

# End-user Computer Security

Inexpensive security

for   ⊙ individuals,   ⊙ sole traders,   and   ⊙ small businesses

*First version written by Mark Fernandes in April 2020.*



# Preliminaries

## Meta information

### Table of Contents

<b>Preliminaries</b>	<b>2</b>
Meta information	2
Table of Contents	2
Foreword to first version	7
<b>Main content</b>	<b>8</b>
Software based	8
Security of BIOS/UEFI firmware	8
Custom BIOS/UEFI and which one to use	8
Regarding operating system	9
Which OS?	9
Qubes OS 4.0.3 side-by-side with other operating systems	10
How to ensure installed operating system is not compromised via an evil maid attack, during periods when machine is not meant to be on	11
Regarding how to obtain software	11
Secure downloading	12
Getting an uncompromised smartphone and obtaining software with it	12
Key advantage	12
Downloading to SD cards	12
Transferring downloads to installation media	12
Cost	12
Old or new phone	13
Some other advantages	13
Similar to 'burner phones'?	13
Whether to use a Raspberry Pi Zero device instead	14
Pros vs Cons	14
Pros:	14
Cons:	15
Conclusion	16
About using a Wi-Fi enabled SD card	16
Detection of malware in software	17
Compiling from source	17
Reproducible builds	18

Using `diff` utilities	18
Full system encryption, full disk encryption (FDE)	18
Malicious sneaky replacement of FDE system with historic clone of system that has known vulnerabilities	19
Description of attack	19
Remedy	19
Bootloader for FDE	19
Factory resets	20
According to device type	20
Web client computers	20
Smartphones	20
Conventional laptops	20
Some other issues	21
Internet connection during reset process	21
Sufficiency	21
Cookies	21
Sandboxing and cloud computing	22
Passwords and digital keys	23
Password security	23
Password managers	23
Multi-step authentication and multi-factor authentication	24
Non-Latin alphabet	24
Concerning certain password attack vectors	24
Screen privacy	24
Keyboard privacy	25
Visual spying of keys pressed	25
Spying of electronic keyboard signals	27
General visual spying	28
Hiding materially-written passwords	28
Overcoming vulnerabilities in visual encodings	29
Psychic spying of password	30
Protection using password management functionality	30
Protection by thinking little	30
Protection using password encryption	30
Based on password reuse	32
Digital cryptography:	
security certificates, keys & tokens	33
Disabling TLS security certificates	33
Making sure certificates are genuine	33
Key servers	33
Cross authentication	34
Non-compromised communication of public keys	34

Sending to trusted recipient such that the recipient can hand it over without encountering MITM vulnerabilities, to the end-user	34
Publishing public keys in a “gazette”	34
Piggy-backing over bank transactions and systems	35
Using bank references	35
Using different monetary amounts	35
Using a weak currency	35
Google Authenticator “key and time”-based app for security	36
Tokens for keys	36
External links for further information on security certificates	36
Backing-up security keys and passwords	37
Shamir's Secret Sharing	37
Wireless Communications	38
Wired vs. wireless	38
Shared WiFi	38
Keep communication systems turned off	38
Digital storage	39
General security risks in digital storage	39
USB devices vs. SD cards	39
Flash memory: NOR flash vs NAND flash	40
NAND flash memory vs magnetic storage	40
Magnetic storage: tapes vs. discs	41
Rewritable media vs optical ROM discs	41
SD cards and USB memory sticks vs. larger devices	41
Drives able to eject hardware-less media vs. other media	42
More about SD cards	42
How to obtain computer media devices	42
Secure data sanitisation	42
Some measures that are primarily physical	43
Physical isolation and locks	43
Physical measures for securing bootloader when using full-system encryption	43
Padlockable laptop bag	43
Metal boxes	43
Combination lock briefcase	44
Physically removing storage component(s) from the rest of the computer system, and then securely storing those components separately	44
Physically securing keys	45
Privacy screens	45
Specifically for goods in physical transit	45
Exploiting unrepeatable patterns for tamper evidence	46
Applying glitter nail varnish to computer screws	46

Tamper-evident security-system ideas	47
Main idea	47
Speculating stronger security again with unrepeatable-pattern principle	49
Similar idea for other circumstances (such as for metal boxes)	49
Perhaps the simplest and best idea	50
Software based tamper checking using security images	50
Mind-reading attacks	52
‘Inception’ styled attacks	52
Simple security measures	53
Put computer to sleep when not at it	53
Shut down device when not in use	53
Play sound continuously on computing device	53
Broad security principles	54
Stop funding the spies and hackers	54
Report cybercrime to the police	54
Think in terms of gradual movement along a security-level continuum	54
Minimally-above-average security	55
Publishing security methods	55
User randomly selecting unit from off physical shelves	56
When random is not random	56
Ordering many units of same product	56
Using multiple channels to obtain product	56
Discerning unit least likely to have been compromised	57
Measuring physical properties for authentication	57
Weight	57
Volume	57
Magnetic weight and images	58
Electric field imaging/detection	58
Electro-magnetic spectrum	59
Visible spectrum photography	59
Infra-red scanning	59
X rays	59
Microwave testing	59
Radio frequency (RF) imaging/detection	59
Ultrasound	60
Other methods	60
Geospatial	60
Based on which region	60
Time based	60
Based on time passed	60
Example 1	60

Example 2	61
Vulnerability when used for software	61
Based on time taken to forge	61
Using most secure window of time	62
Preventing lapses in security	62
DIY security principle	63
“Destroy key when attacked”	63
Relying on high production cost of certain security tokens	63
What to do when you discover your computer has been hacked	64
Backing-up files	64
When to change digital passwords and keys?	64
Further information	64
Miscellaneous notes	65
National Cyber Security Centre	65
Cybersecurity standards	65
Deep hardware hacking	65
Cryptocurrency security	65
Using phones and computers as motion detector alarms	66
Steganography: easy hiding of information in computer documents	66
<b>Appendix</b>	<b>67</b>
New security inventions requiring a non-trivial investment in new technology	67
Cryptocurrency-like mining to increase trust	67
As applied to software	67
As applied to public digital keys (including those used in security certificates)	68
Lock screen with related sound-based security	68
Client-server noise-audio-based secure-password-communication system	69
Port source code to higher-level programming language as a computer-security step having its basis in secure coding	70
Security by pre-loaded private key	70

---

## Foreword to first version

This book was first produced in response to a computer [hacking](#) incident encountered during 2020 by the author of the first version of this book, in the course of his being a [self-employed software developer](#). He had already adopted some [security measures](#) but then felt he really needed an overhaul of the security measures and systems he had in place.

This book is aimed specifically at individuals, [sole traders](#), and [small businesses](#), bearing in mind that they may have [shoestring budgets](#).

It was the author's belief that [end-user security](#) was a real issue of concern because the mindsets of security specialists seemed to be often attuned to examining and proposing solutions within rigid frameworks: such as for example only looking at software security risks but completely ignoring physical aspects of everyday [nuts-and-bolts](#) security. A certain element of being able to [think 'outside the box'](#), and outside one's own specialised domain, is needed. As such, security is really a multidisciplinary field, requiring the creativity of people from all walks of life.

There is special concern for the highlighted entities (individuals, sole traders, and small businesses), because of their being prone to attack due to budget constraints, and a lack of other important resources.

The author of the first version of the book places his contributions into the [public domain](#) (the author's [Google Drive](#) version hosted [here](#) [minus the [Google Docs comments](#)] will always be in the public domain). He feels that end-user security is so important, that intellectual property obstacles should be removed as much as possible, so as to enable everyday users to be able to undertake computing activities safely. This is especially of concern at the time of writing during the 2020 [COVID-19](#) worldwide outbreak. During this outbreak, individuals are being called upon in great numbers to [remote work](#) and also to socialise and conduct recreational activities using computing devices.

The increasing [consumer](#) use of [cryptocurrencies](#) is another reason why a work like this is important.

The author only asks in return that you, if possible, do the following:

1. Amend this work to fix mistakes.
2. Add comments indicating your level of agreement or disagreement with different parts that you read/review.
3. Improve it in other ways.

Please note that because using your contributions might require that you grant [copyright permission](#) for such, it is mostly preferred that you make your contributions to the [Wikibooks version](#) of this book<sup>1</sup>.

---

<sup>1</sup> If you are only pointing out mistakes, or giving [quantitative](#) measurements of your agreement/disagreement with parts of this book, [copyright permission](#) probably won't be required.

# Main content

## Software based

### Security of BIOS/UEFI firmware

A computer may be [hacked](#) due to its [BIOS/UEFI firmware](#) being infected with [malware](#) (such infection occurring due to malevolent forces). This applies to [smartphones](#), [thin clients](#), and conventional [PCs](#).

Because of this possibility, sometimes it is necessary to [reinstall](#) the BIOS/UEFI firmware.

Custom [BIOS/UEFI](#) and which one to use

A custom [BIOS/UEFI firmware](#) that has higher security to a [hardware](#) vendor's BIOS/UEFI, can be [installed](#). It sounds like a good idea to do this. The [Qubes OS 4.0.3](#) guidance in fact recommends installing, in place of a hardware vendor's BIOS, the custom BIOS/UEFI called [Coreboot](#).

*"All ChromeOS devices (Chromebooks, Chromeboxes, Chromebit, etc) released from 2012 onward use coreboot for their main system firmware. ..."*

[\[https://doc.coreboot.org/distributions.html\]](https://doc.coreboot.org/distributions.html)

The above excerpt indicates that [flashing Chromebooks](#) with a Coreboot firmware freshly downloaded through a hopefully non-compromised device, likely works. Using the [MrChromebox Coreboot firmware](#) for a Chromebook seems likely to be more secure than the standard Chromebook firmware.

It seems that Coreboot can [also be installed as the firmware for an Android phone](#).

The [Heads](#) BIOS/UEFI boot firmware system keeps a [bootloader](#) stored in [ROM](#), so that the bootloader no longer has to be an [attack vector](#) in relation to storing a bootloader on a drive ([HDD](#), [SSD](#), etc.) Heads appears either to be an adaptation of Coreboot, or a system that runs over Coreboot. Either way, using Heads instead of vanilla-flavoured Coreboot appears to provide better security.

Heads (and perhaps also Coreboot), can make use of a [motherboard](#)'s TPM ([Trusted Platform Module](#)), to ensure computer firmware has not undergone any tampering<sup>2</sup>. It can also be used to ensure a high-quality [password](#) security system, partly because of the [destroy-key-when-attacked security](#) but also because of the system being present in the TPM.

The National Cyber Security Centre for the UK ([NCSC](#)) highlights the importance of updating firmware, and provides related guidance and information (see [here](#) and [here](#)).

---

<sup>2</sup> Perhaps in an almost full-proof way by the judicious use of [epoxy](#) resin on the [motherboard](#) to encase certain key parts (such as pins and [chips](#)).

## Regarding operating system

Installed operating systems ([OSes](#)) must be safeguarded—they are vital for ensuring computer security.

Using an [OEM](#) preinstalled OS is prone to the [MITM](#) attack of having the related machine intercepted and hacked before reaching the customer (especially on a targeted-individual basis). More secure ways for obtaining an OS's [installation software](#) are described in this document, in the section entitled "[Regarding how to obtain software](#)".

Once installed, the security provided by the OS, will likely mean that one can then [download](#) further [software](#) from over the [internet](#) without much worry (assuming reasonable security precautions are taken).

Some general security advice in relation to using an operating system, is for users to have an [administrator account](#) that is different to the standard account users use for everyday computing. The standard account has fewer [privileges](#). The more powerful administrator account, that also has higher associated security risks, should also have a "minimised" exposure to risk due to it being used only when needed—when not needed, the less risky standard account is instead used.

### Which [OS](#)?

The [Linux](#) operating system ([OS](#)) has a [reputation for being more secure](#) than the [Windows](#) operating system. Being able to [compile Linux from source](#), and not being able to do the same with Windows, adds to Linux's security, which is expanded upon later on in this document (in the section entitled "[Compiling from source](#)").

It can be slightly irritating to have to choose Linux over Windows, given the various advantages in using Windows, but can be necessary to ensure security. Windows [emulators](#) such as [Wine](#), can perhaps be effectively always used whenever needing to run some software meant to be run over Windows. There is unlikely much point in [dual-booting](#) between Windows and Linux because if Windows is hacked, then the Linux installation can then also be hacked through the hacked Windows, thereby undermining the security advantages Linux affords; in fact this point is specifically mentioned in [guidance on the Qubes website](#).

If going for Linux, the [Qubes 4.0.3](#) distribution of Linux appears to be perhaps the best in terms of being most secure, and is at least one of the most secure distributions of Linux. So installing Qubes OS 4.0.3, if installing a Linux distribution, seems like a good idea.

Some platform-specific (and OS) guidance from the National Cyber Security Centre ([NCSC](#)) is available [here](#).

### Qubes OS 4.0.3 side-by-side with other operating systems

[Qubes OS 4.0.3](#) is [documented as not coping well](#) with [software](#) that specifically benefits from [3D-optimised hardware](#). Since a user may well want to use such optimisation, the best way to use such optimisation on the same machine might be to do something like, or the same as, the following:

1. [Install](#) a [Linux operating system](#), with good security but still with the capacity for being able to utilise 3D-optimised hardware, on an [SSD](#) external [drive](#), such that this other operating system is not run over Qubes, but instead run separate to Qubes.
2. When wanting to use this other Linux OS, disable the internal drive (containing Qubes) in either:

- a. the [BIOS](#),

*OR IF WISHING TO BE MORE SECURE,*

- b. both the BIOS

as well as by physically disconnecting the internal drive

*(this latter option might be a good idea to do  
because [malware](#) in a BIOS's [firmware](#)  
can still connect to BIOS-disabled drives).*

3. [Boot](#) off the SSD to run this other Linux.
4. After using the non-Qubes installation, because of the possibility of malware being introduced into the BIOS firmware by the non-Qubes installation, optionally [flash](#) the BIOS's firmware to ensure better the Qubes installation isn't compromised through firmware [malware](#) when you next use Qubes.

By following the above steps, and choosing the most secure options in the steps, because of:

- the disabling of the internal drive via the BIOS,
- the physical disconnection of the drive containing the Qubes installation, *and*
- the flashing of the BIOS firmware before the 'reconnection' of the Qubes installation,

any such other OS should not be able to access or even 'touch' the Qubes OS installation, thereby hopefully safeguarding the Qubes installation from attacks conducted through the other presumably-less-secure OS.

How to ensure installed [operating system](#) is not compromised via an [evil maid attack](#), during periods when machine is not meant to be on

If:

1. [full-system encryption](#) is used,
2. the [operating system](#) has a high level of security,
3. the risk of the hardware being or becoming compromised is very low,
4. the [bootloader](#) is kept on a secure [USB memory stick](#) (or [SD card](#)) that is locked away when not needed,
5. the [password](#) and [key](#) (as applicable) are made difficult to obtain (whether by [password cracking](#) or other means), perhaps with the aid of a [USB security token](#)<sup>3</sup> product

AND

6. the system is re-[encrypted](#) with a brand new key every so often *(to mitigate against the exploitation of newly discovered [security vulnerabilities](#) that are in historic copies of the encrypted system<sup>4</sup>)*,

then it appears that this will quite likely adequately secure the operating system from becoming compromised via an [evil maid attack](#), during those periods in which legitimate users have set the machine to be in a powered-off state.

An alternative way is to keep the operating system stored on a [read-only live DVD](#)<sup>5</sup> that you [lock away](#), and of which you perhaps keep multiple copies stored in different locations in order to help make sure any one selected copy is the “genuine article”<sup>6</sup>; however, be aware that [system speed is often degraded](#) when running the operating system in such fashion.

## Regarding how to obtain [software](#)

[Secret criminal societies](#) will likely find it much harder to replace every single copy of a particular [software](#) with [malware](#)-modified versions, in a large physical shop, than to do the same replacement with [downloads](#) or goods delivered to specific end-users. Because of this, the broad security principle outlined in the “[User randomly selecting unit from off physical shelves](#)” subsection probably applies at least to the obtaining of the foundation software required for operating a computer (i.e. the [operating system](#), [BIOS firmware](#), etc.) and at least in terms of part of the overall solution used.

The broad security principle(s) outlined in the “[Ordering many units of same product](#)” sub-section also applies.

Borrowing [library](#) software [DVDs](#) and [CDs](#) may be another [channel](#) through which software can be usefully obtained.

---

<sup>3</sup> Such products are partly dealt with later on in the section entitled “[Tokens for keys](#)”.

<sup>4</sup> See the section entitled “[Malicious sneaky replacement of FDE system with historic clone of system that has known vulnerabilities](#)” for more about this kind of [attack](#).

<sup>5</sup> See the section entitled “[Rewritable media vs optical ROM discs](#)” for best practices to ensure [optical discs](#) are indeed [read-only](#).

<sup>6</sup> Keeping multiple copies like this, is similar to the principles outlined in the “[Using multiple channels to obtain product](#)” section.

Once an operating system and BIOS firmware have been (re-)[installed](#) for a computing device, downloading software through the device with standard security precautions in place (eg. using [HTTPS](#) connections, [verifying signatures](#), etc.) may become sufficiently secure for obtaining further software.

### Secure [downloading](#)

What are you to do if all your computing devices have been compromised, and security cannot be reestablished in the devices except through using certain software on the devices that is available only via [internet downloading](#) (such as certain [firmware re-installers](#))? It can be expensive to buy a brand new computer, if not on one particular occasion, then on some number of occasions whenever security needs to be re-established.

### Getting an uncompromised [smartphone](#) and obtaining [software](#) with it

#### Key advantage

The advent of [smartphones](#) appears to be of great benefit to this problem. It looks like a cheap uncompromised smartphone for simply [downloading files](#) over [WiFi](#), that is [shrink-wrapped](#) and boxed fresh from its factory, can easily be bought by first physically visiting a large store geographically distant from an [end user](#)'s address, and then secondly by [randomly](#) and personally selecting such a phone from very many other such phones, off the physical shelves of the visited store (principle outlined in "[User randomly selecting unit from off physical shelves](#)" section).

#### [Downloading to SD cards](#)

The section entitled "[Digital storage](#)" in this document, provides information about some of the security issues surrounding the use of [SD cards](#) when using them in [mobile devices](#) for the [downloading](#) of [software](#).

#### Transferring [downloads](#) to [installation media](#)

Getting the [downloads](#) on to suitable [installation media](#) for computing devices is perhaps something about which to think. It seems it is possible to get [DVD burners](#) and [USB memory sticks](#) for [smartphones](#). You can also get [USB adapters for the SD cards](#) used in [mobile phones](#). Many [laptops](#) have built-in [card readers](#) for interfacing with [SD cards](#), and on such laptops, according to [Wikipedia](#), it appears it is sometimes possible to [boot off an SD card](#), which has the potential of greatly simplifying things. From brief [internet research](#), formatting media used just for storage and retrieval (eg. [DVDs](#) with a [DVD drive](#), [memory sticks](#), SD cards, etc.) as [bootable media](#), solely by means of an [Android mobile device](#) and the [storage media](#) connected together, is quite likely not very difficult to do. Bootable media is often needed when [reinstalling](#) certain classes of [software](#). Please consult the section entitled "[Digital storage](#)" for further guidance on which storage medium to use.

## Cost

The price of purchasing such a phone might initially be of concern, with minimum prices for such a phone perhaps starting at around the [£40 mark](#). However, it should be possible to recoup some of the money spent, by simply selling the phone on as a [second-hand](#) device once it has been used. There is a possibility that a [tablet](#) might be cheaper than a phone; prices should be investigated regarding this.

## Old or new phone

It is also generally better to use a brand new phone, rather than an old phone even if you are sure the old phone hasn't undergone any [tampering](#). The reason being, is that brand new phones should be more secure, due to:

- things like [security hole fixes](#),
- possible [improvements in technology](#), AND
- some of the [security certificates](#) in the new phone likely being as new as possible for their respective class of certificate.

## Some other advantages

Using a [smartphone](#) or [tablet](#), perhaps is more secure than using a [computer keyboard](#), because a [screen-based](#) and [software](#)-based key [scrambler](#) perhaps effectively reduces or removes the effectiveness of [key loggers](#). Also, smartphones are very popular, this should make [random](#) selection by physically going to a store<sup>7</sup>, much easier and more potent (because so many units should be available 'on the shelves' and because more stores should stock them.) Also, reliance can be made on a higher amount of security testing because of the greater market need for [secure mobile phones](#), and because of the greater unit production, compared with devices like the [Raspberry Pi](#) device.

## Similar to 'burner phones'?

Interestingly, using so-called '[burner phones](#)', seems to be closely related to the suggestions made in this "[Getting an uncompromised smartphone and obtaining software with it](#)" section.

---

<sup>7</sup> As detailed later on in the section entitled "[User randomly selecting unit from off physical shelves](#)".

## Whether to use a [Raspberry Pi Zero](#) device instead

It initially seemed that the most

- appropriate,
- inexpensive, and perhaps even
- secure way to [download](#) securely,

especially when you suspect your existing computers may have been compromised, was to buy a brand new [Raspberry Pi Zero](#) product, from off the physical shelves of a store, by means of users personally selecting a device at the store using [random selection](#)<sup>8</sup>, and then to use that device for downloading.

### Pros vs Cons

#### *Pros:*

- Because of the device's prevalence and unit price, it probably can be easily bought by physically visiting a store and then personally selecting a unit by [random selection](#) from off the physical shelves<sup>9</sup>—this protocol is a good way to ensure better no [tampering](#) or [fraudulent](#) replacement has been performed on the unit.
- The [OS](#) integrity can perhaps be further [trusted](#) by first [copying](#) the OS off onto an [SD card](#) or [USB memory stick](#), and then by using another computer, sometimes even if the other computer has been potentially compromised, to check whether the OS matches a copy of the OS downloaded through the other computer.
- Very cheap, priced at around [£5 brand new](#).
- It is stripped down to remove many ['bells and whistles'](#) that would otherwise introduce further risks.
- Doesn't have [WiFi](#), which forces users to use a [wired connection](#) for the [internet](#), which is more secure and actually ideal<sup>10</sup>.
- The [operating system](#) that it uses is a [Linux distribution](#), [Linux](#) is known to be relatively secure<sup>11</sup>; also, because it is Linux and [open-source](#), it should be possible to use it to do things like create [bootable media](#) using the [Pi](#) device, which may be needed for the re-[installing](#) of [firmware](#) and more generally, of compromised systems.
- The operating system is updated in a timely manner for better security.
- [Raspberry Pi](#) has been around a while, so is likely more [trustworthy](#) and secure because of that.

---

<sup>8</sup> As detailed later on in the section entitled "[User randomly selecting unit from off physical shelves](#)".

<sup>9</sup> As detailed later on in the section entitled "[User randomly selecting unit from off physical shelves](#)".

<sup>10</sup> See the later section called "[Wired vs. wireless](#)", for more about this.

<sup>11</sup> See the section entitled "[Which OS?](#)" for more about this.

- It appears to have a wide audience, including individuals in the security community, and so likely has improved security and reliability because of that.
- It has [USB](#) connectors for connecting [memory sticks](#).
- It is freely open for [visual inspection](#) of its parts, which has been suggested as enabling easier detection of [hardware tampering](#)<sup>12</sup>.
- It is small, such that securing it, such as in a
  - [safe](#),
  - hidden location,
  - [tamper-evident](#)<sup>13</sup> container, or
  - otherwise,
 might generally be cheaper.

*Cons:*

- Appears to require a [USB keyboard](#).  
According to [Micah Lee](#) in the [Qubes OS](#) video hosted [here](#) <sup>(go to 29m:47s)</sup>,  
using [USB](#) instead of [PS/2](#) for plugging in your keyboard, is something of a security risk.
- Appears to require a [screen](#), which is a [vulnerability](#).  
*For example,*  
if plugging into a [TV](#),  
a [hacker](#) can take control of the TV output to display a completely different output in order to [phish](#) for your security credentials.
- Whilst it is true that related [hardware](#) can be bought in such fashion that the [I/O](#)-device vulnerabilities are not so important, doing so whacks up the overall cost of the set-up in such fashion that alternatives become more appealing.  
Also, the related [hardware](#) somewhat reduces the advantage of removing unnecessary '[bells and whistles](#)'  
(in other words, it reduces the advantage of keeping things simple and thus less risky.)

<sup>12</sup> See [Andrew "bunnie" Huang's](#) blog post entitled "[Can We Build Trustable Hardware?](#)".

<sup>13</sup> See the

- "[Specifically for goods in physical transit](#)" and
- "[Exploiting unrepeatable patterns for tamper evidence](#)"

sections for special information on [tamper-evident systems](#).

## Conclusion

In spite of the cons outlined above, a primary advantage of a [Raspberry Pi](#) or similar solution, remains—namely, that of it moving along the lines of being fairly [bare-bones](#). With some [tweaking](#), the Raspberry Pi or a similar device, can be made part of an effective security solution to the problem outlined in this writing, and perhaps could even be a marketable [product](#). However, until such a product exists, because of the cons, it is advised that one use a [smartphone](#) (as described in the previous section) rather than a Raspberry Pi Zero device. It's a shame because with some fairly straight-forward modification, perhaps the [Raspberry Pi Zero](#) device would be ideal as the main device used to solve the security problem addressed here.

## About using a [Wi-Fi enabled SD card](#)

According to [Andrew "bunnie" Huang](#), because of the presence of [reprogrammable firmware](#) “strapped” to a reasonably powerful [microcontroller](#), combined with a high amount of [flash memory](#), with all of these things [embedded](#) in single [SD cards](#) costing very little, SD cards [could be a very cheap source of hardware](#) for [DIY](#) projects. Additionally, SD cards carrying [WiFi hardware](#) also exist. Therefore, [reprogramming](#) a [WiFi enabled SD card](#) as a “secure [downloader](#)” (as just suggested for some adaptation of the [Raspberry Pi Zero](#) device) may well be possible. If possible, it may be cheaper than trying to do something similar with a “[Raspberry Pi](#)” kind of device, as well as more secure because it has so many fewer “[bells and whistles](#)”, and because SD cards are much smaller meaning that it is more difficult to [tamper](#) physically with SD cards. However, it should be borne in mind that reprogramming it appears only to be able to take the form of a [hacking](#) kind of project because the specifications for interfacing with such embedded microcontrollers appear to be entirely unavailable outside of the production processes used for SD cards.

## Detection of [malware](#) in [software](#)

### Compiling from [source](#)

Should [Qubes OS](#) (and other [software](#)) be [built](#) from scratch? Reason would seem to suggest that detecting [malware](#) in software is easier if you have the software's [source code](#) to check. The reasons for this are as follows:

1. with the source code,  
you can check both
  - the source code and
  - the [compiled code](#)for malware,
2. certain malware patterns are quite likely more discernible upon examination of the source code rather than the compiled code, A N D
3. discrepancies between freshly compiled source code, and historically compiled source code, can uncover malware introductions that occur post compilation.

[GitHub](#)'s security information published [here](#) seems to support point 2. Reflection on how [programming](#) mostly involves the application of [design patterns](#) (such as the implementation of certain [algorithms](#) [data structures](#) etc.) also clearly inclines one to believe that malware can be detected in source code on occasions where detection in compiled code is either difficult or impossible. Also, publicly hosted source code (such as in public [GitHub repositories](#)), can have “many eyes on it” through public [peer review](#) of the [code](#), including those of [security researchers](#), checking over the code (sometimes simply as a step to accomplishing something completely different, such as the contributing of new code to a repository) so as to reveal malware. GitHub also appears to have special provisions for detecting malware, as documented at the GitHub [internet address](#) just mentioned. If the detection of malware in source code is computationally expensive, perhaps providers like GitHub, can (or maybe they already do) run automated [AI](#) malware-detection algorithms in the background on their repositories using powerful [servers](#), perhaps sometimes running for several months, for the detection of malware in source code. Also, GitHub's [collaborative development](#) structure essentially [audits](#) all source code changes, which undermines many of the efforts aimed at trying to introduce malware into software. These thoughts incline one towards choosing to obtain source code from GitHub rather than the website of some specific software development entity. They also incline one towards choosing to build from source.

## Reproducible builds

[Trammell Hudson](#)<sup>14</sup> said that [reproducible builds](#) are important in his 2016 [33c3](#) talk, the video of which is hosted [here](#) [go to time 31m:07s] (presumably for security reasons).

It's not clear that reproducible builds offer a significant security benefit over simply [compiling](#) from [source](#). However, they probably do given the promotion of it by other notable individuals. It seems like reproducible builds are good at uncovering certain security compromises more in relation to compromises in the Provider's system than compromises in the user's system. However, since the two (the Provider's system and the user's system) are interdependent, the uncovering of such compromises necessarily means that the user's security is also somewhat increased.

### Using ``diff`` utilities

``Diffoscope`` appears to be a potentially useful utility for detecting [malware](#) introductions between released versions of [closed-source software](#), when the “[reproducible builds](#) protocol” is being used in [software compilation](#). The [Wikipedia](#) page on [reverse engineering](#), implies that [extracting](#) the [source code](#) from such closed-source software, in an automated way, and then running ``diffs`` on the source code, likely can better uncover malware introductions.

## Full system encryption, full disk encryption (FDE)

[Full-system encryption](#) should probably be considered as required when using a computer for business purposes.

There are [various pieces of software](#) that can provide such [encryption](#). Usefully, the [Qubes OS FAQ](#) says that [Qubes OS 4.0.3](#) has full-disk encryption installed by default, so if you use Qubes 4.0.3, you probably won't need to take extra steps to make sure full-disk encryption is set-up.

---

<sup>14</sup> The creator of the high-security [Heads bootloader firmware](#) system.

## Malicious sneaky replacement of FDE system with historic clone of system that has known vulnerabilities

### Description of attack

An attacker can exploit that a particular system has been full-disk encrypted with the same key used over a long period of time. For example, suppose Windows was full-disk encrypted with the same key three years' ago, and that an attacker cloned the system three years' ago. Then suppose that in the present day it was known that there were security vulnerabilities with that historic cloned Windows installation. An attacker could maliciously replace the contents of the system disk with those recorded three years ago, unbeknownst to the user. The user logs in, with their same password, and doesn't suspect that the system is three years old. Then the attacker can potentially exploit security vulnerabilities in the system that is used by the user in the present day, even though the system is in fact the user's system from three years ago, with security holes in it that are known about in the present day.

### Remedy

Way to mitigate against this type of attack: change full-disk encryption keys and passwords on a regular basis, quite frequently (perhaps once per month) but not too frequently so as to wear out the system disk unduly (usually a HDD or SSD). Interestingly, could this solution benefit from old hard disk drives that users give away for free, so as to mitigate against the costs of having to deal with the deterioration of drives due to them being frequently re-encrypted each time the key changes

### Bootloader for FDE

As was hinted at in the subsection entitled "How to ensure installed operating system is not compromised during periods when machine is not meant to be on, via an evil maid attack", securely storing the bootloader away from the computer during periods when your computer should be in a powered-off state, is a good idea. The reason is that the bootloader, as stored in the computer system, is a potential attack vector<sup>15</sup>. The Heads BIOS firmware system partly overcomes the traditional form of the attack by actually storing the bootloader in the internal ROM of the computer system, rather than on an internal drive (or as just suggested, externally, outside the computer system). It is not certain whether storing the bootloader (such as on an SD card) such that it can be locked away securely, is more or less secure than the solution presented by the Heads system. However, it is suspected that the Heads solution is more secure because otherwise the Heads creator probably would not have implemented it.

---

<sup>15</sup> See [here](#) for further information that supports this belief.

## Factory resets

Could it be a good idea to perform [factory resets](#) on [computing devices](#), on a regular basis? If doing such resets gets rid of any [malware](#) that may have managed to get its way on to the devices, perhaps it is a good idea to do such resets perhaps once per week, on a Monday, just before starting the working week.

### According to device type

#### Web client computers

With [web client computers](#) (like the [Chromebook](#)), it is likely that there are very few [backup](#)-and-restore concerns about doing a [factory reset](#), because [files](#) are often mostly stored (and consequently [backed-up](#)) in the [cloud](#).

#### Smartphones

With a [smartphone](#), it might take longer to configure after each [factory reset](#), but still should be relatively straight-forward to do.

#### Conventional laptops

Doing a [factory reset](#) for a conventional [laptop](#) is more of an issue. Intensive use of the internal [drive](#), for any frequent periodic factory resetting, is likely going to [shorten the lifespan of the internal drive](#), significantly, detrimentally, and unacceptably, in the situation where replacing the drive is not an option (perhaps because of financial constraints). Also, the amount of data requiring [backup](#) between such resets, is likely going to be inhibitive. Please note that if you decide to use a [live DVD/CD](#) for the loading of your [operating system](#) and perhaps also for other pieces of [software](#), where the [DVD/CD](#) has been contrived so that it is virtually impossible to change the data on the medium (see the section entitled "[Rewritable media vs optical ROM discs](#)" to find out how to do this), factory resetting with respect to your operating system and perhaps other software, can become entirely unnecessary, and in certain situations, can overcome these issues regarding the undue [deterioration of internal drives](#).

## Some other issues

### Internet connection during reset process

[Internet bandwidth](#) and quota will be needed to cope each time a [device](#) needs to be updated after such [factory resetting](#). A standard [WiFi router](#) should normally be able to be used without security concerns for such updates because man-in-the-middle ([MITM](#)) [attacks](#) normally are not possible due to [HTTPS](#) connections (or other such connections) being used for the updating. However, a [vulnerability](#) does exist if any of the historic [security certificates](#) upon which reliance is made, and that date back to the time of the production of the device, become compromised (which can happen outside of your systems) in the intervening time between the time of the device's production and the time when the [factory resetting](#) is performed. If this vulnerability is considered a significant danger, an owner may deliberately choose not to factory reset their device to avoid the danger—instead, the device's [firmware](#) and [installed operating system](#) can possibly be reinstalled through other means such as those outlined in the section entitled "[Regarding how to obtain software](#)". Any alternative means should be evaluated according to whether the alternative means are practicable and whether they are less dangerous.

### Sufficiency

Is [factory resetting](#) of devices sufficient for removing [malware](#) even when there is no risk of [MITM](#) attacks happening during the factory resetting? Unfortunately, it is not sufficient in all cases. Whenever malware has infected the [BIOS/UEFI firmware](#), there's a chance that factory resetting will not be sufficient to remove the malware<sup>16</sup>.

### Cookies

An irritating thing about a [factory reset](#) can be that it wipes all of your stored [cookies](#). In order to retain your cookies, try using a [backup](#) and restore utility for your cookies<sup>17</sup>.

---

<sup>16</sup> See the section entitled "[Security of BIOS/UEFI firmware](#)" for information on [BIOS/UEFI firmware](#) security.

<sup>17</sup> See [here](#), [here](#), and [here](#) for more information.

## Sandboxing and cloud computing

Connected with the concepts of [physical isolation and locks](#), are the concepts of [sandboxing](#) and [cloud computing](#).

[Qubes OS 4.0.3](#) and [Chromebooks](#) specifically use sandboxing to mitigate risks in [software malware](#). The concept of sandboxing can, to a certain extent, be extended to cloud computing where there is software isolation based on [server-side](#) processing in the delivery of software that on the [client \(user\) side](#), only requires [thin client](#) software (sometimes the software is simply a [web browser](#)). This essentially shifts the problem of [end-user security](#) to one that is largely a [server-side security](#) problem, which is easier to handle in certain respects. Please note though that cloud computing can present new security risks, such as those related to storing your data remotely rather than locally.

You may wish to run your software using cloud computing resources, and interface with the software just using thin clients, as a possible way to improve your security. For example, you could create an [Oracle Cloud Linux](#) compute instance, install Linux software on it, run the software remotely, and then access the software's [GUI](#) using an [x terminal](#) (thin client) run on the client side. An alternative simpler way to 'cloud compute', is simply to use a Chromebook in conjunction with the many [web-based apps](#) available for [ChromeOS](#).

---

## Passwords and digital keys

### Password security

[Password security](#) is a very important issue as nowadays, it is not at all uncommon for users to be maintaining many accounts, some of which may be providing extremely important functionality, and that each require their own security credentials. Additionally, another reason why it is important, is that it is often good practice, and sometimes even [contractually required](#), that users change their [passwords](#) on a regular basis. All these things put together, can appear to constitute something of an ordeal when it is remembered that users somehow have to keep [secret](#) and ordinarily also be able to call to mind, all of their passwords, which could potentially be very many.

In these respects, it seems prudent that an overall password security strategy be developed.

General information from the National Cyber Security Centre ([NCSC](#)) on password security is available [here](#). Information and guidance from the NCSC on what makes a good password is available [here](#).

### Password managers

[Google allows passwords to be stored in your Google account](#). A user can log into their Google account so that the account providing the settings of their [Chrome internet browser](#), is the Google account. The Google account can then store very many different [passwords](#), for different online accounts, that no one actually knows (it can be set so that even the user doesn't know them). These passwords can also be [randomly](#) generated so as to create [strong passwords](#), again, that no human being knows. When the user goes to the login screen for one of their accounts, where the password for the account has been saved in the manner just described, the Chrome browser automatically fills-in the password for the user, without the user having to call to mind the password or even to know it. In this way, a user can have different strong passwords for their different accounts, that can easily be changed, without any human being actually knowing any of the passwords.

There are other [downloadable](#) tools that also have this kind of [password management](#) functionality.

National Cyber Security Centre ([NCSC](#)) information and guidance on [password managers](#) (aka password vaults) is available [here](#) and [here](#).

## Multi-step authentication and multi-factor authentication

(multi-factor authentication is also known as [MFA](#))

The security of [passwords](#) is ordinarily enhanced by using either [multi-step authentication, or multi-factor authentication](#) <sup>(MFA)</sup>. The 'multi' aspect is often two (i.e. two-step authentication). MFA is a means for overcoming mind-reading attacks<sup>18</sup>.

When Google's [two-step authentication](#) is set for a [Google account](#), so that two-step security is used every time the user logs into their [Google account](#), it in a way, to some extent, translates to a two-step level of security for those online accounts of theirs, that exclusively have their passwords stored in the user's Google account using the secret way outlined in the previous subsection. [Google](#) offers a variety of second steps in their two-step security [authentication](#) settings, including [USB security key token](#)<sup>19</sup> and [text message](#)<sup>20</sup>.

The National Cyber Security Centre for the UK ([NCSC](#)) [recommends that users set-up 2FA](#) <sup>(two-factor authentication)</sup> [on all their important accounts](#). More information and guidance from the NCSC on multi-factor authentication can be found [here](#).

### Non-Latin alphabet

[Computer password security](#) can be [enhanced](#) by using an [alphabet](#) not so well known for the password. Non-[latin alphabets](#) will likely fit this description. It's probably best to mix several different alphabets together, for maximum security. This might be easier than initially thought, because within certain familiar domains such as [mathematics](#) and [science](#), [letters from foreign alphabets are often used](#) (such as from the [Greek alphabet](#)). Choosing letters that have uncommon [pronunciations](#) may provide further security. [Unicode non-verbal symbols and word symbols](#) can also perhaps be used, including [emojis](#), to increase security even more.

### Concerning certain password attack vectors

#### Screen privacy

The [Chrome browser software](#) unfortunately (at the moment) displays suggested [strong passwords](#) on the [screen](#), which is open both to [psychic attack](#) and [VDU signal interception attack](#). To overcome these attacks, simply move the Chrome [window](#) so that when Chrome tries to display the [password](#), the displaying occurs outside the boundary of the screen, thereby preventing it from being displayed.

---

<sup>18</sup> This principle is somewhat related to the later "[Protection using password encryption](#)" subsection.

<sup>19</sup> For guidance on the security of USB key-based security tokens, please see the later section entitled '[Tokens for keys](#)'.

<sup>20</sup> Google's account security is also good because it detects whenever a person has logged into your account from an unfamiliar computer (not one of your usual computers); users receive security warnings informing them of such happenings, which can be an easy way to identify whether you are being [hacked](#), and can also ward off [hackers](#).

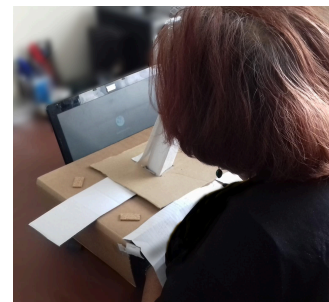
As detailed later in the subsection entitled “[Privacy screens](#)”, [privacy screens](#) can be used as a defence against the [spying](#) of the visual emanations emanating from your screen, which can have the knock-on effect of interfering with [illicit password-capture attempts](#).

### Keyboard privacy

#### *Visual spying of keys pressed*

##### Complete occlusion

A [cardboard](#) “restricted viewing enclosure”, with room for hands, can be constructed to go over keyboards<sup>21</sup>. Once placed over keyboards, the user must look through the two eye holes (like looking through goggles) or some other kind of narrow viewing port, to be able to see the [keyboard](#). User then places hands through hand holes, and types their [password](#). In such ways, casual onlookers as well as cameras ([hidden](#) or otherwise), cannot see what is being typed, because the cardboard construction prevents such seeing, through visual occlusion formed by the cardboard.



*Example cardboard “restricted viewing enclosure” installed on [Chromebook](#)*

---

<sup>21</sup> Using commonly-available [cardboard](#) for such construction could be advantageous.  
See the section entitled “[DIY security principle](#)” for more information on this.

Another way to provide a level of occlusion from spies relying on [spying](#) of the [visible spectrum of light](#), is to move physically, your [computer](#) and/or its parts accordingly. In the case of a [laptop](#), this might involve partially closing your laptop (moving the screen closer to the keyboard based on pivoting around the laptop's hinge[s]) but not so much so as to trigger the [automatic powering off of the laptop](#).



*Partially-closed laptop for keyboard-typing privacy*

#### Distance-dependent occlusion (privacy keyboard screen)

Using the [reflective](#) material commonly found in the [lenses of sunglasses](#), you can create a short table of sorts (perhaps 10cm to 20cm tall) to go over your computer [keyboard](#). Hopefully, when you type with the table installed, you must move your eyes close to the table surface to see which [keys](#) you are pressing. The hope is that from too far away, all you see are [reflections](#), due to the special material being used for the table's surface. By installing such a feature, you will better prevent [cameras](#) from recording your [keystrokes](#). This could be very useful if you travel and spend time using your computer in different hotels where [secret cameras](#) could easily be installed. It is not certain that this will work, but it appears likely that materials with such [transparency](#) and reflective properties really do exist; so far no information on them has been found through [internet searching](#). Perhaps another way to achieve the same effect, is to use [glass](#) or see-through [plastic](#), that has very high [glare](#) properties. It is suspected that old [CDs](#) and [DVDs](#) can possibly be recycled to create such privacy screens. It is also suspected that material that is [polarised](#) and [transparent](#), could quite likely help in creating such privacy screens.

It should be noted that angle-dependent [occlusion](#) (rather than distance-dependent occlusion), as in restricting [viewing angles](#), may also be helpful for establishing keyboard privacy. As such, the later section entitled "[Privacy screens](#)" is relevant to the subject of privacy keyboard screens.

In conclusion however, for simplicity and ease, probably the best way to devise such a screen is simply to use a kind of [pinhole](#) material. For example, [pinhole glasses](#) are used by [optometrists](#) for the [keratoconus](#) condition, to identify vision in spite of the condition. From far away, the glasses just appear as small holes in some plastic. However, from close-by, being worn on an individual, an individual can see through the pinholes. The principle appears to be relatively straight-forward, based on the [geometry](#) of [light rays](#) as they pass from objects ahead of a viewer, towards the viewer. It is strongly suspected, such "pinhole glasses" material can be created by simply poking several holes through a [cardboard](#) sheet (which could provide a "low cost and home made" based advantage<sup>22</sup>).

#### [Using morse code](#)

One of the most straight-forward ways to counter the visual [spying](#) of the pressing of [keys](#), is to resort to entering your [password](#) using [morse code](#) or some similar [single-key](#) password mechanism. A user can easily surreptitiously enter in their password using morse code whilst also having the appearance of writing something completely unrelated to entering a password. The user's hands can additionally be placed so as to [occlude](#) the visual capture of the morse code being entered.

#### [Spying of electronic keyboard signals](#)

Using a [keyscrambler software](#), seems wise and a sensible [countermeasure against the key logging of the electronic signals generated by keyboards](#). As outlined later in the section entitled "[Psychic spying of password](#)", it is also effective against certain [psychic attacks](#). Basically, what you type, say for example, for a [password](#), is not what gets entered as the password—it is [encrypted](#). Without knowing how to encrypt the key presses, it is impossible to know the actual password used, by means of conventional [keyloggers](#).

---

<sup>22</sup> See the section entitled "[DIY security principle](#)" for more information about this kind of advantage.

## General visual spying

### *Hiding materially-written passwords*

Hologram technology can be used for hiding passwords printed on paper, so that they can only be seen from certain viewing angles. Lenticular printing (which appears either to be related to holography or a type of holography) appears potentially capable of providing a cheap, mostly home-made, and DIY<sup>23</sup> way to make holograms or something like holograms, capable of hiding passwords in this manner. Perhaps lenticular-printing security can be achieved simply by using something like corrugated plastic placed over a print-out from a conventional printer. Alternatively, passwords can possibly be hidden by them being printed on paper using ink that has slightly different reflective properties to the rest of the paper. When viewed from a certain angle, the hidden text will possibly be revealed due to the greater reflection of light from the ink, such that from other viewing angles, the reflection is not strong enough to reveal the text. Instead of using reflections, using other optical effects based on the use of transparent materials, perhaps can be employed for roughly the same thing. A reflective effect used in an opposite way, such that higher degrees of reflection effectively blot out the password and lower degrees instead reveal the password, may also be effective. Perhaps an even more interesting effect can be obtained if passwords are only revealed when monitoring all frames in the movement of an object. This could perhaps be done by rotating an object containing the hidden password, where the containment was such that the first frame shows just the first letter, and the last frame, at which time 180° rotation is achieved, shows just the last letter of the password. These kinds of effects appear to be possibly achievable with certain kinds of moving hologram, and also with certain kinds of lenticular printing.

---

<sup>23</sup> See the section entitled "DIY security principle" for information on the potential security advantage of DIY security set-ups.

By using a [UV pen](#), hidden things can be written that are not at all visible to the [naked eye](#). Using an [ultraviolet security lamp](#), ought to reveal such hidden things. But perhaps even more interesting, is that it seems [UV protection glasses](#) may also be able to see such hidden things. This effect appears to be [widely accepted](#) as a means for card players to [cheat](#) at [card games](#) like [poker](#). To obtain such glasses, it might be best to order a pair of ordinary [looking glasses](#) (for the [sight impaired](#)), with the option chosen for them to undergo special treatment for [protection against harmful UV rays](#); if you already use looking glasses, you may find that any extra expense incurred, ends up being inconsequential. Such [invisible ink](#) may be particularly handy for the paper-based keyboard [scrambler](#) mentioned in the “[Protection using password encryption](#)” section that shortly follows.

There are also ‘classic’ old-fashioned [steganographic](#) ways to hide passwords, such as in a completely unrelated document by very weakly [pencil](#)-underlining the letters making up your password.

#### *Overcoming [vulnerabilities](#) in visual encodings*

The prevalence of [camera](#) technology likely means the very many types of visual encoding for [passwords](#), can simply be recorded using camera technology with ease. Later [computer analysis of captured images](#) can assist in [cracking passwords](#) also perhaps with ease.

[Encoding passwords as flavours](#) on a strip of paper, might overcome these [vulnerabilities](#). If the paper can be designed so that it can only be 'tasted' once, then you have the advantage of being able to detect whether the password has already been 'read'. Technology to taste small 'dots' of flavour on pieces of paper, is likely not prevalent and is likely very expensive if it is at all possible (it might be possible with the aid of machines like [GC-MS machines](#), but such machines are normally very expensive and require highly specialised skills that only a small minority possess). Once a flavour-encoded password has been 'read' (tasted), it can be destroyed if it is still able to be 'read'.

Another potential way of encoding passwords to overcome [sight](#)-related vulnerabilities, is to use non-visible [braille](#). The non-visibility aspect of the braille may be hidden by means of [visual noise](#), or simply visual absence (of the specific markings used for the braille).

## Psychic spying of password

### *Protection using password management functionality*

The [password management](#) features of [Google accounts](#) also constitute one way of overcoming [psychic password spying](#) (basically, mind reading). Because only the 'computer knows' the individual [passwords](#), [psychic attacks](#) simply don't work here so long as the [security for engaging the password manager is not compromised](#).

### *Protection by thinking little*

[Passwords](#) extremely well [memorised](#), such that entering doesn't involve the [conscious calling to mind](#) of the password, is perhaps another way to prevent [psychic attacks](#) aimed at illicitly learning of passwords. If the password is called to mind merely as finger presses of keys, where exactly which key is pressed is hardly thought about by the key presser (where reliance is made on [rote memory](#), rather than on conscious thoughts), probably it will be hard for those relying on [psychic](#) powers, to figure out what the password is.

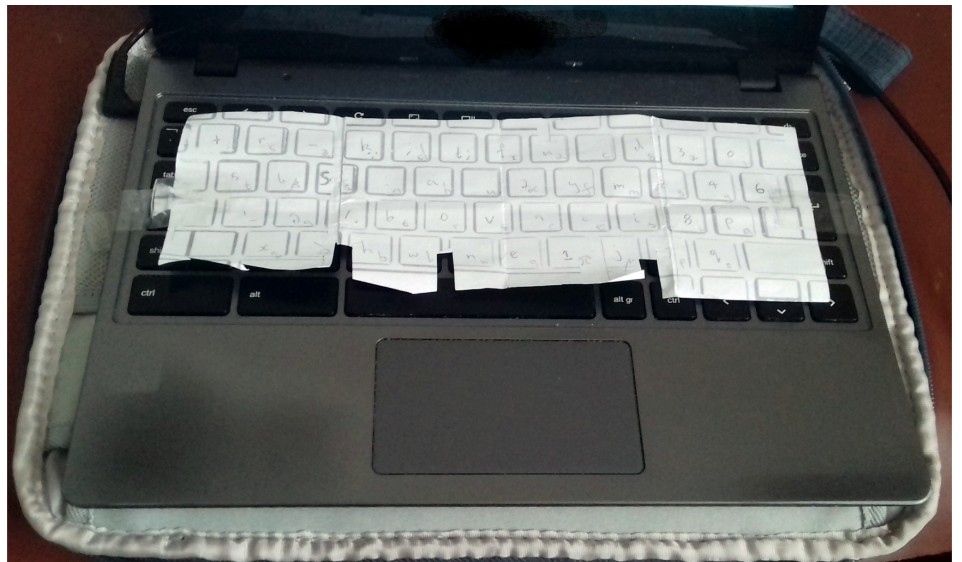
### *Protection using password encryption*

#### With technology

Another mechanism to protect against [psychic attacks](#), is to have some piece of [software](#) on your local computer, that translates the [password](#) in your mind, into another one, where the other one is the one used to log-in (say to your [Google account](#)). Keyboard [scrambling](#) software is one such kind of software. Psychic attacks can get hold of the password in your [mind](#), but because the real password for logging-in, is one derived from the software and your memory combined, the 'psychic attacker' can't get hold of the real password without also having the software. The software's [algorithm](#) could perhaps be simply the [concatenation](#) of some never-known [private key](#) to the password called to mind by the user. In the case of any entity trying to force the account password out of you, [destroying all the media on which the private key is stored, would effectively render recovery of the password ordinarily impossible](#). This kind of security feels like it might be aptly labelled as a kind of [two-factor security system](#).

### Without technology

Using a paper-based keyboard [scrambler](#) is perhaps a good idea to mitigate against [psychic attempts to obtain your password](#). What you do, is you simply create a paper-based overlay for your [keyboard](#), where the key [characters and symbols](#) are scrambled. You type your [password](#) as you remember it, into the scrambled keyboard. You then pack away your keyboard scrambler, perhaps even [locking it up](#) till the next time you need to use it. In such fashion, whilst a [psychic attack](#) might obtain the password from your [mind](#), they will not actually know the password, because that password in effect has been [encrypted](#) through the paper-based scrambler. In fact, if you don't [memorise](#) the paper-based scrambler, then not even you know the true password, which seems really good. It's important to block the [photographing](#) of the paper keyboard scrambler, which can happen by means of [hidden cameras](#). In this regard, folding it up when you don't need to see it, seems like a worthwhile precaution. A paper-based keyboard scrambler is particularly useful because of how much it costs to create. It's entirely feasible and economical to use a new [cipher](#) encoded in the scrambler, each week, by simply discarding the last paper-based keyboard scrambler, and making a fresh one. Also, the fact that it is not based on [computer hardware](#) or [software](#), is another attractive attribute of this [security measure](#).



*Example paper-based scrambler installed on a [Chromebook](#)*

Based on password reuse

For National Cyber Security Centre ([NCSC](#)) information on specific concrete passwords not to use (password blacklisting), see [here](#).

For NCSC security guidance and information in relation to a user reusing one of their previously used passwords, see [here](#).

For NCSC information on the password attack known as credential stuffing, that exploits the habit of users reusing passwords, see [here](#).

## Digital cryptography: security [certificates](#), [keys](#) & [tokens](#)

### Disabling [TLS](#) security [certificates](#)

Why not turn off certain security [certificates](#)? This seems like a prudent measure. Not all certificates are to be [trusted](#) with the same level of trust. Also, it's likely that not all certificates are actually generally needed; in fact, perhaps only a small number of security certificates are in reality needed.

How about disabling all [TLS](#) security certificates by default, and then selectively turning them on as and when they are required? Can even turn them off once they have been used. This should help to detect attempts at [fraud](#) and/or [spying](#) over the [HTTPS](#) protocol (and other like systems).

### Making sure [certificates](#) are genuine

Ensuring security [certificates](#) are not compromised, seems very important. To this end, multiple checks on them, perhaps in an automated fashion, seems like a good idea. Simply doing spot-checking to make sure that [TLS](#) security certificates on a particular device have the correct [keys](#), by comparing them with the keys from another computer or source, may be enough.

A user could export security certificates to an [SD card](#) and then when using a different system (such as a different [WiFi](#) network or different [telecom provider](#)), the user could then obtain differently sourced certificates, and check they are the same to the ones the user already has. Could be particularly applicable when in another country, as well as when visiting friends, etc. For example, in [China](#), the government may [tamper](#) with security certificates, but users can then visit other countries, re-obtain the certificates, and figure out that the government is tampering with their security certificates.

### [Key servers](#)

The notion of [key servers](#) seems very much applicable here—they can probably be used for verifying [TLS](#) security [certificates](#), and generally speaking, they probably ought to be used for this and other types of [digital key](#). Use of software such as [GPG](#) can be used perhaps to automate the task of key-server-based verification of keys. GPG apparently is available in the [Heads firmware boot](#) verification system (Heads was mentioned earlier in the section entitled "[Custom BIOS/UEFI and which one to use](#)")—running GPG as [hardware](#) firmware may be best to minimise the risk of the [software](#) being compromised.

### Cross authentication

Is there a way to get [certification authorities](#) to cross-authenticate certificates from other such authorities? Does it happen automatically? If not, it sounds like a good idea to have such a mechanism in place. For example, you may be able to discover that one certificate has been compromised on your computer, through the signed message of another trusted certification authority revealing what the compromised certificate's [key](#) should have been. Relevant information on this subject might be found [here](#).

### Non-compromised communication of public keys

[Communicating public authentication keys](#) simply through [website publishing](#) might be considered as being [not secure enough](#). The additional use of a [key server](#), or maybe multiple key servers, may be useful for increasing [trust](#) in keys (which is touched upon above in the "[Key servers](#)" subsection). The [novel Bitcoin-related method for increasing trust in public keys](#), described in the [Appendix](#), may also be useful for improving user trust in published keys.

However, what other ways can be used for communication of [public keys](#)?

Sending to trusted recipient such that the recipient can hand it over without encountering [MITM](#) vulnerabilities, to the [end-user](#)

One possible solution is to get the organisation represented by the [website](#), to send the [key](#) to a trusted recipient who can then hand it over to the [end-user](#) where in the handover there is no or virtually no possibility of man-in-the-middle ([MITM](#)) attacks. It might be ideal that the trusted recipient [prints out the key, rather than for them to hand it over to the user as a computer file](#), in order to mitigate against security risks associated with computing technology. The user can physically travel to the recipient, rather than have the key [posted](#) or sent on to the end-user (thereby reducing [transit risks associated with MITM attacks](#)). A perhaps novel solution, that might be ideal, is for the trusted recipient to be an [online printing](#) business, such as [Tesco's](#), [Kodak's](#), or [Boots's](#) photo printing. Several trusted recipients can be used to detect better, compromises in such [communications](#).

### Publishing public keys in a "gazette"

There is still the chance that an [end-user's](#) request for such transmission to a trusted recipient is compromised through [MITM](#) attacks. To mitigate against this, instead of organisations doing such sending of [keys](#) on a per-end-user-request basis, the organisations can periodically do such key sending to key locations in the geographic areas of their markets. Doing such sending, is similar to simply periodically [publishing](#) the [public keys](#) in a well-known [gazette](#).

## Piggy-backing over [bank](#) transactions and systems

### *Using [bank](#) references*

Another method of [communication](#), that might be quite secure, is to [piggy-back](#) over the [financial system's banking system](#). For example, the organisation can do a [bank](#) transfer for a nominal, trivial, and negligible amount to the [end-user](#), with the bank reference being a [public key](#) to be sent. The end user can then visit a local [bank branch](#), where the security is likely to be very high, and find out the bank reference used, and consequently the public key. Not only will the security at the branch be high, the security of the overall banking system is likely to be very high, thereby perhaps ensuring quite secure communication of the public key. To ensure that the transaction is genuinely from the organisation, perhaps your local branch can offer a service where they can give you useful [company](#) details of the account that paid you (such as possibly the [company registration number, address of the registered office](#), etc.) You can then possibly authenticate that the source of the [financial transaction](#) was indeed the organisation. By ramping up the value of the transaction, perhaps further trust can be attained through the relatively high amount of money spent in the transaction.

### *Using different [monetary](#) amounts*

If it is believed that the [bank](#) transaction reference may be compromised, then reliance can instead possibly be made on the monetary amounts of transactions. For example, to send the [key](#) 1735 8513, a sender can make a series of 8 [financial transactions](#): the first being for 1 pence, the second for 7 pence, the third for 3 pence, and so on. Because banks invest heavily in the [financial integrity](#) of transactions, this could be quite a secure way to communicate such information.

### *Using a [weak currency](#)*

Interestingly, using a [weak currency](#) could be better for such transactions, to bring down overall costs—perhaps a suitable [cryptocurrency](#) demonimation might be available in order to bring down the overall costs.

### Google Authenticator “key and time”-based app for security

A security method to mitigate against a certain type of [evil-maid attack](#), involving the [Google Authenticator](#) time-based [app](#) system to ensure a computer has not been secretly replaced with another potentially compromised one, is described by [Trammell Hudson](#) in his [33c3](#) talk<sup>24</sup> (it was previously described by [Matthew Garrett](#)). It is as follows. The [key](#) stored in the Trusted Platform Module ([TPM](#)) can be a private key only shared with [Google](#). Then once you log into your system, the TPM essentially 'signs' the current time (rounded, perhaps to the nearest five minutes) with the key. Google then does the same through the 'Google Authenticator' system at their remote [servers](#). In both instances of signing, the key is already with the respective party (so no transmission takes place, at that time, of the private key). The [message digest](#) created using the TPM is displayed on the computer screen; the digest created by Google, is displayed on the user's [smartphone](#) through the 'Google Authenticator' app. The user compares to make sure the message digests are the same. If they are the same, the user can be sure that an evil-maid attack hasn't occurred where the entire computer was secretly and deceptively replaced with another.

### Tokens for keys

It looks like [USB-Neo tokens](#) and [-Neo-N](#) tokens, in the [Yubikey](#) product family (produced by [Yubico](#)) can have their keys changed from a computer (like the [Nitrokey](#) product) {see [here](#) for info on this}, which meets a requirement of having keys in such tokens different to those set when the tokens are sold off-the-shelf. Additionally and fortuitously, because the Yubico brand appears to be more popular than the Nitrokey brand, it appears that there's an increased chance that it will be found on the shelves of popular computer stores (such as [PC World](#), etc.) {could be handy if employing principle detailed later on in the section entitled “[User randomly selecting unit from off physical shelves](#)”}. Also, they seem to be considered as being secure, with [Google having mandated that all their employees use them for security](#). So it seems likely that generally, instead of opting for the less available Nitrokey product, one should instead get a Yubikey product that has the facility to have new keys assigned to it by the transference of them from a standard [PC](#). If, for some user, it becomes or is such that the availability of Nitrokey products is the same or better than that of the Yubikey products, then it may well be better for that user, to opt for Nitrokey products instead, since in other more conventional respects, Nitrokey products do appear to be more secure.

### External links for further information on [security certificates](#)

For National Cyber Security Centre ([NCSC](#)) guidance describing how [certificates](#) should be initially provisioned, and how supporting infrastructure should be securely operated, see [here](#).

---

<sup>24</sup> A video of the [33c3](#) talk is hosted [here](#).

## Backing-up security keys and passwords

### Shamir's Secret Sharing

The [developer](#) of the [Heads firmware boot](#) verification system (Heads was mentioned earlier in the section entitled "[Custom BIOS/UEFI and which one to use](#)") has provided a potentially useful way to backup [keys](#) and [passwords](#). He suggests splitting a key (or password) into say five different sections, and backing-up each section in a different way (perhaps by sending each section to a different friend). If recovery of the key is then needed, a threshold number of the sections can then be retrieved, and put together to recover the full key (perhaps in an automated way using [specialised software](#).) The idea of splitting the key into several sections is apparently the basis of a [cryptography algorithm](#) known as [Shamir's Secret Sharing](#). An interesting thought, is that [Bitcoin](#) wealth could potentially be [backed-up](#) this way, like creating a [treasure map](#)—perhaps it doesn't matter if it takes a year to trace the treasure using the treasure map, because the wealth is such that it doesn't need to be obtained straight away.

---

## Wireless Communications

If [internet connection](#) is [hacked](#), computers will become [vulnerable](#). Although maintaining [security](#) on the user's machine will probably mean that security overall will be sufficiently maintained, it is still worthwhile safeguarding against such an event.

### Wired vs. wireless

Whenever there is a choice between [wired](#) or [wireless communication](#) (such as in the case of wired [tethering for internet connections](#)), wired communication is likely going to be more secure; it appears that most [wireless routers](#) usefully also permit wired [ethernet](#) connections.

### Shared WiFi

It's best not to use a shared [WiFi router](#) at all, if it can be helped; or if it is used, only to use it with a highly secure [client](#)-machine set-up. When a [router](#) is shared, it can be difficult to convince others about the need to perform certain security tasks, such as the changing of the [WiFi password](#) on a regular basis. [Essex police](#)'s [cyber-security](#) advice as of March 2020, was not to use free [WiFi](#) for anything you wouldn't want a stranger to see.

### Keep communication systems turned off

Keeping wireless technologies such as [WiFi](#), [Bluetooth](#), [NFC](#), and [infrared](#) ports, disabled when not in use, is a good precautionary [security measure](#).

---

## Digital storage

### General security risks in digital storage

[USB devices](#), [SD cards](#), and [drives](#) (whether internal or external, whether a [HDD](#), [SSD](#), [optical disc drive](#), or otherwise), including [USB memory sticks](#), can likely easily be modified so as to transmit data to [snooping devices](#)—a [security risk](#). Device [ROM malware](#) (using [firmware](#) or otherwise) likely always constitutes an [attack vector](#) for such devices (that appear often to have [embedded microcontrollers](#) that run such ROMs). The same applies with respect to [hardware](#) device [tampering](#). However, SD cards are perhaps unlikely to undergo hardware tampering because of their small size (especially with respect to [micro SD cards](#)) and because of how they are constructed.

The custom [BIOS/UEFI](#) called [Coreboot](#) usefully has the option of being able to disable [option ROMs](#) when they are present in USB devices<sup>25</sup>. If you use [Linux](#) in conjunction with such disabling, you can instead rely on the [drivers](#) supplied in Linux to interface with those USB devices<sup>26</sup>—this approach objectively reduces the [attack surface](#) associated with those USB devices (because Linux drivers are considered to be more secure than [manufacturer](#) provided option ROMs that provide the same functionality) and is recommended for improved security.

Further information about such [security weaknesses](#) of [USB](#) and like devices, is available [here](#) on the [Qubes](#) website.

### USB devices vs. SD cards

At first, it was initially thought that [SD cards](#) had no [firmware](#), such that they couldn't be infected with firmware [malware](#). However, as pointed out on the webpage hosted [here](#) (by [Andrew "bunnie" Huang](#)), SD cards do in fact have firmware as well as [embedded microcontrollers](#) that run the firmware. Therefore, firmware malware can be present on SD cards similar to how they can be present on certain [USB devices](#). [The webpage](#) elaborates on these specific [security risks](#).

---

<sup>25</sup> [USB](#) devices that have [embedded microcontrollers](#) for running [firmware](#) on the devices, are probably unlikely to have and use [option ROMs](#) (instead, for such devices, another kind of [ROM](#) is more likely to be contained in the USB device).

<sup>26</sup> It appears that this likely does not apply to [USB memory sticks](#) and [SD cards](#) since they don't appear to use [option ROMs](#) (instead, they appear to use another kind of [ROM](#)).

SD card firmware is mostly a [blackbox](#) according to team Kosagi (in the [video](#) linked-to on [the webpage](#)), whereas [USB device](#) firmware specifications are generally more open and publicly available; this leads one to believe that from a purely [programming](#) perspective, it is easier to create [malware](#) for USB firmware (than it is for SD card firmware). Also, the team's presentation in the [video](#), inclines one to think that [hardware](#) interfacing for the purpose of reprogramming firmware, is much more difficult for SD cards than it is for USB devices. According to the team, it also appears it is harder to [tamper](#) with SD cards by sneakily adding or modifying [hardware](#) elements in them, than it is for USB devices (see same [video](#)); their quite small size especially compared with USB devices, also supports such a conclusion. Because of these things, SD cards may generally have smaller [attack surfaces](#) in relation to both firmware [reprogramming](#), and [hardware](#) tampering. Therefore, it may be best to assume that for higher security, if the alternative is USB storage, the use of SD cards is to be preferred for [peripheral](#)-based storage and retrieval whenever no extra special [security measures](#) are in place.

However, because the USB device firmware specifications are more open and publicly available, and because it appears to be easier to interface with a USB device for the purposes of checking and/or [reinstalling](#) firmware, it could in fact be easier to detect malware in USB devices than it is to do the same in SD cards. Reinstalling USB device firmware is often possible but the same for SD cards is generally almost impossible. So if you have a special process in place for ensuring the integrity of USB device firmware in the ways just touched upon, it may actually be more secure to use USB devices than to use SD cards.

### Flash memory: [NOR flash](#) vs [NAND flash](#)

There appears to be some evidence that using [NOR flash memory](#) for [storage](#) (rather than the more common [NAND flash](#)) on [USB](#) and other devices, might offer a security advantage, most likely due to issues related to NOR flash apparently being better able to store and retain values in each of its [cells](#). In contrast, NAND flash employs [error correction algorithms](#) because values are not so well stored in NAND [flash cells](#) (more of a [probabilistic](#) approach is taken). However, NOR flash appears to be mostly earmarked for things like [BIOS ROM](#) rather than general storage (and also rather than [mass storage](#)).

### [NAND flash memory](#) vs [magnetic storage](#)

Because in comparison to [magnetic storage](#):

- [error correction](#) appears to be required more for [NAND flash](#),
- NAND flash appears generally to have more [bad blocks](#),

 because adversaries can exploit these things,

it would seem that magnetic storage (such as the use of a [HDD](#)) is more secure than NAND flash (NAND flash is commonly used in [memory sticks](#), [SD cards](#), and [SSDs](#)).

### Magnetic storage: [tapes](#) vs. [discs](#)

Because it seems that objectively, random-access memory ([RAM](#)) has a greater [attack surface](#) in comparison to sequential access memory ([SAM](#)), it would seem that [magnetic tapes](#) are more secure than [magnetic discs](#) (magnetic discs are used in [HDDs](#)).

## Rewritable media vs optical ROM discs

"Write once" [optical media](#) is more secure than [rewritable media](#) because it narrows the attack window (and so also reduces [attack surface](#)) by generally significantly shrinking the window of time in which [malware](#) can be written to the [media](#). It should be noted that if malware has already infected the [firmware](#) of an [optical disc writer](#), even though you configure [disc writing](#) to be 'write once', because of the [ROM](#) malware, the writer may instead set the disc to be 'write many'. From this perspective, it is better only to use [optical discs](#) that physically cannot be [rewritten](#), to write such media so that excess space is "blanked out", and to [inspect visually](#), discs written, just after writing, to make sure that the excess space has indeed been blanked out; all of these things are important safeguards against [attack vectors](#) that specifically rely upon being able to make further writes to the disc.

Optical ROM discs are also potentially more helpful compared with rewritable media, in relation to post security checking. Malware in the ROMs of [memory sticks](#) and [SD cards](#), can mostly use the storage capacities of such devices unencumbered whilst also being able to eliminate all traces of malware from the devices, any time the ROM is run. However, the window of time in which such can happen for [optical ROM discs](#), as mentioned in the previous paragraph, is mostly narrowed. By using write-once media, and taking the precautionary measures outlined in the last paragraph, the optical disc contents are frozen. If the frozen contents contain evidence of malware, that evidence cannot be clandestinely destroyed through your computer system once the disc's contents are frozen. This means post analysis, even post analysis one year later, can potentially detect any malware that was on the disc, possibly leading to the identification of points of weakness in your security that are connected to the use of that particular optical disc. The same is not necessarily true with rewritable media because no such freezing takes place.

## SD cards and USB memory sticks vs. larger devices

[Firmware](#) integrity for larger devices is probably easier to establish because:

- larger device products (such as [HDDs](#) and [DVD drives](#)) tend to be kept for longer with fewer units tending to be used per user, when compared with [SD cards](#) and [memory sticks](#), which means that not so many firmware checks overall need to be made.
- their relatively high price inclines one to believe that it is easier to ensure firmware integrity (because of increased [support](#), better written firmware code, etc.)
- reinstalling and checking the firmware is likely much easier than it is with SD cards that are constrained due to their small [form factors](#), and perhaps also when compared with [memory sticks](#).

## Drives able to eject hardware-less media vs. other media

Drives capable of ejecting hardware-less media (such as optical disc drives, tape drives and floppy disk drives) may have special advantages in respect to hardware and firmware tampering, due to users only needing to “keep their eyes” on one unit regardless of how many hardware-less media (such as tapes and discs) are used. This is not true for memory sticks and SD cards because these media are essentially hardware incapable of ejecting hardware-less media, in fact the concept of ejecting media from them doesn’t apply. It is also not true for hard disk drives since the discs in such drives cannot be swapped out for other discs.

## More about SD cards

Team Kosagi in the video mentioned earlier, specifically names the manufacturer Sandisk as possibly creating more secure SD cards when compared to the SD cards of other brands. Their reasoning behind this is that extra security is achieved through Sandisk’s creation of SD cards having firmware that is not so reprogrammable (or even reprogrammable at all), due to Sandisk having higher involvement in the production process. Incidentally, SD cards can often be used for storing downloads made through mobile devices.

## How to obtain computer media devices

It seems like a good idea always to use media devices that were bought in such ways that no significant risks of MITM (man-in-the-middle) compromises, were encountered in the acquisition of the items, and that were also bought from trusted shop sources. It might be preferable to buy such things from large shops, where you can choose any one device at random from a large selection of identical products from off the physical shelves (this is the security principle outlined later on in the section entitled “User randomly selecting unit from off physical shelves”).

## Secure data sanitisation

NCSC (National Cyber Security Centre) information on secure data sanitisation (rendering data unrecoverable) of USB storage devices and SD cards, as well as of other storage media and drives, is available here. Note that conventional data erasure may not work with storage devices (such as SD cards, USB memory sticks, external hard drives, etc.) because of the possible presence of malware in the microcontroller firmware of such devices, especially when the related microcontroller is also embedded in the device. In light of such, it may be necessary to destroy physically such devices to ensure better data sanitisation<sup>27</sup>.

---

<sup>27</sup> Andrew “bunnie” Huang recommends such destruction for those “... in high-risk, high-sensitivity situations ...”.

## Some measures that are primarily physical

### Physical isolation and locks

Physical isolation and locks are also other important elements in maintaining computer security. For example, locking-up a laptop when it is not being used, might be a good idea. If a laptop is left unattended for a long time, someone could perform major tampering to it, in such ways that there is no trace evidence of the tampering. Full-system encryption (dealt with in the earlier section entitled "Full system encryption, full disk encryption (FDE)") can partially defend against this.

#### Physical measures for securing bootloader when using full-system encryption

Having the bootloader for such full-system encryption separately locked away on a USB memory stick rather than stored within the laptop as is usually done, may be a good idea from a security point of view (as documented in the earlier section entitled "Bootloader for FDE"). If the bootloader is instead kept on an internal drive (as is normally done), it's relatively straightforward to tamper with the bootloader when there is physical access to the laptop (by use of so-called bootkits for example). If you are storing the bootloader in the laptop, and you choose to use the Heads BIOS/UEFI firmware system<sup>28</sup>, you will have improved security because Heads stores the bootloader in motherboard ROM rather than on one of the internal drives. If in conjunction with Heads, you make certain judicious use of epoxy resin on the motherboard, it appears likely that tampering with the bootloader can be made extraordinarily unlikely, ensuring such good security in conjunction with full disk encryption, that it may even be unnecessary to lock away your laptop.

#### Padlockable laptop bag

If you have a laptop bag, have a look to see whether you can padlock it, perhaps through zipper sliders on any of the bag's zips. Also consider whether you can padlock all of your computing devices within that bag, for safe-keeping. You may, for example, be able to padlock your laptop, Chromebook, and mobile phone, all within that bag. When you take your devices away from your usual work location (such as when travelling with them), you may wish to take the bag with you, and have your devices padlocked in that bag whenever they are not being used.

#### Metal boxes

If you are locking devices away in metal boxes, you may wish first of all to place them in cushioned bags (such as perhaps a laptop bag), in order to prevent damage from knocks and bumps when the devices are in the metal boxes.

---

<sup>28</sup> Mentioned earlier in the section entitled "Custom BIOS/UEFI and which one to use".

Metal boxes can probably be securely obtained, by buying them over the [internet](#) and having them delivered to you, because there are unlikely to be [security concerns](#) if they are intercepted—they can be manually inspected when they arrive to check for [tampering](#). But it should also be borne in mind that, due to great advancements in [espionage technology](#), detection of embedded espionage technology might not be so straight-forward as simply [visually inspecting](#) any such boxes. From this point of view, it may well be worthwhile purchasing such a box using the principle outlined later on in the section entitled “[User randomly selecting unit from off physical shelves](#)” to minimise the risk of espionage technology being embedded in the metal of the box. In contrast, the [padlocks](#) used for any such metal boxes likely should, without question, be bought using this principle in order to minimise MITM attacks ([man-in-the-middle attacks](#)) that are focused on the padlock element of the set-up. It probably should be ensured that the [security ratings](#) of the selected padlocks are all high.

#### Combination lock briefcase

Computing devices may also fit into a [combination lock briefcase](#) that you already own. You might be able to purchase such briefcases cheaply from a local [second-hand shop](#) however, be aware that second-hand combination locks might pose unacceptable [security risks](#) for your particular [threat model](#).

Physically removing [storage component\(s\)](#) from the rest of the computer system, and then securely storing those components separately

Such securing may be good seeing as ‘[data at rest](#)’ on such [storage components](#) (such as on [SD cards](#), internal [HDDs](#), etc.) are mostly more secure than when in their associated computer systems, whenever reasonable [physical security](#) measures are also taken. The storage components can then be installed in fresh brand new non-compromised computer systems, to retrieve securely the associated data authentically. Also, such storage components tend to be smaller than computer systems, which provides for better secret concealment and storage. However, it should also be noted that because of their small size, they can also be secretly stolen more easily.

On some [laptops](#), removing the internal system disk is easy. If you have such a laptop, you can perhaps simply remove the system disk at the end of each working day, and lock it up. This may not be feasible for laptops not having this feature (perhaps most or all light-weight computers, such as certain [Chromebooks](#), fall into this latter category).

## Physically securing keys

It seems likely that someone can quite easily create [clones of most physical keys](#) by:

1. taking a short (maybe 10 seconds' long) [mobile-phone video](#) of such a [physical key](#),
2. sending the [video](#) for [computer analysis](#), and then
3. creating a [cloned key](#) based on that analysis.

If this is true, then an adversary (or maybe perhaps more specifically an "[evil maid](#)"), just needs access to the physical key and a [camera on their smartphone](#) (most people have smartphone cameras) for maybe just 30 to 60 seconds for the creation of the video that is then sent over the [internet](#) perhaps half-way across the world, for computer analysis to create a cloned key.

It may therefore be best to use [tamper-evident systems](#) (such as those detailed shortly in the next subsections<sup>29</sup>) to protect physical keys. However, it should be borne in mind that if the tamper-evident system employed relies upon padding consisting of something like [rice grains](#), then an adversary may still be able to clone the key by simply using [x-ray photography](#). With this in mind, using [shredded CDs and DVDs](#) for padding might be a good idea, as the metal in them will probably interfere with many [photography](#) methods that rely upon [electric fields](#), and/or [magnetic fields](#) (covers [electromagnetic radiation](#)).

## Privacy screens

A good idea seems to be to get [privacy screen filters](#) for your different devices. They work by inducing some level of privacy by lowering the maximum [viewing angles](#) of the [screens](#) on which they are installed.

## Specifically for goods in physical transit

In order to ensure goods aren't tampered with in transit, it can be requested of the sender that they appropriately use [tamper evident mechanisms](#) such as certain kinds of [security tape](#) and certain kinds of [security bag](#). These appear to be cost-effective ways to help ensure goods aren't tampered with when being sent by [post](#). Unfortunately, from investigations, the current tamper evident solutions that are commonly available appear to be mostly quite poor. A novel solution might be to [sign digitally](#) a customer's address, and then print such signature on the tamper-evident mechanism in such fashion that the mechanism is costly to duplicate (perhaps by use of some kind of supplier '[seal](#)', maybe a [holographic one](#), also printed on the mechanism). However, at present, no such product or other good enough product appears to be generally affordable.

---

<sup>29</sup> In the subsections "[Specifically for goods in physical transit](#)" and "[Exploiting unrepeatable patterns for tamper evidence](#)".

## Exploiting unrepeatable patterns for tamper evidence

### Applying glitter nail varnish to computer screws

[Trammell Hudson](#) has suggested a [low-cost and effective tamper-evident solution](#) that uses the painting of [computer-case screw](#) heads with [glitter nail polish](#) to produce practically [unrepeatable patterns](#) that are destroyed through unscrewing. By [photographing](#) the patterns, and making sure patterns haven't changed, one can be reasonably sure that no one has unscrewed the painted screws. Information on this solution, which seems to have been accepted by the security community, and how it can be slightly extended for other things, can be found [here](#).



Collage illustrating the method of applying [glitter nail varnish](#) to computer screws on a [Chromebook](#)

## Tamper-evident security-system ideas

In parallel with [Trammell](#)'s solution which is at least five years old, [Mark Fernandes](#) has recently devised similar schemes that employ the same [unrepeatable-pattern principle](#). Those schemes, possibly with improvements contributed by others, are detailed in the following subsections.

### Main idea

Note that creating your own home-made set-up out of commonly available materials (like [cardboard](#)) will likely ensure greater security<sup>30</sup>.

#### 1. Submerge a computing device in:

- ⊙ [polystyrene pieces](#) (such as those used for [cushioning](#) parcelled items [aka [foam peanuts](#)]),
- ⊙ [rice](#),
- ⊙ [transparent beads](#),
- ⊙ [imitation diamonds](#),
- ⊙ [shredded](#):
  - ⊙ paper,
  - ⊙ [reflective foil](#),
  - ⊙ [holographic](#) material,
  - ⊙ redundant [CDs](#) and [DVDs](#),  
(paper shredders are commonly capable of shredding CDs and DVDs)  
(such shreds might have reflective, refractive, holographic, and transparent properties)
- ⊙ transparent [cling film](#), OR
- ⊙ other transparent [plastic](#)

⊠ ⊠

- ⊙ a mix of any of these

encased in a transparent plastic:

- box, OR
- bag/pouch  
(a bag/pouch is good because the [shape flexibility](#) of the bag/pouch provides stronger [tamper evidence](#)).

(depending on means and security level desired)

Due to the [optical effects](#) of [refraction](#) and/or reflection,  
increased security can be attained by using materials with  
high amounts of transparency and reflectivity.

But when using reflective properties,  
care needs to be taken to make sure  
reflections of things outside the security zone do not occur in the [photography](#)  
as that might make the [authentication](#) of [photographs](#) not possible.

<sup>30</sup> See the section entitled "[DIY security principle](#)" for more about this.

Mixing different materials together, may provide the optimal solution.

Highest security is likely

plastic bag/pouch containing material with

◦ reflective, ◦ [refractive](#), ◦ [holographic](#), and ◦ transparent

properties,

**A N D**

where those properties are varied amongst the pieces.

2.

⦿ *If using a box,*

after the photographing you can gently move it to the 'at rest' security location being careful not to disturb the positions of the pieces in the box.

⦿ If

⦿ you're instead using a bag/pouch, **O R**

⦿ want the box to be at the 'at rest' security location when photographing,

now put the container in the 'at rest' security location.

⦿ The 'at rest' security location should be a place where nothing  
(such as a person, animal, or insect) is likely to disturb it physically

(can be [locked](#) in a [safe](#) to ensure this).

3. Take at least two photographs, each from quite different angles, with at least one from an [aerial view](#), of the plastic container with its contents visible.

A [tripod](#) ([even used with a mobile-phone camera](#)) can be used, in the scenario that a static [camera](#) position-angle combination is required for the better detection of discrepancies between two security photos.

4. Securely store the photographs so that they undergo no [tampering](#)  
(you may wish to print them out and [sellotape](#) them to your body if you desire high security).

5. After some period of time, return to the container.

6. Check that the photographs match the current state of the container.

If the pieces appear to have moved,

- ☐ then it's a good indication that it has undergone some disturbance where possible tampering may have occurred.

*On the other hand, if all the photographs match,*

- ☐ then it's a very good indication that the computing device has not been accessed in the intervening time.

[Software](#) (such as an [app](#)) may be used to automate security photograph matching however, it should be noted that simply switching back and forth between two photos on a [screen](#), can be sufficient for a human being to detect visually, visual changes, due to the [high visual cognitive power of human beings](#)—it's basically a game of '[spot the difference](#)'.



Video detailing [tamper-evident](#) [unrepeatable-pattern](#) mechanism using [rice](#)



*Illustration demonstrating the effectiveness of using [rice](#) for [tamper-evidence](#)*

Speculating stronger security again with [unrepeatable-pattern principle](#)

A similar idea with potentially even more security may exist where a computing device is placed in a [water-proof](#) container, and then submerged in a bathtub of water that has coloured oil on the water's surface such that there is a [marbling](#) effect on the water's surface—admittedly, it's not clear whether this other idea would actually work in practice.

Similar idea for other circumstances (such as for metal boxes)

Another similar system relies on the likely fact that certain [natural hand-made paper](#):

- can be easily [photographed](#),
- is hard to duplicate as a deceptive [fake](#) in respect of the precise natural colour variations for a particular sheet of paper (they're like [fingerprints](#)), and
- is not capable of having significant tears in it hidden from those wanting to detect any tears in it.

Where such properties are also found in other materials<sup>31</sup>, such other materials probably can also be used instead.

<sup>31</sup> Such as perhaps in [recycled paper](#), [newspaper](#), [tie-dyed material](#), [ink-marbled paper](#), [tea-bag marbled \[stained\] paper](#), or even maybe just straight-forward [printer paper](#).

Any one such paper, can be 'read' by taking a [photograph](#) of it. The colour variations can be measured and in sequence encoded as a long number, which can also be interpreted as being a key (like an [encryption key](#)). With such a paper, it is practically impossible to create another paper with exactly the same colour variations, as a deceptive [fake](#). The paper can be glued over the hinges and crevices of a locked metal box. By opening the box, the paper is torn. Because it is hard to hide tears in it in the case of someone else opening the box before you, and because the precise colour variations are hard to fake deceptively, you'll notice whether someone else has opened the box before you, thereby providing [tamper evidence](#).

Very strong [glue](#) probably would need to be used, to ensure better that the paper is noticeably damaged through the opening of the box. If the paper can be glued on the inside of the metal box (instead of the outside), that might be best in order to prevent someone removing the glue through the use of glue [solvents](#) like [acetone](#), the doing of which would present itself as a point of weakness in the security system if the gluing were instead performed on the outside.

Perhaps the simplest and best idea

Perhaps the simplest and best [tamper evident system](#) is simply to wrap-up computing devices in some material that [retains its shape when at rest](#), but that also can quite easily [lose its shape](#) when slightly disturbed in such fashion that it is [hard to achieve once again the prior shape](#). [Shell-suit](#) material may be good for this. Also [silk](#) (maybe an old silk scarf perhaps) might be good. Certain kinds of [crumpling plastic bags](#) might also be good. Simply using [bubble wrap](#) in such ways, will likely provide at least some level of security.

[Software](#) based [tamper](#) checking using security images

If relying on [software](#) based [authentication](#) for making sure there is no [tamper evidence](#) between two security [photos](#), using a [tripod](#) is likely necessary as two security photos requiring comparison need to have been taken from the same angle and location (at least approximately within a certain degree of deviation). Using a [smartphone camera](#) may be preferred, as the software for checking two security photos can then be automatically run on the same device used for capturing the photos, which in some ways (in respect of being such an integrated approach) presents a more attractive security solution. However, a [digital camera](#) can also be used, with the associated [SD card](#) being then taken out and placed into a device that can run the related security-photo matching software.

The software solution can be manually [coded](#), if none already exists. A rough idea of an [algorithm](#) that might be acceptable for the solution, is as follows:

1. If necessary, generate many security photo pairs, by using the first photo as the first element in each pair, and then by generating different and distinct second elements by re-positioning the second photo within a [frame](#) (solely in computational terms), by small amounts away from its original position, such that all re-positioning is tried in all the generated re-positioned photos, where re-positioning only occurs within a set [radius](#) from the original position of the second photo. Each of the generated security photo pairs should then go through the process outlined in the remaining steps in order to determine whether any security-compromise disturbance has occurred. This step is here to mitigate against the potential of the camera position used between two security-photo-taking events being slightly different (probably because of slight physical disturbances).
  2. Increase size of [pixels](#) in photos so that they look more [pix-elated](#), but not so much so as to have the software fail of its essential purpose.
  3. Calculate [standard deviation](#) on [colour differences](#) (can possibly use [Euclidean distance between two RGB values mapped to 3D space](#)) between corresponding pixels in both photos (using this method requires that [lighting](#) be exactly the same between two security-photo-taking events, which might be contrived by closing curtains and switching on [electric lighting](#)).
  4. Make sure standard deviation is beneath a certain [toleration level](#), where such bounding implies that there were no physical-disturbance-based security compromises.
-

## Mind-reading attacks

[Mind-reading](#) powers may at first appear to you as something confined only to the realm of [science fiction](#) and [fantasy](#). However, technology has now advanced so much that it is now literally possible to read the minds of others (using technology), as is evident from the [Wikipedia](#) page on [brain-reading](#).

It is left up to the reader to decide whether the [psychics](#) described to us in fiction, exist at all in reality. However, you should consider the principle that dictates that, if there is a chance that they do exist, then it may well be prudent mitigating against such [risk](#), even if in reality such risk does not exist. Also, it's worthwhile to consider that phenomena such as [levitation](#) that was previously associated with the [miraculous](#), now can possibly be explained away using the discovered scientific phenomena of invisible forces such as those involved in [magnetism](#), [electric fields](#), etc.

Please refer to the earlier section entitled "[Psychic spying of password](#)", for information on mind-reading attacks related to the psychic spying of passwords, and on how they can be overcome.

### 'Inception' styled attacks

[Psychic attacks](#) may take the form of trying to [crack passwords](#) and security credentials by entering one's dreams (like the film "[Inception](#)"). To mitigate against this [security risk](#), it may in fact be preferable not to [memorise passwords](#). It might be better to store them in [computer files or on paper](#), and then only call them to mind when entering them. Also, the use of [security tokens](#) (such as [USB](#) security tokens<sup>32</sup>) as well as the use of [password encryption](#), can overcome such psychic attacks.

---

---

<sup>32</sup> Some information on certain [USB security tokens](#) is contained in the earlier section entitled "[Tokens for keys](#)".

## Simple security measures

### Put computer to sleep when not at it

When not at your computer, it may be a good idea to put it to sleep. The reason being is that if an intruder has remote control of your computer, and you only leave your computer locked when you are not there, without your knowing, they may start to do screen-visible things on your computer when you are not there. When you are at your computer, the intruder may be frustrated in trying to do such things because you are then monitoring your screen. So putting your computer to sleep when you are not there, may be a good idea in order to frustrate such remote-controlling activities.

### Shut down device when not in use

Shutting down your phone or other computing device before you go to sleep is probably a good idea—it's an easy precautionary measure. Doing the same at other times when they are not much needed, might also be a good idea.

### Play sound continuously on computing device

A simple security measure to monitor whether someone, with physical access to your computing device that is near to you:

- ⦿ has taken it away from you, O R
- ⦿ is shutting down or rebooting the device

*(as well as maybe doing some other things to the device that potentially compromise security)*

is simply to play some continuous sound on it that is heard through its speaker. This could be simply playing the radio on the device. If someone then were to take it away from being near you, normally you would notice the absence of, or lowering of volume in, the sound (due to the greater distance of the device from you). Also, if someone were instead to shut down or reboot the device (perhaps as a precursor to tampering), normally you would notice that too because normally the sound would disappear.

Do note though, it could be possible to replace the sound playing on your device with the playing of the same sound on another device, such that you don't notice the occurrence of any of these security compromising actions. In light of such, it may be good to play sound that is hard for others to reproduce precisely, perhaps something as simple as music from your own personal favourites playlist.

Obviously when using these measures, screen locking should also be employed.

---

## Broad security principles

### Stop funding the [spies](#) and [hackers](#)

As hinted at in “[Desktop Witness](#)” by [Michael A. Caloyannides](#), we may well be mostly inadvertently funding the work of [spies](#) and [hackers](#), by purchasing [personal computing](#) resources without investing reasonable efforts into securing and securely using such resources. So whenever budgeting for computer resources, also factor-in the efforts required to keep and use such equipment securely—you don’t want to be the worse-off because of computer resource purchases you made.

### Report [cybercrime](#) to the [police](#)

[Reporting crime to the police](#) doesn’t only potentially obtain for you, free and tailored [computer security](#) assistance from authorities. It also helps to prevent [crime](#) in general such that there may be fewer criminals around in the first place to attack your systems, perhaps simply because they have been warded off from committing any crime due to your prior involvement of the [police](#).

### Think in terms of gradual movement along a security-level [continuum](#)

Security isn’t simply a question of a [Boolean](#) secure-or-not-secure proposition. There is a [continuum](#) of security levels, from least secure to most secure. Instead of thinking in terms of simply needing to be secure (which is certainly something worthy of your aspirations), think in terms of progressively moving along the continuum, day-by-day, gradually getting more and more secure, on the whole. Such progressive movement makes sense in terms of balancing the risks of security compromise with the efforts required for becoming secure. Also, don’t be concerned about ‘over doing it’—it’s not an [exact science](#), and it’s often better to do too much than to do too little. Also, don’t worry about mistakes too much; so long as they’re not too frequent, [fail-safe](#) mechanisms in your security, coupled with threats being [probabilistic](#), will mean that they won’t matter that much.

The National Cyber Security Centre for the UK ([NCSC](#)) touches upon this principle in their blog post entitled “[Not perfect, but better: improving security one step at a time](#)”.

## Minimally-above-average security

A security principle exists, where entities seek security levels that are minimally-above-average security. The rationale behind such an approach is as follows. It is supposed that adversaries have in their arsenal of weapons, attacks that overcome average security levels, but not attacks that can do much more. They maximally target users such that the “[hacking](#) success to work done” ratio is roughly optimal, which generally means constructing attacks that can be reused as much as possible (to save on work done). Then for every extra unit of work done, the ratio diminishes, equating to less “[bang for your buck](#)” (for the hacker). So to defend against such adversaries, an entity may choose to implement security that is above average, but only so much so as to defend against such adversaries who look more for common [security weaknesses](#) amongst users as a whole, rather than being able to hack the most secure of systems (“[Fort Knox](#)” like systems).

The National Cyber Security Centre for the UK ([NCSC](#)) touches upon this principle in their blog post entitled “[Not perfect, but better: improving security one step at a time](#)”.

Some [NCSC](#) information on the behaviours and characteristics of cyber crime and cyber criminals can be found in their report located [here](#).

It might be that [Trammell Hudson](#) also agrees with this principle since on his website, he sometimes uses phrases such as “[slightly better security](#)”.

## Publishing security methods

It may seem counterintuitive to [publish](#) your security methods. However, doing so can actually increase your overall security (rather than decrease it).

Firstly, by publishing your methods, you can get useful feedback in respect of the weaknesses and strengths of your current security methods, and then adjust them accordingly.

Secondly, your activities are unlikely to be completely isolated; instead, they are likely to depend upon a number of [relationships](#) with other entities. From this perspective, improving the security of the wider [community](#), can provide better security for yourself.

Thirdly and finally, the widespread adoption of better security methods, can make the aspect of [hacking](#) you more difficult than if only you adopt better security methods. Suppose you are a [hacker](#) and you have on your agenda to hack 100 entities. If just one entity improves their security, hacking your 100 entities perhaps isn't negatively impacted so much. However, if all 100 entities improve their security, then hacking any one entity is made even more difficult because the hacker has to bear in mind that they also have to [allot sufficient resources](#) (including time) also to hacking the 99 other entities—the hacker may become simply too stretched for time/resources.

## User [randomly](#) selecting unit from off physical shelves

Getting a unit sent to you in the [post](#) is not necessarily the most secure way to obtain a unit of any particular product. In particular, obtaining units in such ways are vulnerable to man-in-the-middle ([MITM](#)) attacks.

Instead, it may well be more secure to buy a unit by first physically visiting a large physical store selected at [random](#), that is geographically distant from the user's address, and then when at the store to choose and purchase personally, a [shrink-wrapped](#) unit fresh from its factory (or other product source) from off the physical shelves, using random selection, where there are many other units of the same product on the shelves.

When [random](#) is not random

[Behavioural cues](#) can sometimes affect a person's [mental processes](#), or be used to [read a person's mental processes](#), such that trying to implement [random selection](#) solely with your mind, ends up with selection that's not so [random](#). In such cases, you should seek to generate random selection externally, which could be something as simple as [tossing a coin](#) or rolling some [dice](#).

## Ordering many units of same product

Security can be increased by purchasing very many units of the same product. Because you have so many units in your possession, it should be more difficult for an attacker to tamper with every single unit. One unit can then be chosen at [random](#), and all other units can then be [returned](#) for either partial or full [refunds](#).

Using multiple [channels](#) to obtain product

Perhaps a good way to ensure a product isn't tampered with before use, is to obtain the product through multiple [channels](#). For example, [software](#) can be [downloaded](#), bought from a physical shop<sup>33</sup>, and also bought from an online shop<sup>34</sup>. The three versions of what should be the same [installation](#) software, once obtained, can then be [byte-for-byte compared](#) with each other to see whether they are exact copies—if one isn't, then it likely points to [tampering](#) having been conducted to one or more of them.

---

<sup>33</sup> Preferably a large one, having many copies of the software on the shelves (see principle outlined in subsection "[User randomly selecting unit from off physical shelves](#)").

<sup>34</sup> Preferably one that uses tamper-evident security mechanisms to protect against tampering (see subsections "[Specifically for goods in physical transit](#)" and "[Exploiting unrepeatable patterns for tamper evidence](#)" for more information on such mechanisms).

## Discerning unit least likely to have been compromised

If obtaining [hardware](#), instead of using [random](#) selection, a potentially better way to select which unit to keep (one that hopefully hasn't undergone any [tampering](#)) is as follows. First, purchase several units of the same hardware. Once several units have been obtained—let's say for sake of example, seven units—the user needs to figure out which of the units have the least likelihood of having undergone any tampering, ideally without damaging them so that excess units can be [returned for refunds](#) (hopefully full [refunds](#)). To help in this figuring out, the user can use one or more [non-invasive](#) and [non-destructive measuring](#) methods, some of which are documented in the next section entitled “[Measuring physical properties for authentication](#)”.

The measuring can be undertaken before opening the hopefully [shrink-wrapped](#) boxes that contain the units (to facilitate being able to [return](#), later on, no-longer-needed units). The measurements that are most common amongst the units, determine the [quality-control](#) parameters. Then any one unit fitting those parameters can be selected. This selection process on the whole is better than pure random selection, as it is better able to sift out any tampered units that may be in the mix, by using a [balance-of-probabilities](#) approach—tampered units hopefully will have odd measurement readings. So for example, if 6 units have the weight 100g with only one unit having the weight 101g, you would choose not to keep the unit weighing 101g (the odd, and potentially tampered-with unit).

Since the measuring techniques involved hopefully won't damage the units, returning excess units for a full [refund](#) should hopefully not be a problem.

## Measuring [physical properties for authentication](#)

How do you confirm that a physical device in your possession is exactly the same as the device it ought to be, as it was manufactured? It could not only have undergone [tampering](#), but it could in fact be an entirely different device that just has the appearance of the actual genuine device<sup>35</sup>.

Measuring [physical properties](#) appears to be a good way to make sure you have the genuine device in your possession. Whilst an adversary may be able to replicate one of the physical properties, the convolution (or [intersection](#) you might say) of several (or maybe even just two) is likely quite hard to replicate in any kind of imitation or tampered-with device. Once we acknowledge this principle is valid, we now turn to what physical properties we can measure for the application of this principle.

### Weight

[Weight is something very easy to measure](#), so there are no worries about using it.

---

<sup>35</sup> Some of such undermining activities are classified as [evil maid attacks](#).

## Volume

In the case of devices incapable of being damaged by slow-moving water, to measure volume, we can simply measure the displacement of water by placing the device in water<sup>36</sup>.

In the case of other devices, if the device is low cost, a user can purchase two units of the device, one to be [measured using destructive measures](#), the other one to keep, assuming the security tests succeed. The one to be measured, can have its volume measured in the same way as just outlined. Doing so will destroy the device, but you will have the other device that you can keep. Once you have ordered the two units, you select the one to measure by [random](#) selection, which is very important. In the scenario that an adversary was able to replace just one of the units, they will likely have replaced both units (and not just one) with imitation or tampered-with units. So testing one of the units will likely apply to tests of the other unit<sup>37</sup>.

It should be noted that it does appear—from some brief [internet research](#)—that so-called "[pure water](#)" and/or [distilled water](#) are possibly capable of [having electronics submerged in them](#), without any damage to the [electronics](#) occurring as a result.

## Magnetic weight and images

The '[magnetic](#) weight' of a device can be measured, as in the force of attraction between the device and a magnet of some specific number of [newtons](#) force, at a set specific distance away from the device. A simple set of normal ([gravitational](#) based) [weighing scales](#) can be used for this. What you do is:

- i. you place a [magnet](#) a set distance above the scales top;
- ii. you then place the device on the top of the scales;
- iii. you measure the weight;
- iv. you take away the magnet and measure the weight again;
- v. you then calculate the weight difference to give an indication of the '*magnetic weight*'.

It is suspected that magnetic resonance imaging ([MRI](#)) can be used to profile an [electronic](#) device, for security purposes. The question is whether the cost of performing such imaging can be sufficiently reduced for use by general users. It is suspected that it probably can. The "50x50mm Small Magnetic Field Viewing Paper" product available [here](#) appears like it might help in this regard.

## Electric field imaging/detection

Like [RF imaging/detection](#), [electric field](#) imaging/detection of turned-on [electronics](#) might be helpful.

---

<sup>36</sup> As used in the [anecdotal origins story of the Archimedes' principle](#).

<sup>37</sup> Interestingly, as an aside, this idea sounds a bit like measurement in reliance on [quantum entanglement](#) phenomena.

## Electro-magnetic spectrum

### Visible spectrum photography

Straight-forward conventional [photography](#) using the [visible spectrum](#) can be used to check whether a certain device looks externally (and sometimes also internally) like it should (a form of [quality control](#) classed under [visual inspection](#)). With respect to both photographing the internals, as well as convincing users that there is no embedded, hidden [espionage technology](#), it is useful for the device to use transparent materials wherever possible<sup>38</sup>.

### Infra-red scanning

[T rays](#) are analogous to [x rays](#) but instead involve [infra-red](#). [Infra-red filters](#) appear to be readily available, at relatively low prices, for [cameras](#) (including those on [mobile phones](#)). T rays are capable of passing through plastic but not through metal; this property enables t rays to be used to detect metal weapons concealed on persons. Could it be possible that such t rays might also pass through [silicon](#)? If t rays do in fact pass through silicon and related materials, then perhaps [cameras](#) (including those on [mobile phones](#)) can be adapted to take t rays of [electronic](#) devices, as a form of security verification.

### X rays

The "[Mylar X-Ray Film, TF-160-255, 500 Pre-Cut Clrcles, Unopened](#)" product priced at just £12.75, appears to be [X-ray capturing film](#) at a relatively low price. Since [x rays are present in sunlight](#), could it be possible that an [x-ray photograph](#) of a device could be created simply using [sunlight](#) and this film, by keeping the apparatus and device still over the course of hours or days (perhaps with the use of a [filter](#) that blocks out all non-x-ray light)? It looks like x rays already form an accepted method of confirming certain aspects of at least certain [microchip electronics](#) hardware—see [here](#) and also [information on industrial radiography](#).

### Microwave testing

[Microwave testing](#) is already a tool used in [engineering](#) for performing [non-destructive testing](#) to find defects in technical parts. As it is presently used in engineering, it most likely isn't low-cost enough. However, could it be that a [microwave oven](#), in conjunction with a [smartphone](#), might be capable of being able to perform such testing? The project hosted [here](#) could possibly be useful for this.

### Radio frequency (RF) imaging/detection

From [internet research](#), it hasn't been possible to find much information about this. However, it seems like it might be something that is useful, especially with respect to any incidental [RF fields produced by powered-on electronics](#).

---

<sup>38</sup> A point seemingly made by [Andrew "bunnie" Huang](#) in his blog post entitled "[Can We Build Trustable Hardware?](#)".

## Ultrasound

[Ultrasound images](#) and/or readings, appear to be quite likely useful for security [authentication](#) of devices. Unfortunately, normally the price of the associated equipment is relatively high. However, there is such a thing as a [DIY ultrasound imaging kit](#) which could directly or indirectly bring down the cost of using this technology. Also, a simple [ultrasonic sensor](#), which seems potentially to be much cheaper and affordable, may be sufficient for exploiting ultrasound to do security authentications. According to [Wikipedia](#), ultrasound testing has been used for [industrial non-destructive testing of welded joints from at least the 1960s](#).

## Other methods

Other usable [non-destructive testing](#) methods might be documented on [Wikipedia](#), especially at the address of [this hyperlink](#). Other such methods may also be documented [here](#) on the betrustrad wiki.

## Geospatial

Because of things like [hidden cameras](#), you may want to enter security credentials (such as [passwords](#)), only in secure locations. For example, you may choose not to unlock your phone when in public places because of the increased likelihood of an adversary surreptitiously [photographing](#) or [videoing](#) the password you enter.

## Based on which region

The [United States of America](#) used to have [severe restrictions on the export of cryptographic technology](#). In contrast, other geographic areas (such as perhaps [Germany](#) or the [Netherlands](#)) probably are relatively liberal in respect of such technology. With such variations in mind, the level of security you attain may depend on which geographic areas you use for the various elements of your security.

Also, certain geographies are more prone to attack because of various factors such as things of a political nature, as well as things to do with how powerful the local [military](#) are. In this respect, it may even be worthwhile to rely on geographies that are technologically [backward](#), and that have weaker [politics](#) and/or political groups.

## Time based

### Based on time passed

Adversaries may initiate their attacks only from a certain time onward. And perhaps we can make assumptions as to when those attacks were definitely not taking place. For sake of example, let's say that we have an assumption that we were not being attacked six months ago and so were not compromised at that time. If some of the artefacts produced from historic activity dating from before six months ago were isolated six or more months ago, it might be possible to gain extra security drawn from such artefacts.

### Example 1

Nine months ago, I downloaded the [public key](#) for some [website](#), and [saved](#) it in a way that kept it quite isolated from other systems from that time onward. For example, isolation may have been induced by [printing](#) out the key, and keeping it in a [safe](#). Isolation could also have been induced by [burning](#) it to an [encrypted read-only CD](#)<sup>39</sup>.

Now that nine months have passed, I find that adversaries are [tampering](#) with my [internet communications](#) so that I am [downloading](#) the incorrect key. In reliance on '*security based on time passed*', I can retrieve the genuine key I saved nine months ago.

### Example 2

I often engage in [email](#) exchanges with a colleague who always writes their public [authentication PGP](#) key at the bottom of the emails they send. Three months ago, an adversary started intercepting the emails and changing the key in all emails sent henceforth. By relying on '*security based on time passed*', I can access the emails from my colleague from before six months ago (a time at which I know I had system security), and obtain the genuine key. I can then see that the key is now different and therefore figure out that something suspicious is now happening.

### Vulnerability when used for software

Note that '*security based on time passed*' can have a [vulnerability](#) in respect of [security holes](#) in [software](#). For example, software from 5 years ago may have security holes that were discovered and [published](#) since then, and from this perspective, it may be better not to rely on that historic version of the software, and instead try to obtain a newer version of the software securely (where the holes have been fixed). With this in mind, if using '*security based on time passed*' in conjunction with software, it may be best only to use it with very simple software, where the associated [algorithms](#) and [code](#) have been thoroughly security researched and tested. This is to mitigate against the chance that security holes are discovered subsequent to your storage of the software under this principle.

### Based on time taken to [forge](#)

The time taken to [forge](#) something, can be used in security mechanisms. For example, someone could place their [mobile phone](#) in a seamless ink-absorbing paper bag, the tearing of which is hard to repair through the use of things like [super glue](#), and then sign a [random fountain-pen](#) ink squiggle on it. The time taken to forge a replacement paper bag with the same squiggle, can mean that the phone is secured for several hours from the time of the squiggle.

---

<sup>39</sup> For information on best ways to establish such a read-only [CD](#), see the earlier section entitled "[Rewritable media vs optical ROM discs](#)".

## Using most secure window of time

Instead of entering your security credentials at any old time, a selected window of time can be chosen when security is high based on factors such as when [hackers](#) are awake, etc. The security credentials can be entered in that window of time, and they can be entered so that you don't need them again for a while. Eg. log-in to your computer at 6am because most hackers in your own [time zone](#) are perhaps asleep at that time.

## Preventing lapses in security

Simple [measures](#) can be put in place to help remind oneself to stay secure. For example, when using a [laptop](#), why not rest it on its [wallet case](#) (if it has one)? That way, you are reminded to place it in its wallet case when you close it up. Another simple measure is not to leave a [key](#) in an unlocked [self-locking padlock](#)—you no longer need the key, so put it away. That way, you reduce the risk of forgetting to store the key after you've used it.



*Security reminder of resting [Chromebook](#) in its wallet whilst it is being used*



*Security measure of taking key out of self-locking [padlock](#)*

The forming of security [habits](#) is another simple measure to help prevent forgetting to secure things. On day 1 and day 2, perhaps you make mistakes in your security routine; however, through simple repetition over many days, weeks, months, etc., you simply reduce more and more your mistakes because of the force of habit acquired.

## DIY security principle

Building your own devices and security can be good for security reasons. You can better make sure that hidden [espionage technology](#) (possibly embedded in materials) is not present when doing it yourself ([DIY](#)). The DIY principle can be even more effective when you choose to use very cheap and commonly available materials, because you can often be more sure that hidden embedded espionage technology is not in such materials. For example, why not think about possibly building a computer out of [cardboard](#)? You can pick cardboard at [random](#), and be pretty sure no hidden espionage technology is embedded in the cardboard.

The DIY principle can also be applied to building [software](#), as in you can do things like [build software from source](#) as touched upon in the earlier section entitled “[Compiling from source](#)”.

## “Destroy key when attacked”

The [Heads](#) system<sup>40</sup> outlines a security principle that can perhaps be roughly described as 'destroy-key-when-attacked security'<sup>41</sup>. The trusted platform module ([TPM](#)) on a [motherboard](#) stores the [encryption-decryption \(symmetric\) key](#) that is only accessible upon entering the correct [password](#). The Heads system will have the TPM destroy the key if incorrect passwords are entered too frequently (if it seems like the computer system is under attack [in relation to so-called [rate limiting](#)]). Once the key has been destroyed, the key should no longer be on that computer system. The true user will find out that the key has been destroyed, and then use their [backup](#) measures to retrieve the key (relying on the [backup of the key](#) outside the computer system). This security principle can be applied more broadly. For example, [cryptocurrency](#) (such as [Bitcoin](#)) [keys](#) can perhaps be safeguarded in the same way<sup>42</sup>.

## Relying on high production cost of certain security tokens

Perhaps a novel way to ensure better communication of things like [public keys](#), is for an organisation to create a security token containing the key, that is very expensive to produce and replicate. It could be, for example, a genuine [gold coin](#) with the key on it, or some kind of [holographic](#) paper or card with the key on it. What would happen, is that the organisation would [post](#) out the security token to end users. The end users would then possibly pass the security token amongst themselves, or just post it back to the organisation. Because it would be so expensive to (re-)create the token, the token would not be thrown away, but simply shared in order to communicate the public key to end users. Adversaries would find it too costly to create fake tokens, because of the expense in creating the token.

---

<sup>40</sup> As outlined in the earlier section entitled “[Custom BIOS/UEFI and which one to use](#)”.




<sup>41</sup> It is similar to the military policy known as [scorched earth](#).

<sup>42</sup> More information on this may be available under the topics of [TPM binding and TPM sealing](#).

# What to do when you discover your computer has been hacked

## Backing-up files

If it is ever discovered that a computer has been hacked,

-  i) restarting the computer into some kind of safe-mode with WiFi, Bluetooth, NFC and other things deactivated, might be a secure way to get access to your files so that you can back them up.
-  ii) However, likely more secure than this is to run off the same computer, another operating system installation using a physical drive that was never previously used with the computer, or a live DVD/CD, and then access your files through the other OS installation or live DVD/CD so that you can back them up.
-  iii) Even more secure, is to take the drive out containing your files, interface with the drive using a completely different computer that is running a live DVD/CD, and then access your files through the live DVD/CD in order to back them up.

## When to change digital passwords and keys?

You may be tempted to change such security credentials straight-away after discovering that you have been hacked. However, could it be better to wait until you are sure that your system is secure rather than changing such credentials using an insecure system?

## Further information

The National Cyber Security Centre for the UK (NCSC) outlines steps to be urgently taken when you've discovered you've been infected with malware. Also, see here for their guide for small businesses on response and recovery from cyberattacks.

---

## Miscellaneous notes

### National Cyber Security Centre

[Cyber-security](#) advice is freely available on the [National Cyber Security Centre \(NCSC\) website](#). Specific NCSC information that may be of interest to you:

- ⊙ For [small businesses](#): [general information](#); [online security](#).
- ⊙ [Listed, advertised, and supported security products and services](#).
- ⊙ [Advice and guidance on the issue of authentication](#).
- ⊙ [Security advice on home and mobile working](#).
- ⊙ [Information on cyber security in relation to GDPR](#) ([General Data Protection Regulation](#)).
- ⊙ [Information on security in relation to computer peripherals](#)  
(such as [USB keyboards](#), [Bluetooth](#) devices, [memory sticks](#), etc.)
- ⊙ [8 Principles of Secure Development & Deployment](#).
- ⊙ “[Secure design principles: Guides for the design of cyber secure systems](#)” publication
- ⊙ [Information and guidance on the importance of having high security specifically for your email account](#).
- ⊙ [Guidance on whether you need to use antivirus software on your Android device](#).
- ⊙ [Information explaining what is malware and how to defend against it](#).

### Cybersecurity standards

See [here](#) for [cybersecurity](#)-standards information from which might be obtained, insights into the [security measures](#) and processes helpful for an entity's security.

### Deep hardware hacking

Consideration of deep [hardware-hacking attacks](#) is probably an overkill for most individuals, [sole traders](#) and [small businesses](#)—have to draw the line somewhere. However, if you are interested in defending against such attacks, the [Novena open-source](#) computing [hardware platform](#) and other related information hosted [here](#), may be of interest to you.

### Cryptocurrency security

If you are interested in adopting [security measures](#) suggested for [Bitcoin](#) users, have a look at the [Glacier protocol](#).

## Using [phones](#) and computers as [motion detector](#) alarms

[Phones](#) and computers can be turned into [motion detector](#) alarms, by using their built-in [cameras](#) and other [sensors](#), with [apps](#) such as the “[Haven: Keep Watch](#)” app. This could provide extra security during sleeping hours. [Psychic](#) phenomena, including phenomena that have been verified and accepted by the [scientific community](#), might be able to [induce a person to stay asleep or suffer amnesia](#) whilst intruders enter premises and room whilst the person is asleep; technological measures such as loud [alarms](#) may be helpful to defend against such attacks.

## [Steganography](#): easy hiding of information in [computer documents](#)

White-on-white text in [computer documents](#) and [emails](#) can be used for hiding and sending security sensitive information. This might be both simple and effective for low [threat models](#).

---

# Appendix

## New security inventions requiring a non-trivial investment in new technology

### Cryptocurrency-like mining to increase trust

As applied to software

An interesting security idea involves using CPU idle time, and the idle time of other processors, to put the processors to work to perform computations that increase trust in a particular software. The processors would be put to extra work, but not so much extra work so as to have the computer use more electricity (so as not to impact on computer operating costs), to mine so-called “authentication coins” (similar to BitCoins). The so-called “authentication coins” would be mined so as to provide some authentication for some piece of software (perhaps through PGP signing of the installation files of the software), thereby increasing trust in the software. If there are 100,000 users using the particular software, all having each generated these “authentication coins” (which don't cost anything to make when using idle time in such ways), and those coins are published, a new user can download the published coins, and then say to themselves something similar to the following:

1. 1,000,000 “authentication coins” have been produced authenticating this software;
2. I've just algorithmically checked the coins are genuine, using a trusted installation of some trusted software, and they are indeed genuine;
3. That's a great deal of processor-hour work;
4. It's quite unlikely that an adversary has invested this amount of processor work in authenticating any hoax malware-containing software simulating the true software, so I think I can trust this software is the genuine software from the real provider.

A counter-argument to using this protocol is that adversaries with supercomputers can easily fake such numbers of coins. That appears likely to be true when only one piece of software is made a target.

To mitigate against this, wide adoption of this protocol can be promoted. If every provider adopted this protocol, the activities of adversaries in general would be reduced, because of the general increase in work required for each [hacking](#) activity of this type. This would lead concretely to a higher-than-otherwise increase in security for each user (incidentally, this security principle, that can perhaps be termed as “security via mass adoption”, can also be potentially achieved when publishing security methods<sup>43</sup>). The knock-on effect is that of increasing security both for the individual user, as well as for everyone in a general way.

As applied to public [digital keys](#) (including those used in [security certificates](#))

Suppose an organisation [publishes](#) a [digital public key](#) on a [website](#), and suppose the website address is well known. The website address may be printed on business cards, advertising, in magazines, in TV adverts, etc. such that it is well publicised. However, let's say the public [cryptography key](#) isn't so well publicised, and in a connected way, not so well [trusted](#). By using [cryptocurrency](#) technology, [mining](#) activity can take place at the 'node' represented by both the website address and the public key. The mining can take place during the [idle CPU](#) time, and idle time of the other [processors](#), of the organisation's computers. Over the course of a year, let's suppose 100 coins are mined. The [credentials](#) for those coins are published on the web-page on which the digital public key is also published. Users can then download the key, along with the '[proof of work](#)' embodied in the coin credentials, and see that the public key is [buttressed](#) by the mining work conducted to increase trust in the public key. Whilst it's true that adversarial entities can also do mining to buttress fake keys, it does cost them. The “security via mass adoption” principle (mentioned in the [last section](#)) applies, so that mass adoption can substantially increase security both for the individual user and for the wider community. Perhaps mandating that such mining take place for organisations, might happen one day, in order that users are better enabled to trust the digital keys of such organisations.

The same [authentication protocol](#) can also be used for the public digital keys used in [security certificates](#).

## [Lock screen](#) with related [sound](#)-based security

Additional security can be added to [lock screens](#), if:

1. they play music through the [computing device](#)'s [speaker](#) from a 'favourites' [music playlist](#) whilst the lock is on,
2. that music is turned into a high-pitched continuous [beep](#) when a person [inputs](#) into the device (whether that be by touching a [touchscreen](#), pressing a [keyboard](#) key, moving a [mouse](#), or otherwise), and
3. the speakers stop outputting [sound](#) altogether when the lock is unlocked.

Please refer to the earlier section entitled “[Play sound continuously on computing device](#)” for some of the reasons why this improves security.

---

<sup>43</sup> Such achievement is documented in the section entitled “[Publishing security methods](#)”.

Even more security can potentially be attained if you also use 2FA ([two-factor authentication](#)<sup>44</sup>), with the second factor being a [hardware security token](#)<sup>45</sup>, to unlock the lock screen such that when people correctly enter your [knowledge factor](#)<sup>46</sup> without the token being present, the [audio](#) changes to a special [alert](#) sound indicating that someone else has just attempted to unlock your device with what is in fact the correct knowledge factor, but actually failed because of the lack of security token.

The security outlined in this section might be good for users in [public spaces](#) that are shared with others, where the user might move around away from their device but still within [hearing distance](#) of it.

### [Client-server noise-audio-based secure-password-communication system](#)

1. [Server](#) sends to the [client](#), [white noise audio](#) that is genuinely [unpredictable](#) because of the [random](#) element in the noise.
2. The client hears the audio perhaps out of their headphones as well as out of their [laptop speakers](#).
3. The speaker/[earphone](#) set-up is contrived such that the noise audio is heard coming out just near the [microphone](#) (perhaps by placing one of the [headphone](#) earphones just by the microphone).
4. User whispers almost inaudibly, the [password](#) into the microphone.
5. Server receives audio from the client's microphone (the noise originally sent from the server to the client is in the background of such audio [because of the speaker/earphone set-up just mentioned]).
6. Because the server knows the precise noise audio sent to the client, the server can remove the precise noise from the received audio to authenticate the password.
7. If an [eavesdropper records audio](#) sent by client to server, they cannot replay that audio to achieve [authentication](#) (a kind of [replay attack](#)), as the whispered password is inextricably admixed with that specific, precise and unpredictable noise (unpredictable partly because that precise noise audio is not sufficiently repeated on other occasions).
8. Hopefully, the password audio is also not discoverable by an eavesdropper primarily because of the user whispering the password very softly, such that the whispering is completely blotted-out by the noise audio unless you know precisely what noise audio was used—hopefully, standard [denoising algorithms](#) that an eavesdropper might try, won't work, but instead simple subtraction of the precise noise audio (which the server will hopefully be able to do) will work.

---

<sup>44</sup> Please refer to the earlier section entitled "[Multi-step authentication and multi-factor authentication](#)" for information on the benefits of [multi-factor authentication](#), including those of 2FA.

<sup>45</sup> Please refer to the earlier section entitled "[Tokens for keys](#)" for guidance on [key-based security hardware tokens](#).

<sup>46</sup> [Passwords](#) are one kind of knowledge factor.

## Port source code to higher-level programming language as a computer-security step having its basis in secure coding

A possible idea is to translate [open-source source code](#) to a [higher-level programming language](#) (such as perhaps the [Eiffel](#) programming language) [perhaps initially in an automated way], in such fashion that the [software](#) has fewer potential [security compromises](#). The Eiffel language is apparently [good for safety, and so also good for security](#) (in a related way), [when compared with the majority of programming languages](#), so if you want fewer [security risks](#) (even if it is at the price of somewhat slower speed execution), then perhaps porting to Eiffel (where possible) could be a good idea depending upon the circumstances. An example of how Eiffel can improve safety: suppose you have a function that implements some kind of complex [sorting](#); Eiffel can add a [sanity-test](#) constraint using Eiffel's specially-designated [language constructs](#) for [post-conditions](#), to [throw an error](#) if certain output properties are wrong (perhaps something as simple as checking that some chosen element is indeed greater than one of its preceding elements).

## Security by pre-loaded private key

With respect to a [USB cryptography-key security token](#), if the product comes with a pre-loaded [private key](#) for [authentication](#), once the product arrives, the user can use the key to [sign](#) some [long](#) message they've just [randomly](#) generated. The user will then receive the [message digest](#). The user can then with their [secure communications](#) device<sup>47</sup>, submit the long random message to the [authentication server](#) of the company that produced the token. The authentication server has the same private key securely stored, and so, is able to reproduce the same message digest. The user sees the server-generated message digest, and then is willing to trust that the product is genuinely the same product produced by the authorised provider, at least in respect of certain key components of the product. This security method (a kind of [zero-knowledge authentication protocol](#)) works against 'complete fake' [attacks](#) focused on the token, but will not necessarily work against [tampering](#) attacks focused on the token.

If the device is designed so that the private key is destroyed whenever any tampering is attempted, then defence against this latter kind of attack will also be achieved. [Trammell Hudson's Heads boot firmware](#), that destroys a private key within a TPM ([Trusted Platform Module](#)) when incorrect [passwords](#) are entered too frequently (see the section entitled "[Destroy key when attacked](#)"), coupled with the use of [epoxy resin](#) to protect [hardware](#) components from tampering, might be helpful in the drawing-up of such design features.

---

<sup>47</sup> See the following sections for specific information that assists in the establishment of such a device:

- [Regarding how to obtain software](#)
- [Factory resets](#)
- [Digital cryptography:  
security certificates, keys & tokens](#)
- [Wireless Communications](#)