране върху твърдия диск, докато системата е изключена и цялото съдържание е в шифрован вид, би нарушило целостта на шифровката и в най-лошия случай това ще доведе до невъзможност системата да бъде стартирана. В по-добри случаи системата все пак ще стартира, но местата, където е локализирано нововъведеното съдържание, ще бъдат повредени (след преобразуването му от шифровъчния алгоритъм то е станало неизползваемо). Добавеният компрометиран софтуерен компонент би могъл да заработи при включване на системата и да я компрометира, само ако е монтиран на самостоятелно функциониращ хардуерен „имплант“ (на което ще обърнем внимание в част **VII** във връзка с риска от включване на такива „импланти“ към системата).

Както проследихме в Част **II** във връзка с инсталирането на една Свободна операционна система, твърдият диск трябва да бъде разпределен в обособени дялове, в които да се организира операционната система. Стандартното разпределение обикновено включва помощен '**swap**' дял (с размер до два пъти вашата **RAM**), системен '**/**' дял (**20-30GB**) и потребителски '**data**' дял (останалото свободно пространство на твърдия диск). Обикновено програмата **GRUB** е разположена извън дяловете, в началото на твърдия диск (така наречения „зареждащ сектор“ (**Boot sector**)) и се стартира от вградената **BIOS/UEFI** програма. Следователно, както проследихме по-горе, този зареждащ сектор не може да бъде шифрован, понеже стандартната **BIOS/UEFI** програма няма възможност да го разшифрова (дори и да имаше възможност, ние не може да имаме доверие, че разшифроващата парола/ключ няма да бъде компроментирана от **BIOS/UEFI** програмата). Съществува възможност да оставим зареждащия сектор нешифрован, за да можем да шифроваме поне останалата част от системата, но това не ограничава значително възможностите за компрометиране на системата чрез компрометиране на зареждащия сектор. От критично значение за информационната сигурност е твърдият диск да бъде шифрован изцяло – поради която причина ще изведем **GRUB** на външен носител, като осигурим специфичен криптографски **LVM**-слой за цялостно шифроване на твърдия диск. Отделно – ще обособим още един потребителски дял, шифрован допълнително и предназначен за съхраняване на поверително съдържание (който за нуждите на изложението ще наречем '**lock**', но няма пречка да изберете и всякакво друго предпочитано от вас наименование, стига то да не дублира наименованията на други директории от Кореновата директория на системата, като препоръчваме да не въвеждате никаква специфична информация, интервали или други специални символи (освен '**-**' и '**_**') и букви, различни от **US**-стандартизацията на латиницата).

След като вашият **BIOS/UEFI** flash-чип вече е препрограмиран с **Libreboot**, той се превръща (освен всичко друго) в много подходящ външен носител на стартиращата програма **GRUB**. Нещо повече: софтуерният пакет на **Libreboot** включва по подразбиране версия на **GRUB**, която след стартиране на хардуера е настроена да включи стартиращата програма **GRUB** на вашата Свободна операционна система (която от своя страна трябва да стартира операционната система). Следователно бихме могли да настроим първата стартираща програма **GRUB** (тази, която е вградена в **Libreboot**) за това да осигури необходимата минимална операционна среда за зареждане на шифровъчния алгоритъм и осигуряване на интерфейс за въвеждане на парола (и шифровъчен ключ); да осъществи разшифроването и – когато системата вече е в четим вид – да стартира втората стартираща програма **GRUB** (тази на операционната система), която от своя страна да поеме контрола по стартиране на вашата операционна система.



Цялостно шифрован твърд диск с GRUB, изнесена на външен носител

След като изведем стартиращата програма **GRUB** на flash-чипа, можем да решим технологичния въпрос за цялостно шифроване на твърдия диск и едновременно с това – за осигуряването на минимална операционна среда, в която да се зареди необходимият за разшифроване на системата шифровъчен алгоритъм и интерфейс за въвеждане на парола (шифровъчен ключ).

2. Подготовка на системата за цялостно шифроване

За да осигурите надеждно цялостно шифроване на системата, е необходимо най-напред системата да работи с изцяло Свободен софтуер. Това гарантира, че системата работи точно по начина, по който очаквате да работи и в случай на необходимост позволява да осъществите своя независима проверка. За тази цел трябва да имате на „жив“ носител Свободна операционна система (**II**) и компютърът да е препрограмиран с **Libreboot**-софтуер (**III**). Когато тези условия са факт, можете да стартирате компютъра от „живия“ носител с избраната от вас Свободна операционна система и да започнете неговото надеждно цялостно шифроване.

След като изтеглите инсталационния **.iso** файл и се убедите в неговата автентичност и интегритет, можете да създадете „жив“ носител, да стартирате от него компютъра и да обновите софтуерните компоненти, необходими за осъществяване на инсталационния процес. За тази цел трябва да ориентирате правилно системата във времето (защото 'публичните' ключове се променят и често валидността им е ограничена към определен период) и трябва да свържете системата към internet, за да се свърже със софтуерните хранилища.

Можете да настроите времевите параметри на системата чрез проследените по-горе команди за автоматично настройване '**timedatectl set-timezone Континент/Град**' и '**systemctl enable --now systemd-timesyncd**').

Най-лесно можете да свържете системата към internet, като включите към нея **LAN**-кабел с internet-достъп. Достъпът ще бъде осъществен автоматично, като може да е нужно да изчакате малко. Ако не разполагате с **LAN**-кабел (или не желаете да се включите към него), можете да пробвате безжично свързване с проследените по-горе команди '**systemctl enable --now NetworkManager**' и след това '**nmcli dev wifi connect Мрежа password Парола**'.

За да проверите наличието на internet-свързаност, можете да въведете проследената по-горе команда '**ping Сървър**'. До минута-две връзката би трябвало да се получи (прекъснете процеса, стартиран с командата '**ping**', като натиснете **[Ctrl]+[c]**).

И накрая идва ред да обновите софтуерните пакети, необходими за инсталационния процес, чрез командата '**pacman -Syu**' и евентуално '**pacman-key --refresh-keys Адрес_на_неактуализиран_ключ**', а при необходимост – също командите '**pacman -Sy parabola-keyring archlinux-keyring**', след което '**pacman-key --refresh-keys**'.

След като горното бъде направено, за осъществяването на цялостно шифроване можете да инсталирате програмата **Cryptsetup**, която е предназначена за работа с Унифицираната шифровъчна Linux-стандартизация **LUKS** (**Linux Unified Key Setup**); и софтуерния инструмент за управление на дялове **LVM2** – чрез командата '**pacman -S cryptsetup lvm2**'.

Именно чрез **Cryptsetup** ще осъществите цялостно шифроване на вашата система. За тази цел обаче трябва да бъде подбран подходящ шифровъчен алгоритъм (за защита на системата и съхраняваното в нея съдържание) и подходящ хеш-алгоритъм (за защита на паролата, чрез която ще бъде осъществявано самото (раз)шифроване и която от съображения за сигурност не се съхранява в „четим вид“).

Поддържаните от **Cryptsetup** шифровъчни алгоритми работят в два основни режима. При първия режим поверителното съдържание се шифрова като верига от последователни информационни блокове (**Cipher Block Chaining**; **CBC**), като всеки следващ блок е криптографска производна от начина, по който е шифрован предходният блок. При втория режим поверителното съдържание се шифрова като един цялостен блок, запълващ съответното устройство или дял от устройство (**XEX-based Tweaked-codebook mode with ciphertext Stealing**; **XTS**), като шифроването не позволява „маркиране“ на първия блок и проследяване проявите на маркировката в следващите блокове.

AES (**Advanced Encryption Standard**) произхожда от белгийския блоков алгоритъм **Rijndael**. Доразвит е през **2001** г. от Националния институт по стандартизации и технологии (**NIST**) на **САЩ** на база датиращия от **1977** г. Стандарт за шифроване на данни (**Data Encryption Standard**; **DES**) и през **2003** г. се превръща в единствения официално признат от Националната агенция за сигурност (**NSA**) на **САЩ** като алгоритъм за защита на класифицирана информация до ниво „Строго секретно“. **AES** поддържа **128** и **256**-битови ключове (при **CBC**-режим) или **256** и **512**-битови ключове (при **XTS**-режим).

SERPENT е усъвършенствана версия на **AES**, разработена от **Ross Anderson**, **Eli Biham** и **Lars Knudsen**, включваща **32**-кратно последователно преобразуване на поверителното съдържание в процеса на неговото шифроване. Паралелният начин на осъществяваните преобразувания ускорява значително производителността на алгоритъма, но същевременно допуска уязвимости, сходни с тези при датиращия от **1977** г. Стандарт за шифроване на данни (**DES**). **SERPENT** поддържа **128** и **256**-битови ключове (при **CBC**-режим) или **256** и **512**-битови ключове (при **XTS**-режим).

TWOFISH е разработен от **Брус Шнайер** (**Bruce Schneier**) и включва **16**-кратно преобразуване на поверителното съдържание посредством шифровъчните мрежи на **Feistel**. Прилагането на псевдо-трансформацията на **Hadamard** засилва дифузията на шифрованото съдържание и затруднява разбиването на шифровъчния алгоритъм. **TWOFISH** поддържа **128** и **256**-битови ключове (при **CBC**-режим) или **256** и **512**-битови ключове (при **XTS**-режим).

Поддържаните от **Cryptsetup** хеш-алгоритми позволяват обезпечаване сигурността на паролите (което е от критична важност за защита на системата). За да може да бъде направена проверка за правилността на въведената парола, копие от последната трябва да бъде записано в системата. Това обаче позволява записът да бъде извлечен и паролата – разкрита. За да се ограничи този риск, се прилагат хеш-алгоритми – въведената парола се преобразува (хешира) до сложна криптографска производна, която бива записана вместо самата парола. При последващо въвеждане на паролата се осъществява същото криптографско преобразуване чрез хеш-алгоритъма и хешираната производна се сравнява със записаната. Така, дори и да се достигне до извличане на хешираната производна, тази производна няма да бъде особено полезна, тъй като от нея не може да се направи извод за това каква символна комбинация да се въведе, за да бъде изведена същата производна и сравнението да даде положителен криптографски отговор.

PBKDF2 (**Password-Based Key Derivation Function**) представлява специализиран алгоритъм за съхраняване на пароли в защитен вид, който намалява значително възможностите за прилагане на bruteforce технологии за разбиването им. Работи с версии **1**, **256** и **512** на осигуряващия хеш-алгоритъм (**Secure Hash Algorithm**; **SHA**), разработени от Националната агенция за сигурност (**NSA**) на **САЩ** съответно през **1993** и **2010** г. И при трите версии **PBKDF2** поддържа **256**-битови ключове.

RIPEMD160 (**RACE Integrity Primitives Evaluation Message Digest**) е разработен през **1996** г. от **Hans Dobbertin**, **Antoon Bosselaers** и **Bart Preneel**. Работи на принципа на хеш-алгоритъма **MD4** и дава сходни резултати като **SHA-1**. Поддържа **256**-битови ключове. При версиите на **RIPEMD**, различни от **160**, са документирани определени уязвимости и поради това не се препоръчва да бъдат прилагани – въпреки по-дългите ключове, които някои от тях поддържат.

ARGON2 е разработен от **Алекс Бирюков**, **Даниел Дину** и **Димитрий Ковратович**, като до този момент са публикувани три версии: **ARGON2D**, **ARGON2I** и **ARGON2ID**. Първата версия е оптимизирана срещу разбиване чрез ресурсите на графичен ускорител, но проявява слабости при разбиване чрез анализ на напреженията и активностите в централния процесор. Втората версия е оптимизирана срещу анализ на напреженията и активностите в централния процесор, а третата версия е хибрид между предходните две. **Cryptsetup** поддържа втората и третата версия с **256**-битови ключове.

3. Цялостно шифроване, инсталиране и настройване на системата

За да стартирате процеса по самото шифроване, е необходимо най-напред да се ориентирате под какви наименования системата разпознава вашите устройства – чрез командата '**lsblk**', като интересуващото ви устройство (твърдия диск) най-вероятно е обозначен като '**sda**', '**sdb**' или нещо подобно.

След като вече знаете под какво наименование системата разпознава твърдия диск, трябва да осъществите т.нар. рандомизация:

```
# dd if=/dev/urandom of=/dev/Устройство bs=64M \
status=progress oflag=sync
```

(където под '**Устройство**' разбираме наименованието, под което системата разпознава твърдия диск). Процесът по рандомизация запълва цялото пространство на устройството с произволни символи. Това се прави от съображения за сигурност – след като устройството бъде шифровано, рандомизацията няма да позволи установяването чрез фини аналитични методи каква част от пространството е заета с файлове и как точно са разположени. Подобна информация би улеснила съществено опитите за разбиране на шифъра.

Имайте предвид, че горната команда отнема време (при по-големи твърди дискове е възможно изпълнението на командата да отнеме до няколко дни). Организирайте процеса така, че компютърът да работи необезпокоявано в този период и да има непрекъснато електрозахранване. В терминала ще се извеждат в реално време данни за това докъде е достигнал процесът. Докато чакате няма пречка да отворите друг терминал и в него да осъществявате други процеси, стига те да не засягат устройството, което рандомизирате.

След като рандомизацията приключи, идва ред да осъществите цялостно шифроване на твърдия диск:

```
# cryptsetup Настройки luksFormat /dev/Устройство
```

(където под '**Настройки**' разбираме конкретните параметри, съгласно които желаете да бъде осъществено шифроването на твърдия диск).

Най-общо, можете да зададете следните допълнителни настройки, съгласно които да бъде осъществено шифроването (представяме пълните команди (предхождани от символите '**--**') или кратките им алтернативи (предхождани от символ '**-**'), където това е приложимо):

– прилагане на формата на Хедъра, с който да стартира процесът на (раз)шифроване:

--type luks1
-M

(по-долу ще обсъдим критичното значение на Основния ключ и Хедъра);

– прилагане на предпочитан от вас шифровъчен алгоритъм (примерно '**serpent-xts-plain64**'):

--cipher Алгоритъм
-c

(по-горе разгледахме накратко най-известните шифровъчни алгоритми, поддържани от **Cryptsetup**);

– указване дължината (сложността) на шифровъчния ключ (като имате предвид, че всеки шифровъчен алгоритъм работи с ключове, не по-къси от и не по-дълги от определени размери); при шифровъчния алгоритъм '**serpent-xts-plain64**' максималната дължина (сложност) на ключа е **512B** (изписват се само цифрите):

--key-size Дължина
-s Дължина

– прилагане на предпочитан от вас хеш-алгоритъм за записване на паролата като сложна криптографска производна (примерно '**sha512**'):

--hash Алгоритъм
-h

– прилагане на забавящи периоди при проверка на въведената парола, затрудняващи прилагането на bruteforce технологии за разбиване на паролата; примерно **2000 ms** (изписват се само цифрите):

--iter-time Забавяне
-i Забавяне

– осъществяване на допълнителна задълбочена рандомизация, увеличаваща ентропията и затрудняваща прилагането на фини аналитични методи за установяване разположените файлове на твърдия диск:

--use-random

Във връзка с горните параметри следва да се отбележи, че типът хедър по подразбиране е **LUKS2** – но на този етап това е несъвместимо с **Libreboot**. Поради тази причина за параметъра '**type**' въвеждаме по-старата версия **LUKS1**. Останалите параметри могат да бъдат оставени и по подразбиране.

Съгласно горните указания командата за осъществяване на цялостно шифроване на системата би придобила следния пълен вид (където е възможно сме избрали кратките алтернативи на съответните параметри):

**# cryptsetup --type luks1 -c serpent-xts-plain64 -s 512 **
-h sha512 -i 2000 --use-random luksFormat /dev/Устройство

След изпълнението на горната команда ще бъде изведено указание да въведете парола (имайте предвид, че при забравена парола няма да имате възможност да разшифровате системата по какъвто и да било друг начин и ще се

наложи просто да я изтриете и преинсталирате, ако желаете да продължите да ползвате компютъра). Ще бъде създаден шифровъчен ключ (**Volume Key**) за цялостно (раз)шифроване на твърдия диск съгласно указаните параметри. Препоръчва се да си запишете избраните параметри, тъй като след време е възможно да се наложи ръчното им въвеждане, за да разшифровате вашето съдържание. В такъв случай възпроизвеждането по памет на точната конфигурация от параметри може да се окаже трудно и да препятства разшифроването.

Първата стъпка, която трябва да предприемете след като твърдият диск вече е изцяло шифрован, е да го отворите с програмата **Cryptsetup**:

```
# cryptsetup open /dev/Устройство Наименование
```

(където под '**Устройство**' разбираме наименованието, под което системата разпознава твърдия диск (примерно '**sda**'), а под '**Наименование**' разбираме наименованието, под което желаете системата да разпознава вашия изцяло шифрован слой на твърдия диск). При изпълнението на тази команда системата ще отвори изцяло шифрования слой, като го индексира със зададеното от вас наименование.

След като изцяло шифрованият слой вече е отворен, трябва да осъществите и т.нар. „нулиране“ – процес по преобразуване, насочен към увеличаване на информационната сигурност:

```
# dd if=/dev/zero of=/dev/mapper/Наименование bs=64M \
status=progress oflag=sync
```

Имайте предвид, че процесът отнема време (макар да е по-кратък от този по рандомизацията) и е възможно да се наложи да оставите компютъра да работи, включен в захранване.

След като изцяло шифрованият слой бъде „нулиран“, идва ред да създадете слой за единното **LVM**-логическо управление на неговите дялове (**Logical Volume Manager**). Трябва също така да създадете т.нар. „група“ за организиране общото пространство на твърдия диск в тези логически дялове, където ще бъде разпределена вашата операционна система (подобно на начина, по който постъпихме в Част **II** при инсталирането на една обикновена (нешифрована) операционна система). Можете да сторите това чрез тази команда:

```
# vgcreate Група /dev/mapper/Наименование
```

(където под '**Група**' разбираме наименованието, под което желаете системата да разпознава новосъздаващата се група, а под '**Наименование**' разбираме наименованието, под което по-горе указахте системата да разпознава вашия изцяло шифрован слой).

След като групата е създадена, можете да пристъпите към попълването ѝ с необходимите логически дялове. Ще възпроизведем отново познатата ни структура от разменен '**swap**' дял, системен '**/**' дял и потребителски '**data**' дял, като този път ще добавим още един дял, който да бъде шифрован допълнително и ще служи за съхраняването на поверителна информация. На този етап озаглавяваме този допълнителен шифрован дял '**lock**' (като вие можете да изберете и друго наименование, стига то да не се дублира с някое от наименованията във файловата структура на **GNU/Linux-libre** (като препоръчваме имената да не съдържат никаква специфична информация, интервали или други специални символи (освен '**-**' и '**_**') и букви, различни от **US**-стандартизацията на латиницата).

Разменният '**swap**' дял е важно да бъде първи, по две основни причини. Тъй като той ще служи за изпълнението на различни изчислителни задачи, операционната система често ще прибягва до него, а разполагането му най-отпред върху твърдия диск ще улесни (ускори) този процес; системата ще работи малко по-бързо и ще разходва малко по-малко енергия за това. Втората причина е, че следващият системен '**/**' дял е от критично значение за съществуването и функционирането на системата. Понякога стават инциденти и се стартира нежелан процес по изтриване / променяне на информацията върху твърдия диск – в такъв случай е препоръчително да имате време за реакция, за да можете да прекъснете нежелания процес, преди да достигне до критични за системата участъци на твърдия диск. Затова препоръчваме системният '**/**' дял да е втори, а не първи.

Можете да създадете логически разменен '**swap**' дял чрез тази команда:

```
# lvcreate -L 8G Група -n Разменен_дял
```

(където под '**Разменен_дял**' разбираме наименованието, под което желаете системата да обозначи разменния дял – примерно '**swap**').

Можете след това да създадете логически системен '**/**' дял чрез тази команда:

```
# lvcreate -L 30G Група -n Системен_дял
```

(където под '**Системен_дял**' разбираме наименованието, под което желаете системата да обозначи системния дял – примерно '**root**' – тъй като няма как да бъде обозначен само със символа '**/**').

Както посочихме, този път ще създадем и един допълнителен шифрован дял (който по-горе избрахме да назовем '**lock**'). За нуждите на изложението нека приемем, че този дял е с размер **20GB**. Можете да създадете този логически '**lock**' дял чрез тази команда:

```
# lvcreate -L 20G Група -n Шифрован_дял
```

(където под '**Шифрован_дял**' разбираме наименованието, под което желаете системата да обозначи допълнителния шифрован дял – примерно '**lock**').

И накрая остава да създадете потребителския '**data**' дял върху цялото останало пространство от твърдия диск, чрез тази команда:

```
# lvcreate -l +100%FREE Група -n Потребителски_дял
```

(където под '**Потребителски_дял**' разбираме наименованието, под което желаете системата да обозначи потребителския дял – примерно '**data**').

След като горните команди бъдат изпълнени, вашият напълно шифрован твърд диск ще придобие подобен вид:

```
NAME                        MAJ:MIN  RM   SIZE  RO  TYPE  MOUNTPOINT
sda                           8:0     0 100.0G   0  disk
└─Наименование              254:0     0 100.0G   0  crypt
  ├─Група-Системен_дял      254:1     0    30G   0  lvm
  ├─Група-Разменен_дял      254:2     0     8G   0  lvm
  ├─Група-Шифрован_дял      254:3     0    20G   0  lvm
  └─Група-Потребителски_дял 254:4     0    42G   0  lvm
```

След като логическите дялове са създадени (до голяма степен еквивалентно на физическите дялове, които създадохте в Част **II** при инсталирането на обикновена операционна система), идва ред да форматирате всеки от тях (да му определите файлова система и точка на закачане) съобразно функциите, за които е предназначен.

Можете да форматирате логическия системен '**root**' дял:

```
# mkfs.ext4 -O^metadata_csum_seed /dev/mapper/Група-Системен_дял
```

Можете да форматирате и активирате логическия разменен '**swap**' дял:

```
# mkswap /dev/mapper/Група-Разменен_дял
```

... и след това можете да го активирате:

```
# swapon /dev/mapper/Група-Разменен_дял
```

Можете да форматирате логическия потребителски '**data**' дял:

```
# mkfs.ext4 /dev/mapper/Група-Потребителски_дял
```

Накрая идва ред да настроите и предназначения за допълнително шифроване '**lock**' дял, който определихме да служи за съхраняване на поверителна информация. Както стана дума по-горе, този дял ще трябва да се разшифрова отделно, след като бъде разшифрована самата система, за да се достъпи до намиращата се в него информация. Така ще можете да ползвате системата регулярно, без това да дава достъп и до шифрованата в този дял информация. Само в специални моменти, когато сте на сигурно място и системата не е свързана към internet, ще разшифровате допълнително шифрования '**lock**' дял и ще достъпвате поверителната информация.

За горната цел трябва да повторите още веднъж част от операциите, които извършихте върху целия твърд диск – като най-напред осъществите рандомизиране на обособения '**lock**' дял (изпълнението на командата може да отнеме време):

```
# dd if=/dev/urandom of=/dev/mapper/Група-Шифрован_дял bs=64M \
status=progress oflag=sync
```

В резултат (адаптирайте командата съгласно избраните от вас наименования) пространството на предвидения за допълнително шифроване дял от твърдия диск ще бъде рандомизирано с оглед по-трудното подлагане на фини аналитични методи.

Следва проследеният по-горе шифровъчен процес, в който трябва да зададете желаните от вас параметри на шифроване (изпълнението на командата може да отнеме много време и да се наложи да оставите компютъра включен, за да работи няколко часа):

```
# cryptsetup --type luks1 -c serpent-xts-plain64 -s 512 \
-h sha512 -i 2000 --use-random luksFormat /dev/mapper/Група-Шифрован_дял
```

Както отбелязахме по-горе, препоръчва се да запишете избраните параметри на шифроване, тъй като след време е възможно да се наложи ръчното им въвеждане, за да разшифровате вашето съдържание.

Сега идва ред да отворите допълнително шифрования '**lock**' дял чрез програмата **Cryptsetup**:

```
# cryptsetup open /dev/mapper/Група-Шифрован_дял Наименование
```

(където под '**Наименование**' разбираме наименованието, под което желаете системата да разпознава вашия допълнително шифрован дял (примерно '**lock**')). При изпълнението на тази команда системата ще отвори допълнително шифрования дял, като го индексира със зададеното от вас наименование.

След като предвиденият за допълнително шифроване дял от твърдия диск е отворен, трябва да осъществите и т.нар. „нулиране“ (изпълнението на командата може да отнеме много време, макар и по-кратко от това по рандомизацията):

```
# dd if=/dev/zero of=/dev/mapper/Група-Шифрован_дял bs=64M status=progress
```

Накрая идва ред да форматирате предвидения за допълнително шифроване логически дял:

```
# mkfs.ext4 /dev/mapper/Група-Шифрован_дял
```

С изпълнение на горните команди вече ще имате дял от твърдия диск, който е шифрован допълнително и след разшифроване на системата той ще продължи да бъде недостъпен за трети страни. Това позволява да ползвате текущо системата, без да излагате вашата поверителна информация на допълнителен риск. При необходимост ще се оттеглите на сигурно място и ще разшифровате поверителния '**lock**' дял, без да свързвате системата към internet, и отново ще го шифровате, преди да установите контакт с външния свят.

След като логическите дялове, на които планирате да разпределите вашата Свободна операционна система (но вече в напълно шифрован вид) са подготвени, можете да пристъпите и към самата инсталация. За тази цел трябва да закачите логическия системен '**/**' дял към празната директория **/mnt** на помощната система:

```
# mount /dev/mapper/Група-Системен_дял /mnt
```

Идва ред да създадете потребителска директория с тази команда:

```
# mkdir /mnt/Потребителска_директория
```

(където под '**Потребителска_директория**' разбираме наименованието, под което желаете системата да разпознава вашия допълнително шифрован дял (примерно '**data**')), и да монтирате потребителския дял към новосъздадената потребителска директория:

```
# mount /dev/mapper/Група-Потребителски_дял /mnt/Потребителска_директория
```

Следва и самата инсталация на основните софтуерни модули, чрез описаната в Част **II** команда '**pacstrap**':

```
# pacstrap /mnt
```

Възможно е в процеса на инсталирането да се установи необновен шифровъчен ключ, с който е удостоверена автентичността и интегритета на някой от софтуерните пакети. В такъв случай системата ще откаже да извърши инсталирането и ще изведе указание с електронния пощенски адрес на издателя на необновения ключ. Можете да обновите този ключ с тази команда:

```
# pacman-key --refresh-keys Електронен_пощенски_адрес
```

(където под '**Електронен_пощенски_адрес**' разбираме изведения от системата адрес на съответния издател). След изпълнението на тази команда можете да продължите с горепосочената команда '**pacstrap**'.

След като основният пакет от софтуерни пакети е инсталиран, трябва да укажете системата да се зарежда при всяко следващо стартиране на компютъра съгласно точките на закачане и файловите системи, които настроихте по-горе за всеки един от логическите дялове. Добре е обаче преди това да прегледате информацията за това как са настроени точките на закачане и файловите системи, с тази команда:

```
# genfstab -U /mnt
```

Ако някой от логическите дялове не бъде изведен, закачете същия към системата (системния '**root**' дял към '**/mnt**', потребителския '**data**' дял към '**/mnt/data**' и т.н.); респективно, активирайте разменния '**swap**' дял с проследената по-горе команда '**swapon**'. Проверете дали всичко е наред с командата '**lsblk**' и вече можете да извършите и самото указване:

```
# genfstab -U /mnt >> /mnt/etc/fstab
```

Необходимо е след това да укажете ползване по подразбиране на определен базов език от системата. Както забелязахме в Част **II** по-горе, на този етап следва да изберете американската стандартизация на английския език – чрез тези команди:

```
# echo en_US.UTF-8 UTF-8 >> /mnt/etc/locale.gen
```

... и след това:

```
# echo LANG=en_US.UTF-8 > /mnt/etc/locale.conf
```

Следват настройки на основните параметри на системата. Най-напред трябва да укажете, че ще действате като администратор на директорията '**/mnt**' в помощния компютър (където са разположени основните софтуерни пакети на инсталиращата се система):

```
# arch-chroot /mnt
```

Необходимо е да генерирате базовите езикови настройки на системата:

```
# locale-gen
```

Необходимо е да инсталирате ядрото и необходимите му основни софтуерни пакети, а така също – необходимите софтуерни пакети за (раз)шифроване:

```
# pacman -S linux-libre-lts linux-libre-firmware mkinitcpio \
bash-completion networkmanager nano lvm2 cryptsetup
```

Сред цитираните софтуерни пакети освен тези, на които се спряхме в Част **II**, са и тези, необходими за цялостното (раз)шифроване на системата. Както вече стана дума, няма пречки на по-късен етап да инсталирате и всякакъв друг софтуер. Добавените тук софтуерни пакети са следните:

mkinitcpio
генератор на първоначалната система initramfs

lvm2
софтуерен инструмент за управление на виртуални дялове

cryptsetup
програма за цялостно (раз)шифроване на системата

Следват указания към системата за това как да стартира вашата изцяло шифрована система. Необходимо е да отворите с програмата **Nano** конфигурационния файл '**mkinitcpio.conf**', който се намира в директорията **/etc**:

```
# nano /etc/mkinitcpio.conf
```

... и да промените единствения „некоментиран“ (незапочващ със символа '**#**') ред '**HOOKS**', който във вашия случай би трябвало да има подобен вид:

```
HOOKS=(base udev autodetect modconf kms keyboard keymap consolefont block
filesystems fsck)
```

... по следния начин (като добавите параметрите '**systemd**', '**sd-encrypt**' и '**lvm2**' след параметъра '**block**'):

```
HOOKS=(base udev autodetect modconf kms keyboard keymap consolefont block systemd sd-encrypt lvm2 filesystems fsck)
```

(въведете текста на един ред, без да го пренасяте).

Запишете горната редакция, като натиснете **[Ctrl]**+**[o]**, **[Enter]** и след това – **[Ctrl]**+**[x]**.

Сега идва ред да въведете **UUID**-идентификатора на вашия твърд диск в конфигурационния файл '**crypttab.initramfs**', намиращ се също в директорията **/etc**:

```
# cryptsetup luksUUID /dev/Устройство > /etc/crypttab.initramfs
```

UUID-идентификаторът ще бъде извлечен от системата и записан във файла '**crypttab.initramfs**', за да указва системата да пристъпва при стартиране към разшифроването на това устройство. След като **UUID**-идентификаторът е записан във файла '**crypttab.initramfs**', трябва да отворите последния с програмата **Nano**:

```
# nano /etc/crypttab.initramfs
```

... и да го редактирате. Файлът в изходно положение ще изглежда примерно така:

```
b1af91e6-55d8-71ee-bf01-1ce34f2fce21
```

(където под '**b1af91e6-55d8-71ee-bf01-1ce34f2fce21**' разбираме един примерен **UUID**-идентификатор на твърд диск; при вас ще бъде различна конфигурация от същия брой буквено-цифрени комбинации). Трябва да редактирате файла '**crypttab.initramfs**' по следния начин:

```
Наименование UUID=b1af91e6-55d8-71ee-bf01-1ce34f2fce21 none password-echo=no
```

(където под '**Наименование**' разбираме наименованието, под което системата разпознава вашия твърд диск (примерно '**hdd**'); със записа '**none**' указваме системата да изисква ръчно въвеждане на вашата парола; а със записа '**password-echo=no**' – да не бъдат извеждани помощни символи (например '****************' или '**••••••••••••**') при въвеждането на парола за разшифроване).

След като запишете горните настройки (като натиснете **[Ctrl]**+**[o]**, **[Enter]** и след това – **[Ctrl]**+**[x]**), идва ред да генерирате началната система, която ще отговаря за разшифроване на твърдия диск:

```
# mkinitcpio -p linux-libre-lts
```

Системата ще изведе поредица от резултати, указващи осъществяваните процеси. Ако всичко е наред, ще завърши с генериране на необходимата ви начална система. Възможно е да даде предупреждения за слабости, обозначени в началото на съответните редове с '**WARNING:**', но в повечето случаи това не са критични проблеми. Критичните проблеми обикновено се обозначават с '**ERROR:**' и в повечето случаи това означава, че генерирането не е осъществено успешно (в повечето случаи заради допусната правописна грешка при въвеждането на някоя от командите по-горе).

Първата финализираща настройка, на която трябва да обърнете внимание, е да въведете парола за администраторски достъп до системата:

```
# passwd
```

При следващо стартиране на системата от вас ще се очаква да я достъпвате с администраторски права чрез нововъведената парола.

След като приключите с горните настройки, е време да излезете от **chroot**-режима:

```
# exit
```

Сега идва ред да инсталирате и конфигурирате **Libreboot** и стартовата програма **GRUB** – така, че да приведете компютъра в състояние да стартира вашата напълно шифрована операционна система.

4. Настройване на GRUB и Libreboot

Както вече забелязахме, стартирането на цялостно шифрованата система няма да може да става по обичайния начин чрез стартов (**Boot**) сектор върху твърдия диск, където да бъде разположена стартиращата програма **GRUB** – понеже при стартиране на системата той също ще бъде напълно шифрован; а **GRUB** не може да изпълни своите функции в шифровано състояние. Трябва да настроим системата така, че стартиращата програма **GRUB** да изпълнява функциите си изнесена някъде другаде, като поеме основната задача да зареди шифровъчния алгоритъм и необходимата минимална операционна среда, в която да въведете парола за разшифроване на твърдия диск. Подходящо място за тази цел е **BIOS/UEFI** flash-чипът. За тази цел можете да редактирате **Libreboot**-пакета, с който в Част **IV** препрограмирахме flash-чипа; и да извършите повторно препрограмиране (което вече ще можете да извършите директно от терминала на компютъра на който е чипа, без да е необходимо повторно физическо достъпване на flash-чипа чрез външни устройства).

За да редактирате избрания за вашия компютър **Libreboot**-пакет, е необходимо да инсталирате програмите **Cbfstool** и **Flashrom** (първата от които се намира в пакета '**libreboot-utils**', съдържащ набора от софтуерни инструменти за работа с **Libreboot**):

```
# pacman -S libreboot-utils flashrom
```

Ако не сте се убедили в автентичността и интегритета на **Libreboot**-пакета (евентуална предходна проверка на сваляни преди време файлове не гарантира, че впоследствие файловете не са били компрометирани), сега е моментът да направите и необходимите криптографски проверки.

След като програмите **Cbfstool** и **Flashrom** са налице и разполагате с подходящия за вашия случай **Libreboot**-пакет (като автентичността и интегритета е потвърдена), можете да извлечете чрез програмата **Cbfstool** файловете '**grub.cfg**' и '**grubtest.cfg**' от **Libreboot**-пакета.

Тъй като програмата **GRUB** е софтуер от критично значение за системата (без него изобщо не можете да я стартирате), в **Libreboot**-пакета съществуват два паралелни файла за стартовото меню на **GRUB** – основният '**grub.cfg**' и тестовият '**grubtest.cfg**'. Това ви позволява да редактирате тестовия файл '**grubtest.cfg**', да тествате дали нанесените редакции работят правилно и след като се убедите, че всичко е наред, да въведете редакциите и в основния файл '**grub.cfg**'. Тогава ако объркате нещо в тестовия файл '**grubtest.cfg**', ще може да забележите и отстраните грешките, преди да нанесете конфигурацията в основния файл '**grub.cfg**'. Имайте предвид, че проблеми могат да настъпят дори и тогава, когато сте работили абсолютно точно – например поради техническа грешка при копирането. Затова винаги ползвайте горната възможност, като първо редактирате тестовия файл '**grubtest.cfg**'; и едва след това – основния '**grub.cfg**'.

За да работите по-удобно, можете да преместите избрания **Libreboot**-пакет (примерно '**grub_x200_8mb_libgfxinit_corebootfb_usqwerty.rom**') във вашата настояща директория, където сте преместили и терминала. Така ще си спестите необходимостта при следващите команди да въвеждате пълния адрес на **Libreboot**-пакета. Ако **Libreboot**-пакетът е в настоящата ви директория, можете да извлечете тестовия файл '**grubtest.cfg**' така:

```
$ cbfstool Libreboot_пакет extract -n grubtest.cfg -f grubtest.cfg
```

(където под '**Libreboot_пакет**' разбираме избрания **Libreboot**-пакет – например '**grub_x200_8mb_libgfxinit_corebootfb_usqwerty.rom**').

Можете с аналогична команда да извлечете и основния файл '**grub.cfg**', но на този етап е добре да оставите този файл настрана и да работите само в тестовия файл – поне докато се убедите, че редакциите, които ще нанесете, работят правилно.

Отворете тестовия файл '**grubtest.cfg**' с програмата **Nano** и редактирайте кода на файла, като добавите преди реда, в който за първи път се споменава кодът '**menuentry**', следния допълнителен код (копирайте много внимателно и въведете необходимата редакция съвсем точно):

```
menuentry 'Заглавие' --hotkey='Стартиращ_клавиш' {
  cryptomount (ahci0)
  root=lvm/Група-Системен_дял
  linux /boot/vmlinuz-linux-libre-lts root=/dev/mapper/Група-Системен_дял rw
  initrd /boot/initramfs-linux-libre-lts.img
}
```

(където под '**Заглавие**' разбираме заглавието, с което желаете да бъде обозначавана в стартовото меню възможността да стартирате системата от конфигурираната от вас версия на **Libreboot**; под '**Стартиращ_клавиш**' – бърз бутон за стартиране на конфигурираната версия директно чрез неговото натискане (като не трябва да посочвате клавишите **[a]**, **[b]**, **[e]**, **[c]**, **[d]**, **[o]**, **[p]**, **[r]**, **[s]**, **[t]** и **[u]** – които са заети като бързи бутони за други функции от стартовото меню); под '**ahci0**' разбираме наименованието, под което вашата система най-вероятно разпознава цялостно шифрования твърд диск (ако към вашата система са закачени няколко твърди дискове, е възможно да бъдат обозначени със следващи цифри – примерно '**ahci1**'); под '**Група**' и '**Системен_дял**' – съответно наименованието на създадената от вас група и това на системния '**/**' дял). След като въведете горния код (където дадените от нас примерни наименования са заместени с ползваните от вас), трябва да запишете промените и да затворите **Nano** (като натиснете **[Ctrl]+[o]**, **[Enter]** и след това – **[Ctrl]+[x]**). Този допълнителен код ще укаже на стартиращата програма **GRUB** да разшифрова дисковото пространство и да стартира от системата от логическите **LVM**-дялове.

В тестовия файл '**grubtest.cfg**' съществуват няколко групи инструкции, започващи с кода '**menuentry**' – те съответстват на възможностите от стартовото меню на **GRUB**, с което системата се инициализира след стартиране на компютъра. Вашата инструкция (с която по същество добавяте най-отгоре в менюто) е насочена към това – да укаже на системата да зареди напълно шифрования твърд диск, като съобрази, че стартовата програма **GRUB** е изнесена на flash-чипа. Като разположите тази инструкция най-отгоре в менюто, ще я направите възможност по подразбиране (при стартиране на системата, ако до няколко секунди след появяване на стартовото меню потребителят не реагира, ще се зареди именно тази възможност (като по същество изведе указание да въведете парола за разшифроване на твърдия диск)). В случай, че не желаете това да бъде възможност по подразбиране, можете да разположите инструкцията и по-надолу в кода, но внимавайте да не „разкъсате“ някоя от другите групи инструкции и да вмъкнете вашата между нейните половини. Имайте предвид в тази връзка, че първата '**menuentry**' (за разлика от всички останали) е доста дълга – прегледайте кода на файла и забележете къде по-долу идва следващата инструкция (също започваща с '**menuentry**').

Сега трябва да въведете така редактирания тестов файл '**grubtest.cfg**' обратно в **Libreboot**-пакета, за да бъде след това препрограмиран с него flash-чипът. За тази цел трябва първо да изтриете от **Libreboot**-пакета старата версия на тестовия файл '**grubtest.cfg**' (която все още се намира в него):

```
$ cbfstool Libreboot_пакет remove -n grubtest.cfg
```

Системата извежда индикатор '**Performing operation on 'COREBOOT' region...**', от който научаваме, че командата е изпълнена.

Можете вече да въведете редактираната версия на тестовия файл '**grubtest.cfg**':

```
$ cbfstool Libreboot_пакет add -n grubtest.cfg -f grubtest.cfg -t raw
```

Системата отново извежда индикатор '**Performing operation on 'COREBOOT' region...**'.

След като сте нанесли горните редакции в **Libreboot**-пакета, идва ред да стартирате сесия с администраторски права и да препрограмирате flash-чипа. За да сторите това обаче, при последното стартиране на инсталационната система трябва да сте въвели режим за препрограмиране на flash-чипа. Ако последния път не сте направили това, рестартирайте системата и при зареждане на **GRUB**-менюто (или, ако все още не сте осъществили цялостно шифроване на твърдия диск – при зареждане на второто **GRUB**-меню) натиснете **[e]** (от **E**[**dit**]), за да отворите режима за редактиране и въведете в края на реда започваш с командата '**linux**' този параметър:

```
iomem=relaxed
```

След като сторите това, натиснете **[Ctrl]**+**[x]** или **[F10]** (вижте поясненията на екрана най-отдолу) и продължете със стартирането на системата. До следващото си изключване системата ще бъде в режим, позволяващ препрограмирането на flash-чипа.

Да препрограмирате flash-чипа с редактирания **Libreboot**-пакет можете чрез програмата **Flashrom**:

```
# flashrom -p internal -w Libreboot_пакет
```

Ако всичко е наред, системата ще препрограмира вашия flash-чип с редактирания по-горе **Libreboot**-пакет и ще изведе резултат '**VERIFIED**'.

След като flash-чипът е препрограмиран с редактирания от вас **Libreboot**-пакет, идва ред да рестартирате отново и от стартовото меню на **GRUB** (първото, което ще се зареди) да изберете тестовата възможност '**Load test configuration (grubtest.cfg) inside of CBFS**' (която може да се извика и просто като натиснете **[t]** (от '**T**[**est**]'). Така ще укажете на системата да презареди стартовото меню на **GRUB**, но вече чрез тестовия файл '**grubtest.cfg**', а не чрез основния файл '**grub.cfg**'. Най-пряко свидетелство за това е появяването в стартовото меню добавената от вас възможност, която въведохте с горните редакции. Укажете на системата да стартира от тази възможност – би трябвало да бъде изведено указание да въведете парола за разшифроване на твърдия диск. Пробвайте всичките интересуващи ви възможности от стартовото меню (като не забравяте всеки следващ път да зареждате от тестовия файл '**grubtest.cfg**' чрез първоначално натискане на **[t]**).

Ако всичко с тестовия файл '**grubtest.cfg**' е наред, вече можете да нанесете редакцията и в основния файл '**grub.cfg**', като въведете и него в **Libreboot**-пакета, а след това препрограмирате отново flash-чипа и тествате още веднъж, но този път от основния файл '**grub.cfg**'. В случай, че нещо се обърка, можете да рестартирате, да заредите от тестовия '**grubtest.cfg**' файл и да отстраните възникналата грешка.

Препоръчва се да тествате щателно всяка манипулация спрямо **Libreboot**-софтуера – дори и когато сте сигурни, че сте работили съвсем прецизно и напълно съгласно напътствията. Дори и да не сте допуснали грешка, понякога настъпват технически грешки в процеса по копирането или препрограмирането. Наличието в този смисъл на два паралелни файла за стартовото меню на **GRUB** не е излишно. В случай, че настъпи грешка и в двата файла, системата няма да

може да бъде стартирана и като единствен начин да излезете от създалата се ситуация ще остане това да закачите физически пиновете на flash-чипа и да го препрограмирате отново чрез микроконтролер (или едноплатков компютър) и помощен компютър (както направихте в Част **III** при отстраняването на **BIOS/UEFI**).

След като основните софтуерни пакети са инсталирани, зададени са основните конфигурационни настройки и програмата **GRUB** е настроена за стартиране на напълно шифрованата операционна система, можете да рестартирате компютъра:

```
# reboot
```

Ако възможностите от стартовото меню на **GRUB** работят правилно, идва ред да продължите с финализиране инсталирането и настройването на вашата напълно шифрована операционна система. Въпреки, че за обикновения потребител системата все още просто не работи, до тук се решиха базовите въпроси за нейното функциониране и стабилност. Следващите действия ще позволят системата да заработи така, както повечето потребители очакват – с графична среда, потребителски интерфейс и т.н.

5. Финализиране инсталацията и настройките на операционната система

Трябва на първо място да създадете и потребителски акаунт без администраторски права:

```
# useradd -m Потребител
```

Също така трябва да въведете парола на потребителя без администраторски права:

```
# passwd Потребител
```

На следващо място е необходимо да зададете наименование, под което системата ще разпознава вашето компютърно устройство:

```
# hostnamectl hostname Компютър
```

(където под '**Компютър**' разбираме наименованието, под което желаете да бъде разпознавано устройството). Препоръчваме кратко и без никаква специфична информация наименование, което не включва интервали или други специални символи (освен '**-**' и '**_**') и букви, различни от **US**-стандартизацията на латиницата – като например '**computer**' или '**comp**'.

Следва да ориентирате системата относно времето и пространството, където се намирате:

```
# timedatectl set-timezone Континент/град
```

(където под '**Континент/град**' разбираме вашия континент и град, например '**Europe/Sofia**').

Остава да въведете автоматична синхронизация на времето:

```
# systemctl enable --now systemd-timesyncd
```

На следващо място трябва да активирате функционалностите на системата за свързване към internet:

```
# systemctl enable --now NetworkManager
```

Вече можете да се свържете безжично към internet:

```
# nmcli dev wifi connect Мрежа password Парола
```

(където под '**Мрежа**' разбираме наименованието на близката **WiFi** мрежа, а под '**Парола**' – паролата за нейното достъпване). Можете да проверите дали имате internet-свързаност с проследената по-горе команда '**ping**'.

Ако предпочитате кабелна връзка, е достатъчно да включите **LAN**-кабела и да изчакате малко.

След като вече имате internet-свързаност, можете да пристъпите към инсталиране и на следващите базови софтуерни пакети, които осигуряват функциониране на системата (в т.ч. на графичната среда и потребителския интерфейс):

```
# pacman -S gdm gnome-shell gnome-control-center \
ttf-dejavu lxterminal nautilus
```

Впоследствие ще можете да инсталирате и други софтуерни пакети, в зависимост от това какво правите на компютъра и от какъв софтуер се нуждаете. Какви софтуерни пакети сте инсталирали на системата можете да видите с проследената по-горе команда '**pacman -Qs**', а какви софтуерни пакети съществуват, свързани с определена ключова дума – с командата '**pacman -Ss Ключова_дума**'.

Когато необходимите ви основни софтуерни модули са инсталирани, можете да стартирате графичната среда:

```
# systemctl enable --now gdm
```

Добре дошли за втори път във вашата Свободна операционна система – този път изцяло шифрована!

Можете да започнете с актуализиране на списъците и шифровъчните ключове, по които ще бъдат изисквани необходимите ви пакети от софтуерните хранилища и ще бъде проверявана тяхната автентичност и интегритет – чрез представените по-горе команди: '**pacman -Sy parabola-keyring archlinux-keyring**', след което '**pacman-key --refresh-keys**' и накрая '**pacman -Syu**'.

Когато желаете да отворите вашия допълнително шифрован '**lock**' дял, ще можете (след като преустановите всякакво свързване към internet и се оттеглите на сигурно място) да направите това така:

```
# cryptsetup open /dev/Група/Шифрован_диск Наименование
```

(където под '**Наименование**' разбираме наименованието, с което желаете системата да възприеме вашия шифрован '**lock**' дял; което не трябва да се дублира с други наименования, за да не причини грешка); и след това:

```
# mount /dev/mapper/Наименование /Точка_на_закачане
```

(където под '**Точка_на_закачане**' разбираме директорията, която по-горе определихте като точка на закачане за вашия шифрован '**lock**' дял – например директорията **/mnt**). Системата ще извърши указаното закачане и вие ще можете да ползвате вашето поверително съдържание.

Когато приключите работа и желаете да затворите шифрования '**lock**' дял, можете да сторите това така:

```
# umount /dev/mapper/Наименование
```

... и след това:

```
# cryptsetup close Наименование
```

За да въвеждате горните команди по-удобно, можете след като вече сте разшифровали и закачили допълнително шифрования дял, да държите терминала отворен, докато работите с този дял. Когато дойде време да го откачите и шифровате, можете чрез натискане на клавиша **[↑]** да изведете въвежданите по-рано команди за разшифроване, респективно – за закачане; и с помощта на кратки текстови редакции да въведете тези команди отново, но този път редактирани като за откачане, респективно – за шифроване на вашия '**lock**' дял. Това ще ви даде необходимата бързина (която понякога може да ви бъде наистина необходима).

Във връзка с последното се препоръчва да оттренирате бързо и пестеливо редактиране в текстови режим на командите от такива за разшифроване и закачане към такива за откачане и шифроване.

Така например, командата за закачане:

```
# mount /dev/mapper/Наименование /Точка_на_закачане
```

... може да се редактира към такава за откачане:

```
# umount /dev/mapper/Наименование
```

... като изтриете с няколко натискания на бутона **[Back]** израза '**/Точка_на_закачане**' (или като преместите курсора в началото на този израз и просто натиснете **[Ctrl]+[k]**); и след това се преместите в началото на командния ред с бутона **[Home]** или **[Ctrl]+[a]**, въведете '**u**' в началото на '**mount**' и натиснете **[Enter]**;

Командата за разшифроване:

```
# cryptsetup open /dev/Група/Шифрован_диск Наименование
```

... може да се редактира към такава за шифроване:

```
# cryptsetup close Наименование
```

... като се върнете с лява стрелка **[←]** преди израза '**Наименование**', изтриете с бутона **[Back]** израза '**open /dev/Група/Шифрован_диск**', въведете израза '**close**' на мястото на току що изтрития израз и натиснете **[Enter]**.

Можете да откачите и шифровате вашия '**lock**' дял и като просто натиснете в графичния интерфейс иконата **[△]**, разположена до иконата на самия шифрован дял във файловия браузър; или като извикате контекстното меню с десен бутон и изберете възможността за откачане (**Unmount**), за изваждане (**Eject**) или за безопасно отстраняване на устройството (**Safely Remove Drive**). Имайте обаче предвид, че в такъв случай ще трябва да разчитате на посредничеството на графичния интерфейс – което в някои случаи може да компрометира сигурността. Най-добре все пак ползвайте директно командния ред от терминала.

Препоръчва се да оттренирате горните възможности, за да придобиете необходимата рутина и способност да откачите, респективно шифровате вашия '**lock**' дял (което в случай на внезапна необходимост може да се окаже решаващо).

6. Цялостно шифроване чрез ключ от външен носител с памет

Следва междувременно да отбележим и възможността да осъществите цялостното шифроване на системата чрез файл от външен носител с памет (например определен файл, записан на **USB**-памет) вместо чрез парола. Предимство на този подход е, че няма опасност паролата да бъде прихваната при въвеждането ѝ (просто такава няма и съответно не се въвежда). Недостатък е, че външният носител може да бъде откраднат или иззет принудително; или да бъде унищожен физически. Независимо от всичко, понякога може да предпочитате системата да бъде достъпвана именно чрез определена вещ, а не чрез определена тайна. В такъв случай символите, които заместват паролата (битовете във файла, използван като ключ) може да бъдат значително по-големи по обем от дори най-дългите пароли, които човек е

в състояние да въведе от клавиатура. Друго предимство на този подход е, че можете да укажете на вашите довереници как да разшифроват намираща се у тях система – и дори след прихващане на указанията от трети страни сигурността да не бъде компрометирана, ако не са придобили и самия външен носител с файла, използван като ключ.

За да шифровате цялостно вашата система чрез файл от външен носител с памет, непосредствено след като редактирате системния файл '**mkinitcpio.conf**' (както проследихме по-горе), трябва да изпълните няколко специфични действия. Най-напред трябва да подготвите избрания външен носител – като можете да го шифровате по обичайния начин, ако желаете да го направите неизползваем без въвеждането на парола:

```
# cryptsetup --type luks1 luksFormat /dev/Устройство
```

(където под '**Устройство**' разбираме наименованието, под което системата разпознава избрания външен носител с памет. Тъй като в инсталационния процес е възможно да има закачени няколко устройства с памет към системата, всяко от тях ще бъде обозначено със следваща буква от азбуката – примерно '**sda**', '**sdb**', '**sdc**' и т.н. – проверете с командата '**lsblk**' и се ориентирайте кое е избраното от вас устройство, за да не изтриете например твърдия диск на компютъра или „живия“ носител, от който сте стартирали инсталационния процес).

След като подготвите шифроването на външния носител, може да го отворите:

```
# cryptsetup open /dev/Устройство Наименование
```

(където под '**Наименование**' разбираме наименованието, под което системата разпознава избрания външен носител с памет. Тъй като в системата е възможно да има закачени няколко шифровани устройства с наименования, трябва да внимавате тези наименования да не се дублират и да помните кое от тях за кое от устройствата се отнася, за да не изтриете погрешка някое от тях).

Следва да форматирате избрания външен носител с памет с предпочитаната файлова система (например **ext4**):

```
# mkfs.ext4 -O^metadata_csum_seed /dev/mapper/Наименование
```

(където под '**Наименование**' разбираме наименованието, под което системата разпознава избрания външен носител с памет. Тъй като в системата е възможно да има закачени няколко шифровани устройства с наименования, трябва да внимавате тези наименования да не се дублират и да помните кое от тях за кое от устройствата се отнася, за да не изтриете погрешка някое от тях).

След като избраният външен носител с памет е готов, трябва да запишете на него файла, който ще служи като ключ. Можете да изберете или създадете какъвто и да било файл (текст, изображение, аудио, видео или друго), но е важно файлът оттук нататък да не бъде променян. Имайте предвид, че някои файлове (например **.jpeg** изображенията) при копиране е възможно да се променят – в такъв случай вашият ключ може да престане да работи. Възможност е също така да създадете елементарен **.txt** файл с прост текст, точното разположение на всички символи в който и точно му наименование (вкл. интервали, главни и малки букви, и подобни) знаете наизуст. Можете по всяко време да създадете наново файла (със същото съдържание и със същото наименование, на същото местоположение в същия или сходно конфигуриран външен носител с памет – и той ще работи по същия начин; а когато не е необходим, можете да го изтривате – което ще действа на свой ред като парола). Препоръчва се файлът да стои сред масив от други подобни файлове (например албум с изображения на котенца и статии за домашните любимци; или масив от месечни отчети и презентации за двугодишен проект) – за да не привлича излишно внимание към себе си.

След като сте готови, идва ред да извлечете следните **UUID**-идентификатори:

– на твърдия диск на компютъра:

```
# blkid /dev/Твърд_диск > /etc/crypttab.initramfs
```

(където под '**Твърд_диск**' разбираме наименованието, под което системата разпознава твърдия диск на компютъра (и който ще бъде разшифрован при стартиране на операционната система) – примерно '**sda**');

– на външния носител с памет:

```
# blkid /dev/Външен_носител >> /etc/crypttab.initramfs
```

(където под '**Външен_носител**' разбираме наименованието, под което системата разпознава външния носител с памет (където сте избрали да съхранявате файла, който ще служи като ключ) – примерно '**sdb**'); и

– на наименованието, с което идентифицирате този физически дял от външния носител в системата, където сте избрали да съхранявате файла, който ще служи като ключ:

```
# blkid /dev/mapper/Наименование >> /etc/crypttab.initramfs
```

(където под '**Наименование**' разбираме наименованието, с което идентифицирате разшифрования външен носител – зададохте го по-горе при командата '**cryptsetup open /dev/Устройство Наименование**').

След изпълнението на горните команди трябва да редактирате системния файл '**crypttab.initramfs**':

```
# nano /etc/crypttab.initramfs
```

... който след изпълнението на горните команди трябва да има подобен вид:

```
/dev/Твърд_диск: UUID="Идентификатор" TYPE="crypto_LUKS"
/dev/Външен_носител: UUID="Идентификатор" TYPE="crypto_LUKS"
/dev/mapper/Наименование: UUID="Идентификатор" BLOCK_SIZE="...
```

(където на всеки от трите реда под '**Идентификатор**' разбираме съответния **UUID**-идентификатор на посоченото устройство); който файл трябва да редактирате по следния начин:

```
Външен_носител UUID=Идентификатор none read-only,password-echo=no
Твърд_диск UUID=Идентификатор Адрес/Файл:UUID=Идентификатор
```

(където под '**Твърд_диск**' и '**Външен_носител**' разбираме дадените от вас наименования на твърдия диск на системата и на външния носител с памет (примерно '**hdd**' и '**key**'); под '**Адрес/Файл**' разбираме избрания от вас файл, който ще служи като ключ, с местоположението му на съответния физически дял във външния носител; и под '**Идентификатор**' разбираме съответните **UUID**-идентификатори на посочените устройства).

Обърнете внимание, че на първия ред трябва да въведете **UUID**-идентификатора на самия външен носител, а на втория ред в края – **UUID**-идентификатора на (раз)шифрования слой на външния носител, в който се намира файлът, служещ като ключ.

След като горните настройки са въведени и записани, следва да генерирате началната система:

```
# mkinitcpio -p linux-libre-lts
```

... и да продължите с настройването на вашата цялостно шифрована система, както проследихме по-горе.

За да стартирате чрез ключ системата, е нужно да настроите конфигурационните файлове '**grub.cfg**' и '**grubtest.cfg**' в **Libreboot**-пакета, по начин, който указва на системата вместо да търси парола, да приложи правилния файл-ключ, разположен на правилното място в правилното устройство. За тази цел можете да въведете в посочените конфикурационни файлове (работете първо с '**grubtest.cfg**' и след като се убедите, че всичко е наред, копирайте и в '**grub.cfg**', както проследихме по-горе) следните инструкции:

```
menuentry 'Заглавие' --hotkey='Стартиращ_клавиш' {
  cryptomount (usb0)
  cryptomount -k (crypto0)/Адрес/Файл (ahci0)
  root=lvm/Група-Системен_дял
  linux /boot/vmlinuz-linux-libre-lts root=/dev/mapper/Група-Системен_дял rw
  initrd /boot/initramfs-linux-libre-lts.img
}
```

(където под '**Заглавие**' разбираме заглавието, с което желаете да бъде обозначавана в стартовото меню възможността да стартирате системата с файл вместо парола; под '**Стартиращ_клавиш**' – бърз бутон за стартиране по този начин (като не трябва да посочвате клавишите **[a]**, **[b]**, **[e]**, **[c]**, **[d]**, **[o]**, **[p]**, **[r]**, **[s]**, **[t]** и **[u]**); под '**usb0**' – наименованието, под което вашата система най-вероятно ще разпознае цялостно шифрования твърд диск); под '**Група**' и '**Системен_дял**' – съответно наименованието на създадената от вас група и това на системния '**/**' дял).

Ако искате да стартирате шифрованата чрез файл-ключ система без да сте въвели горните инструкции в **Libreboot**-пакета, трябва да въведете ръчно командите в **GRUB** по време на стартирането на компютъра. Трябва първо да закачите външния носител с файла-ключ към системата и да стартирате от **GRUB** командния ред (изберете **[c]**). На всяка от следващите стъпки можете да се ориентирате за наличните възможности чрез команда '**ls**'.

В случай, че външният носител е шифрован, трябва на първо място да го разшифровате:

```
> cryptomount (usb0)
```

... или съответно:

```
> cryptomount (usb0,msdos1)
```

(където наименованията '**usb0**' и '**msdos1**', под които системата разпознава външния носител и неговия формат, могат да варират – ориентирайте се чрез командата '**ls**').

Ще бъде изведено указание за въвеждане на парола и след като сторите това, външният носител ще се разшифрова. Следва да укажете разшифроване на твърдия диск на системата чрез файла, който ползвате като ключ:

```
> cryptomount -k (crypto0)/Адрес/Файл (ahci0)
```

(където наименованието '**ahci0**', под което системата разпознава твърдия диск, може да варира – ориентирайте се чрез команда '**ls**').

Следва ръчно зареждане на шифрования твърд диск от **GRUB**:

```
> root=lvm/Група-Системен_дял

> linux /boot/vmlinuz-linux-libre-lts root=/dev/mapper/Група-Системен_дял rw

> initrd /boot/initramfs-linux-libre-lts.img
```

> boot

(където следва да въведете вашите наименования вместо '**Група**' и '**Системен_дял**'). Ще се зареди операционната система и ще бъде изведен графичният интерфейс. Вече можете да откачите външния носител с файла-ключ от системата и да продължите работата си по обичайния начин.

Ако се наложи да достъпите от помощен компютър твърдия диск на система, шифрована с файл на външен носител, можете да сторите това, като закачите външния носител с файла-ключ към помощния компютър, разшифровате външния носител (ако това е необходимо) и въведете тази команда:

 # cryptsetup open --key-file Адрес/Файл /dev/Устройство Наименование

(където под '**Адрес/Файл**' разбираме файла-ключ с неговото местоположение в паметта на външния носител; под '**Устройство**' – наименованието, под което системата разпознава външния носител; и под '**Наименование**' – наименованието, което вие сте въдели за външния носител при разшифроването му (ако е необходимо).

7. Създаване на шифровани външни хранилища за съдържание

От перспективата на разгледаните по-горе въпроси, създаването на напълно шифровани външни хранилища за потребителско съдържание (външни твърди дискове, **USB**-памет, **SD**-карти) е съвсем лесно, тъй като при тях не стои проблемът за стартиране на намираща се в паметта им операционна система, която в изходното си състояние е напълно шифрована и преди да бъде стартирана, трябва да се разшифрова. Общото за такива хранилища е, че те са предназначени за ползване от вече стартирана операционна система, към която се закачат и чрез приложенията на която се достъпва намиращото се в тях съдържание.

Предимствата на такива външни хранилища е, че (за разлика от евентуално създадения от вас допълнително шифрован '**lock**' дял с поверително съдържание, те не стоят закачени към системата и в случай на неразрешено проникване от internet или внезапно изземване на компютърното устройство, най-вероятно ще останат недостъпни. Обикновено такива външни устройства са малки по размер и се крият лесно; и не се носят постоянно насам-натам – особено ако съдържат информация, която трети страни биха се опитали да достъпят против волята ни.

За да шифровате вашето външно хранилище, трябва да закачите устройството, което ще използвате за тази цел към системата и с разгледаната по-горе команда '**lsblk**' да се ориентирате под какво наименование бива разпознавано. Ще бъде изведен подобен резултат:

```
 NAME                        MAJ:MIN RM   SIZE RO TYPE  MOUNTPOINT
 sda                           8:0    0 100.0G  0 disk
 └─hdd                       254:0    0 100.0G  0 crypt
   ├─Група-Системен_дял      254:1    0    30G  0 lvm   /
   ├─Група-Разменен_дял      254:2    0     8G  0 lvm   [SWAP]
   ├─Група-Потребителски_дял 254:3    0    20G  0 lvm
   └─Група-Шифрован_дял      254:4    0  61.2G  0 lvm
     └─Шифрован_дял          254:5    0  61.2G  0 crypt /Точка_на_зака
чане
 sdb                           8:32   1    10G  0 disk
 └─sdb1                        8:33   1    10G  0 part  /run/media/Пот
ребител/Устройство
```

(където под '**Устройство**' разбираме закаченото от вас устройство (в конкретния пример – **USB**-памет, която системата разпознава като '**sdb**', с единствен дял '**sdb1**'), а като '**/run/media/Потребител/Устройство**' системата обозначава точката на закачане на устройството).

За да можете да пристъпите към шифроване, най-напред трябва да откачите единствения дял на интересуващото ви устройство (или всички негови дялове, ако са повече):

```
# umount /dev/Физически_дял
```

(където под '**Физически_дял**' разбираме наименованието, под което системата разпознава единствения физически дял на външната памет – например '**sdb1**'). След това можете да осъществите цялостно шифроване на устройството като външно хранилище за съдържание (без обособяването на специални дялове), като зададете предпочитаните от вас параметри – например така:

```
# cryptsetup -c serpent-xts-plain64 -s 512 \
-h sha512 -i 2000 luksFormat /dev/Устройство
```

(където под '**Устройство**' разбираме наименованието, под което системата разпознава външната памет – например '**sdb**'). Ще бъде изведено указание да въведете парола. Когато сторите това и шифроването приключи (не забравяйте да си запишете точните параметри на шифроването, защото на по-късен етап може да се наложи да ги въведете ръчно), трябва да отворите шифрованото като външно хранилище устройство:

```
# cryptsetup open /dev/Устройство Наименование
```

(където под '**Наименование**' разбираме наименованието, под което желаете системата да разпознава устройството). Системата ще изведе отново указание да въведете парола и след като сторите това, можете да форматирате устройството:

```
# mkfs.ext4 /dev/mapper/Наименование
```

Можете да закачите устройството към системата, за да го ползвате, така:

```
# mount /dev/mapper/Наименование /Точка_на_закачане
```

(където под '**/Точка_на_закачане**' разбираме директория в системата, от която желаете да достъпвате съдържанието, записано в паметта на устройството). За да направите тази директория достъпна от графичния интерфейс, трябва да укажете на системата да възприема съответната директория като принадлежност на потребителя без администраторски права – чрез описаните по-горе команди '**chmod**' и '**chown**'. В последствие, ако закачате същото устройство със същото наименование към същата точка на закачане, системата ще възприема автоматично съответната директория като принадлежност на потребителя без администраторски права.

Не забравяйте да изключвате шифрованите външни хранилища със съдържание чрез проследените по-горе команди: '**umount /dev/mapper/Наименование**' (за разкачане на хранилището от системата), '**cryptsetup close Наименование**' (за шифроване на съдържанието) и накрая '**udisksctl power-off -b /dev/Устройство**' (за изключване на захранването към съответния **USB**-порт).

8. Критичното значение на Основния ключ и Хедъра

Изложението няма да бъде пълно, ако не отбележим две понятия от критична важност за сигурността на вашата напълно шифрована система. Първото от тях се отнася до Основния ключ (**Volume Key**) на вашите шифровани устройства. Един Основен ключ изглежда по подобен начин:

```
LUKS header information for /dev/Устройство
Cipher name:   aes
Cipher mode:   xts-plain64
Payload offset:4096
UUID:          f11e4960-7db9-8be0-49ce-bb67d63f1fb6
MK bits:       512
MK dump:       2b 09 ac 8f a7 bf 9f 98 5d d5 44 4d 35 c3 ce 0f
               0f bc fc 1e 30 1f 66 d4 5e 31 03 12 a0 61 29 78
               df 4c 8f 43 cc ef 34 8e 13 5d 80 76 ca e0 c7 fe
               50 e1 3b b8 22 7f ed 56 f9 36 bc fe 83 44 cf 53
```

Никога не извличайте Основните ключове за вашите шифровани устройства, освен ако не е наистина необходимо! Можете да извлечете Основния ключ чрез тази команда:

```
# cryptsetup --dump-volume-key luksDump /dev/Устройство
```

Основният ключ се генерира при осъществяване на първоначалното шифроване на съответното устройство и повече не може да бъде променян. Дори и да промените паролите си за достъп, Основният ключ остава същият и чрез него устройството може да се разшифрова, без да е необходима каквато и да било парола. Не случайно системата преди изпълнение на горната команда ще ви предупреди, че Основният ключ трябва да бъде съхраняван на сигурно място и в шифрован вид, и ще изиска да въведете потвърждението '`yes`' с главни букви, преди да ви даде възможност да въведете паролата за извличане на искания от вас Основен ключ. Ако все пак се наложи да извлечете Основния ключ, направете това след като сте се уединили на безопасно място и сте се подготвили за неговото незабавно защитаване и скриване далеч от погледа на трети страни.

Второто понятие от критична важност за информационната сигурност се отнася до резервното копие на Хедъра (**Header Backup**) на вашите шифровани устройства. Един Хедър изглежда по подобен начин:

```
LUKS header information for /dev/Устройство
Version:        1
Cipher name:    aes
Cipher mode:    xts-plain64
Hash spec:      sha512
Payload offset: 4096
MK bits:        512
MK digest:      0a df 4c 62 8f 85 34 f7 a6 e3 0b 6b b2 a0 b8 d5 a1 6f 44 f9
MK salt:        1c 2f a1 8b 48 a5 84 e5 5c c9 4f 56 58 4d 1f 24
                12 4d 6c f1 fa 89 54 d7 dc 0d 61 e8 3c 88 9a ee
MK iterations:  328758
UUID:           f11e4960-7db9-8be0-49ce-bb67d63f1fb6

Key Slot 0: ENABLED
    Iterations:         2703247
    Salt:               9c 5f 2a 62 ce 02 90 2d af dc 0f 1e 00 3d 3a 5d
                        63 54 bf ca 59 4b 70 56 06 f3 04 42 94 64 09 99
    Key material offset:8
    AF stripes:         4000
Key Slot 1: DISABLED
Key Slot 2: DISABLED
Key Slot 3: DISABLED
Key Slot 4: DISABLED
Key Slot 5: DISABLED
Key Slot 6: DISABLED
Key Slot 7: DISABLED
```

От горния пример научаваме, че за интересуващото ни шифровано устройство има една парола (данните в слот '**0**'), докато останалите **7** възможни слота стоят изключени (**DISABLED**). В тази връзка трябва да отбележим, че **LUKS** шифроването позволява да настроите няколко различни пароли, които да действат паралелно и независимо една от друга. Ще се върнем на този въпрос отново в Част **V**, докато обсъждаме сигурността на паролите.

Ако на по-късен етап в системата настъпи срив и по някаква причина Хедърът бъде повреден, няма да има никаква възможност да разшифровате устройството, освен ако не разполагате с резервно копие, което да поставите на мястото на повредения Хедър. В този смисъл не е излишно да направите резервно копие от Хедъра, което да съхранявате в шифрован вид (и отделно от паролата, с която може да бъде ползвано), на сигурно място извън шифрованото устройство, за което се отнася.

Резервно копие на Хедъра можете да извлечете чрез тази команда:

```
# cryptsetup luksDump /dev/Устройство
```

(при което в терминала ще бъде изведено в обикновен текстови формат съдържанието на Хедъра). Можете да изведете резервно копие от Хедъра и обособено в отделен текстови файл, чрез тази команда:

```
# cryptsetup luksHeaderBackup /dev/Устройство \
--header-backup-file Хедър.backup
```

(където под '**Хедър.backup**' разбираме наименованието на текстовия файл, в който желаете да бъде обособено резервното копие от Хедъра).

Ако в следващ момент Хедърът на съответното устройство бъде повреден и вие разполагате с резервно копие, ще можете да го възстановите чрез тази команда:

```
# cryptsetup luksHeaderRestore /dev/Устройство \
--header-backup-file Хедър.backup
```

По този начин ще имате възможност отново да достъпите вашето шифровано съдържание – с паролата, която е била валидна към момента на извличане на резервното копие на Хедъра. Казано с други думи – дори и след време да промените паролата, ако някой разполага с предишната версия на Хедъра и знае тогавашната парола, ще може да разшифрова системата – дори и ако вече сте сменили паролата.

Сега, когато сте в състояние да създавате резервни копия на Хедъра, можете да направите още една стъпка напред за обезпечаване на вашата информационна сигурност – като изтриете Хедъра от вашето шифровано устройство (след като сте си осигурили съхранявано на сигурно място копие от него), чрез тази команда:

```
# head -c 2097152 /dev/urandom > /dev/Устройство; sync
```

След изпълнението на тази команда приблизително **2MiB** (съответно **2097152kB**) от началото на паметта в съответното устройство ще бъде заменена с рандомизирани данни. Евентуален последващ анализ би установил, че на това място в устройството е записана само произволна (рандомизирана) информация (представляваща произволни символи). Вероятният извод би бил, че това е повредено шифровано пространство или такова, което е рандомизирано, но все още не е шифровано. При техническа проверка с тази команда:

```
# cryptsetup -v isLuks /dev/Устройство
```

... ще бъде изведен подобен отрицателен резултат:

```
Device /dev/Устройство is not a valid LUKS device.
Command failed with code 22: Device /dev/Устройство is not a valid LUKS
device.
```

... вместо подобен положителен резултат (съответстващ на шифровано пространство с валиден Хедър):

```
Command successful.
```

Когато желаете да ползвате вашето шифровано устройство, трябва да възстановите Хедъра с проследената по-горе команда '**luksHeaderRestore**' и да го разшифровате.

Разгледаният способ е полезен в две направления. Първото е в контекста на деспотични политически режими или рестриктивна корпоративна политика, където самото притежаване на шифровано устройство може да се превърне в източник на неприятности за вас. Възможно е да бъдете поставени под натиск да разкриете паролата за разшифроване на устройството – което в някои случаи няма как да откажете да направите. Ако обаче не разполагате с Хедър (било защото умишлено или без да искате сте го изтрили, било защото е настъпила повреда), дори да предоставите истинската парола, разшифроване не може да се извърши. Второто направление е в контекста на съществуващите технологични възможности за разбиване на паролата чрез прилагане на bruteforce методи. Ако Хедърът е достъпен, при наличието на достатъчно мотивация и изчислителни ресурси разбиването на паролата е въпрос на време – независимо от нейната дължина и сложност. Далеч не така стоят нещата обаче, ако Хедърът не съществува (или ако не бъде намерен). В този смисъл извличането на резервно копие от Хедъра и унищожаването му от шифрованото устройство може да се окаже здрава защита срещу принуждаването ви да разкриете паролата и срещу подлагане на bruteforce методи за разбиването ѝ.

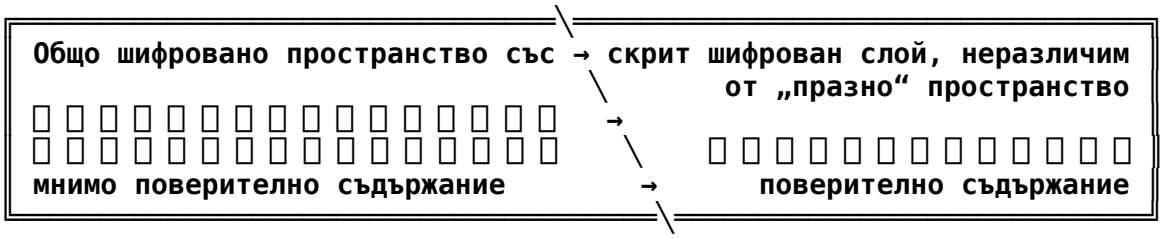
Важно е да имате предвид, че Хедърът се променя всеки път, когато смените, добавите или изтриете парола за разшифроване на съответното устройство. Следователно – ако трета страна се сдобие с резервно копие от Хедъра и

узнае паролата, която е била валидна към момента на извличане на резервното копие, ще може да разшифрова системата – дори и ако вече сте сменили паролата.

Важно е да знаете и това, че Хедърът е достъпен за извличане включително и когато вашето устройство е в напълно шифрован вид – защото самият той стои нешифрован в началото на паметта, за да бъде приложен при въвеждане на паролата преди разшифроването. Следователно – от критична важност остава защитата на самата парола, приложима с конкретен Хедър (конкретна версия на Хедъра); отделно от физическата сигурност на самото устройство, от което не желаем никой да извлича никаква критична информация – на която сигурност ще се спрем в Част **VII** по-долу.

9. Обособяване на скрити шифровани пространства

И преди да завършим темата за цялостно шифроване на вашата система, ще се спрем на възможността да обособите скрити шифровани слоеве в едно общо шифровано пространство. Това е възможно благодарение на цялостната рандомизация, която предхожда първоначалното шифроване; и която прави целият обем на шифрованото пространство да се хомогенизира в обща шифровъчна текстура. В такава текстура е много трудно да се определи коя част е запълнена с изглеждащите като „произволни“ символи от поверителното съдържание, преминали през шифровъчния алгоритъм; и коя част е запълнена с наистина произволни (рандомизирани) символи от празното пространство, също преминали през шифровъчния алгоритъм. Именно в произволните (рандомизирани) символи от едно такова „празно“ пространство можете да обособите скрит шифрован слой с пространство за съхраняване на такова поверително съдържание, за което не желаете дори да възникнат съмнения, че съществува (а още по-малко – да бъдат събрани технически проверими индикации за неговото съществуване на конкретно място във вашето устройство). Така, ако все пак бъдете принудени да разкриете паролата си, ще можете да разкриете тази за отваряне на общото шифровано пространство, в което държите определена мнима поверителна информация. Истинските ви тайни обаче ще останат скрити в „празната“ част на общото шифровано пространство.



общо шифровано пространство със скрит шифрован слой

За първи път тази възможност стана публично достъпна през **2004** г. чрез приложението **TrueCrypt**, предлагащо цялостно шифроване в реално време (**on-the-fly**). Проектът беше внезапно ликвидиран на **28.05.2014** г. от специалните служби на **САЩ**, като екипът на **TrueCrypt** очевидно е бил заставен да обяви своята разработка за „ненадеждна“ и да даде смехотворен съвет към потребителите да ползват вградената функция **BitLocker** на **Windows**. През **2015** г. няколко независими изследвания (в т.ч. на Федералния офис за информационна сигурност в Германия) потвърдиха за пореден път, че в **TrueCrypt** не се установяват критични уязвимости. След ликвидирането му проектът беше продължен в няколко паралелни разработки, най-съществената от които е **VeraCrypt**. За съжаление обаче и **TrueCrypt**, и последващата **VeraCrypt** не са напълно свободни – поради което не можем да препоръчаме тяхното прилагане, независимо от техните доказани предимства.

https://en.wikipedia.org/wiki/TrueCrypt
https://www.veracrypt.fr/

Вашите скрити шифровани слоеве не трябва да разполагат с Хедър (по-горе се спряхме върху критичното значение на Основния ключ и Хедъра) – тъй като съществуването на подобен атрибут би направило съществуването на шифрованите слоеве очевидно. Вместо това ще използваме прост алгоритъм за шифроване без Хедър, преобразуващ произволните (рандомизирани) символи от една обособена „празна“ част в общото шифровано пространство в скрит

шифрован слой, който (докато се намира в шифровано състояние) остава неразличим от произволните (рандомизираните) символи в коя да е друга „празна“ част от общото шифровано пространство. Следва обаче да имате предвид, че отсъствието на Хедър и Основен ключ лишава вашето скрито шифровано пространство от специфичните технологии за забавяне прилагането на въведената парола (предположение за парола) при опит за разшифроване. Следователно, ако се установи наличието и местонахождението на скрит шифрован слой и бъде приложена **bruteforce** атака, една достатъчно мощна система (каквато най-вероятно ще се ползва за провеждането на такава атака) ще пробва много бързо значителни обеми от възможни пароли. Ето защо в такъв случай като единствена и основна преграда срещу проникване във вашите тайни ще остане прилагането на възможно най-сложни и комплексни пароли, които да е трудно да бъдат уцелени – въпреки липсата на забавяне. В тази връзка е важно да разгледате внимателно въпроса за сложността и комплексността на паролите, на който ще се спрем в Част **V** по-долу.

Тук на помощ идва едно друго свойство на простото шифроване без Хедър. Алгоритмите за просто (раз)шифроване без Хедър работят безкритично по отношение на въведената парола – и извършват преобразуване на съответния набор от символи в съответствие със зададения шифровъчен алгоритъм, параметри и парола, без да извеждат индикация дали алгоритъмът, параметрите и паролата са верни или не са. Така, ако направите опит да разшифровате един скрит шифрован слой с неправилна парола, ще получите набор от символи, неразличим от произволните (рандомизирани) символи в едно правилно разшифровано празно пространство; от произволните (рандомизирани) символи в едно погрешно разшифровано пространство; или от произволните (рандомизирани) символи в едно празно пространство, което просто не е шифровано. Следователно можете да направите няколко последователни скрити шифровани слоя, при което дори да бъде уцелена правилната първа парола, остава да се уцели правилната втора парола и т.н. – за да бъде изведен смислен резултат (вашето поверително съдържание). Можете следователно в пространството на един скрит шифрован слой да въведете определено мнимо поверително съдържание, а в „празното“ пространство на този слой да създадете следващ скрит шифрован слой с действителното поверително съдържание. Важно е обаче в такъв случай да въведете най-напред в първия скрит шифрован слой мнимото поверително съдържание и едва след това да създадете следващия скрит шифрован слой с действителното поверително съдържание. След като сторите това, не трябва да променяте нищо в първия скрит шифрован слой – и ако все пак достъпвате мнимо поверителното съдържание (например за да изглежда, че това съдържание е важно за вас), трябва да го достъпвате в режим само за четене (**Read Only**). В противен случай рискувате системата (която „не подозира“ за съществуването на следващия скрит шифрован слой) да направи запис в привидно „празното“ пространство и по този начин да нанесе поражения в символите, които участват в (раз)шифроването на следващия скрит шифрован слой. Резултатът от подобен инцидент би бил пълно или частично унищожаване на вашето действително поверително съдържание.

За да създадете скрит шифрован слой (или няколко последователни скрити шифровани слоя), на първо място трябва да рандомизирате общото шифровано пространство (обособен файл с определен от вас размер; или обособен дял във вашите устройства, било на външен носител, било на твърдия диск на компютъра):

– ако работите в обособена директория или файл:

```
# dd if=/dev/urandom of=Директория_или_файл bs=1M count=1 \
status=progress oflag=sync
```

(където под '**1M**' разбираме размерите на обособената директория или файл – в нашия пример **1MB**);

– ако работите на обособено устройство:

```
# dd if=/dev/urandom of=/dev/Устройство bs=64M \
status=progress oflag=sync
```

(където под '**Устройство**' разбираме наименованието, под което системата разпознава съответното устройство).

След като рандомизацията бъде изпълнена, следва да извършите самото просто шифроване без Хедър, като зададете желаните параметри на шифроването (и не забравяте преди това да закачите чрез командата '**mount**' устройството, ако работите на външен носител):

```
# cryptsetup open --type plain -h Хеш_алгоритъм -c Шифър \
-s Дължина /dev/Устройство Наименование
```

(където под '**Хеш_алгоритъм**' разбираме предпочитания от вас алгоритъм за хеширане (например '**sha512**' или '**sha256**'); под '**Шифър**' разбираме предпочитания от вас алгоритъм за просто шифроване без Хедър (например '**serpent-xts-plain64**'); под '**Дължина**' указваме дължината на шифровъчния ключ (например '**512**'); под '**Устройство**' разбираме наименованието, под което системата разпознава вашето устройство (например '**sdb**'); и под '**Наименование**' разбираме наименованието, под което желаете системата да разпознава вашето устройство). Така горната команда може завършена да придобие подобен вид:

```
# cryptsetup open --type plain -h sha512 \
-c serpent-xts-plain64 -s 512 /dev/sdb Hidden
```

Системата ще изведе указание да въведете парола. Важно е да съобразите, че след еднократното въвеждане на паролата, директно ще се пристъпи към осъществяване на указаното шифроване; няма да бъде изведено нито указание за потвърждаване на паролата, нито ще бъде създаден Хедър. Резултатът от изпълнението на командата ще бъде набор с описаните по-горе рандомизирани символи (резултат от преминаването през указаното шифроване), които са принципно неразличим от първоначалния набор с рандомизирани символи. Можете след това да осъществите още няколко шифрования с горната команда – всеки път ще получавате набори с рандомизирани символи, принципно неразличими от който и да било друг набор с рандомизирани символи. Запишете си някъде точните параметри на шифроването – защото на по-късен етап ще трябва да ги въведете точно, ако искате да извършите правилно разшифроване.

Когато скритият шифрован слой бъде създаден, можете да поставите в него вашето поверително съдържание (за която цел трябва да го монтирате, ако работите на външен носител с памет. След като приключите с поверителното съдържание, трябва да разкачите устройството от системата, ако работите на външен носител (командата '**umount**') и да затворите последователно всеки от скритите шифровани слоеве – като започнете от последния и достигнете до първия:

```
# cryptsetup close Наименование
```

И преди да завършим темата – още една особеност на последователните скрити шифровани слоеве. На първо място – при подобна последователност от слоеве става трудно забележимо евентуалното уцелване на паролата за първия слой и съответно последващо насочване на усилията към уцелване на паролата за следващия слой. И на второ място – при подобна последователност от слоеве можете да поставите мнимо поверително съдържание в първия слой, а действителното поверително съдържание да скриете в следващ шифрован слой, чието съществуване се надявате да не бъде установено при обследване на устройството. В този случай е важно да прецените правилно размерите на общото шифровано пространство и да разположите мнимото поверително съдържание и следващия скрит шифрован слой (следващите скрити шифровани слоеве) на различни места в общото шифрована пространство – така, че разполагането на едно от тях върху определен набор от символи да не засегне набора от символи, върху които се намира другото.

Това обаче е по-сложно отколкото изглежда на пръв поглед. Различните файлови системи имат различни комплексни логики за разполагане на информацията в пространството – което означава, че при запис на съдържание, което отнема дори съвсем малка част от празното пространство на съответния слой, част от него могат да презапишат произволни места в „празното“ пространство – включително върху части от скритите шифровани слоеве. Същевременно, при запис на съдържание във вътрешните слоеве, части от това съдържание могат да презапишат произволни места от външните слоеве. Споменатите презаписвания могат да се случат включително върху места с техническа мета-информация от файловата система, което да остане неочевидно за обикновения потребител. Но при

последващо обследване подобни „дупки“ в техническата мета-информация може да се превърне за специалистите в индикатор за съществуването на скрити шифровани пространства в системата.

Най-добрият начин да избегнете този проблем, е да създадете слоевете, като форматирате (примерно с '**mkfs.ext4**') и запишете информация първо в най-вътрешния слой, след което да форматирате и запишете информация в по-външните и по-външните слоеве (внимавайки да не записвате прекалено много информация в по-външните слоеве, за да не презапишете информацията от по-вътрешните слоеве). След приключване със записванията трябва да сверите дали важната за вас информация във вътрешните слоеве не е повредена. Ако искате да добавите нова информация в някой от шифровъчните слоеве, трябва да повторите цялата процедура, за да не допуснете образуването на „дупки“ от липсваща мета-информация във файловите системи на някой от по-външните слоеве.

Да укажете следващият скрит шифрован слой да бъде създаден „по-напред“ в общото шифровано пространство можете чрез този параметър:

-o Число

(където под '**Число**' разбираме количество блокове от по **512B** всеки, с които желаете да отстъпите навътре от началото на общото шифровано пространство). Така горната команда може завършена да придобие подобен вид:

 # cryptsetup open --type plain -h Хеш_алгоритъм -c Шифър -o Число \ /dev/Устройство Наименование

За да зададете правилен параметър за '**Число**', е необходимо да изчислите в блокове от по **512B** всеки общия размер на мнимото поверително съдържание; да оставите известен запас от действително празно пространство между края на мнимото поверително съдържание и началото на следващия скрит шифрован слой; и да определите такъв размер на следващия скрит шифрован слой (отново в блокове), щото неговият край да не попада извън размера на общото шифровано пространство (като не е излишно тук отново да оставите известен запас от действително празно пространство). Да определите размера на следващия скрит шифрован слой можете чрез този параметър:

--device-size Число

(където под '**Число**' отново разбираме количество блокове от по **512B** всеки, които обаче този път желаете да заеме следващият скрит шифрован слой в общото шифровано пространство, започвайки или от самото му начало, или от „по-напред“, ако сте указали това съгласно посоченото по-горе). Така горната команда може завършена да придобие подобен вид:

 # cryptsetup open --type plain -h Хеш_алгоритъм -c Шифър -s Дължина \ -o Число --device-size Число /dev/Устройство Наименование

Следва да имате предвид, че един блок от **512B** се нанася **2'048** пъти в **1MB** (който от своя страна съдържа **1'048'576B**). Следователно при по-големи файлове може да достигнете до значителни числа, които трябва да внимавате да не объркате – особено при пресмятане размера на мнимото поверително съдържание – за да се ориентирате правилно за това колко „по-напред“ в общото шифровано пространство да започват следващите скрити шифровани слоеве. Ако направите грешка в това изчисление (и съответно – ако на следващ етап въведете неправилни стойности при опит за разшифроване), рискувате да унищожите изцяло или отчасти мнимото поверително съдържание и да поставите под въпрос евентуалните твърдения, че наистина това е вашата тайна и друго няма. Ако файловете бъдат повредени, остава открит въпросът какво е причинило тяхното повреждане – и предполагаемият отговор не е изключено да тласне обследващите към търсенето на следващи скрити шифровани слоеве.

И накрая – не е излишно от време на време да достъпвате мнимото поверително съдържание – ако искате да придадете реалистичност на неговата важност (и от там – на причините то да бъде укривано). Важно е обаче това достъпване да се осъществява в режим само за четене (**Read Only**), за да предотвратите риска системата да извърши запис в общото шифровано пространство, който запис може да се окаже където и да било в „празното“ пространство –

включително върху символи, които попадат в следващите скрити шифровани слоеве. Ако допуснете това, ще унищожите изцяло или отчасти тези слоеве и скритото в тях действително поверително съдържание. За да не се случи подобен инцидент, въведете параметъра '**-r**' в командата за отваряне на първия скрит шифрован слой – при което горната команда може завършена да придобие подобен вид:

```
# cryptsetup open --type plain -h Хеш_алгоритъм -c Шифър -s Дължина -r
```

... или респективно (ако сте въвели указания за създаване на скрития шифрован слой „по-напред“ и не до края на общото шифровано пространство) – при което горната команда може завършена да придобие подобен вид:

```
# cryptsetup open --type plain -h Хеш_алгоритъм -c Шифър -s Дължина \
-r -o Число --device-size Число /dev/Устройство Наименование
```

И в двата случая съответният скрит шифрован слой ще бъде отворен в режим само за четене – който се препоръчва да бъде единственият при достъпване на мнимото поверително съдържание, след като вече сте създали следващите скрити шифровани слоеве в общото шифровано пространство и евентуално сте разположили в тях действителното поверително съдържание.

И накрая – трябва дебело да подчертаем наличието на редица рискове за информационната сигурност при прилагането на скрити шифровани пространства – както от техническо, така и от организационно естество.

На първо място трябва да имате предвид, че скритите шифровани слоеве са зависими от привидно празните части в общото шифровано пространство. Ако тези „празни“ части бъдат запълнени с някакво съдържание в общото шифровано пространство, привидно рандомизираните символи (които в действителност са части от скрития шифрован слой) може да бъдат променени и съответна част от скритото съдържание (или изобщо цялото скрито съдържание) да бъде унищожена. Това е така, защото системата не отчита по никакъв начин съществуването на скрит шифрован слой – в противен случай това би могло да се проследи и съществуването на този слой да бъде разкрито. Ето защо трябва да внимавате обемът на мнимо поверителното съдържание в общото шифровано пространство да не надхвърля размера на общото шифровано пространство, от който е изваден един привидно празен обем, задължително надхвърлящ значително размера на скрития шифрован слой. Добра практика в тази връзка е да си оставите известен буфер от допълнително свободно пространство. Също така трябва да се въздържате от записвания и изтривания на мнимо поверително съдържание, защото системата (дори и да не е запълнен буферът от допълнително свободно пространство) може да направи нови записи в сектори, които разпознава като свободни, но в действителност са заети от скрития шифрован слой – и по този начин да нанесе поражения на скритото съдържание.

Желателно е скритият шифрован слой (и съответно – привидно празното пространство, в което е поместен) да е с колкото се може по-малки размери. Така е по-малко вероятно да възникне съмнение, че в „празните“ части от общото шифровано пространство съществуват скрити шифровани слоеве. Чисто практически, един надеждно шифрован външен твърд диск с размери **1 TB** ще събуди доста подозрения, ако на него няма нищо друго, освен самотен видео-файл от **20 MB** с бебешки рожден ден и няколко снимки с цветя!

Трябва да имате предвид, че обособяването на скрити шифровани слоеве все пак не е напълно сигурен способ. Чрез прилагането на фини аналитични методи е възможно да се установят подозрителни флуктуации в „празните“ части от общото шифровано пространство, които не съответстват напълно на действително празно пространство със случайни (рандомизирани) сивмоли, а може би на шифровано съдържание. Възможно е в тази връзка да ви бъде „пробутано“ поверително съдържание с вградени т.нар. „водни знаци“, които се очаква да се отразят по определен идентифицируем начин на шифровъчната текстура – и това да позволи достигането до извод за съществуването на скрити шифровани слоеве, където е поставено маркираното с „водни знаци“ съдържание.

Сходни последици може да има и продължителното наблюдаване на „празните“ части на общото шифровано пространство. При отваряне на скрития шифрован слой и променяне на съхраняваното в него поверително съдържание, в привидно празното пространство (състоящо се от привидно произволни символи) ще се промени. От там възниква въпросът защо това празно място се променя? Промяната на съдържанието в шифрования слой може да

стане дори без да осъзнаете – ако например някоя програма направи запис на техническа мета-информация във файловата система – затова е силно препоръчително да отваряте скрития слой в режим на четене ('**-r**').

Скритите шифровани слоеве могат да бъдат компрометирани също и от софтуера, посредством който достъпвате, обработвате и записвате скритото съдържание. В много от случаите вашата операционна система и конкретните потребителски приложения запазват резервни копия от файловете, които обработвате или водят журнал относно вашите активности – кои файлове сте отваряли, къде се намират тези файлове, какви устройства сте закачали към системата и т.н. Тези резервни копия и журнални записи могат да разкрият, че например скоро сте обработвали файл, намиращ се на място, което не изглежда да съществува в системата. Опитен специалист бързо би достигнал до извод, че става дума за скрит шифрован слой, който „не съществува“, просто защото не е закачен към системата чрез въвеждане на истинската парола. Добра практика в тази връзка е да създавате и достъпвате вашите скрити шифровани слоеве единствено от операционна система на „жив“ носител, на който е отнета функцията по записване на извършените действия. По този начин в системата няма да остават никакви журнални записи за извършените действия, от които да може да се направи извод за достъпването на скрити шифровани слоеве.

Обикновено в „официалните“ шифровани пространства се поместват „поверителна“ информация, която обаче не е толкова важна, колкото поверителната информация в скритите шифровани слоеве. Често срещана грешка при това е като „официална“ поверителна информация да се подбира очевидно несъществено съдържание, което се превръща в индикатор за това, че съществената информация е някъде другаде, може би в скрит шифрован слой. Друга често срещана грешка е „официалната“ поверителна информация да бъде изоставена след поместването ѝ в общото шифровано пространство, докато активната дейност продължава с действително поверителното съдържание от скрития шифрован слой. Вече стана дума за резервните копия и журналните записи, които операционните системи поддържат при работа с файлове. Ако бъдете принудени да разкриете вашата „официална“ парола и се установи, че „официалната“ поверителна информация не е достъпвана никога от създаването си или е достъпвана много отдавна, ще възникне въпросът не съществува ли и друго поверително съдържание, с което работите по-активно?

Тънък момент е и това след какви усилия и при какви обстоятелства ще склоните да разкриете вашата „официална“ парола, и какво ще бъде поведението ви във връзка с разкриването на мнимото поверително съдържание. Ако разкриете паролата твърде лесно, ще последва въпросът дали не криете и друга парола, която бихте разкрили по-трудно? Ако поведението ви е спокойно след разкриването, ще последва въпросът дали спокойствието ви не е заради това, че действителните ви тайни са някъде другаде? Имайте предвид, че в такива ситуации преиграването и вкарването на пресилена емоция се разпознава точно толкова лесно, колкото и граничещата с безразличие инертност. Съществува и още един риск – да бъдете подложени на масиран натиск дори и тогава, когато вече сте разкрили наистина всичките притежавани от вас пароли – просто защото не съществува доказателство, че във вашата система няма и още един скрит шифрован слой.

И още – достатъчно опитният обследващ в повечето случаи може със значителна степен на сигурност да идентифицира вашите твърдения под натиск, че в обследваните устройства няма поверителна информация, като истинни (като такива, в които вие действително вярвате); или като неистинни (като такива, в които просто не вярвате) – колкото и добре да се опитвате да „играете“ ролята си на вярващ и казващ всичко, което знае.

Не са изолирани и такива случаи, при които се прилагат (например чрез измама или просто чрез принуда) специални психоактивни вещества, които ограничават волята за съпротива и стимулират склонности към сътрудничество. Под въздействието на такива вещества обследваният често действа в еуфория или в ирационален страх, често без да може да осъзнае, че всъщност разкрива тайните си; или без да може да се противопостави на това; като в много от случаите след преминаване на психоактивното въздействие дори не е в състояние да си спомни какво точно се е случило и дали се е случило наистина.

Ето защо – въпреки възможността да бъдат обособявани скрити шифровани слоеве, чието съществуване е технически трудно установимо – като най-добра алтернатива все пак остава това да се погрижите вашата поверителна информация просто да не се намира на устройствата, които някой би обследвал за извличането ѝ. Фрагментирайте вашата информационна сигурност така, че компрометирането на който и да било слой от нея да доведе до компрометиране на колкото се може по-малко елементи от вашата поверителна информация. В това фрагментиране

евентуалните скрити шифровани слоеве далеч не са панацея – бъдете гъвкави и не разчитайте на никакви „непробиваеми“ решения!

V. ОБЕЗПЕЧАВАНЕ СИГУРНОСТТА НА ПАРОЛИТЕ

Съвременните криптографски технологии са толкова мощни, че ако функционират правилно, предоставят практически непреодолима защита (освен ако бъдат приложени неразумно големи усилия в неразумно продължителни периоди от време). Криптографските технологии обаче са напълно безпомощни, ако паролата към тях бъде компрометирана. Често се изтъква, че паролата е „най-слабото звено“ от целия криптографски механизъм. Ето защо е необходимо да се ползват надеждни пароли и въвеждането им да се осъществява с особено внимание. Една парола е надеждна, когато е трудна за извеждане от наличната информация за нейния притежател, когато е трудна за уцелване при композиране на произволен брой случайни или подбрани (вероятни) комбинации, когато не може да бъде намерена изписана някъде (примерно на лист хартия, оставен до компютъра) и когато не може да бъде проследено въвеждането ѝ от нейния притежател.

1. Съставянето на достатъчно трудни за разбиване пароли

На първо място, паролата не бива да съдържа никакви ваши или на близките ви имена, дати на раждане, телефонни номера, адреси, професионална терминология или жаргонни изрази от вашето обкръжение. Паролата не бива да съдържа „очаквани“ комбинации като '**123456**', '**parola**' или '**ABCDEFGH**'. Такива комбинации са сред първите, с които би се направил опит за уцелване на вашата парола при прилагането на **bruteforce** методи (последователно автоматизирано опитване на предполагаеми комбинации, докато паролата бъде най-после уцелена); които методи при достатъчно мощни компютърни системи са в състояние да преодолеят милиони и милиарди по-предвидими комбинации за няколко часа, до дни. Паролата не бива да следва никакви предвидими последователности – каквито са съществуващите думи в речниците (на познатите езици като например '**zelenagora**' или '**Joliot-Curie**'), нито стандартните последователности на клавишите в клавиатурата – например '**qwerty**' или '**1qaz2wsx**'.

Паролата трябва да бъде сложна – по възможност да съдържа символи от няколко или всичките четири символни групи: цифри като '**2**', '**0**' и '**9**'; малки букви като '**v**', '**e**' и '**r**'; големи букви като '**U**', '**A**' и '**S**'; и специални символи като '**<**', '**;**' и '**$**'. Това допълнително увеличава възможните комбинации и намалява шанса за уцелване на паролата при bruteforce (или поне го отлага за период, чиято продължителност при съществуващите изчислителни мощности трябва да бъде толкова необосновано голяма, че практически да обезсмисли опитите за случайно уцелване). Заедно с това паролата трябва да бъде дълга – тъй като всеки следващ символ увеличава експоненциално броя на възможните комбинации и обезсърчава прилагането на bruteforce методи. За да се ориентирате за „стойността“ на всеки следващ символ в паролата, можете да изчислите броя на възможните пароли чрез следната формула:

Б^Д

(където под „**Б**“ разбираме Броя възможни символи (колко различни символи са достъпни за ползване), а под „**Д**“ разбираме Дължината на паролата (колко символи сме включили в състава на конкретната парола); броя възможни Символи повдигаме на степен Дължината на паролата). Ако например ползвате само цифрите от **0** до **9** и съставите парола от **5** символа, това означава общо **10** възможни символа, повдигнати на степен **5** (колкото са символите във вашата парола) или **10^5 = 100'000** възможни пароли от по **5** цифри (преодолими при bruteforce за по-малко от секунда). Ако добавите и малките букви на латиницата (общоутвърдената **US**-стандартизация), това означава още **26** символа (или всичко **36** символа с цифрите), което означава **36^5 = 60'466'176** възможни пароли от по **5** символа (което е с почти **605** пъти повече от възможните **5**-символни пароли, ако ползвате само цифри). Общият брой на четирите символни групи съгласно стандартизацията **ASCII** (**American Standard Code for Information Interchange**) е **95** – или **95^5 = 7'737'809'375** възможни пароли от по **5** символа (което е с почти **128** пъти повече от възможните **5**-символни пароли, ако ползвате само цифри и малки букви). На фона това – ако добавите към паролата дори само още **1** символ, това вече ще означава **95^6 = 735'091'890'625** възможни пароли (което е **95** пъти повече от възможните **5**-символни пароли със същия обем възможни символи от четирите основни групи).

Предвид съществуващите изчислителни мощности към настоящия момент се препоръчват пароли от **12** и повече символа в няколко или всичките символни групи от стандартизацията **ASCII**. Ако към това обаче добавите и символи от стандартизацията **Extended ASCII** (където попадат и символи като например '**Ñ**', '**¼**', '**®**', '**µ**', '**€**', '**ß**', '**†**', '**¿**', '**Æ**' и '**»**'),

общият брой възможни символи ще достигне **218** – при **12**-символна парола това ще означава **218^12 = 11'520'674'946'182'735'813'538'942'976** възможни пароли (което е с **21'320,37** пъти повече от възможните **12**-символни пароли със символи само от основната стандартизация **ASCII**). Нещата стават дори още по-комплексни при съвременната **Unicode** стандартизация, която в актуалната си за момента версия **15.1** включва **149'813** символа от почти всички познати езици и писмености. Трябва обаче да имате предвид, че използването на символи от стандартизацията **Extended ASCII** или **Unicode** не е удобно за пароли, с които например инициализирате системата – тъй като те в повечето случаи не са достъпни от клавиатурата (а от допълнителни приложения като **Character Map** например); съответно е трудно да ги ползвате преди да сте установили достъп до тези приложения (или поне до прост текстови файл с необходимите ви символи, от където наличният интерфейс ви позволява да копирате). Няма обаче никаква пречка да ползвате такива символи за защитаване на допълнително шифровани дялове от вашата система, шифрованите външни хранилища, скритите шифровани пространства, шифровъчни ключове и т.н.

`https://en.wikipedia.org/wiki/ASCII`
`https://en.wikipedia.org/wiki/Extended_ASCII`
`https://en.wikipedia.org/wiki/Unicode`

За да се получи достатъчно дълга, трудна за уцелване и все пак лесна за запомняне парола, могат да се изписват словосъчетания и изрази, при които например част от буквите са заместени асоциативно (не общоутвърдена асоциация като например '**4**' вместо българското '**ч**' или английското '**for**', а ваша лична асоциация) с други букви, символи или цифри – за да се получат непредвидими „безсмислици“. Така например в израза '**АзОбичамБира**' част от буквите може да се въведат като цифри, които ги наподобяват графично – и да се получи '**A3O6i4am6ira**' (не ползвайте точно такъв пример – още повече, че той не отговаря на изискването за необщоутвърденост). Също така паролата може да бъде израз, въведен на определен езиков стандарт, докато компютърът работи на друг езиков стандарт. Така например при въвеждане на '**АзОбичамБира**' по Българския държавен стандарт (**БДС**), докато компютърът работи на латиница (общоприетия стандарт **QWERTY**), се получава „безсмислицата“ '**DpF/r'd;?r,d**'. Предимство на предложените методи (които няма пречка да комбинирате) е, че правят паролата трудно предвидима и същевременно ви дават възможност да я запомните лесно. Трябва обаче да имате предвид, че подобни методи са широко известни на специалистите по разбиване на пароли – и за да не бъдат лесно компрометирани, е необходимо да ги комбинирате с допълнителни усложняващи фактори.

Специфичен усложняващ фактор е прилагането на различни символи от множеството таблици в **Unicode** стандартизацията – например арабски или еврейски абджад, индийски деванагари, корейски хангъл, японски канджи, емотикони. Вашата парола би могла да придобие неузнаваем и много труден за непреднамерено уцелване вид – например '**Åʒ**Øbи 茶 **µБ𝑖®α**'. Трябва обаче да имате предвид, че подобни екзотични комбинации (и въобще всички символи извън обикновената **ASCII** стандартизация) са зависими от настройките на системата и от допълнително инсталираните софтуерни пакети и шрифтове. Възможно е в един следващ момент системата да не работи по същия начин, по който е работела когато сте съставяли паролата (например поради инсталиране на нови пакети или просто поради обновяване на софтуера). Подобни усложнения може да доведат до това да не можете да извлечете необходимите ви символи, за да въведете паролата; или макар и правилно въведени, символите да не бъдат интерпретирани по правилния начин и паролата да бъде отчетена като погрешна. Това е така, защото компютърните системи запазват всички символи като поредица от цифрени кодове, които могат да варират в зависимост от софтуера. Ето защо ползването на пароли със специфични символи не се препоръчва, ако е възможно да възникне необходимост паролите да се въвеждат в импровизирани условия (например при срив на системата и необходимост от инцидентно достъпване на нейното съдържание, без да се ползва обичайният софтуер и настройки). Частично решение на този проблем е да си направите „сборник“ с голям брой специални символи, между които в разбъркан ред и необходимите ви за вашата парола. Но това може също да не проработи в даден момент – защото символите могат да бъдат интерпретирани като „неправилни“ буквено-цифрени кодове или изобщо да не бъдат интерпретирани – като бъдат изведени например в подобен неразличим и безполезен вид: '□□□□□□□□□□□□'. Усложняването на паролите чрез включването на специални символи е приемливо преди всичко за защита на ресурси, които не се очаква да бъдат достъпвани в извънредна ситуация (например допълнително шифровани вътрешни за операционната система хранилища с информация, които се очаква да бъдат достъпвани само в правилно функциониращата операционна система).

Създаването на голям брой пароли с необходимата дължина и сложност е трудна задача особено при системното администриране, където се налага да бъдат генерирани голям брой пароли. Това мотивира някои потребители да се обърнат към услугите на софтуерни генератори на пароли. Софтуерните генератори разчитат на псевдослучайни числа, които да осигурят необходимата рандомизация за създаването на достатъчно сигурна (непредвидима) парола. Можете да ползвате вграден във вашата **GNU/Linux**-система генератор на псевдослучайни числа:

```
$ openssl rand -base64 Брой_символи
```

(където под '**Брой_символи**' разбираме броя символи, от които желаете да бъде съставен псевдослучайният резултат). Последният ще включва цифри, малки и големи букви съгласно основната стандартизация **ASCII** с указания брой символи. Можете да ползвате и друг генератор на псевдослучайни числа, който се поддава на по-прецизни настройки, чрез тази команда:

```
$ tr -dc [:Вид_символи:] < /dev/urandom | head -c Брой_символи; echo
```

(където под '**Вид_символи**' разбираме вида символи, от които желаете да бъде съставен псевдослучайният резултат). Можете да избирате между следните настройки:

'**digit**' – само цифри;
'**lower**' – само малки букви;
'**upper**' – само големи букви;
'**punct**' – само специални символи (без интервали);
'**alpha**' – малки и големи букви;
'**alnum**' – цифри, малки и големи букви;
'**graph**' – цифри, малки и големи букви, и специални символи (без интервали);
'**print**' – цифри, малки и големи букви, и специални символи (включително интервали).

Специалните символи в горните примери включват само такива от основната стандартизация **ASCII** – което гарантира ползваемост на генерираната парола от система с базови настройки съгласно **US**-стандартизацията. Трябва обаче да имате предвид, че интервалите в някои случаи могат да създадат усложнения и да направят паролата неползваема – поради което обикновено не се препоръчват и последната предложена настройка остава по-скоро с тестови функции.

Ако желаете да посочите по-прецизна съвкупност от специални символи, които да бъдат включени в търсения псевдослучаен резултат, можете да укажете конкретните желани символи чрез тази команда:

```
$ tr -dc 'Конкретни_символи' < /dev/urandom | \
head -c Брой_символи; echo
```

(където под '**Конкретни_символи**' разбираме конкретните символи, които желаете да бъдат включени в псевдослучайния резултат). Ако желаете да включите цифри, големи и малки латински букви, можете да сторите това чрез въвеждане на низа '**A-Za-z0-9**' в командата (където първият и последният символ от съответния тип представляват цялата последователност от този тип символи). Ако желаете да включите и конкретни специални символи, можете да добавите в края на низа конкретните желани специални символи, като средното тире '**-**' (ако планирате да включите такова) трябва да бъде въведено най-накрая (без да бъдат оставяни каквито и да било интервали или други разделители между символите). Всичките възможни специални символи от основната стандартизация **ASCII** (с изключение на интервала) са тези:

```
`~!@#$%^&*()_=+[]{};:'"\|,./<>?-
```

Трябва да имате предвид, че никоя конвенционална компютърна система не може да генерира истински случайни числа. Базирана на строго предвидими алгоритми, рандомизацията е само условна и случайността е относителна. Това поставя под въпрос степента на рандомизация (а от там и на сигурност) на паролите. Ето защо, ако прилагате софтуерни генератори, се препоръчва да рандомизирате допълнително получените резултати, преди да ги ползвате

като пароли (като например вземете на произволен принцип в разбъркан ред част от символите в един сравнително по-дълъг низ).

2. Особености на поведението по отношение на паролите

Не записвайте паролата си никъде – поне не като „прост текст“. Ако все пак се налага да систематизирате голям брой пароли, можете да си създадете модел за тяхното „погрешно записване“, който е известен само на вас и без вземането му предвид паролата не може да се ползва. Така например част от символите в паролата може да са излишни или да липсват, или да трябва да се заменят с други, по установено от вас (но незаписано никъде) правило. Препоръчително е записът да не стои в електронна среда (или поне не в устройства, които не са шифровани или се свързват към internet). Важно е записът да не може да се асоциира с конкретния ресурс, за който паролата се отнася. В противен случай, ако попадне в нежелани ръце, въпреки „погрешното изписване“ може да послужи при осъществяването на bruteforce – разбиването на паролата би приключило сравнително бързо, ако вашият погрешен запис послужи като изходна матрица за опитването на възможни вероятностни изменения.

За да ограничите риска от съставяне на такава изходна матрица за вероятностни изменения, можете вместо самата парола да въведете кодирана по ваш собствен начин бележка за това как да бъде изведена паролата. Така например бележка от типа на '**И.с.1 + И.с.3 + И.2.к**' може да означава „Името на съседа от **I** етаж + Името на съседа от **III** етаж + Имената на двете кучета“; което би се преобразувало като парола примерно така: '**AsenTodorkaSharoBalkan**' (който пример е добре да комбинирате с допълнителни усложняващи фактори, тъй като и четирите имена са думи от познатите речници). Предимството на този подход е, че паролата я няма записана нито директно, нито „погрешно“ и никаква изходна матрица не може да бъде съставена (освен логическа такава, ако ключът бъде разгадан). Същевременно вие разполагате с бележката, която при необходимост да ви припомни как да съставите паролата (стига да не забравите ключа за разшифроване на бележката). Особено когато става дума за ресурси, чието достъпване не осъществявате всекидневно, трябва да бъдете особено внимателни и находчиви спрямо такъв тип бележки – във времето неизползваните пароли действително се забравят, а свободните технологии за сигурност наистина не предоставят „задни врати“ за тяхното заобикаляне.

Не ползвайте една и съща парола за различните си ресурси, нито за различните нива от сигурността на един отделен ресурс. Възможно е една парола да бъде компрометирана по някакъв начин (независимо от усилията, които полагате за нейното обезпечаване) и това да компрометира всичките останали ресурси и нива от сигурността, ако са защитени със същата парола. Във вашия компютър например се очаква да имате поне следните пароли: за цялостно разшифроване на твърдия диск; за потребителски достъп до операционната система; и за достъп до операционната система с администраторски права (ако не сте избрали да работите с програмата **Sudo**). Отделно трябва да се добавят паролите за вашите допълнително шифровани дялове (ако решите да имате такива) и за вашите шифровъчни ключове (с които ще се занимаваме подробно в Част **VIII**). Тук дори не споменаваме паролите на вашите шифровани външни носители, скрити шифровани пространства, internet-ресурси (електронна поща, моментна комуникация, web-сайтове) и т.н. Желателно е всичките тези пароли да са достатъчно дълги и сложни, и да са напълно различни, за да може всяка от тях самостоятелно да защитава съответния елемент от вашата информационна сигурност.

Променяйте от време на време вашите пароли. Така, ако някоя стара парола е била компрометирана без да разберете за това, след рутинната промяна съответният ресурс отново ще бъде защитен. (Това обаче не се отнася за цялостното шифроване на вашата система, за достъпа до системата като администратор и за вашите шифровъчни ключове! – след малко ще се спрем по-подробно на този въпрос.) Променяйте паролите незабавно и в случай на инциденти, които са направили възможно (дори само теоретично) разкриването изцяло или отчасти на съответните пароли (например поради въвеждането им на погрешно място или поради установяване, че сте били под наблюдение докато въвеждате). Дори и в действителност инцидентът да не е довел до разкриване, рутинното сменяне на застрашената парола няма да навреди на вашата информационна сигурност. Обратното обаче (ако проявите невнимание в случай на разкриване) е сигурен провал на всичките ви усилия.

При промяна на голям брой сложни пароли възниква въпросът за тяхното запомняне. Не ползвайте софтуер за запомняне на пароли (защото с това всичките ви пароли биха се поставили в зависимост от сигурността на този софтуер). Добра практика е например да си създадете определено правило, съгласно което се променят всичките ви

пароли по общ, но непредвидим начин. Така например, ако към всичките си пароли добавите някаква представка, окончание или друго – примерно окончание '**123aaa**', можете след време да го смените едновременно за всичките си пароли и да запомните лесно тази обща промяна. Така, ако добавите окончание '**123aaa**' в приложения пример '**АзОбичамБира**' (видоизменен като '**A306i4am6ira**' или като '**DpF/r'd;?r,d**'), ще получите парола '**A306i4am6ira123aaa**' или '**DpF/r'd;?r,d123aaa**' – пароли с относително висока сложност и надеждност. На следващ етап можете да решите да замените окончанието например с представка '**5a5a5a**' – което ще промени всичките ви пароли така: '**5a5a5aA306i4am6ira**' или '**5a5a5aDpF/r'd;?r,d**'. Подобен подход съчетава сложността на паролите с възможността да запомните лесно общия модел, по който сте ги променяли във времето.

Но все пак трябва да имате предвид, че общият модел на променяне е сериозно улеснение за специалистите по разбиване на пароли. Затова е препоръчително да се опитате да разработите свои „динамични“ модели на променяне – при които конкретният начин, по който ще се промени определена парола, се явява производна от някакъв трудно предвидим фактор (например броя символи в самата парола, азбучния ред на някои от ползваните символи или друго, което ви е лесно да запомните и приложите, и което ще направи трансформацията различна, зависима от трудно установим фактор). Можете например да определите всяка от първите три букви във вашата парола да се преобразува в следващата буква от азбуката, а последната цифра да се намали с единица (в примерите '**A306i4am6ira**' и '**DpF/r'd;?r,d**' това би довело като резултат '**B306j4bm5ira**' и '**EqG/r'd;?r,d**'); или например да обърнете горния и долния индекс наобратно (в примерите '**A306i4am6ira**' и '**DpF/r'd;?r,d**' това би довело като резултат '**a#)^I$AM^IRA**' и '**dPf.R"D:/R<D**'). Общото в предложените примери е това, че „динамичният“ модел на променяне се проявява по различен начин в отделните пароли (в зависимост от тяхната текстура) и това прави много трудно проследяването и разкриването на този модел. Не се колебайте да приложите казаното дотук по начин, удобен за вас и съобразен с вашите предпочитания!

Запомнянето на голям брой пароли с необходимата дължина и сложност е трудна задача особено при системното администриране, където се налага да бъдат управлявани множество компютърни устройства и акаунти. Това мотивира някои потребители да се обърнат към услугите на софтуерни мениджъри на пароли. Менджърите на пароли разчитат на главна парола, която да защитава всичките съхранявани пароли; така потребителят вместо да помни всяка от отделните пароли, е достатъчно да помни и въвежда главната парола, за да укаже не мениджъра да приложи необходимата парола за съответния защитаван ресурс. Недостатък на всички видове софтуер за съхраняване на пароли обаче е това, че те разчитат на главна парола, чието разбиване (или евентуални слабости в самия софтуер) може да доведе до компрометиране на всичките защитавани пароли. Нещо повече – освен самите пароли, мениджърът обикновено съхранява и данни за защитаваните с тези пароли ресурси (потребителски имена, идентификатори на хардуерните устройства и т.н.). Така, ако бъде получен неразрешен достъп до съхраняваните пароли, заедно с това може да бъде получен достъп и до указания за това къде да бъдат въведени паролите. Ако въпреки тези рискове решите да ползвате подобен софтуер, се препоръчва да не въвеждате разпознаваеми данни за защитаваните ресурси, както и да оставяте в паролите някакви специфични грешки, които да затруднят ползването им в случай на пробив.

Не предоставяйте вашите пароли на другиго. Никога. Едни от най-резултатните атаки срещу информационната сигурност включват т.нар. „социално инженерство“, при което не се прилагат толкова технически способи, колкото социални; създават се предпоставки и изкуствено контролирани ситуации, при които да изглежда „оправдано“ и „необходимо“ да направите „едно изключение“. Интересен факт е, че именно чрез прилагането на такива способи са осъществявани някои от най-дръзките пробиви в сигурността на изключително секретни и защитени информационни масиви – например чрез неочаквано обаждане от „новия“ системен администратор, който цитира убедително типичните „вътрешни“ жаргонни изрази и заявява, че се нуждае от „извънреден“ достъп за отстраняването на някаква „неочаквана повреда“. Понякога отказът да предоставите вашата парола може „да затрудни“ текущата работа или да лиши някого от услуга, която „не е прилично“ да бъде отказана. Никога обаче не подценявайте възможността предоставеният „извънреден“ достъп да послужи и за други цели; вашият довереник да бъде принуден на свой ред да предостави паролата; или тя просто да бъде прихваната от трети страни, докато я съобщавате.

И не на последно място – разкриването на една ваша парола (освен всичко друго) предоставя конкретен пример за това как съставяте паролите си. Не е изключено такава информация да се превърне в изходна матрица за прилагането на вероятностни изменения с цел разбиването и на други ваши пароли; или за съставянето на речници с логически предполагаеми варианти на паролите, които бихте могли да сте измислили (какъвто беше например случаят с паролите '**Moskva1**', '**Moskva2**', '**Moskva3**' и т.н.).

3. Променяне на паролите чрез командния ред

Терминалът ви позволява да сменяте всички пароли във вашата система от командния ред – чрез тази команда:

```
$ passwd
```

Системата ще влезе в режим: '**Changing password for Потребител. Current password:**' и от вас се очаква да въдете настоящата парола на текущия потребител, а след това – новата парола '**New password:**' и още веднъж новата парола '**Retype new password:**' (с цел да се избегне евентуална грешка при въвеждането и от там – компрометиране на съответния акаунт). Както по-горе вече стана дума, от съображения за сигурност при въвеждане на пароли в терминала не се извежда индикатор за броя въведени символи (например '****************' или '**••••••••••••**'). Независимо от това – когато въведете правилно текущата парола и след това въведете два пъти една и съща нова парола, исканата от вас промяна ще бъде извършена. При това внимавайте за големи и малки символи, както и за езиковите настройки на клавиатурата към момента на въвеждането.

Ако желаете да смените паролата си като администратор, трябва първо да влезете в администраторския акаунт с командата '**su**' и след това да въведете командата за сменяне на пароли '**passwd**' – при което системата директно ще влезе в режим на очакване да въведете новата парола – '**New password:**'. Докато работите като администратор, имате право да промените паролата на всеки потребител без администраторски права:

```
# passwd Потребител
```

Отново системата ще влезе директно в режим на очакване да въведете новата парола – '**New password:**' – тъй като администраторските права ви позволяват да администрирате всеки от останалите потребители, без да е необходимо за тази цел да знаете техните пароли.

В част **VIII** ще разгледаме подробно работата с 'публични' и 'частни' шифровъчни ключове, като тук е достатъчно да отбележим, че те се ползват за осъществяване на т.нар. „асиметрично“ шифроване, при което с публично известния 'публичен' ключ (**Public key**) всеки желаещ може да шифрова съдържание или да проверява цифрови подписи, а с еднозначно асоциирания с него и известен единствено на притежателя му 'частен' ключ (**Secret key**) може да се разшифрова съдържанието и да се „полагат“ цифрови подписи (по-горе направихме няколко такива криптографски проверки за потвърждаване автентичността и интегритета на различни интересуващи ни файлове). На този етап трябва да знаем, че 'частните' ключове (чрез които се разшифрова шифрованото с 'публичните' ключове съдържание и се „полагат“ цифрови подписи) е препоръчително да се съхраняват в системата в шифрован вид – тогава е необходима парола, с която се разшифроват в момента на ползването им. Паролата на един 'частен' ключ можете да промените така:

```
$ gpg --edit-key Ключ
```

(където под '**Ключ**' разбираме избрания от вас 'частен' ключ или каква да е достатъчно разпознаваема част от информацията, с която ключът се асоциира). В случай, че интересуващият ви 'частен' ключ е въведен във вашата система, тя ще изведе резултат '**Secret key is available.**' и ще влезе в режим на очакване да уточните кой параметър желаете да промените. В случая ви интересува параметърът '**password**', след указването на който ще ви бъде дадена възможност да въведете текущата парола, а след това – двукратно да въведете и новата парола за разшифроване на този 'частен' ключ.

В Част **IV** засегнахме въпроса за цялостно шифроване на вашата система (което е от съществено значение както за обезпечаване сигурността на самата система, така и на поверителната информация, обработвана в нея). След като вече сте извършили цялостно шифроване на системата чрез програмата **Cryptsetup**, терминалът ви позволява да променяте и паролите за достъп до цялостно шифровани твърд диск (или до обособени допълнително шифровани дялове от него (например допълнително шифровани '**lock**' дял), ако сте създали такива).

Можете да смените паролата на изцяло шифровано устройство (било то целия твърд диск или обособен дял от него) така:

```
# cryptsetup luksChangeKey /dev/Устройство
```

(където под '**Устройство**' разбираме наименованието, под което системата разпознава съответното устройство. Ако не сте сигурни, можете да направите проверка с проследената по-горе команда '**lsblk**').

Трябва да внимавате много при промяна на пароли с **Cryptsetup**, тъй като евентуално изгубване на правилната парола (например грешно въведен нова парола, която е потвърдена грешно) означава край на достъпа ви до системата (освен ако не сте извлекли Основния ключ на устройството или не сте направили резервно копие на Хедъра); няма да остава нищо друго, освен да форматирате цялостно шифрования твърд диск (или обособения шифрован дял от него), губейки цялата информация, която се е намирала там. За да бъде ограничен този риск, **Cryptsetup** дава възможност вместо да променяте пароли, да добавяте нови и да премахвате стари такива. Така можете да добавите нова парола, да се убедите, че всичко с нея е наред и едва след това да премахнете старата. Можете да добавите нова парола така:

```
# cryptsetup luksAddKey /dev/Устройство
```

Можете да премахнете стара парола така:

```
# cryptsetup luksRemoveKey /dev/Устройство
```

Имайте предвид, че **Cryptsetup** премахва посочената от вас стара парола, независимо дали преди това има въведена друга парола. Внимавайте да не се окаже, че за достъпвания от вас шифрован ресурс няма оставена нито една валидна парола!

4. Критичния момент при самото въвеждане на паролата

Бъдете особено внимателни, когато въвеждате своите пароли в компютъра. Убедете се, че сте на правилното компютърно устройство (не на подменено с друго от същия вид) и то не е компрометирано (проникване в корпуса на хардуера, подмяна на клавиатурата и други). Убедете се, че мястото, където се намирате, не се наблюдава от трети лица (включително от прозорец, огледало или подобни) или от видеокамери (които могат да бъдат скрити, включително монтирани в дома или на местоработата ви, докато сте отсъствали съвсем за кратко).

Имайте предвид, че установяването на приблизителната дължина на вашата парола е ключов фактор за нейното разбиване – бъдете особено внимателни със системи, чийто интерфейс дава индикатор за броя въведени символи (например '****************' или '**••••••••••••**').

Практиката познава сценарии с прилагането на термовизионни камери, които заснемат температурата на клавишите и по степента на „затопленост“ установяват каква е била поредността на докосването им. Не е излишно да прикриете клавиатурата, докато въвеждате паролата (например като свикнете да въвеждате паролата само с „напипване“ на клавишите и притворите капака на преносимия компютър ниско над клавиатурата и ръцете си). Препоръчва се да правите това рутинно, дори когато сте сами и „сте сигурни“, че нищо не ви заплашва; или когато някой може да се обиди от вашата „прекалена предпазливост“. Обезпечаването на паролите е базов момент в информационната сигурност и трябва да се съблюдава „по принцип“, а не заради съществуването на конкретна, разпозната от вас заплаха; защото никой от евентуалните ви противници не би ви уведомил за своите намерения. Не очаквайте да знаете, за да започнете тепърва да вземате мерки!

Препоръчва се да се оттеглите на сигурно място (където е трудно да бъдете проследени и е трудно да се предположи, че бихте отишли, за да може някой да постави наблюдение) и там да оттренирате на спокойствие многократно въвеждане на вашите пароли – на клавиатурата, с която обичайно работите. Целта е да придобиете необходимата бързина, автоматизъм и безпогрешност при въвеждането. По-бързите ви движения ще затруднят евентуално проследяване процеса на въвеждане, а точността ви ще намали случаите, в които се налага да дадете „втори шанс“ за

прихващане поради допусната грешка и необходимост от повторно въвеждане на същата парола. В тази връзка се препоръчва непосредствено преди въвеждането на определена важна парола (особено ако досега не сте набирали от същата клавиатура) да поработите малко с клавиатурата, включително като въвеждате някакви ненужни символи, фрази, изречения. Докато правите това, ръцете ви ще се настроят за бърза и прецизна работа – което е особено полезно за потребители без добре развит машинописен навик.

Препоръчва се също така при въвеждането на определена важна парола да „размиете“ момента на самото въвеждане на паролата с въвеждането на ненужни символи, фрази, изречения (включително като ползвате за тази цел полето за въвеждане на самата парола). Докато правите това и се настройвате към клавиатурата, ще можете да съберете мислите си, да огледате още веднъж компютъра и заобикалящата ви среда, и да помислите за последен път дали всичко е наред. Когато сте готови, просто задръжте за малко **[Ctrl]** и натиснете няколко пъти **[Backspace]** (за да е сигурно, че всички ненужни символи са премахнати); и без да правите пауза или друго, от което да си личи някаква промяна, въведете в крайна сметка самата парола и натиснете **[Enter]**. След като сторите това, пак без да правите пауза или друго, от което да си личи някаква промяна, продължете да въвеждате ненужни символи (този път обаче без да натискате **[Enter]**). Подобни действия ще направят трудно установим точния момент на въвеждане на паролата, както и приблизителната ѝ дължина – защото няма да бъде лесно да се отделят предшестващите и последващите натискания на клавиши. За да постигнете това, внимавайте да не допускате две често срещаните грешки при натискането на **[Backspace]** и **[Enter]**: потребителите често влагат малко по-голяма сила в тяхното натискане или оставят малко по-големи паузи преди и/ли след натискането им. Слейте всички натискания на клавиши в едно неразличимо „цяло“.

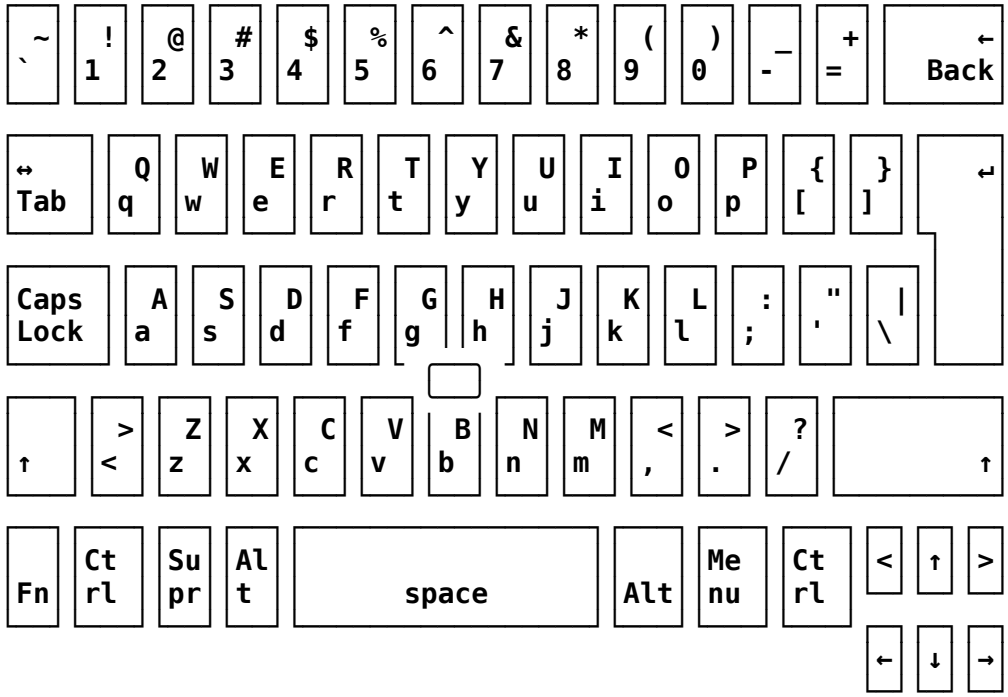
5. Проблеми, които могат да възникнат от клавиатурата

Клавиатурата има ключова роля при въвеждането на пароли. Ето защо е от съществено значение нейното правилно функциониране – от една страна за да позволи въвеждането точно на необходимата ви символна комбинация и от друга страна – за да не „предаде“ тази символна комбинация към трети страни. Последното е възможно да се случи например поради монтирането на софтуерно или хардуерно средство (**Key Logger**) за прихващане на въвежданите от клавиатурата символи. Ако се усъмните в настъпването на такъв риск, не трябва да въвеждате никакви пароли от компрометираната клавиатура. Вместо това можете да закачите външна клавиатура от **USB**-портовете или (ако разполагате с графична среда) да изведете виртуална клавиатура, чиито бутони можете да „натискате“ чрез курсора на мишката.

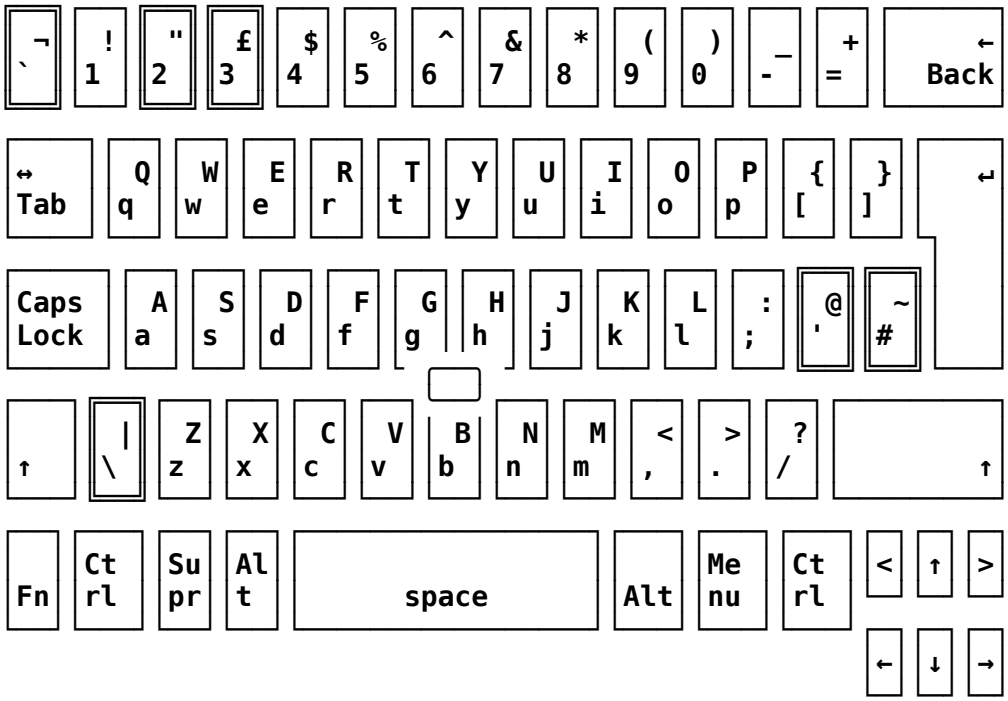
Възможно е в някои случаи да настъпят и по-безобидни проблеми, които обаче препятстват въвеждането на определена парола. Такъв проблем е например софтуерното объркване на клавиатурната подредба или хардуерното повреждане на някой от необходимите ви клавиши.

Поради определена софтуерна грешка (или дори поради неволно въведени настройки) може да попаднете на клавиатурна стандартизация, която е различна от очакваната. Някои езикови стандартизации (например американската и британската) не се различават много и на пръв поглед човек може да си помисли, че са идентични. Ако обаче във вашата парола имате например специалния символ '**@**', при американската стандартизация ще го намерите горе вляво, докато при британската той е долу вдясно; съответно, ако сте попаднали на противоположната клавиатурна подредба, е възможно да въвеждате коректно вашата парола, но въпреки това да получавате отрицателни криптографски отговори. В тази връзка бърз преглед на някои от най-често срещаните клавиатурни подредби би могъл да бъде полезен (целенасочено сме отбелязали клавишите при **x200** (и съответно **x200s**), които съответстват на символи, различни от тези при американската стандартизация):

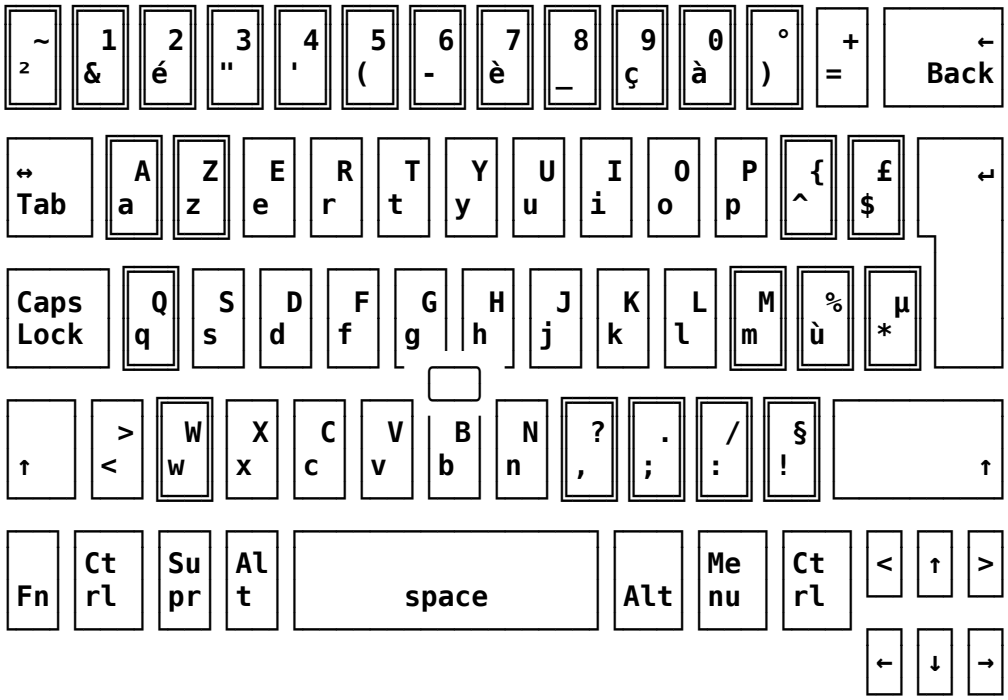
American English (US) QWERTY



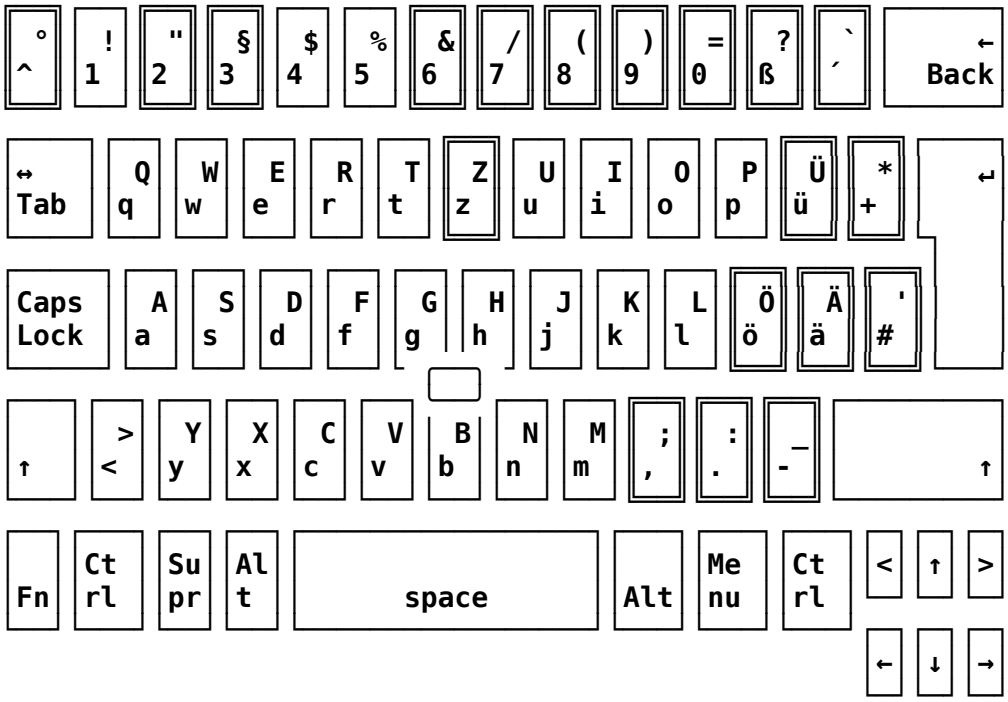
British English (UK) QWERTY



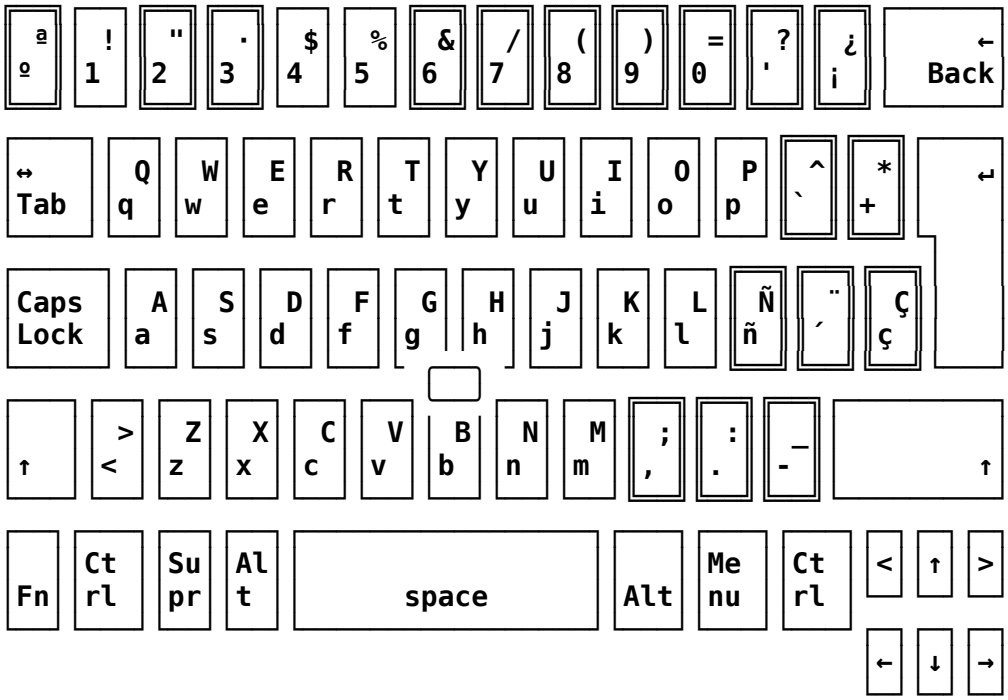
Français (FR) AZERTY



Deutsch (DE) QWERTZ



Español (ES) QWERTY



Някои от най-разпространените клавиатурни подредби

Отрицателни криптографски отговори можете да получите при коректно въвеждане на вашата парола и в случай на хардуерно повреждане на някой от клавишите, необходими за въвеждане на паролата. Възможно е повредата да се отнася само за един от индексите (примерно повредата да обхваща горния индекс на '**A**', докато долният индекс на '**a**' да работи изправно); при което паролата от горния пример '**АзОбичамБира**', макар да е въведена коректно, да се отразява като '**зОбичамБира**'.

В случай, че сте придобили необходимия автоматизъм и прецизност при въвеждане и получите **2** до **3** последователни отрицателни криптографски отговора при опит да въведете някоя добре оттренирана парола, е желателно да преустановите по-нататъшните опити да въведете паролата и да проверите дали клавиатурата работи изправно. Влезете в обикновен текстови режим (където въвежданите символи се показват на екрана в открит вид) и въвеждайте известно време някакви произволни символи или фрагменти от текст. Докато въвеждате, от време на време вграждайте и някои от символите на вашата парола, за да видите дали те се отразяват по очаквания начин. Между тях въвеждайте достатъчно други символи, за да не може да се направи извод за вашата парола. Така ще се убедите в коректността на клавиатурата и ако всичко е наред, ще можете да направите повторен опит за въвеждане на паролата малко по-късно. Ако обаче установите софтуерен или хардуерен проблем, ще ограничите риска от безсмислено многократно въвеждане на вашата парола и ще се насочите към отстраняване на проблема.

6. Неотложни действия при компрометиране на паролите

Ако въпреки взетите от вас мерки паролите ви бъдат компрометирани по някакъв начин (или дори само ако възникне съмнение, че такова компрометиране може да е настъпило), трябва незабавно да ги подмените. В такъв случай не ви съветваме да се задоволявате само с добавянето на представки, окончания и други частични редакции – необходимо е да направите радикална промяна. Имайте предвид, че допуснатото „изтичане“ на пароли може да е засегнало вашата информационна сигурност по начин, който се простира далеч отвъд самите пароли.

Ако е разкрита паролата за цялостно шифроване на твърдия диск (**или на обособен дял от него**) – това може да е довело освен до преглеждане съдържанието на съответното устройство и дори до сваляне на копия от критични за информационната сигурност файлове (Основния ключ, 'частните' ключове и други). Няма никакво значение колко пъти след това ще променяте паролите – който разполага с Основния ключ, може да разшифрова системата и без парола. А който разполага с вашите 'частни' ключове (макар и в техния шифрован вид), няма пречка да вложи време и ресурси за тяхното bruteforce разшифроване. В такава ситуация е задължително да осъществите ново шифроване на засегнатото устройство. Ако на него е била инсталирана операционна система, е необходимо да я инсталирате наново. А ако там са съхранявани 'частни' ключове, е необходимо да ги анулирате (в Част **VIII** ще се спрем специално на този въпрос) и да известите вашите доверници. Ако имате шифровано с тези ключове поверително съдържание, е необходимо да го шифровате с нови ключове и да заличите старите версии.

Ако е разкрита паролата за достъп до системата с администраторски права – това може да е довело до инсталиране на програми и въвеждане на настройки, несъвместими с интегритета и сигурността на системата, а така също – до извличане на критични за сигурността системни файлове (като например вашите 'частни' ключове, макар и в техния шифрован вид). В такава ситуация е задължително да осъществите ново шифроване на засегнатото устройство и да преинсталирате операционната система. Съхраняваните в компрометираната операционна система 'частни' ключове трябва да анулирате. Ако не сте блокирали възможностите за препрограмиране на flash-чипа (на този въпрос ще се спрем в Част **VII** по-долу) или ако не можете да потвърдите, че няма физическо проникване в корпуса на компютъра, трябва също да препрограмирате софтуера от „ниско“ ниво. Прочее, евентуално физическо проникване в корпуса на компютъра е основание да подмените изцяло вашето устройството с ново.

Ако е разкрита паролата за достъп до системата без администраторски права – това може да е довело до преглеждане, копиране и/или подменяне на незащитено (нешифровано) съдържание, а така също – на съдържание, което макар да е защитено (шифровано), към момента на пробива е било налично в разшифрован вид. В случай, че работите с програмата **Sudo**, трябва да приравните пробива към разкриване на паролата за достъп с администраторски права. Същото се отнася и в случай, че към момента на пробива е имало (или е могло да има) отворен терминал с инициирана и непрекратена сесия с администраторски достъп (независимо от това, че самата парола за администраторски достъп не е разкрита).

Ако е разкрита паролата за някой от вашите 'частни' ключове – това може да е довело (ако същевременно е била достъпена системата, където този ключ се съхранява) до извличане на самия частен ключ и до последващото му ползване за разшифроване на поверително съдържание и за „полагане“ на цифрови подписи от ваше име. В такава ситуация трябва задължително да смените паролата на засегнатия 'частен' ключ, а ако е осъществен (или е могло да е осъществен) достъп до системата – е необходимо незабавно да анулирате този ключ. Следва да имате предвид, че дори анулиран, ключът ще продължи да е в състояние да разшифрова поверително съдържание, шифровано преди това с еднозначно свързания с него 'публичен' ключ.

При всички случаи – когато установите пробив във вашата информационна сигурност, е важно да действате бързо, по предварително изготвен план и без паника – като на първо място ограничите всякакъв достъп до вашите системи и информация (физическо местоположение и internet-свързаност), направите подмяна на компрометираните пароли и ключове, и преинсталирате компрометирания софтуер (на нов хардуер, ако има основание да се съмнявате в сигурността и на хардуера, ползван до този момент). В такава ситуация дори и част от действията ви да се окажат излишни, това е по-добрият вариант, отколкото да оставите „изтичането“ неотразено и това да доведе до още по-дълбоко проникване във вашата поверителна информация.

VI. ЗАЩИТАВАНЕ НА СВЪРЗВАНЕТО С INTERNET

Обезпечаването на вашата система чрез нейното цялостно шифроване, препрограмиране на flash-чипа със свободен софтуер „от ниско ниво“ и инсталирането на напълно Свободна операционна система е предпоставка за постигането на висока степен на информационна сигурност, но само докато системата действа и информацията се обработва offline, без да се свързвате към internet. От момента, в който системата се закачи към internet, се превръща в потенциален обект на неизброими по своя вид и характер атаки, осъществявани online. В действителност почти всичките пробиви в електронната информационна сигурност се осъществяват именно чрез атакуване на набелязаните системи от internet.

Добрата новина е, че вашата система (ако сте последвали напътствията от предходните части в изложението) поддържа много високи нива на сигурност и ще бъде сериозно предизвикателство в случай на опит да се осъществи пробив. Въпреки това е от съществена важност да спазвате някои прости правила при работа в internet, за да не попаднете в обхвата на значително по-непретенциозни атаки от нивото, с което сте защитили системата си.

Специално внимание трябва да обърнете на вашата online-комуникация – включително по отношение на начина, по който изпращате поверителна информация и очаквате да получите такава. Колкото и добре да е шифрована информацията, ако позволите тя да бъде прихващана и анализирана систематично, е възможно във времето ползваната от вас шифровъчна технология все пак да бъде разбита и всичките ви усилия да останат напразни.

В някои случаи дори само установяването на факта, че осъществявате комуникация (при това шифрована), може да се превърне в проблем (особено в условията на деспотични политически режими или рестриктивна корпоративна политика). Ето защо – макар да е отделен предмет – ще трябва да засегнем и въпроса за „скриване“ на осъществяваната комуникация в internet чрез различни способи за нейното привидно обезличаване и анонимизиране; привидно, защото пълно обезличаване и анонимизиране в действителност няма как да се гарантира.

1. За фалшивите адреси и подправените internet-ресурси

Преди всичко, докато работите в internet, трябва да съблюдавате някои съвсем основни правила, валидни за всеки компютърен потребител. Възможно е някой да се опита да ви убеди, че се свързвате с web-ресурс, с който всъщност не сте се свързали или да ви накара да мислите, че получавате съобщения от трети страни, от които всъщност не сте получили нищо (или сте получили нещо съвсем друго, което обаче е прихванато и не е достигнало до вас; или поне не е достигнало в първоначалния му вид). По тези вектори протичат повечето атаки, насочени срещу човешкия фактор (вместо срещу технологичното обезпечение, което ползвате и което от техническа гледна точка може да е доста надеждно и трудно за компрометиране, ако бива управлявано правилно).

Ако влизате в web-базирани ресурси, където се очаква да въвеждате пароли или да качвате поверително съдържание, вижте дали оперирате през защитена (**Secured**) връзка (**`https://`**) и се убедете, че се намирате наистина на web-адреса, на който очаквате да сте (като внимавате адресът да съвпада точно, буква по буква, с очаквания от вас; например '**`https://googl1e.com/`**' не е същото като '**`https://google.com/`**'). Внимавайте дали адресите, с които обменяте съобщения, са точно тези, които очаквате да бъдат (например '**`advocati@qmx.com`**' не е същото като '**`advocati@gmx.com`**'). Внимавайте и за такива измами, при които очакваният от вас web-адрес е изписан коректно, но в действителност се явява само част от действителното място, към което ви насочват (като например '**`google.com.login.fraud.site/settings`**', където (независимо от коректното начало '**`google.com`**') действителният адрес е '**`fraud.site`**').

Възможно е някои символи от една азбука да бъдат подменени с графично идентични или подобни символи от друга азбука. Например латинските '**a**', '**c**', '**e**', '**o**', '**p**', '**x**' и '**y**' са графично идентични със съответни букви от кирилицата, а когато става дума за главни букви, към това трябва да добавим '**B**', '**H**', '**K**', '**M**' и '**T**'; при някои курсивни шрифтове идентични (или подобни) могат да се окажат и символи като латинските '**n**' и '**m**' с кирилските '**п**' и '**т**'. Също гръцките '**A**', '**B**', '**E**', '**Z**', '**H**', '**I**', '**K**', '**M**', '**N**', '**O**', '**o**', '**P**', '**T**', '**Y**' и '**X**' са графично идентични със съответни букви от латиницата, а в зависимост от шрифтовете – графично идентични или подобни на латинските могат да се окажат и гръцки букви като '**α**', '**γ**', '**η**', '**ι**', '**ν**', '**ρ**', '**τ**', '**υ**' и '**χ**'. Графично идентични на букви от кирилицата са гръцките '**A**', '**B**', '**Γ**', '**E**', '**H**', '**K**', '**κ**', '**M**', '**O**', '**o**', '**П**', '**P**', '**T**' и '**X**', а в зависимост от шрифтовете – графично идентични или подобни могат да се окажат и гръцки букви като '**α**', '**γ**', '**Λ**', '**ρ**', '**Φ**' и '**χ**'. За да ограничите риска от измами с графично идентични или подобни символи, се препоръчва да въвеждате „ръчно“ интересуващите ви адреси от клавиатурата – вместо да активирате предоставени препратки или да копирате от „услужливо“ предоставени подсказвания.

Бъдете особено внимателни с такива съобщения, в които има препратки или прикачени файлове и се очаква да ги последвате / отворите. Не бързайте да изпълните каквито и да било „делови“ указания за въвеждане на ваша поверителна информация, а още по-малко – за въвеждане на пароли или за стартиране на софтуер, за който не сте сигурни откъде идва и какво извършва. Свободните операционни системи се отличават с много висока сигурност, но в някои случаи продължават да бъдат уязвими на заплахи, специално подготвени за компрометирането на системи от този вид или дори на конкретния потребител (с всичките налични данни за неговите компютърни навици и поведение).

Препоръчително е във връзка с горното да работите с два отделни компютъра – един за „всекидневни“ цели и още един за обработка на вашата поверителна информация (който в най-добрия случай изобщо не се свързва с internet). Ако обаче нямате възможност да осигурите две отделни устройства и все пак ви се налага да отворите „несигурен“ файл (или да последвате „несигурна“ препратка), можете да сторите това от случаен компютър (например в някое internet-кафене или обществена библиотека). По този начин ще запазите неприкосновеността на вашата система, дори и „несигурният“ файл или препратка да е част от атака срещу информационната сигурност.

2. За опасностите на програмни езици като JavaScript

Независимо колко и какви устройства ползвате, във всички случаи препоръчваме да изключите всякакви възможности за изпълнение на **JavaScript** във вашата система; особено ако обработвате и поверителна информация на нея. **JavaScript** е програмен код, който може да се зареди в системата ви от посещаваните от вас web-сайтове и да се изпълни във вашето устройство – с цел да направи съответните internet-ресурси по-интерактивни (падащи менюта, аудио-визуални ефекти, автоматично обновяване с ново съдържание и т.н.). В някои случаи обаче чрез подобен програмен код могат да бъдат осъществени атаки срещу системата. В случай, че ползвате браузъра **Firefox** (или аналогичните **Icecat** и **Iceweasel** при **GNU/Linux-libre**) можете да инсталирате добавката **NoScript**, достъпна от този адрес:

`https://addons.mozilla.org/firefox/addon/noscript`

NoScript блокира всички опити да бъде зареден програмен **JavaScript** код и да го изпълни на вашия компютър – освен ако съответният web-сайт не е включен във вашия „бял списък“ (**White List**) със сигурни (**Trusted**) internet-ресурси, на които сте избрали да се доверявате. При това положение голяма част от съвременните web-сайтове просто няма да работят при вас или ще работят с ограничена функционалност (тъй като те разчитат в много голяма степен на **JavaScript** технологии). Тук е мястото да решите кое е по-приоритетно за вас – доброто графично и интерактивно представяне на интересуващите ви internet-ресурси или вашата информационна сигурност. В случай, че все пак решите да позволите зареждането и изпълняването на **JavaScript** код, трябва да прецените на кои от интересуващите ви internet-ресурси ще продължите да се доверявате и кои ще откажете да ползвате.

При всички случаи не се препоръчва да допускате зареждането на какъвто и да било **JavaScript** код на устройствата, с които обработвате поверителна информация. В противен случай отваряте потенциален вектор за атаки и компрометиране на поверителната информация. Както вече стана дума в тази връзка, препоръчва се да обособите едно устройство за „всекидневни“ цели и за свързване с всякакви ресурси в internet, и още едно за обработване на поверителна информация, което поначало не се свързва с internet и на него не се изпълнява никакъв **JavaScript**.

3. За ползването на „защитни стени“

В съвременни условия е невъзможно да ползвате компютър за комуникация, ако той не се свързва към internet. От момента на неговото свързване обаче той се излага на редица рискове, насочени общо към такъв тип системи или дори специално към конкретния компютър. Ето защо прилагането на настройки, които действат като „защитна стена“, е съществено за информационната сигурност – особено ако не разполагате с две отделни устройства, едното от които да ползвате за комуникация и подобни „всекидневни“ дейности, а другото – само за обработване, съхраняване и (раз)шифроване на поверителна информация, без изобщо да го свързвате с internet.

Различаваме най-общо три вида заявки, които образуват трафика между вашето устройство и външния свят: инициирани от вашето устройство изходящи (**Output**) заявки; постъпващи отвън входящи (**Input**) заявки; и такива, които постъпват отвън, но се препращат (**Forward**) към други устройства (например ако вашият компютър действа като рутер по отношение на други компютри от вътрешна мрежа). Вашата система не се предвижда да служи като сървър (или като рутер) и не следва да допуска **Forward**-заявки. Между постъпващите отвън входящи заявки различаваме такива, които са в отговор на инициирана от вашето устройство изходяща заявка; и такива, които са инициирани от другаде (не са в отговор на нещо, инициирано от вас). Съществуват много софтуерни решения, които изпълняват функцията на „защитна стена“ (**Firewall**), като проследяват трафика между вашето устройство и външния свят, и блокират тези от заявките, които е вероятно да представляват заплаха. Заплаха е например необичайният опит за осъществяване на отдалечен достъп до вашата система.

Кои заявки ще се третират като заплахи, се определя от въведените в съответната система правила (**Rules**). За да въведете такива правила при **GNU/Linux** може да използвате командния ред, който позволява да достъпите директно настройките за управляване на трафика на ниво софтуерен код в ядрото на операционната система. За тази цел служат т.нар. **iptables** – регистри с вашите правила за това какви заявки е позволено (**Accept**) и какви не е позволено (**Drop**) да бъдат изпълнявани. Управлението на **iptables** обаче е комплексна задача, изискваща задълбочени познания в областта на комуникационните технологии и новопоявяващите се заплахи – което далеч надхвърля задачите на настоящото ръководство. Поради тази причина препоръчваме да приложите някоя от утвърдените **Firewall** програми, каквато е например **UFW**. Предимството на подобни програми е, че съдържат набор от правила за достъп, които могат да бъдат настроени по начин, ограничаващ повечето опити за свързване с вашата система и от там – ограничаващ повечето рискове, идващи отвън. Можете да инсталирате програмата:

```
# pacman -S ufw
```

Следва да настройте защитната стена да стартира при стартиране на системата:

```
# systemctl enable --now ufw
```

Можете да въведете правило за блокиране по подразбиране на всички заявки, идващи отвън, които не са предхождани от инициирана от вас заявка:

```
# ufw default deny
```

Следва да активирате защитната стена (което същевременно ще приложи горното правило за блокиране на всичко):

```
# ufw enable
```

В случай, че желаете времено да спрете защитната стена и да приемате всякакви заявки идващи отвън (например докато тествате някаква функционалност, заради която се нуждаете от подобен достъп), можете да сторите това така:

```
# ufw disable
```

(след което няма пречка отново да активирате защитната среда с по-горната команда '**`ufw enable`**').

Вместо да спрете изцяло прилагането на горното правило за блокиране на всичко, можете да оставите това правило в сила, но да въведете допълнително правило за разрешаването на конкретен вид заявки, които да бъдат изключение в общото забранително правило:

```
# ufw allow Заявки_изключение
```

(където под '**Заявки_изключение**' разбираме съответния вид заявки, които желаете да бъдат изключени от общите забранителни правила). Най-често това се прави за **SSH** протокола за защитен достъп и за **HTTP** протокола за web-достъп.

Можете също така, вместо просто да разрешите интересуващите ви заявки-изключения, да ги разрешите лимитирано – чрез тази команда:

```
# ufw limit Заявки_изключение
```

Горната команда ще ограничи разрешената скорост и брой заявки към вашето устройство (до **6** заявки на всеки **30** sec от конкретен източник). Това ще предотврати евентуални bruteforce атаки (например опити за уцелване на паролата за достъп на потребителски акаунт) или **DDoS** атаки (опити за блокиране на системата чрез претоварване) – което горещо ви препоръчваме, ако изобщо решите да въведете подобни разрешени заявки-изключение.

4. За данните, с които компютърът се представя в internet

Всяко компютърно устройство, което се свързва към определена локална мрежа (например към рутера, от който получавате internet-свързаност), се идентифицира с уникален за съответната локална мрежа **MAC**-адрес, наименование на системата и в някои случаи – допълнителни данни за хардуера и софтуера.

MAC-адресът представлява **6** двуцифрени шестнадесетични числа, разделени от двуеточия, които при **LAN**-устройството на компютри, препрограмирани с **Libreboot**, по подразбиране са винаги '**00:f5:f0:40:71:fe**'. Ако към съответната локална мрежа бъдат включени чрез **LAN**-кабел две препрограмирани с **Libreboot** устройства, чиито **MAC**-адреси са оставени по подразбиране, ще настъпи конфликт, тъй като локалната мрежа не може да се ориентира кои заявки към кое от двете „еднакви“ устройства да насочва. На второ място – въпреки, че **MAC**-адресът е видим само в локалната мрежа (за външните мрежи видим остава само **MAC**-адресът на общия за локалната мрежа рутер) – вашият адрес може да послужи за еднозначното ви идентифициране в рамките на тази локална мрежа (ако се установи, че държите устройството с търсения **MAC**-адрес, записан например в журнала на рутера). Поради тази причина, ако смятате да ползвате кабелно свързване на вашия лаптоп, може да се наложи да смените **MAC**-адрес – включително и за да не издавате, че вашият компютър е препрограмиран с **Libreboot** (голяма рядкост, която може да ви идентифицира еднозначно в определени среди).

Чрез **MAC**-адрес се идентифицират не самите компютри, а устройствата за кабелно свързване с internet и тези за безжично свързване. Следователно във вашия компютър най-вероятно има (поне) **2 MAC**-адреса.

За да смените **MAC**-адреса на устройството за кабелно свързване към internet, е необходим инструментът **ich9gen**, който се съдържа в пакета '**libreboot-utils**', който може да инсталирате така:

```
# pacman -S libreboot-utils
```

След като инсталирате горния пакет, можете да смените **MAC**-адреса на устройството за кабелно свързване към internet:

```
$ ich9gen --macaddress MAC-адрес
```

(където под '**MAC-адрес**' разбираме избраните от вас **6** двуцифрени шестнадесетични числа, разделени с двуеточия (например '**00:1f:16:80:80:80**' или подобни)). Като резултат от изпълнението на командата в текущата директория ще бъдат изведени файлове с вписан в тях желания от вас **MAC**-адрес:

```
ich9fdgbe_16m.bin
ich9fdgbe_16m_ro.bin
ich9fdnogbe_4m.bin
ich9fdnogbe_4m_ro.bin
ich9fdnogbe_8m.bin
ich9fdnogbe_8m_ro.bin
ich9fdnogbe_16m.bin
ich9fdnogbe_16m_ro.bin
```

В случая интерес представлява файлът '**ich9fdgbe**' (който завършва на '**.bin**' (не на '**_ro.bin**') и има обозначен капацитета на вашия flash-чип (например **8MiB**) след долната черта).

Вече можете да въведете новосъздадения **MAC**-адрес в **Libreboot**-софтуерния пакет:

```
$ dd if=ich9fdgbe_8m.bin of=Libreboot_пакет bs=1 count=12k conv=notrunc
```

(където под '**Libreboot_пакет**' разбираме **Libreboot**-файла, от който в Част **III** препрограмирахме **BIOS/UEFI** flash-чипа).

След като новосъздаденият **MAC**-адрес е въведен в **Libreboot**-софтуерния пакет, можете да препрограмирате flash-чипа с командата '**flashrom -p internal -w Libreboot_пакет**', както проследихме по-горе. Сторите ли това, устройството за кабелно свързване на вашия компютър към internet ще започне да се идентифицира с новия MAC-адрес.

Можете да промените също **MAC**-адреса на устройството за безжично свързване към internet – като редактирате системния файл '**NetworkManager.conf**':

```
# nano /etc/NetworkManager/NetworkManager.conf
```

Към заварените редове в този файл:

```
# Configuration file for NetworkManager.
# See "man 5 NetworkManager.conf" for details.
```

... трябва да добавите следните допълнителни редове:

```
[connection]
wifi.cloned-mac-address=MAC-адрес
```

(където под '**MAC-адрес**' разбираме **6**-те двуцифрени шестнадесетични числа, които желаете да се вижда като ваш **MAC**-адрес от локалните мрежи, към които се свързвате – например **00:1f:16:80:80:80** или подобни).

Можете също да укажете **MAC**-адресът на устройството за безжично свързване към internet да бъде винаги различен при всяко следващо свързване – в такъв случай към заварените редове в системния файл '**NetworkManager.conf**' трябва да добавите следните допълнителни редове:

```
[connection]
wifi.cloned-mac-address=random
```

След като промените системния файл '**NetworkManager.conf**', трябва да активирате въведените нови настройки:

```
# systemctl reload NetworkManager
```

Вашето устройство за безжично свързване към internet ще се рестартира и ще приложи настройките. Няма пречка след време да промените настройките, за да съответстват на вашите нужди – включително като изтриете въведените по-рано допълнителни редове, за да преустановите генерирането на произволни **MAC**-адреси при всяко ново свързване.

Можете също така да смените **MAC**-адреса на устройството за безжично свързване към internet и само за конкретна **WiFi**-мрежа, към която вече сте се свързвали:

```
# nmcli con modify Мрежа cloned-mac MAC-адрес
```

... или дори към конкретна **WiFi**-мрежа, към която никога досега не сте се свързвали (но знаете нейните параметри за свързване):

```
# nmcli con add type wifi con-name Мрежа ssid Мрежа wifi-sec.key-mgmt \
wpa-psk wifi-sec.psk Парола cloned-mac MAC-адрес
```

(където под '**Мрежа**' разбираме **WiFi**-мрежата, към която желаете да се свържвате с променен **MAC**-адрес; под '**wpa-psk**' – вида шифровка на мрежата (друг вид шифровка е например '**wep**'); и под '**Парола**' – паролата за достъп на мрежата). Тази настройка не е необходимо да се активира, за да влезе в действие.

Също така можете за въведете различни **MAC**-адреси за различни **WiFi**-мрежи, към които се свързвате. Но ако желаете след време да ги премахнете, трябва да изтриете записа във вашето компютърно устройство за мрежата:

```
# nmcli con delete Мрежа
```

Няма пречка след като изтриете записа, отново да се свържете със съответната мрежа, като въведете съответните данни за свързване – по обичайния начин.

Следва да имате предвид, че **6**-те двуцифрени шестнадесетични числа в един **MAC**-адрес обозначават производителя, марката, модела на мрежовото устройство и неговата индивидуална идентичност. Поради тази причина не е добра идея да въвеждате напълно произволни числа, ако желаете компютърът да изглежда „нормално“ в очите на системните администратори на локалните мрежи, към които се включвате. Препоръчваме ви да се ориентирате към предпочитаните компютърни устройства и да се запознаете с „граматиката“ на **MAC**-адресите, прилагана при тях, и да я спазвате, доколкото е възможно.

Можете да укажете наименованието на вашата система да не бъде споделяно в мрежа, към която вече сте се свързвали (и в системата има въведени необходимите параметри за свързване):

```
# nmcli con modify Мрежа ipv4.dhcp-send-hostname FALSE
```

Ако все още не сте свързвали системата към интересуващата ви мрежа (в системата ги няма въведени необходимите параметри за свързване), можете да укажете наименованието на системата да не бъде споделяно в мрежата, към която планирате да се свържете, като междувременно въведете и самите параметри за свързване:

```
# nmcli con add type wifi con-name Мрежа ssid Мрежа wifi-sec.key-mgmt \
wpa-psk wifi-sec.psk Парола ipv4.dhcp-send-hostname FALSE
```

Можете също така да укажете да бъде споделяно конкретно наименование на системата (различно от това, с което се идентифицира вашата система):

```
# nmcli con modify Мрежа ipv4.dhcp-hostname Наименование
```

... или респективно (ако все още не сте свързвали системата към интересуващата ви мрежа):

```
# nmcli con add type wifi con-name Мрежа ssid Мрежа wifi-sec.key-mgmt \
wpa-psk wifi-sec.psk Парола ipv4.dhcp-hostname Наименование
```

На следващо място – при ползването на internet-ресурси (и особено при посещаването на web-сайтове) обслужващите сървъри изискват редица данни за вашата система – поради техническата необходимост да бъдат ориентирани правилно за начина, по който да предоставят интересуващото ви съдържание, а така също – поради аналитични и разузнавателни причини. Вероятно ще представлява интерес това да се запознаете с данните, които вашият браузър предоставя на сървърите, към които изпращате заявки в internet. За да сторите това, най-напред трябва да инсталирате програмата **Netcat**:

```
# pacman -S netcat
```

След като вече имате инсталирана програмата **Netcat**, можете да прегледате какви данни изпраща вашият браузър към достъпваните от вас internet-ресурси:

```
# nc -lp 8080
```

... като отворете проверявания браузър след въвеждането на горната команда на този адрес:

```
http://localhost:8080/
```

Системата ще изведе подобен резултат:

```
GET / HTTP/1.1
Host: localhost:8080
User-Agent: Mozilla/5.0 (X11; Linux i686; rv:38.0) Gecko/20100101
Firefox/38.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

(където '**HTTP/1.1**' е ползваният от вас комуникационен протокол; '**localhost:8080**' – вашият локален сървър, с който се свързвате; '**Gecko/20100101 Firefox/38.0**' – идентификатор на проверявания браузър; '**X11; Linux i686; rv:38.0**' – идентификатор на вашата операционна система; '**Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/***' – указания към отдалечения сървър за това каква информация се приема от вашия браузър (където '***/***' обозначава готовност браузърът да приеме всякаква подадена информация, а стойностите '**q=0.9**' и '**q=0.8**' обозначават каква информация браузърът би приел с приоритет пред друга); '**Accept-Encoding: gzip, deflate**' – какъв формат трябва да бъде приеманата информация (където '**gzip**' обозначава допустимост информацията да бъде компресирана);

'**Connection: keep-alive**' – указание отдалеченият сървър да не прекратява сесията след изпълнение на постъпилата заявка, тъй като е възможно да последват още заявки от ваша страна);

... или подобен резултат:

```
GET / HTTP/1.1
Host: localhost:8080
User-Agent: Mozilla/5.0 (Windows NT 6.1; rv:56.0) Gecko/20100101
Firefox/56.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
DNT: 1
Connection: keep-alive
Upgrade-Insecure-Requests: 1
```

(където при някои от по-новите версии на браузъра '**Windows NT 6.1; rv:56.0**' е фалшифициран идентификатор на вашата операционна система (с цел анонимизирането ви в internet, където едва **3%** от потребителите ползват **GNU/Linux-libre**); и '**Upgrade-Insecure-Requests: 1**' е указание (при по-новите браузъри) отдалеченият сървър да предостави при възможност '**https://**' защитена връзка).

5. За цялостната несигурност на web-базираните услуги

Една услуга е web-базирана, когато се предоставя от отдалечен сървър и е достъпна за потребителите чрез internet посредством потребителски (най-често графичен) web-интерфейс. Такава услуга няма как да гарантира сигурността на вашата информация. На първо място, вие не можете да знаете какво друго (освен официално заявеното) се случва в отдалечения сървър, докато получавате услугата. Като потребител на услугата вие не разполагате нито с физически достъп до сървъра, нито с правата на системен администратор – следователно трябва да разчитате за обезпечаване на вашата информационна сигурност единствено на обещанията на компанията-собственик (както проследихме в Част **I** по отношение на несвободния софтуер). И на второ място – дори и услугата да е напълно коректна, тя достига до вас чрез internet – следователно е напълно възможно да бъде компрометирана по време на трансфера от отдалечения сървър до вашата система и обратно. Ето защо средствата за обезпечаване на вашата сигурност не могат да бъдат „извън“ вашето полезрение и контрол, и не могат да идват „отвън“, от internet.

Най-добрият подход за обезпечаване на сигурността в електронна среда е да осъществявате шифроване на поверителната информация в надеждно обезпечена система, която изобщо не се свързва с internet – и едва след това да изпращате тази информация (вече шифрована) чрез internet – по възможност от друго устройство (в което изобщо не се осъществява (раз)шифроване). Ако не можете да осигурите отделно устройство за шифроване и съхраняване на информацията, което да стои винаги offline, можете да обособите определен дял от твърдия диск и да го шифровате като отделна част от компютъра, която не се разшифрова никога, освен от „жив“ носител, който не свързвате към internet. Когато се налага да обработвате поверителна информация, стартирайте от „живия“ носител. В някои случаи това може да бъде неудобно, но препятства в значителна степен рисковете от компрометиране на информационната сигурност. Добра практика е да ползвате напълно шифрован външни носители (например **USB**-памет или **SD**-карта), които никога не закачате към системата, когато има повод за съмнения. Имайте предвид и това – връзката с internet (и с каквото и да било друго) е изключена най-сигурно тогава, когато е изключена хардуерно, а не само софтуерно. Колкото и надеждно да е обезпечена вашата система, физическото прекъсване на свързаността си остава най-надеждният способ, измислен до този момент. Тъкмо затова в Част **VII** ще се спрем на въпроса за физическата сигурност.

Напълно противопоказни за вашата информационна сигурност са радващите се на широко приложение в днешно време масови пощи, облачни услуги, социални медии, както и популярните търсещи машини – дори и когато не споделяте „никакви тайни“ в тях.

Облачните услуги (**Cloud Services**) представляват хранилища за съдържание, разположени на отдалечени сървъри, където можете да депозирате свои файлове, да ги преглеждате, редактирате и споделяте с трети страни. Но след като съдържанието ви е достъпно от internet, може да се превърне в обект на постоянни атаки за неразрешено достъпване. Няма никакви гаранции, че собственикът на облачната услуга не ползва вашето съдържание за свои собствени цели (без ваше разрешение и понякога дори откровено във ваша вреда). Дори да е шифровано, вашето съдържание (след като се намира на отдалечения сървър) няма пречка да бъде подлагано неопределено време на bruteforce атаки – до разбиване на паролите за достъп. И накрая – вашият достъп до собственото ви съдържание във всеки един момент може просто да бъде отказан.

Социалните медии (**Social Media**) представляват платформи за мрежова комуникация, формиране на приятелски „кръгове“, споделяне на съдържание, коментари, „харесване“. Дизайнът на подобни платформи предразполага потребителите да въвеждат неусетно впечатляващи обеми от лична информация за своите приятели, отношения, наклонности, предпочитания, интереси, преживявания. На базата на все по-съвършени алгоритми за анализ тази информация се ползва за подробно и задълбочено „картографиране“ на индивидуалните профили и социалните отношения, и позволява моделиране. Съществуват недвусмислени доказателства за пряката връзка между такива платформи и различни разузнавателни служби, които ги ползват за проследяване и контрол на потребителите (например създаването на **Facebook** е подпомогнато пряко от разузнавателните служби на **САЩ**, а специалните служби на Руската федерация преди време просто „иззеха“ контрола над рускоезичната социална медия **ВКонтакте**). Подобни услуги не бива да бъдат ползвани – дори и „да не мислите“ да споделяте поверителна информация в тях, техните алгоритми за анализ на вашите интеракции много бързо биха достигнали до изводи, които може би не искате да бъдат правени от трети страни.

Търсещите машини (**Search Engines**) представляват алгоритми за индексиране на съдържанието в internet и предлагане на препратки към web-сайтове, релевантни на заявките на потребителите. Много скоро след своето възникване започват да индексират и самите потребители – под благовидния претекст, че по този начин се усъвършенства услугата и се предлагат препратки, които по-точно отговарят на търсенето. Това се ползва за маркетингови цели (например насочване на таргетирана реклама), но така също за проучване, прогнозиране и моделиране: на обществото като цяло (масови увлечения и страхове, политически настроения, склонности към протести и размирици) и персонално на отделния потребител (разкриване на личните предпочитания и убеждения, прогнозиране на склонностите към покупка, престъпна дейност и други). Все по-съвършения анализ на подаваните заявки и избора между предложените препратки позволява задълбочени изводи за личността, без да сте въвели дори и един символ поверителна информация. Редица доказателства и примери сочат за злоупотреби с такава информация от различни центрове на интерес, далеч отвъд официалните им цели, функции и правомощия.

Няма как да бъдем сигурни за начина, по който функционира един отдалечен сървър. Потребителският интерфейс и предоставената на потребителите функционалност е само малка част от системата. „Отвъд“ видимата за потребителите страна на графичния web-интерфейс може и в повечето случаи действа допълнителна функционалност, чиито възможности и цели не могат да бъдат предвидени, нито гарантирани. Следователно не можете да поверите никой аспект от сигурността на вашата информация на отдалечен сървър. Но независимо от това, в съвременното общество е трудно да се осъществи нормална комуникация без ползването на web-базирани услуги в internet. Поради това е необходимо да подбирате такива услуги, които обезпечават вашите изисквания във възможно по-висока степен, като избягвате очевидно несъстоятелните по отношение на сигурността.

Не е лесно да се предложи сигурен критерий за разграничаване на едните от другите, но можете да следвате някои общи насоки. При всички случаи трябва да избягвате най-масовите web-базирани услуги, насочени към събиране на информация от потребителите. **Facebook**, **Google** и **TikTok** са безспорни лидери в класацията за нарушаване на вашата поверителност. В челните места на класацията от услуги, които се препоръчва да не ползвате, трябва да запишем търсещи машини като **Google**, **Bing**, **Yahoo Search** и др.; облачни услуги като **Amazon Cloud Drive**, **DropBox**, **Google Drive**, **iCloud**, **OneDrive**, и др., а също специализирани облачни услуги като **Azure**, **Office 365**, **iTunes** и др.; социални медии като **AOL**, **Facebook**, **Instagram**, **LinkedIn**, **Twitter**, **ВКонтакте**, **Одноклассники** и др.; електронни пощенски услуги като **ABV**, **Gmail**, **GMX**, **Mail.ru**, **Yahoo!** и др.; моментни комуникатори като **Hangouts**, **iMessage**, **Messenger**, **Telegram**, **Viber**, **WhatsApp** и др.

Ако прегледате Общите условия за ползване на такива услуги, ще видите, че техните собственици в прав текст „си разрешават“ да проникват във вашата информация, да я анализират и ползват за свои собствени цели, и да я предават по свое усмотрение на трети страни – без да искат вашето разрешение и без да ви дават обяснения за това. Общото за такива услуги е, че действат централизирано и са ориентирани към анализ на вашата информация, като доказателствата за целенасоченото събиране и ползване на тази информация (включително във ваша вреда) са неизброими. Препоръчваме в тази връзка да се насочвате към децентрализирани услуги, базирани на свободни технологии (доколкото това е установимо при отдалечени сървъри) и към такива, които са позиционирани в държави с по-висока правозащитна култура; далеч от властите, под чийто контрол (и в чийто интерес) се намирате вие самите. Тези критерии не гарантират надеждност на предпочитаните от вас web-услуги, но поне затрудняват в известна степен безогледното нарушаване на вашата неприкосновеност. По-долу ще предложим някои алтернативи на изброените нежелателни web-базирани услуги.

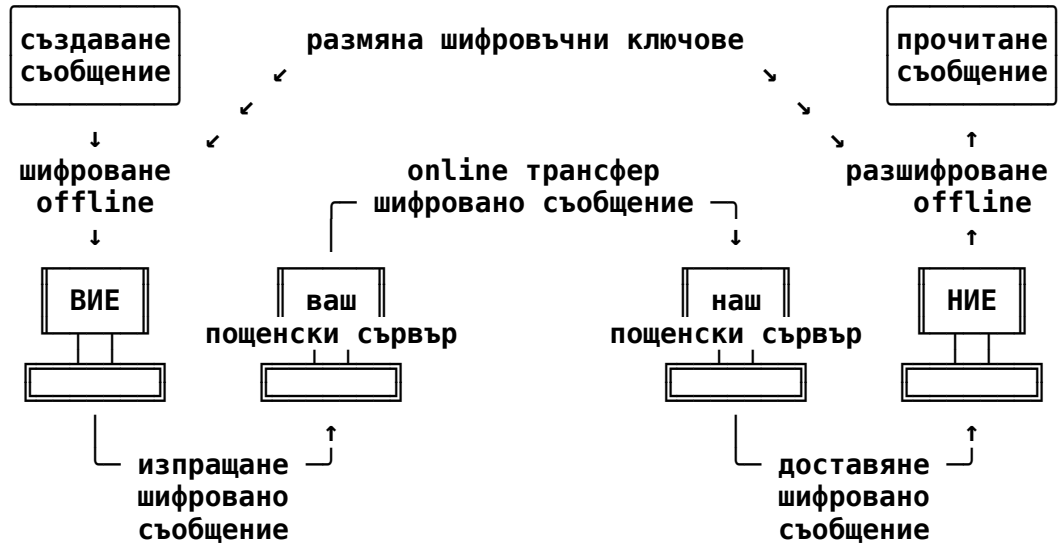
6. За ползването на електронна поща

Най-старата услуга в internet, която и до днес продължава да е сред най-масово ползваните, е електронната поща. Услугата функционира на базата на пощенски сървъри, които приемат постъпващата кореспонденция (електронни писма) и я разпределят по директории на съответните потребители. Всеки потребител може да преглежда само своята директория, за достъп до която се изисква да въведе парола. Паролата е записана и се съхранява на пощенския сървър, което позволява при заявка за достъп да се направи проверка за наличието или липсата на тъждество между въведената парола и преди това записаната – съответно заявеният достъп да бъде разрешен или отказан. Особеност на пощенската услуга е, че и паролите, и кореспонденцията се намират записани на сървъра – следователно вашата информационна сигурност при ползването на електронна поща зависи изцяло от благоволението на системния администратор на пощенската услуга. Въпреки, че в повечето случаи паролите биват съхранявани „в защитен вид“, тяхното извличане е напълно възможно (особено от система, до която системният администратор има едновременно и физически достъп, и администраторски права). Спорен остава и въпросът – изобщо необходимо ли е извличането на паролата ви – след като системният администратор е напълно възможно да достъпва вашата директория и със своите администраторски права, без изобщо да се налага да въвежда паролата ви.

Независимо от тази слабост обаче, повсеместното ползване на електронна поща налага тази услуга като стандарт, който трудно може да заобиколите в съвременни условия. Някои потребители опитват да ограничат посочените рискове за информационната сигурност, като създават свои собствени пощенски сървъри (всеки може да стори това, стига да разполага с internet-адрес (например '**Ваше_име.com**') и компютър с постоянна internet-свързаност). Това позволява пощенският сървър физически да се намира при вас и вие лично да действате като негов системен администратор. Да се гарантира обаче сигурността на устройство, което е постоянно свързано с internet, не е лесна задача (особено за потребител без задълбочени компютърни познания и време, за да осъществява постоянен системен надзор). На следващо място – необходимостта пощенският сървър да работи непрекъснато във времето изисква осигуряването на техническа поддръжка, което не всеки потребител е в състояние да направи. И накрая – web-адресите на собствените пощенски сървъри в повечето случаи не са широко известни – следователно изпращаните от тях съобщения често се квалифицират от утвърдените пощенски сървъри като нежелана поща (**Spam**) и се сортират в директориите за нежелана поща; или директно се изтриват, без изобщо да достигнат до получателя. Ето защо за справянето с подобна задача е необходимо да се въоръжите с достатъчно търпение и да положите известни усилия – които излизат извън обхвата на това ръководство.

Пощенската услуга, предоставяна от отдалечен пощенски сървър, който не е под ваш контрол, по дефиниция няма как да бъде сигурна. Ето защо поверителната информация, която изпращате и получавате чрез електронна поща, трябва предварително да бъде шифрована offline в системата на изпращача и да постъпва в неговия пощенски сървър във вече шифрован вид. Респективно, шифрованата информация трябва да бъде разшифрована отново offline, едва след като бъде свалена от пощенския сървър на получателя в неговата собствена система, offline. В никакъв случай това (раз)шифроване не трябва да се поверява на web-базирани приложения (каквито все по-голяма част от електронните пощенски услуги се опитват да предлагат на своите потребители). Ако се възползвате от такова предложение, означава да поверите и шифроването, и трансфера на един и същи източник. И ако системният администратор на пощенския сървър бъде въвлечен в евентуална атака срещу вас и/или вашите кореспонденти, това ще компрометира информационната сигурност. Ето защо се препоръчва да разглеждате подобни приложения единствено като

допълнителен слой към сигурността, който само може да допълни и надгради обезпечаваната лично от вас и от вашите кореспонденти сигурност.



Принципна схема на надеждното шифроване при ползване на e-mail

За шифроването на електронни съобщения (особено между кореспонденти, които не са се срещали преди това, за да могат да разменят помежду си общ шифровъчен ключ или парола за (раз)шифроване на съобщенията), се ползва т.нар. „асиметрично“ шифроване, на което ще се спрем подробно в Част **VIII**. Тук е достатъчно да отбележим, че този вид шифроване позволява предоставянето на специален 'публичен' ключ (в нешифрован, публично достъпен вид), който обаче може да се ползва единствено за шифроване. Обратният процес по разшифроването е възможен единствено чрез прилагането на 'частен' ключ, асоцииран еднозначно с 'публичния' ключ и пазен в дълбока тайна от неговия притежател. Това позволява кореспондентите да разменят помежду си своите 'публични' ключове и да започнат да разменят надеждно шифровани съобщения помежду си, без да е необходимо преди това да са съставили обща конвенция за (раз)шифроване и да са разменили по сигурен и независим канал каквито и да било общи шифровъчни ключове или пароли.

Едно от предимствата на „асиметричното“ шифроване е това, че то позволява (раз)шифроването да се осъществи изцяло offline, напълно независимо от пощенските сървъри. В повечето случаи това изглежда по-неудобно от предлаганите web-функции на вашата електронна поща: не може да шифрова заглавието (**Subject**) на електронното писмо; шифроването на прикачените файлове (**Attachments**) трябва да стане отделно от шифроването на самото писмо (освен ако например не оформите и писмото като прикачен файл); и видът, броят, размерът и наименованията на прикачените файлове продължава да бъде видим след шифроването (освен ако не поставите тези файлове в общ архивен **`.zip`** файл с неутрално наименование (например '**`attachments.zip`**') и по този начин ги шифровате наведнъж, вместо един по един). Въпреки тези неудобства ви насърчаваме да предпочетете именно този подход, който разчита на вашите независими от пощенската услуга възможности за обезпечаване на сигурността.

Независимо от цялата условност на сигурността при отдалечените пощенски сървъри, все пак за предпочитане са тези пощенски услуги, които са базирани физически в държави с по-стабилна правозащитна рамка и утвърдени традиции в отстояване Върховенството на закона. Това би направило относително по-трудно нарушаването на вашата неприкосновеност и по-непримирима съпротивата на трети страни (включително системните администратори) срещу подобни посегателства. Като пример за такава съпротива можем да посочим пощенската услуга **Lavabit**, която предпочете да затвори врати, вместо да предаде на американските специални служби своите потребители. Добра мярка е да се ориентирате към пощенски услуги, базирани в такива държави, които не си сътрудничат с властите у вас – това ще ограничи възможностите на вашите специални служби (и приближените до тях криминални кръгове) да получат съдействие от тамошните служби за проникване във вашето потребителско съдържание (или поне ще ги принуди да потърсят формално приемлив повод за това).

Желателно е също така вашата пощенска услуга да е базирана на свободни технологични решения (колкото и да е спорен въпросът за установяване на технологиите, с които работи един отдалечен сървър, до който нямате физически достъп и права като негов системен администратор).

Прилагането на **SSL**-технологии за шифроване на вашата връзка с пощенския сървър (видно от извеждането на **https://** в адресната лента) е задължително и вече е наложено като масов стандарт сред почти всичките доставчици на електронни пощенски услуги.

За предпочитане са електронните пощенски услуги, които не събират данни за своите потребители (с цялата условност на гарантирането на подобно обстоятелство). В този смисъл решения като „двуфакторната“ защита (налагащи например да въведете свой **GSM**-номер) са неприемливи – въпреки благовидния им претекст. Не е желателно също така да работите с един и същи доставчик на електронна поща и на други услуги, които ползвате в internet. Класическа грешка е например да работите едновременно с **Gmail** и с **Google** – това би позволило на доставчика на тези услуги напълно координирано и удобно да проследява както вашата e-mail кореспонденция, така и осъществяваните от вас търсения в internet. Сегментирайте информацията и насочвайте отделните сегменти през различни информационни потоци, вместо да предоставяте единен и централизиран достъп до всичките си активности.

Без да сме изчерпателни, следните електронни пощенски услуги в по-голяма или по-малка степен се съобразяват с изложените по-горе изисквания (като пак подчертаваме, че сигурността е ваша отговорност и не следва да я прехвърляте на никакви online услуги):

`https://www.autistici.org/`
`https://bitmessage.ch/`
`https://disroot.org/`
`https://protonmail.com/`
`https://riseup.net/`
`https://tutanota.com/`

Независимо от всичко казано дотук, за вашите най-поверителни дейности трябва да бъдете особено предпазливи при ползването на електронна поща. Може би няма да бъде излишно, ако си създадете няколко различни акаунта за различни функции и цели, като влизате в различните акаунти от различни устройства и различни точки за свързване с internet. Така бихте сегментирали комуникацията и това би затруднило насочването на евентуални атаки поне срещу част от вашите информационни потоци. При това не пропускайте да помислите и за дизайна на вашите потребителски наименования, като проявите известно творчество. Например електронните пощенски адреси '**`ivan-ivanoff-lawyer@protonmail.com`**' и '**`legal.office.ivan.ivanov@unseen.is`**' изобщо не изглежда да са притежание на различни лица и (ако данните не са фиктивни като тези в примера) установяването на конкретния им притежател (идентифициран например с неговите действителни имена и професия) няма да бъде трудно. Същото се отнася за начина, по който оформяте цялостното съдържание на вашите електронни писма (заглавия, визитки и други подобни). Съществуват технологии за търсене в internet на специфични думи и изрази, които е възможно да бъдат насочени към локализиране на вашата комуникация чрез прихващането на типични за вас начини на изразяване. Ако се налага, проявете творчество и по този въпрос – дори и това да не съответства на представите ви за стил.

Препоръчва се да създадете различни електронни пощенски адреси за вашите различни дейности и нужди. По този начин ще сегментирате различните си роли и позиции в обществото, и ще направите по-трудно цялостното установяване на кръговете от кореспонденти, с които общувате. Също така това ще повиши нивата на информационна сигурност по отношение на други internet-ресурси, които управлявате от вашите електронни пощенски адреси. Добра идея е в тази връзка електронният пощенски адрес, от който например се очаква потвърждение за пренасочване на вашите web-адреси, да бъде отделен от адресите, през които минава текущата ви комуникация и които достъпвате ежедневно (съответно въвеждате паролите за достъп до тях постоянно, на различни компютърни устройства, достъпващи internet чрез всевъзможни канали и доставчици).

За някой особено поверителен случай може дори да се наложи да отидете на неочаквано място (където не е лесно да се предположи, че бихте отишли и където не е лесно да бъдете проследени), за да създадете отделен, инцидентен

акаунт, предназначен само за този случай. Такъв акаунт не трябва да се достъпва от традиционните за вас устройства и точки за свързване с internet. И разбира се – това няма да допринесе по никакъв начин за вашата информационна сигурност, ако от „новото“ място започнете да осъществявате традиционните си активности в internet (да влизате в други ваши акаунти, да отваряте любимите ви новинарски и развлекателни сайтове, да пишете и да се държите по характерния за вас начин). За да се защитите от евентуален интерес към вашата internet-комуникация в такъв особено поверителен случай, не трябва да давате никакъв повод за извеждането на заключение, че „това сте вие“ (в най-общия смисъл на думата). В противен случай (особено ако евентуалният интерес към вас е обезпечен с достатъчно ресурси) е много вероятно идентичният (или достатъчно сходен) модел на поведение да бъде засечен и разпознат, и всичките ви усилия да отидат напразно.

7. За ползването на моментна комуникация

Моментната комуникация позволява провеждането на чат-разговори и директен (за предпочитане **Peer-to-Peer**) обмен на файлове, а в някои случаи – осъществяването на аудио- и видео-конференции. Предпочитана алтернатива на несвободните моментни комуникатори е например **Jabber**, който функционира на база децентрализирания **XMPP** протокол (**Extensible Messaging and Presence Protocol**). Този протокол позволява всеки желаещ (вие включително) да пусне и поддържа свой собствен **XMPP**-сървър като част от децентрализираната комуникационна мрежа на **Jabber**. Това отличава **Jabber** от повечето масови системи за моментна комуникация (**Hangouts**, **iMessage**, **Telegram**, **Viber**, **WhatsApp** и др.), при които целият трафик преминава през един единствен централен сървър (или клъстер от сървъри) и позволява лесно да бъде установен пълен контрол над комуникацията (независимо от многобройните заявки за „сигурно шифроване“).

`https://xmpp.org/`

Освен всичко друго, **Jabber** предлага цялостно (**End-to-End**) шифроване (което действително се осъществява на вашия компютър, с ключове, които са създадени и се съхраняват на вашия компютър).

За да можете да ползвате **Jabber**-комуникация, трябва на първо място да си направите акаунт на някой от действащите сървъри в децентрализираната мрежа. За целта можете да изберете един от тях (или ако предпочитате и имате възможност, да създадете свой собствен) и да впишете свое потребителско име и парола. В повечето случаи това може да се осъществи от графичен web-интерфейс, който не се отличава например от регистрирането на електронна поща. Самият потребителски акаунт изглежда тъкмо като електронна поща (например акаунтът на **`www.Advocati.org`** е наименован **`ailiev@advocati.org`**) – въпреки, че това не е електронна поща и не функционира като такава).

`https://list.jabber.at/`

На следващо място трябва да се снабдите с **Jabber**-клиент – програма, която да инсталирате на вашата система и от нея чрез избрания от вас сървър да приемате и изпращате моментни съобщения. Често ползван **Jabber**-клиент е например **Pidgin**. Предимството на **Pidgin** е, че той позволява поддръжката на много акаунти едновременно, включително на редица други комуникационни протоколи освен **XMPP** (сред които **AIM**, **Bonjour**, **Gadu-Gadu**, **Google Talk**, **Groupwise**, **ICQ**, **IRC**, **SILC**, **SIMPLE**, **Sametime**, **Zephyr**). Алтернативно можете да изберете например **Jabber**-клиента **Gajim**, който в някои отношения е по-интензивно поддържан и в последно време се предпочита например от потребители, работещи паралелно с няколко отделни устройства.

`https://www.pidgin.im/`
`https://gajim.org/`

Можете да инсталирате **Pidgin** чрез командата '**`pacman -S pidgin`**'; респективно – да инсталирате **Gajim** чрез командата '**`pacman -S gajim`**'.

След като сте създали свой акаунт на някой от действащите **XMPP**-сървъри, трябва да впишете своя акаунт в избрания от вас клиента за моментна комуникация. От графичния интерфейс на **Pidgin** можете да направите това от менюто

Accounts / **Manage accounts** / **Add...**, при което ще се отвори диалогов прозорец, където трябва да отбележите от падащото меню, че ще ползвате **XMPP**-протокол и да впишете потребителско наименование (**Username**), избрания от вас **XMPP**-сървър (**Domain**) и паролата си за достъп до акаунта. Останалите възможности не са задължителни. По сходен начин трябва да постъпите и при **Gajim**, като направите това от менюто **Accounts** / **Modify Accounts...**, при което ще се отвори диалогов прозорец, в който трябва да натиснете '**AddAccount**'.

За да започнете комуникация с друг **Jabber**-потребител, трябва да го потърсите от **Buddies** / **Add Buddy...** (респективно **Accounts** / **Add Contact...**) при което ще се отвори диалогов прозорец, където трябва да въведете наименованието на потребителя, с който желаете да установите контакт – пълно изписване, наподобяващо електронна поща. Ако вече сте вписали няколко свои акаунта в **Jabber**-клиента, трябва да уточните от името на кой от тях желаете да потърсите въпросния потребител (**Jabber**-клиентите ви позволяват да поддържате паралелно няколко канала на комуникация с групи от кореспонденти, които нямат нищо общо помежду си). До посочения от вас потребител (ако има такъв) ще достигне известие, че акаунт с вашето наименование желае да установи контакт с него; и той ще може да ви ауторизира (**Authorize**). Докато това не бъде извършено и от него, и от вас, въпросният потребител ще стои с червен забранителен индикатор в списъка ви, отбелязан като неоторизиран (**Not Authorized**). Когато ауторизацията приключи и от ваша, и от негова страна, ще започнете да виждате текущ online-статус (зелен – когато акаунтът е достъпен; сив – когато не е на линия; жълт – когато от известно време отсъства; червен – когато е отбелязал, че не желае да бъде обезпокояван).

Jabber позволява размяна на моментни текстови съобщения – включително тогава, когато единият от кореспондентите в момента не е на линия. В такъв случай съобщенията до него се запазват временно на неговия **Jabber**-сървър и се получават при първото му следващо включване).

Jabber позволява и директен (**Peer-to-Peer**) трансфер на файлове, включително ако са с голям обем (докато двамата потребители са едновременно на линия). Трансферът в този случай се осъществява директно между двата компютъра, без посредничеството на **Jabber**-сървърите. В останалите случаи файловете се пазят временно на сървъра на получателя.

https://en.wikipedia.org/wiki/Peer-to-peer

Преди да започнете да изпращате поверителни съобщения през **Jabber**, е добре да въведете указание разговорите ви да не се архивират от системата. Това можете да направите в **Pidgin** от **Tools** / **Preferences** / **Logging**, където да премахнете отметките и на трите възможности: записване на цялата моментна комуникация (**Log all instant messages**), записване на всички разговори (**Log all chats**) и записване на всички промени в системния журнал (**Log all status changes to system log**). В **Gajim** можете да направите същото от **Accounts** / **Modify Accounts...**, при което трябва да изберете съответния ваш акаунт и в отворения диалогов прозорец да влезете в **Privacy** / **Keep Chat History** (където от падащото меню трябва да посочите **Until Gajim is Closed**).

Pidgin притежава още една важна функционалност, за която съществуват редица независими потвърждения относно затрудненията, които създава при опити за проследяване в internet. Става дума за **OTR** (**Off-the-Record**) шифроването, с което може да бъде надграден вашият **Pidgin**-клиент. **OTR** служи за надеждно идентифициране на вашите кореспонденти и за допълнително шифроване на комуникацията ви с тях на базата на специални **OTR**-шифровъчни ключове, генерирани по време на идентификацията.

https://otr.cypherpunks.ca/

Ако сте избрали да ползвате **Jabber** посредством клиента **Pidgin**, можете да инсталирате **OTR** чрез командата '**pacman -S pidgin-otr**'.

Ако сте избрали да ползвате клиента **Gajim**, можете да инсталирате аналогичната шифровъчна функция **OMEMO**, която улеснява работата паралелното от няколко отделни устройства и позволява надеждно шифроване включително когато единият потребител е offline – за сметка обаче на по-малко надеждни подходи и инструменти за автентикация.

https://conversations.im/omemo/

След като инсталирате **OTR**, можете да го активирате от графичния интерфейс на **Pidgin** – като изберете **Tools / Plugins...**, където трябва да поставите отметка за обмяна на съобщения с **OTR**. Имайте предвид, че за да можете да ползвате функциите на **OTR**, вашите кореспонденти трябва също да го активират. Имайте предвид и това, че **OTR** по подразбиране може да блокира разговорите ви с кореспонденти, които не са идентифицирани надеждно и съответно разговорът с тях не е шифрован. За да позволите протичането на разговори и без **OTR**-шифроване, след като активирате тази функция, трябва да изберете **Tools / Plugins / Off-the-Record Messaging** и от там да премахнете отметката за изискване на поверително съобщаване (**Require private messaging**). В противен случай системата няма нито да изпраща, нито да приема съобщения от и до неавтентицирани и нешифровани кореспонденти.

Когато **Pidgin** бъде надграден с **OTR**, в горната рамка на комуникационните прозорци ще започне да се появява допълнителната опция **OTR**, която със сив цвят показва липса на активиране; със зелен цвят – положителна автентикация на вашия кореспондент и шифрованост на разговора с него; и с червен цвят – съмнение относно автентичността на кореспондента и съответно липсата на шифроване. **OTR**-шифроването се активира от опцията **OTR** в комуникационния прозорец със съответния кореспондент, от която трябва да изберете започване на поверителен разговор (**Start private conversation**). При това системата ще стартира процедура по автентикация на вашия кореспондент (съгласно вашите предпочитания – посредством някой от следните три метода: Въпрос и отговор; Споделен секрет; или Ръчно сравняване на 'отпечатъци'). Ако от процедурата се изведе положителна автентикация, системата ще създаде **OTR**-шифровъчен ключ, с който да се шифрова по-нататъшната комуникация между вас и вашия кореспондент.

Въпрос и отговор (**Question and answer**) е метод, който позволява да въведете в системата кратък и конкретен въпрос, на който кореспондентът ви да отговори също толкова кратко и конкретно. Трябва да сте сигурни, че вашият кореспондент и единствено той би се сетил и би въвел напълно вярно заложения от вас отговор. Системата ще изпрати вашия въпрос (без да показва отговора) и ще предостави празно поле за попълване на отговор. Ако отсрещната страна въведе точно отговора, който сте въвели и вие (напълно точно, без да променя нито един символ!), ще бъде изведена положителна идентификация; ако няма пълно съответствие, ще бъде изведена отрицателна идентификация. Тази опция е полезна, ако например сте били заедно наскоро или имате устойчиви отношения с вашия кореспондент, което ви позволява да зададете специфичен въпрос, на който е трудно трети страни да отговорят вярно – например „Какъв цвят беше колата под наем, която наехме миналата година в Прага?“; или „Каква дума ми се падна от късметчето, което ми сервираха с кафето вчера?“. При това трябва да имате предвид, че системата различава големи и малки букви, кирилица и латиница, цифри и думи – като при парола. За да не бъде извеждана отрицателна идентификация дори и тогава, когато кореспондентът ви отговаря принципно вярно (например на късметчето с кафето може да е пишело '**любов**', '**Любов**', '**lyubov**', '**liubov**' или '**LUBOV**') – е препоръчително с вашия кореспондент да уговорите някаква предварителна конвенция относно начина за въвеждане на отговорите (примерно: само на кирилица и само с големи букви; което би свело вариациите до единствената възможна: '**ЛЮБОВ**'). Не е невъзможно такава конвенция да направите и по време на самата моментна комуникация, но тогава давате повече възможности на трети страни, които биха прихванали все още нешифрованата комуникация, да вземат предвид конвенцията при опит за компрометиране на процедурата.

Споделен секрет (**Shared Secret**) е метод, който позволява да въведете конкретна, предварително уговорена с вашия кореспондент парола. Системата ще изпрати запитване за споделения секрет (без никакви допълнителни подробности) и ще предостави празно поле за попълване на паролата. Ако отсрещната страна въведе точно паролата, която сте въвели и вие (напълно точно, без да променя нито един символ!), ще бъде изведена положителна идентификация; ако няма пълно съответствие, ще бъде изведена отрицателна идентификация. Недостатък е, че този метод не предвижда никаква гъвкавост (за разлика от предходния, който позволява задаването на всевъзможни инцидентни въпроси, без предварителна уговорка). Заради това се препоръчва при прилагането на този метод да се ползват предварително уговорени „динамични пароли“ – които зависят например от поредния ден на седмицата, от съдържанието на последното въведено съобщение и т.н. Това би направило следващата парола трудна за предвиждане (дори и ако предходния път паролата е била прихванатата), тъй като допълнително трябва да бъде прихванат и договореният помежду ви фактор за динамично променяне на паролите.

Ръчното сравняване на 'отпечатъци' (**Manual fingerprint verification**) се базира върху т.нар. 'отпечатъци' (идентификатори; за чието функциониране ще говорим подробно в Част **VIII**). На този етап е достатъчно да

отбележим, че 'отпечатъците' (идентификаторите) са специални буквено-цифрени комбинации, които представляват сложна криптографска производна – в настоящия случай от ползвания **Jabber**-акаунт и неговото хардуерно-софтуерно осигуряване. Когато изберете тази функция, системата ще изведе вашия 'отпечатък' и предполагаем 'отпечатък' за отсрещната страна. 'Отпечатъкът' за отсрещната страна е предполагаем, защото той представлява криптографска производна от един **Jabber**-акаунт и неговото хардуерно-софтуерно осигуряване, но все още няма доказателство, че този акаунт и това хардуерно-софтуерно осигуряване действително се ползват от вашия кореспондент. Поради тази причина трябва да получите (или да сте получили предварително) действителния 'отпечатък' на кореспондента ви – чрез независим канал, който е трудно да бъде компрометиран едновременно с **Jabber**-комуникацията. След като прегледате 'отпечатъка' (един по един всичките символи – тъй като понякога е възможно два 'отпечатъка' да се окажат „твърде подобни“), трябва да отбележите сами (ръчно) дали получената идентификация е положителна или отрицателна. Недостатъкът на този метод е, че при преинсталиране на системата или при преминаване към друго устройство, ако не прехвърлите автоматично генерирания шифровачен **OTR**-ключ, тогава **Pidgin** автоматично ще генерира нов 'отпечатък', което същевремено ще наложи неговото повторно удостоверяване.

При **Gajim** (след като инсталирате **OMEMO** чрез софтуерния пакет '**`gajim-plugin-omemo`**') се ползва единствено процедурата за ръчно сравняване на 'отпечатъци' (идентификатори) – със съществено предимство при паралелно ползване на няколко устройства, позволяващо вписването на 'отпечатъци' за всяко от тях и продължаване на комуникацията необезпокоявано, въпреки междувременното превключване на някой от кореспондентите от едно от неговите устройства на друго. Когато инсталирате **OMEMO**, долу вдясно на комуникационния прозорец с конкретен потребител ще видите малък катинар с удивителна, от който можете да избирате между възможностите **disabled** и **OMEMO**. Ако изберете втората възможност, **OMEMO** ще се включи и ще извлече 'отпечатък' от хардуерно-софтуерното осигуряване на вашия кореспондент, като вие трябва да решите дали се доверявате (**Trust**), доверявате се на сляпо (**Blind Trust**) или отхвърляте (**Untrust**) този 'отпечатък'. За целта трябва да получите потвърждение по независим канал от вашия кореспондент, че действително това е неговият 'отпечатък' (идентификатор). За да можете да потвърдите 'отпечатъка' на собственото ви устройство, трябва след като изберете възможността **OMEMO** долу вдясно, да изберете иконата с щита в ъгъла и да погледнете горе на диалоговия прозорец (**Fingerprint for this Device**), където ще намерите подобна **64**-буквено-цифрена комбинация:

`6EB94D72 C1F8DD05 808E4512 70AD7E04 773406B4 823E7531 42B00437 AE0D4490`

'Отпечатъците' на вашия кореспондент (с цветни икони, обозначаващи статуса им преди всеки от тях) са под този на вашето устройство. От всяка от иконите можете да променяте статусите, включително да изтривате съответните 'отпечатъци'.

Имайте предвид, че ако настройките на **Gajim** бъдат загубени (примерно при преинсталация на системата), или ако инсталирате **Gajim** на ново устройство, тогава ще бъде изведен нов 'отпечатък' и **OMEMO** ще изиска ново потвърждаване.

Следва за пълнота да отбележим, че **OMEMO** се поддържа и от клиента **Pidgin** – за повече информация можете да погледнете този адрес:

`https://github.com/gkdr/lurch`

Независимо от ползвания метод, крайният резултат е положителна идентификация на отсрещния шифровъчен ключ, който ползвате за шифроване на комуникация с отсрещния кореспондент. В комуникационния прозорец с този кореспондент се извежда индикатор за нормално състояние. Това състояние ще се запази и в следващите ви сесии с този кореспондент – до момента, до който вие или вашият кореспондент не премине на ново устройство или преинсталирана системата си, без да прехвърли своя шифровачен ключ. Това довежда до генерирането на нов ключ с нов 'отпечатък', при което кореспондентът, който преди е бил идентифициран положително, ще бъде отбелязан като такъв със съмнение относно автентичността и ще трябва да прибегнете отново до идентификация на неговия ключ по някой от предложените методи.

Следва да подчертаем, че и при **Pidgin**, и при **Gajim** системата извежда индикатор за нормално състояние само на база непроменените от последния път 'отпечатъци' на отсрещния шифровачен ключ – а не на база някакво

„разпознаване“ на потребителите, които стоят зад този хардуер и софтуер. Винаги (особено ако ще доверявате поверителна информация) имайте предвид възможността вашият кореспондент да е бил компрометиран по някакъв начин и сега вместо от него, системата му да се ползва от трета страна.

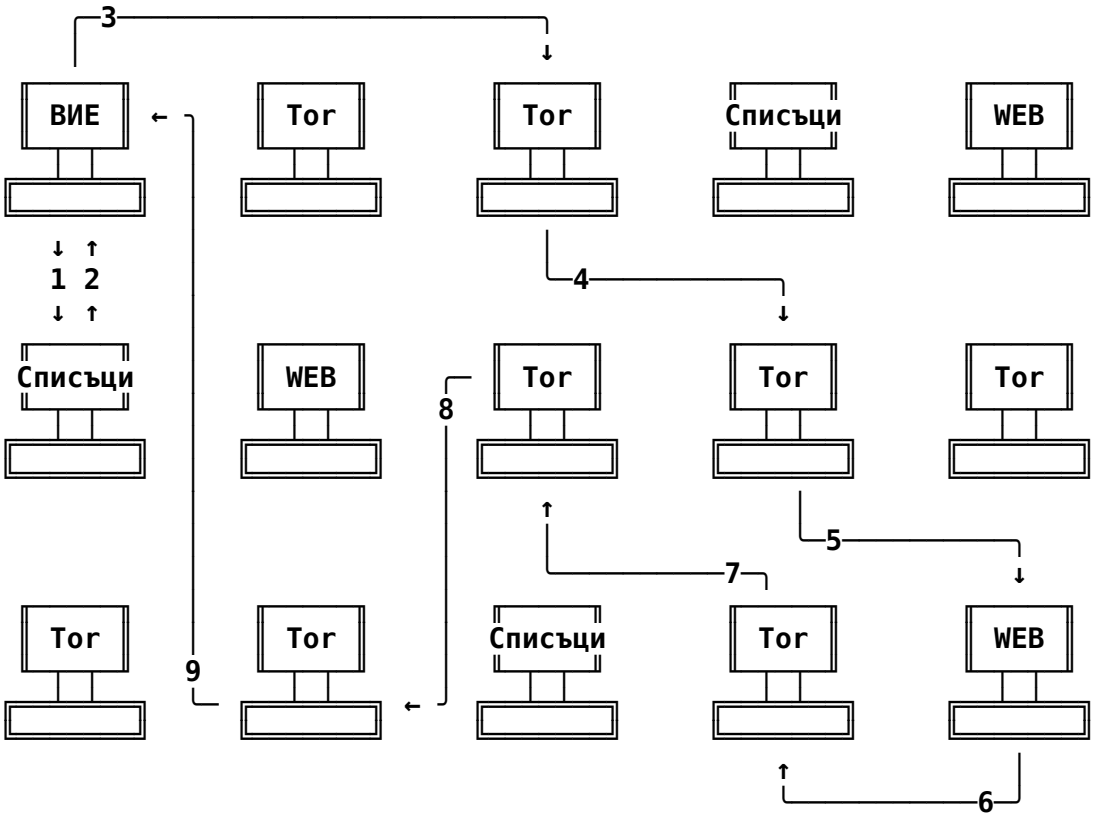
8. За ползването на анонимизация в internet

В някои условия е възможно да бъдете подложени на проследяване, при което дори самото свързване с определен „забранен“ или „подозрителен“ ресурс в internet може да застраши вашата сигурност (например в условията на деспотичен политически режим или рестриктивна корпоративна политика). В подобни условия е възможно дори самото посещаване например на предложените в настоящото изложение internet-връзки или свалянето на препоръчания от нас Свободен софтуер да ви обозначи (в нечии очи) като „нарушител“ и да се превърне в компромат срещу вас. В такъв случай се нуждаете освен от сигурна система, в която не е лесно да се проникне без разрешение, също така и от сигурно средство за достъп до internet, което препятства проследяването на вашите активности – както от вашия internet-доставчик, така и от системния администратор на интересуващите ви internet-ресурси. Един от може би най-развитите проекти с такава функционалност е **Tor** (**The Onion Router**).

https://www.torproject.org/

Ирония на съдбата е, че първоначално **Tor** е разработван и пуснат в действие от Военното разузнаване на **САЩ** – за осигуряване надеждното (скрито) свързване на агентурния апарат на терен във вражески територии с американските служби. За да може да се случи това по надежден начин (без да е необходимо агентът да рискува своето разконспириране, ако например у него се намери копие от секретен американски софтуер), властите в **САЩ** са били принудени да оповестят публично изходния код на **Tor** и да го направят свободен за сваляне от internet. Били са принудени също така да обезпечат надеждността на **Tor** и да не поставят „задни врати“ в кода на софтуера – защото в противен случай чуждите контраразузнавателни служби (които също развиват своя технологичен потенциал) биха могли да локализират „задните врати“ и да компрометират свързването. Така е бил създаден надежден софтуер за анонимизация на активностите в internet, който да гарантира, че трети страни не могат да проследят нито с какви internet-ресурси се свързвате, нито откъде идват вашите заявки спрямо проследяван от тях internet-ресурс; който софтуер е бил предоставен за свободно сваляне, за да могат и агентите да го свалят, когато се наложи. Днес **Tor** вече се поддържа и усъвършенства от редица университети и неправителствени организации, които работят за осигуряването на надеждна internet-комуникация на журналисти, дисиденти, правозащитници. Въпреки цялата условност на сигурността на **Tor** (не е изключено все пак службите на **САЩ** да са запазили „за себе си“ определени компрометиращи функционалности в този софтуер, които независимите разработчици да не успяват да разберат), индикатор за неговата надеждност е например забраняването му в редица държави с деспотични политически режими и усилията за ограничаването му от някои западни правителства.

Tor работи на базата на доброволна мрежа от сървъри, действащи като пренасочващи възли (**Tor Nodes**), които са разположени по целия свят и служат за анонимизиране на internet-трафика. Всеки (включително и вие) може да пусне в действие свой собствен пренасочващ възел. Когато активирате **Tor**, системата най-напред изгражда шифрован канал и се свързва с някой от сървърите, на които има актуален списък с пренасочващите **Tor**-възли (**1**). След като стори това, системата актуализира своя собствен списък (**2**) и ви позволява да работите със специално конфигуриран браузър **Firefox**, който е програмиран да фалшифицира параметрите на вашата система и да насочва трафика през произволен ред от посочените в списъка пренасочващи възли (**3**, **4**, **5**). Същевременно никой от пренасочващите възли не разполага с пълната информация откъде идва конкретна заявка, какво съдържа и накъде отива. Така – дори и някой от пренасочващите възли да се окаже компрометиран, е трудно неговите собственици да извлекат нещо съществено за вашата идентичност и за активността ви в internet; още повече, че следващата ваша заявка ще бъде пренасочена по нов маршрут и е малко вероятно отново да премине през възел, който да е компрометиран от същата трета страна. Резултатите от подадените от вас заявки към съответния internet-ресурс се връщат към вас отново през произволен ред от пренасочващи възли (**6**, **7**, **8**, **9**), като маршрутът всеки следващ път отново е различен. Така, ако достъпваният от вас internet-ресурс на свой ред се окаже компрометиран, от там няма да бъде лесно да се проследи къде е физическото ви местонахождение, какви са параметрите на ползваната от вас система и какво друго правите в internet, освен че ползвате съответния ресурс.



Принципна схема на начина на функциониране на мрежата Tor

За да ползвате **Tor**, може просто да свалите **Tor Browser** от официалния web-сайт на проекта **Tor** и да го разархивирате на предпочитано от вас място във вашата система. **Tor Browser** представлява портативен (**Portable**) софтуер (не се нуждае от инсталиране в системата и работи направо от своята директория, която можете да държите където желаете, включително на външен носител с памет). Единствено трябва да съобразите версията на **Tor Browser** с вида на вашата система (**GNU/Linux**, **Windows**, **MacOS**, **Android** или друго) и нейната архитектура (**32**- или **64**-битова). Софтуерът включва инструменти за анонимизация в internet и специално конфигуриран портативен браузър **Firefox**, който функционира независимо от системните настройки на вашето устройство, като препятства събирането на данни за вас от вашия internet-доставчик и от системните администратори на достъпваните internet-ресурси (или по-точно – предоставя фалшифицирани данни за обикновен **Windows**-потребител, които е трудно да бъдат разпознати като фалшиви).

Когато свалите софтуера, трябва първо да го разархивирате. На съответното място във вашата система ще се образува директорията '**`tor-browser_en-US`**' (в дадения пример '**`en-US`**' обозначава англо-американската версия; в зависимост от вашите предпочитания можете да свалите и версия на друг език, но ви съветваме да се придържате към най-масовата англоезична **US**-версия, която е и най-безлична – освен ако определени съображения във вашата държава не налагат друго). В разархивираната директория ще откриете още една директория (с наименование '**`Browser`**') и файл за стартиране '**`Tor Browser Setup`**'. Можете да задействате софтуера, като изберете стартиращия файл '**`Tor Browser Setup`**' – ще се стартира диалогов прозорец за конфигуриране (когато стартирате **Tor Browser** за първи път). Ако вече сте конфигурирали софтуера, файлът за стартиране ще има наименование '**`Tor Browser`**' (за стартиране на вече конфигурирания софтуер).

Когато го отворите, **Tor Browser** ще се отвори в по-малка рамка от вашия дисплей. Не го преоразмерявайте! Достъпваните от вас internet-ресурси вземат предвид размера на рамката, за да представят по подходящ начин своето съдържание. По размера на рамката обаче могат да се правят извод за размера на дисплея и от там – за вида на устройството. Ето защо рамката на **Tor Browser** по подразбиране се отваря в по-малки размери, съответстващи на някое от другите възможни устройства, различни от вашето – добре е да поддържате тази заблуда, въпреки неудобството от по-малките размери.

В случай, че всичко е наред, когато отворите **Tor Browser**, в прозореца ще видите страница със съобщение „Поздравления! Този браузър е конфигуриран да работи с Tor“ (**Congratulations! This browser is configured to use Tor**). Ако вашето копие е остаряло (междувременно са направени подобрения в софтуера, които е наложително да актуализирате с оглед отстраняване на новоустановени слабости в сигурността), ще видите съобщение „ВНИМАНИЕ! този браузър не е актуализиран“ (**WARNING: this browser is out of date**) с покана да го обновите. С плътна стрелка ще бъде посочено логото на **Tor** горе вляво, откъдето в падащото меню можете да изберете опцията „Виж за обновяване на **Tor Browser**...“ (**Check for Tor Browser update...**), при което обновяването ще започне. Когато процесът приключи, ще ви се даде възможност да продължите да ползвате **Tor Browser** в завареното му състояние или да го рестартирате, за да се приложат всичките обновления. Препоръчваме да изберете втората възможност, като натиснете '**Restart Tor Browser**' – ще последва рестарт и ще видите съобщение „Tor Browser е обновен“ (**Tor Browser has been updated**). Сега вече можете да ползвате софтуера пълноценно.

Недостатък на **Tor** е, че работи чувствително по-бавно, отколкото когато се свързвате с internet директно. Причината за това е, че вашият трафик се пренасочва многократно към различни точки по света, което бави изпълнението на заявките ви. Друг недостатък е затрудняването ви да ползвате някои internet-ресурси, които поддържат технологии за проследяване на потребителите (независимо дали това е насочено към компрометирането им или чистосърдечно служи за „подобряване на услугите“). Пряка причина за това е честото сменяне на вашето физическо местоположение (от гледна точка на достъпвания internet-ресурс, който получава заявките ви ту от едни, ту от други пренасочващи възли, които могат да са разположени на хиляди километри един от друг). Подобни „телепортации“ несъмнено пораждат подозрение за това, че ползвате **Tor** (което не се харесва нито на специалните служби, нито на маркетинговите агенции) и в редица случаи води до ограничаване на достъпа (или до по-задълбочено разследване). За да избегнете подобен проблем (а така също, за да се съобразите с изискванията – когато например определен internet-ресурс е забранен за определени територии или е разрешен само за тях), можете да настроите **Tor** така, че вашият последен пренасочващ възел (**Exit Node**) да бъде винаги с физическо местонахождение в конкретна посочена от вас държава (или обратното – да бъде с физическо местонахождение в държави, различни от посочените). Това е полезно, включително и когато желаете да скриете от достъпваните internet-ресурси, че ползвате **Tor**.

Можете да настроите вашия **Tor Browser** да осъществява достъп до интересуващите ви internet-ресурси само от последен пренасочващ възел, базиран в предпочитана от вас държава, като въведете указание за това в конфигурирания файл '**torrc**', намиращ се в директорията '**tor-browser_en-US/Browser/TorBrowser/Data/Tor**' (където под '**en-US**' разбираме предпочетената в нашия пример англоезична версия на софтуера). В изходно положение системният файл '**torrc**' е празен (в него няма никакви указания за предпочитана държава), като можете да добавите вашето изискване (чрез програмата **Nano** или чрез командата '**cat**') с подобен код:

```
ExitNodes {us}
StrictNodes 1
```

(където под '**us**' разбираме примерното ни предпочитание последният пренасочващ възел да е базиран винаги в **САЩ**). Няма пречка да въведете двубуквеното кодово обозначение на коя да е предпочитана държава съгласно международния стандарт **ISO 3166-1 alpha-2**).

```
https://bg.wikipedia.org/wiki/ISO_3166-1
```

Няма пречка да конфигурирате **Tor** да насочва вашите заявки така, че техният последен пренасочващ възел да бъде базиран и в една от няколко предпочитани от вас държави – с подобно указание:

```
ExitNodes {us},{fr},{de}
StrictNodes 1
```

(където сме посочили примерното ни предпочитание последният пренасочващ възел да е базиран в някоя от държавите **САЩ**, **Франция** или **Германия**).

Можете да конфигурирате **Tor** и по такъв начин, че да не насочва никога вашите заявки към указани от вас нежелателни държави – с подобно указание:

```
ExcludeExitNodes {cn},{kp}
StrictNodes 1
```

(където сме посочили примерното ни предпочитание последният пренасочващ възел никога да не бъде базиран в някоя от държавите **Китай** и **Северна Корея**).

Имайте предвид, че в някои по-малки държави (или такива със слабо развит или силно контролиран internet) не можете да разчитате на голям брой **Tor**-сървъри и е възможно вашите указания да доведат до драстично забавяне на трафика или до пълно блокиране на **Tor**. В такъв случай трябва да промените настройките, за да можете да ползвате софтуера. Също така, ако вместо стойност '**1**' в указанието '**`StrictNodes 1`**' посочите стойност '**0**', ще дадете възможност на **Tor** да прави изключения от указаните изисквания, когато тяхното стриктно спазване по някаква причина прави невъзможно или силно затруднява изпълнението на вашите заявки. Можете да се възползвате от тази възможност, когато не е абсолютно необходимо указанието ви за изходен възел в предпочитани (или нежелателни) държави да се спазва на всяка цена.

Възможно е да се окажете в ситуация, в която дори самият факт, че насочвате вашите заявки към **Tor**, крие рискове (например в условията на деспотичен политически режим или рестриктивна корпоративна политика). Тогава е желателно да прикриете **Tor** (и) от гледна точка на вашия internet-доставчик. За такива ситуации са предвидени т.нар. „мостове“ (**Bridges**). Това са сървъри с „невинни“ web-сайтове, които при определен начин на шифровано свързване с тях ви пренасочват към мрежата на **Tor**. Така, дори и вашата активност в internet да бъде проследена, вашият трафик на пръв поглед ще изглежда насочен към „невинния“ web-сайт, а не директно към анонимизиращата **Tor**-мрежа. Въпреки това трябва да бъдете особено внимателни при ползването на подобни анонимизиращи средства в ситуации, в които самото пускане на **Tor** може да бъде опасно за вас. В подобна ситуация не правете това, преди да сте се запознали подробно с цялостната документация на проекта (отделно от кратките въвеждащи бележки, до които се ограничаваме в настоящото изложение). И имайте предвид, че прикриването с „мостове“ не прикрива надеждно вашия трафик. Ако ползването на **Tor** е рисковано във вашата ситуация, прибягвайте до това колкото е възможно по-инцидентно, с възможно най-кратки сесии и не от места и точки за internet-достъп, които биха могли по някакъв начин да се свържат с вас; нито от устройства, които не са изцяло под вашия контрол (като за целта е най-добре да стартирате вашата операционна система от „жив“ носител и да съхранявате **Tor**-директорията на шифрован външен носител).

„Мостовете“ към **Tor**-мрежата изглеждат по подобен начин:

```
obfs4 52.4.159.235:9443 9F34C828CEA1D1AB5B882EA4BBF3B4144CA341CE
```

(където '**`obfs4`**' е ползваната технология (съществуват и други технологии, с които можете да се запознаете в случай на интерес и необходимост); '**`52.4.159.235`**' е **IP**-адресът на обслужващия сървър; '**`9443`**' е ползваният порт; и '**`9F34C828CEA1D1AB5B882EA4BBF3B4144CA341CE`**' е 'отпечатъкът' на съответния „мост“, служещ за шифровано свързване с „мостовия“ сървър и отключване на неговата скрита функция за насочване към **Tor**-мрежата).

Да насочите вашия **Tor**-браузър към „мост“ можете още при първото му стартиране или на по-късен етап, когато вече сте го конфигурирали и го ползвате.

Ако решите да пренасочите **Tor**-браузъра към „мост“ още при първото му пускане в действие, след като изберете файла за стартиране '**`Tor Browser Setup`**' и отваряне на диалоговия прозорец, трябва да изберете долната опция **Configure** (вместо горната **Connect**); на въпроса „Нуждае ли се компютърът от прокси за достъп до internet?“ (**Does this computer need to use a proxy to access the Internet?**) да отговорите съгласно вашия случай (като при отговор '**`Да`**' е необходимо да въведете данни за прокси); и да въведете отговор '**`Да`**' на следващия въпрос „Вашият internet-доставчик блокира ли / цензурира ли по друг начин връзките към **Tor**-мрежата?“ (**Does your Internet Service Provider (ISP) block or otherwise censor connections to the Tor Network?**). Тук имате две възможности: да изберете „Свържи се с предложени „мостове“ (**Connect with provided bridges**), при което ще можете да изберете вида

технология (**Transport type**) от падащото меню (дадената по подразбиране в повечето случаи е най-подходяща); или да изберете „Въведи ръчно „мостове“ (**Enter custom bridges**), при което ще трябва да въведете в текстовото поле един или няколко „моста“ (по един на ред), във формат като дадения по-горе пример; и накрая да натиснете '**Connect**'.

Можете да пренасочите **Tor**-браузъра към „мост“ и на по-късен етап, като активирате вече конфигурирания софтуер от файла за стартиране '**Tor Browser**' и инициирате неговото преконфигуриране. Това става от логото на **Tor** горе вляво на **Tor**-браузъра, откъдето трябва да изберете в падащото меню „Отвори настройки на мрежата...“ (**Open Network Settings...**) и да поставите отметка на най-долната опция „Моят internet-доставчик блокира връзките към **Tor**-мрежата“ (**My Internet Service Provider (ISP) blocks connections to the Tor Network**). При това ще се отвори описаният по-горе диалогов прозорец, на мястото, където трябва да изберете една от двете възможности – „Свържи се с предложени „мостове““ (**Connect with provided bridges**) или „Въведи ръчно „мостове““ (**Enter custom bridges**) – и да въведете съответните настройки в зависимост от вашите предпочитания, като накрая натиснете '**Connect**'.

В случай, че системата не може да се свърже успешно със зададените от вас „мостове“, пробвайте с други (възможно е зададените от вас да са били междувременно локализирани от противници на **Tor**-анонимизацията и да са били блокирани). Ако се нуждаете от нови „мостове“, можете да потърсите такива на този адрес:

`https://bridges.torproject.org/options`

Имайте предвид, че властите в държави с деспотични политически режими и системните администратори в организации с рестриктивни корпоративни политики издирват и блокират „мостовете“, до които успеят да достигнат. Ето защо е възможно да ви се налага да подновявате често своите „мостове“, а при някои особено чувствителни действия в internet е препоръчително да пренасочите вашия **Tor**-браузър към съвсем нов „мост“. В случай, че срещате трудности с набавянето на необходимите ви „мостове“, можете да заявите екипът на **Tor** да ви изпрати няколко, на този e-mail:

`bridges@bridges.torproject.org`

Единственото съдържание на съобщението ви трябва да бъде '**get bridges**' (нищо повече) и трябва да бъде изпратено от ваш e-mail акаунт в **Gmail**, **Riseup!** или **Yahoo!** – защото екипът на **Tor** е приел, че изброените са достатъчно популярни и същевременно затрудняващи в значителна степен създаването на поредици от изкуствени акаунти с единствената цел да се прихващат и блокират по-значителни количества от новосъздаваните „мостове“.

При ползване на **Tor** е необходимо да се лишите от някои удобства, с които сте свикнали в internet (освен ако не желаете да компрометирате своята анонимност). През **Tor**-браузъра няма да работят някои от по-екстравагантните функционалности на интересуващите ви web-сайтове. Причината за това е, че такива функционалности много често служат за проследяване на потребителите или директно за атакуване на техните системи – поради което те просто са изключени в **Tor**-браузъра (без значение, че заради това е възможно да не можете да преглеждате нормално интересуващото ви съдържание, да ви се отказва вписване в определени сайтове и т.н.).

Не инсталирайте никакви приставки към **Tor**-браузъра и не правете никакви допълнителни настройки по него – той е конфигуриран изцяло в интерес на вашата анонимност и подобни действия биха могли да го компрометират. Изключително внимавайте при свалянето на файлове! Ако свалите файл и го отворите (в зависимост от файла и от отварящата програма) е възможно компютърът ви да се свърже директно с посочен във файла сървър и да бъдат изпратени данни за вашето действително местонахождение и действителните параметри на системата ви. Изключение правят само простите текстови файлове, които могат да се отворят безопасно с обикновена програма за просто текстово съдържание (това не включва офис програмите и техните файлови формати).

Освен че анонимизира вашето поведение в internet, **Tor Browser** също така ви позволява да достъпвате т.нар. „тъмна мрежа“ (**Dark web**), която функционира на базата на „невидими с просто око“ .onion базирани web-ресурси. Специфично за **`.onion`** базираните web-ресурси е, че тяхното местоположение също е анонимизирано по начина, по който вие самите анонимизирате себе си чрез **Tor**. Това са отдалечени сървъри, достъпни само от **Tor**-мрежата и в голяма степен защитени срещу опитите на властите, частни разследващи институции и други заинтересувани страни да локализират тяхното физическо местонахождение и да разкрият кой стои зад тях. За съжаление това често се ползва

от престъпници, които чрез **.onion** технологии просто анонимизират своята престъпна дейност (търговия с наркотици и оръжие, детска порнография, трафик на хора, тероризъм). Извън това обаче **Tor**-мрежата ви позволява да ползвате **Dark web** и като място за търсене и споделяне на информация, за чието цензуриране деспотичните политически режими и рестриктивните корпоративни политики се сблъскват с редица технологични затруднения. Ето няколко **.onion** адреса, които можете да достъпите чрез **Tor**-браузъра и да започнете да опознавате **Darknet** от тях:

Търсещата машина **DuckDuckGo** за достъп до обичайни internet-ресурси:
duckduckgogg42xjoc72x3sjasowoarfbgcmvfimaftt6twagswzczad.onion

Търсещата машина **TORCH** за достъп до **.onion** internet-ресурси:
xmh57jrzrnw6insl.onion

Нецензурирана **.onion** алтернатива на **Wikipedia**:
zqktlwi4fecvo6ri.onion

Web-директория с тематично структурирани **.onion** internet-ресурси:
dirnxxdraygbifgc.onion

Както ще забележите, в **Dark web** съществуват голям брой **.onion** ресурси – web-сайтове, web-директории, търсещи машини, пощенски и такива за моментна комуникация сървъри, социални медии. Вие също можете да създавате и поддържате свои **.onion** ресурси, ако счетете това за необходимо – за която възможност можете да се информирате по-подробно от предложените по-горе адреси (достъпни единствено чрез вашия **Tor**-браузър).

И преди да завършим, няколко думи за поведението ви като цяло докато ползвате **Tor**. Този софтуер полага усилия да анонимизира вашето присъствие в internet, като „фалшифицира“ параметрите за вашата система; скрива от вашия internet-доставчик „накъде“ отивате, респективно – скрива от администраторите на достъпваните от вас internet-ресурси „откъде“ идвате; и ограничава тези функционалности, които по една или друга причина могат да се превърнат във вектор за проследяване и разкриване на данни за вас. Никоя от горните мерки обаче няма да има смисъл, ако към това не се добави и вашето разумно поведение в internet. На първо място имайте предвид, че **Tor** не е панацея и не предлага „абсолютно“ гарантирана анонимност. Не са изолирани случаите, при които са разкривани слабости в софтуера, способни да компрометират анонимността (и по всяка вероятност ползвани от трети страни за тази цел). Нищо не гарантира, че и в момента не съществуват такива слабости, които все още не са разкрити публично, но тихомълком се ползват от трети страни. Ето защо ви насърчаваме да прибягвате към **Tor** колкото е възможно по-инцидентно, с възможно най-кратки сесии. Свързвайте се от неочаквани места, където не сте се появявали преди, не е лесно да бъдете наблюдавани (например от охранителни камери) и скоро няма да се появите пак там. Местата е за предпочитане да са посещавани от всевъзможни хора и да е трудно да се изолира конкретно вашето посещение. Ползвайте контролирано от вас устройство, а ако имате особени опасения – стартирайте операционната система от „жив“ носител и съхранявайте **Tor**-директорията на шифрован външен носител. И не на последно място – не очаквайте да получите каквато и да било анонимност, ако през **Tor** влезете във вашите обичайни акаунти, отворите всекидневните си новинарски и развлекателни сайтове, и въобще продължите да се държите така, както сте се държали винаги. Поведението на потребителите в internet се наблюдава и моделира, и много скоро (ако някой се занимава с вашето целенасочено проследяване) биха се забелязали подозрителни сходства, които ще свържат активността на „неизвестния“ потребител с вас.

Пример в тази връзка е **Едуард Сноудън** (бивш служител на разузнаването на **САЩ**, направил публично достояние техните секретни дейности за повсеместно следене). В продължение на няколко месеца той копирал секретна информация на американските тайни служби и търсел журналисти, които да се заемат с нейното публично оповестяване. Докато работел, разграничавал стриктно своето поведение „като служител“ от това „като изобличител“. Един единствен път, докато ползвал публично достъпната internet-връзка от някакво кафене, проверил електронната си поща, която ползвал „като изобличител“. Тази поща отдавна се наблюдавала от службите, което му било известно, но продължавал да я използва, за да не разкрие какво му е известно и да ги направи още по-предпазливи. Тъй като спазвал безупречна технологична хигиена, те така или иначе нямало как да установят кой и откъде се свързва с пощата. Докато пиел кафето си, Сноудън променил както обикновено своята електронна идентичност на компютъра си и взел всичките необходими мерки, за да не бъде проследен. Тъкмо тази промяна обаче оставила съмнителна следа в

историята на трафика, осъществяван от клиентите на кафенето: определен компютър до някакъв момент се свързвал с internet под една идентичност, а малко след това – под друга. Това дало повод да се прегледат записите от видео-наблюдението в кафенето, където се виждало как служителят работи на своя компютър. Установило се и това, че тъкмо в същия момент е достъпена и оперативно интересната електронна поща. Професионалната подготовка на Сноудън му позволила веднага да си даде сметка за допуснатия пробив и незабавно да вземе мерките, които в крайна сметка го отведоха на летище „Шереметьево“ в Москва и не само спасиха живота му, но позволиха да осъществи поне част от планираните разкрития. Но така или иначе – всичко е тръгнало от това, че един компютър в кафенето в някакъв момент е сменил своята идентичност и тази „аномалия“ е провокирала специален интерес.

9. За ползването на търсещи машини

Както стана дума по-горе, търсещите машини служат за индексиране на съдържанието в internet и предлагане на релевантни спрямо потребителските заявки препратки към web-ресурси. Заедно с това обаче търсещите машини могат да индексират и самите потребители, като събират, анализират и изпращат информация за тях до трети страни. В действителност данните за това какво търсите и какво от предложените препратки избирате да достъпите могат да предоставят учудващо задълбочени детайли както за обществените, така и за личните ви наклонности, предпочитания и интереси, а от там – и за вашите устойчиви стереотипи и модели на поведение. Така – дори без да сте въвели каквато и да било „поверителна информация“, можете да се превърнете в обект на неподозирано задълбочено проучване и получените изводи да бъдат изпратени (или систематично изпращани) към трети страни. За да бъде ограничен този риск, се препоръчва да ползвате следните алтернативни на **Google** търсещи машини:

https://duckduckgo.com/
https://startpage.com/
https://www.ixquick.com/

Веднага трябва да отбележим две неща: първо, предложените алтернативи извеждат значително по-некачествени резултати от най-голямата търсеща машина в internet; и второ, не съществува (и няма как да съществува) сигурна гаранция за това, че алтернативните търсещи машини не събират, не анализират и не препращат информация за вас към трети страни.

Въпреки тези недостатъци обаче, ползването на алтернативни търсещи машини (доколкото това е възможно) би диверсифицирало вашите източници на информация, а така също – би сегментирало възможностите за събиране на информация за вашите интереси и поведение. Ако приложите подобна диверсификация и по отношение на ползваните от вас устройства, софтуерно обезпечение и точки на свързване към internet, бихте могли да си създадете няколко отделни „персоналности“, чрез които да осъществявате своите различни дейности.

За да бъдат усилията ви резултатни, трябва да спазвате известна дисциплина, като на първо място никога не смесвате отделните „персоналности“. Можете например да си създадете правило, че търсенето на чувствителна информация ще се осъществява само от алтернативни търсещи машини, устройства и точки за свързване с internet, които не ползвате в обичайното си ежедневие и за обичайните си дейности. Препоръчително е също така да сменяте периодично ползваните устройства, софтуерно обезпечение и точки за свързване. Така – дори и в някакъв момент дейността ви да предизвика интерес – след сменянето на „персоналността“ ще ограничите значително възможностите за продължаване на вашето пълно и всеобхватно проучване и профилиране.

VII. ОРГАНИЗАЦИЯ НА ФИЗИЧЕСКАТА СИГУРНОСТ

Дори вашият компютър да е инсталиран с напълно Свободна операционна система и свободен софтуер от „ниско“ ниво, които функционират безупречно, на изцяло шифрован твърд диск, с който работите при спазване на всичките възможни правила за безопасност в internet, хардуерът продължава да бъде уязвим физически.

1. Неразрешено физическо проникване в хардуера на компютъра

Възможно е корпусът на компютъра да бъде разглобен във ваше отсъствие (опитен специалист може да се справи с това буквално за минута) и определен хардуерен компонент да бъде подменен с друг или в системата да бъде вградено допълнително устройство за прихващане на вашите пароли или за препращане на трафика ви към трети страни. Възможно е също така след физическото достъпване на вашия flash-чип и препрограмирането му с компрометирана версия на **Libreboot**. Възможно е не на последно място **RAM**-чиповете да бъдат откачени от компютъра непосредствено след неговото изключване или докато е в режим на очакване и от тях да бъдат извлечени вашите шифровъчни ключове и пароли. Ето защо компютърът трябва да се пази далеч от достъпа на трети лица, а корпусът да бъде защитен със знаци, разкриващи евентуални опити за проникване.

Можете да запечатате винтовете за разглобяване на корпуса с втвърдяваща паста, нанесена над тях в гнездата им и белязана с някакви ваши знаци. Ако някой след това иска да разглоби корпуса, ще трябва да разбие пастата със знаците, преди да развие винтовете – и няма да е лесно да нанесе отново същия вид паста и да имитира знаците ви.

Можете също така да поставите лепенки като тези на гаранционното обслужване върху винтовете – така ще трябва да бъде нарушена целостта на лепенките, за да се разглоби корпусът на компютъра. Трябва обаче да имате предвид, че някои видове лепила могат да бъдат разлепени и залепени след това без нарушаване целостта на лепенките. Също така, ако ползвате широко разпространени лепенки, може вашите да бъдат подменени лесно с други от същия вид.

Препоръчваме да приложите няколко различни способа едновременно. Така, дори един от способите да бъде преодолян, ще бъде трудно да се преодолеят всичките заедно – особено ако времето на вашето отсъствие е кратко и трябва да се действа бързо. Можете например част от винтовете да запечатате с втвърдяваща паста и вашия знак, а друга част – с различни видове лепенки. За да сте сигурни, че лепенките не могат да бъдат лесно подменени с други от същия вид, можете да поставите знаци върху тях или да ги нараните по специфичен начин, който е трудно да бъде имитиран.

Специално внимание трябва да обърнете на достъпа до **RAM**-чиповете (които при моделите **x200** и **x200s** например са достъпни през отделен капак от долната част на корпуса). Както стана дума, тези чипове до няколко минути от изключването на компютъра продължават да съхраняват информация, критична за информационната сигурност. Желателно е в този критичен период да не може да се осъществи лесен достъп до тях – било поради блокиране отварянето на капака, било поради блокиране на самите чипове към дънната платка.

2. Подменяне на устройството с друго, което прилича на вашето

Знае се за случаи, при които компютърът бива откраднат и подменен с друго устройство, напълно изглеждащо като откраднатото; от жертвата се очаква да въведете паролата за разшифроване на системата (която е най-важната за обезпечаване на нейната сигурност) и тази парола незабавно да бъде изпратена от програмирания за тази цел подменен хардуер до авторите на атаката, за да я въведат в истински компютър (който вече се намира при тях). Ето защо трябва да внимавате дали това е действителното устройство, което очаквате – или само прилича на него. Препоръчва се да оставите някакви специфични знаци (трудни за забелязване и трудни за наподобяване) по вашето устройство, които да сверявате преди да започнете да въвеждате паролите си.

Допълнителна възможност е да поставите специфично изображение (идентични копия от което няма откъде да бъдат придобити) като стартово изображение във вашата версия на **Libreboot** софтуера от „ниско“ ниво. Можете да замените

вграденото във вашия **Libreboot**-пакет изображение, което се зарежда при стартирането системата (тъкмо преди да дойде ред за въвеждане на паролата за разшифроване).

Добра практика е сами да създадете това специфично изображение или да редактирате такова, което сте придобили отнякъде, с едва забележими, но ясни за вас детайли, които трудно биха били забелязани от трети страни, докато вие знаете къде са и ги виждате лесно. Така, ако някой се опита да имитира вашето изображение, е малко вероятно да имитира и едва забележимите детайли (освен ако не получи пряк достъп до вашата версия на **Libreboot**-пакета и извлече от него оригинала на вашето редактирано изображение). Дори и да бъде заснет екранът на вашия компютър с прецизна техника, ще бъде трудно да се повтори същото качество на стартовото ви изображение – ще забележите макар и малки недостатъци в имитацията – нюанс в цветовете, размазване на фокуса, изместване на позицията спрямо рамката на монитора.

Както в Част **IV** въведохте в **Libreboot** редактирана версия на **GRUB**, за да настроите зареждането на напълно шифрования твърдия диск, сега ще въведете и редактирано изображение на мястото на заложеното по подразбиране. Така ще забележите, ако вашата версия на **Libreboot**-пакета е подменена с нещо друго. За тази цел ви е необходима вашата версия на **Libreboot**-пакета, която по-горе редактирахте и препрограмирахте с нея flash-чипа.

Необходимо ви е също (ако още не сте) да инсталирате набора от софтуерни инструменти:

```
# pacman -S libreboot-utils flashrom
```

Първо извлечете изображението '**background.jpg**' от **Libreboot**-пакета:

```
# cbfstool Libreboot_пакет extract -n background.jpg -f background.jpg
```

Ще бъде изведено копие от стартовото изображение '**background.jpg**', което се зарежда при показване на стартовото меню в програмата **GRUB**, когато към компютъра се подаде електричество. Можете да отворите стартовото изображение '**background.jpg**' с предпочитана от вас програма за обработка на изображения (например **KolourPaint** или **GIMP**) и да го редактирате. Препоръчва се да въведете едва забележима редакция или да създадете изцяло ново изображение, като и в двата случая трябва да запазите неговия оригинален размер (който е **1280** x **800** при **x200** и **x200s** компютрите), оригиналното заглавие '**background**' и оригиналния **.jpg** формат.

За да можете да въведете редактираното изображение '**background.jpg**' обратно в **Libreboot**-пакета, трябва първо да изтриете старата версия на изображението '**background.jpg**' (която все още се намира в пакета):

```
# cbfstool Libreboot_пакет remove -n background.jpg
```

Ако всичко е наред, системата извежда индикатор '**Performing operation on 'COREBOOT' region...**'.

След това идва ред да въведете редактираното стартово изображение '**background.jpg**' в **Libreboot**-пакета:

```
# cbfstool Libreboot_пакет add -n background.jpg \
-f background.jpg -t raw
```

Системата отново извежда индикатор '**Performing operation on 'COREBOOT' region...**'.

Следва препрограмиране на flash-чипа с редактирания **Libreboot**-пакет. За целта при зареждане на **GRUB**-менюто (или, ако все още не сте осъществили цялостно шифроване на твърдия диск – след зареждане на второто **GRUB**-меню) трябва да натиснете **[e]** за да отворите режима за редактиране и да въведете в края на реда започващ с командата 'linux' допълнителния параметър:

```
iomem=relaxed
```

След като сторите това, натиснете **[Ctrl]+[x]** или **[F10]** (вижте поясненията на екрана най-отдолу) и продължите със стартирането на системата (която до следващото си рестартиране ще бъде в режим, позволяващ препрограмиране на flash-чипа).

Накрая идва ред да препрограмирате flash-чипа:

```
# flashrom -p internal -w Libreboot_пакет
```

Ако всичко е наред, системата ще препрограмира вашия flash-чип с редактирания **Libreboot**-пакет и ще изведе проследения в Част **III** резултат '**VERIFIED**'.

При следващо стартиране на системата вече ще видите на екран вашето специфично стартово изображение (вместо вграденото в **Libreboot** по подразбиране). Научете се с бърз поглед (и без очевидно втренчване) да разпознавате вашите едва забележими редакции, които отличават това изображение от всяко друго. За да забележите евентуална подмяна на изображението, трябва да познавате вашия оригинал достатъчно добре. И не на последно място – унищожете първообраза на изображението или го приберете на сигурно място.

3. Стартиране на операционната система от „жив“ носител

Добра практика за гарантиране неприкосновеността на вашата Свободна операционна система е да подготвите „жив“ носител, от който да стартирате системата (например **Live-USB** или **Live-CD/DVD**), когато желаете да сте напълно сигурни, че работите наистина с неманипулиран софтуер и, че няма да оставите следи от действията си в компютъра. **GNU/Linux-libre** позволява (както изяснихме по-горе) да работите с операционната система от „живия“ носител, без изобщо да я инсталирате на вашето устройство – като се зареди от „живия“ носител в **RAM**-чиповете на компютъра и работи чрез тях.

Недостатък при работа от „жив“ носител е, че системата работи малко по-бавно и се зарежда само с базовите настройки, които са заложени по подразбиране (можете да правите промени, но след изключване на „живия“ носител те ще бъдат „забравени“ изцяло и при следващо стартиране ще намерите операционната система обратно в нейния изходен вид). Ще бъдат „забравени“ и всичките файлове, които сте свалили или сте създали в системата (освен ако не ги запишете на външен носител).

Към настоящия етап Свободната операционна система **Heads**, както споменахме в Част **I**, все още е на етап **β**-тестване (основните функции работят, но несигурно и се отстраняват съществени слабости), поради което не можем да ви препоръчаме надеждна Свободна операционна система, която да е предвидено да работи от „жив“ носител и да запазва вашите настройки и файлове на съответния „жив“ носител. В следващо време се надяваме това да се промени и да можете да се доверите на **Heads** за важите „живи“ сесии.

Работата от „жив“ носител има и три много сериозни предимства по отношение на информационната сигурност. Първо, с всяко следващо стартиране на „живия“ носител имате гарантирано некомпрометирана операционна система – тъкмо във вида, в който тя е предоставена за ползване от екипа по нейната разработка. Второ, в стартирания чрез „жив“ носител компютър след неговото изключване не остава никакъв „спомен“ за това, което се е случило по време на „живата“ сесия (дори инсталираната на този компютър операционна система да е компрометирана, това няма да ви засегне – стига да не е компрометиран самият хардуер или софтуерът „от ниско ниво“). И трето, „живият“ носител (за разлика от един стандартен компютър) може да бъде много малък по обем (представете си например една **microSD**-карта), да се укрива и пренася незабелязано, и неговата неприкосновеност да се гарантира сравнително лесно (което е от особена полезност, когато не можете да гарантирате физическата сигурност на цял компютър).

4. Защита срещу неразрешено стартиране на компютъра

Както вече проследихме, веднъж препрограмиран с **Libreboot**, flash-чипът може да продължи да бъде препрограмиран направо от терминала на компютъра, без да е необходимо отново физическото му достъпване (освен ако

възможността за препрограмиране не е блокирана). Достатъчно е да бъде наличен необходимият софтуер (който е публично достъпен), да бъде осъществен достъп до системата с администраторски права и да се въведат съответните команди. Трябва също да имате предвид, че хардуерът не прави разлика между вашата операционна система и коя да е друга операционна система, стартирана на него. Това дава възможност трета страна да препрограмира flash-чипа на вашия компютър с компрометирана версия на **Libreboot** или да добави допълнителен компрометиращ софтуер, като просто стартира системата от свой „жив“ носител. Както е известно, „живият“ носител предоставя администраторски достъп по подразбиране. Така, след като „живият“ носител бъде стартиран и поеме контрол върху компютъра, ще може без затруднения да се извърши препрограмиране на flash-чипа от когото и да било (стига да има няколко минути физически достъп до системата).

За да ограничите този риск, трябва на първо място да блокирате възможностите компютърът да се стартира от „живи“ носители. Както е известно, „живи“ носители могат да се закачат към входящия интерфейс на компютъра (**USB**-портовете, тези за **SD**-карти, **CD/DVD**-устройствата; **LAN**-устройството и т.н.) и след това да се укаже компютърът да стартира (**Boot**) от тях (както направихте и вие в Част **II**, за да инсталирате Свободна операционна система). Следователно софтуерът от „ниско“ ниво (който управлява пряко хардуера и разрешава или отказва достъп до различните му функции) трябва да бъде настроен така, че да не разреши стартирането от „живи“ носители – поне не без ваше съгласие.

За да изключите възможността компютърът да бъде стартиран от външни „живи“ носители (освен с ваше съгласие), трябва още веднъж да извлечете тестовия файл '**grubtest.cfg**':

```
$ cbfstool Libreboot_пакет extract -n grubtest.cfg -f grubtest.cfg
```

Както вече проследихме в Част **IV**, в тестовия файл '**grubtest.cfg**' съществуват инструкции (към които в Част **IV** добавихме още една), всичките започващи с кода '**menuentry**' и продължаващи с редове от стартовото меню на **Libreboot**, подобни на този:

```
menuentry 'Search ISOLINUX menu (USB) [u]' --hotkey='u' {
      search_isolinux usb
}
```

От горния пример научаваме, че става дума за една от възможностите в стартовото меню на **Libreboot**, указваща системата да потърси операционна система в закачена към компютъра **USB**-памет и да стартира от нея, като се предлага и бърз бутон (**Hot Key**) за стартирането на тази възможност – **[u]**.

Оригинално възможностите в стартовото меню на **Libreboot**-пакета съответстват на '**menuentry**' командите и може да изглеждат така:

```
Load Operating System (incl. fully encrypted disks)  [o]
Search ISOLINUX menu (AHCI)  [a]
Search ISOLINUX menu (USB)  [u]
Search ISOLINUX menu (CD/DVD)  [d]
Load test configuration (grubtest.cfg) inside of CBFS  [t]
Search for GRUB2 configuration on external media  [s]
Load SeaBIOS (payload) [b]
Poweroff  [p]
Reboot  [r]
```

(първата от които указва компютърът да стартира нормално операционната система от вашия твърд диск **[o]**; втората, третата и четвъртата указват да бъде потърсен „жив“ носител (съответно външен твърд диск **[a]**, **USB**-памет **[u]** или оптичен диск **[d]**); петата и шестата указват зареждането на тестова версия на стартиращата програма **[t]** или на такава от външен носител **[s]**; седмата указва да бъде стартирана свободната **SeaBIOS** програма **[b]**; осмата и деветата указват компютърът да се изключи **[p]** или да се рестартира **[r]**); и десетата **[?]** (с избран от вас бърз

бутон), която създадохме в Част **IV** (и която може да бъде позиционирана на който ред предпочитаме в стартовото меню) указва на системата да стартира вашия изцяло шифрован твърд диск.

За всяка от наличните възможности в стартовото меню можете да въведете правило да се изисква специална парола, когато се прави опит съответната възможност да бъде избрана. Можете също така да забраните изпълняването на възможности, които не желаете изобщо да могат да се изпълняват на вашия компютър.

В случай, че желаете една или няколко от възможностите в стартиращото меню да бъдат достъпни след въвеждането на специална парола, трябва да въведете тази парола във файла '**grubtest.cfg**'. Можете да направите това във вид на прост текст – но тогава има риск паролата да бъде разкрита, ако по някакъв начин файлът бъде извлечен от **Libreboot**-пакета. Поради тази причина се препоръчва паролата да бъде хеширана, преди да се запише във файла '**grubtest.cfg**'. Това означава, че вместо паролата във вид на прост текст, ще бъде записана една сложна криптографска производна от нея. И когато на следващ етап въведете съответната парола, от нея ще бъде генерирана по същия алгоритъм сложна криптографска производна, като първоначалната и след това генерираната производни ще бъдат сравнени, за да се установи дали сте въвели правилната парола. Така, дори и хешираната производна да бъде извлечена, това ще бъде безполезно, защото не дава информация за първоначалната поредица от символи, чието хеширане би дало такава криптографска производна. А да се въведе направо криптографската производна вместо паролата също е безполезно, тъй като системата ще извърши отново хеширане и едва тогава ще направи исканата съпоставка, но с новополучената производна.

За да хеширате парола за достъпване на съответните възможности от стартиращото меню, трябва най-напред да инсталирате програмата **GRUB** чрез познатата команда '**pacman -S**'. Можете след това да хеширате самата парола:

```
$ grub-mkpasswd-pbkdf2 >> grubtest.cfg
```

При изпълнение на горната команда системата ще влезе в режим на очакване да въведете желаната от вас парола. Когато завършите въвеждането и натиснете **[Enter]**, системата няма да изведе никакъв индикатор; трябва да въведете повторно паролата и отново да натиснете **[Enter]**, за да постигнете търсения резултат.

Сигурността на тази парола трябва да бъде от най-високо ниво – като паролата за цялостно шифроване на системата. Това е така, на първо място защото стартирането на компютъра от външен носител позволява осъществяването на достъп с администраторски права, а това от своя страна позволява препрограмиране на flash-чипа – независимо, че самата система продължава да стои напълно шифрована. На второ място – тази парола не може да бъде променена по друг начин, освен чрез последващо препрограмиране на flash-чипа. За да защитим обаче срещу евентуални опити за манипулация на софтуера от „ниско“ ниво, малко по-долу ще блокираме възможностите за последващо препрограмиране. Следователно, за да промените паролата, ще трябва да премахнете блокирането (което включва и физическо достъпване на flash-чипа, и е неудобно да се прави често). Поради това е желателно да създадете достатъчно дълга и сложна парола за разрешаване стартирането от „жив“ носител, която няма да ви се налага да сменяте дълго време напред.

След като въведете паролата и натиснете **[Enter]**, в изпълнение на горната команда паролата ще бъде хеширана до необходимата сложна криптографска производна и последната ще бъде изведена най-отдолу във файла '**grubtest.cfg**', в подобен вид:

```
Enter password:
Reenter password:
PBKDF2 hash of your password is grub.pbkdf2.sha512.10000.0E38292AA24D
115AE3F40F88E82AB9C61FDD341DFA114417594128C5D1B6A0450C0FCC35D263D4432
F9250B9E5759C0519D02A332CC2F01AF5467E77F36AC893.8B2DA429438C59530A148
D8FFF5DFA9CCD875FB61609425CF8E20E21A01FE27F338A93E03FAE813D9EF4CE09D4
1C57191F87B73C138D25283A965CF22CDFA055
```

Дългата поредица от буквено-цифрени символи, която по-горе сме разположили за прегледност на няколко реда, оригинално се генерира само на един единствен дълъг ред и трябва да остане в оригиналния си вид (не я прекъсвайте

с интервали и не я пренасяйте на няколко реда. От цитирания запис трябва внимателно да отстраните встъпителната част:

```
Enter password:
Reenter password:
PBKDF2 hash of your password is
```

... и на нейно място (на редовете над самата парола) трябва да въведете следния код:

```
set superusers=""
password_pbkdf2 Потребител
```

(където под '**Потребител**' разбираме наименованието на вашия потребител). Под въведения съгласно горното указание код трябва да оставите самата производна, започваща с '**grub.pbkdf2.sha512.10000.**' и продължаваща с дългата буквено-цифрена комбинация, представляваща вашата хеширана парола и разположена на един единствен ред. Вашата редакция трябва да имате подобен вид:

```
set superusers=""
password_pbkdf2 Потребител grub.pbkdf2.sha512.10000.0E38292AA24D115AE
3F40F88E82AB9C61FDD341DFA114417594128C5D1B6A0450C0FCC35D263D4432F9250
B9E5759C0519D02A332CC2F01AF5467E77F36AC893.8B2DA429438C59530A148D8FFF
5DFA9CCD875FB61609425CF8E20E21A01FE27F338A93E03FAE813D9EF4CE09D41C571
91F87B73C138D25283A965CF22CDFA055
```

(Представеният от нас пример на хеширана парола е разположен за прегледност на няколко реда, но в действителност кодът ще бъде разположен на един единствен дълъг ред и вие трябва да го оставите в този му вид. В случая не може да бъде приложено правилото за графично пренасяне на нов ред посредством символа '\'.)

След като вашата хеширана парола бъде въведена във файла '**grubtest.cfg**' по описания начин, изпълнението на коя да е от предлаганите в стартовото меню възможности ще стане невъзможно, освен ако не укажете изрично съответната възможност да се изпълнява след въвеждането на вашата парола или да е достъпно свободно, без каквато и да било парола (както беше досега).

За да укажете една възможност от стартовото менюто да се изпълнява след въвеждане на вашата парола, трябва да добавите кода '**--users Потребител**' непосредствено след указанието за бързия клавиш и преди отварящата фигурна скоба в съответния ред, започващ с '**menuentry**'. Така например кодът за възможността компютърът да бъде стартиран от „жив“ носител:

```
menuentry 'Search ISOLINUX menu (USB) [u]' --hotkey='u' {
      search_isolinux usb
}
```

... ще придобие следния вид:

```
menuentry 'Search ISOLINUX menu (USB) [u]' --hotkey='u' --users Потребител {
      search_isolinux usb
}
```

(където под '**Потребител**' разбираме наименованието на потребителя, от чийто акаунт ще бъде възможно избирането на съответната възможност (препоръчваме това да бъде потребителят без администраторски права).

За да укажете една възможност от стартовото менюто да се изпълнява свободно, без никаква парола, трябва да добавите кода '**--unrestricted**' непосредствено след указанието за бързия бутон и преди отварящата фигурна

скоба в съответния ред, започващ с '**menuentry**'. Така например кодът за възможността компютърът да бъде рестартиран:

```
menuentry 'Reboot [r]' --hotkey='r' {
      reboot
}
```

... ще придобие следния вид:

```
menuentry 'Reboot [r]' --hotkey='r' --unrestricted {
      reboot
}
```

Препоръчва се да оставите достъпни за свободно изпълнение възможностите за рестартиране **[r]** и изключване **[p]** на компютъра, а така също – въведената от вас **9-та** възможност **[?]** (с избран от вас бърз бутон) за стартиране на напълно шифрования твърд диск посредством изнесената на flash-чипа стартираща програма **GRUB**. Тези три възможности не могат да застрашат вашата информационна сигурност, доколкото първите две просто преустановяват или подновяват захранването, а последната води единствено до това да бъде изискана парола за разшифроване на твърдия диск. Да въведете правилото за стартиране на тези три възможности чрез вашата специална парола би било проява на излишна предпазливост, която ще отслаби информационната ви сигурност, тъй като ще доведе до редовно (и ненужно) въвеждане на парола, чиято основна функция е да защитава компютъра срещу неразрешено стартиране от „жив“ носител. Не се препоръчва да правите това!

Заедно с това се препоръчва да въведете правило за стартиране с вашата хеширана парола освен при стартиране от **USB**-памет **[u]**, също и при стартиране от тестовата версия **[t]** на **GRUB**. Второто е важно в случай, че настъпи някакво объркване, в следствие на което стартирането от току що редактирания файл се окаже невъзможно и се наложи да отстранявате настъпилите грешки чрез стартиране на системата от другия файл на **GRUB**. Не по-малко важно е също така да въведете правило за стартиране с вашата хеширана парола при стартиране от външен носител **[s]** и при стартиране от свободната **SeaBIOS** програма **[b]** (която от своя страна също позволява стартиране от външен носител).

Всички останали възможности от стартиращото меню (за които не сте въвели нито правилото да изпълняване с парола (посредством кода '**--users Потребител**'), нито правилото за свободно изпълняване без парола (посредством кода '**--unrestricted**'), ще останат недостъпни. В случай, че някоя от тях бъде избрана, системата ще изведе указание '**login:**' в очакване да бъде посочен потребител и след това ще изведе указание '**password:**' в очакване да бъде въведена парола – но каквото и да бъде въвеждано, резултатът ще бъде отказ да се изпълни съответната възможност.

След въвеждането на горните редакции трябва да изтриете стария файл '**grubtest.cfg**' от **Libreboot**-пакета:

```
$ cbfstool Libreboot_пакет remove -n grubtest.cfg
```

... да въведете в **Libreboot**-пакета току що редактирания файл '**grubtest.cfg**':

```
$ cbfstool Libreboot_пакет add -n grubtest.cfg -f grubtest.cfg -t raw
```

... и накрая да препрограмирате flash-чипа с току що променената версия на **Libreboot**:

```
# flashrom -p internal -w Libreboot_пакет
```

(за която цел трябва при зареждане на **GRUB**-менюто (или, ако все още не сте осъществили цялостно шифроване на твърдия диск – след зареждане на второто **GRUB**-меню) чрез натискане на **[e]** да сте отворили режима за редактиране на **GRUB** и да сте въвели в края на реда '**linux**' отдолу параметъра '**iomem=relaxed**').

Когато приключите с горните манипулации (успешното препрограмиране на flash-чипа ще бъде обозначено с извеждането на резултат '**VERIFIED**'), можете да рестартирате системата, да изберете при зареждането на **GRUB**-менюто стартиране на тестова версия (като натиснете **[t]**), за да превключите към редактирания преди малко файл '**grubtest.cfg**' и да проверите една по една съществуващите възможности.

Когато се убедите, че въведените в '**grubtest.cfg**' настройки работят правилно, можете да копирате редакциите от файла '**grubtest.cfg**' в основния файл '**grub.cfg**' (като не забравяте, че възможността за тестване **[t]** в основния файл '**grub.cfg**' трябва да сочи към тестовия файл '**grubtest.cfg**'). След като сторите това, остава да въведете редактирания основен файл '**grub.cfg**' в **Libreboot**-пакета и за пореден път препрограмирате flash-чипа. Преди да приключите с работата си, не забравяйте да тествате отново настроените възможности (този път от основния файл '**grub.cfg**'). В случай, че нещо се объркa, вече разполагате със сигурна възможност за стартиране на системата в лицето на тестовия файл '**grubtest.cfg**' и ще можете да се върнете към него (като изберете от основното стартово меню тестване **[t]** и след като се отвори тестовото стартово меню – необходимата ви, вече тествана и работеща възможност).

5. Защита срещу неразрешено препрограмиране на flash-чипа

Както вече проследихме, софтуерът от „ниско“ ниво е от критично значение за сигурността на системата. Ето защо е важно да препятствате всякакви възможности flash-чипът да бъде препрограмиран с компрометирана версия на **Libreboot** или към него да бъде добавен допълнителен компрометиращ софтуер. Нагоре проследихме препрограмирането на **BIOS/UEFI** flash-чипа с **Libreboot** – което нямаше как да се осъществи, освен ако не получите физически достъп до пиновете на flash-чипа, за да ги свържете с микроконтролер (с едноплатков компютър) и да подадете към тях необходимите инструкции за препрограмиране. След като вече е препрограмиран с **Libreboot** обаче, за последващото препрограмиране на flash-чипа не е необходимо повторно физическо достъпване на неговите пинове – достатъчно е да бъде осъществен достъп с администраторски права от терминала на коя да е **GNU/Linux** операционна система.

Тук е важно да отбележим, че flash-чипът не може да направи разграничение между „вашата“ операционна система и коя да е „друга“ операционна система. Следователно, ако се закачи „жив“ носител с чужда операционна система и тя бъде стартирана в административен режим (който при такова стартиране се предоставя по подразбиране), може да се осъществи препрограмиране на flash-чипа. За да се ограничи този риск, възможностите за по-нататъшно препрограмиране на flash-чипа е добре да бъдат блокирани физически.

По-горе разгледахме възможността за създаване на правило стартирането от „живи“ носители да бъде ограничено с парола. Това ограничаване обаче е само софтуерно – теоретично е възможно да се преодолее и flash-чипът все пак да бъде препрограмиран. Подобен пробив би бил особено опасен предвид критичното значение на **Libreboot** за сигурността на системата. Ето защо – за да можете наистина да вярвате, че препрограмиране на flash-чипа не може да бъде осъществено, е необходимо да блокирате тази възможност и физически. Така – поне докато не бъде нарушена целостта на запечатването на корпуса – ще можете да бъдете наистина сигурни, че **Libreboot** работи така, както очаквате – без допълнителна намеса.

Блокирането на възможностите за по-нататъшно препрограмиране на flash-чипа включва няколко софтуерни и една хардуерна манипулация. За да осъществите софтуерните манипулации, трябва да компилирате **Chromium** версията на програмата **Flashrom** (която е различна от версията на програмата **Flashrom**, ползвана до момента). Изходният код на **Chromium** версията на програмата **Flashrom** е достъпен от този адрес:

https://chromium.googlesource.com/chromiumos/third_party/flashrom

За да можете да пристъпите към необходимото компилиране, трябва да инсталирате програмите **Git**, **GCC**, **Pkg-config** и **Make**:

```
# pacman -S git gcc pkg-config make
```

Сега вече можете да пристъпите към изтеглянето и компилирането на изходния код до изпълним софтуер. Изтеглянето ще доведе до създаването на нова директория '**flashrom**', която ще се позиционира в текущата ви директория. Позиционирайте терминала където искате да бъде работите и пуснете:

```
$ git clone \
https://chromium.googlesource.com/chromiumos/third_party/flashrom
```

При изпълнението на горната команда ще бъде създадена споменатата по-горе директория '**flashrom**' и в нея ще бъдат изтеглени файловете с изходния код на **Chromium** версията на програмата **Flashrom**. След като директорията '**flashrom**' бъде създадена и в нея се зареди съответното съдържание, трябва да преместите терминала в тази директория и да въведете следната команда:

```
$ git checkout 91f320eaab5d7fdd68980b3dbeb64fd30f47aad5
```

... и след това:

```
$ make WARNERROR=no
```

При изпълнението на горната команда ще се осъществи компилиране на изходния код на **Chromium** версията на програмата **Flashrom** до машинен код, който може да се изпълнява като софтуер.

Сега можете да проверите състоянието на вашия flash-чип посредством току що компилираната версия на **Flashrom**:

```
# ./flashrom --wp-status
```

(където под '**./flashrom**' разбираме току що компилираната Chromium версията на програмата **Flashrom**, а не програмата **Flashrom**, която инсталираме чрез '**pacman -S flashrom**' и прилагаме на други места в изложението). Ако при стартиране на системата не сте конфигурирали **GRUB** за стартиране с параметър '**iomem=relaxed**', изпълнението на горната команда ще ви бъде отказано с извеждането на резултат, че операцията не е разрешена (**Operation not permitted**). Ако сте отворили сте стартирали с параметър '**iomem=relaxed**', ще бъде изведен подобен резултат:

```
flashrom v0.9.9 chromium.googlesource.com/chromiumos/third_party/flashrom :
91f320e : Jul 25 2018 07:14:47 UTC on Linux 4.17.6-gnu-1 (x86_64)
coreboot table found at 0xbfad6000.
WP: status: 0x0000
WP: status.srp0: 0
WP: status.srp1: 0
WP: write protect is disabled.
WP: write protect range: start=0x00000000, len=0x00000000
```

(където изразът '**write protect is disabled**' указва, че защитата срещу препрограмиране е изключена и съответно зоните на flash-чипа, които би трябвало да са защитени от препрограмиране, започват (**start=**) от **0x00000000** и продължават (**len=**) отново до **0x00000000**; т.е. – няма такива зони).

Вече идва ред да укажете на системата софтуерно да блокира по-нататъшното препрограмиране на flash-чипа:

```
# ./flashrom --wp-range 0 0xКапацитет
```

(където под '**Капацитет**' разбираме някое от числата '**400000**', '**800000**' или '**1000000**' – съответно при flash-чип с капацитет **4MiB**, **8MiB** или **16MiB**);

... и след това:

```
# ./flashrom --wp-enable
```

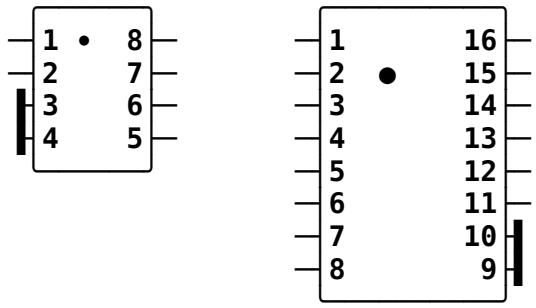
Сега можете да проверите отново състоянието на вашия flash-чип:

```
# ./flashrom --wp-status
```

Ако всичко е наред, ще бъде изведен подобен резултат:

```
flashrom v0.9.9 chromium.googlesource.com/chromiumos/third_party/flashrom :
91f320e : Jul 25 2018 07:14:47 UTC on Linux 4.17.6-gnu-1 (x86_64)
coreboot table found at 0xbfad6000.
sh: dmidecode: command not found
dmidecode execution unsuccessful - continuing without DMI info
WP: status: 0x009c
WP: status.srp0: 1
WP: status.srp1: 0
WP: write protect is enabled.
WP: write protect range: start=0x00000000, len=0x00800000
```

След като получите търсения резултат ('**write protect is enabled**'), идва ред и на хардуерното блокиране по-нататъшното препрограмиране на flash-чипа. В противен случай при софтуерно подаване на команда чрез програмата **Flashrom** при осъществен административен достъп до системата, току що въведеното софтуерно блокиране на по-нататъшното препрограмиране ще бъде отменено и препрограмиране все пак ще се извърши. За да блокирате хардуерно по-нататъшното препрограмиране, трябва да разглобите компютъра, да достъпите физически flash-чипа и да споите пинове **3** и **4** (при flash-чипове с **8** пина); респективно пинове **9** и **10** (при flash-чипове с **16** пина).



Спояване на пиновете при BIOS/UEFI flash-чипа (8 и 16 пина)

Можете да осъществите необходимото спояване, като нанесете върху интересуващите ви пинове малко флюс-паста, нагреете поялника до температура около **375°C** и свържете интересуващите ви пинове един към друг с нанасянето на малко тинол. Докато указаната двойка пинове на flash-чипа е свързана хардуерно с така описаната електропроводима връзка, по-нататъшно препрограмиране няма да може да се осъществява. За да се препрограмира отново flash-чипът, трябва преди това електропроводимата връзка да бъде отстранена физически.

Преди да приключите своята работа и да сглобите корпуса на компютъра, е добре да тествате дали наистина блокирането на по-нататъшното препрограмиране е осъществено успешно. За тази цел можете да направите първоначално тестово копие на информацията, записана на flash-чипа:

```
# flashrom -p internal -r Първоначално_тестово_копие.rom
```

... и след това да направите опит за препрограмиране на flash-чипа:

```
# dd if=/dev/zero of=zero.rom bs=8M count=1
```

... и още веднъж, по друг начин:

```
# flashrom -p internal -w zero.rom
```

В случай, че блокирането на по-нататъшното препрограмиране е въведено успешно, системата ще изведе подобен резултат: ТРИЕНЕТО ПРОВАЛЕНО (**ERASE FAILED!**) и няма да пристъпи към последващо записване на фиктивните файлове **zero.rom**, които се опитахме да въведем в паметта на flash-чипа.

Независимо от този резултат се препоръчва да направите последващо тестово копие на информацията, записана на вашия flash-чип:

```
# flashrom -p internal -r Последващо_тестово_копие.rom
```

След като вече разполагате с тестови копия от момента преди и от момента след опита ви да препрограмирате flash-чипа, за да сте напълно сигурни в постигнатите резултати, остава да сравните двете копия – ако са напълно еднакви, означава, че никаква част от опитаното препрограмиране не се е осъществила и блокирането работи напълно. Можете да направите сравняването така:

```
$ cmp Първоначално_тестово_копие.rom Последващо_тестово_копие.rom
```

... и след това:

```
$ md5sum Първоначално_тестово_копие.rom Последващо_тестово_копие.rom
```

Системата ще изведе 'проверовъчните суми' на двата файла, удобно представени една под друга. Ако е налице пълно тъждество, означава, че първоначалното и последващото тестово копие са напълно еднакви – следователно възможностите за препрограмиране на flash-чипа са наистина блокирани. Вече можете да сглобите компютъра, като запечатате корпуса му срещу евентуални опити за неразрешено проникване. Оттук нататък остава да проверявате редовно автентичността на устройството, целостта на запечатването на корпуса и оригиналността на вашето специфично стартово изображение. И ако някой от елементите за установяване на автентичност и интегритет бъде нарушен, това ще бъде ясен сигнал да вземете мерки за ограничаване последиците от настъпилия пробив и за осъществяване при необходимост на миграция към други, сигурни устройства.

6. Защита срещу неразрешено закачане към USB-портовете

Изключително опасен вектор на рискове представляват **USB**-портовете на вашия компютър. Специфичното техническо предимство на този вид портове – да разпознават типа закачени към компютъра устройства (и в много от случаите – да ги монтират автоматично към системата) – разкрива възможност чрез такива устройства да бъде направен опит за неразрешено включване към системата и извършване в нея на определени компрометиращи действия (например – въвеждане на команди в терминала, когато е отворен в административен режим). Подобна атака е възможна със специално подготвени устройства (най-често **USB**-памет или дори обикновена оптична мишка), при които във фабрично предвидените микроконтролери освен очакваните функции е добавен допълнителен програмен код. Такива добавени програми могат след закачането на устройството към системата (например за да свалите интересуващ ви

файл или просто за да ползвате една обикновена мишка) да направят опит да подадат компрометиращи команди към системата през ползвания **USB**-порт. Подобни компрометиращи команди могат да бъдат осъществени с отлагане (за да се притъпи вашата първоначална бдителност) и могат да бъдат извършени с много висока скорост (правещо почти невъзможно забелязването им, дори и ако в този момент наблюдавате дисплея).

Макар да не е особено практично, можете да стартирате вашата напълно шифрована система, като още от софтуера на „ниско“ ниво зададете изключване на **USB**-портовете. За тази цел можете да заредите системата ръчно от **Libreboot**, като добавите към указанията за зареждане на системното ядро параметъра '**nousb**' – по този начин:

```
> linux /boot/vmlinuz-linux-libre-lts root=/dev/mapper/Група-Системен_дял rw
nousb
```

Ако стартирате по този начин системата, до приключване на текущата сесия **USB**-портовете няма да бъдат активни. Ако желаете отново да ги ползвате, трябва да рестартирате компютъра и при повторното му стартиране да не въвеждате посочения параметър.

Можете да проверите какви устройства са закачени към USB-портовете на системата (трябва да имате инсталиран софтуерния пакет '**usbutils**'):

```
$ lsusb
```

... при което ще бъде изведен подобен резултат:

```
Bus 002 Device 014: ID 346d:5678 USB Disk 2.0
Bus 002 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 008 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 007 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 006 Device 008: ID 1241:1177 Belkin Mouse [HT82M21A]
Bus 006 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 005 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 004 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 003 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
```

(където в дадения пример виждаме закачена **USB**-памет на ред **1** и оптична мишка на ред **5**). Останалите редове показват **USB**-шините на компютъра, върху които могат да се закачат **USB**-устройствата. Част от тези **USB**-шини са достъпни само за вътрешни устройства (като напр. **Bluetooth**-устройство, web-камера, четци за пръстови отпечатъци, и други).

Ако не сте изключили напълно **USB**-портовете на вашия компютър, се препоръчва да проверявате регулярно с проследената команда '**lsusb**' и да сте наясно какви устройства нормално се закачат към системата (включително какви са устройствата, които вие лично нормално закачате към системата). Трябва в тази връзка да имате предвид, че е възможно едно устройство (включително такова, което ползвате редовно) във ваше отсъствие да бъде компрометирано и паралелно с обичайните си функции да направи опит за осъществяването на неразрешени действия в системата. Възможно е също да бъдат мониторани хардуерни „имплант“ във ваше отсъствие и в самите USB-портове – по начин, който не се забелязва при обикновен оглед и не препятства обичайното ползване на компрометираните портове. В такива случаи е от особена важност да забележите необичайните устройства, които системата установява като закачени към портовете – и това да бъде сигнал за необходимостта да предприемете съответните защитни мерки. Следва във връзка с горното да имате предвид, че такъв имплант може да е настроен да се активира само в един кратък промеждутък – и в останалото време да не бъде извеждан в списъците).

Интерес във връзка с казаното дотук представлява колона **6** от дадения пример, където първата половина от **8**-символните буквено-цифрени кодове (например '**346d**') указва конкретния производител; а втората половина (например '**5678**') указва конкретния модел на съответното закачено устройство). Понятно, кодовете '**1d6b:0001**' и

'**1d6b:0002**' се отнасят до виртуални **USB** устройствата **GNU/Linux-libre** системата. Пълен списък с идентификаторите на познатите и документирани устройства е достъпен на този адрес:

http://www.linux-usb.org/usb.ids

Независимо от наличието на такива идентификационни кодове обаче, следва да имате предвид, че тези кодове могат да бъдат променяни – така едно устройство може „да се представи“ за друго устройство в опит да бъдете подведени.

Във връзка с горното е добра практика да блокирате включването на каквито и да било устройства от вашите **USB**-портове и когато все пак желаете да ползвате някакво устройство, да укажете изрично неговото разрешаване. За тази цел можете да инсталирате програмата **USBguard**:

 # pacman -S usbguard

Следва да въведете базови настройки:

 # usbguard generate-policy > /etc/usbguard/rules.conf

(като имате предвид, че закачените към момента на изпълнение на горната команда устройства към вашите **USB**-портове ще бъдат въведени като „разрешени“ по подразбиране за ползване от вашата система – и няма да бъдат блокирани при тяхното последващо закачане. Следователно, ако не желаете да имате такива разрешени по подразбиране устройства – е необходимо да ги разкачите преди въвеждането на горната команда.

За да работи програмата, трябва да я активирате:

 # systemctl enable --now usbguard

Ако закачите устройства към някой от **USB**-портовете след активирането, това устройство няма да бъде закачено към системата (освен ако не е било закачено и по време на първоначалното настройване на програмата).

Можете да проследите какви устройства са закачени към вашите **USB**-портове към настоящия момент и същевременно са блокирани:

 # usbguard list-devices -b

Ако в момента има закачени блокирани устройства, ще бъде изведен подобен резултат:

```
12: block id 346d:5678 serial "45509511022382698 37" name "Disk 2.0" ha
sh "3tFZGFEHLIRm0fo61YQ+at6PmWePvwruxz5hXoN3QG4=" parent-hash "9JszuPW
ESBEur6P34acG7ZtBUqRa6fRgdsma6hSNpOk=" via-port "2-2" with-interface 0
8:06:50 with-connect-type "unknown"
13: block id 1241:1177 serial "" name "USB Optical Mouse" hash "pCr9UO
nNvk9W518fD66XxU1903xYdjjbB8oWXjkf6OE=" parent-hash "7/Y7ayZhucdn9UUOS
c/idMgOqIKmfZucV47nbxk0OLo=" via-port "3-1" with-interface 03:01:02 wi
th-connect-type "unknown"
```

(откъдето е видно, че под №**12** има закачена **USB**-памет, а под №**13** има закачена оптична мишка – и двете блокирани, съответно неизползваеми).

Можете да дадете разрешение конкретно закачено устройство да бъде ползвано чрез **USB**-портовете на системата:

```
# usbguard allow-device 13
```

(където под '**13**' в дадения пример разбираме номера, под който системата идентифицира съответното блокирано устройство – в случая оптична мишка с идентификатор '**1241:1177**'. Следва да имате предвид, че хешираната криптографска производна на устройство от определена марка и модел е идентична с тази на друго устройство от същата марка и модел. Не е излишно в тази връзка да помислите за маркиране корпусите на устройствата, които ползвате – за да ограничите риска от подхвърляне на компрометирано устройство от същата марка и модел като вашето, което се очаква да включите към системата и то да извърши вменените му „допълнителни“ функции.

След като разкачите устройство, при последващото му закачане системата отново ще го блокира. Ако желаете да го ползвате отново, трябва отново да проследите под какъв номер се разпознава и да го разрешите повторно.

Можете обаче да въведете това устройство като „разрешено“ по подразбиране и то да не бъде блокирано при последващо закачане (например оптичната мишка, която ползвате всеки ден и не желаете при всяко следващо закачане да разрешавате ръчно):

```
# usbguard allow-device -p 13
```

Можете да прегледате всичките въведени правила за блокиране и разрешаване:

```
# usbguard list-rules
```

Ще бъде изведен подобен резултат:

```
1: allow id 1d6b:0002 serial "0000:00:1a.7" name "EHCI Host Controller" hash
"qb6ikMLAJ13NeCmjHAxFkngOyIoJA/UHBAyxhglGwc0=" parent-hash
"yWtwmKzYb9cybHOymCnoVfpedSWnCtYfN7eSUQx7wGU=" with-interface 09:00:00 with-
connect-type ""
2: allow id 1d6b:0002 serial "0000:00:1d.7" name "EHCI Host Controller" hash
"9JszuPWESBEur6P34acG7ZtBUqRa6fRgdsma6hSNpOk=" parent-hash
"S0B+OWTcDg6PfHFlF0chk7joLm2raiSYdhNb1N1/BOk=" with-interface 09:00:00 with-
connect-type ""
3: allow id 1d6b:0001 serial "0000:00:1a.0" name "UHCI Host Controller" hash
"7/Y7ayZhucdn9UUOSc/idMgOqIKmfZucV47nbxk0OLo=" parent-hash
"e/RW0mMbM+TSFQxpRiMEfL7/3RJfKVdqffBm9F5qA+E=" with-interface 09:00:00 with-
connect-type ""
4: allow id 1d6b:0001 serial "0000:00:1a.1" name "UHCI Host Controller" hash
"uKF8muAZ/ZwTnGOEa97Vx7kX6Ty3AFju2zOWmg1gQyM=" parent-hash
"u+Q+yP/dUE7P1BP6elGXoDiB0tN2AQko8a2NqHjtIwg=" with-interface 09:00:00 with-
connect-type ""
5: allow id 1d6b:0001 serial "0000:00:1a.2" name "UHCI Host Controller" hash
"vQ/6Pys9qoHsEq8TLrUen99AJhsryBex/b6PhpZH3GA=" parent-hash
"GElG6oq2EB1lLUtC3adS8yzvhlSQWdT3TB4R6WBeYlA=" with-interface 09:00:00 with-
connect-type ""
6: allow id 1d6b:0001 serial "0000:00:1d.0" name "UHCI Host Controller" hash
"bjtVSUusIDxj13VXcZojsuMRNykZ7U/PLmXWG7xPhDo=" parent-hash
"jW2YTPWRLeQOE7Q8I2f0pdN13zFYXVQGQoNmr1gDZgg=" with-interface 09:00:00 with-
connect-type ""
7: allow id 1d6b:0001 serial "0000:00:1d.1" name "UHCI Host Controller" hash
"Z0pthcq0nFBS1408iQbne9dbBzJ5DDg5j3DOTFBO+C0=" parent-hash
"1f6hp0G45b+z7zSlP4CX4HXDdlrldzpbVhwZ1h7zsjI=" with-interface 09:00:00 with-
connect-type ""
```

```
8: allow id 1d6b:0001 serial "0000:00:1d.2" name "UHCI Host Controller" hash
"BKvEUUpIAivwthZQbvlpoMynYY8ZgHaR3v2nUupxuIE=" parent-hash
"nKOQytSaVkGSK63fd4ahzmPpgWm2LNkuHaRNyk+xbkg=" with-interface 09:00:00 with-
connect-type ""
12: allow id 346d:5678 serial "4550951102238269837" name "Disk 2.0" hash
"3tFZGFEHLIRm0fo61YQ+at6PmWePvwruxz5hXoN3QG4=" parent-hash
"9JszuPWESBEur6P34acG7ZtBUqRa6fRgdsma6hSNpOk=" via-port "2-2" with-interface
08:06:50 with-connect-type "unknown"
13: allow id 1241:1177 serial "" name "USB Optical Mouse" hash
"pCr9UOnNvk9W518fD66XxU1903xYdjjbB8oWXjkf6OE=" parent-hash
"7/Y7ayZhucdn9UUOSc/idMgOqIKmfZucV47nbxk0OLo=" via-port "3-1" with-interface
03:01:02 with-connect-type "unknown"
```

От горния пример установяваме като „разрешени“ **8-те** виртуални системни устройства и в допълнение – една **USB**-памет (под №**12**) и една оптична мишка (под №**13**) – двете разрешени от нас самите.

Ако желаете да премахнете по-рано въведеното разрешително правило за определено устройство (например защото вече нямате доверие на това устройство), трябва да се ориентирате под какъв номер системата разпознава съответното правило и да укажете премахването му:

```
# usbguard remove-rule Номер
```

(където под '**Номер**' разбираме номера на съответното разрешително правило – в горния пример '**12**' или '**13**').

Добра практика е регулярно да проверявате какви устройства се включват към вашите **USB**-портове – и ако засечете непознато устройство, на първо място да изследвате къде е закачено физически, респективно – дали поражда обезпокоителни съмнения. Последното може да означава, че вашата система е компрометирана и се налага да предприемете действия за ограничаване на евентуалните рискове пред информационната сигурност.

7. Обслужване на компютъра в сервиз

Особено критичен е моментът, в който компютърът трябва да се обслужи в сервиз. Тогава се налага трети страни да получат достъп до хардуера, като по време на профилактиката и ремонта е възможно да бъде компрометирана вашата информационна сигурност. За да ограничите този риск, трябва да ползвате услугите само на доверени хардуерни специалисти (или на такива, които са отдалечени от вас и не може да се предположи, че са специално мотивирани да ви компрометират). В някои случаи се препоръчва дори да изпратите компютъра през ваш доверен посредник, за да не може да се установи връзката му с вас. Имайте обаче предвид, че е възможно компрометиране на сигурността „по подразбиране“ и усилията ви да останете анонимни да се окажат напълно безполезни.

По възможност твърдият диск трябва да бъде демонтиран и заменен от такъв, който не ползвате за поверителна информация – за да няма възможност съдържанието ви да бъде копирано и подложено на анализ.

Препоръчва се (ако компютърът не се обслужва чрез посредник) да участвате лично в манипулациите, които се извършват в сервиза, да се осведомявате подробно за естеството на всяко действие и да повишавате колкото е възможно повече своето собствено разбиране за хардуера. Добро начало на това е да се запознаете подробно с техническата спецификация на вашия компютър и да проучите вида, функциите и особеностите на всеки от по-важните компоненти от хардуера, а така също – да се интересувате и да проучвате всяка манипулация, която специалистите осъществяват софтуерно или хардуерно спрямо вашия компютър.

Постарайте се да присъствате и да се интересувате от всяко действие, като при необходимост си водете бележки какво точно се извършва и какво се твърди, че се случва с всяка от съответните манипулации. Така ще можете след това да направите свое проучване и да установите доколко дадените разяснения отговарят на истината.

Ако последвате горните напътствия, на първо място ще бъдете в състояние да разбирате какво се извършва с вашата система при профилактика и ремонт, и на второ място ще придобиете способност да извършвате сами повечето манипулации, които са ви необходими. Последното е гаранция срещу евентуална намеса на трети страни в правилното функциониране на системата. Вашата информационна сигурност е преди всичко ваша лична (не на другиго) отговорност!

8. Заключване на системата с бърза клавишна комбинация

Добра практика е да програмирате бърза клавишна комбинация (или дори един-единствен клавиш), чието натискане незабавно заключва системата. Това е особено важно за потребители, които работят в присъствието на трети лица (например колеги от офиса, посетители в кафенето и т.н.), за да могат да заключват бързо и лесно компютъра, когато им се налага например да се отдалечат за малко от него. Дори обаче да сте сами, не можете да бъдете сигурни дали в един следващ момент това няма да се промени. Практиката познава много случаи, при които след внезапно нахлуване потребителят бива отстранен насила от системата, преди да е успял да я заключи – съответно цялата информация, с която е работил в този момент, попада във владението на трети страни (разшифрована и достъпна). В такава ситуация няма да имате време да посегнете към мишката, да изберете от графичния интерфейс съответните падащи менюта и да натисне върху **Shut Down...**; нито да отворите терминала и да въведете командата '**`poweroff`**'. Бутонът за изключване на компютъра също не върши работа, защото изисква да бъде държан натиснат продължително време, преди да произведе действие.

Ето защо препоръчваме да си изберете бутон от клавиатурата, който не ползвате за нищо друго (често такива са десните **`[Ctrl]`**, **`[Alt]`** и някои от функционалните бутони) и да го програмирате да заключва системата при натискане. Подходящ за тези цели при моделите **x200** и **x200s** е също така продълговатият бутон горе вляво **`[ThinkVantage]`**. Да настроите бърз бутон можете от менюто **Settings / Keyboard / View and Customize Shortcuts / System** в графичния интерфейс (или нещо подобно, в зависимост от особеностите на вашата система). В повечето случаи всяка от функциите за заключване на екрана (**Lock screen**), за излизане от сесията (**Log out**) и за изключване на компютъра (**Shut down**) би ви свършила работа. Ние обаче препоръчваме най-меката от тях (**Lock screen**), за да ви бъде удобна и в „нормални“ ситуации (например когато просто отивате до бара, за да вземете кафето си). Най-меката функция е за предпочитане и заради това, че по-твърдите функции изискват допълнително потвърждаване, за да произведат действие (а в критична ситуация е възможно да нямате възможност за такова допълнително потвърждаване). Повечето системи ви позволяват да добавите и собствени функции (**Custom Shortcuts**) – отваря се диалогов прозорец, в който като '**Command**' можете да въведете например '**`systemctl suspend`**' (**Lock screen**), '**`exit`**' (**Log out**) или '**`poweroff`**' (**Shut down**), а след това и самата клавишна комбинация, която да задейства интересуващата ви функция.

Не забравяйте на няколко пъти да пробвате своята клавишна комбинация (за да се убедите, че наистина работи) и да оттренирате нейното бързо въвеждане. Така ще можете само с едно мръдване на пръстите да заключвате системата – което е удобно при често ставане от компютъра за някакви кратки действия и може да бъде спасително в критичен момент. И накрая – създайте си навик да заключвате системата винаги, когато я оставяте – дори и да сте сигурни, че е „само за малко“ и да сте сигурни, че наоколо няма „нищо застрашаващо“. В много случаи кратките отдалечавания се удължават неограничено поради възникването на непредвидени обстоятелства, а почти винаги нападенията срещу информационната сигурност са внезапни, неочаквани и много бързи. Утвърждаването на един такъв навик, който да следвате инстинктивно, без замисляне и без изключение, може да ви бъде само от полза – дори и в един единствен критичен момент, някъде нататък във времето.

9. Защитен достъп до системата при настъпило компрометиране

Възможно е вашият компютър да бъде компрометиран и това да наложи да откачите твърдия диск, за да стартирате вашата система от сигурен компютър. Възможно е също така системата да бъде компрометирана и по-нататъшното ѝ стартиране да разкрива опасност от активирането на компрометирани софтуерни модули. В такива случаи е необходимо да осъществите инцидентно стартиране от друг компютър или инцидентно достъпване на системата без да

бъде стартирана (например от „жив“ **USB**-носител), за да можете да извлечете безопасно интересуващата ви информация и да изтриете системата (или да подмените компютъра).

Както вече проследихме, стартирането на системата в никакъв случай не трябва да става от компютър, чието софтуерно обезпечение от „ниско“ ниво не е свободно. Възможно е обаче софтуерът от „ниско“ ниво да не е настроен да работи с напълно шифрована система, което ще му попречи да стартира вашия напълно шифрован твърд диск. За да преодолеете този проблем (и евентуално за да препрограмирате flash-чипа, за да можете оттук нататък да стартирате напълно шифрованата система), трябва да въведете „ръчно“ няколко команди, които при други обстоятелства биха се изпълнявали автоматично.

За да може да бъде стартирана операционната система, в стартиращата програма **GRUB** (която в нашия случай е част от **Libreboot**-пакета, записан в flash-чипа) трябва да въведем съответни инструкции и параметри за стартиране на системното ядро и началната система. При шифровани системи за тази цел трябва първо да разшифроваме дисковото пространство. Едва след това би било възможно да се изпълнят въведените инструкции и параметри за стартиране.

Ако се наложи да стартирате инцидентно от компютър с Libreboot без настройки за шифровани системи (например защото хардуерът ви е компрометиран и сте преместили твърдия диск с вашата информация на друг компютър), трябва при стартиране на компютъра да изискате системата да ви предостави възможност за въвеждане на команди – като натиснете **`[c]`** (от **C**[**ommand**]). След като сторите това, идва ред да въведете една след друга няколко команди към стартиращата програма **GRUB** (явяваща се част от **Libreboot**-пакета).

Първата команда указва на стартиращата програма да разшифрова вашия твърд диск:

```
> cryptomount (ahci0)
```

(където символът '**>**' обозначава командния ред на **GRUB**, не е част от командата и не трябва да го въвеждате). Ако имате закачени няколко твърди диска към компютъра, е възможно изразът '**`ahci0`**' да придобие следващ номер – примерно '**`ahci1`**', '**`ahci2`**' и т.н. (може да прегледате достъпните устройства с команда '**`ls`**').

Ще бъде изведено указание да въведете парола за разшифроване на напълно шифрования твърд диск и след като сторите това, ще бъде изведено подобно положително съобщение:

```
Slot 0 opened
```

Следващите команди указват къде да бъде локализирана стартиращата програма за системното ядро и началната система (отново с команда '**`ls`**' можете да се ориентирате за наличните възможности). Първо трябва да укажете в кой дял на устройството се намира стартиращата програма:

```
> root=lvm/Група-Системен_дял
```

... след което трябва да укажете местоположението и параметрите на системното ядро:

```
> linux /boot/vmlinuz-linux-libre-lts \
root=/dev/mapper/Група-Системен_дял rw
```

(където трябва да замените изразите '**`Група`**' и '**`Системен_дял`**' с дадените от вас за вашата система).

В случай, че желаете да осъществите препрограмиране на вашия flash-чип (например защото този компютър вече ще се стартира с вашата напълно шифрована система), трябва да добавите и познатия параметър '**iomem=relaxed**':

```
> linux /boot/vmlinuz-linux-libre-lts \
root=/dev/mapper/Група-Системен_дял iomem=relaxed
```

Следват да укажете на програмата **GRUB** къде се намира стартиращата система, която да бъде заредена в **RAM** - паметта на компютъра:

```
> initrd /boot/initramfs-linux-libre-lts.img
```

... и накрая можете да пристъпите към самото стартиране на системата:

```
> boot
```

Имайте предвид, че горните команди ще действат само за сесията, която току що стартирахте. При повторно стартиране на компютъра ще бъде необходимо отново да въведете „ръчно“ същите команди, за да укажете **GRUB** да стартира вашата изцяло шифрована система. За да не се налага да въвеждате тези команди всеки път, е необходимо да въведете параметъра '**iomem=relaxed**' и да препрограмирате вашия flash-чип с **Libreboot**-пакет, в който стартиращите конфигурационни файлове '**grub.cfg**' и '**grubtest.cfg**' са редактирани съгласно посоченото в Част **III**.

Ако вашата операционна система бъде компрометирана и по-нататъшното ѝ стартиране вече е неуместно (например защото това може да стартира и инсталиран в нея зловреден код), възниква необходимост да отворите системата по начин, който да не я стартира, но да ви позволи да достъпите необходимата ви информация и да я извлечете, преди предстоящото форматиране на твърдия диск и изтриване на компрометираната система.

Ако вашата система не беше напълно шифрована, това можеше да се осъществи съвсем просто – като закачите твърдия диск към друга система и осъществите достъп до информацията в диска от тази друга система, или като стартирате компютъра от „жив“ носител. При напълно шифровани системи можете да постъпите по сходен начин, но за да осъществите достъп, е необходимо да извършите няколко допълнителни операции.

На първо място трябва да разшифровате твърдия диск (без това да довежда до стартиране на системата):

```
# cryptsetup open /dev/Устройство Наименование1
```

(където под '**Устройство**' разбираме наименованието, дадено от вас на общия физически дял, в който е организирана системата, а под '**Наименование1**' – произволно дадено от вас обозначение, под което желаете системата да възприема физическия дял в настоящата сесия и което не дублира никое друго обозначение в системата).

Възможно е (например при еднотипно инсталирани системи) да се получи дублиране в наименованията на логическите **LVM**-групи – което ще породи конфликт при опит същите да бъдат достъпени от система с идентични наименования (еднаквите наименования водят до грешка, тъй като системата не може да установи към кое устройство, дял или група насочвате съответните команди). В такъв случай може да се наложи аварийно да промените част от наименованията, за да достъпите по описания по-горе начин, без да предизвикате конфликт.

На първо място трябва да извлечете **VG UUID**-идентификатора на виртуалната група:

```
# vgdisplay
```

Ще бъде изведен подобен резултат:

```
# vgdisplay
   WARNING: VG name group is used by VGs 0eDk8D-KKZD-1DGU-uytL-fEk9-Uz5K-
hfjB7A and 4QrMKx-S2zh-mh2B-S4X5-83ee-CGNt-AaBsyE.
  Fix duplicate VG names with vgrename uuid, a device filter, or system IDs.
  --- Volume group ---
  VG Name               group
  System ID
  Format                lvm2
  Metadata Areas        1
  Metadata Sequence No  2
  VG Access             read/write
  VG Status             resizable
  MAX LV                0
  Cur LV                1
  Open LV               0
  Max PV                0
  Cur PV                1
  Act PV                1
  VG Size               28.63 GiB
  PE Size               4.00 MiB
  Total PE              7330
  Alloc PE / Size       7330 / 28.63 GiB
  Free  PE / Size       0 / 0
  VG UUID               0eDk8D-KKZD-1DGU-uytL-fEk9-Uz5K-hfjB7A

  --- Volume group ---
  VG Name               group
  System ID
  Format                lvm2
  Metadata Areas        1
  Metadata Sequence No  7
  VG Access             read/write
  VG Status             resizable
  MAX LV                0
  Cur LV                4
  Open LV               3
  Max PV                0
  Cur PV                1
  Act PV                1
  VG Size               <465.76 GiB
  PE Size               4.00 MiB
  Total PE              119234
  Alloc PE / Size       119234 / <465.76 GiB
  Free  PE / Size       0 / 0
  VG UUID               4QrMKx-S2zh-mh2B-S4X5-83ee-CGNt-AaBsyE
```

(където можете да се ориентирате например по размерите (**VG Size**); или като неколкократно повтаряте горната команда, като междувременно ту закачате, ту разкачате външния твърд диск, докато различните кои редове ту се появяват, ту изчезват от извежданите резултати – следователно се отнасят за интересуващата ви виртуална група.

След като се ориентирате за **VG UUID**-идентификатора на интересуващата ви виртуална група, можете да промените нейното наименование:

```
# vgrename 0eDk8D-KKZD-1DGU-uytL-fEk9-Uz5K-hfjB7A Ново_наименование
```

(където под '**0eDk8D-KKZD-1DGU-uytL-fEk9-Uz5K-hfjB7A**' разбираме **VG UUID**-идентификатора на първата от виртуалните групи в горния пример, а под '**Ново_наименование**' – такова наименование, което не съществува във вашата система и съответно няма да причини конфликт.

Остава да разкачите външния носител и отново да го закачите към системата, която вече ще го инициализира с новото му наименование. Сега вече можете да определите точка на закачане за този от логическите дялове на външния носител, който ви интересува (например вашия потребителски '**data**' дял):

```
# mount /dev/mapper/Група-Логически_дял /Точка_на_закачане
```

(където под '**Точка_на_закачане**' разбираме мястото в системата, където желаете да бъде закачен съответният логически дял, най-често **/mnt**).

В случай, че сте обособили допълнително шифрован дял в компрометираната система (например '**lock**'), можете да укажете неговото разшифроване (след като вече твърдият диск с цялостно шифрованата система е разшифрован):

```
# cryptsetup open /dev/mapper/Група-Шифрован_дял Наименование2
```

(където под '**Шифрован_дял**' разбираме наименованието, дадено от вас на допълнително шифрования логически дял (например '**lock**'), а под '**Наименование2**' – произволно дадено от вас обозначение, под което желаете системата да възприема допълнително шифрования логически дял в настоящата сесия и което не дублира никое друго обозначение в системата).

И накрая остава да закачите разшифрования допълнително шифрован логически дял към системата, за да можете да достъпите неговото съдържание:

```
# mount /dev/mapper/Наименование2 /Точка_на_закачане2
```

Системата ще закачи указаните от вас (и вече разшифровани) дялове към посочените точки на закачане и ще ви позволи да работите с намиращото се в тези дялове съдържание, без това да включва стартирането на какъвто и да било софтуер, който евентуално е бил инсталиран и е действал в компрометираната система.

Когато приключите работа, ако желаете да изключите компрометираната система безопасно, трябва най-напред да откачите закачените преди това дялове:

```
# umount /dev/mapper/Наименование2
```

... и да върнете съдържанието в шифровано състояние:

```
# cryptsetup close Наименование2
```

... респективно:

```
# umount /dev/mapper/Наименование
```

... и:

```
# cryptsetup close Наименование
```

... и накрая да укажете системата да преустанови подаването на електричество към твърдия диск (за да предотвратите риска от токов удар и загуба на данни при хардуерното му разкачане):

```
# udisksctl power-off -b /dev/Устройство
```

Помощната система, от която достъпвате вашата компрометирана система, ще приведе съответните дялове в пасивно, напълно шифровано състояние, като направените от вас промени (ако има такива) ще останат записани в шифрован вид.

10. Инцидентна загуба на 'частния' ключ

Възможно е поради някаква причина определено критично съдържание (например вашият 'частен' ключ) да се изгуби и това да постави под въпрос информационната сигурност (която включва не само преграждането на евентуални опити за неразрешен достъп, но също и обезпечаване успешното осъществяване на разрешен достъп до съответното поверително съдържание). Ако 'частният' ключ бъде изгубен, това ще направи цялото шифровано със съответстващия му 'публичен' ключ съдържание недостъпно за неговия притежател. Поради тази причина не е излишно да положите усилия за неговото възстановяване – ако това все още е възможно. По същия начин ще действате и при загубата на друго критично за информационната сигурност съдържание.

Шифровъчните ключове на **GnuPG** се съхраняват в защитената директория **`/home/Потребител/.gnupg/`** със следните файлове и поддиректории:

`openpgp-revocs.d`
сертификати за анулиране на 'частните' ключове;

`private-keys-v1.d`
'частни' ключове;

`pubring.kbx`
'публични' ключове с метаданни за тях;

`pubring.kbx~`
резервен архив на 'публичните' ключове;

`trustdb.gpg`
мрежа на доверието с други потребители.

Видно от наименованията на файловете, 'частните' ключове се съхраняват в поддиректорията '**private-keys-v1.d**', а 'публичните' – във файла '**pubring.kbx**'. Възможно е директорията **.gnupg/** (ведно с шифровъчните ключове) да бъде инцидентно изтрита. Това не е голям проблем за 'публичните' ключове, ако сте ги споделили в internet например (ще се спрем на този въпрос в Част **VIII** по-долу) – просто ще ги свалите от там и ще ги инсталирате в системата. Ако обаче нямате резервно копие на вашите 'частни' ключове, след инцидентното изтриване на директорията **.gnupg/** като единствен шанс за възстановяване на тези ключове остава това да ги намерите „на ниско ниво“ в паметта на твърдия диск – ако междувременно не са били заличени и все още могат да бъдат извлечени от там.

Във връзка с горното – изтритото от гледна точка на потребителския интерфейс съдържание продължава да бъде налично в паметта, но с времето в тази памет се извършват нови записи; някои от тези записи могат да попаднат върху мястото с вашите изтрити 'частни' ключове и да ги заличат (вече необратимо). Следва да имате предвид, че такива записи се осъществяват от системата включително и когато вие не записвате нищо – така че самото стартиране и ползване на системата е предпоставка да изгубите окончателно инцидентно изтритото съдържание. В такава ситуация е важно да не стартирате повече системата, както и евентуално дори да не монтирате нейните дялове към друга система (или ако ги монтирате, това да е само в режим на четене (**Read-only**) – с команда '**mount -o ro /dev/Устройство /Точка_на_закачане**'). Така ще се гарантира, че инцидентно изтритото съдържание стои в пасивно (непроменено) състояние в паметта и няма риск да го изгубите изцяло или отчасти заради евентуални последващи записи (ако досега вече не сте го загубили).

На първо място свалете 'публичния' ключ в друга система, за да извлечете наименованията на двойката файлове на 'частния' ключ (единият от които служи за „полагане“ на цифровите подписи, а другият – за разшифроване на поверителното съдържание). След като разполагате с 'публичния' ключ, извлечете наименованията така:

```
$ gpg --with-keygrip -k Идентификатор
```

(където под '**Идентификатор**' разбираме която и да било разпознаваема част от данните на вашия шифровъчен ключ). Ще бъде извлечен подобен резултат:

```
sec#  rsa4096 2017-10-21 [SC]
      AE06191DE7B72F74ACE2BB257B62D2A7D14C089B
      Keygrip = 4A6A0AF85193D8DA3B435C9BC20A2E4C5F60C3AE
uid      [ unknown] Наименование (Коментар) <e-mail@example.com>
ssb   rsa4096 2017-10-21 [E]
      Keygrip = AAFF07743CC12D22D8D76E3FDA7360E9B2BCFD79
```

(откъдето е видно, че ключът е създаден на **21.10.2017** г.; има идентификатор '**AE06191DE7B72F74ACE2BB257B62D2A7D14C089B**'; файлът за цифрово подписване е с идентификатор '**4A6A0AF85193D8DA3B435C9BC20A2E4C5F60C3AE**'; и файлът за разшифроване е с идентификатор '**AAFF07743CC12D22D8D76E3FDA7360E9B2BCFD79**').

След като разполагате с тези данни, идва ред да достъпите системата, в чиято памет се надявате все още да се съхранява изтритото съдържание на директорията **.gnupg/**. Както стана дума, не трябва да стартирате тази система и не трябва да монтирате нейните дялове към друга стартирана система – за да запазите нейното съдържание в пасивен (непроменен) вид.

Достъпете системата от външен „жив“ носител или закачете твърдия диск към помощен компютър, разшифровайте с проследената по-горе команда '**cryptsetup open /dev/Устройство Наименование**' и проверете дали дяловете на системата са налични (командата '**lsblk**'). Ако всичко е наред, можете да пристъпите към издирване на вашите шифровъчни ключове в паметта. Това е възможно, защото всеки шифровъчен ключ, съхраняван на твърдия диск в своето шифровано състояние, съдържа специфични маркери, които го отличават от друго съдържание. И ако потърсите тези маркери на ниско ниво във вид на прост текст, който се надяваме все още да е достъпен, може би ще локализирате къде точно е необходимият ви запис и ще можете да го извлечете – без значение, че от гледна точка на

потребителския интерфейс това съдържание вече е изтрито и не съществува. В тази връзка – един примерен 'частен' ключ (в своето шифровано състояние) има подобен вид:

```
Created: 20230723T203603
Key: (protected-private-key (rsa (n F#00CF52CFC144B9A1AD076D989E83CAF435
 3296759AE55A5BF7C1BACBDB3243A87773F203A5996B2235F5AEBD5B763F7CC39A2E0B
 DE4D1471D983EF7946AB5DD11A816C98837880EEE7DDCB34207EC87E339C692EEFF03A
 56F62F195BB6F0317D676DCC81EEDB284D9874B6BDD10688B3782F1B3060B543E32731
 C5F5441C43ABE2F094D24A6D0C7BC5FFBDB370127660C2A23C33F911923884285194B8
 2110AB6F1CF304A5D3ED69C09A8462FF214F940418FD71DF775501170F08F53A835D6B
 CBF7020408F74FCF188C30E79A529E31DDFFE34C1546C9CC4E3B36D71AA760F7F9A523
 C8F330E7483338FBF9877175C72BD0EB8E92949EB3FDFE9E31AE8BEA179559CEC9D75A
 212E51C445C9A17279CF5CDB866DF23A1C519013723E469FC567919A1586B00BDC2CCD
 68CA47F15B09D2AA7AA58BFD9C896FD04334D23E9F66D76BC747281A20B4B8C0A43DB9
 1F005838E0C5AFDDBADFDC34059809F19FEE0DE05DD6F0EF8E663642C2179B71A5C0E3
 BA04F9F24904F430710F91E77DFAD74E5391#)(e #010001#)(protected
  openpgp-s2k3-ocb-aes ((sha1 #F6E0602376174228#
  "17753088")#A754C176F20F98D3C884931A#)#3FBBC8F634FB78CC207105D3AF53C4
 E5FE171B65AA2B34D94E053677EAC1220639FB61A6238BF875C8BB08C1F69A31CF87EE
 AC9C3574A7A9B15B31CC8BF1B92AF5E6522C22DF6FF0DD8FCB11B3B27B58197B6D121C
 A7DB84735C2600F133F32B93E509F7C92D8E11DB11681E4208BDBDAAB3ECB1F1DD4566

(...)

 8643A0FDED4C6AD8EE7B31A810C7AC367751DE5A7FA8A0C2AB738A5CED49D2C8722B51
 CAF6831130AF2DAB7AAA63EEC6DF20D63CBA3DCD666854755897C695BF3E7916CDB209
 09A1DB345D9ECADD2B7DFCDB4282370C9D7A92B2B02163#)(protected-at
  "20230723T203631")))
```

Впечатление в дадения примерен 'частен' ключ правят няколко разпознаваеми маркери с подобен вид:

`Created: 20230723T203603`
където се посочва, че примерният 'частен' ключ е създаден в **20:36** ч. на **23.07.2023** г.;

`Key: (protected-private-key (rsa`
където се посочва, че интересуващият ни 'частен' ключ е от типа **RSA**;

`(protected-at "20230723T203631")))`
където се посочва, че даденият в примера 'частен' ключ е защитен в този му вид в **20:36** ч. на **23.07.2023** г.

Възможно е маркерите да варират (но най-вероятно ще останат низове като '**`protected-private-key`**'). Също така най-вероятно няма да помните точния час и минута на създаването на ключа, но все пак можете да извлечете от 'публичния' ключ точната дата на неговото създаване и да редактирате горните низове така: '**`Created: ГГГГММДДТ`**' или съответно '**`protected-at "ГГГГММДДТ`**' (отказвайки се от часа, минутите и секундите на създаването; където под '**`ГГГГММДД`**' разбираме годината, месеца и деня на създаване на ключа). След като имате предвид всичко, казано дотук, можете да опитате да търсите низове като горните в паметта на вашето устройство (и по-точно – във вашия системен дял, където е директорията **`/home/Потребител/.gnupg/`** с изтритите ключове в нея):

```
# LANG=C grep -obUaP "Търсен_низ" /dev/mapper/Група-Системен_дял \
| tee Файл_с_общи_резултати
```

Когато системата се върне в режим на очакване, можете да прегледате файла с общите резултати с проследената по-горе команда '**less**' или например с програмата **Nano**. Ще бъде изведен подобен резултат:

```
120549778:Търсен_низ
4832925383:Търсен_низ
12863322838:Търсен_низ
32086099466:Търсен_низ
```

(където под '**Търсен_низ**' разбираме търсената поредица от символи; предхождана от номера на поредния байт в паметта, откъдето започва тази поредица). В цитирания пример са локализирани **4** съвпадения.

Можем да прегледаме всяко от локализираните съвпадения така:

```
# dd bs=1 skip=Пореден_байт count=Брой_символи \
if=/dev/mapper/Група-Системен_дял | less
```

(където под '**Пореден_байт**' разбираме номера на поредния байт в паметта, откъдето започва търсеният низ (напр. '**120549778**'); а под '**Брой_символи**' разбираме точния брой символи, които желаем да видим (напр. '**1000**'). Системата ще изведе поисканите символи и от тях можем да се ориентираме дали наподобяват нещо като дадения по-горе примерен 'частен' ключ. Ако резултатът е удовлетворяващ, но не съвпада точно с 'частния' ключ, можем „да преместим“ малко по-напред и по-назад чрез променяне на числата '**Пореден_байт**' и '**Брой_символи**', и когато постигнем напълно точно съвпадение (виждаме целия 'частен' ключ, без нито един липсващ или излишен символ), можем да изведем във файл:

```
# dd bs=1 skip=Пореден_байт count=Брой_символи \
if=/dev/mapper/Група-Системен_дял of=Файл_с_търсен_резултат
```

... който файл да въведем в **/home/Потребител/.gnupg/private-keys-v1.d/** под съответното заглавие с разширение '**.key**' – нашият 'частен' ключ е възстановен!

Още една възможност за извеждане на всичките съвпадения наведнъж в отделен текстов файл се открива чрез тази команда:

```
# (while read line; do
offset=$(echo $line | cut -f 1 -d ':')
echo $offset
hexdump -s $(($offset - 256)) -n 256 -C /dev/mapper/Група-Системен_дял
echo
done) < Файл_с_общи_резултати > Файл_с_подбрани_резултати
```

(където под двата параметъра '**256**' разбираме брой символи съответно преди и след съвпадението, които да бъдат изведени в удобен за четене вид във '**Файл_с_подбрани_резултати**'. При преписване на горното може да напишете всичко на един ред или на отделни редове, като не е нужно да слагате '****' в края на всеки ред, понеже в началото на кода е поставена отваряща скоба '**(**', затворена на последният ред с израза '**done)**'.

След стартиране на командите, остава да прегледате този файл (команда '**less**' или програмата **Nano**) и докато се движите надолу и нагоре (бутоните **[PgDn]** и **[PgUp]**), да потърсите символи, които приличат на 'частен' ключ:

```
120549778
00000000 43 72 65 61 74 65 64 3a  20 32 30 32 33 30 37 32 |Created: 2023072|
00000010 33 54 32 30 33 36 30 33  0a 4b 65 79 3a 20 28 70 |3T203603.Key: (p|
00000020 72 6f 74 65 63 74 65 64  2d 70 72 69 76 61 74 65 |rotected-private|
00000030 2d 6b 65 79 20 28 72 73  61 20 28 6e 20 23 30 30 |-key (rsa (n #00|
00000040 43 46 35 32 43 46 43 31  34 34 42 39 41 31 41 44 |CF52CFC144B9A1AD|
00000050 30 37 36 44 39 38 39 45  38 33 43 41 46 34 33 35 |076D989E83CAF435|
00000060 0a 20 33 32 39 36 37 35  39 41 45 35 35 41 35 42 |. 3296759AE55A5B|
(...)
00000bd0 34 35 44 39 45 43 41 44  44 32 42 37 44 46 43 44 |45D9ECADD2B7DFCD|
00000be0 42 34 32 38 32 33 37 30  43 39 44 37 41 39 32 42 |B4282370C9D7A92B|
00000bf0 32 42 30 32 31 36 33 23  29 28 70 72 6f 74 65 63 |2B02163#)(protec|
00000c00 74 65 64 2d 61 74 0a 20  20 22 32 30 32 33 30 37 |ted-at.  "202307|
00000c10 32 33 54 32 30 33 36 33  31 22 29 29 29 0a       |23T203631"))).|
```

(където най-отгоре вляво е номерът на поредния байт, от който започва записът; първата колона представя номерата на редовете в шестнадесетичен формат; следват шестнадесет шестнадесетични двуцифрени числа, представляващи записаната „на ниско ниво“ в паметта информация; която информация е представена в най-дясната колона в човешки четим формат във вид на прост текст, заграден от двете страни – от типа на '**|Created: 2023072|**'.

Когато локализирате резултат, приличащ на търсения 'частен' ключ, остава да запомните номера на поредния байт горе вляво и да нацелите точното начало '**Created**:' и точния край '**))).**' на ключа, който в крайна сметка можете да изведете във файл '**.key**':

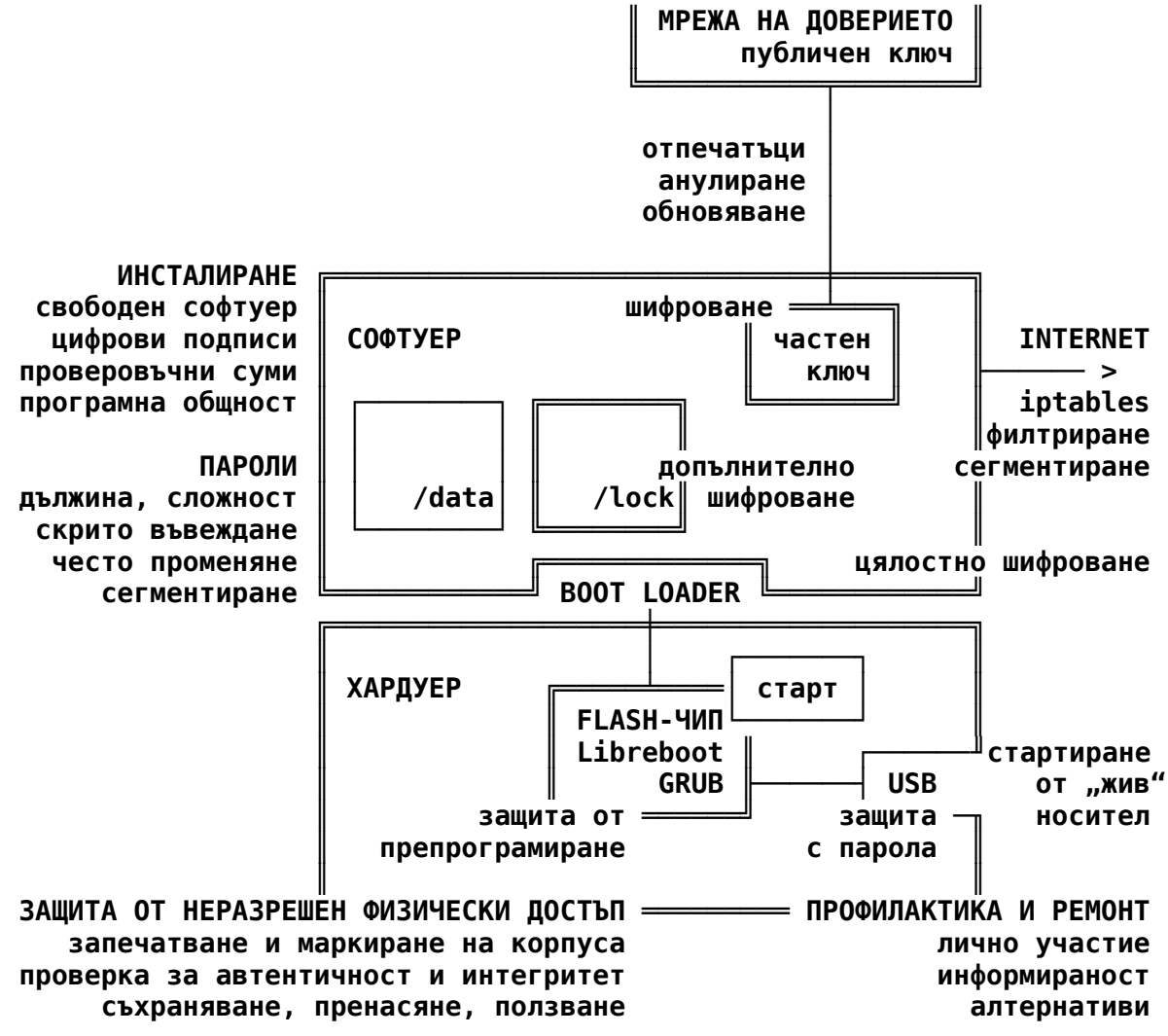
```
# dd bs=1 skip=Пореден_байт count=Брой_символи \
if=/dev/mapper/Група-Системен_дял of=Файл_с_търсен_резултат
```

Ако всичко е наред, търсеният 'частен' ключ ще бъде възстановен.

11. Цялостна организация на информационната сигурност

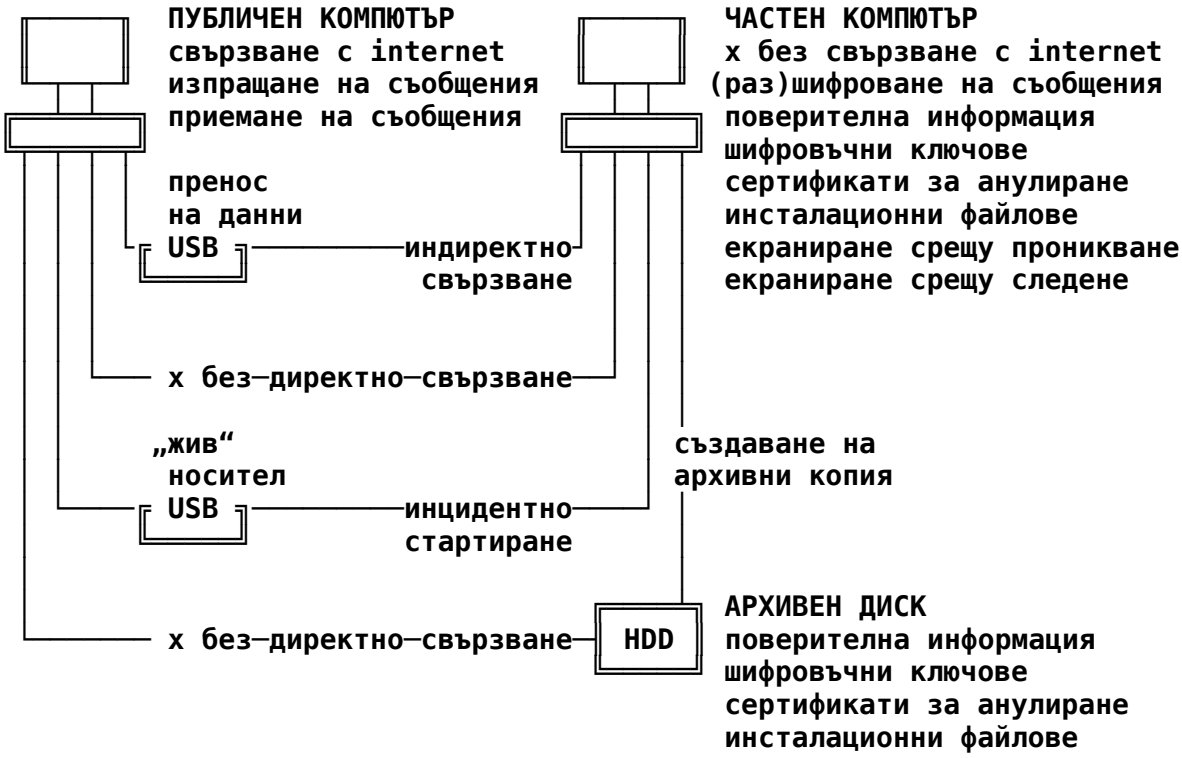
Както вече многократно обърнахме внимание, информационната сигурност е комплексен въпрос, чието адекватно отработване включва системни и взаимосвързани мерки, отразяващи всеки един от възможните рискове. Дори само един аспект от сигурността да бъде компрометиран, това може да доведе до пълен провал на вашата защита и да обезсмисли всичките ви усилия. Ето защо трябва да обърнем внимание на цялостното организиране на информационната сигурност, което включва на първо място мерки за обезпечаване на отделното компютърно устройство и на второ място – такива за организирането на цялостна система от устройства, всяко от които изпълнява строго специфични функции.

Обезпечаването на отделното компютърно устройство е предмет на почти цялото изложение и може да бъде обобщено в подобна примерна схема:



Принципна схема на информационната сигурност при отделно устройство

Организирането на цялостна система от устройства, всяко от които изпълнява строго специфични функции и по този начин се превръща в елемент от вашата информационна сигурност, е предмет преди всичко на настоящата Част **VII** и донякъде на предходната Част **VI**. Тъй като този кръг от проблеми значително надхвърля целите на нашето изложение, се ограничаваме до следната примерна обобщаваща схема:



Принципна схема на информационната сигурност при система от устройства

Ключов момент в предложената схема е поддържането на две отделни компютърни устройства, едното от които е „публично“ и предназначено за ежедневна обичайна работа и свързване с internet, а другото е „частно“ и предназначено за съхраняване на поверителна информация (включително 'частни' ключове) и за (раз)шифроване на съобщения, без да се свързва с internet и без да се свързва директно с такива компютърни устройства, които се свързват с internet. По този начин се ограничават голяма част от рисковете за информационната сигурност, произтичащи от осъществяването на дистанционни високопрофилни атаки; като единствена алтернатива остава това – да се осъществи физическо проникване в помещенията, където „частното“ компютърно устройство се съхранява; и да бъдат разбити неговите защити, за да се извлече записаната в него поверителна информация. Съществуването на специално обособена външна памет (в примера – твърд диск) за съхраняване на архивни копия обезпечава безболезнено възстановяване на основните информационни масиви в случай на компрометиране на някое от компютърните устройства, а специален „жив“ носител позволява инцидентно стартиране на компрометираните компютърни системи, от които желаем да извлечем безопасно интересуващата ни информация.

Горните схеми представляват единствено примерен вариант за обезпечаване сигурността на отделното компютърно устройство и за цялостно организиране на информационната сигурност чрез система от няколко устройства с различно предназначение и функции. Предложените схеми в никакъв случай не бива да ви ограничават. Конкретният механизъм, по който ще гарантирате технически и организационно вашата информационна сигурност, зависи преди всичко от естеството на обработваната от вас поверителна информация, от това кой би могъл да има интерес от информацията и какви ресурси би могъл да мобилизира за нейното неразрешено достъпване; зависи също от вашето занятие, от начина ви на живот, от вашите довереници, от по-широкото ви обкръжение и от други фактори, чиято специфика не бихме могли да засегнем в пълнота в настоящото изложение.

12. Стандартите за информационна сигурност на вашите довереници

Колкото и добре да сте организирали вашата информационна сигурност, усилията ви няма да бъдат особено полезни, ако вашите довереници също не поддържат адекватни нива на сигурност. Преди да започнете да споделяте поверителна информация, опитайте да установите доколко страните, на които мислите да се доверите, са в състояние да обезпечат вашата (и тяхната собствена) информационна сигурност. В случай, че сте обект на високопрофилна атака, но поддържате високи стандарти за информационна сигурност, е напълно вероятно векторът на атаката да се насочи към трети страни, които работят с ваша поверителна информация, но се явяват „слаби звена“ във веригата. Обикновено сигурността се компрометира от най-уязвимото място.

Добър пример в тази връзка е **Едуард Сноудън**. Преди да направи своите разкрития, в продължение на няколко месеца бившият служител на разузнаването на **САЩ** е търсил журналист, който да се съгласи да последва неговите напътствия за информационна сигурност. В противен случай е щял да бъде разкрит и „неутрализиран“ много преди светът да разбере за него. По същия начин ви насърчаваме преди да се доверите на някого, да проучите как вашата поверителна информация ще бъде обработвана и съхранявана; и ако откриете слабости – да подпомагате вашите довереници да направят необходимите подобрения в начина си на работа – както ние сега ви помагаме чрез нашето ръководство.

13. Неотложни действия при пробив в сигурността

Тъй като „абсолютна сигурност“ няма (особено в електронна среда), е напълно възможно (независимо от всичките мерки, които вземате) вашата информационна сигурност все пак да бъде компрометирана. Възможно е например компютърът да бъде отнет от вас (изгубване, кражба, грабеж, изземване). Възможно е майсторски прикрито проникване в хардуера да компрометира тайната на вашите пароли, правилното протичане на шифровъчния процес или осъществявания трансфер на данни. Възможно е паролите ви да „изтекат“ поради въвеждането им в подменено компютърно устройство или в неправилно работещ софтуер, или поради заснемане на момента, в който ги въвеждате. Възможно е паролите ви да бъдат разбити и чрез класически bruteforce методи. Възможно е паролите и шифровъчните ключове да бъдат извлечени и от **RAM**-чиповете, докато още се съхраняват в тях непосредствено след изключване на компютъра. Възможно е просто да бъдете задържани и принудени със заплашване, сила или измама да дадете достъп до вашите пароли и шифровъчни ключове. Възможно е и това – да бъдете предадени от ваши довереници, с които споделяте тайните си или им възлагате да поддържат вашите машини. Възможно е най-накрая да изгубите (забравите) своите пароли или да настъпят технически/организационни сривове, довеждащи до изгубване на достъп до вашето съдържание.

Всяко от горните събития би компрометирало вашата информационна сигурност и ви задължава да пристъпите незабавно към допълнителни мерки, с които да отразите допуснатия пробив. На първо място – трябва да подмените незабавно вашите пароли, а ако е необходимо – да преинсталирате системата. Ако е получен неразрешен достъп до системата ви и до паролите за вашите шифровъчни ключове, е вероятно те да са копирани. Понякога се налага да анулирате всичките шифровъчни ключове и да ги замените с нови, и да смените цялото компютърно оборудване, като прехвърлите своята поверителна информация другаде.

Не забравяйте, че данните не се заличават напълно от хардуера с простото им изтриване – което също може да се превърне в предпоставка за „изтичане“ на поверителна информация. Особено при **SSD**-устройствата и **USB**-паметите не само изтриването, но дори презаписването на произволни символи в същите сектори не е надеждно средство срещу последващо извличане на поверителната информация. Понякога се налага да унищожите физически старите носители, преди да се разделите с тях – за да сте сигурни, че поверителната информация (или разпознаваема част от нея) няма да бъде извлечена. За не толкова критична информация в повечето случаи можете да изтриете надеждно, като укажете на системата да презапише определен брой пъти произволни символи в секторите, където преди това е била информацията:

```
$ shred -n Брой -u -z Файл
```

(където под '**Брой**' разбираме указания от вас брой презаписвания с произволни символи (например '**5**'); под '**-u**' разбираме указание системата да изтрие файла (информацията) след презаписването; и под '**-z**' разбираме указание след презаписването и изтриването системата запише нули (от '[**Z**]**ero**') в съответните сектори, за да прикриете факта на изтриването). Можете да зададете и по-голям брой презаписвания, което ще направи евентуално последващо извличане на информацията още по-трудно – но в повечето случаи не може да имате пълна сигурност, че заличаването е напълно необратимо. За тази цел трябва просто да унищожите физически носителя на информацията – като се съсредоточите не толкова на кутията, а на самите елементи с памет (записващите повърхности на твърдия диск, **SSD**-чиповете с памет и т.н.).

Добре е да имате готовност при необходимост да предприемете набелязаните стъпки внезапно, без колебание и без допускане на грешки поради паника. Добре е да работите по предварително подготвен, добре обмислен и оттрениран план, включващ превантивно съхраняване на сигурно място на резервни копия от необходимия ви софтуер, техническа документация и хардуер. Така ще ограничите (доколкото е възможно) последиците от евентуално настъпило компрометиране на сигурността и ще минимизирате възможностите за по-нататъшно проникване във вашата поверителна информация, като същевременно си осигурите средства за възможно най-бързо възстановяване интегритета на засегнатите слоеве от вашата работна среда.

С оглед обезпечаването на ясен и детайлен план, трябва да отчетете възможните рискове, вероятността всеки един от тях да настъпи и последиците, които този риск би предизвикал. В тази връзка от помощ могат да ви бъдат представените по-горе примерни схеми за обезпечаване на информационната сигурност по отношение на отделното компютърно устройство и по отношение на цялостна система от устройства. При всички случаи трябва да адаптирате предложените схеми към вашите специфични особености или най-малкото да ги проиграете старателно, до постигането на необходимата рутина и стабилност в следването на набелязаните подходи, включително в условията на стресова ситуация.

14. Информационната сигурност като слоеве

Мислете за сигурността като за слоеве, които се напластяват един върху друг. Ако единият слой бъде компрометиран, след него идват следващи.

Не се доверявайте на една парола, на един шифровъчен ключ, на едно компютърно устройство. Ако допуснете това, сигурността на вашата информация може да се компрометира доста лесно и бързо. Ако някое ваше устройство бъде компрометирано, е за предпочитане то да не съдържа особено ценна информация, а разкритата парола сама по себе си да не е достатъчна за достигане до най-съкровените тайни.

Изтривайте всичко, което не ви е наистина необходимо. Поверителна информация съхранявайте само в шифрован вид, по възможност на носители, които не се свързват с internet, никой не знае къде се намират и дори не подозира за съществуването им.

Във връзка с горното е препоръката да използвате допълнително шифровани дялове, които остават защитени, докато останалата част от системата е разшифрована и работите с нея. В същия дух е възможността да ползвате няколко отделни устройства, част от които не се свързват с internet и не е известно къде се намират, за да се превърнат в сигурни убежища за поверителното съдържание. Пак във връзка с горното са възможностите за обособяване на скрити шифровани пространства.

Но каквито и технологии да приложите, за предпочитане е все пак най-съкровените ви тайни никога да не се оказват на устройство, което някой обследва за извличането им. В този смисъл известната сентенция, приписвана неправилно на Конфуций: *„Трудно е да търсиш черна котка в тъмна стая – особено когато котката я няма вътре.“*

VIII. ИЗВЪРШВАНЕ НА НАДЕЖДНО GPG-ШИФРОВАНЕ

Шифроването няма как да бъде надеждно, ако не се извършва от надежден софтуер с надежден хардуер. Ето защо по-горе се спряхме обстойно на обезпечаване на сигурността на вашата компютърна система – за да бъде възможно след взетите дотук мерки да пристъпим към извършването на наистина надеждно шифроване. В противен случай, дори да приложим най-сигурните шифровъчни технологии, информационната сигурност може да бъде компрометирана от най-разнообразни вектори. Дори може би няма да се стигне до атакуване на самата шифровъчна технология (която сама по себе си е едно от най-надеждните звена в сигурността, но оставена сам на себе си е напълно безполезна; като блиндирана врата, около която липсват стени и тя стои заключена насред нищото).

Настоящата Част **VIII** беше поводът изобщо да започнем да работим по това ръководство. Много скоро обаче си дадохме сметка, че надеждното шифроване е само една част от комплексните задачи, които трябва да бъдат решени за постигане на действителна информационна сигурност. Ето защо настоящата част се оказа сред още осем части, които единствено заедно могат да решат интересуващите ни проблеми. Само една напълно Свободна операционна система (**II**), действаща на изцяло шифровано устройство (**IV**), което работи със Свободен софтуер от „ниско“ ниво (**III**), може да обезпечи представеното в настоящата част надеждно **GPG**-шифроване (**VIII**). При това не можете да разчитате на никоя шифровъчна технология, ако не създавате сигурни пароли и ако не боравите адекватно с тях (**V**). Никоя от горните мерки не би имала особен смисъл, ако заедно с всичко друго не поддържате и разумни нива на вашата (и на вашите устройства) физическа сигурност (**VII**), и ако не положите дължимите грижи за защита на вашата internet-свързаност (**VI**). Ето защо – преди да се запознаете с настоящата част и да шифровате надеждно вашето поверително съдържание, установявайки надежден контакт с нас или с другиго (**VIII**), ви насърчаваме да се запознаете внимателно с цялото изложение – и едва накрая да пристъпите към настоящата част. Процесът на (раз)шифроването е само „връхчето“ на така описания айсберг.

Gnu Privacy Guard (известен още като **GnuPG**) е Свободен софтуер за надеждно шифроване на вашата поверителна информация – било за да я съхранявате на вашия компютър, било за да я предавате на вашите довереници във вид, защитен от погледа на трети страни. **GnuPG** е базиран на софтуера **Pretty Good Privacy** (известен още като **OpenPGP**), разработен от **Филип Цимерман** (**Philip Zimmermann**), като от него са отстранени всички несвободни компоненти и е оптимизиран за работа със Свободни операционни системи.

https://gnupg.org/
https://www.openpgp.org/

Работата на **Цимерман** е повратен момент в криптографията, тъй като за първи път технология за надеждно шифроване става достъпна за обикновените потребители (макар и с несвободен лиценз). През **1991** г. (малко след като споделя изходния код на **OpenPGP** в internet) срещу **Цимерман** е образувано досъдебно производство по Закона за контрол върху износа на оръжие (**Arms Export Control Act**), съгласно който щатското законодателство разглежда технологиите за надеждно шифроване като оръжие и забранява тяхното разпространяване извън **САЩ**. Производството е прекратено през **1996** г. (без привличане към отговорност), тъй като година преди това кодът е публикуван като хартиена книга, а свободното издаване и разпространяване на книги се гарантира от Първата поправка в Конституцията на **САЩ**; без значение от факта, че напечатаният на хартия изходен код може след това да се препише на компютър, да се компилира до бинарни файлове и да се стартира като софтуер. Така **OpenPGP** се разпространява по целия свят, а към **1999** г. **Вернер Кох** (**Werner Koch**) разработва и свободната версия **GnuPG**. Това слага окончателния край на монопола на специалните служби върху технологиите за надеждно шифроване.

1. Обща характеристика и теоретичен модел на шифровъчния процес

GnuPG позволява прилагането на т.нар. „симетрични“ шифровъчни алгоритми (**Symmetric Chipher Algorithms**) и „асиметрични“ шифровъчни алгоритми (**Asymmetric Chipher Algorithms**), всеки от които притежава определени предимства и недостатъци.

„Симетричните“ алгоритми разчитат на един 'симетричен' ключ (**Symmetric Key**), най-често представляващ парола (или символите в избран от потребителя файл, които служат като ключ вместо парола). Този 'симетричен' ключ, съчетан със „симетричния“ шифровъчен алгоритъм на софтуера, преобразува нешифрованото съдържание в шифровано и обратно – преобразува шифрованото съдържание в нешифровано. Тъй като 'симетричният' ключ (паролата) се преобразува до неузнаваемост посредством шифровъчния алгоритъм и тази криптографска производна на свой ред преобразува до неузнаваемост шифрованото съдържание, практически няма възможност да бъде разбит един „симетричен“ алгоритъм (особено когато съобщението е кратко и не могат да се проследят закономерности в шифровъчната текстура) – освен ако не бъде разкрита паролата или прихванато едно съобщение в шифрован и в нешифрован вид. Това прави „симетричните“ алгоритми особено устойчиви срещу разбиване и поради тази причина са предпочитани за осъществяването на надеждно шифроване. В допълнение към това, „симетричните“ алгоритми работят много бързо и позволяват лесното (раз)шифроване на значителни обеми от съдържание – което ги прави предпочитани при защитаването например на масиви с архивирано съдържание или при цялостното шифроване на устройства и системи.

Заедно с всичките си предимства обаче, „симетричните“ алгоритми не могат да отговорят на един съществен проблем: как две страни да обменят надеждно шифровано съдържание помежду си, ако преди това не са обменили надеждно 'симетричния' ключ (парола) за (раз)шифроване на съдържанието? Решението идва с „асиметричните“ алгоритми, които разчитат на един 'публичен' ключ (**Public Key**) и на още един 'частен' ключ (**Secret Key**), асоциирани еднозначно един с друг, но всеки от тях работещ само еднопосочно. 'Публичният' ключ е свободно (публично) достъпен за всеки желаещ, но с него не може да се извърши разшифроване. 'Публичният' ключ позволява само шифроването на съдържание и осъществяването на криптографска проверка на цифрови подписи, „положени“ с асоциирания към него 'частен' ключ (каквато проверка направихме в Част **I**, докато обследвахме автентичността на цифрово „подписания“ инсталационен файл). Обратният процес по разшифроването на съдържание (и този по „подписването“ на файлове) е възможен единствено чрез прилагане на 'частния' ключ, асоцииран еднозначно с 'публичния' ключ. 'Частният' ключ е известен само на своя притежател и се съхранява на сигурно място в неговата система, като за допълнителна сигурност е шифрован със 'симетричен' ключ и ползването му е възможно само след въвеждане на парола за неговото „симетрично“ разшифроване.

Разполагайки с нечий 'публичен' ключ, вие можете свободно да шифровате съобщения, които да бъдат разшифровани от кореспондента ви посредством неговия 'частен' ключ. Ако на свой ред предоставите ваш 'публичен' ключ на вашия кореспондент, той също може да ви изпраща съобщения, които се разшифроват само чрез вашия 'частен' ключ, асоцииран с предоставения от вас 'публичен' ключ. Това позволява изграждането на сигурен канал за шифрована комуникация, без да е необходимо преди това кореспондентите да са разменили ключове, пароли или други средства за надеждно шифроване.

Недостатък на „асиметричните“ алгоритми е, че работят по-бавно от „симетричните“, не са подходящи за шифроване на големи обеми от съдържание и най-важното – поддават се на относително по-лесно разбиване в сравнение с „асиметричните“. Това е така, защото могат да бъдат анализирани криптографските зависимости между символите от 'публичния' ключ и символите в шифрованото съдържание, преобразувано (шифровано) с този 'публичен' ключ. Всеки желаещ да осъществи подобен анализ, може да шифрова неограничен брой известни за него съобщения с достъпния му 'публичен' ключ и да работи върху получените криптографски производни. Поради това „асиметрични“ алгоритми се ползват предимно за текуща комуникация, докато сериозното шифроване на съдържание (и особено на архиви и системи) се осъществява „симетрично“. Именно заради това в Част **IV** осъществихме цялостно шифроване на нашата система чрез „симетричен“ алгоритъм.

Като математико-теоретична илюстрация на съвременната криптография можем да предложим „асиметричния“ шифровъчен алгоритъм **RSA** (по първите букви от фамилиите на неговите създатели **Ron Rivest**, **Adi Shamir** и **Leonard Adleman**). Алгоритъмът **RSA** е представен публично за първи път през **1978** г., патентован е през **1983** г. и е наложен като технологичен стандарт към **1993** г., а патентната му закрила е изчерпана на **21.09.2000** г. След тази дата **RSA** навлиза в свободните криптографски технологии (вкл. **GnuPG**) и се нарежда сред най-разпространените шифровъчни алгоритми в света. Неговата надеждност до този момент не е компрометирана (поне няма официални данни за това) и до постигането на необходимия технологичен пробив при т.нар. „квантови компютри“, този алгоритъм би следвало да продължи да се ползва с доверие. Счита се, че появата на „квантов компютър“ с капацитет **20'000'000** кубита (какъвто се очаква да бъде създаден до **2040** г.) би разбил **RSA** в днешния му вид за **8** часа. Следва в тази връзка да се

отбележи, че към момента няма публично известни достатъчно надеждни шифровъчни алгоритми, които да отговорят на „квантовите“ предизвикателства (дори и в техния все още теоретичен и експериментален вид).

https://en.wikipedia.org/wiki/RSA_(cryptosystem)

Подобно на всеки „асиметричен“ алгоритъм, **RSA** разчита на двойка от 'публичен' и 'частен' ключ. В основата на тези ключове стоят две достатъчно големи прости числа (делят се единствено на себе си и на **1**), подбрани със сходен, но различен брой цифри и представени двоично. Ще наречем тези числа '**q**' и '**p**'. Произведението на '**q**' и '**p**' дава модула '**n**', служещ като една от двете съставни части на 'публичния', съответно на 'частния' ключ. Другата съставна част на 'публичния' ключ представлява експонентата '**e**' (число, по-голямо от **1** и по-малко от **φ(n)**, като '**e**' и '**φ(n)**' нямат общи делители). Другата съставна част на 'частния' ключ представлява експонентата '**d**' (удовлетворяваща Функцията на **Кармайкъл** '**d*e ≡ 1 (mod φ(n))**') и следователно позволяваща разделяне без остатък на израза '**e*d-1**' или '**(p-1) * (q-1)**'). Така можем да обобщим отношенията между еднозначно асоциираните един с друг 'публичен' ключ (**n**;**e**) и 'частен' ключ (**n**;**d**).

При „асиметрично“ шифроване чрез **RSA** поверителното съдържание '**m**' (представено двоично, като цяло положително число) се преобразува посредством 'публичния' ключ (**n**;**e**) до „асиметрична“ шифрограма със стойност '**c**'. Стойността '**c**' представлява остатъкът от операцията '**m^e / n**' или '**c ≡ m^e (mod n)**'. Нито „асиметрично“ шифрованата стойност '**c**', нито 'публичният' ключ (**n**;**e**) позволяват да се изведе първоначалното поверително съдържание '**m**'. За тази цел стойността '**c**' трябва да бъде преобразувана посредством 'частния' ключ (**n**;**d**) съгласно израза '**c^d (mod n)**'. За осъществяването на тази разшифроваща операция експонентата '**d**' има критично значение, тъй като стойността '**c**' и модулът '**n**' са известни на всеки, който разполага с 'публичния' ключ (**n**;**e**) и е прихванал шифрограмата; единствената неизвестна стойност за осъществяване на разшифроващото изчисление '**c^d (mod n)**' остава именно експонентата '**d**', която от своя страна може да бъде изведена повторно, ако са известни първоначалните прости числа '**q**' и '**p**'.

И за да постигнем пълна яснота, нека проследим алгоритъма **RSA** още веднъж чрез естествени числа в рамките на една проста математическа сметка. Да приемем като стойности на първоначалните прости числа '**q**' и '**p**' (делими единствено на себе си и на единица) числата **3** и **11** (в съвременната криптография се използват числа с дължина от по няколкостотин символа). Произведението на '**q**' и '**p**' дава модула '**n**' (в нашия случай **n = 3 * 11 = 33**). Като експонента '**e**' на 'публичния' ключ трябва да изберем такова число, което е по-малко от числото **φ = (q-1) * (p-1)** и няма общ делител с него. В нашия случай **φ = (3-1) * (11-1) = 20** и следователно на горните условия отговарят числата **3**, **7**, **9**, **11**, **13**, **15**, **17** и **19**. Нека изберем от дадените възможности числото **7** като стойност на експонента '**e**'. Следователно нашият 'публичен' ключ (**n**;**e**) може да се представи с естествени числа като (**33**;**7**). Като експонента '**d**' на 'частния' ключ трябва да изберем такова число, чието произведение с експонентата '**e**' може да се раздели на '**φ**' с остатък единица (т.е. функцията '**d*e ≡ 1 (mod φ(n))**'). Такова число може да се извлече от израза **(d*e) / n** (в нашия случай **(d*7) / 20**). Нека изберем като отговарящо на изискуемото условие числото **3**, защото **(3*7) / 20 = 21 / 20** (което е делимо с остатък **1**). Следователно нашият 'частен' ключ (**n**;**d**) може да се представи с естествени числа като (**33**;**3**). От тук можем да шифроваме поверителното съдържание '**m**' съгласно израза '**m^e (mod n)**' и да разшифроваме получената шифрограма '**c**' съгласно израза '**c^d (mod n)**'. Нека приемем като стойност на нашето поверително съдържание '**m**' числото **5**. В такъв случай шифроването ще протече като **5^7 (mod 33)** и шифрограмата '**c**' ще придобие стойност числото **14**. Разшифроването ще протече като **14^3 (mod 33)** и ще резултира обратно в първоначалната стойност **5**. Ако трета разполагаща с 'публичния' ключ (**33**;**7**) страна прихване шифрограмата '**c**' (**14**), няма как да осъществи изчисление, което да я доведе надеждно до първоначалната стойност на поверителното съдържание '**m**' (**5**), без да разполага било с експонентата '**d**' (**3**) на 'частния' ключ, било с първоначалните прости числа '**q**' (**3**) и '**p**' (**11**), от които в крайна сметка да изведе експонентата '**d**'. В това се изразява сигурността на „асиметрични“ алгоритми като **RSA**.

2. Основни криптоаналитични методи и атаки

Преди да се заемем с осъществяването на надеждно **GPG**-шифроване, е необходимо да се спрем за малко на основните криптоаналитични методи и атаки, насочени към разбиване на шифровъчните технологии. Принципното разбиране за тези методи и атаки е необходимо за ограничаване рисковете от прилагане на подобни подходи за компрометиране на информационната сигурност – и ще позволи по-разумно управление на шифровъчния процес.

Честотният криптоаналитичен метод е първият документиран анализ на шифровъчни алгоритми. Той разчита на особеностите на езика и по-специално – на честотата, с която определени символи или комбинации от символи се появяват в текста. Така например, в предходното изречение (ако всеки от символите би бил шифрован просто чрез заместването му с друг символ, както се постъпва при по-примитивните шифровъчни алгоритми), можем да проследим **15** появи на символа '**о**' и толкова на символа '**и**', **10** появи на символа '**а**', **9** появи на символа '**е**'. На този етап не знаем кои са тези символи, но честотата на появата им и спецификите на българския език ги дефинира като гласни букви, при това – най-вероятно именно цитираните (тъй като те обикновено се срещат по-често). В другия край са символи като '**ц**' и '**ч**', които се срещат съвсем рядко (както е характерно за подобни редки символи). С по-нататъшен криптоанализ могат да бъдат установени значенията на достатъчен брой символи, което накрая да доведе и до пълно разчитане на шифрограмата. Защитата срещу подобни методи включва прилагането на съвременни шифровъчни алгоритми, при които криптографското преобразуване не е еднозначно и включва голям брой преобразувания с висока степен на рандомизация.

Статистическият криптоаналитичен метод разчита на относителните зависимости в случайни статистически величини. Като илюстрация можем да приложим т.нар. „парадокс на рождените дни“: ако бъдат избрани произволно **23**$^{\text{ма}}$ души, статистическата вероятност да се намерят двама, родени на една и съща дата, достига **50,7%**; ако бъдат избрани произволно **50**$^{\text{ма}}$ души, статистическата вероятност ще достигне до **97,0%** (при възможни **365** дати за рождени дни и изключена датата **29** февруари от високосните години като пренебрежимо малка вероятност за подобен рожден ден). Изчислението е по формулата **1-365 ! / ((365-n) ! x 365^n)**, където под '**365**' разбираме всичките възможни рождени дни, а под '**n**' – броя случайно избрани хора. Подобни вероятностни зависимости (от езиково и технологично естество) позволяват на атакуващата страна да проследи криптографските текстури, които са относително (но не абсолютно) рандомизирани и на тази база да извлече статистически обосновани предположения за шифровъчния ключ. Защитата срещу подобни методи включва прилагането на по-сложни и непредвидими пароли, по-дълги шифровъчни ключове и по-мощни технологии за рандомизация.

Диференциалният криптоаналитичен метод е насочен към проследяване на зависимости между символите в определено нешифровано съдържание, станало известно на атакуващата страна и символите в шифрограмата, която се образува при неговото шифроване. **Интегралният криптоаналичитен метод** е частен случай, приложим при т.нар. „асиметрични“ шифровъчни алгоритми (където 'публичният' ключ за шифроване е публично известен). При този частен случай на криптоанализ се осъществяват голям брой шифрования на идентично съдържание посредством публично известния 'публичен' ключ, като при всяко следващо шифроване се променят само няколко бита (символа) от съдържанието. Последното позволява криптоанализ на флуктуациите, които настъпват в образуващите се шифрограми и от там – реконструиране на 'частния' ключ, еднозначно асоцииран с прилагания за шифроването 'публичен' ключ. Защитата срещу подобни методи включва въвеждането на по-кратки срокове на валидност на двойките 'публичен' и 'частен' ключ, и тяхното редовно подменяне с нови шифровъчни ключове; включително и провеждане на комуникация с верижно подменяне на ключовете, където във всяко следващо съобщение се предлага новият 'публичен' ключ за очаквания отговор.

Линейният криптоаналитичен метод включва относими към определено съдържание, към неговата шифрограма и към известния 'публичен' ключ линейни уравнения, насочени към изчисляването на линейни приближения, с помощта на които да бъде разбит 'частният' шифровъчен ключ. Последователно с различни уравнения се осъществяват голям брой такива приближения до установяването на такава част от символите в 'частния' ключ, която да позволи провеждането на Bruteforce атака с разумна продължителност за окончателно разбиване на шифровъчния ключ. Защитата срещу подобни методи включва създаването на възможно по-дълги и по-сложни пароли и шифровъчни ключове, въвеждането на по-кратки срокове на валидност на ключовете и тяхното редовно подменяне с нови.

Bruteforce атаките, за които споменаваме многократно нагоре в изложението, са първият документиран подход за разбиване на шифри. Състоят се от последователно опитване на вероятни пароли (или шифровъчни ключове), докато накрая действителната парола (шифровъчен ключ) се уцели и настъпи разшифроване. Това се осъществява на базата на съществуващи или оптимизирани към конкретна цел „речници“ с всевъзможни комбинации от символи, чийто брой започва от **1**, **2**, **3** и т.н., и включва вероятностно подбрани думи, изрази, последователности от клавиатурни подредби, предвидими резултати от недобра рандомизация. Защитата срещу подобни атаки включва създаването на възможно по-дълги и по-сложни пароли и шифровъчни ключове, които не включват нито предполагаеми думи, изрази или

последователности от символи, нито предвидими техни пермутации; и прилагането на технологии за възможно най-мощна рандомизация, осигуряващи наистина безразборни символни последователности за шифровъчните ключове.

Ciphertext-only атаките се отличават с това, че атакуващата страна разполага с определен брой шифрограми (възможно е това да бъде резултат от системно прихващана кореспонденция). Криптоанализът в такъв случай стъпва върху известните параметри на съдържанието в нешифрован вид, с което атакуващата страна не разполага: например език на текста (с произтичащите от това честотни характеристики), очаквани елементи и формат на съдържанието, и т.н.). Този метод е ефикасен при по-ранните и по-примитивни шифри (отречени в голямата си част от съвременната криптография), а така също – при по-кратки шифровъчни ключове. Защитата срещу подобни атаки включва ползването на по-дълги шифровъчни ключове, приложени с по-сложни шифровъчни алгоритми. Големият брой преобразувания при такива шифровъчни ключове и алгоритми рандомизира шифрограмата в степен, която прави нейното последващо анализиране до възможно най-голяма степен трудно и безполезно за довеждане на атаката до успешен резултат. Полезно е също така прилагането на по-кратки срокове на валидност на шифровъчните ключове, и тяхното редовно подменяне с нови.

Known-plaintext атаките се отличават с това, че атакуващата страна разполага едновременно с определена „симетрична“ шифрограма и с нешифрованото съдържание, чието криптографско преобразуване е довело до създаването на тази шифрограма (възможно е това да бъде резултат от прихванати архиви, в които са съхранявани едно до друго съобщенията в нешифрован вид и съответстващите им шифрограми). Наличието на подобна информация дава възможност на атакуващата страна да анализира начина, по който съдържанието е било преобразувано от първоначалния си вид до шифрования – и това в някои случаи може да доведе до реконструиране на шифровъчния ключ. Защитата срещу подобни атаки включва внимателно унищожаване на ненужните копия от информация и съхраняването им само във вид на шифрограми или само във вид на разшифровано съдържание, а така също – последващо шифроване с други шифровъчни ключове.

Chosen-plaintext атаките са приложими при „асиметричната“ криптография, където атакуващата страна може да шифрова избрано от нея (и известно за нея) съдържание с набелязаните 'публични' ключове (които по дефиниция са публично достъпни). От тук атаката продължава своето развитие като Known-plaintext атака, при която атакуващата страна разполага едновременно с определена шифрограма и с нешифрованото съдържание, чието криптографско преобразуване е довело до създаването на тази шифрограма. В основата на този вид атаки стоят диференциалният и интегралният криптоаналитичен метод. Защитата срещу подобни атаки включва въвеждането на по-кратки срокове на валидност на двойките 'публичен' и 'частен' ключ, и тяхното редовно подменяне с нови шифровъчни ключове; а така също – поверителното споделяне на 'публични' ключове, които да останат недостъпни за трети страни.

Side-channel атаките се ориентират към проследяване на начина, по който функционира хардуерното оборудване и софтуерното обезпечение. Използват се криптоаналитични методи, насочени към фино наблюдаване на системата (количества разходвана енергия за отработване на подаденото натоварване, процесорно време за протичането на определени процеси, електромагнитни лъчения по време на съответните изчисления, излъчвани микрошумове от функционирането на хардуера и други параметри), от които да се направи извод за степента на приближаване до необходимата парола или шифровъчен ключ; или дори „да се подслуша“ и „да се прочете“ конкретно шифровъчно преобразуване. Защитата срещу подобни атаки включва физическа организация на информационната сигурност (в т.ч. екраниране на компютърното оборудване) и осъществяване на критичните процеси по (раз)шифроване в защитена среда, далеч от евентуални атакуващи страни и от възможности за наблюдаване (включително през internet) и последващо документиране и анализ на начина, по който системата работи.

Cold-boot атаките се ориентират към извличане на паролите и шифровъчните ключове от **RAM**-чиповете на компютъра, докато те все още се съхраняват в тях. Докато системата работи (в т.ч. докато е в състояние на хибернация или е в режим на очакване), а така също до няколко минути след изключването, **RAM**-чиповете поддържат заредени значителни части от системата, в т.ч. съхраняваните в нея пароли и шифровъчни ключове (които от своя страна може да са в открит, шифрован или хеширан вид). Със специални охлаждащи спрейове съхраняваната в **RAM**-чиповете памет може да бъде консервирана до около час. Това позволява **RAM**-чиповете да бъдат откачени от местата им и свързани към друга система, предназначена за извличане на съхраняваната в тях информация; или просто стартиране на първоначалната система от „жив“ носител (т.нар. „студено стартиране“ на системата) с операционна система, предназначена за извличането на тази информация. Защитата срещу подобни атаки включва ограничаване на

физическия достъп до хардуера (в т.ч. чрез спояване на **RAM**-чиповете към местата им, което да предотврати тяхното бързо откачане от там), цялостно изключване на системата при приключване на работа (вместо поставянето ѝ в хибернация или в режим на очакване) и ограничаване стартирането от „живи“ носители.

Black-bag атаките се ориентират към компрометиране на системата, без това да засяга пряко самите криптографски технологии. Използват се key-logger технологии за извличане на въвежданите от клавиатурата символи, зловреден софтуер за компрометиране функционирането на софтуера и други, целта на които е да запишат паролата по време на нейното въвеждане и да я изпратят към атакуващата страна, да компрометират процеса на шифроването или да свалят копие от поверителното съдържание преди да бъде шифровано. Същата цел може да се преследва и чрез разполагането на скрити камери, с които да бъде наблюдавана клавиатурата при въвеждането на паролите; или чувствителни микрофони, които „да прослушат“ специфичните звуци от натисканите клавиши в този критичен за сигурността момент (всеки клавиш при натискането му издава звук, който може изцяло или отчасти да бъде отграничен от звука на останалите клавиши в клавиатурата). Защитата срещу подобни атаки включва организация на физическата сигурност на компютърното оборудване, опазване на системата срещу осъществяване на неразрешен физически достъп, спазване на дисциплина, филтриране и сегментиране на работата в internet, редовно обновяване на софтуера до последните му версии, цялостно преинсталиране на системата при наличие на съмнения.

Man-in-the-middle атаките са приложими, когато атакуващата страна е компрометирала „симетричния“ ключ (или 'частния' ключ на една от страните) и прихваща изпращаните шифрограми; разшифрова ги с компрометирания ключ, прочита ги, редактира ги ако сметне това за необходимо, шифрова ги отново със съответния компрометиран ключ и ги изпраща до оригиналния получател. Възможно е заедно с това да изпрати и фалшиф шифровъчен ключ, с който да покани компрометираната страна да шифрова своите насрещни съобщения, които атакуващата страна да може да прочита и шифрова с действителния шифровъчен ключ за страната, за която тези съобщения са предназначени, преди да ги достави до нея. Защитата срещу подобни атаки включва прилагането на процедури за автентикация на страните и на шифровъчните ключове, цифрово „подписване“ на шифрограмите и въвеждане на по-кратки срокове на валидност на двойките 'публичен' и 'частен' ключ, и тяхното редовно подменяне с нови шифровъчни ключове.

Social-engineering атаките се ориентират към социално-психологическо манипулиране на притежателите на пароли и шифровъчни ключове към това сами да разкрият паролите (шифровъчните ключове) или да извършат нещо, което да доведе до компрометиране на тяхната информационна сигурност. Така например атакуващата страна може да се представи за „новия системен администратор“ с искане за спешен достъп с цел отстраняване на внезапно възникнала повреда, или да изпрати фалшифицирано съобщение уж от доверен източник с искане за предоставяне на резерви копия от шифровъчни ключове. Често този вид атаки се подготвя в продължителни периоди от време, като атакуващата страна изгражда връзки на доверие с атакуваната страна, преди да я тласне към самото компрометиращо действие. Защитата срещу подобни атаки включва спазване на дисциплина, следване на строги процедури по автентикация, филтриране и сегментиране на работата в internet, на цялостното поведение и достъпа до системата; и отказ от правенето на „благовидни“ изключения.

Rubber-hose атаките се ориентират към директно принуждаване чрез заплахи и/или физически посегателства притежателите на пароли и шифровъчни ключове да ги предоставят на атакуващата страна, която, получила физически достъп до съответните устройства, въвежда съответните пароли и шифровъчни ключове, и извлича интересуващото я поверително съдържание. Често този вид атаки се подготвят чрез събирането или фабрикуването на компромати, използването на организационни, икономически и други зависимости и уязвимости; и се прилагат от криминални структури, деспотични режими, авторитарни лица и организации. Възможно е по този начин едната страна да бъде задържана за известен период, докато в комбинация с други видове атаки (например от типа на **Social-engineering**) у другата страна се поддържа заблуждението, че „всичко е наред“, докато нейната информационна сигурност също бъде компрометирана. Защитата срещу подобни атаки включва организация на физическата сигурност, спазване на дисциплина, филтриране и сегментиране на дейностите, в т.ч. на устройствата и на местата, където се съхранява поверителна информация, а така също – поддържането на скрити шифровани пространства (както проследихме в края на Част **IV**).

3. Инсталиране на GnuPG във вашата система

Подобно на повечето свободни програми, **GnuPG** може да работи с разнообразни видове удобен за потребителите графичен интерфейс, включително web-базиран, включително на операционни системи като **Windows**, **MacOS**, **iOS**, **Android**. Както обаче стана дума, само изцяло Свободните операционни системи, базирани на **GNU/Linux-libre** и поставени напълно в контрола на потребителя, са в състояние да гарантират (някаква) сигурност. Ето защо не желаем да обсъждаме ползването на **GnuPG** при несвободни операционни системи. По сходни причини не желаем да обсъждаме и ползването на графичен интерфейс, особено online-базиран. Макар такъв интерфейс да е удобен и предпочитан от мнозина потребители, той крие допълнителни рискове за сигурността (а когато е online-базиран, практически води до отказ от каквато и да било сигурност). Ако въпреки казаното дотук решите да направите това и се нуждаете от съдействие, препоръчваме да се обърнете към вашия системен администратор.

GnuPG е автоматично инсталиран при голяма част от съществуващите Свободни операционни системи. Ако случайно не е наличен, може да го инсталирате с команда '**`pacman -S gnupg`**' (или при **Debian** безирани системи – '**`apt install gnupg`**'). Вашата система ще потърси **GnuPG** в софтуерните хранилища на доверените сървъри, разпространяващи Свободен софтуер и ще инсталира **GnuPG**.

Преди обаче да започнете да ползвате **GnuPG** по предназначение, препоръчваме ви да деактивирате **GnuPG** сесиите, които временно запомнят използваните частни шифровъчни ключове (с цел да спестят многократното въвеждане на парола). В противен случай има опасност вашият ключ да бъде ползван неправомерно, докато все още стои „отворен“. Можете да деактивирате сесиите на **GnuPG** така:

```
$ echo 'max-cache-ttl 0' >> ~/.gnupg/gpg-agent.conf
```

След изпълнението на горната команда в системния файл '**`gpg-agent.conf`**', намиращ се в „скритата“ конфигурационна директория ('**`/home/Потребител/.gnupg`**') ще бъде отразена тази инструкция:

```
max-cache-ttl 0
```

(което указва на **GnuPG** да не оставя ключовете в „отворено“ състояние – същите се шифроват веднага след прилагането им, когато пожелаете да ги ползвате).

Можете да проверите дали инструкцията е въведена в системния файл '**`gpg-agent.conf`**':

```
$ cat ~/.gnupg/gpg-agent.conf
```

Ако всичко е наред, в терминала ще бъде изведен горният конфигурационнен ред (съдържание на посочения системен файл).

Друга настройка, която е съществена за информационната сигурност и е добре да въведете, преди да започнете да ползвате **GnuPG** по предназначение, е насочена към това да отстраните помощния прозорец, който се отваря при въвеждане на пароли. Недостатък на този помощен прозорец е, че той представлява допълнителен графичен елемент, увеличаващ ненужно рисковете за пробив в сигурността; и не на последно място – при въвеждане в него се извежда индикатор за броя въведени символи (от типа на '**`**********`**' или '**`••••••••••••`**').

Както стана дума в Част **V**, това крие риска трета страна да установи приблизителния брой символи, от които се състои вашата парола (например като заснеме екрана ви) – което от своя страна би ускорило съществено разбиването на паролата чрез прилагането на bruteforce. Да премахнете помощния прозорец можете чрез тези команди:

```
$ echo 'pinentry-program /usr/bin/pinentry-tty' \
>> ~/.gnupg/gpg-agent.conf
```

и след това:

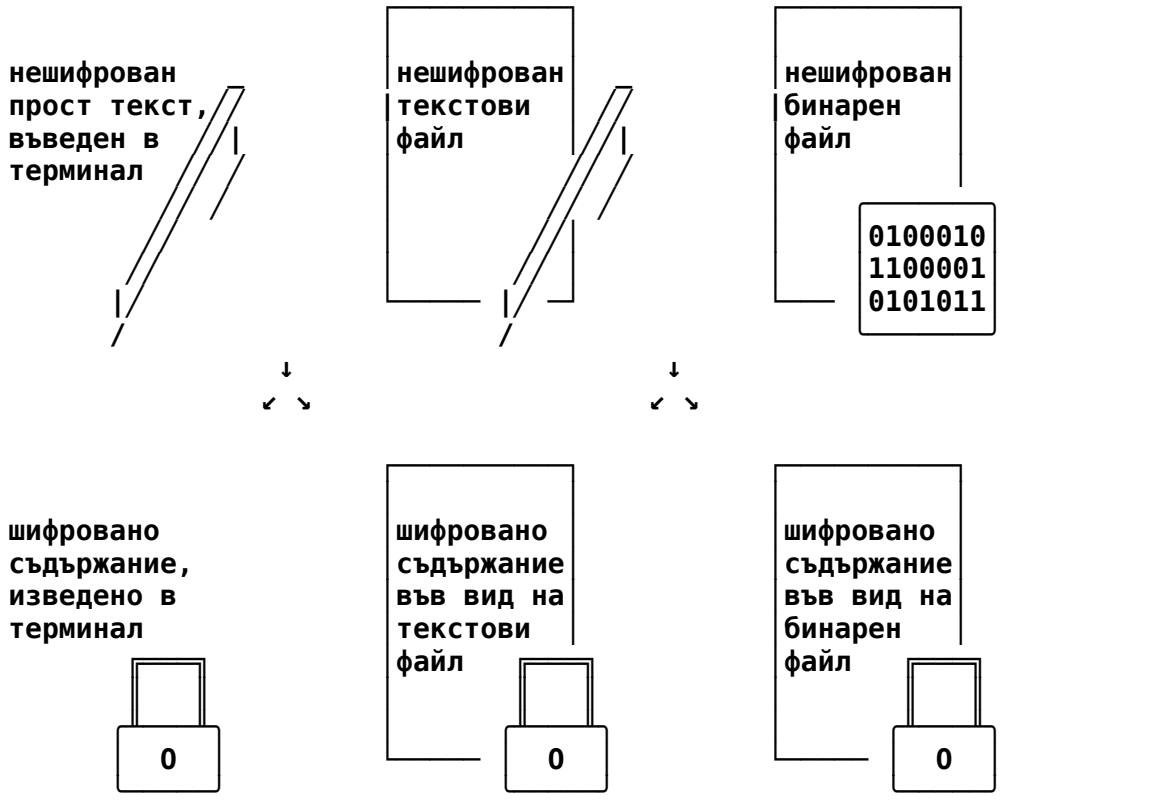
```
$ gpg-connect-agent <<< RELOADAGENT
```

Ако отново въведете посочената по-горе команда '**`cat ~/.gnupg/gpg-agent.conf`**', системата отново ще изведе съдържанието на конфигурационния файл '**`gpg-agent.conf`**' и ако всичко е наред, в него ще видите записани две инструкции:

```
max-cache-ttl 0
pinentry-program /usr/bin/pinentry-tty
```

След като настроите **GnuPG** по горния начин, най-после ще бъдете готови да започнете да (раз)шифровате надеждно вашата поверителна информация.

Независимо от вида на шифровъчните алгоритми, които ще ползвате, вашето шифровано съдържание може да бъде форматирано по два начина – като прост текст или като бинарен (двоичен) код. Шифрованата информация може на свой ред да бъде запазена във файла или да бъде директно изведена в терминал.



Разновидности на поверителното съдържанието в (не)шифрован вид

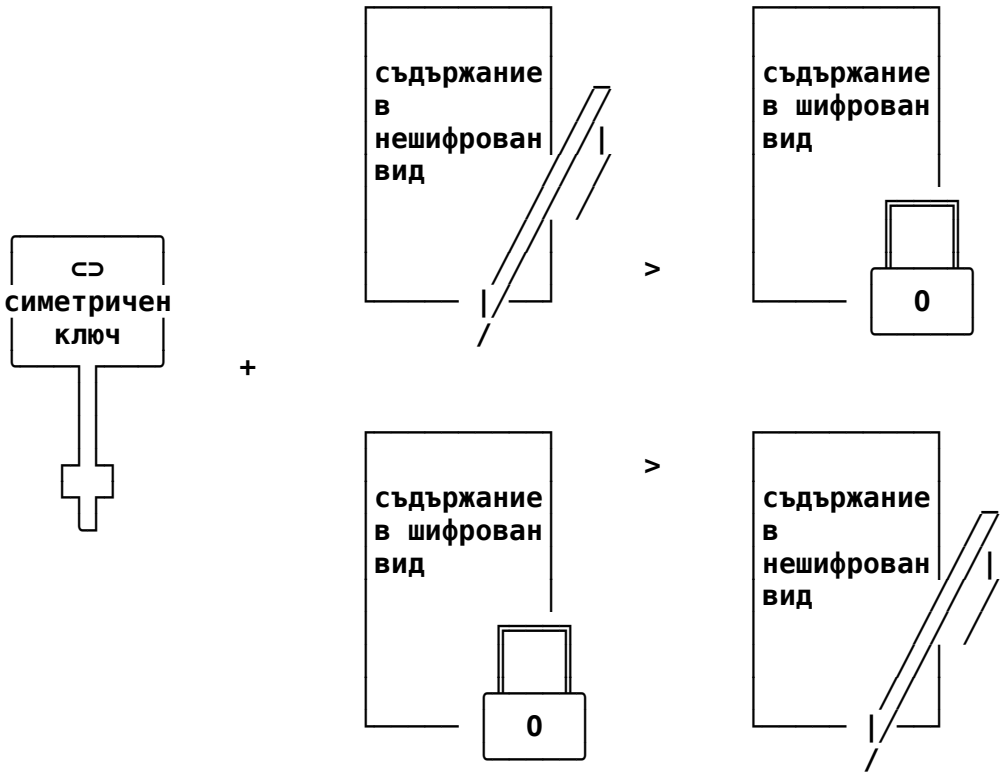
Шифроването като прост текст (независимо дали текстът се въвежда направо в терминала или е предварително обособен в отделни текстови файлове) позволява да изпратите това съдържание (без да променяте нито един символ!)

чрез обикновена текстова комуникация (например e-mail, instant-съобщения, поредици от **SMS**-и или дори в обикновено хартиено писмо с пощенска марка). От друга страна обаче този вид шифроване не е подходящ при обработката например на изображения, аудио- и видео-файлове, компютърни програми, форматиран текст или архиви. Естеството и обемът на такива файлове би направил простия шифрован текст неразумно дълъг. Поради това се препоръчва шифроването на подобно съдържание да става като бинарни файлове – което не може да се изпрати чрез обикновена текстова комуникация, но прави шифрованите файлове компактни и удобни за прикачането им (**Attachments**) или за предоставянето им във външни носители с памет (например **SD**-карта или **USB**-памет).

Препоръчва се да шифровате най-съкровените си тайни само във вид на прост текст, който въвеждате направо в терминала (без да копирате текста от предварително подготвени файлове). Препоръчва се и това – да не записвате поверителното съдържание в електронна среда, преди да бъде надеждно шифровано (което най-сигурно може да се постигне, когато набирате текста директно в терминал). По този начин ще ограничите рисковете, които могат да произтекат от инцидентно или планирано записване на вашето поверително съдържание в нешифрован вид. Например повечето текстови редактори правят регулярни (и невидими за обикновения потребител) backup-копия на обработваните данни, за да предотвратят нежелано изгубване на въвеждания текст, който все още не е записан. Дори да изтриете такъв файл, това не води до сигурно заличаване на информацията от паметта на съответното устройство и в някои случаи тази информация може да бъде извлечена изцяло или частично. И накрая – с работа директно в терминала ограничавате рисковете, които могат да произтекат от евентуално компрометиране на някое от потребителските приложения или графичния интерфейс като цяло.

4. Шифроване чрез „симетрични“ шифровъчни алгоритми

Както вече стана дума, **GnuPG** предоставя възможност за прилагането на т.нар. „симетрични“ шифровъчни алгоритми (**Symmetric Cipher Algorithms**) – при които един 'симетричен' ключ (или парола) служи както за шифроване, така и за разшифроване на съдържание.



„Симетрично“ (раз)шифроване посредством 'симетричен' ключ (парола)

Технологията получава широко разпространение през Втората световна война в лицето на прочутите шифровъчни машини **Enigma** на Третия райх. Тези машини действат на базата на „симетричен“ алгоритъм и предварително разпространени в свързочните звена на Вермахта секретни книги със 'симетрични' ключове (пароли) за всеки следващ ден от годината. Така, дори и когато съюзниците успеели посредством криптоанализ да разбият ключа за определен ден, това не им давало никакви предимства за следващите дни, тъй като за тях били предвидени други 'симетрични' ключове. Всеки следващ ден аналитичните звена на съюзниците трябвало да започват нова битката за разбиване на ключа – веднага след като прихванат първите съобщения, шифровани със съответния за този ден 'симетричен' ключ. И тъй като при „симетричните“ алгоритми няма сравнителен материал, на база на който да бъдат търсени зависимости и от там – аналитичните методи за разбиването на такива ключове е изключително трудно.

`https://en.wikipedia.org/wiki/Enigma_machine`

За разлика от „асиметричните“ алгоритми (където има поддаваща се на анализ криптографска зависимост между символите от 'публичния' ключ и тези от единица шифровано съдържание), при „симетричните“ алгоритми разбиването на ключа чрез анализ на шифрованата информация е много по-трудно. Разбиването на 'симетричния' ключ (паролата) може да се окаже неразумно трудно при достатъчно дълги и сложни пароли (както проследихме в Част **V**). Поради тази причина в Част **IV** препоръчахме цялостното шифроване на вашата система да бъде осъществено с **Cryptsetup** – технология, разчитаща именно на „симетрични“ шифровъчни алгоритми.

Можете да шифровате „симетрично“ ваше поверително съдържание като прост текст:

`$ gpg -a -c`

Системата ще влезе в режим на очакване да въведете парола (**`Enter passphrase:`**) и след като направите това, ще ви се даде възможност да въведете в терминала и самото поверително съдържание като прост текст. Можете да набирате текста направо в терминала или да копирате от предварително подготвен текстови файл (като вторият метод не се препоръчва при работа с особено поверителна информация поради риска ползваният текстови редактор в някакъв момент да запише резервно копие от набирания текст). Когато приключите с набирането на текста (или неговото копиране от другаде), трябва да укажете край на въвеждането, като натиснете **`[Enter]`** и после **`[Ctrl]`**+**`[d]`** (или два пъти **`[Ctrl]`**+**`[d]`**) – в терминала ще бъде изведено вашето съдържание (преобразувано „симетрично“ на база въведената от вас парола), в подобен вид:

```
-----BEGIN PGP MESSAGE-----
Version: GnuPG v1.4.10 (GNU/Linux)

jA0EAwMCwRw1fz5GHVxgyel5O3xIZiHjTrvI7KrqPQ/H4GVnG6VCM59AzrHIFLR/
XSUpg/b36arr0cyk0IJNVEKQjO9PiePToFiC1x1BT1iJqi94a2G4oB20Hj09pgf2
QPNEGHu7wUVyvpmPsPAqF7hRcvV2qQTFsn6fZ5xYXlxve5LyM06AcXJ4aBSP8b0j
CMvWKAAI2gQcCmkHyqrwFK3EXhVP9+psHE6rJAmjKxwQ2IviZTfdrRk92DgR74Vj
m8lGIS8xGSr1Vb+vFf89W82YNAEfkhHVPKLxX2O4UebvWBF26ZKvcTRumDVdrIxT
qfDAOUAiZuz6mMi9Die8/a7zjkKRAwR7bH4FK0iSd11pQFDv5V8/yI8aypse/jNB
CQZXi5ExkV3eLHBU9lKYfda4S5NFwey1TsR7pUNculgOGk2+e9h3zqZVjmQZM5ai
DhwPwj6d5p3OdARbbfqY6GyjPXnghHTKcZ9JtWWScm9rWbwfKFp9YaqwZBs5ZWsy
HX1KS5JH5SiVxxayJ++BXg3G5nPSyDSvNE4T1GYsqFXxe9w0TP1YAEyBwLy3Uq3a
HdHJSZr/I2BrwSpm5LblBE32dQvRiZmtp1O69snHHmZgsW5z90HIuyCoNfHG/jhB
uD8n2IYInJlilqWURGX22oKhaot/Cg1G0nLCbt9kMJop7Xypxm9n5YtyqVuJlBsS
tcE/lgm9hR2lEQOqPljRu0uMTlgCN4V1y2WwFDRyoyrs3KR6r9EVJA8w8iprMEqp
kMjzgzpXJrtkN03VRITH6geKUoM6j0vGhsCmgxD5EYBWBJUTvoqPkUE+s134u3E2
Yb0DVELrFUDq8mpdVzPC/xiWKUkKKmcIVkO/1WiQXcuAi3bbHXigxdzkeFwHRLf3
AmCjUSPIUl3n8eW55gUcLjX/qbJGsWGLK3h1nSYABkEvJ8zXNkpMbJj3/Jt6oZbt
BP02xo8bHxszUXyAuVzdMFbZkC+atflxV0+X4areNKfQ/NClAst2NNedc9sxCPye
ThG6O8HbPDs8j6EtH2TQOvndxW6k4aDbR1Z63QPlK8FxJet0Q341GkG2WUrMofZT
IpDOmBVazCei41NERBDMclPQ+bSSfSWf4ANV7e1dxlb/R9E4CHjuwmPNCAAiLG2S
62yofH+5Hq5jShyluDEJJjWKyeCV3HgrmXJxjFTxTJbeMzKYHs149cNAe41QMg+i
/FLWh5QKR5sbw3mkPQ+yC98j/96wRjG0FLcGb9HTf5l5a/WFXacfCz0QEP1NBcV7
Bd/TE32Tk7od+4ZOr0PrYe5wdyw243pawKBhoNe0ai2kv0bNG2kTq+3sLLrZ8LQQ
XlsoowZSF4LB5PXZI8tRNSbFWkXwyp5olwLRHvZtNEOiRQ==
=aWUv
-----END PGP MESSAGE-----
```

Можете да разшифровате „симетрично“ шифровано съдържание като прост текст:

```
$ gpg
```

Системата ще влезе в режим на очакване да продължите нататък, като въведете вашето шифровано съобщение (**gpg: Go ahead and type your message...**). Трябва да въведете шифрованото като прост текст съдържание (целия шифрован блок, без да променяте нито един символ!). Системата ще разпознае въведения блок като „симетрично“ шифровано съдържание и отново ще влезе в режим на очакване да въведете паролата, с която този блок е шифрован (**Enter passphrase:**). Когато въведете паролата и натиснете **[Enter]**, на база въведената парола шифрованият блок ще се преобразува и в терминала ще бъде изведено съдържанието, но вече във вид на разшифрован прост текст. Може да прекъснете **GPG** сесията с **[Ctrl]+[d]**.

Можете да шифровате „симетрично“ ваше поверително съдържание и като бинарен файл:

```
$ gpg -o Желан_бинарен_шифрован_файл -c Съществуващ_нешифрован_файл
```

(където под '**Желан_бинарен_шифрован_файл**' разбираме шифрования файл, който желаете да бъде създаден след като се осъществи шифроването; а под '**Съществуващ_нешифрован_файл**' – съществуващия към момента файл, чието съдържание желаете да бъде шифровано в бинарен вид. При изпълнение на горната команда на съответното място в системата ще бъде създаден указаният от вас желан файл със „симетрично“ шифровано съдържание в бинарен вид. Както стана дума, този вид шифроване се предпочита при съдържание, което не е във вид на прост текст (например изображения, аудио-визуални файлове, програми, архиви).

Можете да разшифровате „симетрично“ шифрован бинарен файл:

```
$ gpg -o Желан_нешифрован_файл -d Съществуващ_бинарен_шифрован_файл
```

При изпълнение на горната команда (и въвеждане на съответната парола) на посоченото място в системата ще бъде създаден указаният от вас файл, но вече в разшифрован вид.

Когато (раз)шифровате бинарни файлове, трябва да имате предвид, че указването на идентични наименования и идентични местоположения на съществуващите и на желаните файлове в системата ще доведе до изтриване на старите файлове и създаването на нови на тяхно място.

Сред най-разпространените „симетрични“ шифровъчни алгоритми, поддържани от **GnuPG**, могат да се посочат тези:

IDEA (**International Data Encryption Algorithm**) е блоков шифровъчен алгоритъм, утвърден през **1991** г. като алтернатива на компрометирания алгоритъм **DES** (**Data Encryption Standard**) на **IBM** (който до тогава бил признаван от Националното бюро за стандарти (**NBS**) на **САЩ** като надеждно средство за защита на некласифицирано съдържание) и до **1999** г. продължава да се разглежда като един от най-сигурните „симетрични“ алгоритми. **IDEA** поддържа **128**-битови „симетрични“ ключове и разбива поверителното съдържание на **64**-битови блокове, в които то се подлага на **8**-кратно последователно преобразуване.

3DES (**Triple Data Encryption Algorithm**) е блоков шифровъчен алгоритъм, при който трикратното прилагане на вече остарялото **DES**-шифроване (всеки път с различни шифровъчни ключове) води до по-трудно разбиване на шифъровъчния алгоритъм. **3DES** продължава да се използва масово например в системите за електронни разплащания, но от **2016** г. насам неговата надеждност се поставя под въпрос.

CAST5 (**Carlisle-Adams-Stafford-Tavares**; известен още като **CAST-128**) е блоков шифровъчен алгоритъм, при който поверителното съдържание се разбива на **64**-битови блокове, шифровани чрез променлив шифровъчен ключ и **12**-кратно или **16**-кратно преобразуване посредством шифровъчните мрежи на **Feistel**. **CAST5** е създаден през **1996** г. и е възприет от Ведомството по комуникационна сигурност (**CSE**) на Канада за ползване за правителствени нужди.

BLOWFISH, създаден от **Брус Шнайер** (**Bruce Schneier**) е блоков шифровъчен алгоритъм, при който поверителното съдържание се разбива на **64**-битови блокове, шифровани чрез променлив шифровъчен ключ и **16**-кратно преобразуване посредством шифровъчните мрежи на **Feistel**.

AES (**Advanced Encryption Standard**), както проследихме в Част **IV**, е единственият официално признат от Националната агенция за сигурност (**NSA**) на **САЩ** като алгоритъм за защита на класифицирана информация до ниво „Строго секретно“. **AES** поддържа **128**, **192** или **256**-битови симетрични ключове, които водят съответно до **10**, **12** или **14**-кратно последователно преобразуване на поверителното съдържание в процеса на неговото шифроване.

TWOFISH е блоков шифровъчен алгоритъм, при който (както проследихме в Част **IV**) поверителното съдържание се разбива на **128**-битови блокове, шифровани чрез **128**, **192** или **256**-битов шифровъчен ключ и **16**-кратно преобразуване посредством шифровъчните мрежи на **Feistel**. За разлика от **BLOWFISH**, при **TWOFISH** се прилага псевдо-трансформацията на **Hadamard**, насочена към засилване дифузията на шифрования текст и от там – към затрудняване разбиването на шифровъчния алгоритъм.

CAMELLIA е японски блоков шифровъчен алгоритъм, при който поверителното съдържание се разбива на **128**-битови блокове, шифровани чрез **128**, **192** или **256**-битов шифровъчен ключ. Надеждността му е призната към **2003** г. (и приравнена на тази при **AES**) в рамките на Новата европейска схема за [цифрови] подписи, интегритет и криптография (**NESSIE**) и паралелно с това в рамките на Комитета за криптографски изследвания и оценка (**CRYPTREC**) на японското правителство.

По подразбиране при „симетрично“ шифроване **GnuPG** прилага не особено надеждния шифровъчен алгоритъм **AES128** (фаворизиран от **NSA**, със **128**-битови ключове).

Можете да укажете да бъде приложен и по-благонадежден „симетричен“ алгоритъм:

```
$ gpg -a -c --cipher-algo Алгоритъм
```

... или респективно:

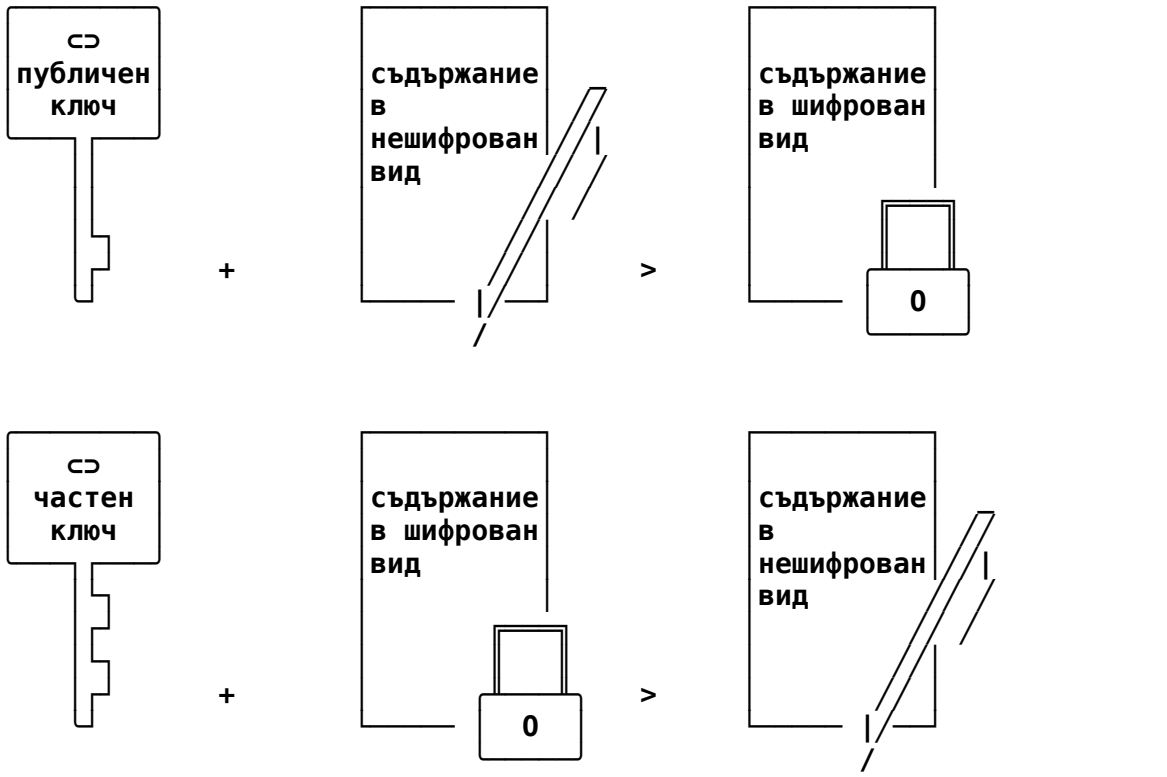
```
$ gpg -o Желан_шифрован_файл -c --cipher-algo \
Алгоритъм Съществуващ_нешифрован_файл
```

(където под '**Алгоритъм**' разбираме предпочитания от вас „симетричен“ алгоритъм, следван (ако е приложимо) слято от число, обозначаващо дължината на желания шифровъчен ключ – примерно '**TWOFISH**' или '**CAMELLIA256**').

Впоследствие при разшифроване е възможно да се наложи да въведете ръчно параметъра '**--cipher-algo**', ако системата не го разпознае автоматично. Добра практика в тази връзка е да си създадете навика да помните (или да си записвате) точните параметри на извършваните шифрования.

5. Създаване на двойка „асиметрични“ ключове

GnuPG предоставя възможност за прилагането и на т.нар. „асиметрични“ шифровъчни алгоритми (**Asymmetric Cipher Algorithms**) – при които двойка от еднозначно асоциирани един с друг 'публичен' и 'частен' ключ служи както в процеса по (раз)шифроване на поверителното съдържание, така и в този по цифрово „подписване“ на файлове и криптографска проверка на „положените“ подписи. Въпреки, че „асиметричните“ алгоритми са по-ненадеждни от „симетричните“ и работят по-бавно при големи обеми шифровано съдържание, са се наложили в практиката поради решаването на ключовия въпрос – установяване на надежден канал за комуникация между страни, които не са се срещнали предварително, за да обменят по сигурен начин един общ 'симетричен' ключ (или парола). Вместо това при „асиметричните“ алгоритми се ползва 'публичен' ключ (разпространяван публично и достъпен за всеки желаещ), който обаче служи само за шифроване на съдържание и за криптографска проверка на „положени“ цифрови подписи. Разшифроването на съдържание и „полагането“ на цифрови подписи е възможно единствено чрез 'частния' ключ, еднозначно асоцииран с 'публичния' и съхраняван в „симетрично“ шифрован вид на сигурно място в системата на своя притежател.



„Асиметрично“ (раз)шифроване посредством 'публичен' и 'частен' ключ

За да можете да приложите „асиметрични“ шифровъчни алгоритми, на първо място е необходимо да създадете двойка от еднозначно асоциирани един с друг 'публичен' и 'частен' ключове:

```
$ gpg --full-generate-key
```

GnuPG ще влезе в диалогов режим и ще укаже да въведете стойности за вида на двойката ключове (най-често възможност '**1**' е най-подходяща); дължина (сложност) на ключовете (по-дългите ключове са принципно по-сложни за разбиване, но работят малко по-бавно, като максималната поддържана дължина към този момент е **4'096B**); срок на валидност (при стойност '**0**' ключът ще има безсрочна валидност); наименование, с което желаете да се асоциират ключовете (не е задължително това да бъде вашето реално име); адрес на електронна поща (желателно е да имате контрол над посочения адрес, тъй като той ще се асоциира с ключа и е възможно да получите съобщения на него);

допълнителен коментар (кратка бележка, ако желаете да оставите някакво допълнително пояснение). Системата ще изведе указаните от вас стойности в подобен вид:

Наименование (допълнителен коментар) <e-mail@example.com>

и ще даде възможност да нанесете корекции, да потвърдите стойностите или да се откажете от започнатата процедура по създаване на двойка „асиметрични“ ключове:

Change (N)ame, (C)omment, (E)mail or (O)kay/(Q)uit?

Ако въведете '**O**' (от англоезичното '**O**[**kay**]'), системата ще укаже да въведете парола за „симетрично“ шифроване на 'частния' ключ и ще започне създаването на вашата двойка ключове.

Тъй като криптографията разчита преди всичко на случайни числа (каквито компютърът не е способен да създава със своите предвидими алгоритми), процесът разчита на относително случайните събития, които протичат към този момент във вашата система. Ще бъдете поканени да подпомогнете създаването на относително случайни числа, като извършвате каквито и да било произволни действия с компютъра (движите мишката в разни посоки, отваряте файлове и програми, пускате музика или видео, въвеждате някакви символи от клавиатурата). Препоръчваме обаче да не въвеждате произволни символи в терминала, тъй като някои от тях е възможно да се интерпретират като команди и да протекат процеси, които всъщност не желаете да стартирате. След малко процесът по създаване на вашата двойка ключове ще завърши и те ще бъдат записани в техния „симетрично“ шифрован вид на тези адреси:

/home/Потребител/.gnupg/private-keys-v1.d
(двата файла на 'частния' ключ, служещи съответно за разшифроване и за цифрово подписване); и

/home/Потребител/.gnupg/pubring.kbx
('публичният' ключ с относимите към него метаданни – сред които наименование, електронна поща, допълнителен коментар; дата на създаване, срок на валидност).

Сред най-разпространените „асиметрични“ шифровъчни алгоритми, поддържани от **GnuPG**, могат да се посочат тези:

RSA (**Rivest-Shamir-Adleman**) е първият публично известен „асиметричен“ шифровъчен алгоритъм, представен през **1978** г. (през **1997** г. беше разсекретена информация, че британските тайни служби са разработвали подобен алгоритъм още към **1973** г.). **RSA** е базиран на математическата трудност да бъде факторизирано произведението на две големи прости числа, щото да се установи кои са тези две прости числа. Днес **RSA** е може би най-разпространеният „асиметричен“ алгоритъм.

ELG (**ElGamal**) е представен през **1985** г. и включва „симетрични“ ключове, с които бива шифровано поверителното съдържание, а след това тези ключове се шифроват „асиметрично“ (т.нар. „хибридна“ криптография с едновременното участие на „симетрични“ и „асиметрични“ ключове). Този подход съчетава високата надеждност и бързина на „симетричното“ шифроване с възможността да се приложи 'публичен' ключ, характерен за „асиметричното“ шифроване. Слабост на **ELG** е възможността шифърът да бъде разбит на базата на шифровано съобщение, чието разшифровано съдържание е станало известно.

DSA (**Digital Signature Algorithm**) е разновидност на алгоритъма **ELG**, утвърдена през **1994** г. от Националния институт по стандартизации и технологии (**NIST**) на **САЩ** като Федерален стандарт за цифрово „подписване“. Алгоритъмът е уязвим при разкриване на дори малка част от символите в 'частния' ключ или при липса на достатъчно висока степен на ентропия (случайност; непредвидимост) в символите, чрез които е генериран 'частният' ключ.

ECDH (**Elliptic-curve Diffie-Hellman**) е вариант на протокола **Diffie-Hellman**, базиран на т.нар. елиптични криви. **ECDH** позволява страните да създадат и разменят помежду си чрез „асиметрично“ шифроване двете половини на един споделен секрет, който да им служи като „симетричен“ ключ или да способства за допълнителното създаване на такъв „симетричен“ ключ, приложим в по-нататъшната им комуникация.

ECDSA (**Elliptic Curve Digital Signature Algorithm**) е разновидност на алгоритъма **DSA**, базирана на т.нар. елиптични криви върху крайни полета. Това позволява значително намаляване дължината на шифровъчните ключове при запазване съответното ниво на сигурност. Подобно на **DSA** обаче, **ECDSA** също е уязвим при разкриване на дори малка част от символите в 'частния' ключ и при липса на достатъчно висока степен на ентропия. Заради такива слабости през **2010** г. е бил разбит например 'частният' ключ на **Sony** за софтуера на игровата конзола **PlayStation 3**.

EDDSA (**EDwards-Curve Digital Signature Algorithm**) ползва разновидност на системата за цифрови подписи **Schnorr** (явяваща се сред най-семплите и все пак надеждни системи за цифрово „подписване“), базирана на усуканите криви на **Едуардс**. За разлика от алгоритмите **DSA** и **ECDSA**, при **EDDSA** цифровият подпис е производна от 'частния' ключ и поверителното съдържание – което намалява уязвимостите от разкриване на дори малка част от символите в 'частния' ключ и от липса на достатъчно висока степен на ентропия.

По подразбиране при „асиметрично“ шифроване **GnuPG** прилага надеждния шифровъчен алгоритъм **RSA**. В случай, че желаете да ползвате някой от другите поддържани „асиметрични“ алгоритми, ще имате възможност да укажете това при създаването на вашата двойка от 'публичен' и 'частен' ключ. Веднъж създадена, двойката ключове ще действа съгласно избрания „асиметричен“ алгоритъм. Имайте обаче предвид, че някои от поддържаните от вашата версия на **GnuPG** алгоритми е възможно да не се поддържат от системите на вашите довереници и това да затрудни комуникацията с тях. В този смисъл не е излишно винаги да разполагате с една двойка „асиметрични“ ключове, базирани на най-широко разпространения алгоритъм **RSA**, като няма пречка да прилагате и по-специфичните възможности, след като обаче съгласувате това с конкретните ваши кореспонденти.

6. Извличане и въвеждане на 'публичните' ключове

Всеки, който притежава определен 'публичен' ключ, може да шифрова с този ключ съдържание, предназначено само за притежателя на 'частния' ключ, еднозначно асоцииран с този 'публичен' ключ. Всеки, който притежава определен 'публичен' ключ, може също така да провери цифрови подписи, „положени“ чрез еднозначно асоциирания с него 'частен' ключ. Докато 'частните' ключове се пазят в най-дълбока тайна, 'публичните' ключове – тъкмо обратното – могат да се разпространяват публично, за да се ползват от всеки желаещ да ви изпрати шифровано съобщение или да провери истинността на „положени“ от вас цифрови подписи. За да можете да публикувате вашия 'публичен' ключ и по този начин той да стане достъпен за трети страни, трябва първо да го извлечете от вашата системата:

```
$ gpg -a --export Ключ
```

(където под '**Ключ**' разбираме каква да е достатъчно разпознаваема част от информацията, с която сте асоциирали вашата двойка ключове (например наименованието, адреса на електронната поща или допълнителната бележка), или някаква разпознаваема част от тях. Внимавайте за случаи, при които няколко ключа са асоциирани със сходна информация – защото в такъв случай може да извлечете информацията за тези няколко ключа в един общ блок! В резултат от горната команда в терминала ще бъде изведено текстово съдържание, което в своята цялост (без да променяте нито един символ!) представлява вашият 'публичен' ключ.

Като пример (и за да добиете визуална представа) представяме нашия 'публичен' ключ:

```
-----BEGIN PGP PUBLIC KEY BLOCK-----

mQINBGTOSbEBEADKgDNeZXi5fC63/Sy6pilcQLEhnHV7XgG/NTRlOFi93jxrVaE0
3bO4DZds0T0uZa6ycHtLPNN6ftWsfG4LD6uESXxinVJ8eZJXckBJ7JoeUysz4fx6
xuR4uEaMc9ac74ySdXUc4Jl6R1yd1OM/VrBerDgT9NNntTws5UVNyJXkNzmqCzg0
ncQG+Xc+oRr+PbBMm75ATA6E9BL0kzrqUansyxqbVW5z85/MgQxNlLt0VIkQDFoA
8vbDjpMejOUiy3OEqR5hGCCzl/xF2o/nNuBiiKD9lNjZWjSPNm2UiEfHtQC/6PHH
MMvoNeX/+KHmdZ2NmbZzilNrV+VB7mKecVJddhX20UOkt+gYfbmlGptoSO2GaQX8
xCI+wvPjsawif0jLTigr74zF6vnTugQkvdtw7CX6f+clPu90qYsxqhk+MbpB6CEO
Qn+A4CyehP4oXoJcpYjSQ/vAh506Hmn5Ml4tJLDGXdPIobiQyRvdwnOWoufE9Gzp
6KJSej1eUtyaobqJnFPvF2023z6EEX4ATC1hPvqxAA0kwcbI5fQ5QChSgUAlRSn4
EXvzR8mtR0grqP5o5nF7iwXfqqY6oX7hlJtR2KsFhKlRXi15xHvUIxpsHNQL8fWt
rL+IpAAO6HIUjUbAsymR2kLUqDwy7ru1wcpnVqoDammo/CKKAL/KL+bKbwARAQAB
tEV3d3cuQWR2b2NhdGkub3JnICjQsNC00LLQvtC60LDRgtGB0LrQsCDRgtCw0LnQ
vdCwKSA8YWR2b2NhdGlAZ214LmNvbT6JAk4EEwEIADgWIQSYAP3DPwTsBxrRQsqP
F+7Re55TowUCZM5JsQIbAwULCQgHAgYVCgkICwIEFgIDAQIeAQIXgAAKCRCPF+7R
e55Tox8KD/4zUAb3VNN71otaEQ3ht3w2VPlJAeClJEOxl4z2wTK5f6uQLsWGuXzx
Dpls5kt7Cmmc9KZmXMKSGEvwVGEGYCJiOxfItG6jWmMVPWkQeAzWo5kegVpC40X4
01GT6NyMRWVIAdU5shP/Z1JTQQ8oCDQEVqQ7jAb6IDJZVjiNVxCL3U/eA8wmcRum
eFaUBELEQ5oBXIIe7VGu7L0ZM327Q7WTFA5wnSAkNky4Y86I9h/+BsgLxejAnXDD
aLwfDgUQV1q6pHwF7X+2BtacFM+bSyFKIk0p4Sn2ITaoSrXCS8mJS7/CkEklP/Vr
+iM8dZbMsMyt66oFtM6hZ2ExkDBO/ViRVDAZgzdY0CZRNDoU8GWEitVwT7juNquX
//+GP24OWVydw78pK94fu53gELw9U/TkZfGhlJTyM+kbfMrmqC4up3xnl2Aio3ic
r1T94T6glNTfiJHOkY6gQ45ZgdzMkojOdsQcb7eYPUyCWuKYIA8potOpIxxMzXk9
jf4RJm165wquAsWc1lFfeuJX/PmuYpRkNaLKrXpZ56mtjNV6NmQSR4UbLGKKuHe+
xlY9IVR3aZIrvL6VJjlndIQ3RiarbyuPz601NBoQdB4GYQU0GYbd8MZgrDk8KYoJ
GnSJv03O9+8xl+BhD9jiyxG20NmtLclhCdCfYfyVHgE6DkWvodni27kCDQRkzkmx
ARAAp6ob6+MYSRey55cwSVfNuo5267VkmoqHo74gn52nULfUzdYfrSGs/uCxH9+m
PUoPMynwBHTbyBxM9nTsjMYBtYkrR3EXIHq6DWR8wryEPRL81W/vaCbkPGH/YZvx
iszZvzpJr2HCnl85Oi3BG8zTy6qaU6IymlziLXeDJPG44PpGHuSdfoHKpK+pu3RR
3DqijCM9VZweK9eCBMOX//SfWSYHLUGh2WZl18yG9ryPT8ldOMN23p7yeyoktoKE
db7D9p04VMG4p7z2NK6r3U86nvr0lxakLOxqQUkHL0yfBo717Cll4ERBIyj9gqn3
CUAG2hMGtmEdY/KwBMdzBw2AwVcAjeVNCzj+b7/XIIa6Eq+1lsF65btoNAD/7moe
5cixc66xcJjC9ZPecU3mFeQqCgR2pRIoiUxI3744a74/D6rZALdkXSotpuwG3rEP
5LUMwWkmlXJD0yWxrcSP7k0p22CgLGzzqElKPiZDpMOj0bsf36hWfYsnnOZeNqnx
mFuQVLaaYkxcGNI0SpTUiTSnA/cPfm4tecq+mDJ5QvGSJApoZ0iK7udBdfp+q1eB
6L6qoylWtuZC0sxfJh/OecUerVs/KKOntPNfGDK8im7CIcwz3q4p1IRiPLjTa2yU
HrlfHW8GU38SeejV9yPNyOxC5bAP01yjXPuX6DyCDyDTpJUAEQEAAYkCNgQYAQgA
IBYhBJgA/cM/BOwHGtFCyo8X7tF7nlOjBQJkzkmxAhsMAAoJEI8X7tF7nlOjavoP
/j2bzrgFbAxO/RslOKxUzdo6XZred7RLWPgpENBrgX82AA8C3lkpS2tyWXmd2MB7
83cLOAJud0ivR4L+V4XYXM4RXy4rsvdUIQvRG87Q7JK+QPzUMaVUb3izQARno3z4
3LHPIOmXmslil6y+/xT7OTKFKD6gS4jF2OTQnkekfUkulgKjNneqZqPhRACTLo34
AvCzrlg8dpSY1Rj3Y6xXL4cp9DR0lbkdtdkGa6/NUFwtJLPWrNx7iBpYtIZ+2WXg
68ptC32dLx6L2j0w27Atxyt58ZcdBHJjbos79wEwm3QgGO9p3LOls8/MT1LeoYDe
gUe+ZE/0fp5cEKpyI/YODX8ahN2AhTppU0BShV8KV41W9wfnRGBnnJi8vdI3fDTn
SXYI22ifdnpLT/yzmJ1WVwihJxPH/SotqGkIEtePEEuL28Y1azrN6dLkVnMwHO0l
3FcY6dZKu/m0KDgwIyCfYzTdbJ5q7jT86BwvWM6yIa/TZGMcgRQWnoIvIT969pqs
0D7vfkeLPYIJ8a5Q/6LrjiJZ53AfTw6b+jM0genbze7yj6f2aGInsmbdf9l8OGin
6F9StyirRac4EYpYNOP2iCirIcINO4Ve7jc3uEvjMatEn+Jzb2YCsF6kRp2ZUlFR
Z8gs3szzYhvBopBM+XqQRPfQCe0F/EcpbMT54RzqAarI
=hqZP
-----END PGP PUBLIC KEY BLOCK-----
```

Публикуването на 'публичния' ключ (без да променяте нито един символ!) може да стане във вид на прост текст (както сме постъпили в примера по-горе); или във вид на отделен текстови файл (както сме постъпили в сайта **www.Advocati.org** с предоставената възможност за сваляне на текстови **.pub** файл с нашия 'публичен' ключ). Можете да създадете вашия **.pub** файл така:

```
$ gpg -a --export Ключ > Файл.pub
```

(където под '**Файл**' разбираме наименованието на файла с вашия 'публичен' ключ, който искате да създадете; а '**.pub**' е типично разширение за файлове, съдържащи 'публични' ключове).

За да можете да ползвате един 'публичен' ключ за шифроване на съдържание и за проверяване на цифрови подписи, той трябва да бъде въведен във вашата система, откъдето **GnuPG** ще го прилага при осъществяване на съответните криптографски операции. Създадените от вас двойки „асиметрични“ ключове са въведени автоматично в системната част на устройството, на което сте ги създали. За да бъдат ползвани на друго устройство обаче, те трябва да бъдат въведени в тях ръчно:

```
$ gpg --import
```

(при което системата ще влезе в режим на очакване да копирате в терминала интересуващия ви 'публичен' ключ, във вид на прост текст). Когато сторите това и натиснете **[Ctrl]+[d]**.

Можете също да въведете ключовете от свален на системата файл:

```
$ gpg --import Файл
```

Който и от горните методи да приложите, ще бъде изведен подобен резултат:

```
gpg: key 8F17EED17B9E53A3: "www.Advocati.org (адвокатска тайна)
<advocati@gmx.com>" imported
gpg: Total number processed: 1
gpg:               imported: 1
```

От горната информация научаваме, че съответният 'публичен' ключ е въведен (**imported**) в системата. Сега вече сте готови да шифровате съдържание с този 'публичен' ключ и да проверявате цифрови подписи, „положени“ с 'частния' ключ, еднозначно асоцииран с този 'публичен' ключ.

Можете да въведете 'публичен' ключ и от графичния интерфейс, като просто натиснете върху иконата на текстовия **.pub** файл, който сте свалили примерно на своя Работен плот (или изберете съответна функция от контекстното меню в графичния интерфейс). При това (ако **GnuPG** е инсталиран във вашата система) ще бъде изведена графична индикация на екрана, че въвеждането на ключа е извършено. По редица съображения обаче препоръчваме да работите в терминал и да ползвате командния ред, вместо да се доверявате на графичния интерфейс.

7. Извличане на 'частни' ключове от системата

В някои случаи може да се наложи да извлечете от системата и вашите 'частни' ключове – например за да създадете резервно копие, за да мигрирате към друго устройство или като реакция срещу пробив в сигурността, за който обаче сте сигурни, че все още не е компрометирал шифровъчните ви ключове (в противен случай (ако 'частният' ключ е компрометиран, дори и да не е извлечен от системата) се налага незабавно да анулирате съответната двойка ключове и да ги замените с нови – както ще изясним в края на настоящата част).

В случай, че се налага да извлечете вашите 'частни' ключове от системата (не правете това, освен ако наистина е необходимо!), трябва да сте преустановили всякакво свързване с internet и да сте взели всички необходими мерки за обезпечаване на вашата, на устройствата ви и на извлечените ключове физическа сигурност. Процесът трябва да бъде

организиран така, че без излишно бавене да въведете 'частните' ключове отново на сигурно място, като заличите по сигурен начин направените временни копия. Препоръчваме по възможност да унищожите физическите носители, върху които са били временно записани 'частните' ключове при тяхното извличане – защото (както стана дума в Част **VII**) простото изтриване на определена информация не я заличава напълно от устройството, на което е била записана; и впоследствие тя може все пак да бъде извлечена от там изцяло или частично. А в някои случаи дори една част от вашия 'частен' ключ може да доведе до цялостно компрометиране на шифровъчния процес.

Можете да извлечете вашия 'частен' ключ от системата така:

```
$ gpg -a --export-secret-keys Ключ
```

В резултат от горната команда в терминала ще бъде изведено текстово съдържание, което в своята цялост (без да променяте нито един символ!) представлява вашия 'частен' ключ. За да добиете визуална представа, представяме примерен образец на един несъществуващ 'частен' ключ:

```
-----BEGIN PGP PRIVATE KEY BLOCK-----
Version: GnuPG v1.4.10 (GNU/Linux)

lQc+BFoTanYBEADANUA2uCzai23Fw4XHZQf9h7cx9sg8t++1/du6MGq0+Ugb16yu
PPQbtpcyYbcXi7gTb+IYFtqZor7FYUTEzC3chU2Naxu5CPb+4wotaHrcRYLTmLw+
volURJngcuXdarLZnLw8LYpXxsS4Fvun6rk4E2W3qyhTvkBTyS5WvtbwUmDRxlxO
oTed62rWNDRkxS4rAmvbskp2OdwV33wIWTq/ws7M79Ag0HJPeFat74jpRuwfrGbP
Ay8xbg2CyVfPP8yqipTl7h3NmuUAr0C/QWg6JNTUFfap1/5SAfAckvMmCogj/M0D
CNZLUwcDCnLfYksfmFBElWOyBp/gesquT7OmrbTYIfY+W1gACDwYmPoha1Oa38bB
hnTIqakU1MsTe0M5hqesqZE0cJcsDlzAKDSa+6pTNWyOY9Gr59gmmOl/ISCfINtz
keW/6uw6cALH4BNDFaCZR1fHHE7erTTZZ8kvj0IUa+BD+0NbOa/+1GOBKg+d5VhF
8O1m6TOBih1Sug1rkX0HpsyFuWctOcyi8ZOr4G7kEL3XjqLB1VjJvcYZPzgHD1Ta
ZGhQfpamyudEuoGts1SvfFGr3NiUgxWK1wcLSq+JEvnjQe/42+Gj7Ve/FdCZPXjm
tMOAsya/3BDR9uVCuDx1QsKOb1GnktpN7TU/j4FiIPnjoI75dsqLgNGkVwARAQAB
/gMDAuQwxE1bzjEWYMkku+ZO7yb4FW4UDZhElvv+wrxyKOu2gxHhUFGFWTEKtM8H
Rnq81za5Ux+W1QqJX2zzEK9bpLQN54yTWa1CXmRDHtR9/G+CjJfAkIF+cl8oibNF
2fpbXRKabBzdL/4UvmAoIJ61EwyXaZrywqBxKqYh6d2wD3QrUEXFgW8kKZDMzgWW
Y7MIHuE8XxNXHezOO0vv2pEQLrXG1pL2ZkH4ZQTKd5247MAIW9s+unyAkZt/yK17
ibp6TW359FpiswqmUaGdXwsnRf2bxEOCwpEzbnNlB2vxJU51aM53cu2Js6GRruPU
ws/WLBcYsMEtm1Xk+3JbKx6jgQu/IWQyLn+I0M3dX3RHH3HGj6nrsKJkaBsI4hMm
oebV0lefEGghQBOLH05wC76C3CzSNtNVWHvNW5+d1NfdYf3DREFtWufCO4h99brh
swLrN5iJsNftxtm6yZMUBccTB2ZjsQXORvraPa5/rOMU30FCv0QyDXJfANOo/o/v
wAx0i5N/gMxLr1O0pS/QFFyRONSzAAUtQHk6AroyDlZjb+tMwxEpyhQJbqUdXkCw
BWVJTJAPGflac91ML3k4H3F3vs1eIb6wENsxa5qYQxXKXAj9lluUtKkPNoGy1iN6
ILkOBP2IqPS0K+ToIGYQ7aDzRmCX83t102SBMFarv0AOBTM8ZMmA3UFEb2QyOsny
Aq5mV2k2yR9oP8NuNilEkkNk6+i93nliTnH2Z93ydNLuwodh2gjS2NIYc4zyX+t/
/8XCN6Gln46SB4z+ZkqUAqLlVF0gzbrdVVWKk9Y96eT0LK85/8C5gjqr2LKfO0zr
ncyQ8wwOic0L721PrR2ycmw4TR2lmvCK7kGqx39YLonzP1aMqTZrH64iJYsyPZnv
Z5xbkKF0DFWQ2i6MphqwKdtfjJuongr2VDvoCNu8LL0gQvt0FK2zH5YUAwfUiRSk
XF4Yu5knfybTpJpAkGeZeJKnCj7/oxjWGyCDMbnKqpgN3RtUzfpNyqYnAI1BxzEz
WODe0h+6hSbg0kfVokRptYHil/OYEKBbQ5IG3Ss4fBy3fWqrlRNEL8D5/wi4+jzJ
C69RcwDpfgqy437Wc/wGiYl07lQ9kxkp/tWKlMxwzNf3NBNpwSwXG9L23cXvcHzO
NU1q/urkTewc4T8hp2T5gjTzipHlzz+umCXyR1uRvG7pKicO9/UDYxmKCzBeTASc
4Fps0xVG8ZjfFf4x9BpLGRPII9NlATYpLEbW/Ean+6LeF2dY9RSR36FLXtQi7UKN
TsTQqdighcSuVVmcqGaOR241u0wKgwiJmWppw3+9Ivzk/8U+fvj9kOWIBu+xTMy5
9MrSb0XV3yJA97+DcdmPZeRkAL60U8XVP3a0Wst6dyDtAOai+dZcfVnGwzx/Tmkb
0pfqpA0kntMySLTTcKev12OuHy8yr+lfz2Di6I71VxPFdJ+ZO9gdlV8esQWUwaFC
+GxxVmDrRodI4Qd0uzpaR0s51oeXHSrRCHK/a4navni9+J+02xHm4F53wnMS8aXx
0TSQER59XdOTpK7AEraXVm8PplNEjZndeSCHWr7fjTbeo6W+AoR3MF3h80y35qks
HdBK/BEGk8pUIlrbUemjE3kG+X1h06tL++t8UcWDV7ne/lYsJVIpmO/F98yOB6LR
```

xDNFooZu0p2tzWZ3vnTCjRba4SvEFINIyMGzg7+uJvGMaTixLbimICY64sHyMZo4
xfkLyuK58BrAEucg/EsVC5gKU1a285xPK5QAuZYFlWBdtEzQn9GA0LjQvNC10YDQ
tdC9ICfRh9Cw0YHRgtC10L0nINC60LvRjtGHIChFeGFtcGxlIE9ubHkpIDxlLW1h
aWxAZXhhbXBsZS5jb20+iQI4BBMBAgAiBQJaE2p2AhsDBgsJCAcDAgYVCAIJCgsE
FgIDAQIeAQIXgAAKCRACaXbfgK4z12X3D/927zlN7kQskF1L8L27LrKwug/n58n9
HBv2ZlUlKnlmxqMW2AleQlnsxlv5NacjQz33of0Rp0vRmkeHgvMgDlOnUCX43gAp
13XlRX0w21nOGEYtDxArmtXhXxcI4b3UY6q+uSaCCeLYY2wH0GgahdPI6WJtgdMv
kuAwAqxNIxlhhcBmQU9BGqrSDQXaUN/reTbZby2zdV3pj5rmyJhrhP7sJ3MKnk+A
ibzmtUWH+D4Stqhb7ZKOkvZOUeEMTIxwALjwiKtgkSNpEcPvdjSs7vA6UwYFy3R/
oCJRKKbm/IKHZ5XxdPjVoqgz17dCFjUCML8Pwo6G186uOL7B/O5LmTSyTXtSPPox
mb7cfi1OoGZtGmhoiF1Iw8jXLHIIl+L00QiPtxWnGDIQPnTG8XhgcXaTi02uUdb1
JbI8kncDswYvOBLTCEmnc2RQfa0X0DFZ5SWZqq10l8ucQwtDYImpf2x60Xy6ztzg
VfcFZHeewELWZ7wP3wqQbRFTZVUTr4Pnfr4Q+egdvT6/oo5BH3eT1jo3EWnfcV7e
PrawDSVeh7CIP3LsyzxmT4FlwDXu/YrGz6bi6C+BQDq5GlQ5hRJQmXs+8MhzQxYB
a41URuO1R42vlGNapoqK0ftoxfCgBo9FeOOmP9Q4X/JEMSlFZ/FuxNigCOz6NlQL
vIgbaAkMUvPCI50HPgRaE2p2ARAAouqxdKJ/tgWHyj2/2QWNfQo75Mu/ZfEPG4ah
UWckbkauPxMRIi0L9P82Mkthc99ca81bNtiexDDrE1dgZoUMw0feT2RHQIn8NCZF
NtbASg468HJbiv5JFU5UnqAxvWx2jMUJUU1wJ6QZxRIjZCTyUWaVUS4DsfBiyM0u
f3pIJOsrOwGmKkeqE+faSlw/zePkQ8QHX/YPgwU7b/oqwfh5IjUm7TueCFstPLi3
qWvNc8gBoH3YfjP2CX5FBPXvZOdLfmXA2p2nVH74XOnJ9lborQWzH10P5fEb6mew
4HayP+FbXp0F1EceHSl8aqrn8V2ViWFkPc+tnKwlbj4MJnJV20ZicFiRuvgjqlhj
SDUZ+uJ/8dyqZY3B9QKTigps9pFQZRz2+QYZOHKCKQQ6Ax95BLAeqyrwyUkD+rGp
rGnOlGX0BpqjJuTooV28dfhMa0pxq6B9u2ix7XSwY8+Fgon1B0od3xhl3QDes6aV
ZjtehG4VN3BR0eqSflIBIt8oQwUFQUHHftxK+JwVKF36ar411m7WiCMAKMYwzEH5
LurS6c99AxzpiZN3Ht5P73z9IpnATmTtesj6tL6STYct9jWXFvlKbLeOOn4KMihR
Ty04eUtEnhenwGCz1U8LaGaw71eX1END192b8LO0CS4B6erV337fTDd5PsqN1u/S
iFSaISsAEQEAAf4DAwLkMMRNW84xFmAdZG/PCEpDjQilhi3472Iwi/do3NYiIa9u
w7rkLz6MUtdKQ8ufGkg9rKqgvlXlfYzHkUFhOMHjZO+Jg+cBiGg9YFxQJMPwJzjA
/ootw0PQ95Udn+rHzGlLV6dUOZ4abXHBARe0CATNQu4MIO4orMnsUAezpQcz6e+s
C9apZvdkRmYfR41lERdkBG4Slt6Y7/OqJ76jdkTmW2zeV37JjQfexbZR2UqWaFQe
DszVsDnXqARdfIEiJjJpE/qlvH6jJvhgtNiyw4ynXxfUza8fNNJE7z3joth72/1z
vn0Bvt5HHYk69Mh/Iy5lQ72wWgXjxg5RSeDCrVKvJJrEHQRb6C+Iza1qoykaU6ot
VvPoFLAkXsX5hugImpRLYqpJ+iCX5mKpmDlEqxfH+raimW3mieCwsXTYsyAU8EkZ
B1CDHPKjpbm69bs7UtsF4pR8ZPKy9dhz1X4eYLQ6qu2/QCm3ZcujTtJUmdHhcFvJ
d5U+Z0fQgp3RbzF2m4EkWnPaikCnzrTBxIK1cXoi1h+mdKmBvg2BayxN/L0/tSbh
MWX08PcUzwxZCSFG0KkFASiyegegXO8hxppuQzSsE02EAg969DqMTKOCYf6NpqYd
3J+jB9irCyUgQQa6q+3SlWA6rZAiRWrByPxY4HA0wV+Rl17URQ0ycIA7hy2O/sKs
tLQOhUjOLDOxMbkWNaP241nGy9PqiWH9xiNvXG3IiK/wD8M0/8UJT8qPjrZRDP0G
CbI4ddxHaB803l8/gtSptT770ThSjkVLqRKnnLAkePbHRsvzpxbsDCEx6uyzh2ez
tY0oWdobpGDBk5AQBUR26cfMber3OSVy2bsixVi2Vwm/49DAbR9eXXG8PpX+nndQ
gUzl1Ie1SHYCkElRdjo5KCyhiGZQ9ztQTR+DyHJE4IZ+xE356rwB+2OhaX9mN8vY
7+tJSjyTxhjU6391Ll/+m1lAfuF1Z4EQWM6gaszw/G5cn0sq1jVXv+L8zdWV1iiM
X74xT8qoxeIB9AZgMG4ItVRoedGraybe/oZJvHUpBRZgTbaUwxnGpS0+e+23DDCK
peByJG4KtOL2mxxseujhqMdzPMGuGV8RjvaiE360u2rpex7utatv5Uw1MNzaqCBv
XgFAYvl8dYLxh9lQdocGpPZCRYJ+YjAhcarHmGDCLNIQLo5u+FT/nTmeYnW+ggRL
J1Vcp2t2z8mCBFfzXfVwQ9ewpjQ1vl99j0S9kwGrY6YEnpWHDVb6NkYzuy7HYzlW
UK7S6gkJIVK34UezBdoM8pwi284CGzaCn4oUjixsXpoVBuSWxlF1cgejakRpZhPx
UjzfSp0UNSY4RycvVqcdD8VV918/1HYLRuC5ALfgnVyS+UKPEOf8Pthb+SHCuDlU
S5Xuzo4jNMssFgZmncFv+xral8gfPao5w5vg75cLcaUPXKI/WncziDZZQJwlkg1I
lVkmgNhxa59LZs77TQBZL6TWNmtXbCxhiH8yGSeiFaZR2ZMkjIbLdSoPgnLspOUH
BM4mK9WdL4QSnvwlUbLdVae4j4DX4lKSMDuiUtqdltHH70DXIMDN5vGYHN+t5luD
Hyc3RWw+gIYpGE81/aN64ywEinIcq+PjwNcab0D4fKalFeq7cbxhIlJAsFl7VIaR
kXbWg/SUwO8UpOMFNUU6HAJXgtEpwJgrrrMhvVr26hZy3Vj/D67o+44W1LrjH+yG
rxo1gcR0qb7SYAY4gZc8ERaLUAU/SR3Bk+2H10Pl6ogVDQA2VBEmW7RBBIkCHwQY
AQIACQUCWhNqdgIbDAAKCRACaXbfgK4z1/n+EAChSMZoC98QfvhQvKWvreBCB1Vl

```
UxM8gwO0z6YdtTUNQ4MhohpgxXNwoip49kx2CFbXDkxfYOpDyGxrKstDYMtVtQEW
wYX8kv8N5w7pvmx5C+WIyiqg6qmWbEyxb2Jly/YE9wl1V/5kaTCO6bOqe+efqUfR
ADod4tr8qK594A5fDqLhYHzFjBarAe8HdJL+4AO9sIqyEKtY0d1J+mJQgo9aWV/P
kFEKOb2QPv8auxN4Po5j9AYo3o/RfgsUryzDsSeXECG8rqQk0HcA93Y+6aaG0zRh
lKrxOxuUKYwE++DOU0bps60ygz4P0gABa3hxOgGs4E5QzuwsmXzPNEi14renHurW
A/dUF/TySqrHUEAmB+TZtWpUdsem4pi27j17lKEcUPTVqINN8Dg3LAVtmSyTWUm1
7KnjGl/mSNaUjWY+ZcCpd0SMmw+F3vO3N0ZrkbVESF/PVmlG/NgoTZ0cHYJEoaHQ
MpdK1rn8PzBCbMZxvYCmp5ID2serz8hSrX93Uq8fKDxArF5qhHp7IR7WgTwfZYca
52qPLeE/OL6EYmOYn+SIAxZJglArv8ZrpiaqBth6rM2J+fryH7czX5Kk5Zjl32qF
7QpZqFR9ADRr0wIYGLBLxz/fzQS6e4KHtUR6itsmskN5C8mVEoQwKTaGccrA68UX
4KUT0854gomWaLGCeg==
=LLnb
-----END PGP PRIVATE KEY BLOCK-----
```

Както стана дума, трябва да бъдете особено внимателни – от момента на извличането на вашия 'частен' ключ, до момента, в който го въведете в новата ви система и унищожите носителя, на който е бил временно записан. Препоръчваме при планирането на подобна миграция, по възможност да обмислите предварително процеса, да подготвите всичко необходимо и да се оттеглите на безопасно място, където да извършите съответните операции – без излишно бавене и без отвличане на вниманието с други задачи.

Следва да имате предвид и това – символите в един 'частен' ключ са взаимосвързани и наличието на една достатъчно голяма част от символите на ключа дава възможност да се направят определени изводи и за липсващите символи. Следователно – не е добра идея да променяте или заличавате определени символи, за да направите ключа неизползваем за този, който не знае какво е променено / заличено: за специалист, който разполага с променения 'частен' ключ не би представлявало голяма трудност да локализира неправилните / липсващите символи и да ги възстанови в техния първоначален вид.

8. „Асиметрично“ шифроване и разшифроване на съдържание

„Асиметричното“ шифроване позволява да изпратите шифровано съдържание до кореспондент, с когото преди това никога не сте се срещали и не сте имали възможност да обмените надеждно каквито и да било общи шифровъчни ключове или пароли. Благодарение на вашия 'публичен' ключ съдържанието може да бъде шифровано по начин, който не позволява обрано разшифроване без прилагането на 'частния' ключ, асоцииран еднозначно с приложения 'публичен' ключ. И тъй като 'частният' ключ е известен само на вас, а прилагането му е възможно само с парола, „асиметричното“ шифроване се превръща в предпочитан метод за поверителна комуникация (поне докато не бъде обменен по сигурен начин 'симетричен' ключ (парола) за още по-надеждно шифроване).

„Асиметричното“ шифроване на съдържание (аналогично на разгледаното по-горе „симетрично“ шифроване) може да стане по два начина – като прост текст направо в терминала или като обособени файлове; които файлове на свой ред могат да бъдат текстови или бинарни. Шифроването като прост текст се предпочита при изпращане на шифрованото съдържание чрез текстова комуникация, а шифроването в бинарен вид – при изображения, форматирани документи, аудио, видео, софтуер, архиви. При „асиметричното“ шифроване остава валидна препоръката да шифровате особено поверително съдържание само като прост текст направо в терминала (без да копирате от предварително подготвени текстови файлове и без изобщо да записвате съдържанието в нешифрован вид).

Можете да шифровате „асиметрично“ като прост текст чрез тази команда:

```
$ gpg -a -r Ключ -e
```

В случай, че желаете да шифровате като прост текст по начин, позволяващ разшифроване от няколко различни притежатели на 'частни' ключове, командният ред позволява да добавите и техните 'публични' ключове в едно общо шифроване чрез тази команда:

$ gpg -a -r Ключ_1 -r Ключ_2 -r Ключ_3 -e

(където под '**Ключ_1**', '**Ключ_2**', '**Ключ_3**' и т.н. разбираме какви да е достатъчно разпознаваеми части от информацията, с която съответните ключове се асоциират). Полученото шифровано съдържание ще може да се разшифрова с всеки един от 'частните' ключове, чиито 'публични' ключове са участвали в шифроването.

При въвеждане на горните команди по всяка вероятност системата ще попадне на 'публичен' ключ, който не е „подписан“ от някого, на когото вярвате, в потвърждение на това, че наистина принадлежи на този, с чиято информация е асоцииран. (Както проследихме по-горе, създаването на нова двойка „асиметрични“ шифровъчни ключове включва въвеждането на информация (наименование, адрес на електронна поща, пояснителна бележка), с която ключовете се асоциират; но не включва никакъв контрол за достоверността на тази информация. Всеки би могъл да създаде „асиметрични“ ключове с каквато си пожелае информация за тях.) Ако пристъпите към шифроване с 'публичен' ключ, за който липсва такова потвърждение, системата ще изведе указание, че няма потвърждение този ключ да принадлежи на цитирания потребител (**There is no assurance this key belongs to the named user**) и ще посочи допълнителна информация за този 'публичен' ключ, поставяйки пред вас въпроса дали все пак този ключ да бъде приложен (**Use this key anyway?**), като ви дава възможност да стартирате шифровъчния процес с въвеждането на '**y**' (от „**Y**[**es**]“), или да се откажете от него с въвеждането на '**n**' (от „**N**[**o**]“).

Вашето поведение в този момент зависи преди всичко от това по какъв начин сте придобили съответния 'публичен' ключ и доколко можете да бъдете сигурни, че той наистина принадлежи на този, на когото се доверявате. По този въпрос ще вземем отношение по-долу, където засягаме понятието 'отпечатък' (**Fingerprint**) или 'идентификатор' (както намираме за по-правилно да се нарича). Препоръчваме да не въвеждате '**y**' преди да сте се запознали с това понятие.

След като въведете командата '**gpg -a -r Ключ -e**' (или '**gpg -a -r Ключ_1 -r Ключ_2 -r Ключ_3 -e**' за няколко получатели едновременно) и след като потвърдите желанието си да бъдат приложени непотвърдените ключове, системата ще влезе в режим на очакване да въведете и самото съдържание, което желаете да бъде шифровано като прост текст. Както отбелязахме по-горе, можете да наберете текста направо в терминала или да копирате от предварително подготвен текстови файл. Когато сте готови с въвеждането (или копирането), натиснете **[Ctrl]+[d]** и системата ще изведе подобно „асиметрично“ шифрованото съдържание:

```
-----BEGIN PGP MESSAGE-----

hQIMA2GkANRDLiljAQ/9E+Och9cGxcNOT3fHTkYw+KbboykyDLtxtQUpumIuZopc
LIei0Hjw57SQ5Pb+Ar4Gy5VBaLoTcXjY8Blbfxp4G0Tk+mrcY/f+n2jY6FhzlJ1d
orCoK1SMXgaqX3wYzw6+fkHmHUrf+ytV2D1EGY3Fo4jHRdubhjtAspgd147vFqCV
vzt5eW+svgilB8R6ynDTvgrAOEepuGcSgZBMWmdt8rDhwYYYZWzBH46/emVe5Ehn
W/CyJe6pmGzM9sXcQS+FQpHxYAUqjRQe1fRC5gvYKItdG2THemxQtUxZrg5KFGxe
Ecu0JUDdmgBXe9NHYHMiZ7vncxK/wZmU2rkQAnQ1abs/e3uG64FTpgvL81jvt5jS
8j5AKIJEnv93PcXACKNxP3RGMMTXXcq7AguPhJFPeUqb1ZHewrPsMXPf/eR/HMmf
qjE6wHuBU0EWq4d3hUhhTjnfFzdM6J4eTEtTgBn11dOtVJ6f7pyDc9Y1krIE6E1b
Vf3PXoQvLPgNxXenk7dZt7MhQtFXoqeTMLxJwVO+FvDZ1HdRNxcxlk/ToALqrDOI
cuI9sHe7gHmLIfx+S5VVHlx5cTtx6Q1e/3Wf/lVnjKNF4+S2JMfGv6SrHbktFDBi
vlMCb8KmPhXs+cP06FZa0kdc8FLXUwak1AFaKJXlSvVfk4AMAaN11r9AW+SSecjS
fQGLEKrY5KwoaG2sxNBA42EellWJtf577J7igaGE8iML2V4f9nxXwc7LmcieZcIj
OicIm5B4DgYDTO7lmLYtgn2/CxsWBe0elnCIIfRYMUwWu9U5lG23xEDME+3ygPAW
UCww5ORGHDKK62hzi2V9PKQDfZAnqbx90+E8NszS
=1YnG
-----END PGP MESSAGE-----
```

Можете да шифровате „асиметрично“ и обособен текстови файл, без да въвеждате (копирате) текста в терминала:

```
$ gpg -a -r Ключ -e Съществуващ_текстови_файл
```

На същото място в системата ще бъде създаден '**Текстови_файл.asc**' с шифровано съдържание във вид на прост текст. Можете да отворите този файл в текстови редактор или в терминала и да копирате шифрованото като прост текст съдържание, за да го изпратите чрез текстова комуникация или да изпратите самия текстови файл чрез прикачане.

Можете да шифровате „асиметрично“ и бинарен файл:

```
$ gpg -r -e Ключ Съществуващ_бинарен_файл
```

На съответното място в системата ще се генерира **Шифрован_бинарен_файл.gpg** с бинарно съдържание. Това съдържание не може да бъде копирано и изпращано във вид на прост текст, но е значително по-компактно и подходящо при шифроването на изображения, форматирани документи, аудио- и видео-файлове, софтуер, архиви.

За да разшифровате съдържание във вид на прост текст направо в терминала, е достатъчно да въведете тази команда:

```
$ gpg
```

(при което системата ще влезе в решим на очакване да въведете (копирате) шифрованото съдържание като прост текст). След като сторите това, ще бъде потърсен 'частният' ключ, асоцииран с приложения при шифроването 'публичен' ключ. Ако необходимият 'частен' ключ бъде намерен в системата, ще бъде изведено указание да въведете неговата парола. Когато сторите това и натиснете **[Enter]**, шифрованото съдържание ще бъде преобразувано и изведено в терминала нешифровано, във вид на прост текст (след което може да прекъснете **GnuPG** с **[Ctrl]+[d]**). Както вече стана дума, най-поверителните съобщения е за предпочитане да бъдат прочетени само в този вид в терминала и изобщо да не бъдат записвани в нешифрован вид.

Можете също да разшифровате „асиметрично“ шифрован файл и да го изведете като самостоятелно обособен файл (без значение дали файлът е текстови или бинарен):

```
$ gpg Шифрован_файл
```

(не забравяйте да въведете наименованието на файла коректно, с неговото разширение – което в нашия случай най-вероятно ще бъде или '**.asc**' (при текстови файлове), или '**.gpg**' (при бинарни файлове)). Ако необходимият 'частен' ключ бъде намерен в системата, ще бъде изведено указание да въведете неговата парола. Когато сторите това, шифрованият файл ще бъде преобразуван и изведен на същото място в системата, но с нешифровано съдържание и с наименование без допълнителното разширение '**.asc**' или '**.gpg**'.

9. Цифрово „подписване“ чрез „асиметрични“ шифровъчни алгоритми

Цифровото „подписване“ позволява удостоверяване интегритета и автентичността на интересуващото ни съдържание – чрез криптографска проверка за това дали съдържанието не е променяно от първоначалния вид (към момента на неговото „подписване“) и за това дали то изхожда наистина от този, който се сочи за негов автор. Още в Част **II** проследихме такава проверка относно инсталационните файлове, обезпечаваща интегритета и автентичността на Свободния софтуер, който се готвите да инсталирате на вашето устройство.

Както вече стана дума, цифровото „подписване“ действа на базата на двойката от 'публичен' и 'частен' ключ, еднозначно асоциирани един с друг. Цифровите подписи се „полагат“ посредством 'частния' ключ, известен само на своя притежател (следователно няма как трета страна да „положи“ цифров подпис от името на притежателя). Така „положеният“ цифров подпис отразява точното състояние на „подписания“ файл (следователно дори само един символ

да бъде променен, това при проверка ще доведе до отрицателен криптографски отговор). Да провери цифровия подпис може всеки посредством 'публичния' ключ, съответстващ еднозначно на 'частния' ключ, с който е бил „положен“ цифровият подпис.

Можете да „положите“ цифров подпис върху определено съдържание поне по три начина – в зависимост от това как желаете да обособите цифровия подпис и от това желаете ли цифровото „подписване“ да бъде съчетано с последващо шифроване на съдържанието.

Можете например да „подпишете“ обикновен текстови файл, като вградите „положения“ цифров подпис в самия текст на файла:

```
$ gpg -u Подписващ_частен_ключ --clearsign Текстови_файл
```

(където под '**Подписващ_частен_ключ**' избираме ключ, с който да подпишем (ако имате въведен в системата само един 'частен' ключ, можете да пропуснете параметъра '**-u**' и неговата стойност '**Подписващ_частен_ключ**'); а под '**Текстови_файл**' разбираме наименованието на файла, който подписвате). При изпълнение на командата на съответното място в системата ще бъде изведен подобен текстови **.asc** файл:

```
-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA256

тект текст текст тект текст текст тект текст текст тект текст те
кст тект текст текст тект текст текст тект текст текст тект текс
т ект текст текст тект текст текст тект текст текст тект текс те
кст тект текст текст тект текст текст тект текст текст тект ...

-----BEGIN PGP SIGNATURE-----

iMkEAQEIADMWIQT3H/B+m0Gde+jZC+IrHvMPwEqAQQUCW1e6bRUceW91cm5hbWVA
ZXhhbXBsZS5jb20ACgkQKx7zD8BKgEFeZwQAmCx6yxmBsG5Ns3MzAQTqprkpIMpz
P3+wUT5E0qvpbQc1YSNyh8vq0RDCufQ6360Wx/TpNIW651DDCOEU/5GyPR3lD16T
1HFOA0FwZs1udb0Z0YpsIrmxNlL/WowrMRFTqoYJr3c14ss+Dpoknom1GKHcz+zW
i0sc7fbMVqcywWg=
=1HPK
-----END PGP SIGNATURE-----
```

(където под '**текст текст тект ...**' разбираме „подписаното“ текстово съдържание на файла, над което е въведен индикатор за „положения“ цифров подпис '**-----BEGIN PGP SIGNED MESSAGE-----**' с указание за приложения хеширащ алгоритъм (в случая – '**SHA256**'); а най-отдолу е въведен и самият цифров подпис, отново с индикатори за неговото начало и край). Именно по този начин е „подписана“ официалната версия на настоящото ръководство във файла **in-need-encrypt-bg.asc** с обикновен текстови формат.

Можете да „подпишете“ и бинарен файл, като изведете „положения“ цифров подпис в отделен бинарен **.sig** файл:

```
$ gpg -u Подписващ_частен_ключ --detach-sign Бинарен_файл
```

... или съкратено:

```
$ gpg -u Подписващ_частен_ключ -b Бинарен_файл
```

(като можете да спестите параметъра '**-u**' и идентификатора, ако имате въведен в системата само един 'частен' ключ). При изпълнение на командата ще бъде изведен бинарен **.sig** файл с „положения“ цифров подпис. При проверка на „положения“ подпис системата ще сравни символите в този бинарен файл и символите в „подписания“ бинарен файл

със символите в посочения 'публичен' ключ, за да изведе съответно положителен или отрицателен криптографски отговор.

Можете накрая да „подпишете“ определено съдържание и като допълнение към последващото му шифроване. Тази функция е полезна например в случай, че 'частният' ключ на вашия довереник е бил компрометиран, но това е останало неизвестно за него. Така трета страна би могла да прихване вашите съобщения, шифровани с 'публичния' ключ на вашия довереник, да ги разшифрова с неговия компрометиран 'частен' ключ, да ги прочете, да нанесе промени в тях и след това отново да ги шифрова с неговия 'публичен' ключ, за да му ги изпрати „от ваше име“. Ако обаче съобщенията са „подписани“ с вашия 'частен' ключ преди да бъдат шифровани с 'публичния' ключ на вашия довереник, за довършване на описаната измама ще бъде необходимо да бъде компрометиран и вашият 'частен' ключ (в противен случай няма как да бъде „положен“ цифров подпис от ваше име).

Можете да „подпишете“ определено съдържание и едновременно с това да го шифровате – като вградите параметъра '**`--sign`**' (или съкратено '**`-s`**') в процеса на шифроване.

Можете да сторите това с прост текст направо в терминала:

```
$ gpg -a -u Подписващ_частен_ключ -r Шифроващ_ключ -s -e
```

(при което системата ще влезе в режим на очакване да въведете направо в терминала текстовото съдържание, което желаете да подпишете и шифровате). Когато приключите с въвеждането и натиснете **`[Ctrl]`**+**`[D]`**, съдържанието ще бъде „подписано“ и шифровано, и резултатът ще бъде изведен като прост текст направо в терминала.

За да бъде разшифровано съдържанието във вид на прост текст и паралелно с това да бъде направена криптографска проверка на „положения“ цифров подпис, е необходимо в системата да имате въведен 'публичния' ключ, съответстващ на 'частния', с който е „положен“ подписът, а така също – 'частния' ключ, съответстващ на 'публичния', с който е осъществено шифроването. Достатъчно е да въведете команда '**`gpg`**' и след като системата влезе в режим на очакване, да въведете шифрованото съдържание. След като натиснете **`[Ctrl]`**+**`[D]`**, съдържанието ще бъде разшифровано и ще бъде осъществена криптографска проверка за интегритет и автентичност на „положения“ цифров подпис.

Можете също да „подпишете“ и шифровате обособен текстови файл:

```
$ gpg -a -u Подписващ_частен_ключ -r Шифроващ_ключ -s -e Текстови_файл
```

Системата ще изведе цифрово подписаното и шифровано съдържание в обособен текстови **`.asc`** файл.

Можете накрая да „подпишете“ и шифровате обособен бинарен файл:

```
$ gpg -u Подписващ_частен_ключ -r Шифроващ_ключ -s -e Бинарен_файл
```

Системата отново ще изведе цифрово подписаното и шифровано съдържание в обособен бинарен **`.gpg`** файл.

Независимо от това кой подход за цифрово „подписване“ е бил предпочетен, криптографска проверка на „положения“ цифров подпис можете да направите чрез параметъра '**`--verify`**', като преди това сте въвели във вашата система проверявания файл, „положения“ цифров подпис (ако е обособен в отделен файл) и 'публичния' ключ, за който се твърди да е еднозначно асоцииран с използвания при цифровото „подписване“ 'частен' ключ:

```
$ gpg --verify Проверяван_файл
```

... или респективно:

```
$ gpg --verify Подпис.sig
```

И в двата случая системата ще направи криптографска съпоставка между „подписания“ файл, 'публичния' ключ и цифровия подпис (независимо дали е вграден в самия „подписан“ файл или е обособен в отделен файл) и при наличие на пълно съответствие ще изведе положителен криптографски отговор (**Good signature**), а при липса на пълно съответствие – отрицателен криптографски отговор (**BAD signature**).

10. Установяване притежателя и актуалността на 'публичния' ключ

Съществен въпрос при ползването на 'публични' ключове за шифроване на съдържание (специално предназначено за конкретен получател) и при проверяване на цифрови подписи (за които се сочи да са „положени“ от конкретен автор) е установяването на това дали 'частният' ключ, еднозначно асоцииран с този 'публичен' ключ, действително принадлежи на този, на този, на когото мислите, че принадлежи. Ако например оригиналният 'публичен' ключ бъде подменен по някакъв начин от трета страна, тя (вместо този, на когото се доверявате) ще може да разшифрова съдържанието, предназначено само за вашия довереник; и тя (вместо този, на когото се доверявате) ще може да ви „пробута“ файлове, „подписани“ уж от вашия довереник.

Практиката познава атаки, при които на доверителя се предоставя подменен 'публичен' ключ; шифрованото с този ключ съдържание се разшифрова от атакуващата страна и веднага след прочитането му се изпраща на действителния получател, но вече шифровано с истинския 'публичен' ключ. В такава ситуация е възможно дори да не разберете, че разговорът се проследява.

За да се ограничат такива рискове, се ползват 'отпечатъци' (**Fingerprints**), които представляват **40** шестнадесетични цифри (най-често представени за прегледност в **10** групи от по **4** символа). 'Отпечатъкът' (или 'идентификаторът', както намираме за оправдано да го наричаме) представлява сложна криптографска производна от символите в съответния ключ и теоретично не може да бъде дублиран с 'отпечатъците' на други ключове. Ако дори само един символ в ключа е различен, ще се получи напълно различен 'отпечатък' и незабележимото несъответствие в ключа ще стане очевидно. Относително малкият размер на 'отпечатъците' (общо **40** символа) и тяхната уникалност за всеки ключ позволява ключовете да бъдат сравнявани лесно – както относно коректността на проверявания ключ, така и относно неговия действителен притежател. Можете да видите 'отпечатъците' на въведените във вашата система ключове така:

```
$ gpg --fingerprint
```

Ако сте въвели 'публичния' ключ на **www.Advocati.org**, ще бъде изведен този резултат:

```
pub   rsa4096 2023-08-05 [SC]
      9800 FDC3 3F04 EC07 1AD1  42CA 8F17 EED1 7B9E 53A3
uid           [ unknown] www.Advocati.org (адвокатска тайна)
<advocati@gmx.com>
sub   rsa4096 2023-08-05 [E]
```

'Отпечатъкът' на нашия ключ е '**9800 FDC3 3F04 EC07 1AD1 42CA 8F17 EED1 7B9E 53A3**', което може да бъде представено и слято: '**9800FDC33F04EC071AD142CA8F17EED17B9E53A3**'). От дадения пример е видно, че цитираният 'публичен' ключ е създаден на **05.08.2023** г. с общоизвестния шифровъчен алгоритъм **RSA** и е с дължина **4'096** бита (максималната възможна при този алгоритъм). Индикаторът '**[unknown]**' идва да покаже, че във вашата система няма потвърждения за това дали този ключ наистина принадлежи на този, който се сочи в записаната за ключа информация като негов притежател.

След като вече знаете 'отпечатъка' (идентификатора) на определен 'публичен' ключ, трябва да потърсите начин този 'отпечатък' (идентификатор) да бъде потвърден от вашия довереник (за когото се сочи да е притежател на 'частния' ключ, еднозначно асоциирания с интересуващия ви 'публичен' ключ).

Сравняването на 'отпечатъците' трябва да се извършва прецизно, тъй като понякога (макар и трудно) могат да се създадат ключове с „много подобни“ 'отпечатъци' на тези на вашите довереници. Също така, от критична важност е да разполагате с достоверен 'отпечатък' от интересуващия ви ключ. Това не може да стане през същия канал, от който

придобивате 'публичния' ключ – тъй като в случай на компрометиране на канала едновременно ще бъдат компрометирани и 'публичният' ключ, и неговият 'отпечатък'; от „външна страна“ 'публичният' ключ ще съответства на 'отпечатъка', но в действителност нито 'публичният' ключ, нито 'отпечатъкът' ще принадлежат на вашия довереник.

Поради горното е необходимо да потвърдите 'отпечатъка' през независим алтернативен канал, който да е трудно да се компрометира паралелно с канала, от който придобивате 'публичния' ключ. За да потвърдите нашия 'отпечатък' например, можете да ни потърсите по телефона (за да ви продиктуваме 'отпечатъка' на глас); да попитате друго лице, което ни се доверява и на което вие също имате доверие; или да се снабдите с наша визитна картичка (на нея имаме записан 'отпечатъка' на нашия 'публичен' ключ). След това остава да извлечете 'отпечатъка' на 'публичния' ключ, който сте въвели във вашата система чрез проследената по-горе команда '**gpg --fingerprint**' и да сравните (символ по символ) дали едното съвпада с другото.

Сигурен начин да свалите коректно копие от определен 'публичен' ключ, е да го изискате от сървър с 'публични' ключове в internet. Тази мрежа включва голям брой свободни и независими един от друг сървъри, предлагащи списъци с 'публични' ключове. Обичайна практика е притежателите на „асиметрични“ ключове да качват своите 'публични' ключове на такива свободни сървъри – което позволява всеки желаещ да придобива коректни копия от тях.

hkps://hkps.pool.sks-keyservers.net
keys.gnupg.net

Можете да придобиете коректно копие от нашия 'публичен' ключ чрез тази команда:

```
$ gpg --recv-keys 9800FDC33F04EC071AD142CA8F17EED17B9E53A3
```

(където '**9800FDC33F04EC071AD142CA8F17EED17B9E53A3**' е слято изписване на нашия 'отпечатък'). Вашата система ще изпрати запитване до достъпните сървъри с 'публични' ключове и ако се намери ключ с указания 'отпечатък', ще бъде изведен подобен резултат:

```
gpg: requesting key 8F17EED17B9E53A3
gpg: key 8F17EED17B9E53A3: public key "www.Advocati.org
(адвокатска тайна) <advocati@gmx.com>" imported
gpg: Total number processed: 1
gpg:               imported: 1
```

В цитирания случай '**8F17EED17B9E53A3**' е идентификатор на съответния 'публичен' ключ, който възпроизвежда в съкратен вид неговия 'отпечатък'. Можете да потърсите 'публичен' ключ и без да въвеждате целия му 'отпечатък' (което обаче има риск да ви подведе, тъй като е възможно да има ключ с „почти същия“ 'отпечатък'). Можете да търсите 'публични' ключове и по други части от информацията, с която са асоциирани – например наименование, адрес на електронна поща, допълнителни бележки. Така например нашия 'публичен' ключ можете да потърсите с коя да е от тези команди:

```
$ gpg --search-keys '8F17EED17B9E53A3'

$ gpg --search-keys 'www.Advocati.org'

$ gpg --search-keys 'адвокатска тайна'

$ gpg --search-keys 'advocati@gmx.com'
```

(или с достатъчно разпознаваема част от тази информация – например част от адреса на електронната поща).

Ако при търсене в internet се открият 'публични' ключове, които съответстват на търсения от вас критерий (например по критерия '**advocati.org**' ще откриете няколко 'публични' ключа), същите ще бъдат изведени в терминала като списък с 'отпечатъци' и с асоциирана информация за всеки от тях. Вие можете да въведете във вашата система

интересуващия ви 'публичен' ключ, като въведете поредния му номер от списъка (примерно '**1**') и натиснете **[Enter]**. Ако съответстващите на търсения от вас критерий ключове са по-голям брой и на екрана няма място да бъдат показани всичките, списъкът ще се раздели на страници, които можете да „прелиствате“ с '**N**' (от англоезичното '**N**[**ext**]'). Независимо от тези възможности обаче, не се препоръчва да се доверявате на асоциираната с конкретен 'публичен' ключ информация, нито на неговия съкратен идентификатор. Единствено пълните 'отпечатъци' могат да се приемат като надеждно средство за установяване цялостния интегритет (тъждество) на конкретно копие от един 'публичен' ключ с неговия първообраз.

Можете на свой ред да направите вашия 'публичен' ключ публично достояние, като го качите в internet:

```
$ gpg --send-keys Отпечатък
```

(където под '**Отпечатък**' разбираме **40**-шестнадесетична цифрова комбинация на вашата двойка ключове, изписана слято). Системата ще направи връзка с доверените сървъри и след като изпрати вашия 'публичен' ключ, ще изведе подобен резултат:

```
gpg: sending key Ключ to hkps://hkps.pool.sks-keyservers.net
```

Така, ако някой пожелае да изпрати шифровано съобщение до вас или да направи криптографска проверка на „положен“ от вас цифров подпис, ще бъде достатъчно да удостовери надеждно вашия 'отпечатък' и ще може да изтегли коректно копие от 'публичния' ви ключ от internet – без да е необходимо да се свързва с вас и да иска да му го предоставите. Последното само по себе си може да бъде компрометирано по-лесно, отколкото ако се осъществява от internet – например защото е трудно да се продиктуват и запишат точно няколко хиляди символа. Можете да качвате своите 'публични' ключове и ръчно в избрани от вас сървъри (**Key Servers**) през предоставения от тях web-интерфейс, като за целта най-често ви се предлага възможност да копирате ключа във вид на прост текст или по-рядко да го качите като текстови **.pub** файл. В потвърждение на успешното качване най-често сървърите извеждат подобен резултат:

```
Key block added to key server database. New public keys added:
1 key(s) added successfully.
```

Доверените сървъри обикновено работят в система и обменят получените данни помежду си, така че е достатъчно вашият 'публичен' ключ да бъде качен на няколко от тях, за да бъде скоро на разположение във всички.

Препоръчва се да публикувате вашите 'публични' ключове в internet не само за да позволите на вашите довереници да изтеглят коректни копия от там. Съществуването на коректно копие от вашия 'публичен' ключ, което винаги можете да изтеглите и вие самите, е полезно и за вас, в поне два случая. Може например в някакъв момент да се намирате далеч от вашите устройства, но да се наложи да изпратите до себе си някакво поверително съдържание. В такава ситуация е най-лесно просто да го шифровате със собствения си 'публичен' ключ. Може също така внезапно да ви бъде отнето устройството, на което съхранявате вашата двойка „асиметрични“ ключове. В такъв случай (както ще проследим след малко) трябва незабавно да анулирате компрометираните ключове. Това обаче няма как да стане, ако не разполагате със специален 'сертификат за анулиране' (който по дефиниция съхранявате на отделно устройство, някъде далеч от подозренията на трети страни) и с копие от съответния 'публичен' ключ. И не на последно място – наличието на копие от вашия 'публичен' ключ в internet е един от способите, чрез които ще направите публично достояние факта на анулирането, ако това се наложи.

Във връзка с горното е необходимо да отбележим и възможността информацията за интересуващите ви шифровъчни ключове да бъде обновявана в internet:

```
gpg --refresh-keys
```

При изпълнение на горната команда системата изисква от доверените сървъри да укажат дали при тях няма постъпили нови съобщения относно ключовете, които са въведени във вашата система. Най-често такива съобщения могат да бъдат в смисъл, че няма изменения (**not changed**), че трети страни потвърждават със своите цифрови подписи

самоличността на притежателите на съответните 'публични' ключове (**2 new signatures**) или накрая – че някой от 'публичните' ключове е бил анулиран (**Revoked**) и вие не можете повече да се доверявате на този ключ. Излишно е да ви убеждаваме в полезността на това от време на време да обновявате информацията за интересуващите ви 'публични' ключове (особено когато се готвите да шифровате с тях наистина поверителна информация). Така бихте могли да избегнете пробив в сигурността, в случай, че междувременно шифровъчните ключове на някой от вашите довереници е бил компрометиран, поради което той е съумял да го анулира и да оповести това публично.

11. Анулиране на компрометирани „асиметрични“ ключове

Независимо от всичките положени усилия за обезпечаване на информационната сигурност, е възможно сигурността все пак да се провали поради някаква причина и вашият 'частен' ключ да бъде компрометиран. Това може да се дължи на узнаване от трета страна на паролата за неговото „симетрично“ разшифроване (което позволява при наличието на достъп до вашата система да бъде извлечено копие от 'частния' ви ключ или той да бъде изтрит). Възможно е също така да изгубите контрол над устройството, в което съхранявате ключа (фатално повреждане на твърдия диск, изгубване на устройството, кражба, грабеж, изземване от властите). В случай на някой от горните инциденти е желателно да „анулирате“ компрометираната двойка шифровъчни ключове и да направите обществено достояние факта, че съответният 'публичен' ключ вече не следва да бъде ползван от вашите довереници – нито за проверка на „положените“ от вас цифрови подписи, нито за шифроване на съдържание, предназначено само за вас.

За да можете да анулирате компрометираните шифровъчни ключове, трябва да притежавате предварително подготвен сертификат за анулиране (**Revocation Certificate**), който да задействате при необходимост. Такъв сертификат обаче не може да бъде създаден, ако вече сте изгубили контрол върху вашия 'частен' ключ. Ето защо трябва да създадете своя сертификат веднага, след като сте създали двойката шифровъчни ключове и да запазите 'сертификата' на отделен носител, който по възможност съхранявате на отдалечено тайно място. Така ще можете да задействате сертификата, дори и ако устройство, в което съхранявате 'частния' ключ, бъде внезапно компрометирано – защото е трудно паралелно с това да бъде компрометиран и носителят, който съхранявате другаде.

Можете да създадете сертификат за анулиране така:

```
$ gpg --gen-revoke Ключ
```

Системата ще влезе в диалогов режим и ще укаже да изберете някоя от възможните причини за създаване на сертификата, като ви даде възможност да въведете и пояснителна бележка. (Ние например сме въвели пояснителна бележка '**!!! KEY COMPROMISED !!!**' с цел да направим факта на компрометирането (ако това се случи) колкото се може по-разпознаваем.) След като въведете указаните данни и паролата на вашия 'частния' ключ, ще бъде създаден подобен сертификат за анулиране:

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
Comment: This is a revocation certificate

iQF5BCABCABjFiEEfoLj5LoFximEwMdiXGoi+EuH69QFAlno3QFFHQDQmtCb0K7Q
p9Cq0KIg0JrQntCc0J/QoNCe0JzQldCi0JjQoNCQ0J0gfCBLRVkgQ09NUFJPTUlT
RUQKXHgwNApceDA0AAoJEFxqIvhLh+vU7mQIAJh/L6Ynjx41cLfHhAXUlkw/47nP
IUS2MULuFI0Tra+BXl0hSgxWHlb/Od6Ze3DRpLswvfYY10a1jsNE1mSKk57vR1ul
utz/L8CZfgnA6OScTJDoAHWh2KLL/oiz+/TKhJXIoNFEv0K+RGlhBeBDOcUHz+ZW
C5JgJGzs3o6/Epw0bXCyoDMXcNh3AnV71c0vzlkUjHq+on08WLsBbykJuHOCkRM0
xOiAgRdTokEqP+ier2AoDGt0AfKDK1S2DI4WfDCF3l3wcM/pziIjZEP/isW/3WX4
guAVQYb3jOYNGnRpVCKZfiFbXVkDPs6MwXYQ93DYEixosEFW15t3dsA1Hyo=
=DhMs
-----END PGP PUBLIC KEY BLOCK-----
```

Можете да изведете сертификата за анулиране в прост текстови файл чрез описаната по-горе команда '**cat > Сертификат.asc**' или да го копирате ръчно в прост текстови файл, след чието наименование поставите

разширение **.asc**. Ако сте избрали да работите от графичния интерфейс (което не ви препоръчваме да правите), внимавайте да не стартирате файла със сертификата поради грешка (като например натиснете върху него).

В случай, че на един следващ етап вашият 'частен' ключ бъде компрометиран, трябва възможно най-бързо да се доберете до сертификата за анулиране и да стартирате процедура по анулиране на компрометираната двойка ключове. Тази процедура включва две части: локално анулиране на ключовете в тесния смисъл на думата (само във вашата система); и публично оповестяване на анулирането (за да бъдат вашите довереници предупредени повече да не шифроват поверително съдържание с компрометирания 'публичен' ключ и да не проверяват цифрови подписи, „положени“ с компрометирания 'частен' ключ след момента на неговото компрометиране.

За първата част от процедурата (локално анулиране на ключовете във вашата система) е необходимо да имате въведен в настоящата ви система поне 'публичния' ключ от двойката компрометирани ключове. Имайте предвид, че може внезапно да изгубите контрол над вашата система (например поради нейното завладяване от трети страни, които именно по този начин са компрометирали вашите ключове). При това положение – за да можете да осъществите анулирането, ще трябва да придобиете собствения си 'публичен' ключ от другаде. Единият вариант е да имате копие от 'публичния' ключ на отдалечения носител, където съхранявате и 'сертификата за анулиране'. Другият вариант е да свалите копие от вашия 'публичен' ключ от сървърите в internet (стига да сте го качили там преди това) чрез командата '**gpg --import**'. Когато вече имате въведен в настоящата ви система вашия 'публичен' ключ, можете да осъществите и самото анулиране чрез тази команда:

```
$ gpg --import Сертификат.asc
```

Съществува възможност да въведете сертификата за анулиране и във вид на прост текст направо в терминала, като за тази цел е достатъчно да въведете командата '**gpg --import**' и да натиснете **[Enter]**, след като сте копирали сертификата в терминала. Независимо кой от двата варианта ще изберете, шифровъчният ключ ще бъде анулиран локално (само във вашата система). Това обаче не важи за копия от шифровъчния ключ, които може да са били създадени преди анулирането и да се съхраняват някъде другаде. Ето защо втората част от процедурата е толкова важна – чрез нея трябва да оповестите публично (колкото се може по-публично) факта на анулирането. За тази цел трябва да извършите две неща: да качите (чрез описаната по-горе команда '**gpg --send-keys Отпечатък**') вече анулирания 'публичен' ключ в поне няколко сървъра с 'публични' ключове в internet; и по възможност да съобщите лично на вашите довереници за настъпилото компрометиране и анулиране.

След като анулираният 'публичен' ключ бъде качен в internet, всеки притежател на копие от този ключ ще може да получи съобщение за неговото анулиране при изпълнение на проследената по-горе команда '**gpg --refresh-keys**' за обновяване на наличните в неговата система „асиметрични“ ключове.

След като вашите довереници обновят интересуващите ги 'публични' ключове, при опит да шифроват съдържание с вашия анулиран 'публичен' ключ системата им ще изведе резултат, че ключът е АНУЛИРАН (**REVOKED**) (заедно с вашата поясняваща бележка, ако сте задали такава при създаването на сертификата). Същото ще се случи и при опит да извършат проверка на „положен“ цифров подпис с вашия 'частен' ключ.

Въпреки препоръката за често изпълняване на командата '**gpg --refresh-keys**' (и особено преди шифроване и изпращане на поверителна информация), все пак не трябва да се очаква, че вие или вашите довереници ще анулирате своевременно компрометираните 'частни' ключове (понякога това се оказва невъзможно – например поради липса на достъп до необходимите архиви, устройства, internet-свързаност при настаняване в болница, задържане и т.н.). Ето защо е много важно да изградите автономен механизъм, който сте в състояние да се задейства лесно и неговото изпълнение да доведе до информиране на довереницте ви за настъпилото компрометиране. Автономността на механизма е ключова за неговата надеждност. Би било полезно например трето доверено лице да е инструктирано да разпрати съобщение до вашите довереници, ако вие самите изпаднете в невъзможност да го направите.

Имайте предвид, че анулирането преграждa шифроването и „подписването“ само занапред (и то само частично). Съдържанието, което е шифровано с вашия 'публичен' ключ, продължава да може да се разшифрова чрез асоциирания с него 'частен' ключ, респективно „положените“ чрез 'частния' ключ цифрови подписи продължават да могат да се проверяват чрез 'публичния' ключ – дори и след като двойката шифровъчни ключове вече е анулирана.

Нещо повече – ако съществуват други копия от вашите компрометирани ключове, които не са анулирани локално и не са обновени с актуални съобщения от internet, те дори запазват своята пълна функционалност от преди анулирането. Ето защо при никакви обстоятелства не бива да поставяте сигурността на вашата поверителна информация в зависимост само от един 'частен' ключ и неговата парола. Препоръчваме да съхранявате своите архиви на отделни носители, които се шифроват с отделни средства (например със „симетрични“ ключове). Това би ограничило рисковете за вашата поверителна информация, които произтичат от компрометирането на един конкретен 'частен' ключ.

IX. ПРЕДСТАВЯМЕ ВИ НАШИЯ 'ПУБЛИЧЕН' КЛЮЧ

Преди да завършим, предлагаме на вашето внимание кратък указател с основните web-адреси и команди, на които се спряхме в изложението. Този път те са структурирани по темите, за които се отнасят – но вече в своята технологична последователност, без отношение към плавното въвеждане на потребителя в материята. Насърчаваме ви преди да пристъпите към този указател, да се запознаете с цялостното изложение – тъй като указателят сам по себе си не е подходящ за „прескачане“ на дадените по-горе въвеждащи бележки – особено ако не сте сигурни какво правите.

Отделно от всичко казано дотук – насърчаваме ви да продължите да изучавате командния ред и да ползвате терминала на вашата Свободна операционна система колкото е възможно повече. Това ще повиши значително информационната ви сигурност, но също така ще ви даде широта, която графичният интерфейс отказва на потребителите. Не се стеснявайте да се държите с вашия компютър като програмисти! В противен случай – отделно от компрометирането на сигурността – бихте се оставили да ви управлява някой друг, чрез потребителския дизайн и чрез ограничените функции, които той предлага. Бъдете свободни и управлявайте това, което по право и изначално ви принадлежи!

НАШИТЕ АВТОРИ

https://www.advocati.org/
https://www.libtec.org

ОФИЦИАЛНА ВЕРСИЯ НА НАСТОЯЩОТО РЪКОВОДСТВО

https://advocati.org/consultation/security/in-need-encrypt/
https://libtec.org/in_need_encrypt/

СПЕЦИАЛНА БЛАГОДАРНОСТ

Richard M. Stallman
Denis 'GNUtoo' Carikli

СВОБОДНИ ТЕХНОЛОГИИ И СИГУРНОСТ

https://en.wikipedia.org/wiki/Free_as_in_Freedom

https://en.wikipedia.org/wiki/Source_code

https://en.wikipedia.org/wiki/Backdoor_(computing)

https://bg.wikipedia.org/wiki/GNU_General_Public_License
https://en.wikipedia.org/wiki/Unix
https://www.gnu.org/gnu/thegnuproject.en.html
https://www.fsf.org/

https://en.wikipedia.org/wiki/Linux-libre
https://www.gnu.org/distros/free-distros.html

https://www.parabola.nu/
https://trisquel.info/

https://www.dragora.org/
https://pureos.net/
https://www.replicant.us/

https://heads.dyne.org/
https://tails.boum.org/

ИНСТАЛИРАНЕ НА СВОБОДНА ОПЕРАЦИОННА СИСТЕМА

https://en.wikipedia.org/wiki/Live_USB
https://en.wikipedia.org/wiki/Live_CD

https://wiki.parabola.nu/Get_Parabola
https://trisquel.info/en/download
https://guix.gnu.org/en/download/
https://pureos.net/download/
https://www.gnu.org/distros/free-distros.html

https://labs.parabola.nu/

УДОСТОВЕРЯВАНЕ НА АВТЕНТИЧНОСТТА И ИНТЕГРИТЕТА

https://en.wikipedia.org/wiki/Public-key_cryptography
https://en.wikipedia.org/wiki/Digital_signature
https://en.wikipedia.org/wiki/Checksum

hkps://hkps.pool.sks-keyservers.net
keys.gnupg.net

http://www.linux-usb.org/usb.ids

СВОБОДЕН ХАРДУЕР

https://h-node.org/wifi/catalogue/en
https://libreboot.at/docs/hardware/index.html#supported_list

https://technoethical.com/
https://libiquity.com/

http://www.linux-usb.org/usb.ids

СВОБОДЕН СОФТУЕР ОТ „НИСКО“ НИВО

https://en.wikipedia.org/wiki/Intel_Management_Engine

https://en.wikipedia.org/wiki/EEPROM
https://en.wikipedia.org/wiki/Flash_memory

https://en.wikipedia.org/wiki/BIOS
https://en.wikipedia.org/wiki/UEFI

https://www.coreboot.org/
https://libreboot.org/
https://libreboot.at/

https://libreboot.at/docs/hardware/
https://libreboot.at/download.html#https

https://store.arduino.cc/products/arduino-nano
https://www.pololu.com/product/2595
https://beagleboard.org/black
https://store.arduino.cc/products/arduino-uno-rev3

https://github.com/noblepepper/serprog-duino.git
https://beagleboard.org/distros
https://phreedom2600.net/

https://chromium.googlesource.com/chromiumos/third_party/flashrom

СИГУРНО ШИФРОВАНЕ

https://en.wikipedia.org/wiki/Enigma_machine
https://en.wikipedia.org/wiki/RSA_(cryptosystem)

https://gnupg.org/
https://www.openpgp.org/

https://en.wikipedia.org/wiki/TrueCrypt
https://www.veracrypt.fr/

https://en.wikipedia.org/wiki/ASCII
https://en.wikipedia.org/wiki/Extended_ASCII
https://en.wikipedia.org/wiki/Unicode

СИГУРНОСТ В INTERNET

https://en.wikipedia.org/wiki/List_of_TCP_and_UDP_port_numbers

https://addons.mozilla.org/firefox/addon/noscript
http://localhost:8080/

https://duckduckgo.com/
https://startpage.com/
https://www.ixquick.com/

https://www.autistici.org/
https://bitmessage.ch/
https://disroot.org/
https://protonmail.com/
https://riseup.net/
https://tutanota.com/

https://en.wikipedia.org/wiki/Peer-to-peer
https://xmpp.org/
https://list.jabber.at/
https://www.pidgin.im/
https://gajim.org/
https://otr.cypherpunks.ca/
https://conversations.im/omemo/
https://github.com/gkdr/lurch

https://www.torproject.org/
https://bg.wikipedia.org/wiki/ISO_3166-1
https://bridges.torproject.org/options
Gmail; Riseup!; Yahoo!
bridges@bridges.torproject.org
get bridges

3g2upl4pq6kufc4m.onion
xmh57jrzrnw6insl.onion
zqktlwi4fecvo6ri.onion/wiki/index.php/Main_Page
dirnxxdraygbifgc.onion

КОМАНДЕН РЕД

https://www.linuxcommand.org/lc3_learning_the_shell.php
https://www.tldp.org/LDP/GNU-Linux-Tools-Summary/html/GNU-Linux-Tools-Summary.html

ОЩЕ ЗА КОМАНДНИЯ РЕД

```
$ man Програма_или_команда

$ Програма_или_команда --help

$ Програма_или_команда -h

Програма_или_команда [Възможност1|Възможност2] Задължителен_параметър

# pacman -S bash-completion
# apt-get install bash-completion
```

ИЗПЪЛНЕНИЕ НА ОСНОВНИ КОМАНДИ

```
[Ctrl]+[Alt]+[F1]
[Ctrl]+[Alt]+[F2]
[Ctrl]+[Alt]+[F3]
[Ctrl]+[Alt]+[F4]
[Ctrl]+[Alt]+[F5]
[Ctrl]+[Alt]+[F6]

[Ctrl]+[Alt]+[F7]

$ su
# Команда
[Ctrl]+[d]

$ sudo Команда

[Потребител@Компютър ~]$
[root@Компютър ~]#

# exit
# systemctl suspend
# reboot
```

```
# poweroff

[Tab]

[↑]
```

ПРЕМЕСТВАНЕ В СИСТЕМАТА

```
/home/Текущ_потребител

$ cd
$ cd /Адрес
$ cd Директория
$ cd Директория/Следваща_директория
$ cd Desktop
$ cd /home/Потребител/Desktop
$ cd ..
$ cd -
$ cd ~
$ cd ~Потребител
# cd ~root

Наименование\ с\ интервали

Команда \
на няколко реда
```

ПРЕГЛЕЖДАНЕ НА СЪДЪРЖАНИЕ И ДАННИ

```
$ pwd

$ ls
$ ls -a
$ ls -l
$ ls -la
$ ls -la Директория Друга_директория
$ ls > Текстови_файл

$ ls /Адрес/Файл_или_директория > /Адрес/Текстови_файл
$ ls -la /Директория /run/media/Устройство > Текстови_файл

$ less Текстови_файл
[HOME]; [PGUP]; [PGDN]; [END]
[h]
/Търсене
[n]
[q]

$ less ~/.bash_history
# less ~/.bash_history

[Ctrl]+[r]
```

СТАРТИРАНЕ НА ПРОГРАМИ

```
$ Програма
$ Програма Файл
$ Програма Файл
$ Браузър Адрес_на_web_сайт
$ Релативен/адрес/Програма
$ /Абсолютен/адрес/Програма
$ ./Програма
$ Програма Релативен/адрес/Файл
$ Програма /Абсолютен/адрес/Файл
```

СЪЗДАВАНЕ И ПРЕМАХВАНЕ НА ФАЙЛОВЕ И ТЕКСТОВО СЪДЪРЖАНИЕ

```
$ touch Празен_файл

$ cat > Текстови_файл
[Enter]; [Ctrl]+[d]

$ cat >> Текстови_файл
[Enter]; [Ctrl]+[d]

$ rm Файл

$ shred -n Брой -u -z Файл
```

РЕДАКТИРАНЕ НА ТЕКСТОВО СЪДЪРЖАНИЕ В ПРОГРАМАТА NANO

```
$ nano Текстови_файл

[Ctrl]+[g]

[Ctrl]+[o]
[n]; [y]; [Enter]
[Ctrl]+[x]

[Ctrl]+[r]

[Alt]+[u]
[Alt]+[e]

[Ctrl]+[Shift]+[c]
[Ctrl]+[Shift]+[x]
[Ctrl]+[Shift]+[v]

[Ctrl]+[a]
[Ctrl]+[k]
[Ctrl]+[y]
```

УПРАВЛЕНИЕ НА ДИРЕКТОРИИТЕ

```
$ mkdir Директория

$ cp Директория_или_Файл Нова_директория_или_Файл
```

```
$ cp -r Директория Нова_директория

$ mv Заварен_адрес/Директория_или_Файл Нов_адрес/Директория_или_файл
$ mv Адрес/Директория_или_Файл /dev/Устройство/Директория_или_файл

$ rmdir Празна_директория
$ rm -r Директория_със_съдържание

$ rm -r *
$ rm -r .[^.]* *
$ echo rm -r *
$ echo rm -r .[^.]* *
```

(РАЗ)АРХИВИРАНЕ И (РАЗ)КОМПРЕСИРАНЕ

```
$ tar xf Архивен_файл
$ tar xf Архивен_файл -C Адрес
$ tar cf Нов_архивен_файл.tar Съществуваща_директория_или_файл
$ xz Файл_или_архивирана_директория
$ unxz Компресиран_файл_или_архивирана_директория
```

УПРАВЛЕНИЕ НА ПРАВАТА

u[**ser**]
g[**roup**]
o[**thers**]

```
# chmod ??? /Директория_или_Файл
```

4; **r**[**ead**]
2; **w**[**rite**]
1; [**e**]**x**[**ecute**]
0; -

```
# chmod +x Изпълним_файл

# chown Потребител:Група /Директория_или_Файл
```

УПРАВЛЕНИЕ НА УСТРОЙСТВАТА

```
$ lsblk

$ mount /dev/Устройство /Точка_на_закачане
$ umount /dev/Устройство
$ udisksctl power-off -b /dev/Устройство

$ mount -o ro /dev/Устройство /Точка_на_закачане

# mkfs.ext4 /dev/Физически_дял -n Наименование
# mkfs.fat /dev/Физически_дял -n Наименование

> linux /boot/vmlinuz-linux-libre-lts root=/dev/mapper/Група-Системен_дял rw
nousb

$ lsusb
```

```
# pacman -S usbguard
# usbguard generate-policy > /etc/usbguard/rules.conf
# systemctl enable --now usbguard

# usbguard list-devices -b
# usbguard allow-device Номер
# usbguard allow-device -p Номер
# usbguard list-rules
# usbguard remove-rule Номер
```

СВЪРЗВАНЕ КЪМ INTERNET И СВАЛЯНЕ НА СЪДЪРЖАНИЕ

```
# nmcli dev wifi connect Мрежа password Парола
# dhcpcd

# ping Сървър
[Ctrl]+[c]

$ wget Адрес_в_internet/Ресурс

# git clone Адрес_в_internet
```

СИГУРНОСТ В INTERNET

```
A;a;B;C;c;E;e;H;K;M;m;n;O;o;P;p;T;y;X;x
A;a;B;C;c;E;e;H;K;M;т;п;O;o;P;p;T;y;X;x

n;m
п;т

A;a;B;E;H;I;i;K;M;N;n;O;o;P;p;T;t;v;u;X;x;Y;y;Z
A;α;B;E;H;I;ι;K;M;N;η;O;o;P;ρ;T;τ;ν;υ;X;χ;Y;γ;Z

A;a;б;B;Г;E;K;к;Л;M;H;O;o;П;п;P;p;T;y;Ф;X;x
A;α;б;B;Г;E;K;к;Л;M;H;O;o;П;π;P;ρ;T;γ;Ф;X;χ

# pacman -S ufw
# systemctl enable --now ufw

# ufw default deny

# systemctl enable --now ufw

# ufw default deny

# ufw enable
# ufw disable

# ufw allow Заявки_изключение
# ufw limit Заявки_изключение

# pacman -S libreboot-utils
$ ich9gen --macaddress MAC-адрес
```

```
$ dd if=ich9fdgbe_4m.bin of=Libreboot_пакет bs=1 count=12k conv=notrunc
$ dd if=ich9fdgbe_8m.bin of=Libreboot_пакет bs=1 count=12k conv=notrunc
$ dd if=ich9fdgbe_16m.bin of=Libreboot_пакет bs=1 count=12k conv=notrunc

# nano /etc/NetworkManager/NetworkManager.conf
wifi.cloned-mac-address=MAC-адрес
[Ctrl]+[o],[Enter]
wifi.cloned-mac-address=random
[Ctrl]+[o],[Enter]

# systemctl reload NetworkManager

# nmcli con modify Мрежа cloned-mac MAC-адрес

# nmcli con add type wifi con-name Мрежа ssid Мрежа wifi-sec.key-mgmt \
wpa-psk wifi-sec.psk Парола cloned-mac MAC-адрес

# nmcli con delete Мрежа

# nmcli con modify Мрежа ipv4.dhcp-send-hostname FALSE

# nmcli con add type wifi con-name Мрежа ssid Мрежа wifi-sec.key-mgmt \
wpa-psk wifi-sec.psk Парола ipv4.dhcp-send-hostname FALSE

# nmcli con modify Мрежа ipv4.dhcp-hostname Наименование

# nmcli con add type wifi con-name Мрежа ssid Мрежа wifi-sec.key-mgmt \
wpa-psk wifi-sec.psk Парола ipv4.dhcp-hostname Наименование

# pacman -S netcat
# nc -lp 8080
http://localhost:8080/

nano tor-browser_en-US/Browser/TorBrowser/Data/Tor
ExitNodes {Предпочитана_държава}
StrictNodes 1
[Ctrl]+[o],[Enter]
ExitNodes {Предпочитана_държава_1},{Предпочитана_държава_2}
StrictNodes 1
[Ctrl]+[o],[Enter]
ExcludeExitNodes {Нежелана_държава_1},{Нежелана_държава_2}
StrictNodes 1
[Ctrl]+[o],[Enter]

StrictNodes 1
StrictNodes 0

Gmail; Riseup!; Yahoo!
bridges@bridges.torproject.org
get bridges
```

```
# pacman -S Програма
# pacman -R Програма

# apt-get install Програма
# apt-get remove Програма

$ gedit Несъществуващ_файл.txt

# pacman -Ss Ключова_дума
# pacman -Qs

# pacman -Syu
[Y]; [n]
[y]; [N]

# pacman -Syyu

# pacman -Sud

# pacman -Sy parabola-keyring archlinux-keyring
# pacman-key --refresh-keys

# pacman-key --refresh-keys Адрес_на_неактуализиран_ключ

# rm /var/cache/pacman/pkg/*

# rm /var/lib/pacman/db.lck

# pacman -Fy
# pacman -F Команда
# pacman -F Системен_файл
# pacman -S --overwrite '*' Проблемен_пакет

# paccheck --quiet --file-properties
```

ИНСТАЛАЦИОНЕН НОСИТЕЛ (КЛАСИЧЕСКИ МЕТОД)

```
# dd if=Файл.iso of=/dev/Устройство bs=64M
# sync

# fdisk /dev/Устройство
# o
# n
# p
# w
[Enter]

# mkfs.ext4 -O^metadata_csum_seed /dev/Физически_дял
# mount /dev/Физически_дял /mnt

# pacman -Sy arch-install-scripts
# pacstrap /mnt
```

```
# genfstab -U /mnt >> /mnt/etc/fstab
# nano /mnt/etc/fstab
UUID=Идентификатор none swap defaults 0 0
[Ctrl]+[a]; [Ctrl]+[k]
[Ctrl]+[o], [Enter]

# echo en_US.UTF-8 UTF-8 >> /mnt/etc/locale.gen
# echo LANG=en_US.UTF-8 > /mnt/etc/locale.conf
# echo LANG=en_US.UTF-8 > /etc/locale.conf
# nano /mnt/etc/pacman.conf
ParallelDownloads = 5
[Ctrl]+[o], [Enter]

# arch-chroot /mnt/

# locale-gen

# pacman -S linux-libre-lts linux-libre-firmware grub \
cryptsetup lvm2 networkmanager nano bash-completion \
arch-install-scripts

# grub-install --recheck /dev/Устройство
# mkdir /boot/grub
# grub-mkconfig -o /boot/grub/grub.cfg

# passwd
# exit
```

ИНСТАЛАЦИОНЕН НОСИТЕЛ (НАПРЕДНИЧАВ МЕТОД)

```
# fdisk /dev/Устройство
# o
# n
# p
# w
[Enter]

# mkfs.ext4 -O^metadata_csum_seed /dev/Физически_дял

# mount /dev/Физически_дял /mnt
# pacman -Sy arch-install-scripts
# pacstrap /mnt

# genfstab -U /mnt >> /mnt/etc/fstab

# nano /mnt/etc/fstab
UUID=Идентификатор none swap defaults 0 0
[Ctrl]+[a]; [Ctrl]+[k]
[Ctrl]+[o]; [Enter]

# nano /mnt/etc/locale.gen
en_US.UTF-8 UTF-8

# echo LANG=en_US.UTF-8 > /etc/locale.conf
```

```
# nano /mnt/etc/pacman.conf
ParallelDownloads = 5

# arch-chroot /mnt

# locale-gen

# passwd

# pacman -S linux-libre-lts linux-libre-firmware cryptsetup \
lvm2 networkmanager nano bash-completion arch-install-scripts

# exit
```

ИНСТАЛИРАНЕ НА НЕШИФРОВАНА СВОБОДНА ОПЕРАЦИОННА СИСТЕМА (КЛАСИЧЕСКИ МЕТОД)

```
[Esc], [Del]; [F1]; [F2]; [F3]; ...
[Ctrl]+[Alt]+[Del]
[↓]; [↑]
[Enter]; [Backspace]; [+]; [-]; [Space]

# nmcli dev wifi connect Мрежа password Парола
# ping Сървър
[Ctrl]+[c]

# fdisk /dev/Устройство
# o
# n
# p
# w
[Enter]

# mkswap /dev/Разменен_дял
# swapon /dev/Разменен_дял

# mkfs.ext4 -O^metadata_csum_seed /dev/Системен_дял
# mount /dev/Системен_дял /mnt

# mkfs.ext4 /dev/Потребителски_дял
# mkdir /mnt/Потребителска_директория
# mount /dev/Потребителски_дял /mnt/Потребителска_директория

# pacstrap /mnt

# genfstab -U /mnt >> /mnt/etc/fstab
# echo en_US.UTF-8 UTF-8 >> /mnt/etc/locale.gen
# echo LANG=en_US.UTF-8 > /mnt/etc/locale.conf

# arch-chroot /mnt

# locale-gen

# pacman -S linux-libre-lts linux-libre-firmware \
grub bash-completion networkmanager nano
```

```
# grub-install --recheck /dev/Устройство
# mkdir /boot/grub
# grub-mkconfig -o /boot/grub/grub.cfg

# passwd
# useradd -m Потребител
# passwd Потребител

# chown Потребител /Потребителска_директория

# exit
# reboot

# hostnamectl hostname Компютър
# timedatectl set-timezone Континент/град
# systemctl enable --now systemd-timesyncd
# systemctl enable --now NetworkManager

# nmcli dev wifi connect Мрежа password Парола
# ping Сървър
[Ctrl]+[c]

# pacman -S gdm gnome-shell gnome-control-center \
ttf-dejavu lxterminal nautilus

# systemctl enable --now gdm
```

ПЪРВОНАЧАЛНО ПРЕПРОГРАМИРАНЕ НА FLASH-ЧИПА (ARDUINO NANO)

```
# pacman -S git make avr-gcc avr-libc avrdude flashrom

# git clone https://github.com/noblepepper/serprog-duino.git
# cd serprog-duino/

# nano config/user_settings.h
#define S_SPEED        115200    /* Serial speed */
[Ctrl]+[o]; [Enter]; [Ctrl]+[x]

# make ftdi
# make flash-ftdi
```

Свързване: помощен компютър > микроконтролер > flash-чип

(flash-чип с 8 пина)
пин **1** – пин **•** **L1**
пин **2** – пин **•** **L3**
пин **4** – пин **x** **GND**
пин **5** – пин **•** **L2**
пин **6** – пин **•** **L4**
пин **8** – пин **•** **LV**; **3V3**

(flash-чип с 16 пина)
пин **2** – пин **•** **LV**; **3V3**
пин **7** – пин **•** **L1**
пин **8** – пин **•** **L3**
пин **10** – пин **x** **GND**
пин **15** – пин **•** **L2**
пин **16** – пин **•** **L4**

```
# flashrom -p serprog:dev=/dev/ttyUSB0:115200

# flashrom -c Чип -p serprog:dev=/dev/ttyUSB0:115200 -r Резервно_копие
$ md5sum Резервно_копие_1 Резервно_копие_2

wget web-сайт/директории/stable/20230625/roms/Libreboot_пакет

# flashrom -c Чип -p serprog:dev=/dev/ttyUSB0:115200 -w Libreboot_пакет
```

ПЪРВОНАЧАЛНО ПРЕПРОГРАМИРАНЕ НА FLASH-ЧИПА (BEAGLEBONE)

```
# dd if=Файл of=/dev/Устройство bs=64M status=progress oflag=sync
```

Свързване: помощен компютър > USB-Serial > едноплатков компютър

кабел **GND** – пин **x** **1**
кабел **TXD** – пин **•** **4**
кабел **RXD** – пин **•** **5**

```
$ screen /dev/ttyUSB0 115200

$ ssh debian@IP_адрес
# ssh root@IP_адрес
192.168.7.2

debian@beaglebone:~$
password: root
root@beaglebone:~#
[Ctrl]+[d]
[Ctrl]+[d]

[~]
[.]

# /sbin/poweroff

# apt update
```

```
# apt install flashrom
```

Свързване: едноплатков компютър > flash-чип

(flash-чип с 8 пина)

```
пин  1 – щифт • 17
пин  2 – щифт • 21
пин  4 – щифт x  1
пин  5 – щифт • 18
пин  6 – щифт • 22
пин  8 – щифт •  3
```

(flash-чип с 16 пина)

```
пин  2 – щифт •  3
пин  7 – щифт • 17
пин  8 – щифт • 21
пин 10 – щифт x  1
пин 15 – щифт • 18
пин 16 – щифт • 22
```

```
$ screen /dev/ttyUSB0 115200
# ssh root@IP_адрес

# config-pin p9_17 spi_cs
# config-pin p9_18 spi
# config-pin p9_21 spi
# config-pin p9_22 spi_sclk

# /usr/sbin/flashrom -p linux_spi:dev=/dev/spidev0.0,spispeed=512

wget web-сайт/директории/stable/20230625/roms/Libreboot_пакет
libreboot-20230625_x200_8mb.tar.xz

# /usr/sbin/flashrom -p linux_spi:dev=/dev/spidev0.0,spispeed=512 -w \
Адрес/Libreboot_пакет

# /sbin/poweroff
```

ПОСЛЕДВАЩО ПРЕПРОГРАМИРАНЕ НА FLASH-ЧИПА

```
# pacman -S libreboot-utils flashrom
# cbfstool Libreboot_пакет extract -n Файл -f background.jpg
# cbfstool Libreboot_пакет remove -n Файл
# cbfstool Libreboot_пакет add -n Файл -f Файл -t raw

[e]
iomem=relaxed
[Ctrl]+[x]; [F10]

# flashrom -p internal -w Libreboot_пакет
```

```
# timedatectl set-timezone Континент/Град
# systemctl enable --now systemd-timesyncd

# systemctl enable --now NetworkManager
# nmcli dev wifi connect Мрежа password Парола
# dhcpcd

# pacman -Syu
# pacman-key --refresh-keys Адрес_на_неактуализиран_ключ
# pacman -Sy parabola-keyring archlinux-keyring
# pacman-key --refresh-keys

# pacman -S cryptsetup lvm2

# dd if=/dev/urandom of=/dev/Устройство bs=64M \
status=progress oflag=sync

# cryptsetup Настройки luksFormat /dev/Устройство
# cryptsetup --type luks1 -c serpent-xts-plain64 -s 512 \
-h sha512 -i 2000 --use-random luksFormat /dev/Устройство

# cryptsetup open /dev/Устройство Наименование

# dd if=/dev/zero of=/dev/mapper/Наименование bs=64M \
status=progress oflag=sync

# vgcreate Група /dev/mapper/Наименование
# lvcreate -L 8G Група -n Разменен_дял
# lvcreate -L 30G Група -n Системен_дял
# lvcreate -L 20G Група -n Шифрован_дял
# lvcreate -l +100%FREE Група -n Потребителски_дял

# mkfs.ext4 -O^metadata_csum_seed /dev/mapper/Група-Системен_дял

# mkswap /dev/mapper/Група-Разменен_дял
# swapon /dev/mapper/Група-Разменен_дял

# mkfs.ext4 /dev/mapper/Група-Потребителски_дял
# mkdir /mnt/Потребителска_директория
# mount /dev/mapper/Група-Потребителски_дял /mnt/Потребителска_директория

# dd if=/dev/urandom of=/dev/mapper/Група-Шифрован_дял bs=64M \
status=progress oflag=sync
# cryptsetup --type luks1 -c serpent-xts-plain64 -s 512 \
-h sha512 -i 2000 --use-random luksFormat /dev/mapper/Група-Шифрован_дял
# cryptsetup open /dev/mapper/Група-Шифрован_дял Наименование
# dd if=/dev/zero of=/dev/mapper/Група-Шифрован_дял bs=64M status=progress
# mkfs.ext4 /dev/mapper/Група-Шифрован_дял

# mount /dev/mapper/Група-Системен_дял /mnt

# pacstrap /mnt
```

```
# genfstab -U /mnt
# genfstab -U /mnt >> /mnt/etc/fstab
# echo en_US.UTF-8 UTF-8 >> /mnt/etc/locale.gen
# echo LANG=en_US.UTF-8 > /mnt/etc/locale.conf

# arch-chroot /mnt

# locale-gen

# pacman -S linux-libre-lts linux-libre-firmware mkinitcpio \
bash-completion networkmanager nano lvm2 cryptsetup

# nano /etc/mkinitcpio.conf
HOOKS=(base udev autodetect modconf kms keyboard keymap consolefont block
systemd sd-encrypt lvm2 filesystems fsck)
[Ctrl]+[o], [Enter]
[Ctrl]+[x]

# cryptsetup luksUUID /dev/Устройство > /etc/crypttab.initramfs
# nano /etc/crypttab.initramfs
Наименование UUID=Идентификатор none password-echo=no
[Ctrl]+[o], [Enter]
[Ctrl]+[x]

# mkinitcpio -p linux-libre-lts

# passwd

# exit

# pacman -S libreboot-utils flashrom

nano grub.cfg
nano grubtest.cfg

$ cbfstool Libreboot_пакет extract -n grubtest.cfg -f grubtest.cfg
nano grubtest.cfg
menuentry 'Заглавие' --hotkey='Стартиращ_клавиш' {
  cryptomount (ahci0)
  root=lvm/Група-Системен_дял
  linux /boot/vmlinuz-linux-libre-lts root=/dev/mapper/Група-Системен_дял rw
  initrd /boot/initramfs-linux-libre-lts.img
}
[Ctrl]+[o], [Enter]
[Ctrl]+[x]

[a], [b], [e], [c], [d], [o], [p], [r], [s], [t] и [u]

$ cbfstool Libreboot_пакет remove -n grubtest.cfg
$ cbfstool Libreboot_пакет add -n grubtest.cfg -f grubtest.cfg -t raw

# reboot
```

```
# useradd -m Потребител
# passwd Потребител
user
me

# hostnamectl hostname Компютър
computer
comp

# timedatectl set-timezone Континент/град
# systemctl enable --now systemd-timesyncd
# systemctl enable --now NetworkManager

# nmcli dev wifi connect Мрежа password Парола
# dhcpcd

# pacman -S gdm gnome-shell gnome-control-center \
ttf-dejavu lxterminal nautilus

# systemctl enable --now gdm
```

ЦЯЛОСТНО ШИФРОВАНЕ С КЛЮЧ ОТ ВЪНШЕН НОСИТЕЛ

```
(...)

# cryptsetup --type luks1 luksFormat /dev/Устройство

# cryptsetup open /dev/Устройство Наименование

# mkfs.ext4 -O^metadata_csum_seed /dev/mapper/Наименование

# blkid /dev/Твърд_диск > /etc/crypttab.initramfs
# blkid /dev/Външен_носител >> /etc/crypttab.initramfs
# blkid /dev/mapper/Наименование >> /etc/crypttab.initramfs

# nano /etc/crypttab.initramfs
Външен_носител UUID=Идентификатор none read-only,password-echo=no
Твърд_диск UUID=Идентификатор Адрес/Файл:UUID=Идентификатор
[Ctrl]+[o], [Enter]

# mkinitcpio -p linux-libre-lts

# nano grubtest.cfg
menuentry 'Заглавие' --hotkey='Стартиращ_клавиш' {
  cryptomount (usb0)
  cryptomount -k (crypto0)/Адрес/Файл (ahci0)
  root=lvm/Група-Системен_дял
  linux /boot/vmlinuz-linux-libre-lts root=/dev/mapper/Група-Системен_дял rw
  initrd /boot/initramfs-linux-libre-lts.img
}
[Ctrl]+[o], [Enter]
```

```
> cryptomount (usb0)
> cryptomount (usb0,msdos1)

> cryptomount -k (crypto0)/Адрес/Файл (ahci0)

> root=lvm/Група-Системен_дял
> linux /boot/vmlinuz-linux-libre-lts root=/dev/mapper/Група-Системен_дял rw
> initrd /boot/initramfs-linux-libre-lts.img
> boot

# cryptsetup open --key-file Адрес/Файл /dev/Устройство Наименование
```

СКРИТИ ШИФРОВАНИ ПРОСТРАНСТВА

```
# dd if=/dev/urandom of=Директория_или_файл bs=1M count=1 \
status=progress oflag=sync

# dd if=/dev/urandom of=/dev/Устройство bs=64M \
status=progress oflag=sync

# cryptsetup open --type plain -h Хеш_алгоритъм -c Шифър \
-s Дължина /dev/Устройство Наименование

# cryptsetup open --type plain -h sha512 \
-c serpent-xts-plain64 -s 512 /dev/sdb Hidden

# cryptsetup open --type plain -h Хеш_алгоритъм -c Шифър -o Число \
/dev/Устройство Наименование

# cryptsetup open --type plain -h Хеш_алгоритъм -c Шифър -s Дължина \
-o Число --device-size Число /dev/Устройство Наименование

1MB = ( 512B x 2'048) = 1'048'576B

# cryptsetup open --type plain -h Хеш_алгоритъм -c Шифър -s Дължина -r

# cryptsetup open --type plain -h Хеш_алгоритъм -c Шифър -s Дължина \
-r -o Число --device-size Число /dev/Устройство Наименование

# cryptsetup close Наименование
```

ЗАЩИТА СРЕЩУ НЕРАЗРЕШЕНО СТАРТИРАНЕ ОТ ВЪНШЕН НОСИТЕЛ

```
# pacman -S grub

grub.cfg
grubtest.cfg

$ cbfstool Libreboot_пакет extract -n grubtest.cfg -f grubtest.cfg
```

```
(...)
Search ISOLINUX menu (USB)  [u]
Search ISOLINUX menu (CD/DVD)  [d]
Load test configuration (grubtest.cfg) inside of CBFS  [t]
Search for GRUB2 configuration on external media  [s]
Load SeaBIOS (payload) [b]
(...)

$ grub-mkpasswd-pbkdf2 >> grubtest.cfg

nano grubtest.cfg
set superusers=""
password_pbkdf2 Потребител grub.pbkdf2.sha512.10000.Хеширана_парола...
[Ctrl]+[o]; [Enter]
menuentry 'Заглавие_в_менюто' --hotkey='Бърз_бутон' --users Потребител {
      Функция_с_парола
}
[Ctrl]+[o]; [Enter]
menuentry 'Заглавие_в_менюто' --hotkey='Бърз_бутон' --unrestricted {
      Функция_без_парола
}
[Ctrl]+[o]; [Enter]

[r], [p], [Стартиране_на_шифрована_система]
[t], [s], [b]

cbfstool Libreboot_пакет remove -n grubtest.cfg
$ cbfstool Libreboot_пакет add -n grubtest.cfg -f grubtest.cfg -t raw

# flashrom -p internal -w Libreboot_пакет
```

ЗАЩИТА СРЕЩУ НЕРАЗРЕШЕНО ПРЕПРОГРАМИРАНЕ НА FLASH-ЧИПА

```
# pacman -S git gcc pkg-config make
$ git clone \
https://chromium.googlesource.com/chromiumos/third_party/flashrom

$ git checkout 91f320eaab5d7fdd68980b3dbeb64fd30f47aad5
$ make WARNERROR=no

# ./flashrom --wp-status

# ./flashrom --wp-range 0 0xКапацитет
 400000 ≈  4MiB
 800000 ≈  8MiB
1000000 ≈ 16MiB

# ./flashrom --wp-enable
# ./flashrom --wp-status
```

Споряване: пин **3** + пин **4** (flash-чип с **8** пина)
пин **9** + пин **10** (flash-чип с **16** пина)

```
# flashrom -p internal -r Първоначално_тестово_копие.rom
# dd if=/dev/zero of=zero.rom bs=8M count=1
# flashrom -p internal -w zero.rom
# flashrom -p internal -r Последващо_тестово_копие.rom
$ cmp Първоначално_тестово_копие.rom Последващо_тестово_копие.rom
$ md5sum Първоначално_тестово_копие.rom Последващо_тестово_копие.rom
```

ЗАЩИТА СРЕЩУ НЕРАЗРЕШЕНО ЗАКАЧАНЕ КЪМ USB-ПОРТОВЕТЕ

```
> linux /boot/vmlinuz-linux-libre-lts root=/dev/mapper/Група-Системен_дял rw
nousb

# pacman -S usbutils usbguard

$ lsusb

# usbguard generate-policy > /etc/usbguard/rules.conf
# systemctl enable --now usbguard

# usbguard list-devices -b
# usbguard allow-device Номер_на_устройство
# usbguard allow-device -p Номер_на_устройство

# usbguard list-rules
# usbguard remove-rule Номер_на_правило
```

ЗАЩИТЕН ДОСТЪП ДО СИСТЕМАТА ПРИ НАСТЪПИЛО КОМПРОМЕТИРАНЕ

```
[c]
> ls
> root=ahci0
> linux /boot/vmlinuz-linux-libre-lts root=/dev/sda1 rw
> initd /boot/initramfs-linux-libre.img
> boot

[c]
> cryptomount (ahci0)
> root=lvm/Група-Системен_дял
> linux /boot/vmlinuz-linux-libre-lts \
root=/dev/mapper/Група-Системен_дял rw
> initrd /boot/initramfs-linux-libre-lts.img
> boot

> linux /boot/vmlinuz-linux-libre-lts \
root=/dev/mapper/Група-Системен_дял iomem=relaxed

# cryptsetup open /dev/Устройство Наименование
# mount /dev/mapper/Устройство /Точка_на_закачане
# umount /dev/mapper/Наименование
# cryptsetup close Наименование

$ mount -o ro /dev/Устройство /Точка_на_закачане
```

```
# cryptsetup open /dev/Устройство Наименование1
# vgdisplay
# vgrename Идентификатор Ново_наименование
# mount /dev/mapper/Група-Логически_дял /Точка_на_закачане
# cryptsetup open /dev/mapper/Група-Шифрован_дял Наименование2
# mount /dev/mapper/Наименование2 /Точка_на_закачане2
# umount /dev/mapper/Наименование2
# cryptsetup close Наименование2
# umount /dev/mapper/Наименование
# cryptsetup close Наименование
# udisksctl power-off -b /dev/Устройство
```

ВЪНШНИ НЕШИФРОВАНИ ХРАНИЛИЩА

```
# umount /dev/Физически_дял

# mkfs.ext4 /dev/Физически_дял
# mkfs.ext4 /dev/Физически_дял -L Наименование

# mkfs.fat /dev/Физически_дял
# mkfs.fat /dev/Физически_дял -n Наименование

# mount /dev/Физически_дял /Точка_на_закачане
```

ВЪНШНИ ШИФРОВАНИ ХРАНИЛИЩА

```
# umount /dev/Физически_дял

# cryptsetup -c serpent-xts-plain64 -s 512 \
-h sha512 -i 2000 luksFormat /dev/sdb

# cryptsetup open /dev/Устройство Наименование

# mkfs.ext4 /dev/mapper/Наименование
# mount /dev/mapper/Наименование /Точка_на_закачане
# chmod ??? /Точка_на_закачане
# chown Потребител:Група /Точка_на_закачане

# cryptsetup open /dev/Устройство Наименование
# mount /dev/mapper/Наименование /Точка_на_закачане

# umount /dev/mapper/Наименование
# cryptsetup close Наименование
$ udisksctl power-off -b /dev/Устройство
```

СИГУРНОСТ И ПРОМЕНЯНЕ НА ПАРОЛИТЕ

Б^Д
ASCII
Extended ASCII
Unicode Standard

```
$ openssl rand -base64 Брой_символи
```

```
$ tr -dc [:Вид_символи:] < /dev/urandom | head -c Брой_символи; echo
[:digit:]
[:lower:]
[:upper:]
[:punct:]
[:alpha:]
[:alnum:]
[:graph:]
[:print:]

$ tr -dc 'Конкретни_символи' < /dev/urandom | \
head -c Брой_символи; echo

A-Za-z0-9
`~!@#$%^&*()_=+[]{};:'"\|,./<>?-

$ passwd
# passwd
# passwd Потребител

$ gpg --edit-key Ключ

# cryptsetup luksChangeKey /dev/Устройство
# cryptsetup luksAddKey /dev/Устройство
# cryptsetup luksRemoveKey /dev/Устройство

**********
••••••••••

$ echo 'max-cache-ttl 0' >> ~/.gnupg/gpg-agent.conf

$ echo 'pinentry-program /usr/bin/pinentry-tty' \
>> ~/.gnupg/gpg-agent.conf

cat ~/.gnupg/gpg-agent.conf
max-cache-ttl 0
pinentry-program /usr/bin/pinentry-tty
```

УПРАВЛЕНИЕ НА ОСНОВНИЯ КЛЮЧ И ХЕДЪРА

```
# cryptsetup --dump-volume-key luksDump /dev/Устройство

# cryptsetup luksDump /dev/Устройство

# cryptsetup luksHeaderBackup /dev/Устройство \
--header-backup-file Хедър.backup

# cryptsetup luksHeaderRestore /dev/Устройство \
--header-backup-file Хедър.backup

# head -c 2097152 /dev/urandom > /dev/Устройство; sync

# cryptsetup -v isLuks /dev/Устройство
```

```
# cryptsetup open /dev/Устройство Наименование
# mount -o ro /dev/Устройство /Точка_на_закачане

$ gpg --with-keygrip -k Идентификатор

... Created: ГГГГММДДТЧЧММСС ...
... Key: (protected-private-key (rsa ...
... (protected-at "ГГГГММДДТЧЧММСС"))) ...

# LANG=C grep -obUaP "Търсен_низ" /dev/mapper/Група-Системен_дял \
| tee Файл_с_общи_резултати

# dd bs=1 skip=Пореден_байт count=Брой_символи \
if=/dev/mapper/Група-Системен_дял | less

# dd bs=1 skip=Пореден_байт count=Брой_символи \
if=/dev/mapper/Група-Системен_дял of=Файл_с_търсен_резултат

# (while read line; do
offset=$(echo $line | cut -f 1 -d ':')
echo $offset
hexdump -s $(($offset - 256)) -n 256 -C /dev/mapper/Група-Системен_дял
echo
done) < Файл_с_общи_резултати > Файл_с_подбрани_резултати

$ less Файл_с_подбрани_резултати
$ nano Файл_с_подбрани_резултати
[PgDn]; [PgUp]

Created: (...) ))).

# dd bs=1 skip=Пореден_байт count=Брой_символи \
if=/dev/mapper/Група-Системен_дял of=Файл_с_търсен_резултат

Идентификатор.key

/home/Потребител/.gnupg/openpgp-revocs.d
                       private-keys-v1.d
                           Идентификатор1.key
                           Идентификатор2.key
                       pubring.kbx
                       pubring.kbx~
                       trustdb.gpg

$ gpg --with-keygrip -k Идентификатор

$ LANG=C grep -obUaP "Търсен_низ" /dev/mapper/Група-Системен_дял\
| tee Файл_с_общи_резултати

$ dd bs=1 skip=Пореден_байт count=Брой_символи\
if=/dev/mapper/Група-Системен_дял | less
```

```
$ dd bs=1 skip=Пореден_байт count=Брой_символи\
if=/dev/mapper/Група-Системен_дял of=Файл_с_търсен_резултат

$ (while read line; do ofset=$(echo $line | cut -f 1 -d ':'); echo $offset;\
hexdump -s $(($offset - 256)) -n 256 -C /dev/mapper/Група-Системен_дял;\
echo; done) < Файл_с_общи_резултати > Файл_с_подбрани_резултати

$ dd bs=1 skip=Пореден_байт count=Брой_символи\
if=/dev/mapper/Група-Системен_дял of=Файл_с_търсен_резултат
```

ОБЕЗПЕЧАВАНЕ НА АВТЕНТИЧНОСТ И ИНТЕГРИТЕТ

```
$ gpg --search-keys 'Идентификатор'
[n]+[Enter]; [Номер]+[Enter]; [q]+[Enter]

$ gpg --recv-keys Отпечатък
[n]+[Enter]; [Номер]+[Enter]; [q]+[Enter]

$ gpg --import
[Ctrl]+[d]

$ gpg --refresh-keys

$ gpg --search-keys 'Разпознаваема_информация'

$ gpg --fingerprint

$ gpg --recv-keys Отпечатък
$ gpg --send-keys Отпечатък

$ gpg -u Подписващ_частен_ключ --clearsign Текстови_файл

$ gpg -u Подписващ_частен_ключ --detach-sign Бинарен_файл
$ gpg -u Подписващ_частен_ключ -b Бинарен_файл

$ gpg -a -u Подписващ_частен_ключ -r Шифроващ_ключ --sign -e
[Ctrl]+[D]
$ gpg -a -u Подписващ_частен_ключ -r Шифроващ_ключ -s -e
[Ctrl]+[D]

$ gpg -a -u Подписващ_частен_ключ -r Шифроващ_ключ -s -e Текстови_файл

$ gpg -u Подписващ_частен_ключ -r Шифроващ_ключ -s -e Бинарен_файл

$ gpg --verify Проверяван_файл
$ gpg --verify Подпис.sig

$ Стандартизациясum Проверяван_файл
$ Стандартизациясum -c Чексума

$ md5sum Сравняван_файл_1 Сравняван_файл_1
```

„СИМЕТРИЧНО“ ШИФРОВАНЕ

```
$ gpg -a -c
[Enter]
[Ctrl]+[d]
[Ctrl]+[d]

$ gpg
[Enter]
[Ctrl]+[d]

$ gpg -o Желан_бинарен_шифрован_файл -c Съществуващ_нешифрован_файл
$ gpg -o Желан_нешифрован_файл -d Съществуващ_бинарен_шифрован_файл

IDEA
3DES
CAST5
BLOWFISH
AES128        AES192        AES256
TWOFISH128    TWOFISH192    TWOFISH256
CAMELLIA128   CAMELLIA192   CAMELLIA256

$ gpg -a -c --cipher-algo Алгоритъм
[Enter]
[Ctrl]+[d]
[Ctrl]+[d]

$ gpg -o Желан_шифрован_файл -c --cipher-algo \
Алгоритъм Съществуващ_нешифрован_файл
```

СЪЗДАВАНЕ И УПРАВЛЕНИЕ НА „АСИМЕТРИЧНИ“ КЛЮЧОВЕ

```
q = ( n ; e ) = 'Публичен' ключ
p = ( n ; d ) = 'Частен' ключ

n = q * p

e < φ = ( q-1 ) * ( p-1 )
e няма общ делител с φ

d * e ≡ 1 ( mod φ ( n ) )
( d * e ) / n = частно без остатък

c ≡ m ^ e ( mod n ) = Шифровано съдържание
m = c ^ d ( mod n ) = Разшифровано съдържание

RSA
ELG
DSA
ECDH
ECDSA
EDDSA

$ gpg --full-generate-key
[n]; [c]; [e]; [o]; [q]
```

```
$ gpg -a --export Ключ
$ gpg -a --export Ключ > Файл.pub

$ gpg --import
$ gpg --import Файл

$ gpg -a --export-secret-keys Ключ
```

„АСИМЕТРИЧНО“ ШИФРОВАНЕ

```
$ gpg -a -r Ключ -e
[Ctrl]+[d]

$ gpg -a -r Ключ_1 -r Ключ_2 -r Ключ_3 -e
[Ctrl]+[d]

$ gpg -a -r Ключ -e Съществуващ_текстови_файл

$ gpg -r -e Ключ Съществуващ_бинарен_файл

$ gpg
[Enter]
[Ctrl]+[d]

$ gpg Шифрован_файл

.asc
.gpg
```

АНУЛИРАНЕ НА „АСИМЕТРИЧНИ“ КЛЮЧОВЕТЕ

```
$ gpg --gen-revoke Ключ

$ gpg --import
[Ctrl]+[d]

$ gpg --import Сертификат.asc

$ gpg --send-keys Отпечатък

$ gpg --refresh-keys
```

НАШИЯТ 'ПУБЛИЧЕН' КЛЮЧ

Следва нашият 'публичен' **GPG**-ключ за шифроване на поверителната информация, която мислите да изпратите до нас електронно. Информацията ще бъде прегледана чрез нашия 'частен' ключ и ако заедно с написаното от вас сте ни предоставили ваш 'публичен' ключ (или 'отпечатък' за ваш 'публичен' ключ, който междувременно сте качили на някой сървър с 'публични' ключове), отговорът ни ще бъде шифрован с този ключ, преди да го изпратим на посочения от вас адрес. Но преди да се възползвате от тези възможности, не забравяйте да се убедите по предпочитан от вас независим канал, че предложеният 'публичен' ключ наистина принадлежи на този, на когото сте избрали да се доверите!

www.Advocati.org

advocati@gmx.com

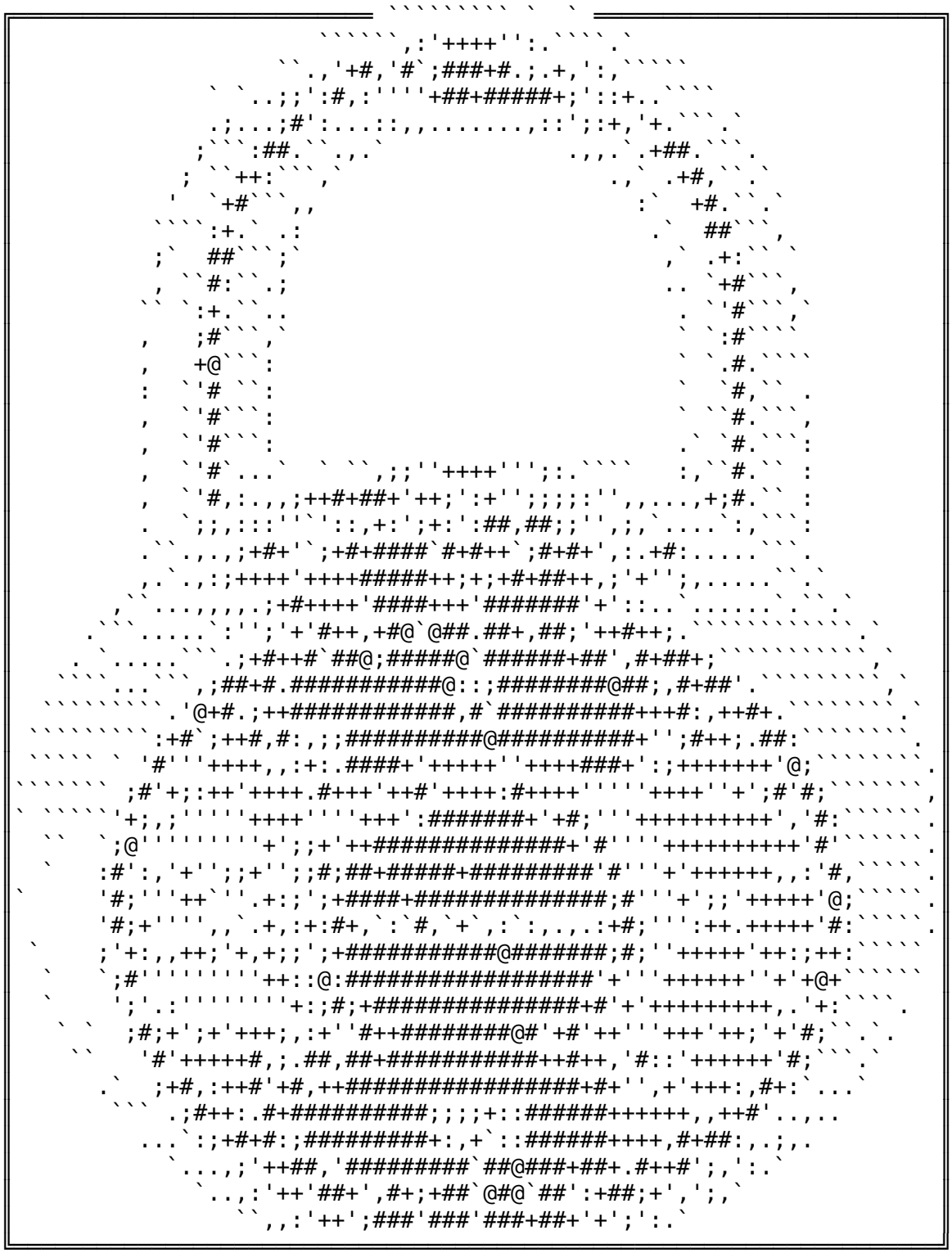
9800FDC33F04EC071AD142CA8F17EED17B9E53A3

КАТО ОБОБЩЕНИЕ

За да ограничите рисковете от компрометиране на вашата информационна сигурност (доколкото изобщо това е възможно в електронна среда), ви насърчаваме:

- ☐ да ползвате усилено Свободна операционна система, базирана на **GNU/Linux-libre**;
- ☐ да се доверявате само на сигурно компютърно оборудване (без никакви несвободни компоненти);
- ☐ да замените **BIOS/UEFI** със свободен софтуер от „ниско“ ниво;
- ☐ да обновявате системата си редовно до последни версии на софтуера;
- ☐ да избягвате графичния интерфейс и да работите с командния ред;
- ☐ да ползвате сложни и дълги пароли, да ги променяте редовно и да сте особено внимателни при въвеждането им;
- ☐ да шифровате изцяло вашия твърд диск, като изнесете стартиращата програма на външен носител;
- ☐ да блокирате по-нататъшното препрограмиране на вашия flash-чип;
- ☐ да ограничите стартирането от „живи“ носители без парола;
- ☐ да пазите вашите устройства от неразрешена физическа намеса, включително като запечатате корпуса срещу неразрешено проникване;
- ☐ да проверявате редовно закачените към **USB**-портовете устройства, като блокирате неразрешеното закачане на **USB**-устройства към системата;
- ☐ да се интересувате активно от Свободните технологии и по възможност да ремонтирате лично вашите устройства;
- ☐ да имате на разположение няколко резервни устройства и в случай на съмнение да смените незабавно устройството;
- ☐ да не разшифровате вашата поверителна информация, докато сте свързани с internet и по възможност да я съхранявате само на устройства, които изобщо не се свързват с internet;
- ☐ да въвеждате най-съкровените си тайни само като прост текст в терминала и изобщо да не ги записвате, преди да сте ги шифровали;
- ☐ да спазвате определена дисциплина и предпазливост в internet, като „разделяте“ различните си активности и не ги „смесвате“;
- ☐ да проверявате внимателно 'отпечатъците' на 'публичните' ключове чрез предпочитани от вас независими канали за потвърждение;
- ☐ да поддържате готовност да анулирате компрометирани ключове и да уведомите незабавно вашите кореспонденти за това;
- ☐ да изисквате вашите довереници да боравят с поверителна информация по достатъчно надежден начин, преди да започнете да споделяте;
- ☐ и никога да не разчитате на един единствен слой сигурност, като помните, че „абсолютна сигурност“ не съществува. Още по-малко в електронна среда.

Можете да ползвате този „списък“ като своеобразен проверовъчен лист за проследяване степента на обезпечаване на вашата и на вашите довереници информационна сигурност съгласно настоящото ръководство.



© Благодарим ви, че проявихте търпение да се запознаете с нашето ръководство! То (както и цялото съдържание на **www.Advocati.org** и **www.LibTec.org**) се разпространява съгласно усилено свободния лиценз **GNU Free Documentation License** (**GFDL**). Насърчаваме ви да го ползвате свободно и споделяте с всеки, който би имал нужда.

https://www.gnu.org/licenses/fdl.html
https://advocati.org/
https://libtec.org/