Navigating Privacy, Anonymity, and Security in the Digital Age
V0.6

Mohammad Kamal

April 16, 2025

Contents

About

Who am I?

I am currently a university student at Shiraz University, one of the top five universities in Iran, where I am proud to hold the rank of fourth among my peers in my cohort.

My journey into serious writing and editing began when I took on the role of student editor for Mobtakeran Publisher. This experience ignited my passion for the written word and led to several publications, including a book titled "Introduction to Python," which is available in Persian on my GitHub profile.

In addition to my writing activities, I am teaching a course called "Online Security and Privacy Workshop" in our department at the university. This marks my third time delivering this course, and it has inspired me to compile the related content into a comprehensive book.

Github — Telegram Channel — Youtube

This Book's Approach

I have observed that many books on security often skip the fundamental concepts, making it difficult for newcomers to fully understand the material. This realization inspired me to write an introductory book that thoroughly covers the essentials of privacy, security, and a touch of anonymity for those just starting their journey in this field. When I began learning about security, I relied on blogs, but I struggled to find high-quality resources. My aim is to bridge that gap.

With my writing experience, I aim to present the fundamental concepts in a way that fosters critical thinking and helps readers develop a hacker's mindset. I do this by introducing ideas that are relevant across various domains, rather than confining the discussion to a single area. Additionally, I utilize a question-and-answer approach in my teaching, which I find to be an effective method for enhancing understanding and fostering intellectual growth.

Why This Book Doesn't Dive Deep into Practical Perspectives

Security is a unique topic that differs significantly from others. For instance, if you want to learn about concept X in Python, there are countless resources available that are sufficient for your implementation and generally work well. However, the crucial question is: are they based on secure-by-design principles? The answer is often no.

The challenge in security lies not only in knowing how to do things correctly but also in understanding how systems can fail. This dual requirement makes the subject particularly complex. Let me quote a paragraph from *Cryptography Engineering*[1] book:

> As a colleague once told Bruce: "The world is full of bad security systems designed by people who have read Applied Cryptography."
>
> That makes this a very dangerous book. Some people will read this book, and then turn around and design a cryptographic algorithm or protocol. When they're finished, they'll have something that looks good to them, and maybe even works, but will it be secure? Maybe they'll get 70% right. If they're very lucky, they may get 90% right. But there is no prize for being almost right in cryptography. A security system is only as strong as its weakest link; to be secure, everything must be right. And that is something you simply can't learn from reading books.

Security, privacy, and anonymity are complex topics. *Bruce Schneier* says:

> Anyone, from the most clueless amateur to the best cryptographer, can create an algorithm that he himself can't break.

We often design systems in a way that gives us the illusion of security, but the truth is that no one can anticipate all potential problems. Practical implementations typically require extensive reading, including multiple books and papers, as well as years of experience. This is evident in the fact that NIST recommendations are developed by teams of experts rather than a single individual.

I do not claim to be an expert in practical applications, nor do many who assert such expertise. Therefore, I will not attempt to explain how to design specific systems. Most of these topics require hundreds of pages of reading and in-depth knowledge, which is beyond the scope of this book.

It is essential to verify anything you read. To support this, I strive to provide the reasoning behind my statements and cite relevant papers and books. I welcome any errors you may find or suggestions for improvement.

License

Donation

Why do you pay for ice cream? Because there are costs associated with producing the milk, as well as expenses for the people involved in the production process.

In the digital world, applications and content are also created by individuals who invest their time and effort. Therefore, using these resources naturally incurs a cost. Just because something isn't physical doesn't mean it doesn't have associated expenses.

Free services often rely on various revenue models:

- The provider values privacy and receives funding from organizations. In some countries, when you engage in nonprofit activities, you can request funding by stating that you are helping people for free and need financial support. Thus, you may receive a portion of public revenue or taxes.
- Advertising.
- Donations (which we discussed at the beginning of the book).
- Collecting and selling data to other companies. This is where we say, "If you don't pay for the product, you are the product!"

For example, you might say, "Hey, you who are providing free education are dedicating your time and effort to improve society. I appreciate your work and would like to contribute a certain amount to support you."

Alternatively, you might say, "Firefox, you are creating a free browser that respects our privacy and doesn't exploit our data like other browsers do. I see your work as beneficial to the community, so I want to donate a certain amount to you." Essentially, financial support for individuals is what we call donations.

Many free services around the world operate on this donation model. For instance, there are nonprofit organizations where volunteers pick up trash, plant trees, and so on. Their living expenses are also covered by donations and financial support from the public.

When you want to use Microsoft Office, such as Word, PowerPoint, or Excel, you have to pay for the software. The price for Microsoft Office 2021 for home use is over $100, which not everyone can afford! This is where an alternative comes in. Some individuals voluntarily create free and open-source software, so

if I need to open a file, I don't necessarily have to buy Microsoft Office; I can open that file for free. Additionally, these alternatives often strive to respect our privacy more! One such option is LibreOffice. A group of concerned individuals came together to create a program that allows us to open our files without cost.

Where does their income come from? Donations. They might say, "Did you enjoy using my program? Did you appreciate the good work I did? If you want, feel free to support us with whatever amount you can!"

Donations are one of the most noble forms of support because they make software accessible to everyone, allowing anyone to contribute financially as they are able. We need to learn to spend money on our digital needs just as we do in the physical world. After all, content creators and developers have their own lives and expenses. Just as you receive a salary, they also need to earn a living. If we don't support them, they may have to sell their programs, which could limit access for many people. They might even compromise their values and start collecting and selling our data.

NOTE

If you enjoy my writing, you can support me with your donation.[a] Donation really helps me continue my work and write more content. Thank You :)

[a] https://github.com/Mohammad-Kamal-mk/Books?tab=readme-ov-file#donate

1.0 Difference between Privacy, Anonymity, and Security

These three terms are often used interchangeably, but they have different meanings.

1.1 Security

In simple terms, security is like a lock on a door. It prevents unauthorized access to your data—meaning anyone you don't want to have access. Security ensures that your data is not tampered with or stolen. It protects your information from unauthorized access, disclosure, or modification. For example, you certainly wouldn't want your bank account data to be accessed or altered by someone else. Even if that person couldn't access the actual data, they could still damage it, making it unusable.

Mainly security is defined by three principles (CIA Triad):

1. **Confidentiality:** It ensures that the data is only accessible to authorized users.
2. **Integrity:** It ensures that the data is not tampered with.
3. **Availability:** It ensures that the data is available when needed.

1.1.1 Prevention, Detection and Response

Security consists of three steps: prevention, detection, and response.

Prevention is the first step in security. It involves implementing measures to prevent attacks/make it difficult from occurring in the first place. This can include using strong passwords, enabling two-factor authentication, not downloading untrusted apps, and not using unsafe VPNs.

An important note is that you always need to overestimate attacks. (It's not limited to just writing programs; it also includes protecting your data.)

There are certainly attacks you're not aware of. New attacks are discovered daily. Additionally, computational power is increasing rapidly. What was secure yesterday might not be secure today. For instance, in 1977, *Ron Rivest* (one of the inventors of the RSA) said[19] for factoring a 125-digit number, we need 40 quadrillion years of running power. Yet in 1994, researchers factored a 129-digit number![20]

Not only that, there are some groups and agencies that collect data now and attack later when they have enough computational power or when a new vulnerability is discovered.[1]

This is also effect our privacy. See

If you are a developer, you should use higher security parameters for the protocols you use because attackers

An approach is to have multiple layers of defense (Defense in Depth). In this way, if one layer is compromised, the attacker still needs to pass through the other layers.

Detection is the second step in security. It involves monitoring systems and networks for signs of attacks or breaches. This can include using intrusion detection systems, log analysis[2], and network monitoring tools.

You can use tamper-evident seals to detect if someone has tampered with your device. For example, if you have a laptop and you want to make sure no one has opened it, you can put a tamper-evident seal on it. If someone opens the laptop, the seal will break, and you will know that someone has tampered with it.

Some individuals position their office chairs in a specific way. If they notice that the chair has been moved, they immediately recognize that someone has entered their office.

Response is the third step in security, involving actions taken in reaction to an attack or breach. This may include isolating affected systems, notifying users, and searching for traces of the attackers.

Disaster recovery is an essential component of the response process. It involves having a plan in place to recover from an attack or breach. This can include restoring the system form backups. For instance, having a complete disk backup allows you to restore the system to a point in time—such as ten days ago—when it was free from viruses.

Finally, we should learn from our mistakes. After an attack, we should analyze what went wrong and how we can prevent it from happening again.

A robust protocol includes **all** three essential steps.

1.1.2 No System is 100% Safe

Systems are created by humans, and humans are not infallible. No one knows all the security issues, and no one can write code without flaws, which is why

[1] For instance, *Tempora* is a British surveillance program that taps into fiber-optic cables to collect data and store it for later analysis.

[2] Logs are files that record events that occur in a system or network. They can be used to track user activity, system performance, and security events.

all systems are hackable.
• even computers that have no external ports and no internet connection?!
→ Yes! The co-inventor of the RSA algorithm[3] himself demonstrated that we can use computer noise[4] to steal private data of RSA operations![2] He hacked one of the implementation of the algorithm that he himself created! (Algorithm was created by him. Not implementation) This illustrates the beauty of falsifiability in science, which will be discussed later.

• So, does that mean there's no security at all? Should we just give up and not do anything about security?
→ Well, that's exactly one of the common mistakes people make. They think, "Since everything is hackable, why bother?" That's a misconception! When we say everything is hackable, it means there's always a chance that someone with enough: • **Knowledge** • **Resources** • **Capabilities** • **Money** • **Time** can come along and hack it.
Almost all door locks can be opened. There's always someone who knows a specific method to unlock them. So, do you not lock your door at home? Why do you lock it then? Because you know that if it's left open, it's easy for someone to come in. But if I lock it, it protects me against a group of thieves. At the very least, a thief might think, "Why would I go to a place with a locked door? I'll go somewhere that's easier to break into, so I don't waste my time."

1. Some individuals lack technical knowledge and are simply nosy about your life, wanting to snoop on your phone. You can likely prevent these people from accessing your device with a good screen lock. (Specifically, if your password is a pattern you draw on the screen, it can be easily recognized after just one glance. Even after you set your phone aside, if you hold the black screen towards the light, the shape you drew remains visible due to the oils from your fingers, making it detectable again.)

2. There are individuals with some technical knowledge, limited money, and time, who know it's easy to boot up a live operating system and steal your data. For example, some people steal laptops and want to extract the information from them to sell later. With a good BIOS password (and by keeping your system and BIOS updated) and full-disk encryption, you can likely prevent them from accessing your data. If they want to bypass these protections, they will need time and money, which they may not have, or they might lack the knowledge to do so. (Didn't understand what these mean? No problem! We're just outlining things for now. We'll go into detail later!)

So far, with simple security measures, we've prevented the intrusion of two groups. These are the two groups that threaten us the most, and most daily threats come from them.

Don't you want to take some simple steps to prevent many everyday attacks? With just a few straightforward actions, you can stop them. You will definitely take these steps! This is the most basic and effective approach, and it prevents

[3] The algorithm behind HTTPS and many other protocols that keep you safe
[4] Any electronic device creates some noise. With enough analysis of the noise behavior, we can understand what is going on inside that electronic device, such as stealing passwords.

many attacks. It's like a lock; there's someone in the world who can open it, but you still lock the door! Those who say that no matter what you do, you'll get hacked, so don't bother, are giving completely wrong advice. Be careful not to fall for such statements!

But there's a third group as well!

3. Individuals, groups, or organizations with extensive knowledge, resources, capabilities, and significant funding; Organizations like the FBI fall into this category.

In this case, you can reduce the chances of them accessing your information, but you can't bring that risk down to zero! These groups are familiar with rare and hidden vulnerabilities, and they have the means to exploit them.

If your goal is to prevent access from this third group, you need to implement more robust security measures. Pay attention to the details and dig deeper!

1.1.3 The Weakest Link

Suppose you have fastened your bike with a lock and chain. If someone wants to steal your bike, they must break the lock or chain. Where does the chain break? At the weakest link (where it is vulnerable to attack). Similarly, in the digital world, security is only as strong as its weakest link. No matter how robust your security measures are, if there is a weak link, an attacker will exploit it. This is exactly what hackers do: they find the weakest link and exploit it to gain unauthorized access to your data. To ensure security, you must secure all the links, including hardware, operating systems, networks, applications, and users.

Like a broken window that thieves use to enter your house, a security vulnerability (or security bug) is a point that hackers exploit to gain access to your system. That's why we recommend that you update the software you use. Programmers regularly search for vulnerabilities and fix them through updates.

Have you ever noticed that whenever you update WhatsApp, the app doesn't change much? That's because WhatsApp mostly just fixes vulnerabilities rather than adding many new features. In contrast, Telegram frequently introduces new features.

Many times, bad actors gather information about what software and version you are using and then search for vulnerabilities in that software. If they find a vulnerability, they can exploit it to hack into your system.

Additionally, some use package managers to install software. Package managers are tools that help you install software on your system. However, some package managers don't provide updates in a short time. It's important to support package managers with our donations to help them provide updates faster.

In this book, I will provide you with the mindset of a hacker so that you can think like one and secure your data. I will start with the basics and mention some real-world examples to help you understand the concepts better. I will also explain how famous privacy tools work and how you can use them to secure your data.

1.1.4 The Importance of Transparency: Public Algorithms in Security Assurance

In security, we either identify a provably secure algorithm or method (protocols that have been proven to be secure), or we rely on a method that many experts have not found any vulnerabilities in and believe to be secure.

Falsifiability

Let's first discuss falsifiability, a concept introduced by Karl Popper. Especially in the context of the COVID-19 pandemic, there have been many odd claims. For instance, some people say, "Take this medicine; it's totally natural and will get rid of the virus." When you ask if they have tested it, they respond, "Yes, I gave it to some people, and they got better." They use these examples as evidence and proof. However, Popper argues that just because something seems to have worked doesn't mean it's true. It might be due to other factors, such as the person's immune system, a healthy diet, trust in the process, and a lack of worry[5].

So, Popper asserts that for something to be considered true, it must be falsifiable.

The beauty of science lies in its foundation of falsifiability. Scientists make hypotheses and then attempt to disprove them. If they can't disprove a hypothesis, it becomes stronger (but not an absolute truth). This is how science progresses.

• But science is unreliable; they change their minds all the time.

→ Yes, things change in science. However, science provides the best available solution at the moment. Science doesn't claim, "This is the truth; either accept it or die." Instead, science encourages questioning, testing, and attempts to disprove ideas. This is the beauty of science.

Science does not claim that examples serve as proof. This is because human nature tends to seek explanations for the phenomena it observes. Just as in ancient times, people believed that certain actions would anger the sea god (because they had once seen a flood occur after performing that action), countless superstitions and deities were created from this belief. Numerous gods were invented to explain the phenomena that humans observed.

The difference between scientific experiments and the claims of a seller is that, in drug development, researchers typically gather, for example, two thousand participants and divide them into two groups. They place both groups in identical environments, with the same diet and physical activity. One group receives the actual drug, while the other receives a placebo. If, for instance, 90% of the group that received the real drug improves, while only 5% of the placebo group does, this indicates that the drug has been effective in this limited sample.

[5] Many people believe that if they get sick, they will die. This belief can make them more anxious and stressed, which can weaken their immune system. On the other hand, if someone believes that they will get better, they are more likely to recover.

However, it must undergo further testing with larger groups. In reality, science never proves anything definitively. The humility of science lies in its flexibility in the face of experimentation.

Moreover, science has advanced significantly. Today, there are newer and more controlled methods than there were thirty years ago. Laboratory conditions are much more refined now, leading to more precise results.

Returning to the example of medicine, while it is true that many drugs are derived from natural substances, researchers examine which components are beneficial and which may be harmful. They isolate the beneficial compounds and formulate them into drugs, ensuring precise dosages, such as 5 milligrams of a specific ingredient. This is not the same as someone claiming to have created a remedy and suggesting that you take a spoonful every day without any scientific basis or testing to verify its effectiveness. Some medications are effective only up to a certain dosage; if taken in excess, they can even have adverse effects. For example, a drug may be beneficial at 5 milligrams, but beyond that, it may not only lose its effectiveness but also harm the body.

Falsifiability in Security

In security, the same principles apply. We primarily have to trust algorithms that have been tested over several years by many experts, who have not found any vulnerabilities in them (often, these are established standards). Regular testing is also crucial. Sometimes, companies claim they have tested their products and that they are secure, but they do not provide the results of these tests and do not conduct regular testing. Whenever you see a company claiming that they test their product, ask them whether they test regularly and when the last test was conducted.

Security does not come from obscurity; it comes from transparency. Additionally, the implementation of the algorithm is also important. We have seen many instances where companies used self-developed algorithms or their own implementations, which turned out to be vulnerable [3]:

> The design and implementation details should be well audited and reviewed by independent researchers and should not rely on the difficulty of reverse engineering proprietary systems

Bruce Schneier says:

> Anyone, from the most clueless amateur to the best cryptographer, can create an algorithm that he himself can't break.

Even if you are a good cryptographer/security expert, it doesn't mean that your algorithm is OK. It should be tested. For instance *Vincent Rijmen* and *Bart Preneel* about *McGuffin* (a cipher created by *Bruce Schneier*) say[4]:

> Modifying a scheme with only existing attacks in mind however is not a good design principle.

Bruce Schneier himself says[6]:

> Key schedules are very hard, and I didn't understand them very well in 1993 when I designed Blowfish. I'm much prouder of the key schedule in Twofish and Threefish.

When big companies like *Samsung* and *Qualcomm*, as well as prominent figures in cryptography like *Bruce Schneier*, can go wrong, how can you trust a proprietary algorithm?

Conclusion: Use well-audited algorithms. Don't change them, even if you think a modification would improve them. Simply don't! For this reason, we sometimes prefer older algorithms (as long as they are still secure, tested, and updated) over newer ones because they have been tested for a longer time. Untested algorithms are dangerous.

Reading List

- Snake Oil[7] (Highly recommended)
- An example of snake oil[8] (Highly recommended)
- Snake Oil Warning Signs: Encryption Software to Avoid[9]
- Desktop Google Finds Holes[10] (Cache of encrypted files, must be inaccessible)

1.1.5 Open source and It's Security

Imagine you've bought a car designed by the manufacturer in such a way that you can only drive it. There's no way to see the engine, the radiator, or any of the wiring—everything is enclosed in a metal box that you can't open. The only thing you can do is get in, drive around, and come back! You have no idea if quality parts were used or not!

On the other hand, there's another company that provides complete transparency. They don't hide the engine in a box! They say, 'Look, everything is clear. You can see everything!'

Behind every program, there's a set of code—a series of texts and instructions that tell the computer what to do. This code is like the engine of a car. If you can't see the code (in closed-source programs), you can't know what the program is doing. You can't know if it's secure or not. You can't know if it's doing something harmful or not. You can't know if it's collecting your data or not.

[6] https://www.schneier.com/blog/archives/2009/09/the_doghouse_cr.html
[7] https://www.schneier.com/crypto-gram/archives/1999/0215.html#snakeoil
[8] https://www.schneier.com/crypto-gram/archives/2003/0215.html#4
[9] https://web.archive.org/web/20030207174457/https://www.interhack.net/people/cmcurtin/snake-oil-faq.html
[10] https://www.schneier.com/blog/archives/2004/11/desktop_google.html

An important note is that this doesn't mean that closed-source programs cannot be reviewed. If you have an executable binary file, you can disassemble it to an assembly file to be able to analyze it. Ghidra[11] goes further by giving you a C file! Additionally, there are many tools to analyze what the program does.

Let's go back to that car we were discussing. Yes, you can see the car's parts and check if they're good or not, but there's a problem! If something goes wrong with the car, you have to wait for the company to come and fix it! You can't fix it yourself! If you try to fix it, the company can sue you! They might say, 'Sure, I let you see the parts, but you don't have the right to change them! You have to come to me to get them replaced!'

Moreover, they make you sign a lengthy contract before selling you the car, stating that you can only use it in a specific city. If you drive it on banned roads, they'll sue you. They impose a lot of conditions on you.

Now, you might be thinking, 'What do you mean?! I bought the car! I want to be able to repair it myself! What do you mean if I try to fix it, you'll sue me?! What do you mean I can't drive on certain roads?! I bought it!'

So, what can we do? Here's where the concept of Free Software comes in. Some people say this situation isn't right! We should allow everyone to modify their own software. What do you mean by imposing rules that prevent changing parts? They proposed that any software should have:

Freedom 0: The freedom to use the software for any purpose you want! This means you have the right to use the software for whatever reason you like! To clarify, regular software has specific rules set by the creator. For example, programs may state that you can only use them for personal and non-commercial purposes. If you want to use it for commercial purposes (like in your company), you have to pay to obtain the right to use it! Because you're using my program in a company that generates income, and my program has presumably helped your business, you should give me a portion of that money. For instance, $20 a year.

Freedom 1: The freedom to study the program, understand how it works, and change it so it does what you want. What does it mean to have the freedom to study a program to understand how it works? It means you can see the code! You can see the engine! Free Software states that the program's code must be accessible! To understand how a program works, I need access to its code so I can read it and see how it operates. Unlike open-source software, Free Software asserts that just having the source code isn't enough! I must also have the right to change it. Why?

I might want to modify a program to add a new feature. For example, a messaging app has voice calling but lacks video calling. I can change the code myself to add video calling! I can add a feature that I want! I can fix a bug that annoys me! I can remove a feature that I don't like! I can do whatever I want with the program! I have the freedom to change it!

Imagine there's a security vulnerability in a program. If you don't have the

[11] `https://github.com/NationalSecurityAgency/ghidra`

right to change and fix the program, you have to wait for the developer to address it whenever they feel like it, and they might not even care to fix it, leaving you at risk! That's really bad!
• But I don't have coding skills to fix it!
→ Yes, you might not, but others do! Others can fix it and provide you with the corrected version. This is exactly what we call Freedom 3, which we'll discuss in a moment.

Freedom 2: The freedom to redistribute copies so you can help others. In the licenses of most software, it's specified that no one should copy the program files and share them with others! It's not very clear, is it? Let me give you an example:

You can't take the program file, put it on a USB drive, and give it to your friend so they can install it too! You can't upload a program file to your website for others to download, even if that program is free! You can't send a program file to your friend on Telegram! But Free Software says, "What's the point of that? Let people copy the program files and share them with their friends! Let's not impose so many restrictions!"

Freedom 3: The freedom to distribute copies of your modified versions to others. This is the point we discussed earlier. If there's a problem and the company doesn't fix it, others have the right to fix it and share it with the public!

For instance, Telegram has a lot of features. There are also many unofficial versions of Telegram with various functionalities. In fact, Telegram is free software, which means anyone can work on improving it. That's why Telegram has so many features. For example, if someone notices that this messaging app has voice calling but lacks video calling, they can modify the code to add video calling! This way, an unofficial version of Telegram is created. Another person might say, "Oh, it has video calling, but it doesn't support group calls," so they add group calling. Yet another person might think, "It has this feature, but it's missing that one," and they add it too.

In essence, people create and release new versions. They can also share their code with the original developer, who might incorporate it into the main Telegram, or they can choose not to and create a new, unofficial version to publish.

In contrast, we have WhatsApp, which doesn't operate this way at all. It has very limited features, and no one can improve WhatsApp on their own! This highlights the difference between free software and closed software that imposes countless restrictions on you.

In fact, the existence of the source code is essential for all freedoms 1 to 3.

Now, you should understand the difference between Free Software and Freeware. Freeware simply refers to a program that is free of charge. However, Free Software means software that is truly free, encompassing the freedoms to use, study, modify, and distribute the software.

To understand the concept, you should think of "free" as in "free

> speech," not as in "free beer." We sometimes call it "libre software," borrowing the French or Spanish word for "free" as in freedom, to show we do not mean the software is gratis.[12]

The above quote really explains the concept well.

- If everyone can modify the program, doesn't that put security at risk?

→ while everyone can modify and distribute the program, you have the option to use the original version from the original developer if you're not confident in the security of modified versions.

1.1.6 Attack Surface

Entering a house with 10 doors is easier than entering a house with only one door. The more doors you have, the more chances you have to be attacked. Programs are based on code. The more code you have, the greater the chances that a vulnerability exists.

Many security-focused apps reduce the attack surface by removing unnecessary features. For example, a browser that doesn't have a camera feature is more secure than a browser that does. The camera feature might have a vulnerability that can be exploited by an attacker. This principle can be extended to any app (including messaging apps). For instance, I personally don't recommend Telegram for someone who wants to be anonymous because it has many, many features.

Is editing a book with 1000 pages easier than editing a book with 100 pages? The more lines of code your app has, the greater the chances that vulnerabilities won't be detected by developers. We mostly choose programs and protocols that have a smaller amount of code. For example, *WireGuard* is a VPN protocol that has only around 7000 lines of code. This is much less than other VPN protocols, which is an advantage.

As another example, we can mention the "enable editing" feature in Microsoft Office. If you enable it, an attacker can run a macro and execute code. This results in a security risk, so it's better to disable it.

Many people install numerous dependencies and IDE extensions in their projects and don't remove the ones they don't use. Keep a list of dependencies and remove the ones you don't need. This reduces the attack surface.

In fact, some bad actors during job interviews ask you to do a live project and provide you with a project that has malicious dependencies to hack into your system. Additionally, some give you a project with legitimate but outdated dependencies. They want to know whether the project works on your computer (if you are using an old dependency) and then hack into your system based on vulnerabilities that exist in that old dependency.

Remove dead code (code that is no longer used). It can be a security risk.

[12] `https://www.gnu.org/philosophy/free-sw.html`, Creative Commons Attribution-NoDerivatives 4.0 International License

Open-source Does NOT Mean Free of Bugs

First of all, how can you be sure that the code provided is the same as the code of the executable file? Some provide a way to check it[13]. But I bet many of you have never checked it. I understand that checking it mostly requires technical knowledge and dependencies that you might not have.

Moreover, there was a problem in *OpenSSL* (a library that is used to secure the connection between your browser and the server and is used by millions of people), and nobody discovered it for years. This is an example that even big projects can have bugs.

• Why?

→ Well, nothing is 100% perfect! Sometimes, issues slip through the cracks of the people reviewing the code! No one has read all of the code of a program like *OpenSSL*! It's a huge project!

Additionally, many free software programs rely on libraries (pieces of code written by others that we can use instead of coding everything from scratch, like code to determine if a number is prime or not).

• What does that have to do with it?

→ I want you to see the connection. Think about it: okay, the product itself might not have any issues, but there could be problems in its supply chain... Many times, problems arise from the supply chain. This means that the developer of the program might be using a service, and that service uses libraries that are no longer being updated! The library developer may have abandoned it, leaving it without updates; when that happens, our program is using outdated code that may have security vulnerabilities or bugs that won't be fixed!

Actually, *supply chain attacks* are a serious issue. For example, the *SolarWinds* attack was a supply chain attack. The attackers didn't attack the target directly. Instead, they attacked a company that provided services to the target. The attackers compromised the software update mechanism of the company, and when the company sent an update to the target, the update contained malware. The target installed the update, and the attackers gained access to the target's network. This is a supply chain attack. The attackers didn't attack the target directly; they attacked a company that provided services to the target. This is a very serious issue.

A bad actor can gain trust in the community by contributing to open-source projects. They can contribute to the project for a long time, and when they gain the community's trust, they can insert malicious code (e.g., a backdoor) into the project. This is why it's important to review the code of the libraries you use. You can't just trust them blindly. You have to review them and make sure they're secure.

Let's assume the libraries are being updated. The developer of our program might not care about using the updated libraries. They might still be using older versions that have security issues!

[13] e.g. Telegram: `https://core.telegram.org/reproducible-builds`

- Why?
  - Many open-source projects are written for fun and to learn.[14]
  - A large number of developers are unaware of security issues and may not realize how important they can be! So when they're not aware of their significance, they don't prioritize updating the libraries!
  - They might simply lack the motivation to update the libraries. They might think, "Why should I update the library? It's working fine!"
  - The new versions of the libraries might come with new features and changes that our developer doesn't know how to work with, which is why they don't update!
  - Updating that library might cause the existing code to break, requiring several changes in the code. And the developer might not want to do that!

What to do? → Use libraries that are actively maintained and updated. Libraries that are made by real experts in the field, not just written for a fun project.

Typically, large companies use free libraries but don't provide any benefits to the creators. Essentially, these companies are profiting from someone else's work without giving any share of their earnings to the original developers. It would be better if the law required them to give a portion of their profits to the creators of those libraries, ensuring that the developers are financially supported and can dedicate more time and motivation to building their projects!

One issue that arises in free software is that developers may become discouraged by the fact that so many different entities are using their code without any compensation. After all, they need an income to sustain their lives! Eventually, they may get tired and abandon their projects. However, if they receive some income, they will be more motivated, spend more time on their work, and produce higher-quality code instead of just throwing together something mediocre!

Moreover, companies that use this code should make an effort to review it and not just be passive consumers! Unfortunately, many companies simply use the code without examining it. If every company were to review the code they use, the situation would change significantly. This way, many experts would be monitoring and reviewing the code, and the security of open-source software relies on the fact that many people check the code. Otherwise, if everyone is just a user and doesn't check the code, it doesn't enhance security!

It might be funny, but we have open-source malware. They update it after anti-viruses detect it. They update it to bypass the anti-viruses.

[14] While companies have a security team to review the code, many open-source projects don't have a security team.

Reading List

- Choosing Secure Open Source Package[15][16] (From Intel Open Source. Highly recommended!)
- The Cathedral and the Bazaar[17]

The section 'VERY OLD CODE' explains why we should avoid using very old and unmaintained code. I'd like to add a point: for example, if I wrote a library back in 2000, there were certain types of attacks that either didn't exist at that time or were so impractical that they were unlikely to occur. As a result, I didn't implement any defenses against them. However, now those attacks have become viable and could easily exploit vulnerabilities in the code.

It doesn't matter if it was written by the best cryptographer in the world or if it used the best encryption methods available at the time. What's important is that the world has changed. What was once theoretical or impractical has now been discovered to be feasible![18]

1.1.7 Don't Rely on Trial and Error When Creating Services and Troubleshooting Computer Errors!

Imagine you are trying to set up a program. During the installation, you encounter an error. You go online to search for a solution, and a website or video suggests that you run a specific command to fix the issue. Without fully understanding the implications, you blindly follow that advice. However, instead of resolving the problem, you encounter another error that states you need to run the program with admin access (the highest level of permission). Once again, you proceed without realizing that this could be potentially harmful.

In each instance, you are trying solutions that may either be safe or malicious. It's like a house where, at every step, you are opening different doors and windows that could later be exploited.

Instead of relying on blind trial and error, it's better to seek out reputable sources and official documentation. This approach will help you resolve issues effectively and safely while also preventing potential risks.

If you're a developer, you should write clear and detailed documentation for your software. This documentation should explain how to use the software, how to troubleshoot common issues, and how to fix errors. By providing this information, you can help users avoid potential risks and ensure they have a positive experience with your software.

[15] `https://01.org/blogs/terrioda/2017/choosing-secure-open-source-packages-part-1`
[16] `https://01.org/blogs/terrioda/2017/choosing-secure-open-source-packages-part-2`
[17] `https://en.wikipedia.org/wiki/The_Cathedral_and_the_Bazaar`
[18] For example, *differential cryptoanalysis* was previously unknown to the academic world while it was known to the NSA: `https://www.schneier.com/blog/archives/2013/09/the_nsas_crypto_1.html`

Warning

Never do something you're not aware about its consequences. Unfortunately, many people blindly follow instructions found online, assuming that just because a method works or has received positive comments from others[a], doesn't mean it is safe.

[a]These comments may be fake or even written by a malware infected the user

1.1.8 Be Cautious of Hacking and Security Groups

I know many of you are eager to join hacking and security groups to learn more about hacking and security topics. However, it's crucial to be aware that 90% of hacking and security groups on platforms like Telegram do not provide you with valuable knowledge. A large portion of these groups focuses on cracking software and distributing pirated programs that have been cracked by Russian hackers, along with sharing cracked accounts and engaging in illegal activities. Moreover, individuals with substantial knowledge typically do not participate in these groups.

It's important to understand that many of the people in these groups may know more than you do, and among them could be cybercriminals, such as ransomware developers. Therefore, avoid downloading any files from these sources, including PDFs! Yes, even PDFs can be infected, and we will discuss this later. So, exercise caution. I don't want to scare you, but at the very least, you should be aware of malware and the different types of files.

Also, be very careful of individuals looking for hackers. Why? Because these people often say things like, "Come here and do this for us; we have your back," or "The boss is watching out for you." You might end up doing hacking work for them, but when things go wrong, you will be the one held responsible, and they will take no accountability. So, stay away from those seeking hackers. You might find yourself in trouble with international law enforcement or the FBI, facing travel bans and more.

Avoid Getting Involved in Government Hacking Games

The beginning of their friendship might seem friendly. For example, they might pay for your purchases at a store or say, "Don't worry about the bill; we've got you covered!" They try to appear very friendly, saying things like, "Why can't people be kind to each other? Why is the bond between countries so weak? We should all be friends." While their statements may have some truth, the intention behind them is to gain your trust and present a false image of being good people, which is concerning.

In some countries, students may not have the right to work or may have limited working hours, such as being allowed to work only 20 hours a week. However, under these circumstances, you may need money to cover your expenses. They might say, "We've got your back; we'll give you work, but keep it

quiet." For example, they might ask you to write something for them, assuring you that they won't disclose you as the author. They may even say, "We just need you to translate a text from Chinese to English," but in the process, they will ask you to add sensitive information that is crucial to their agenda. Read about *Game of Pawns from* the FBI.[19].

This path has no way back!

Once you are involved in such activities, you will be unable to escape. They will have evidence against you, and you will be forced to comply with their demands.

1.1.9 The Sofware Industry Needs More Secure Products, Not More Security Products–CISA

Never design a system that requires user action for security. Most people still use the default browser, and many don't even know there are other browsers out there! Do you really want to rely on the security of the system being in their hands? No! Never do it! Security and privacy must be by default. As CISA in *Shifting the Balance of Cybersecurity Risk: Principles and Approaches for Secure by Design Software*[20] suggests:

> The complexity of security configuration should not be a customer problem. Organizational IT staff are frequently overloaded with security and operational responsibilities, thus resulting in limited time to understand and implement the security implications and mitigations required for a robust cybersecurity posture.

Additionally, you must not trust user input. Always check whatever the user does on the server side[21]. Many security problems occur because of trusting user inputs.

For instance, some password managers (an app that stores passwords securely) don't encrypt any fields other than passwords. This is a huge risk. Many people store backup codes for 2FA (Two-Factor Authentication) in the notes field of the password manager[22]. Some also store security questions there. You should not have pre-assumptions about what users will do. You always need to think like this: "I have designed it like this. What if the user does something else?"

Secure-by-design also means I should have options to disable features I don't use. By doing this, I reduce my attack surface.

• I have hardening guide. I don't need secure-by-design practices!

→ CISA gives a good answer:

[19] `https://www.fbi.gov/video-repository/newss-game-of-pawns/view`

[20] `https://www.cisa.gov/resources-tools/resources/secure-by-design`

[21] Many client-side checks can be bypassed

[22] Which is wrong. Do you know why? Because if the password manager gets compromised, 2FA becomes lossless: `https://keepassxc.org/docs/KeePassXC_UserGuide#_adding_totp_to_an_entry`

> Hardening guides suffer from several common problems Some hardening guides are hard to find and are not well supported Others are complex to implement, occasionally requiring software development to write an extension module Still, others assume the reader has extensive cybersecurity experience to understand the ways in which various settings change the attack surface Practitioners who have an incomplete understanding of the ways in which attackers work may fail to properly implement hardening guide instructions, especially if the instructions do not make the trade offs clear. Further, not all hardening guides are written by engineers who are intimately familiar with attacker tactics and economics, causing them to create hardening guides that are inefective even if faithfully implemented Millions of customers are taking on the responsibility to harden multiple instances of software or systems, often in resource constrained environments Relying on hardening guides simply does not scale

Additionally CISA says:

> Similar to seat belt chimes in cars that continuously make noise when seat belts are not fastened, manufacturers should implement timely and repeated alerts when users or admins are in truly unsafe states.[23]

1.1.10 Downgrade Attacks

Imagine you enter Iran; conversations typically start with greetings and handshakes. In Japan, for example, people begin conversations by nodding their heads. Essentially, the protocol (or agreement) for conversation in Iran is a greeting and handshake—it's a set of rules for interaction.

Computers are machines that require everything to be clearly defined for them. They need to know what to do in specific situations. For two computers to communicate with each other over a VPN connection, they must speak based on a shared discourse and agreement that allows them to understand each other. This method of communication is referred to as a *protocol*.

For example, the process of securing HTTP and converting it to HTTPS works like this: first, you send a message to the website saying, "Hello, I support version 1.2 and 1.3 of the security protocols with these specific algorithms." Then, the server responds with a greeting and says, "Okay, from what you mentioned, I will choose this option. From now on, let's communicate based on the agreement we've made!"

For that reason, in order for older systems to be able to communicate with newer systems, they must support the same protocol. For this reason, most systems support multiple versions of the same protocol. For example, a server might support both TLS 1.2 and TLS 1.3. When a client connects to the server, the server and the client negotiate the version of the protocol to use. Mostly, one

[23] e.g. not having 2FA

party sends a message to the other party saying, "I support these versions of the protocol." The other party choose the highest version that both parties support. This is called *version negotiation*. For example, if the ony party supports TLS 1.2 and 1.3 and the other party only supports 1.2, 1.2 will be chosen.

However, this negotiation mostly happens when the connection is not yet secure (because the client and the server don't know each other yet). This means that an attacker can intercept the communication between the client and the server and modify the messages to make it appear as though the newer version of the protocol is not supported. The attacker can then force the client and the server to use an older, less secure version of the protocol that has known vulnerabilities that can be exploited. This is called a *downgrade attack*.

One approach is to drop support for older versions of the protocol. This way, the attacker cannot force the client and the server to use an older version of the protocol. However, this approach can cause compatibility issues with older systems that do not support the newer versions of the protocol. Another approach is to implement mechanisms that detect and prevent downgrade attacks.

As CISA — NSA — FBI — ACSC — CCCS — CERT NZ — NCSC-NZ — NCSC-UK — BSI — NCSC-NL NCSC-NO — NÚKIB — INCD — KISA — NISC-JP — JPCERT/CC — CSA — CSIRTAMERICAS suggest[24]:

> Prioritize security over backwards compatibility, empowering security teams to remove insecure features even if it means causing breaking changes.

Some softwares drop the support for older versions (the software stops working) to protect users from downgrading attacks.

1.1.11 Social Engineering

Social engineering is the art of manipulating and controlling people. It is a psychological manipulation technique that exploits human error to gain access to systems, networks, or physical locations.

Have you ever noticed that some people, when they speak, seem to have you under their control, making you feel almost enchanted, doing whatever they want? You might agree with what they say without realizing that you are under their influence. This is because they have information about you and know how to behave.

Many high-impact attacks on companies are related to social engineering. For example, I can call the IT department and say, "I am from the head office. Why is our system always broken? Can't you do your work correctly? We always have problems with our system. I am tired of this. I want you to fix it now! I am going to report you to the CEO!" The IT department might be scared and do whatever I want. This is a simple example of social engineering.

Alternatively, I can go to my SIM card provider in a Mercedes and wearing a suit and say, "My phone has been stolen. I am waiting for a very important call.

[24] `https://www.cisa.gov/resources-tools/resources/secure-by-design`

Will you please forward my calls to this number?" Based on my appearance, they might think I am a legitimate person and do whatever I want. This is another example of social engineering.

Social engineering is a very powerful tool. It is much easier to exploit a person's natural inclination to trust than to find ways to hack into a system. Kevin Mitnick[25] is a prime example of a social engineer. He was able to gain access to computer systems simply by talking to people and convincing them to give him the information he needed.

We will later discuss why privacy matters and how the data you leak can be used to manipulate you.

1.2 Privacy

In many cases, privacy is confused with security. Let me give you an example to illustrate the Difference.

Imagine you are trapped in a cage. There is food, water, and exercise equipment available. There are a hundred guards surrounding you, along with a thousand surveillance cameras, and a hundred people are monitoring everything. In this situation, you are safe and not in any danger, but you have no privacy! Whatever you do, the hundred guards are watching you! Do you like it?

1.2.1 Why Privacy Matters?

This topic is highly controversial. When we talk about using strong passwords, avoiding unsafe VPNs, and not downloading untrusted apps, people often say, "'Come on, take it easy! I have nothing to hide!"

This mindset comes from the fact that people don't have a realistic understanding of the digital world.

- Let me ask you something. Do you like someone to watch your phone while you are using it on the bus?

→ Of course not! Even if you are not doing anything wrong, you don't want someone to watch your phone.

- Do you want someone to always be with you and watch everything you do?

→ If you do, go and find a random person and ask them to watch you all the time for the rest of your life.

- But I don't know him. The person could do something bad to me. Additionally, for the whole life is too much.

→ Exactly! So tell me, do you really know the people who are watching you online? Do you know that anything you do online is permanent forever?

We know the physical world, and we've been told to lock our cars because if we don't, someone might steal them. We see the absence of the car and know that

[25] `https://en.wikipedia.org/wiki/Kevin_Mitnick`

it's stolen. But in the digital world, you don't see the absence of your data. You don't know that your data is stolen. You don't know that your data is being used against you. You don't know that your data is being sold to the highest bidder. You don't know that your data is being used to manipulate you. You don't know that your data is being used to make you do things you don't want to do. You don't know that your data is being used to ruin your life!

Just like kids who ask why we should lock the doors—there are many other houses around us. Why would someone steal our house? They are much richer than us. People also ask why we should secure our data—there are many other people around us. Why would someone steal our data? They are much richer than us.

Yes, they might be richer, but you are still valuable. Furthermore, thieves mostly target the easiest targets. If you don't lock your door, you become the easiest target. If you don't secure your data, you are the easiest target. We should be taught to secure our data just like we are taught to lock our doors.

I have read a really good ELI5 (Explain Like I'm 5) explanation of privacy on *Reddit*:

> Everybody poops, but we still close the bathroom door. [26]

Are we doing something wrong in the bathroom? No. We need space to do our private things. We need privacy.

Another good quote from *Tor Project*:

> Privacy isn't about hiding bad things. It's about protecting what makes us humans: our day-to-day behavior, our personality, our fears, our relationships, and our vulnerabilities. Everyone deserves privacy online. [27]

Another good quote from *Bruce Schneier*:

> Privacy is an inherent human right, and a requirement for maintaining the human condition with dignity and respect

Surveillance changes behavior

Think about being in a restaurant. Do you choose a table in the center of the restaurant or a table in the corner? Most people choose the table in the corner. Why? Because they want privacy. They don't want everyone to watch them while they are eating. Is eating a bad thing?

Have you ever encountered a situation where you perform worse when you are being watched? For example, you might be playing a game and doing well. Then someone comes and watches you, and you start to do worse. Why? Because you are being watched. You are being judged. You are being evaluated.

People's behavior changes when they know they are being watched. They become more cautious. They become more self-conscious. They become more

[26] Mockingbird, Comment on 'How do you counter the "I have nothing to hide?" argument?', Reddit, August 22, 2015, `https://www.reddit.com/r/privacy/comments/3hynvp/comment/cubsirr`

[27] https://twitter.com/torproject/status/1450488158416687107

anxious. They become more stressed. They become more paranoid. They become more fearful.

In the Digital World, Anything Might Be Permanent

Keep in mind that anything that has gone into the digital world might be permanent. A simple search you make might manifest later when you become someone important. Are you okay with that?!

Have you ever read your past messages and thought, "What was I thinking?" The world is changing rapidly. For example, in very ancient times, slavery existed and was a common and normal thing. It was completely normal to have slaves. But now we say that slavery is wrong. Do we go back to the past and criticize those who had slaves? No! Because that was the norm of that time and the general understanding of society then. If someone didn't treat their slave harshly, we actually commend them. It's the same now, with the difference that society is changing quickly.

For instance, one of the tactics used to eliminate a competitor—whether in business, elections, or other areas—is to say, "You said this in the past! Shame on you!"

Nobody knows whether you might become an important person in the future. Nobody knows whether you might become a target in the future.

While that statement might have been acceptable at the time, the situation has changed now. It's also possible that the person didn't have a good understanding of the issue back then but has since learned and changed their mind. Yet, they are still judged based on their past. Did everyone believe in social equality from the very beginning? People grow up in different environments and have different experiences.

Unfortunately, society does not provide an opportunity for defense. The media can lead to someone being fired from their job without waiting for a response from the individual or proof in court. This is why privacy is important. People should have the right to change their minds, to learn, to grow, to make mistakes, and to be forgiven. People should have the right to be human.

Warning

Anything that goes into a digital device might be revealed to the public in the future. In a world where information is permanent, just one mistake can lead to irreversible consequences.

But what if you no longer use a service? Or you want to sell your SIM card? Is there any way to delete your data from an app? Fortunately Yes! Some services provide you with the ability to delete your data. This is called *Right to be Forgotten (RTBF)*. But keep in mind that data might still be stored in backups, shared with third parties, swap files[28], hibernate files, temporary files in RAM, etc. Furthermore, they might have used it to train their AI. Additionally, note that not all "delete account" buttons actually delete your

[28] Of course, this is not used for everyone but rather targeted attacks

data; they might still keep your data. That being said, I personally recommend you to edit your data before deleting it. However, this is not a guarantee that your data will be replaced with the new data in backups and logs.

For example, I was arrested on charges of fraud. All the news outlets headline that this person has been caught as a fraudster. But then I go to court, and it turns out there was a mistake, and I am innocent.

So, where's the problem? The issue is that anyone who searches my name online will find that I am labeled as a fraudster. A thousand headlines calling me a fraudster pop up. Now, how do I fix this?

I could go to Google and say, "Look, here's the court ruling. I am innocent." Or I could approach the news agencies and ask them to reconsider, saying, "Look, I am innocent. Please take down your article."

This is where RTBF comes into play.[29] It allows me to request the removal of that information from search engines and news outlets.

I have the right not to share everything

Have you heard that people say not to ask these questions:

- When do you want to get married?
- When do you want to have children?
- How much do you earn?
- Why are you getting leaner?
- Why are you getting fatter?

This is because these questions are related to privacy. People might not want to share everything about their lives. They might want to keep some things private. I have the right not to share my diary with you. It's not that you are necessarily a bad person; it's that I might want to keep some specific things private.

You don't need to have bad things to hide to want privacy. You need privacy because you are human! Or, according to *Edward Snowden*:

> Arguing that you don't care about the right to privacy because you have nothing to hide is no different than saying you don't care about free speech because you have nothing to say.

Read-world Scenarios

Did you know that the NSA (National Security Agency) was collecting data on everyone? They were collecting data on everyone, even if you were not a criminal, and this was against the law.

Why I Mostly Talk About the NSA?

In this book, I mostly talk about the NSA because they are the most famous and

[29] See also: `https://gdpr.eu/right-to-be-forgotten/`, `https://gdpr.eu/article-17-right-to-be-forgotten/`

have the most resources. They are the most powerful.[30] There is real evidence that they have been collecting data on everyone. Furthermore, there are limited resources about other countries. But keep in mind that every country has an intelligence agency, and they are doing similar things or even worse things.

1.2.2 Story of the Government

There is a story that the government is actually my servant because we humans have historically tried to divide our tasks. For example, a person with high physical ability was assigned the responsibility of protecting others. We appointed someone to be responsible for cooking. We divided responsibilities. The modern government is the same. The police are actually my employees; I pay them to protect me.

This leads to the following law:

1.2.3 Freedom of Information Act (FOIA)

The Freedom of Information Act (FOIA) is a law that gives you the right to access information from the federal government. It is often described as the law that keeps citizens informed about their government. Under the FOIA, agencies must disclose any information requested unless it falls under one of nine exemptions that protect interests such as personal privacy, national security, and law enforcement.

Why? Because you are paying them! You have the right to know what they're doing with your money. This might seem strange in some countries because the government is seen as a higher authority. But in reality, the government is your servant. You are the boss.

A clear example of this is the request to the FBI regarding what data it can collect from encrypted messaging services through law enforcement actions.[31]

Or, according to *Jadi Mirmirani*:

> I must have the right to get accessed to the information of things that are done with my money. For example I might want to create a program that helps people to find nearest trash cans based on their location. I need to have the data of the trash cans.

FOIA Further helps us to disclose other real-world Scenarios.

1.2.4 Who Watches the Watchers?

Agencies are watching us. But who watches the agencies? Who watches them to ensure they are not doing bad things? To ensure they are not using their

[30] They knew about some attacks before academics knew about them. For instance, they knew *differential cryptoanalysis* twenty years before Adi Shamir found it.

[31] `https://web.archive.org/web/20230108195447/https://propertyofthepeople.org/document-detail/?doc-id=21114562`

power against those they don't like? To ensure they are not using their power to manipulate people? To ensure they are not creating a dictatorship?

An agency might be good now, but is there any guarantee that it will be good in the future?

This is why police officers have body cameras.[32] Without body cameras, police officers can act as they wish. They can extort money from people. They can beat people. Subsequently, they can say that the person was resisting arrest. They can say that the person was attacking them, etc.

1.2.5 It's Not All About Security Agencies

Many companies do collect data on you.

1.2.6 Analysis on Data & Data is Power

Why should we be careful about data? Even if the data is not sensitive, it might be used against you. Analysis makes it possible to know more about your past and predict or even change your future. When we talk about collecting data, we don't mention only the data that you have shared; we also mention the data that has been collected about you. This data might be collected unintentionally. It might be collected from other people you're connected with. Or you might have accepted it through the tick you marked to accept the terms and conditions. Or you might have accepted the cookies.

When we talk about analyzing or monitoring data, we don't mean someone is sitting and watching you. We mean that the data is being processed by a computer. There is a huge list of tools that can search for you around the internet with just one click.[33] With a huge amount of data, computers are able to know more about you and possibly change your behavior.

For example, there is a famous story that says those who buy milk in large bottles likely have ancestors who experienced famine. They do this to ensure they always have a supply on hand, driven by their subconscious. Or see *Epigenetic signatures of intergenerational exposure to violence in three generations of Syrian refugees*[5]. Or *Pizza Meter* story[34]

Have you noticed that when you search for something or watch videos related to a topic, you start to see ads about that thing? This is because the data is being analyzed and used to show you ads that you might be interested in. This is called targeted advertising. This is not bad in itself, but it might be used to manipulate you.

As this is possible, many more things can happen with your data. It can be used by bad people for their own benefit. If you know how a person behaves, you can control them. Have you ever noticed that some people, when they speak,

[32] Read more: `https://nij.ojp.gov/topics/articles/body-worn-cameras-what-evidence-tells-us`

[33] `https://github.com/The-Osint-Toolbox/Email-Username-OSINT`

[34] A sudden increase in pizza orders a from Domino's by U.S. government offices, prior to major events like the invasion of Iraq into Kuwait, etc. See more at: `https://tradesmith.com/the-pizza-meter-a-seven-figure-retirement-and-you/`

seem to have you under their control, and you feel almost enchanted, doing whatever they want? You might agree with what they say without realizing that you are under their control. This is because they have information about you and know how to behave.

Of course, this is not common in daily life. But in the digital world, this is made possible by analyzing your data.

With large-scale data collection, they can change your future. For example, they know how to give you ads in order to convince you to vote for a specific person. For instance, they know you like cats. They can create fake news about a person who is against cats. They can show you that news and make you hate that person. They may promote a certain individual in silence, embedded within the texts they publish without you even noticing [6].

Challenging Yet Essential: The Value of Upholding Small Privacy Rights%

Protecting privacy can be challenging and comes with limitations, but failing to do so has even heavier consequences. When you don't defend your basic and small rights, you won't be able to stand up for larger rights later on! If you don't resist companies collecting data, it becomes normal for all companies to gather data. Each time, the next company will collect even more data for its own advancement, and you won't be able to question it. In the end, they will say, "Don't want to? Then don't use this app!" This means you won't be able to do anything about it later! Ultimately, all companies and agencies will say, "Well, if you don't want to, then don't use it!"

People often realize the value of something only when they lose it or find themselves in a difficult situation. Like people who have experienced war, they know the value of peace. You won't realize the value of privacy until you lose it.

So, defend your small rights, even if it doesn't seem to harm you personally. Because it harms others! Eventually, the world will become so bad that you will feel the impact too!

There is a really beautiful poem about this in Persian:

> small stream can be blocked by a shovel. But when it becomes a river, you can't even pass through it with an elephant.[35]

So, don't let the small things go. Because they will become big things later on.

They lye to you

Surely, no government or agency would tell you that they are collecting your data for a dictatorship. They always say they are collecting your data for your safety. They want to catch terrorists. They have to catch criminals. See also (How they manipulate): [7] [36]

[35] Saadi

[36] https://www.reuters.com/world/china/china-bank-protest-stopped-by-health-codes-turning-red-depositors-

Once senator *Ron Wyden* asked *James Clipper*: → Does the NSA collect any type of data at all on millions or hundreds of millions of Americans?

• No sir.
→ It does not?

• Not wittingly. There are cases where they could inadvertently, perhaps, collect, but not wittingly.[8]
He later said: "I responded in what I thought was the most truthful, or least untruthful manner by saying no" This is the reason why Edward Snowden leaded the collection of data:

> I would say sort of the breaking point is seeing the Director of National Intelligence, James Clapper, directly lie under oath to Congress. [37]

President Obama said:

> Nobody is listening to your telephone calls. [38]

Let me respond to this with a quote by Stewart Baker (former NSA director):

> Metadata absolutely tells you everything about somebody's life. If you have enough metadata you don't really need content

1.2.7 The Double-Edged Sword of Monitoring and Collecting Data

Having data is dangerous. How can we be sure that employees of the company don't use the data for their own benefit?
• No, we regularly check our employees.
→ How can we be sure you actually do this correctly?! You checked Edward Snowden! But what happened? He leaked the data about your data collection!
→ How can we be sure that a mistake won't happen and the information won't get leaked? We've seen many times that an employee from a security organization took confidential information home, and that's how the information got exposed![39]
→ Let's assume all the employees are good and won't make any mistakes (which is not possible). How can we be sure that nobody from outside would pressure them and threaten their families to get the data?
→ How can we be sure that the agency or company won't get hacked? At the very beginning, we said any system can be hacked! No matter how secure it is!

Nobody cares about you. They want money.

[37] Snowden-Interview: Transcript: `https://web.archive.org/web/20140131063748/https://www.ndr.de/ratgeber/netzwelt/snowden277_page-2.html`
[38] Obama: Nobody is listening to your calls `https://youtu.be/zOkHtzHJEZ0?si=kqHvxx2LVQAE-I84`
[39] `https://www.schneier.com/blog/archives/2017/10/yet_another_rus.html`

1.2.8 The Chain Reaction of Backdoors: How one Backdoor can Lead to Many

Imagine you have a house with a front door that is locked with a secure and strong lock. But you also have a door at the back of the house that is not locked. The door is hidden in the back, and nobody knows about it. But what if a thief finds it? He can easily enter your house.

Let me give you another example: You have a safe with a strong lock. But you also have a hidden key under the safe. Nobody knows about it. But what if a thief finds it? He can easily open the safe.

Definition 1. *Backdoor: A backdoor is a method, often secret and hidden, to bypass security mechanisms. It can be used to access a computer system, a mobile device, or a software program.*

One of the security agencies' ideas is that companies should implement a backdoor in their software to allow the agencies to access the data to catch criminals.

The problem here is that nobody can guarantee that the backdoor won't be found by bad people. If a backdoor is found by bad people, they can access the data and use it for their own benefit. This is why many people are against the idea of implementing a backdoor.

The same goes for security based on secrecy. If the security of a system is based on secrecy, as soon as the secret is revealed, the security is gone.

Also, if a country decides to force a company to implement a backdoor, other countries might do the same. They will say, "If you want to sell your software in our country, you have to implement a backdoor for us too. If you don't, we will ban your software for our citizens." This will lead to a situation where all companies have to implement a backdoor for all countries. This is a nightmare for privacy. [40]

This is the exact idea of the conversation between Alex Stamos (CSO of Yahoo) and the director of the NSA [9].

Backdoors can't bring national security; they bring insecurity!

We have to have both privacy and security at the same time. *Benjamin Franklin* once said:

> They who can give up essential Liberty to obtain a little temporary Safety, deserve neither Liberty nor Safety.

1.2.9 The Paradox of Data Overload: Valuable Insights Lost in the Noise

Useful information gets lost among all the other data. Organizations like the NSA collect so much information that valuable insights become buried within

[40] A little related -¿ Special Report: Amazon partnered with China propaganda arm `https://www.reuters.com/world/china/amazon-partnered-with-china-propaganda-arm-win-beijings-favor-document-shows-2021-12-17/`

it. It's like searching for a needle in a haystack. They gather so much data that finding useful information, responding to it, and analyzing it becomes extremely difficult. [41]

1.2.10 Not Only yourself, but You harm others

By not respecting privacy, you also harm others. This way, individuals who need to use privacy-based tools like Tor for their work (such as journalists) become easily distinguishable, and that distinction is very harmful. It becomes clear that anyone using Tor must be hiding something, and they end up on a list that is always monitored.

For example, one of the advantages of Telegram is that you can communicate with others without revealing your phone number. However, when you use an unofficial client of Telegram, it might collect your contacts' numbers. This is detrimental to someone who wants to remain anonymous on Telegram, as others will now know the number associated with a particular account. Take a journalist who writes anonymously and goes against the opinions of powerful individuals; they would want to stay anonymous. Similarly, someone investigating cybercriminals must remain anonymous to avoid threats from those individuals!

- But is are no such person around me!

→ How do you know there aren't? Certainly, someone who wants to stay anonymous is not someone you would recognize. The person behind a certain account could very well be your next-door neighbor! That individual is likely not showing their true self in the real world either. If someone wants to remain anonymous, they definitely wouldn't want even their closest friends to find out! Because if they do, under certain circumstances, those friends might inadvertently reveal their identity or even be pressured to do so. Because they know your relationships. So, you definitely wouldn't know. Search "Ross Ulbricht"; does this face show he has launched one of the biggest drug websites on the dark web?

Or government can have excuse that why you don't want us to collect data? Are you a criminal?

1.2.11 Solution

Privacy means that I have boundaries for myself! It's not about saying, "I won't give my data to Meta, but I will give it to Mozilla!" That's a mistake! The right approach is to not give my data at all! It's not about what's acceptable for one and wrong for the other; it's about maintaining my privacy across the board.

- So, what's the solution? What should we do?

[41] 1. NSA is so overwhelmed with data, it's no longer effective, says whistleblower (Former NSA official):ttps://www.zdnet.com/article/nsa-whistleblower-overwhelmed-with-data-ineffective/ 2. The NSA's Call Record Program, a 9/11 Hijacker, and the Failure of Bulk Collection: https://www.eff.org/deeplinks/2015/04/nsas-call-record-program-911-hijacker-and-failure-bulk-collection

→ Well, to put it simply, there is no definitive solution! We can only offer some general suggestions.

• What do you mean? Please explain more!
→ Privacy boundaries are different for everyone. I might not need to follow certain precautions, but you might need to based on your circumstances! For example, someone working in a specific section of the NSA has a job that requires confidentiality. If they don't maintain that secrecy, their life could be in danger! So, we really need to pay attention to our personal privacy boundaries and our threat model!

Definition 2. *Threat Model is a way of thinking about the potential risks you face and how you can protect yourself against them.*

1.2.12 If You Don't Pay the Product, You Are the Product!

Why do you pay for ice cream? Because there are costs associated with producing the milk, as well as expenses for the people involved in the production process.

In the digital world, applications and content are also created by individuals who invest their time and effort. Therefore, using these resources naturally incurs a cost. Just because something isn't physical doesn't mean it doesn't have associated expenses. Free programs often have various revenue models:

- The provider values privacy and receives funding from organizations. In some countries, when you engage in a nonprofit activity, you can request funding by stating that you are helping people for free and need financial support. Thus, you can receive a portion of public revenue or taxes.
- Advertising
- Donations (We have talked about it in the beginning of the book)
- Collecting and selling data to other companies. This is where we say that if you don't pay the product, you are the product!

1.2.13 Reading List

- Citizenfour: A documentary about Edward Snowden and the NSA.

1.3 Anonymity

On the internet, everything you do is converted into messages, like sending a letter. Of course, in order to deliver that letter, you need to have an address. On the internet, you have an address too. This is called an IP address. This way, anything you do is traceable.

- Privacy: It is OK to others to know that you are doing something, but they shouldn't know what you are doing. (The thing is hidden) For example, in bathroom, you don't want others to know what you are doing. You just want them to know that you are in the bathroom.
- Anonymity: It is OK to others to know what is happening, but they shouldn't know who is doing it. (The person is hidden) For example, in bathroom, people see whats happening, but they don't know who is doing it. (The person is wearing a mask)

This, of course, requires a lot of work. You need to first learn about privacy and security and what actions you should take based on your own threat model. You should read all the topics in this book and take action. For now, to make it **harder** (not impossible) to trace you, you can use *Tor Browser*. It is a browser based on Firefox that routes your traffic through multiple servers. This way, the website you are visiting can't know who you are. It is slower than normal browsers because of the multiple servers, but it is worth it. It is important to not reveal anything that seems directly or indirectly related to your identity there. For example, don't use your real name, don't use your real email, don't use your real phone number, and don't talk about your interests, etc.

• Does Incognito mode/Private window help?

→ No! It only deletes browsing data from your computer after you've finished your browsing. Some settings for better privacy are also on. However, it doesn't prevent others (like your internet service provider or anyone who has access to your network) from seeing your browsing data.

1.4 End of the Chapter Questions

1. In situation where a critical vulnerability is found, people should update their software as soon as possible. But many people don't update their software soon. A programmer suggests that the your software should have a backdoor to we ourself can update the software. What do you think about this idea?

answer: This is a bad idea. Because if a backdoor is found by bad people, they can access the data and use it for their own benefits. Or one of the programmers can sell it to bad people. or one of the employees may be forced to give the backdoor to bad people.

1. Implement a backdoor for a vending machine. Example answer: if a person enters a specific code, the vending machine will give them a free drink.

2. Some people suggest that with smart phones, protecting privacy is a joke. They suggest you should have a classic push button phone. In this way you cannot be tracked and hacked. What do you think about this idea? answer: No. Many push button phones use simple and old protocols which are not secure. For example, in Russia, some push button phones were sold with malware preinstalled. [42]

[42] `https://therecord.media/malware-found-preinstalled-in-classic-push-button-phones-sold-in-russia/`

3. threat model for a journalists
they share with hundreds. just one time see cookies vendors

2.0 Cryptography

2.1 Why you should know about cryptography

Cryptography is one of the most important tools for securing data. Everyone who work in the field of computer, should have a basic understanding of cryptography. Without cryptography your data can be read by anyone. Without cryptography:

- Anyone can steal your banking information when you are doing online shopping
- Anyone can read your messages when you are chatting with your friends
- Someone can change your grades in the school system
- Someone can send a message to your friend with your name and your profile
- People can see what websites you are visiting
- You can't be sure the file you downloaded is the original one
- We won't have crypto currencies
- Secret military information can be stole

With cryptography you can search something in a search engine (e.g. Google, Duckduckgo, etc.) and without revealing who you are and without search engine know what you've searched, give the correct result! Yeah! This is possible. It is called *Privately Outsourcing Computation*

2.2 Reading List

I will try to explain the basics of cryptography mainly for non-technical people. It's important to note that cryptography is a complex field, and you should **NOT** take advice from me or anyone who is not a qualified professional. I recommend viewing this document as a good starting point, but please do not

rely on the advice or information provided here. Instead, seek out more authoritative sources for guidance in this area. There are sources you may want to consider:

- **Books:**
  - *Applied Cryptography: Protocols, Algorithms, and Source Code in C*, second edition, by Bruce Schneier (I recommend anyone who want to know more about security, read this book. Although it is a little bit old, it is still a wonderful book.)
  - *Serious Cryptography: A Practical Introduction to Modern Encryption*, second edition, by Jean-Philippe Aumasson
  - *Cryptography Engineering: Design Principles and Practical Applications*, by Niels Ferguson, Bruce Schneier, and Tadayoshi Kohno
- **Online Courses:** There are many online courses on cryptography. Some of them are:
  - *Cryptography I* by Stanford School of Engineering[1]
  - *Cryptography II* by Stanford School of Engineering[2]
  - *Cryptography* by Christof Paar on YouTube
- **Conferences:** There are many conferences on cryptography and security. Some of them are:
  - *Crypto* by IACR
  - *Eurocrypt* by IACR
  - *RSA Conference* by RSA
  - *Black Hat* by Black Hat

2.3 What problem am I solving?

There are many documents out there. Why another one? Most of them are either very technical, involving number theory basics, or they just use tools with no explanation about how they work or what not to do with them.

I want to explain wrong behaviors while using crypto technologies and suggest some better (but not ideal!) practices. In addition to that, I hope you become interested in the field and start to learn more about it, as well as write your programs and act in cyberspace more securely.

[1] `https://online.stanford.edu/courses/soe-y0001-cryptography-i`
[2] `https://online.stanford.edu/courses/soe-y0002-cryptography-ii`

2.4 Cryptographic Hash Function

We will talk about **cryptographic** hash functions. Not those hashes you may heard in hash table.

Think you have a website. People come and create an account there. You have to store username and password associated with it for each user. Something like this[3]:

Username	Password
Alice	1234
Bob	5678
Charlie	9012

This structure helps you authenticate a user for login. You simply get the username and password from the user, then check whether they are valid or not. But is it safe and secure? **NO!** The problem is that if a person hacks into your servers, they can see all the passwords! So, what should you do?! Let me introduce you to hash functions:

I) Hash is a **one-way** function where gives a strange output for a given input. For example, if you hash `1234`, you get

```
03ac674216f3e15c761ee1a5e255f067953623c8b388b4459e13f978d7c846f4
```

it is a strange text nah?! As it is one-way, it is impossible to get `1234` from this text. It means there is no function to give that strange text and get `1234`

II) Hash is an **deterministic** function. It means if you run that algorithm in any system, you always get the same result.

III) Each text (or to be more precise, each binary), has a its own hash (two different binary, have two different hash):

`hello` → `2cf24dba5fb0a30e26e83b2ac5b9e29e1b161e5c1fa7425e73043362938b9824`

`Kamal` → `d091eea64f6591f3e4f5f484543bfe196ce1c59f9cbb8b0fe1e7ee477a5fdb87`

`hi` → `8f434346648f6b96df89dda901c5176b10a6d83961dd3c1ac88b59b2dc327aa4`

IV) If you change one bit of the text, the hash will be completely different. For example, if you hash `hello`, you get:

```
2cf24dba5fb0a30e26e83b2ac5b9e29e1b161e5c1fa7425e73043362938b9824
```

and if you hash `Hello`, you get:

```
185f8db32271fe25f561a6fc938b2e264306ec304eda518007d1764826381969
```

see? I only changed some bits of the text (See in ASCII), but the hash is completely different.

[3] Never ever use this method! Additionally never ever use this simple passwords

V) Not only can I not find a function to give a hash and get the text, but I also can't guess what to input to get a specific hash. This means I can't determine if changing the 4th bit will change the output in a specific way (without running the algorithm).

In summary:

1. It is one-way. From output, you can't get the input.
2. It is deterministic. You always get the same output for a given input.
3. Each input has its own hash. No two different inputs have the same hash. [4]
4. Even if you change one bit, the hash will be completely different. [5]
5. It is fixed-length. No matter how long the input is, the output is always the same length. (e.g. 256 bits)[6]
6. An arbitrary output cannot be generated from a given input. [7]
7. It is a like a random mapping from all possible inputs to all possible outputs.[10]

The above list suggest that we can use a hash function for better password storage[8]. Instead of storing the password, we store the hash of the password:

Username	Hash
Alice	03ac674216f3e15c79e13f978
Bob	185f8db32271fe25f561a938b
Charlie	8f434346648f6b96df89dda90

When a user wants to log in, we get the password, hash it on the client side[9], and compare it with the stored hash. If they are the same, we let the user in. This way, even if a person hacks into your servers, they can't see the passwords. They can see the hash, which, because of its one-way nature, prevents them from retrieving the password.

Lets create a hash algorithm. I associate each character with a number:

Character	Number
A	1
B	2
C	3
D	4
E	5

[4] Collision-resistant

[5] Avalanche effect

[6] Although we have variable length output hashes, but here we only consider fixed length one which are easier to understand.

[7] Pre-image resistant

[8] Still not secure!

[9] Because of the security of the user, it is important that passwords never leave the client device.

Now lets hash `ABD` $\rightarrow 1+2+4=7$ For better one-wayness, I can use modulo operation. So, I can hash `ABD` $\rightarrow (1+2+4) \mod 5 = 2$ Now you see? Someone doesn't know whether 2 means `AA` or `B` or `AF` etc. I wanted to tell you that one-wayness is possible.[10]

But my hash function has several major problems:

- It just supports characters. Not all binaries.
- Two different inputs, can have same output. This is called collision.

2.5 Birthday Problem and Hash Collision

I have brought you this problem to explain why self-made hash functions are not secure and why it is important for cryptography-related algorithms to be public and analyzed by mathematicians and cryptographers.

Example 2.5.1. *In a room with 23 people, what is the probability that at least two people share the same birthday?*

The first answer that comes to mind is that the probability is something like $\frac{23}{365}$ or $1-\frac{23}{365}$. But lets play with its mathematics. When it asks at least two people share the same birthday, we can find the probability of no two person share the same birthday and subtract it from 1.

With two people, the probability that they don't share the same birthday is $\frac{365}{365}\times\frac{364}{365}$. It means that first one can choose any day and left the second person with 364 choices. With three people, the probability that they don't share the same birthday is $\frac{365}{365}\times\frac{364}{365}\times\frac{363}{365}$. It means that first one can choose any day, left the second person with 364 choices. After the second person choose a day, the third person has 363 choices. With 23 people, the probability that they don't share the same birthday is $\frac{365}{365}\times\frac{364}{365}\times\cdots\times\frac{343}{365}\equiv 0.5$ So the probability that at least two people share the same birthday is $1-0.5=0.5$. It means that with 23 people, the probability that at least two people share the same birthday is 50%. It is pretty high, isn't it?

Conclusion: In general, if an element can take on N different values, then you can expect the first collision after choosing about $\sqrt{N}$ random elements (Because the probability is 50%)

Example 2.5.2. *You want to assign a unique and random number to each person in the world. If the population of the world be 8 billion, is 64-bit integer enough for this task?*

The answer is no! with this, you will find collision in roughly $\sqrt{2^{64}}=2^{32}=4,294,967,296$ people. It means that with 4,294,967,296 people, you will find two people with the same number.

• There is a question in my mind. You said that each binary has its own hash and no two hash are the same. But I suspect to that. We can give infinitely

[10] Same is possible with XOR operation.

many inputs to a hash function. So, there should be a collision.
→ You are right. There is a collision. It is called *Pigeonhole Principle*. If you have N pigeonholes and M pigeons, and $M > N$, then there must be at least one pigeonhole with more than one pigeon. Nowadays we use 256-bit hash functions. 2^{256} is a very big number. For collision, $\sqrt{2^{256}} = 2^{128} = 340,282,366,920,938,463,463,374,607,431,768,211,456$ is a very big number. It is very very very hard to find two different inputs with the same hash.

For example *MD5* is a 128-bit hash function. It means that with 2^{64} inputs, you will find two inputs with the same hash. It is not secure anymore.[11] *SHA-1* is a 160-bit hash function. It is not secure anymore.[12]

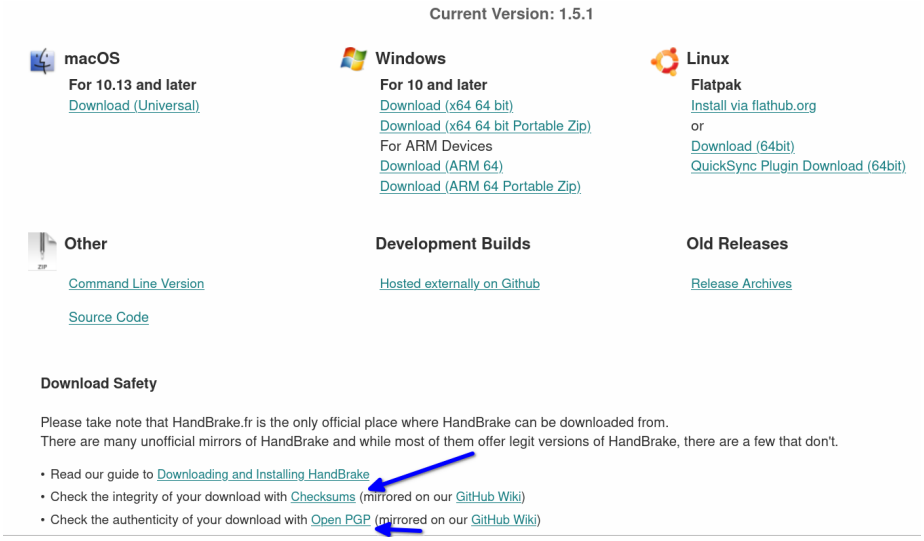
We use *SHA-2* and *SHA-3* which are recommended by NIST.
• Finding collision for a hash is not practical. It requires lots of memory to store values and each time checking the new value with the database
→ No. There is a method, called *Pollard's Rho* which doesn't require much memory.

2.6 Applications

Do you remember I told you that with any changes to the input, the output will be completely different? Think about and find an application for this property.

You have received a file of an application from your friend. You don't trust him, and you want to know whether or not the file has changed (and possibly contains viruses). You can hash the file and compare it with the hash provided by the developer. If they are the same, you can be sure that the file has not changed. If they are different, you can be sure that the file has changed. Most developers provide the hash of the file on their website. For example, *Handbrake* (a video converter software)[13] provides the hash of the file on their website:

[11] https://mail.python.org/pipermail/python-dev/2005-December/058850.html
[12] https://security.googleblog.com/2017/02/announcing-first-sha1-collision.html
[13] You can also use Handbrake to reduce the size of a video by 80 percent without loss of quality! In the Dimensions menu, set the cropping option to None. In the Video section, test the RF number to see what works best for you. You can record a very short video and use it for testing, so it doesn't take too much time to reduce the file size. For example, record a 30-second video and test different numbers. Usually, 32 or 28 is a good value. You can test slightly higher or lower values. The higher the number, the more the file size will be reduced (but beyond a certain point, the quality will decrease). If the video quality is lower, you can increase this number. After that, click Start to begin.



Current Version: 1.5.1

macOS
For 10.13 and later
Download (Universal)

Windows
For 10 and later
Download (x64 64 bit)
Download (x64 64 bit Portable Zip)
For ARM Devices
Download (ARM 64)
Download (ARM 64 Portable Zip)

Linux
Flatpak
Install via flathub.org
or
Download (64bit)
QuickSync Plugin Download (64bit)

Other
Command Line Version
Source Code

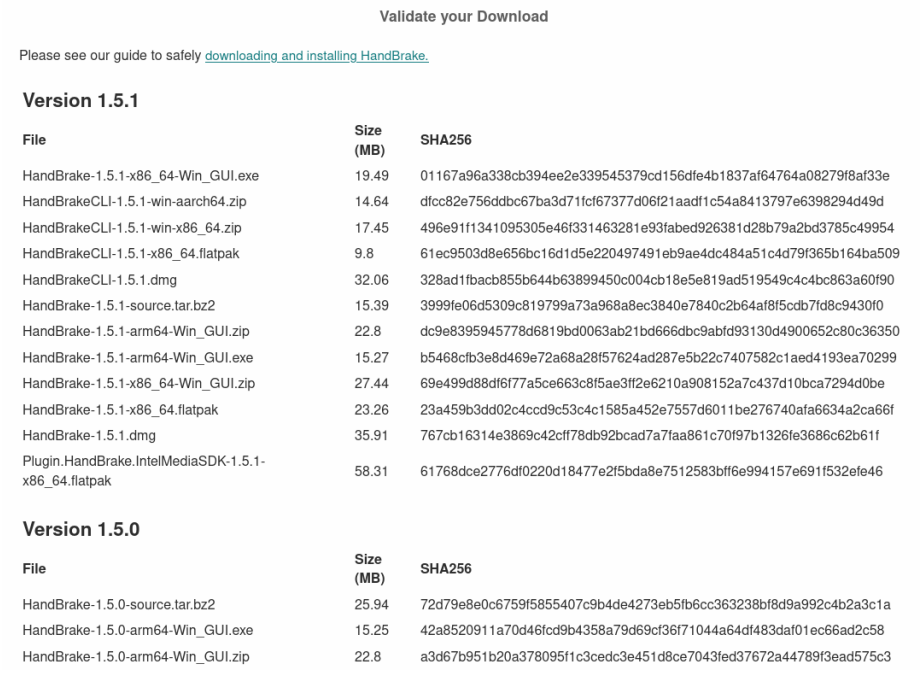
Development Builds
Hosted externally on Github

Old Releases
Release Archives

Download Safety

Please take note that HandBrake.fr is the only official place where HandBrake can be downloaded from.
There are many unofficial mirrors of HandBrake and while most of them offer legit versions of HandBrake, there are a few that don't.

- Read our guide to Downloading and Installing HandBrake
- Check the integrity of your download with Checksums (mirrored on our GitHub Wiki)
- Check the authenticity of your download with Open PGP (mirrored on our GitHub Wiki)



Validate your Download

Please see our guide to safely downloading and installing HandBrake.

Version 1.5.1

File	Size (MB)	SHA256
HandBrake-1.5.1-x86_64-Win_GUI.exe	19.49	01167a96a338cb394ee2e339545379cd156dfe4b1837af64764a08279f8af33e
HandBrakeCLI-1.5.1-win-aarch64.zip	14.64	dfcc82e756ddbc67ba3d71fcf67377d06f21aadf1c54a8413797e6398294d49d
HandBrakeCLI-1.5.1-win-x86_64.zip	17.45	496e91f1341095305e46f331463281e93fabed926381d28b79a2bd3785c49954
HandBrakeCLI-1.5.1-x86_64.flatpak	9.8	61ec9503d8e656bc16d1d5e220497491eb9ae4dc484a51c4d79f365b164ba509
HandBrakeCLI-1.5.1.dmg	32.06	328ad1fbacb855b644b63899450c004cb18e5e819ad519549c4c4bc863a60f90
HandBrake-1.5.1-source.tar.bz2	15.39	3999fe06d5309c819799a73a968a8ec3840e7840c2b64af8f5cdb7fd8c9430f0
HandBrake-1.5.1-arm64-Win_GUI.zip	22.8	dc9e8395945778d6819bd0063ab21bd666dbc9abfd93130d4900652c80c36350
HandBrake-1.5.1-arm64-Win_GUI.exe	15.27	b5468cfb3e8d469e72a68a28f57624ad287e5b22c7407582c1aed4193ea70299
HandBrake-1.5.1-x86_64-Win_GUI.zip	27.44	69e499d88df6f77a5ce663c8f5ae3ff2e6210a908152a7c437d10bca7294d0be
HandBrake-1.5.1-x86_64.flatpak	23.26	23a459b3dd02c4ccd9c53c4c1585a452e7557d6011be276740afa6634a2ca66f
HandBrake-1.5.1.dmg	35.91	767cb16314e3869c42cff78db92bcad7a7faa861c70f97b1326fe3686c62b61f
Plugin.HandBrake.IntelMediaSDK-1.5.1-x86_64.flatpak	58.31	61768dce2776df0220d18477e2f5bda8e7512583bff6e994157e691f532efe46

Version 1.5.0

File	Size (MB)	SHA256
HandBrake-1.5.0-source.tar.bz2	25.94	72d79e8e0c6759f5855407c9b4de4273eb5fb6cc363238bf8d9a992c4b2a3c1a
HandBrake-1.5.0-arm64-Win_GUI.exe	15.25	42a8520911a70d46fcd9b4358a79d69cf36f71044a64df483daf01ec66ad2c58
HandBrake-1.5.0-arm64-Win_GUI.zip	22.8	a3d67b951b20a378095f1c3cedc3e451d8ce7043fed37672a44789f3ead575c3

This is what happens when you download a package or software in GNU/Linux distributions. The package manager automatically checks the hash of the package and compares it with the hash provided by the developer. If they are the same, it installs the package. If they are different, it shows an error message.

What do you think are the properties of hash functions used in this application? Mainly collision resistance. Additionally, it must be fast to compute.

Why? Because you don't want to feel like you are waiting a long time to install software. On servers, many hashes are computed every second, so it must be fast.

2.7 Standardization of Hash Function

Why do we need to standardize a hash function? One of the most important things in security is that the algorithm should be public. This means that everyone can see the algorithm and analyze it. Who can promote this better than a standard organization? This is the best way to encourage public analysis of an algorithm.

NIST (National Institute of Standards and Technology) is one of the most important standard organizations. They organize competitions for new algorithms. Over several years, people try to attack each other's algorithms. The algorithm that is not broken is selected as the new standard.

NSA (National Security Agency) is another important organization. They are involved in the process of standardization. They have a lot of experience in the field of security and can help find vulnerabilities in the algorithms.

2.8 SHA-3

SHA-3 is the latest member of the Secure Hash Algorithm family. It was released by NIST and originated from a competition called the SHA-3 competition. The goal was to find a new secure hash algorithm. First, some workshops were held to discuss the requirements of the new algorithm. Then, NIST told the world to bring their algorithms for the competition. After 5 years of competition, Keccak was selected as the winner. It is now known as SHA-3. Hooray! We have a new secure hash algorithm.

Now you see why we say that self-created hash functions (and any security protocols) are not necessarily secure? Five years of research!

Let's delve deeper into the SHA-3 project.[14]

> As was the case for the AES competition, security is the most important factor when evaluating the candidate hash algorithms. However, there remains significant disagreement within the cryptographic community-at-large over what security definitions should be used to evaluate hash algorithms. While initially proposed for use in digital signatures, cryptographic hash algorithms are used in a wide variety of applications, including message authentication codes, pseudorandom number generators, key derivation, and one-way functions for obfuscating password files. All of these applications have different security requirements.

[14] https://csrc.nist.gov/Projects/hash-functions/sha-3-project

It is obvious that a hash should be secure. A hash that doesn't have those properties we talked about is not secure.

> FRN-Nov07 identified cost as the second-most important criterion when evaluating candidate hash algorithms. In this case, cost includes computational efficiency and memory requirements. Computational efficiency essentially refers to the speed of an algorithm. NIST expects SHA-3 to offer improved performance over the SHA-2 family of hash algorithms at a given security strength. Memory requirements refer both to code size and random-access memory (RAM) requirements for software implementations, as well as gate counts for hardware implementations

If something wants to be used in the whole world and by any device, it must be fast. Don't just consider your laptop; think about a smartwatch or a smart card. They have very low computational power. It must be fast for them too. Additionally, it must be memory efficient.

> The SHA-3 competition has received many candidate algorithms with new and interesting designs, and with unique features that are not present in the SHA-2 family of hash algorithms. Candidate algorithms with greater flexibility may be given preference over other algorithms. This includes algorithms capable of running efficiently on a wide variety of platforms, as well as algorithms that use parallelism or instruction set extensions to achieve higher performance. In addition, simple and elegant designs are preferable, in order to encourage understanding, analysis and design confidence.

As we talked about falsifiability, in order to analyze and implement different attacks on an algorithm, it must be simple and elegant. If it is complex, it is hard to understand and analyze. It is hard to find vulnerabilities and design attacks on it.

2.9 Password Hashing Algorithms

Until now, we have only talked about certain types of hash functions. But here we want to know more about password hashing algorithms.

2.9.1 Traditional Brute-Force

Imagine you want to open a safe, but you don't have the password. What can you do? You can try all the possible passwords, like `1111`, `1112`, `1113`, and so on. Eventually, you will find the password. This is called a brute-force attack. It is the most basic type of attack. It is very slow, but it is effective.

If the database of passwords is stolen, the attacker can use this method to find the passwords. He can try all the possible passwords, find the hashes of them, and compare them with the database. How can we make it challenging?

Password hashing algorithms are designed to be slow and to consume a lot of memory. This way, calculating too many hashes for attacks becomes very hard (compared to SHA).

```
time printf "password" | argon2 somesalt -id -t 4 -m
    16 -p 1
```

You will find that it roughly takes a fraction of a second (compared to SHA, which takes a fraction of a millisecond or less). But this process takes a lot of time.

2.9.2 Hash Table Attack

What if there was a table which contains all the hashes of all the common passwords? Like:

Password	Hash
Admin	c1c224b03cd9bc7b6a86d77f5d
Alex2000	85566b2fad150f8d8297b383c5
123456	8d969eef6ecad3c29a3a629280

The attacker can just compare the hash with the table. This is called *Hash Table Attack*. Actually there are a lot of tables on the internet. People can download it and use it for their attacks.

2.9.3 Rainbow Table Attack

The problem with a hash table attack is that it is very large. For 1 billion passwords (aside from the passwords themselves), we need $1,000,000,000 \times 256 = 256$GB. What do you think we should do? Of course, we should store less data. But how?

Remember that we said if one bit of the hash changes, the hash will be completely different? A hash is like a random mapping. We can use a portion of the hash instead of the whole hash. For example, we can use the first 16 bits and the last 16 bits of the hash:

Password	Hash
Admin	c1c224b0 - 86d77f5d
Alex2000	85566b2f - 97b383c5
123456	8d969eef - 3a629280

So if the hash is 256 bits, we save 224 bits (87.5% smaller). This method is called *Rainbow Table Attack*.

- Well, you're right that we first and last check the hashes, and if they match, it's very likely that the hash belongs to the password, for example, "admin", which we stored the first and last characters of. However, it's possible that another password could also have similar first and last characters. For instance:

- Admin hash: **c1c224b0**3cd9bc7b6a**86d77f5d**
- Another password hash: **c1c224b0**abc8ab61c9**86d77f5d**

→ I agree that there could be another hash with the same first and last characters but different in the middle. However, the probability of that happening is very low for several reasons:

- The hash function is designed to be collision-resistant. This means that it is very, very hard to find two different inputs with the same hash.
- We are storing well-known passwords, so the likelihood of encountering a famous password is much higher than finding a random password that just happens to have the same hash.
- Let's assume we are extremely unlucky and your point is valid. That's fine! When I'm trying to breach accounts in the database, even if I manage to access just 1,000 out of a million accounts and sell their information, that's enough for me. I've made my money, and that's what matters! So, it doesn't really concern me if, with that very low probability, I can't access a specific account. It's not a big deal unless I specifically want to access a particular account. But again, based on points 1 and 2, the likelihood of that happening is very low!

2.9.4 Salting

The Rainbow table attack arises from the fact that everyone knows the hash of "admin." So, if the database is compromised, they can access it. What can we do? We need to ensure that the hash of "admin" is different in each database. If we add a random string to the beginning of the password, the hash will change. For example, instead of calculating the hash of `admin`, we calculate the hash of `abcdeadmin`! People know the hash of `admin`, but they haven't calculated the hash of `abcdeadmin`! This way, we can prevent rainbow table attacks and hash table attacks. For instance, we can change the table to:

Username	Password	Salt
Alice	e5791349f63f04c733cb6ac4d	a1b59e0c811fd
Bob	1557ff6a427896d3b5c0e867c	b2ef229ba023b
Charlie	e4bef5d7964c9efea8a211332	eef6ecad3c29f

In this way, the attacker needs to calculate the hash of "admin" with each salt. And remember that password hashing algorithms are slow. It will take a lot of time. And as the salts are random, the attacker can't use precomputed tables.

The security of this method depends on the randomness and length of the salt. If you are working with these algorithms, you should seek advice about the length and how to generate the salt from authentic sources.

2.9.5 Different Password Hashing Algorithms

Many times password hashing algorithms are used to generate an encryption key. See NIST recommendation and its FAQ[15].

Argon2

- Winner of the Password Hashing Competition[16]
- Used in KeepassXC[17], Bitwarden[18], Tuta[19]
- RFC 9106[20]

Scrypt

- Used in Litecoin[21]
- RFC 7914[22]

Bcrypt

- Used in Proton Pass[23]
- Hashing in Action: Understanding bcrypt[24]

PBKDF2

- Used in Lastpass[25], Bitwarden[26], 1Password[27][28], Apple[29]

NIST says[30]:

> In response to the public comments received, NIST proposes to revise SP 800-132,
>
> - to approve an additional memory-hard password-based key derivation function and password hashing scheme, and

[15] https://pages.nist.gov/800-63-FAQ/ Especially "Q-B17"
[16] https://www.password-hashing.net/
[17] https://keepassxc.org/docs/KeePassXC_UserGuide#_database_settings
[18] https://bitwarden.com/help/what-encryption-is-used/#argon2id
[19] https://tutanota.com/blog/best-encryption-with-kdf
[20] https://datatracker.ietf.org/doc/html/rfc9106
[21] https://litecoin.org/
[22] https://datatracker.ietf.org/doc/html/rfc7914
[23] https://proton.me/blog/proton-pass-security-model
[24] https://auth0.com/blog/hashing-in-action-understanding-bcrypt/
[25] https://support.lastpass.com/s/document-item?bundleId=lastpass&topicId=LastPass%2Fabout-password-iterations.html
[26] https://bitwarden.com/help/what-encryption-is-used/#pbkdf2
[27] https://support.1password.com/1password-security/
[28] https://1passwordstatic.com/files/security/1password-white-paper.pdf
[29] https://help.apple.com/pdf/security/en_US/apple-platform-security-guide.pdf
[30] https://csrc.nist.gov/news/2023/proposal-to-revise-nist-sp-800-132-pbkdf

- to provide additional guidelines and clarifications on the use of PBKDF2.

NIST[31] and RFC 8018[32] say:

> The iteration count shall be selected as large as possible, as long as the time required to generate the key using the entered password is acceptable for the users. Since the capability of user machines varies (e.g., from a smart card to high- end workstations or servers), reasonable iteration counts vary accordingly. For especially critical keys, or for very powerful systems or systems where user-perceived performance is not critical, an iteration count of 10,000,000 may be appropriate

Conclusion: Always overestimate the iteration count and security measures. It is better to have a slow login than a hacked account. Additionally use an algorithm that slows down attackers more than users.[33]

2.10 Applications of Hash Functions

2.10.1 Bitcoin (SHA2-256)

Currency Evolution: How Trust and Value Shaped Modern Money

A very simple and imprecise explanation of Bitcoin and blockchain (this explanation is not detailed and is just meant to provide a basic understanding! The focus here is not on the technical accuracy of the issues; it's simply to understand that hashes have a purpose):

First, we need to talk about the philosophy of money. In the past, people engaged in barter, meaning they exchanged goods directly. For example, one person might say, "I'll give you two apples if you give me two oranges." However, this system had its problems. For instance, if I wanted to travel to another city, what would I do? I couldn't carry apples and oranges with me. So, people started going to trusted individuals who were honest and reliable. For example, someone would give ten apples to a trusted person, and that person would provide a note stating that this individual has deposited a certain value with them. When the person arrived at their destination, they could receive goods or whatever they needed equivalent to that value, and the trusted person would later return the apples. Gradually, this practice became common and laid the foundation for modern money.

In fact, modern money works in a similar way. The cash you hold doesn't have intrinsic value; rather, it represents that you have a certain amount of money, say five dollars, stored in the central bank. This is why people talk about the backing of money. The backing of a country's currency consists of

[31] https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-132.pdf
[32] https://datatracker.ietf.org/doc/html/rfc8018#section-4.2
[33] Read More at: https://blog.1password.com/bcrypt-is-great-but-is-password-cracking-infeasible/

the gold reserves that correspond to the banknotes it prints in its central bank [34]. However, it's not necessarily just gold; other things can also provide value to money. Gold is just a prime example.

This is why the question "Why don't they just print more money so everyone can be rich?" is dismissed. The reason is that money itself has no intrinsic value; its worth comes from the backing behind it. That's why it's said not to print money arbitrarily, as it would decrease the value of the currency.

For instance, imagine we have one million dollars in cash and one million dollars' worth of gold. If we print another million dollars in cash, we would then have two million dollars in cash for the same one million dollars' worth of gold. This would halve the value of our cash because now, for every two million dollars in cash, we only have one million dollars' worth of gold. It's as if every two banknotes represent one unit of gold. Therefore, any cash printed without backing will harm the value of the currency.

Now that we understand how money came into existence, let's consider another question: Why does gold have value?

One reason is that gold has properties that make it useful in various applications. However, aside from its industrial uses, another reason for its value is its scarcity. Because it is rare and in demand, it holds value. If a large gold mine were discovered tomorrow, the value of gold would decrease. Similarly, if one day everyone woke up and decided they no longer accepted gold, demand would drop, and when demand decreases, people would no longer attribute value to gold, causing its worth to diminish even further (regardless of its physical properties).

Bitcoin is Here

Bitcoin came along and said, "Anyone with a regular computer should be able to own and mine me." That's great! Everyone should have a share. But how do we distribute Bitcoin among people? Literally, individuals need to put in the effort. Just like finding gold requires hard work, here too, people must prove that they have put in the effort, which is essentially what Proof of Work means.

So, what is one way to prove that work has been done? Can you guess? Yes, it's calculating hashes! Bitcoin uses the SHA-256 hash function, which we've shown can be calculated relatively quickly. That's why Satoshi Nakamoto said that people need to keep calculating hashes until they find a certain number of leading zeros in their hash. This means they have to brute-force their way to find those specific zeros. Up to this point, everything seems normal.

However, things became unfair. Specialized devices were created (ASIC/FPGA) solely for mining Bitcoin. We know that the computers we use can run games, browse the web, watch videos, and do a thousand other tasks. Their versatility means they can do everything, but not necessarily at very high speeds. But then, devices were developed that could only calculate hashes, and they did so at a much higher speed. These are called miners.

[34] *Inside the Bank of England's gold vaults* From World Gold Council in Youtube: `https://youtu.be/FVnxhsB92v4?feature=shared`

This is where the situation became unfair. Because anyone with more money could buy special devices that could mine at a much higher speed. This meant that the rich got richer, and the poor got poorer, as the competition for calculating hashes was dominated by those who could afford to buy the devices. That's why other coins emerged that used more resistant hashes against these devices, making it difficult to create specialized mining equipment for them. The cost of building those devices also increased significantly, which helped maintain a bit more fairness. In contrast, with a relatively low investment in Bitcoin mining, you could earn a lot more money.

What kind of hashes do you think these could be? Those which require more hardware resources. Password hashes! Yes, you can use password-related hashes. This is what some of the newer coins do:

- Litecoin (Scrypt)
- Zcash (Equihash)

Researchers say[11]:

> A function that is memory-hard under this definition requires the adversary to use either a lot of working space or a lot of execution time to compute the function. Functions that are memory-hard in this way are not amenable to implementation in special-purpose hardware (ASIC), since the cost to power a unit of memory for a unit of time on an ASIC is the same as the cost on a commodity server. An important limitation of this definition is that it does not take into account parallel or multiple-instance attacks.

Also we have[12]:

> GPU/FPGA/ASIC-unfriendly. Argon2 is heavily optimized for the x86 architecture, so that implementing it on dedicated cracking hardware should be neither cheaper nor faster. Even specialized ASICs would require significant area and would not allow reduction in the time-area product.

Warning

Bitcoin is not designed to make you anonymous. All transactions are recorded. Eventually you need to convert that bitcoin into something else (money, goods, etc.). At that point, you will be identified. So, don't think that you can do anything with Bitcoin and remain anonymous. It is not true. There are other coins that are designed to make you more private. For example, Monero.

2.10.2 Cloud-base Applications

When you want to send a file to your friend in a regular Telegram chat, you might notice that as soon as you hit the send button, the progress bar jumps

to 100%. The reason is that Telegram checks and sees, "Oh! The hash of the file you are sending matches the hash of one of the files on my server." So, why would you want to waste your internet data sending it? There's no need to send it! I can just send the file with the same hash from my server to the other person!

For example, if we want to find a copyrighted image, we can scan the entire phone. If we find a matching hash, we've located it! :)

Google Safe Browsing uses the hash of the URLs to check if the website is safe or not. If the hash of the URL is in the database, it is not safe.[35]

2.10.3 Reading List

- Hash functions: Theory, attacks, and applications[13]
- SHA256 Algorithm Explained (A wonderful website that interactively shows how SHA-256 works)[36]
- How the MD5 hash function works (from scratch)[37] (A wonderful video)

2.11 Coding

Think you want to transmit your words via computers. Computers only know zeros and ones. What to do then? A simple solution is to associate each character with a number. For example, we can associate 'A' with 1, 'B' with 2, 'C' with 3, and so on. Then we can represent each number in base 2 (binary). For example, 'A' is 1, which is 0001 in binary. 'B' is 2, which is 0010 in binary. 'C' is 3, which is 0011 in binary. Then we can transmit the binary numbers. This is what happens in computers. Each character is associated with a number, and then the number is represented in binary.

Definition 3. *The process of converting A to B is called encoding A with B.*

Definition 4. *The process of converting B to A is called decoding B with A.*

To have a common means of communication, we have to rely on a standard. ASCII (American Standard Code for Information Interchange) is one of the most common standards. It associates each character with a number. For example, 'A' is 65, 'B' is 66, 'C' is 67, and so on. (Search for ASCII table.) Unicode is another standard. You can search about it.

The important note is that coding is not encryption. Coding is just a way to represent data.

[35] `https://security.googleblog.com/2022/08/how-hash-based-safe-browsing-works-in.html`

[36] `https://sha256algorithm.com/`

[37] From Rare skills Youtube Channel: video link

2.12 Encryption

Imagine I want to write a letter to someone. Well, it needs to be sent by mail. How can I be sure that the postal worker won't open it to read it? There's no guarantee that someone won't read it along the way!

This is where the concept of encryption comes in. It means that I can write a message in a coded form that only I and the recipient can understand. For example, what do you think the following text means?

```
uoy era woh
```

If you read it backwards, you'll see that it says:

```
how are you
```

This is a simple example of encryption. The message is encoded in a way that only the recipient (who knows how to decode it) can understand. This is the basis of encryption. It's like a lock. You have the key, and only you can open it. If someone else tries to open it, they won't be able to. This is the essence of encryption.

Now what do you think about the following text?

```
ipx bsf zpv
```

It is a simple Caesar cipher. It means that each letter is shifted by a fixed number of positions down the alphabet.[38] For example, if the shift is 1, A becomes B, B becomes C, and so on. This is another example of encryption. Now shift the letters by 1 in reverse and you will see that it says:

```
how are you
```

Definition 5. *The key principle in cryptography is this: It doesn't matter whether a message is being sent or not. What's important is that the actual meaning of our message should not be revealed. In other words, it's acceptable for an encrypted message to be transmitted and for someone to realize that a message has been sent; the only requirement is that the content must be encrypted so that the original intent and message remain unknown.*

2.12.1 Kerckhoff's Principle

Kerckhoff's in the 19th century said that the security of a cryptographic system should not depend on the secrecy of the algorithm. It should depend on the secrecy of the key. This means that the algorithm should be public. Everyone should see it. But the key should be secret.

Many of our previous examples (and other more complex hand-made ciphers) can be easily broken by computers[39]. Do NOT rely on them.

[38] See more at: `https://en.wikipedia.org/wiki/Caesar_cipher`

[39] Hacking Secret Ciphers with Python: `https://inventwithpython.com/hacking/`

2.12.2 Steganography

Steganography is sometimes misunderstood as encryption. Although they both help security and privacy, they are different. What do you think the following text means?

```
jf@hdfs87@ijksd-isj!mjlsfj9$bkjsd+akdlfj42fs%cjsdlf$kjoi1
```

If you look at it, you will see that it is a random text. But if you look at the first letter after each symbol, you will see that it says:

```
Hi I'm back
```

A similar thing can be done with a long letter. The first character of each sentence can be used to hide a message.

You can also hide a message in an image.[40] For example, you can change the least significant bit of each pixel to hide a message. Because LSBs are changed, changes are very hard to detect by the human eye. (But it can be detected by a computer.)

Definition 6. *In steganography, the actual message must be hidden. It means that nobody should know that a message is hidden in something else.*

Sometimes steganography is used with encryption. The message is first encrypted and then hidden in something else. In this way, even if someone finds the hidden message, they can't understand it.

Reading List

- Steganography (Wikipedia)[41]
- Exploring Steganography: Seeing the Unseen[42]
- Hide secret messages into a spammic message[43][44]

Function

A function is like a machine. You give it an input, and it gives you an output. For example, when we say $y = 2x$, it means that if you give 3 to the function, it will give you 6. If you give 4, it will give you 8. If $y = 2x - 3$ is our function, it means that if you give a number as x, it will double it and subtract 3 from it.

Functions can be used to encrypt data (though it is not really secure!). For example, your password is:

[40] See also: `https://www.jjtc.com/stegdoc/sec313.html`
[41] `https://en.wikipedia.org/wiki/Steganography`
[42] `https://www.jjtc.com/pub/r2026a.htm`
[43] `https://www.spammimic.com/`
[44] Be careful! It is a website. Don't use it for sensitive data. Never ever send a sensitive data to somewhere else. Even if they don't store it, it can permanently be stored in the logs or hard disks unintentionally. We will talk about swap/shadow/temp files later.

`kAmZ)!rf3V+qzQ2iVQfPVcXaE4`

Using a function, it can be written as:

`kAmZ)!rf5V+qzQ2iVQfPVcXaE8`

What function did I use? The function is $3x - 4$.

There are various ways to slightly alter or obscure the text so that no one notices. Of course, these methods do make it harder for a bad actor, but we shouldn't say they make it impossible! (Because they might be able to guess what function we used!)

For example, your password is:

`aWf8$dG`H5b#jfRgT¿st3ch:tDa7gS

You can write it like this:

`aWf8$dG`H5b#jfR2gT¿st3ch:tD@a7gS

And you know that you never use the number **2** or the **@** symbol in your passwords. So when entering it, you don't include those, but someone who gains access to the password won't know that they need to remove those characters and that we added them as distractions, so they won't be able to log in.

Or, for example, you can write:

`aWf8$dG!H5b#jfRgT>st3ch:tDa!7gS`

like this:

`sv#1af8ws2Ca:seaWf8$dG!H5b#jfRgT>st3ch:tDa!7gSE%dTrâxj7p`

And you keep in mind that your password is between e and E, so anyone reading the text won't understand what your password was.

When all these techniques are combined, they make it so that even if your password falls into someone else's hands, they will have to work harder to understand it. But remember, it's not impossible! It's just a bit harder.

Conclusion: Combine methods before inserting your password into password managers. If your master password for the password manager is compromised and your passwords are leaked, it will be harder for the attacker to understand your passwords.

2.12.3 Number Station

Let's see some more interesting examples. Listen to the following audio:

`https://www.numbers-stations.com/english/e01-ready-ready/`

Strange, isn't it? Now listen to this one:

`https://www.numbers-stations.com/morse/m01/`

The latter one was stranger. But what are they? The second one was transmitting Morse code.[45] The first one was transmitting numbers. But why?

For example, this sequence has been transmitted:

```
A - F - H - H - P - X
34
```

These are also a form of encryption. Imagine a country has a spy in another country. They certainly can't send their messages in a way that anyone can understand. If they send it in a coded format, while the government of that country may not understand the code, they might notice that every day at 9 AM, a certain individual receives an encrypted message. Naturally, the intelligence agency of that country would become suspicious about why this person is receiving encrypted messages at such a specific time. As a result, they might go and arrest that individual.

This led to the idea of a "numbers station," a public source where each person listening to it cannot be distinguished. For instance, the spying country gives its spy a notebook and tells them to write down the code we provide and find the corresponding message in the notebook. The numbers indicate pages, so for example, "34" means to look at page 34, and the random letters also have meaning.

The spy then matches those letters with a table to see what message they were meant to receive.

Letter	Meaning
A	R
P	O
H	N
F	U
X	W

So the message is "RUN, NOW!"

Important Notes

I) A pad should never be used twice! II) The letters should not belong to any specific language. For instance, if Persian letters are transmitted, it would indicate that there is an Iranian spy or at least that the person has some familiarity

[45] If there are sounds, a short sound represents a "dot," while a long sound represents a "dash." Using light, a brief flash of light represents a dot, and a longer light represents a dash.

During the Vietnam War, when Vietnam was capturing prisoners, they would torture them. North Vietnam arranged for a Japanese journalist to conduct an interview with a prisoner, aiming to portray that everything was fine and that they treated the prisoners well. However, the prisoner certainly couldn't say on camera that they were being tortured. So, he used a clever technique. He communicated in Morse code by blinking. A long blink indicated a "dash," while a quick blink represented a "dot." This individual managed to convey the message "T-o-t-u-r-e" to the world, meaning "torture." https://www.military.com/video/operations-and-strategy/vietnam-war/pow-blinks-torture-in-morse-code/1381254901001

with the Persian language.
• What's the problem?!
→ The circle of identifying the spy and the radio broadcaster becomes smaller and smaller. (We will talk about it later when we discuss anonymity.)

• What if sending letters in Persian is a kind of deception, leading people to mistakenly think that it is an Iranian spy?
→ Consider a situation where they forget how to write a specific Persian letter; they might search for the Persian alphabet. Now, by examining the search logs of people who searched for the alphabet, the identification circle can be narrowed down. (They see that every time the radio broadcasts, shortly after, this person is searching for the Persian alphabet or the meaning of a word. It becomes obvious!)

Or suppose I want to set up a radio station; I certainly wouldn't want to broadcast it in my own voice. So what do I do? I search the internet for "text to speech." Well, later they can check the logs of people who searched for that and identify me more easily. Or rather, they might find out that these sounds were taken from a specific website, and then they could go to that website and ask for the IP addresses and device details of those who visited the site or who requested that specific text to be converted to speech. This way, I would effectively be exposed.

There are many cases where individuals who wanted to remain anonymous ended up being identified because, during certain specific events, they were careless and searched for topics using their real IP addresses or wrote about those subjects. This was exactly the beginning of their identification. We will know more about it in the **Dangerous behaviors lead to de-anonymizing** section.

2.12.4 What if using a pad several times?

Reading List

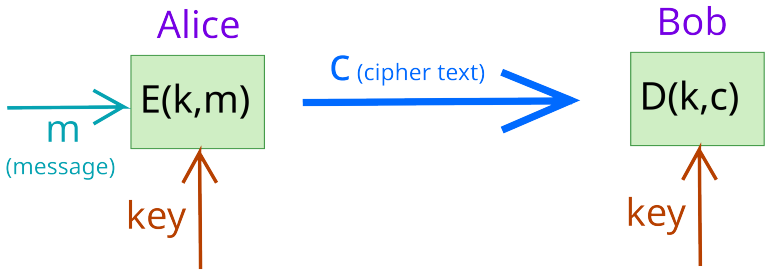
- Jadi's Radiogeek, Episode 24[46]
- Jadi's Radiogeek, Episode 39[47]
- Webdriver Torso YouTube mystery clips' French connection[48]

2.13 Symmetric Encryption

Symmetric encryption is like a traditional lock. You can lock it with a key, and you can unlock it with the same key. For example, if you want to send

[46] https://jadi.net/2013/04/radiogeek-24-numbers-station/
[47] https://jadi.net/2014/05/radio-geek-39-narenji-bleed/
[48] https://www.bbc.com/news/technology-27238332

a message to someone, you can encrypt it with a key, and the recipient can decrypt it using the same key. The Caesar cipher is an example of symmetric encryption. The key is the shift number. If you shift the letters by 3, you can decrypt it by shifting the letters by 3 in reverse.



If you use the wrong key, data is decrypted to random data. A good encryption algorithm is one that only a specific key can use to encrypt data correctly. Any other key will decrypt the data to different random data.

If we have the key, the decryption is very fast. But if we don't have it, **if the encryption algorithm is ideal**, we need to brute-force it. If the key is 256 bits, we need to brute-force 2^{256} values of the key to find the correct one. But is it possible? *Bruce Schneier*, in chapter 7 of his book *Applied Cryptography*, explains [14] that it is not possible.

When we say an attack exists on an encryption algorithm, it means that in addition to brute-force, there is another way to recover the plaintext or key in less time or more. A 2^{250} attack does **NOT** mean it is faster than brute-force. The operations may be different, but the time may be longer. For example, if an attack requires 2^{250} operations but each operation takes 1 second, it means that it requires 2^{250} seconds to break the encryption. But if brute-force requires 2^{256} operations and each operation takes 1 microsecond, it means that it requires 2^{256} microseconds to break the encryption, which is 2^{250} seconds. So, the attack is not faster than brute-force; it is just another way to break the encryption.

Some papers may title their attack as *break*. Don't be confused. It is not a break; it is just another way to break the encryption. It may not even be practical!

A larger key size does **NOT** always mean better security. It is important to stick to authorized recommendations. For example, an attack is possible on *AES-256* and *AES-192*, but not on *AES-128*.[15] This doesn't mean that

AES-256 is insecure.[49]

The important thing is to use a standard encryption algorithm. We talked about it in the *Use of Self-Invented Algorithms* section. The *Vigenère Cipher* tells us that knowing the length of the key should not result in a significant reduction in the time required to break the encryption.

2.13.1 Application of Symmetric Encryption

Some of you may have already used password-protected zip files. In order to prevent unauthorized access to the files, the binary behind the file is encrypted[50] using a symmetric encryption algorithm. The password you enter is used as the key to decrypt the file. If the password is correct, the file is decrypted correctly, and you can access the files. If the password is incorrect, the file is decrypted incorrectly to random data, and this is detected by the software, which tells you that the password is incorrect.

You may want to send a sensitive file through email. As we will discuss later, email is not a secure protocol for sending sensitive data (the email provider sees the actual data). You can encrypt the file using AES and send it to your friend. Your friend can use the same key that was previously agreed upon between you and him to decrypt it.

2.13.2 One-time Pad

There is a perfect encryption. If we use a key that is as long as the message and use it only once, the encryption will be perfect. A simple *XOR* operation can be used:

```
E(m, k) = m XOR k
D(c, k) = c XOR k
```

Why is it unbreakable? Say someone gets *avfsg* as the ciphertext. The person can't know what the plaintext is. It could be *hello*, *world*, *night*, or anything else. There is no pattern. If the key is **as long as the message**[51], **completely random**, and **used only once**[52], the ciphertext will be completely random. Any permutation of the characters will be a valid plaintext.

- Why don't we use it?! It is perfect!

→ Think of having a hard drive with 1TB of data. You want to encrypt it with a one-time pad. You need to generate a random key with 1TB of data. This means not only do you have to find a massive source of randomness[53], but you also have to store it somewhere. You need another 1TB hard drive to store the

[49] Page 38, BSI TR-02102-1: "Cryptographic Mechanisms: Recommendations and Key Lengths" Version: 2024-1: `https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/TechGuidelines/TG02102/BSI-TR-02102-1.html`

[50] It is important for the data to be encrypted. If this were only a software check without any encryption, anyone could read the data behind it with another program.

[51] Not half or 99% of it

[52] Never using even a part of it twice

[53] Generating random bits is very, very, very hard. We will talk about it later.

key.

Moreover, many applications, such as communication via the internet, require real-time transmission of data. Think about wanting to download a file that is 2GB. You somehow need to constantly generate random bits and exchange them with the other side. In the end, you've generated 2GB of random bits and exchanged them with the other side. This significantly increases the time required to download the file.

That was just for one website. Imagine your browser is open on 100 tabs. A huge source of randomness is needed to generate 100GB of random bits. This is not practical. We will see that generating random bits is very hard.

Conclusion: As *Bruce Schneier* says[54], many commercial products that claim to use one-time pads are not actually using them. They just use a symmetric encryption algorithm with a long key. But the key is not equal to the length of the message, nor is it random or used only once.

2.14 Symmetric Encryption Operating Modes

It is easier to design an algorithm that operates on fixed length of data rather than variable length. As a result, we design algorithms that operate on fixed length of data (e.g. 128 bits). Thus a protocol should be used to encrypt and decrypt the data using that. Some are:

- ECB: Electronic Codebook
- CBC: Cipher Block Chaining
- CFB: Cipher Feedback
- OFB: Output Feedback
- CTR: Counter

Definition 7. *Random access means that each block of plaintext can be encrypted or decrypted independently of the other blocks. This is a desirable property for applications that require the ability to access individual blocks of data without having to process or decrypt the entire message. (e.g. video streaming, database, disk encryption)*

Definition 8. ***Error Propagation:*** *The extent to which an error in the ciphertext (e.g. noise) affects the decryption of subsequent blocks.*

2.14.1 ECB (Electronic Codebook)

Divide the plaintext into blocks and encrypt each block separately.
Q: Why code book?

[54] https://www.schneier.com/crypto-gram/archives/2002/1015.html#7

A: Because we can create a table of all the blocks and their corresponding ciphertexts (a code book) with a specific key.
In some applications, the ciphertext should replace the plaintext. Hence, the size of the ciphertext has to be the same as the plaintext. However, in ECB, dividing the plaintext into blocks may result in the last block being smaller than the block size. In this case, we need to pad the last block to make it the same size as the block size. In general, $\frac{\text{block size}}{2}$ bits are added to the last block (as it can vary from 0 bits (no padding required) to block size - 1 bit (only 1 bit of data is in the last block)).

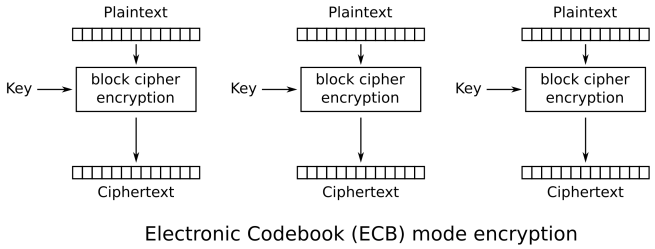


Figure 2.1: src: Wikipedia, Public Domain

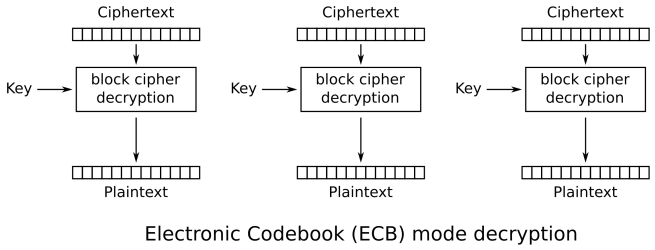


Figure 2.2: src: Wikipedia, Public Domain

Pros:

- Simple
- Parallelizable
- Random access (can decrypt any block independently)
- Error propagation is limited to one block
- No need for initialization vector

Cons:

- Identical plaintext blocks result in identical ciphertext blocks (vulnerable to pattern analysis)[55]
- Susceptible to replay/insertion/deletion attacks (not providing integrity)
- ciphertext size = plaintext size + padding (block size / 2)

2.14.2 CBC (Cipher Block Chaining)

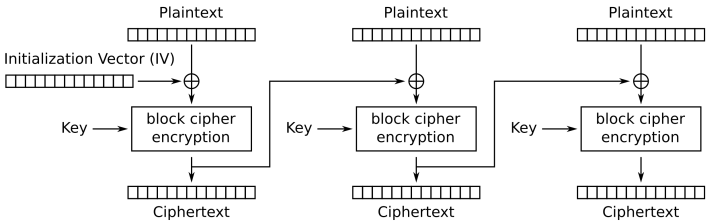


Figure 2.3: src: Wikipedia, Public Domain

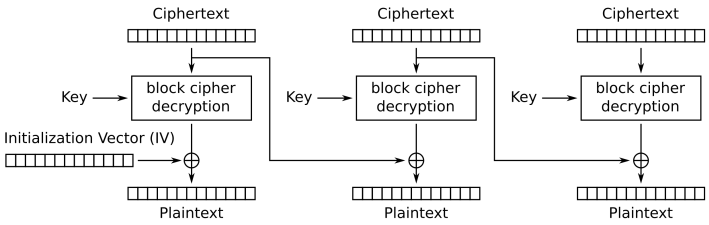


Figure 2.4: src: Wikipedia, Public Domain

https://defuse.ca/cbcmodeiv.htm

- XOR the plaintext block with the previous ciphertext block before encryption.
- The first block is XORed with an initialization vector (IV).
- The IV should be random and unique for each message.
- The IV is sent along with the ciphertext.

[55] See image of Linux at `https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation#Electronic_codebook_(ECB)`

IV should be random, unique, and unpredictable, protected against unauthorized modification. But it doesn't have to be secret.

Counter IV! DON't use it.

Pros:

- Random access (can decrypt any block independently. Because we only need IV or previous ciphertext block)
- Error in plaintext block is limited to that block (Although it affects the ciphertext of that block and all subsequent blocks, doesn't affect the decryption as we only need IV or previous ciphertext block. As long as we use the **same** previous ciphertext block, it doesn't matter if we use *1a2b* or *1a2c* for *XOR*.)
- Error in a ciphertext block garbles the entire corresponding plaintext block and the corresponding bit in the next block. (Again the rest of the blocks are unaffected since we use the **same** previous ciphertext block for decryption)
- Identical plaintext blocks result in different ciphertext blocks (since IV is different for each message)

Cons:

- Not parallelizable for encryption
- Needs using IV
- Susceptible to change/insertion/deletion to last block
- Bit-flipping attack: If an attacker know the structure of the plaintext, changing a bit in the ciphertext will change the corresponding bit in the plaintext.
- plaintext size != ciphertext size
- Cannot recover from synchronization errors (loss/add of a bit)

2.14.3 PCBC (Propagating Cipher Block Chaining)

This mode was designed to cause small changes in the ciphertext to propagate indefinitely when decrypting, as well as when encrypting.

Pros:

- Semi-parallel for decryption (as the most expensive operation (block cipher decryption) can be done in parallel)

Cons:

- No Random access (as we need the previous plaintext block)

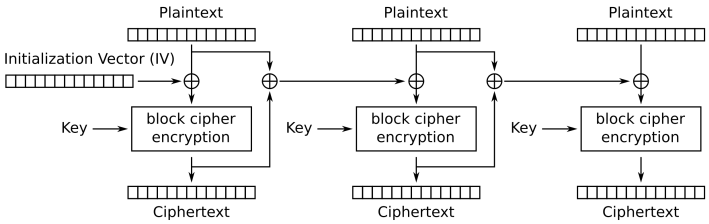


Figure 2.5: src: Wikipedia, Public Domain

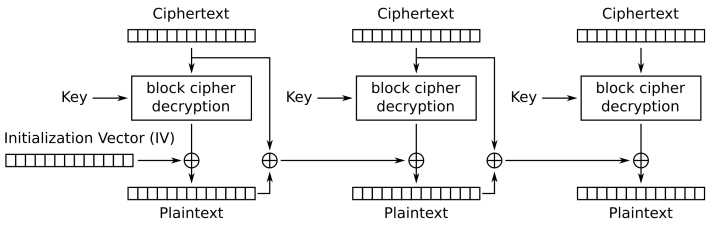


Figure 2.6: src: Wikipedia, Public Domain

- Needs using IV
- Swapping two ciphertext blocks results in garbled corresponding plaintext blocks but the rest of the blocks are unaffected. [16]

For the last con, consider the following example:

- $C_0 = E_k(P_0 \oplus IV)$
- $C_1 = E_k(P_1 \oplus P_0 \oplus C_0)$
- $C_2 = E_k(P_2 \oplus P_1 \oplus C_1)$
- $P_0 = D_k(C_0) \oplus IV$ (I)
- $P_1 = D_k(C_1) \oplus P_0 \oplus C_0$ (II)
- $P_2 = D_k(C_2) \oplus P_1 \oplus C_1$ (III)

If we substitute (I) and (II) in (III), we get:

$$\begin{aligned} P_2 &= D_k(C_2) \oplus D_k(C_1) \oplus P_0 \oplus C_0 \oplus C_1 \\ &= D_k(C_2) \oplus D_k(C_1) \oplus D_k(C_0) \oplus IV \oplus C_0 \oplus C_1 \end{aligned}$$

If we swap C_0 and C_1, in decryption we get:

- $P'_0 = D_k(C_1) \oplus IV$ (I)
- $P'_1 = D_k(C_0) \oplus P'_0 \oplus C_1$ (II)
- $P'_2 = D(C_2) \oplus P'_1 \oplus C_0$ (II)

If we substitute (I) and (II) in (III), we get:

$$\begin{aligned} P'_2 &= D_k(C_2) \oplus D_k(C_1) \oplus P'_0 \oplus C_1 \oplus C_0 \\ &= D_k(C_2) \oplus D_k(C_1) \oplus D_k(C_0) \oplus IV \oplus C_0 \oplus C_1 \end{aligned}$$

We see that $P_2 = P'_2$.

2.14.4 CFB (Cipher Feedback)

Simple variant of CFB decryption is almost identical to CBC encryption performed in reverse. IV is shifted to shift register ans s bit is gone. s bit right is

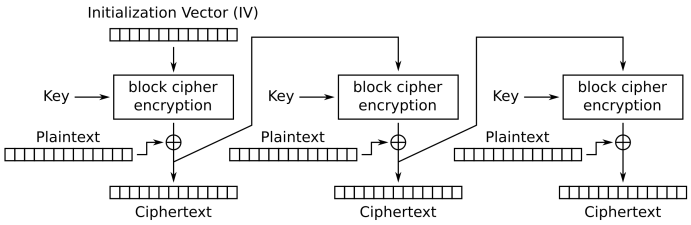


Figure 2.7: src: Wikipedia, Public Domain

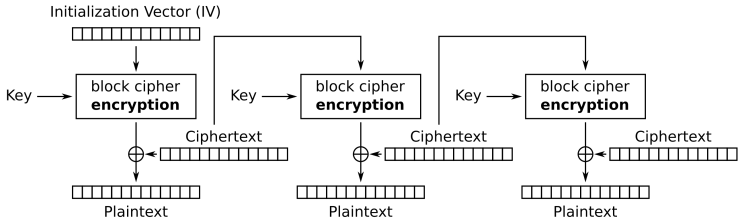


Figure 2.8: src: Wikipedia, Public Domain

replaced by c0

Pros:

- plaintext size = ciphertext size (can be used for streaming)

- Same algorithm for encryption and decryption (Can be used to create special purpose hardware -¿ increases speed + avoids side-channel attacks)
- Random access and parallelizable for decryption
- Self-recover (partial noise-resistant) error in a ciphertext block changes the corresponding bit in the corresponding plaintext block and B/O -1 blocks (B is the block size and O is the output size)??????!!!!!!!!!!
- self-recovering from synchronization errors
- Pattern hiding
- Authentication code*

Cons:

- Not parallelizable for encryption
- Susceptible to bit-flipping attacks (next block is garbled though)

2.14.5 OFB (Output Feedback)

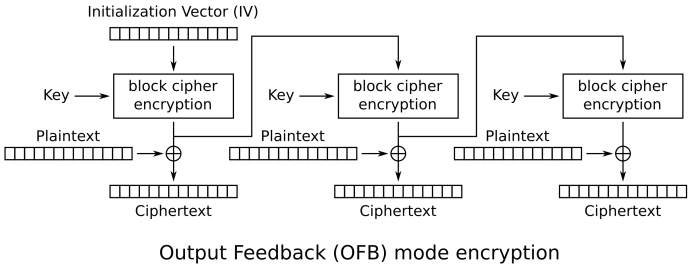


Figure 2.9: src: Wikipedia, Public Domain

Pros:

- a little bit parallelizable for encryption (after calculating the expensive block cipher encryption operations in serial) and decryption
- We can use CBC mode encryption with input of all zeroes, to create key of XOR of OFB mode. (Since $O_i \oplus 000...000 = O_i$) Useful to use CBC hardware acceleration
- Noise-resistant
- Same algorithm for both encryption and decryption
- Doesn't need padding (For the last block you can send only the ciphertext bits corresponding to the plaintext bits) [17]

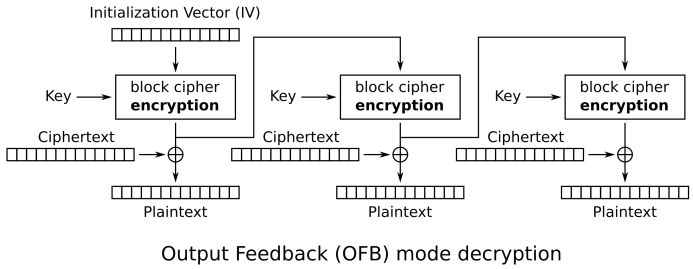


Figure 2.10: src: Wikipedia, Public Domain

Cons:

- No authentication code
- Bit-flipping attack

OFB has birthday problem, problem. (Same key after $2^6$4 block) -¿ also cfb? Short cycle problem.

2.14.6 GCM

See *AES-GCM and breaking it on nonce reuse*[56]

End questionsss: 1. What happens if we use same IV for two different messages in CBC? It will result in the same ciphertext until the first different block.

2.15 ASymmetric Encryption

In symmetric encryption, the same key is used for both encryption and decryption. This means that anyone who wants to encrypt a message must have the key, and anyone who wants to decrypt it must also have the same key.

Now, imagine I want to visit a website. I can't simply send my encryption key over the internet to the site and say, "Here's my key; use it to decrypt my message." I would have to send the key in plain text, and anyone who intercepts my internet connection—such as people sharing the same Wi-Fi, the internet service provider (since my data passes through their servers), or anyone else along the communication path—could see the key in plain text. This means they could potentially decrypt my messages later on. This scenario introduces the need for a different type of encryption: *asymmetric encryption*.

Let me explain the method this way: imagine you have a box, a lock, and a pair of keys. The lock works in such a way that if you lock it with one key, it can only be opened with the other key.

[56] `https://frereit.de/aes_gcm/`

One of the keys I keep with me at all times; I'll call it my private key. The other key can be given to anyone; this is known as the public key.

Now, if you want to send me a message, I send you the lock and my public key. You write your letter, place it in the box, lock the box, and secure it with my public key. Then you send the box to me. Since the lock is locked with my public key, only I can unlock it with my private key. This way, even if someone intercepts the box, they won't be able to open it because they don't have my private key. If I want to send you a message, I would ask you to send me your lock and your public key. I write my letter, place it in a box, lock it, and secure it with your public key. Since your private key is only with you, only you can unlock it!

• Why do we even need symmetric encryption? We can just use asymmetric encryption. Even remote communication is possible.

→ Asymmetric encryption is very slow—thousands of times slower than symmetric encryption. We normally use asymmetric encryption to transmit or create a symmetric key on both sides. Then we use symmetric encryption to encrypt the message.

For now, assume that we can do two things:

- Encrypt the symmetric key with the public key and send it to the other side.
- Use an asymmetric algorithm to generate the symmetric key on both sides (Diffie-Hellman-Merkle Key exchange[57]).

2.15.1 Man-in-the-middle Attack

There is a problem with asymmetric encryption. Think about what can go wrong in the protocol. The issue is: how can you be sure that the public key you received is the real one? What if someone intercepts the connection and sends you their public key? The protocol goes like this:

1. Alice tell the server to send its public key.
2. The server sends its public key.
3. Mallory who is sitting between you and the server intercepts the connection. Gets the public key from the server and sends Alice her public key.
4. Alice encrypts the message with Mallory's public key and sends it to Mallory.
5. Mallory decrypts the message with her private key and reads it.
6. Mallory encrypts the message with the server's public key and sends it to the server.
7. The server decrypts the message with its private key and reads it.

[57] Read more at: `https://skerritt.blog/diffie-hellman-merkle/`

See? Mallory is now able to read the messages, while the server and Alice think that they are talking to each other. This is called a *man-in-the-middle attack*. What do you think we should do to detect or prevent it?

We know that public-key cryptography is very slow. Thus, if someone is intercepting, the connection should be slower than usual (because encryption is done twice and decryption is done once). So, one way is to use machine learning to detect unusual delays in the connection (considering the delay of the user's internet and its packets).

However, the above solution won't always work. Delays vary, and the mechanisms may not detect every situation. We want something that will always work. A better way is to trust an authority to verify the public key. This is called a *Certificate Authority*. The authority signs the public key of the server. When you receive the public key, you also receive the signature. You can verify the signature with the public key of the authority. If it is correct, you can trust the public key. If it is not, you can't trust it.

This method is used in HTTPS. Have you ever heard experts say that you should not install certificates from unknown sources? This is the reason. If you install a certificate from an unknown source, they can intercept your connection, and you won't know it. [58][59]

A country can promote its *National Browser*, which has a certificate from the government. If you install it, the government can intercept your connection.

There have been some cases where certificate authorities were compromised. See Diginotar[60].

This method needs a trusted authority. In many applications, we don't have a trusted authority. For instance, in a peer-to-peer network, WhatsApp, Signal, Telegram Secret Chat, etc., there is no authority to verify the public key. So, what do you think we should do?

We can build trust based on community agreement. For example, if Alice wants to talk to Bob, she can ask her trusted friends to send her Bob's public key. If all of them send the same public key, Alice can trust it. This is called a *Web of Trust*. It is used in *PGP* (Pretty Good Privacy) encryption. *PGP* is software that encrypts and decrypts messages and is used in email encryption.

Another approach is to generate hash of both public keys ($H(\text{Public}_{\text{Alice}}, \text{Public}_{\text{Bob}})$) in both parties. Parties can exchange the hash in another channel which is previously trusted. If the hashes are the same, it means that the public keys are the same:

- Alice side: $H(\text{Public}_{\text{Alice}}, \text{Public}_{\text{Bob}})$
- Bob side: $H(\text{Public}_{\text{Alice}}, \text{Public}_{\text{Bob}})$

Because if an attacker does MITM attack, The hash will be:

[58] Continuing to Protect our Users in Kazakhstan `https://blog.mozilla.org/netpolicy/2020/12/18/kazakhstan-root-2020/`

[59] Apple, Google, Microsoft, and Mozilla ban Kazakhstan's MitM HTTPS certificate `https://www.zdnet.com/article/apple-google-microsoft-and-mozilla-ban-kazakhstans-mitm-https-certificate/`

[60] `https://en.wikipedia.org/wiki/DigiNotar`

- Alice side: $H(\text{Public}_{\text{Alice}}, \text{Public}_{\text{Mallory}})$
- Bob side: $H(\text{Public}_{\text{Mallory}}, \text{Public}_{\text{Bob}})$

This is called *Fingerprint Verification*. It is used in *Signal*[61], WhatsApp[62].

Warning

As Techlore also suggest[a], if you want to verify the fingerprint and you don't trust the other party, do not send the fingerprint to the other party. Because the other party can simply resend it to you and say mine is the same as yours.

[a] *A Guide To Verifying Signal Safety Numbers*: `https://discuss.techlore.tech/t/a-guide-to-verifying-signal-safety-numbers/2289`

Now let me ask you a question! Are those services that claim they offer a secure way for you to send a message "anonymously" to another person real? (Like those Telegram bots.) Of course not! They get your plaintext message and store it in their database. Then they send it to the receiver. The owner of the server/bot has access to all the messages (and may even be selling them). Be careful about them!

2.15.2 Digital Signature

Remember I said a message encrypted with one key can be decrypted by the other key? What if I encrypt a message with my private key? People can try to unlock it with my public key. If it works, it means that I am the one who sent the message because only I have my private key. This is called a digital signature.

NOTE: It is not safe to use the same algorithm and key blindly for both encryption and digital signature [18]. Also *NIST* says[63]:

> . . . the key used for key agreement shall be different from the ECDSA key used for digital signatures

Remember we said asymmetric cryptography is slow? How can we use it for a digital signature? If we have a 10GB file, we need to encrypt it with our private key. That is very slow. What do you think we should do? **HINT:** To do it more quickly, we should encrypt less data. But how?

Instead of encrypting the whole file, we can just encrypt its fingerprint. Like humans, files can have a unique fingerprint. This fingerprint is called a *hash*. We can find the hash of the file, encrypt it with our private key, and send it with the file. The recipient can find the hash of the file and decrypt it with our

[61] `https://signal.org/blog/safety-number-updates/`
[62] `https://faq.whatsapp.com/1524220618005378/?cms_platform=web`
[63] 5.6.2 Using Algorithm Suites and the Effective Security Strength, Page 59: `https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-57pt1r5.pdf`

public key. If the decrypted hash is the same as the hash of the file, it means that the file is ours.

Legal issues are also important. You may need to change your algorithm for export to another country. Moreover, a country may only accept a digital signature under certain circumstances (e.g. acceptable protocols) and may not accept it in other cases in court.[64] What if a person alleges that their private key has been stolen?

2.15.3 RSA

For asymmetric encryption, we seek a function such that if you know some parameters, you can easily calculate the result. But if you don't know the parameters, it is very hard to calculate the result. For example, if you know two prime numbers p and q, you can easily find their product: $p \times q = n$. But if someone gives you n, it is very hard to find p and q. This is the basis of the RSA algorithm.

• Why is it hard? For example, if you give me 14, I'll easily tell you the two prime numbers behind it: 2 and 7!

→ For small numbers, yes, it is easy. But in RSA, the numbers are each 4096 bits! You can't factor a number that has hundreds of digits!

To date[65], there hasn't been a **general algorithm** found for **classical computers** to factor an integer n in polynomial time[66].[67]

• Wait a minute. I know a linear algorithm! Just start from 2 and go all the way up to the number. If it is divisible by a number, it is not a prime number. Even this easy algorithm is linear. How can you say that there is no polynomial algorithm?

→ Good question! In cryptography, we don't express the complexity in the number itself. We express it in the number of bits.

Any number can uniquely be expressed in base b like this:

$$n = a_0 + a_1 \times b + a_2 \times b^2 + a_3 \times b^3 + \cdots + a_k \times b^k$$

Where b is the base and a_i are the coefficients with $0 \leq a_i < b$. If $a_n \geq b$, we can divide it by b and add the remainder to the next coefficient. Moreover, $a_n > 0$ because if $a_n = 0$, we can also add $a_{n+1}b^{n+1}$ where $a_{n+1} = 0$ and continue adding zeros to the end of the number. In this way, the representation is not unique.

Now we want to find the number of bits of a number. We know that:

[64] Almost all digital signatures have a timestamp.

[65] It hasn't been proved that it won't be found. It just hasn't been found until now. Some mathematicians believe that factoring is and will be hard. But others think it will get much easier as time passes. But for now, it is hard.

[66] Polynomial time algorithms are considered efficient for our computers

[67] Yes, there are algorithms that use certain properties of a number (e.g. *Pollard's p-1*) and can factor it easily. But there is no such general algorithm.

$$\texttt{count} = \texttt{last} - \texttt{first} + 1$$

So:

$$\texttt{count} = n - 0 + 1$$

Thus, the number represented has $n + 1$ bits. We now need to find $n + 1$ in terms of the number (N). We know that:

$$b^n \leq N < b^{n+1}$$

If the number was less than b^n, it would be represented by a smaller power of b (b^{n-1}), and if it was greater than or equal to b^{n+1}, it would be represented by a greater power of b (b^{n+1}). We want to find $n + 1$. We can use logarithms to find it:

$$n \leq \log_b N < n + 1$$

Is it a little familiar to you? A number that is greater than or equal to n and less than the next number $(n + 1)$ is the definition of the floor function. So we can write it as:

$$n = \lfloor \log_b N \rfloor$$

Thus:

$$n + 1 = \lfloor \log_b N \rfloor + 1$$

We can also say:

$$N \sim b^n$$

Now say! Is your algorithm polynomial?! It is not. Something that is linear in terms of N is exponential in terms of bits.

- Why in algorithms, we don't express the complexity in term of bits?

→ Because there, they want to know how the algorithm behaves when the number grows. But in cryptography, operations are done one bit at a time.

2.15.4 Low-entropy Message Attack

Think about how we can use brute-force to find the message behind the ciphertext.

We can encrypt common sentences using the public key. If the encrypted message matches one of the ciphertexts transmitted, we'll know that the text behind it is that. To prevent this attack, we use something similar to salting. We add random bits to prevent the attack.

2.15.5 RSA Problems

It is not proved that the plaintext cannot be recovered without factoring.

In 1977, *Ron Rivest* (one of the inventors of the RSA) said[19] for factoring a 125-digit number, we need 40 quadrillion years of running power. Yet in 1994, researchers factored a 129-digit number![20]

A RSA-2048 doesn't provide 2048 bits of security. It provides only about 112 bits. This means bandwidth and computational power are wasted. This is why we use elliptic curve cryptography (ECC). ECC provides the same level of security with smaller key sizes. For example, a 256-bit key in ECC provides the same level of security as a 3072-bit key in RSA. [68]This means that ECC is faster and uses less bandwidth. A good explanation is on youtube by *Christof Paar*.

2.16 Randomness

Our computers are not designed to do random things. They are designed to run deterministic instructions. They cannot generate random numbers because they are not designed to do so. Many libraries use pseudo-random number generators to generate numbers that are not suitable for cryptographic purposes.

According to *Bruce Schneier*[21]:

> Random-number generators are another place where cryptographic systems often break. Good random-number generators are hard to design, because their security often depends on the particulars of the hardware and software. Many products we examine use bad ones. The cryptography may be strong, but if the random-number generator produces weak keys, the system is much easier to break. Other products use secure random-number generators, but they don't use enough randomness to make the cryptography secure.

`random` library from python itself says[69]:

> However, being completely deterministic, it is not suitable for all purposes, and is completely unsuitable for cryptographic purposes.

conversly, *secrets* module provides better randomness:

> The secrets module provides access to the most secure source of randomness that your operating system provides. [22]

And According to *PEP 506*:

[68]NIST Special Publication 800-57 Part 1, Revision 5: Recommendation for Key Management: `https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-57pt1r5.pdf`

[69]random — Generate pseudo-random numbers: `https://docs.python.org/3/library/random.html`

`secrets` will be based on `os.urandom` and `random.SystemRandom`, which are interfaces to your operating system's best source of cryptographic randomness. On Linux, that may be `/dev/urandom` on Windows it may be `CryptGenRandom()`, but see the documentation and/or source code for the detailed implementation details. [23]

Warning

Don't use unmaintained libraries. For example, *pycrypto* is not maintained anymore.[a][b]

[a] `https://www.pycrypto.org/`
[b] `https://github.com/pycrypto/pycrypto`

How random numbers are generated? There are two ways to generate random numbers:

- Pseudo-random number generators (PRNGs)
- True random number generators (TRNGs)

2.16.1 Pseudo-random number generators (PRNGs)

PRNGs are algorithms that use mathematical formulas or precalculated tables to produce sequences of numbers that appear random. They are not truly random because their output is determined by an initial value, known as a seed. If you know the seed, you can predict the sequence of numbers that will be generated. This is why they are called pseudo-random. They are not suitable for cryptographic purposes because they are predictable.
- Why do we use them?
→ They are fast. They are used in simulations, games, and other applications where true randomness is not required.

2.16.2 True random number generators (TRNGs)

TRNGs generate random numbers using physical processes rather than mathematical algorithms. They are based on unpredictable physical processes, such as electronic noise, radioactive decay, or thermal noise. These processes are inherently random[70] and produce numbers that are truly random. TRNGs are used in cryptographic applications where true randomness is required.
- Why don't we use them always?
→ They are slow. They are used in applications where true randomness is required, such as generating cryptographic keys, creating digital signatures, and securing communications.

[70] I am not an expert. Maybe there are some predictable patterns.

Sometimes, some sources are combined to generate random numbers. For example, the Linux kernel uses a combination of sources, including keyboard and mouse activity, disk activity, network activity, light sensors (including webcams), and other sources to generate random numbers. This is why some people say that moving the mouse cursor around the screen can help improve the security of your system while generating keys.

Reading List

- random.org[71]
- lotterycodex (Full of analysis data)[72]
- Random number generator attack[73]
- Ensuring Randomness with Linux's Random Number Generator[74]
- Could RDRAND (Intel) compromise entropy?[75]
- PCG, A Family of Better Random Number Generators[76]

2.17 An example of an E2EE system

Now that you know encryption and hashing, let's see how they are used in a real-world application. Let's take the example of *Tuta* email service. Tuta is an email service that provides end-to-end encryption (E2EE) for its users. This means that only the sender and the recipient can read the messages (end-to-end); no one else, not even Tuta, can read them. User passwords are combined with salt[77] and hashed with a password hashing algorithm (Argon2). The hash produces 256 bits[78].

This hash is used as the encryption key for AES-256. The message is encrypted with this key. The salt is transferred to the server. When you use another device to decrypt the messages, the server gives you the salt so you can combine it with your password for hashing.

So far, we have solved the problem of encryption. But what about the authentication problem? A person might need to log in to the email service from different devices. How can we be sure that the person is the same one who created the account? We somehow need to store a hash in the database. But wait a minute! If we store the previously discussed hash in the database,

[71] https://www.random.org/
[72] https://lotterycodex.com/
[73] https://en.wikipedia.org/wiki/Random_number_generator_attack
[74] https://blog.cloudflare.com/ensuring-randomness-with-linuxs-random-number-generator/
[75] https://crypto.stackexchange.com/a/10285
[76] https://www.pcg-random.org/
[77] Remember salt must be completely random for each user, and the size should be considered based on recommendations from authorities.
[78] https://datatracker.ietf.org/doc/html/rfc9106#name-argon2-inputs-and-outputs

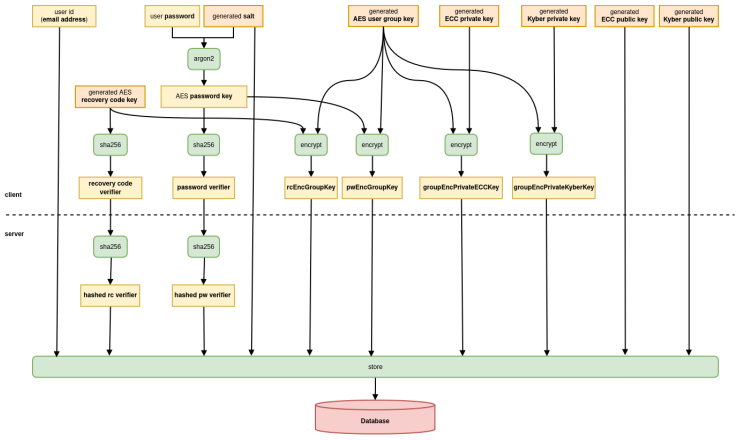


Figure 2.11: src: `https://tuta.com/encryption`

anyone who has access to the database can use it as an encryption key and can decrypt messages. So what should we do?

Tuta hashes this Argon2 hash with SHA2-256 and transfers it to its servers. If someone gains access to this hash, they need to brute-force 256 bits (as the key is 256 bits) and each time compare it with the hash of SHA2-256. As the key is random (remember the hash (here Argon2) is a random mapping of inputs to outputs), it is impossible to brute-force 256 bits to compare to SHA2-256. Why would someone bother to brute-force 256 bits to compare to SHA2-256 while they can brute-force keys for AES directly?

Then, another time, SHA2-256 is applied to the hash of SHA2-256, and the result is stored in the database.

Here both ends are users. In messengers, the ends are users communicating with each other.

Remember that developers should never access users' passwords. If something happens and hackers gain access to the server, they can use the passwords to log in to other services.

You may want to visit Bitwarden[79] for the solution they use.

You may also want to visit the Notesnook website[80] which interactively shows how an end-to-end application works. It's wonderful.

2.18 Post-Quantum Cryptography

Remember we just said that for **classical** computers, there has not been found any **general** algorithm to factor an integer n in polynomial time. But what

[79] `https://bitwarden.com/help/bitwarden-security-white-paper/#hashing-key-derivation-and-encryption`

[80] `https://vericrypt.notesnook.com/URL`

about **quantum** computers? They are based on quantum-mechanical phenomena, such as superposition and entanglement, which allow them to perform **some**[81] tasks in such a way that is faster to solve compared to classical computers. One such task is factoring. There are some algorithms[82] that can factor a number in polynomial time for quantum computers. This means that RSA will be broken. Of course, there are some details that we don't cover here. To date, quantum computers are not powerful enough to break RSA (or ECC) in the near future. But who knows? Maybe in 20 years, they will be powerful enough to break it. This is why we need to find new algorithms that are secure against quantum computers. This is called *Post-Quantum Cryptography*.

Should we be worried? No. Because NIST (and independent experts) is working on it. Post-quantum cryptography is hard. NIST itself says:

> NIST anticipates that the evaluation process for these post-quantum cryptosystems may be significantly more complex than the evaluation of the SHA-3 and AES candidates. One reason is that the requirements for public-key encryption and digital signatures are more complicated. Another reason is that the current scientific understanding of the power of quantum computers is far from comprehensive. Finally, some of the candidate post-quantum cryptosystems may have completely different design attributes and mathematical foundations, so that a direct comparison of candidates would be difficult or impossible.

But attackers and governments are collecting encrypted data and storing it. They can decrypt it in the future. This is why we need to be prepared for it. Remember we said that anything that goes onto the internet stays there forever and may end up in the hands of attackers later on.

However, our symmetric encryption algorithms won't be completely broken.[83]

Hash functions are also less affected by quantum computers.[24][84][85][86]

Don't get super excited about those apps that use post-quantum algorithms. They may be using weak algorithms. For instance, one of the candidates sent to NIST was broken by classical computers![87]

---

[81] One common mistake is that people think quantum computers are much faster and better at any task. No, they are not! They just solve problems in a different way.

[82] e.g. Shor's algorithm

[83] Quantum Computing and Cryptography: `https://www.schneier.com/blog/archives/2018/09/quantum_computi_2.html`

[84] BSI TR-02102-1: "Cryptographic Mechanisms: Recommendations and Key Lengths" Version: 2024-1: `https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/TechGuidelines/TG02102/BSI-TR-02102-1.pdf?__blob=publicationFile&v=7`

[85] `https://www.schneier.com/blog/archives/2022/08/nists-post-quantum-cryptography-standards.html`

[86] `https://www.nist.gov/cybersecurity/what-post-quantum-cryptography`

[87] `https://csrc.nist.gov/csrc/media/Projects/post-quantum-cryptography/documents/round-4/submissions/sike-team-note-insecure.pdf`

As Bruce Schneier[88] and NIST say, post-quantum cryptography is hard and current knowledge is far from comprehensive. For now, it's better to use quantum-resistant algorithms as an **additional layer** to our current algorithms. The *Signal* app is one of the apps that uses post-quantum algorithms, but it uses them as an additional layer:

> We believe that the key encapsulation mechanism we have selected, CRYSTALS-Kyber, is built on solid foundations, but to be safe we do not want to simply replace our existing elliptic curve cryptography foundations with a post-quantum public key cryptosystem. Instead, we are augmenting our existing cryptosystems such that an attacker must break *both* systems in order to compute the keys protecting people's communications.

Reading List

- A Gentle Introduction to Lattices and Lattice-Based Key Exchange: Part 1[89]
- A Gentle Introduction to Lattices and Lattice-based Key Exchanges: Part 2[90]
- Lattice-based cryptography[91]

2.19 MAC and HMAC

2.19.1 Message Authentication Code (MAC)

Think you want to send a message to somebody. How can your friend be sure that the message hasn't been changed?

One way is to use a digital signature. Hash the message, then encrypt the hash with your private key. On the other side, your friend decrypts the hash and compares it with the hash of the message that he calculated himself. If they are equal, it means the message hasn't been altered. But what if you don't want to use a digital signature? We somehow need to use hash functions, as they can help us detect even small changes.

One can say we can send the hash alongside the message. So instead of sending the message, we can send: `m, Hash(m)`[92]. How can this go wrong?

An attack in the middle can change the message, recalculate the hash, then replace the new hash with the existing hash and send it to the receiver. By doing this, the attacker successfully changed the message without the parties

[88] https://www.schneier.com/blog/archives/2023/08/you-cant-rush-post-quantum-computing-standards.html

[89] https://writing.chelseakomlo.com/gentle-introduction-lattice-crypto/

[90] https://writing.chelseakomlo.com/a-gentle-introduction-to-lattices-and-lattice-based-key-exchanges-par

[91] https://en.wikipedia.org/wiki/Lattice-based_cryptography

[92] `||` means concatenation

noticing. What do you think we should do in order to prevent the attacker from being able to do this? (Hint: Change the hash mechanism so the attacker can't replace it with their own hash.)

We can use something similar to salting here. Instead of sending the message, we can send: `m, Hash(secret || m)`[93]. Where `secret` is a shared secret between both parties. If an attacker wants to change the message, they need to know the secret in order to calculate the hash and replace it with the existing hash. Otherwise, the change will be detected.

One can go with `m, Hash(secret || m)`, while somebody else can use `m, Hash(m || secret)`. To have a standard mechanism that prevents certain attacks (e.g., length-extension attack), we can have a standard protocol to do the authentication. This is called *HMAC*.

2.19.2 Designated Verifier Signature

Think Alice is working for a corrupt company. She decides to blow the whistle on the corruption and tell the newspaper the whole story. The newspaper should know that the message came from Alice. Still, Alice should only be able to prove the originality of the message to the newspaper. She doesn't want the newspaper to be able to prove the originality of the message to anyone else. This is called a *Designated Verifier Signature*. It is like a digital signature, but the verifier is designated. The newspaper should not be able to tell others that the message is from Alice.

For this, we can use a symmetric MAC. The newspaper can be sure that the message is from Alice, but they can't prove it to others (since the MAC can be generated by both parties, as they both have the secret). See also *Stronger Security and Constructions of Multi-Designated Verifier Signatures*[25] and its presentation[94].

2.20 Homomorphic Encryption

Think you want to process your data, but you don't have enough resources. You want someone else to use their computational power to give you the answer. But what if the data is sensitive? Cryptography helps you give them transformed data without them knowing what the actual data was. They can give you the answer. You then retransform the answer and find the actual result.

Think you want to find the result of $100 + 200$. You apply the Caesar cipher with a shift of 40. So the two numbers will be 140 and 240. You give these two numbers to a third party to do the calculation. They find $140+240 = 380$. They give you 380. You then need to shift it by 80 (as each number was shifted by 40). Thus, $380 - 80 = 300$. Now your result is ready! You gave someone a question, they solved it without knowing the actual question, and you have the actual

[93] `||` means concatenation

[94] IACR, Stronger Security and Constructions of Multi-Designated Verifier Signatures`https://youtu.be/tJqK9zJ10iA`

result! This is a simple example of *Homomorphic encryption*. Another similar example can be privately outsourcing computation, which we talked about in the beginning of the chapter.

2.21 Shamir's Secret Sharing

Bruce Schneier, in his book *Applied Cryptography* [14] talks about a scenario:

> You're setting up a launch program for a nuclear missile. You want to make sure that no single raving lunatic can initiate a launch. You want at least three out of five officers to be raving lunatics before you allow a launch. How can you set up a system that will allow a launch?

This is a classic example of secret sharing. In this case, the secret is the launch code, and it is shared among five officers. The secret is divided into five shares, and each officer is given one share. The secret can be reconstructed only when at least three officers come together and combine their shares. This is a simple example of secret sharing. We will later discuss secret sharing in detail.

2.21.1 Secret Sharing

Secret sharing is a technique in cryptography where a secret is divided into multiple shares, and each share is distributed among a group of participants. The secret can be reconstructed only when a certain number of participants come together and combine their shares. The secret cannot be reconstructed when the number of participants is less than the required number. This ensures that no single participant can reconstruct the secret. Secret sharing is used in various applications, like nuclear launch codes, cryptographic keys, etc.

Definition 9. *A (k, n) secret sharing scheme is a method to divide a secret S into n shares such that the secret can be reconstructed when at least k shares are combined.*

In the physical world, secret sharing can be implemented using physical objects like keys, cards, etc. A lock can be opened if and only if a certain number of keys are used. In contrast, in the digital world, things are more complex.

2.21.2 Naive Approach

Definition 10. *Encryption is the process of converting a plaintext into a cipher-text using a key. Only the person who has the key can decrypt the cipher-text and recover the plaintext.*

Definition 11. *Security bits is a measure of the strength of a cryptographic algorithm. It is the number of bits required to break the algorithm. A cryptographic algorithm with n bits of security requires 2^n operations to break.*

A naive approach is to encrypt the secret using a key and then divide the key into shares. Shares can be distributed among participants. The secret can be reconstructed when the shares are combined and the key is recovered.

While this approach is easy to implement, it is completely insecure. According to *NIST*, if data should remain secure until 2030 and beyond, it should have at least 112 and 128 bits of security, respectively [26]. If our key has 128 bits of security, dividing it into 4 shares will reduce the security to 32 bits. Brute-forcing a 32-bit key can be done in a matter of seconds [14]. So, this approach is not secure.

2.21.3 Lagrange Interpolation

Suppose we have certain points $(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)$. The Lagrange interpolation formula is used to find a polynomial that passes through these points. Lagrange interpolation does this by first creating a set of polynomials $L_i(x)$ such that

$$L_i(x_j) = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases}$$

Because each Lagrange polynomial only contributes to its own point, we can multiply by the corresponding y_i to get the final polynomial:

$$L(x) = \sum_{i=0}^{n} y_i L_i(x)$$

How can we create a polynomial that is zero at n points and 1 at the $(n+1)$th point? Suppose we have points $x_0, x_1, \ldots, x_n$. The polynomial that is zero at all points except x_0 is

$$L(x) = (x - x_1)(x - x_2)\ldots(x - x_n).$$

Example 2.21.1. *Find the polynomial such that* $L_1(1) = 1$, $L_1(0) = L_1(2) = L_1(3) = 0$.

We can guess that the polynomial is

$$L_1(x) = a(x - 0)(x - 2)(x - 3).$$

At $x = 1$,

$$L_1(1) = a(1)(1 - 2)(1 - 3) = 2.$$

So, $a = \frac{1}{2}$. The polynomial is

$$L_1(x) = \frac{1}{2}x(x - 2)(x - 3).$$

How can we generalize this to any set of points? (Because we need to find a general formula.) Let our new guess be:

$$L_1(x) = \frac{(x-0)(x-2)(x-3)}{(1-0)(1-2)(1-3)}.$$

This will give us $L_1(1) = 1$ and $L_1(0) = L_1(2) = L_1(3) = 0$.
In this way, we can find a general formula for $L_i(x)$:

$$L_i(x) = \prod_{j=0, j\neq i}^{n} \frac{(x-x_j)}{(x_i-x_j)}.$$

Given n points $(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)$, the polynomial that passes through these points is:

$$L(x) = \sum_{i=0}^{n} y_i L_i(x).$$

With $n+1$ points, we can find a polynomial of degree n that passes through these points.

Proof. Let $L(x_i) \neq p(x_i)$ for some i. We know that:

$$p(x_i) = y_0 L_0(x_i) + y_1 L_1(x_i) + \ldots + y_i L_i(x_i) + \ldots + y_n L_n(x_n).$$

As $L_i(x_i) = 1$ and $L_j(x_i) = 0$ for $j \neq i$, we get:

$$p(x_i) = y_i L_i(x_i) = y_i.$$

So, $p(x_i) = y_i$ for all i. □

To prove that the polynomial is unique, we can use the following theorem

Theorem 2.21.1. *Any polynomial of degree n has at most n roots.*

Suppose that $L(x)$ is the Lagrange interpolation for $n+1$ points $(x_0, x_1, \ldots, x_n)$. Suppose there is another polynomial $q(x)$ of degree m where $m < n$ that passes through these points.

Then, $L(x) - q(x) = 0$ as both of them pass through $p(x_i)$ for all $i \in \{0, 1, \ldots, n\}$. But $L(x) - q(x)$ is a polynomial of at most degree n and has $n+1$ roots. This is a contradiction. So, $q(x)$ cannot exist.

Theorem 2.21.2. *The Lagrange interpolation formula finds the lowest degree polynomial that passes through the given points.*

2.21.4 Shamir's Secret Sharing

Background

- **Question:** How many lines pass through one specific point?
- **Answer:** Infinitely many.

- **Question:** How many lines pass through two specific points?
- **Answer:** One.
- **Question:** How many degree 2 polynomials pass through two specific points?
- **Answer:** Infinitely many.
- **Question:** How many degree 2 polynomials pass through three specific points?
- **Answer:** One.
- **Question:** How many degree 3 polynomials pass through three specific points?
- **Answer:** Infinitely many.
- **Question:** How many degree 3 polynomials pass through four specific points?
- **Answer:** One.

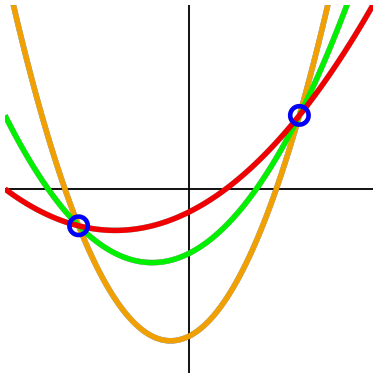


Figure 2.12: src: Vlsergey, 3 polynomials of degree 2 through 2 points, CC BY 3.0

We can see that with $n+1$ distinct points, we can find a unique polynomial of degree n that passes through these points.

Shamir's Secret Sharing Scheme

Adi Shamir introduced a secret sharing scheme in 1979 [27]. For (k, n) secret sharing, a polynomial of degree $k-1$ is created such that the secret is the constant term of the polynomial. The polynomial is evaluated at n points, and

the shares are distributed among participants. The secret can be reconstructed when at least k shares are combined.

What coefficients should we use for the polynomial? Computers can't store floating point numbers precisely. This may result in errors. So, we should use integers for the coefficients. Integer coefficients can lead to loss of perfect secrecy.

Definition 12. *Perfect secrecy is a property of a cryptographic scheme where the ciphertext reveals no information about the plaintext.*

Here, information about $k-1$ points must not result in any information about the secret. But using integers in the polynomial can result in information leakage. Consider the following example:

Example 2.21.2. *Let $f(x) = 1+x$. The secret is 1. Points $(3,4)$ and $(1,2)$ are given to Alice and Bob respectively. Alice can't find what the secret would be, but she can find what values can't be. She can find out that 2 is not the secret. If 2 was the secret, then the function would be $f(x) = 2 + \frac{2}{3} \cdot 3 = 4$. But the function does not have non-integer coefficients. So, 2 is not the secret.*

In this way, having partial information would result in information leakage.

Finite Fields

Finite fields have certain properties that make them suitable for cryptographic applications. The proof of the prefect secrecy of finite fields is beyond the scope of this paper.

Definition 13. *A finite field is a field that has a finite number of elements which operations like addition, subtraction, multiplication and division can be performed. properties of finite fields:*

- *Closure under addition, subtraction, multiplication and division. $(a, b \in F \implies a + b \in F)$*
- *Associativity of addition, subtraction, multiplication and division. $(a + (b + c) = (a + b) + c)$*
- *Commutativity of addition and multiplication. $(a + b = b + a)$*
- *Distributive property. $(a(b + c) = ab + ac)$*
- *Existence of additive and multiplicative identity. $(a+0 = a$ and $a \times 1 = a)$*
- *Existence of additive and multiplicative inverse. $(a + (-a) = 0$ and $a \times a^{-1} = 1)$*

Shamir's Secret Sharing in Finite Fields

Instead of using infinite field of integers which requires infinitely many bits and can result in information leakage, we can use finite fields.

Definition 14. *A finite field of order p is a field that has p elements where p is a prime number and is denoted by $\mathbb{Z}_p$.*

Example 2.21.3. *$f(x) = 1 + x \pmod 7$ which is a polynomial in $\mathbb{Z}_7$.*

Theorem 2.21.3. *If a and m are relatively prime, then a has a multiplicative inverse modulo m.*

Prime number is used as the order of the finite field because the multiplicative inverse exists for all elements in the field:

$$ax \equiv 1 \pmod p$$

Example 2.21.4. *Find the multiplicative inverse of 3 in $\mathbb{Z}_7$.*

We need to find x such that $3x \equiv 1 \pmod 7$. $3 \times 5 = 15 \equiv 1 \pmod 7$. So, the multiplicative inverse of 3 in $\mathbb{Z}_7$ is 5.

2.22 Zero-Knowledge Proofs

Think of a scenario where you want to prove that you know a secret without revealing the secret. For instance, you want to prove that you know the password to a system without revealing the password, or you know the solution to a Sudoku puzzle[95] without revealing the solution. This is where zero-knowledge proofs come into play.

Imagine we have a cave with a secret door in the middle. Alice knows the secret to open the door, while Bob doesn't know the secret. Alice wants to prove to Bob that she knows the secret without revealing it. This is a zero-knowledge proof. Alice goes to the cave and randomly chooses one side. Then ask Bob

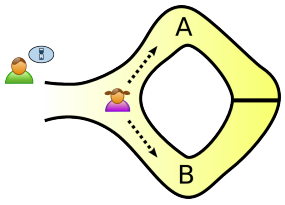


Figure 2.13: src: Dake, Zkip alibaba1, CC BY 2.5

to enter the cave. Then ask Bob to choose a side to Alice come out from. For example here, Bob chooses side A. If Alice was at side A, she can come out from

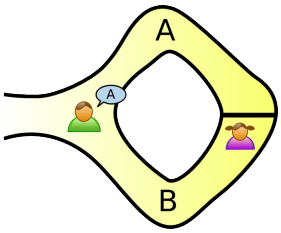


Figure 2.14: src: Dake, Zkip alibaba1, CC BY 2.5

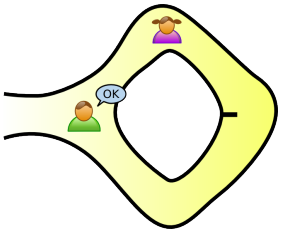


Figure 2.15: src: Dake, Zkip alibaba1, CC BY 2.5

side A. If she was at side B, she has to know the secret to open the door and come out from side A.

In this way, Alice can prove to Bob that she knows the secret without revealing the secret. However, Bob may think that Alice has entered from side A and, without knowing the secret, she can come out. There is a 50% chance that Alice has entered from side A. Bob knows to request that she repeat the process. If Alice can repeat the process multiple times, the probability that Alice doesn't know the secret and can come out from the side Bob has chosen is reduced. For the first time, the probability is $1/2$. For the second time, the probability is $1/2 \times 1/2 = 1/4$. For the third time, the probability is $1/8$. For the nth time, the probability is $1/2^n$.

If we do it 20 times, the probability that Alice doesn't know the secret and can come out from the side Bob has chosen is $1/2^{20} = 1/1048576$. The probability is very low. Thus, Bob can be sure that Alice knows the secret.

Zero-knowledge proofs have many applications. For instance, anonymous coins use ZKP. Or[28]:

> Currently, when a security analyst discovers a vulnerability in critical software system, they must navigate a fraught dilemma: immediately disclosing the vulnerability to the public could harm the system's users; whereas disclosing the vulnerability only to the software's vendor lets the vendor disregard or deprioritize the security risk, to the detriment of unwittingly-affected users. ... by using Zero

[95] Interactive Sudoku Zero-knowledge Proof: `https://manishearth.github.io/blog/2016/08/10/interactive-sudoku-zero-knowledge-proof/`

> Knowledge (ZK) protocols that let analysts prove that they know a vulnerability in a program, without revealing the details of the vulnerability or the inputs that exploit it.

2.23 Bit-commitment

Read about Bit-commitment in section "4.9 Bit-commitment" of the book "Applied Cryptography" by Bruce Schneier[14].

2.24 What to know?

- SHA2, SHA3: Used for authentication, check for similarity
- Argon2, Bcrypt, PBKDF2 Used for generating symmetric encryption key
- Symmetric Encryption: Used for fast encryption of data. Example: AES, TWOFISH, Serpent
- Asymmetric Encryption: Used for transfering/generating symmetric keys for both sides. Example: RSA, ECC, Diffie-Hellman

Cryptography is very hard. Let me quote a paragraph from *Cryptography Engineering*[1] book:

> That makes this a very dangerous book. Some people will read this book, and then turn around and design a cryptographic algorithm or protocol. When they're finished, they'll have something that looks good to them, and maybe even works, but will it be secure? Maybe they'll get 70% right. If they're very lucky, they may get 90% right. But there is no prize for being almost right in cryptography. A security system is only as strong as its weakest link; to be secure, everything must be right. And that is something you simply can't learn from reading books.

For instance, while working with encryption keys, many things can go wrong. Operating systems use temporary files, hibernation files, and swap files to keep track of changes for cases like power cuts or RAM management. An encryption key might get stored on the hard disk and never be deleted! You may work with an encryption key in a function and not clear its location in the RAM before exiting the scope of that function. This is very dangerous. The location can be given to another process, and that process can access the encryption key stored there. With high-level programming languages, it is sometimes impossible to clear the RAM correctly. With languages like C, you can, of course, use something like the `memset` function. But wait a minute! What if the compiler sees that an array is zeroed and never used again and then optimizes it to avoid doing that?! If a person is writing code for critical systems, they should read assembly code generated by the compiler.

Who has access to encryption keys? The CEO? Does the CEO have enough knowledge about security? Can the CEO protect it well enough? What if the CEO dies? Many times, parameters that can then be used to recreate the encryption key are distributed among people (e.g., Shamir's Secret Sharing).

Imagine a situation where, during the encryption process, a crash happens. Your data may be half-encrypted or may become corrupted. You may not be able to decrypt it.

There are many things that can go wrong in cryptography and security. Do **NOT** implement your self-made security mechanisms. Simply don't!

Reading List

- *cryptocoding v2* Presentation[96] by JP Aumasson
- Cryptocoding[97] ("coding rules" for implementations of cryptographic operations, and more generally for operations involving secret or sensitive values.)
- Are garbage-collection programming languages inherently unsafe for use in cryptography[98] (Didn't fully read it but seems related)
- Cryptofails[99] (Really interesting)

2.25 End of Chapter Questions

1. Why we prefer to use different symmetric keys for each session (a session is a period of time where two systems communicate with each other) instead of using the same key for all sessions?

1.a. If one session's key got leaked (e.g. a hacker stole it), the other sessions' keys are still secure. It is important that because of a problem, Past and future data don't be decrypted. Read about *Heartbleed* bug[100] and *perfect forward secrecy*.

[96] https://www.aumasson.jp/data/talks/cryptocodingv2_zn14.pdf
[97] https://github.com/veorq/cryptocoding
[98] https://crypto.stackexchange.com/questions/113619/are-garbage-collection-programming-languages-inherent
[99] https://www.cryptofails.com/archive
[100] Heartbleed: https://en.wikipedia.org/wiki/Heartbleed

3.0 Hardware Security

3.1 Physical Access

The most important thing you always have to remind yourself is that physical access is equal to getting hacked. If I have physical access to your computer, I can do anything with it. I can simply download and install a backdoor!
• How can you download one?
→ I can upload a password-protected zip file to cloud storage and download it from your computer. (Why a zip file? I'll explain it after I talk about encryption.)

• No. My system requires a password for installing software.
→ For most operating systems, there are portable versions of software. I can use that feature!

• No! My system doesn't have an internet connection!
→ I can use a USB drive!

• I have an anti-virus!
→ Many targeted malwares[1] are not detected by anti-viruses. Some are embedded into the firmware and cannot be removed after formatting! If the target worth 100,000$ and the malware costs 10,000$ to develop, it is worth it!
→ I can quickly code a program to spy on you. Many systems have some programming language compilers/interpreters installed.

• My system is locked. You cannot run the portable software.
→ Malwares are not the only way to hack. Most hacks are done via vulnerabilities. I can use a vulnerability to get access to your system.
→ I can use a live USB to boot your system and access your data. Many operating systems have a feature to boot from a USB drive. This means you can download it and boot your system from it. It is like a normal operating system, but it runs on your USB drive. The only thing you may need to change is the priority of booting devices in the BIOS. With this, I can access the data on your

[1] If you don't know what malware is, just read it as a virus. We will cover it later.

hard drive.

• I have my BIOS encrypted. [2]
→ No worries :) I can physically remove your hard drive and connect it to another computer. I can access the data on it.

• I have encrypted my important files.
→ How about your swap files, temp files, log files, and other hidden files? Full disk encryption must be used alongside file encryption. If only file encryption is used, the data in the hidden files can be accessed. If only full disk encryption is used, after decrypting the disk, all data is in danger.

• I have encrypted my hard drive.
→ Maybe your hard drive password is the same as your login password. I can use that password to access the data.

• No. I have never used this password for anything else. It is long and random.

→ I can use a hardware keylogger. It is a small device that is connected between the keyboard and the computer. It records all the keystrokes and sends them via a SIM card to me.

• I don't let anybody access my computer or attach anything to it.[3]
→ You may leave your laptop in the hotel when you're going out. The hotel staff can access your data.[4]
→ How are you sure that no hidden camera is installed in your home/hotel room?
→ I can gift a gaming keyboard to you for your birthday. It has a feature to record the keystrokes! I can use that to find your password!

• No, I don't accept gifts from strangers.
→ How are you sure that nobody from your friends won't do it? How are you sure that the keyboard you bought from the store or online shop hasn't targeted you?!
→ I can invite you to a café that belongs to my friend. I can later watch the camera footage to find your password. Not only that, many cameras are insecure and can be accessed by hackers. This include your CCTV camera. So you should not type your password where there is a camera. Even your home camera can be accessed by hackers.

[2] See also: BIOS Password Backdoors in Laptops: `https://dogber1.blogspot.com/2009/05/table-of-reverse-engineered-bios.html`

[3] It's important not to attach anything you're not sure about to your computer. Some public charging stations have a device that can install malware on your phone. Search about it on the web.

[4] See also: When in China, don't leave your laptop alone `https://www.infoworld.com/article/2279646/data-security-when-in-china-don-t-leave-your-laptop-alone-2.html`

→ I can act as a support agent and ask you to install software to help you (mostly remote access software). I can use that software to access your data. If I cannot do anything, I can simply destroy your computer

You delete a sensitive file and think it's gone. But it is not! Only the pointer to the file for the operating system (OS) is deleted. This means that the OS no longer knows if a file is there and where it is. But if we scan the disk with special software, we can find the file and recover it. It's not that hard. Files have special structures. Just open a file with a text editor, and you will see that it has a structure. Software can search for these structures and recover them.

Wow! How can we prevent it? You need to change the actual data of the file on the disk. This means you need to overwrite bits of the file with something else. There are many tools that can do this.

- How should we overwrite it?

→ I don't have enough knowledge (neither do many people who falsely claim they do). In some old devices, a single pass of zeros is not enough. See section *2. Methods of Recovery for Data stored on Magnetic Media* from [29]. However, we read from section *2.4 Trends in Sanitization* from *NIST NIST SP 800-88 Rev. 1*:

> For storage devices containing *magnetic* media, a single overwrite pass with a fixed pattern such as binary zeros typically hinders recovery of data even if state of the art laboratory techniques are applied to attempt to retrieve the data.

See also *Can data be recovered from a zero-filled hard drive?*[5], which mentions quotes from different sources. However, the only way to be sure is to destroy the hard drive into really small pieces. Hard disks are not expensive. If your data is worth 10 thousand dollars, you better spend 20 dollars to buy a new hard drive than live with the risk of someone recovering your data.

3.2 What to do?

- Set a password for your BIOS.
- Use secure boot.
- Encrypt your SSD/hard drive.
- Use a strong password.
- Don't type your password where there is a camera.
- Disable your computer ports.
- If you have a CCTV camera, be careful that many of them use weak protocols and can be hacked.

[5] `https://tinyapps.org/docs/recovering_data_from_zero_filled_hard_drive.html`

- Don't let anybody access your computer, even for one second. (They may be forced to do bad things.)
- Use tamper-evident technologies. [6]
- Don't do encryption/decryption of sensitive data in public (because of side-channel attacks. We will talk about it later.)
- Some security mechanisms are implemented in the hardware. For example, AES-NI is a set of instructions that can be used for encryption and decryption. This prevents some side-channel attacks.
- Many employees are careless. They may through away sensitive data (including passwords) in the trash. Some hackers search the trash to find sensitive data. Always shred sensitive data.

3.3 End of Chapter Questions

1. Why we prefer disk-encryption over file-encryption?
   1.a. Although file encryption is useful and is good to do so, it is not enough. Because you can't be sure about temp files, swap files and other important hidden files.

[6] See also: `https://en.wikipedia.org/wiki/Tamper-evident_technology`

4.0 Operating System (OS) Security

4.1 What is an Operating System?

Think of a big restaurant. You have many robots that are chefs. If two robots want to access the refrigerator at the same time, they will crash. So you need a manager to manage them—deciding when to give access to the refrigerator to one and when to give it to another. This is the job of the operating system.

You have hardware. Programs need to access the hardware. But if two programs want to access the hardware at the same time, what should we do? When should we give the CPU to one and when to another? This is the job of the operating system.

Operating systems manage the memory (RAM). They take control of what part of the memory should be used by which program. They free the memory when a program is closed.

The OS provides an API (Application Programming Interface) to the programs. This means that programs don't need to know how to access the hardware. They just need to call the OS functions. The OS will take care of the rest. It also helps to make the programs portable. They don't need to know the hardware details. In addition to functionality, it also provides security. It doesn't allow programs to access the hardware directly. Accessing the hardware directly is dangerous. Programs might inadvertently or intentionally harm our hardware. They might access the data of other programs (which may be sensitive).

4.2 OS security

The operating system is the core of the functionality of a computer. So its security is very important.

I am not going to talk about the security of the operating system itself. The topic is very broad, and I don't have enough knowledge to discuss it. I am going to talk about the security of the operating system for the user or features and programs that can help you secure your data related to the operating system.

We have many operating systems. The most famous ones are Windows, macOS, GNU/Linux, BSD, Android, iOS, etc. Each of them has its own security features.

If you want to improve your security mindset, you have to think like an attacker. Now think of yourself as an attacker. Would you write a bad program for Windows or for GNU/Linux? Would you rather target hundreds of millions of users or a few million users? You would probably choose Windows because it has more users. Also, many Windows users are average people who don't know much about security (if they know at all!). So it is more likely that you can find a victim.

This is one of the reasons that people mostly say Windows is insecure. I don't have enough knowledge to talk about the security of the operating system itself. Neither do most people who blindly say Windows is insecure. They just say it because they have heard it from others. Just ask them how Address Space Layout Randomization (ASLR) works, and you will see that they can't explain it in depth. Just take advice from those who are really experts in the field.

Windows **home users** are the most targeted for **common attacks**. This doesn't mean GNU/Linux is secure; it means that it is less targeted for **common attacks**.

4.3 What OS to use?

Before that, let's talk about updating programs. Many people ask, "I updated WhatsApp, but it doesn't change. Why should I update?"

Think of it this way: you have a house. The person who built it regularly comes and checks the house and fixes the problems. Many times, you don't see the problems and changes. But the person who built it sees them and fixes them. It is the same with programs. The developers of the programs regularly check the program and fix the problems. Many times, you don't see the problems and changes, but the developers see them and fix them. So you should update your programs regularly. Updates also bring new features, like changing the color of the main page of a program. This is something you will see.

• I am afraid that the new version has a bug and doesn't work properly.

→ It is possible. I don't say it is not. But it is less likely because the developers have tested the program before releasing it. Also, if a bug is found, they will fix it in the next version. But if you don't update, you will be vulnerable to the bugs that are already fixed, like a thief who knows that you have a hole in your house and can enter it.

Some people say to use Windows 7 because it has received many updates over the years. But they don't consider new bugs! What if a bug is found? It won't be fixed. You will remain vulnerable forever.

Have separate accounts for your work and personal life. Don't use the same account for both. Some people use just one account (an admin account) for everything. This is dangerous. If a program is infected, it can access all your

data.

Don't give admin accounts to employees. Employees should not use company computers for personal tasks. Even a simple quick payment of bills may become a security problem for the company.

4.4 Malware

In the beginning, because viruses were the first type of malware, the term "virus" became very well-known. This led to the creation of antivirus software. However, it's more accurate to use the term "malware," as viruses are actually a specific type of malware. Malware is a general term that refers to "any software intentionally designed to cause disruption to a computer, server, client, or computer network, leak private information, gain unauthorized access to information or systems, deprive access to information, or which unknowingly interferes with the user's computer security and privacy."[1]

Why is knowing about malware important? Because it is the most common way of harming a computer and stealing data (e.g., banking information).

Furthermore, certain countries have been affected by malware more than you think. Every year, Kaspersky releases statistics on the users of its antivirus software, and each year, Iran ranks as the country most affected by mobile malware! 2017[2], 2018[3], 2019[4], 2020[5], 2021[6], 2022[7]

This is because of certain reasons (I will explain it more in the future):

- The use of insecure marketplaces
- Downloading apps from unofficial sources, such as Telegram channels!
- Value-Added Services (VAS) that are designed to infect users' devices

[1] From Wikipedia Creative Commons Attribution-ShareAlike 4.0 License
[2] https://securelist.com/mobile-malware-review-2017/84139/
[3] https://securelist.com/mobile-malware-evolution-2018/89689/
[4] https://securelist.com/mobile-malware-evolution-2019/96280/
[5] https://securelist.com/mobile-malware-evolution-2020/101029/
[6] https://securelist.com/mobile-malware-evolution-2021/105876/
[7] https://securelist.com/it-threat-evolution-in-q1-2022-mobilestatistics/106589/, https://securelist.com/it-threat-evolution-in-q2-2022-mobile-statistics/107123/, https://securelist.com/it-threat-evolution-in-q3-2022-mobile-statistics/107978/

NOTE

When I mention the names of different malware, I don't want you to try memorizing them. My main goal is for you to understand their functions so you can recognize what might threaten you. Memorizing names isn't useful; we need to know how they operate. What's important is to identify that malware can do this, that, and the other thing, which means I need to be cautious in these areas. Additionally, most modern malware is a combination of several types, such as being a keylogger, ransomware, and spyware all at once. Names are more relevant for those who want to delve deeper into research or whose work is specifically related to this field.

4.4.1 Virus

A virus is a type of malware that can replicate itself and spread to other computers. It can attach itself to a program or file, and when that program or file is executed, the virus is activated. It can spread through email attachments, infected files, or even USB drives.

A prime example of a virus is the *Stuxnet* virus. It was designed to target Iran's nuclear program. It spread through USB drives and infected the systems that were connected to them.

Stuxnet is a sophisticated computer virus discovered in 2010, designed to target Iran's nuclear facilities, particularly the centrifuges at Natanz. Developed by the U.S. and Israel, it was based on footage taken during a past visit to the site. The virus did not operate in any system except the one it was designed for. I highly recommend reading Langer's analysis[8].

The lesson we can learn from Stuxnet is that when dealing with sensitive systems, no information about them should be disclosed. In the case of Stuxnet, even background noise could reveal the rotation speed of the motors. Any information about the system, such as the software version, can lead to the exploitation of vulnerabilities in that specific version for an attack. Targeted attacks use small pieces of information to exploit vulnerabilities.

After Stuxnet, there was a significant media frenzy. Amidst this, scammers attempted to gain positions and promotions by making false claims. For instance, the news about the discovery of the Stars virus was one such example, which is claimed by Webamooz to be a scam[9].

The important takeaway is that after such attacks, some individuals exploit the situation, trying to present themselves as heroes amidst the chaos and hype. We must be cautious and not let our fear of attacks lead to misplaced trust in those who claim to have solutions. It's essential to remain vigilant and not fall for scams.

[8] `https://www.langner.com/stuxnet/`

[9] `https://t.me/webamoozir/4286`, `webamooz.com/?p=20294`

4.4.2 Ransomware

In simple terms, ransomware is a type of malware that, once it gets into your computer, locks all your files. Then it demands money to give you the password to unlock them! For example, one day you might wake up and find that all your files are locked and you can't access them! This can lead to significant losses!

Background (Encryption)

Behind everything on your computer, there is a series of text (to be more precise, a series of 0s and 1s). The computer reads this specific text to understand the file and display it for you.

Was that a bit confusing? No problem! Let's explore what it means that there's text behind everything. I have a picture. I mentioned that there's text behind everything, right? So how can I see that text? Think about it for a moment! Instead of opening this picture normally, I can use something that can read the text to see what's behind this image! What can read the text? Text editor software like Notepad! Normally, double-clicking opens the image, but I can right-click and choose to open it with Notepad.

Now, the computer sees this text and understands it needs to display a certain type of image. But what does this have to do with encryption? Think about it! If I change this text, the computer won't be able to understand what the original image was anymore because the text has been altered! For example, instead of each letter, I can write the next letter in the alphabet. So, A becomes B, B becomes C, and so on. Now, when I open the image, it will be completely different! This is the basis of encryption. It's like a secret code that only you and the computer know. If someone else tries to read it, they won't understand anything!

That's why the altered files won't open. This is exactly the technique that ransomware uses. Ransomware is a type of malware that enters your computer and changes the text behind your files. Essentially, it encrypts them, making your files inaccessible! Then it demands, for example, $300 to give you the key to unlock your files! For instance, they might display messages like the one below:

Or, for example, they might not show a message, but instead, they place a TXT file next to your files. When you open it with Notepad, you can read the text inside.

Sometimes, they lie and say they are police or the FBI to scare you into paying quickly. They claim that your computer has been locked due to illegal activities or illegal downloads, and that you need to pay a fine:

Or sometimes, they say your files are locked because you have not paid for licensing.

Sometimes they don't encrypt files once they enter your computer. Instead, they first steal your files and then threaten to publish them if you don't pay.

But it doesn't stop there! They also set a deadline for you. For example, they might say that if you don't pay within a week, all your data will be deleted

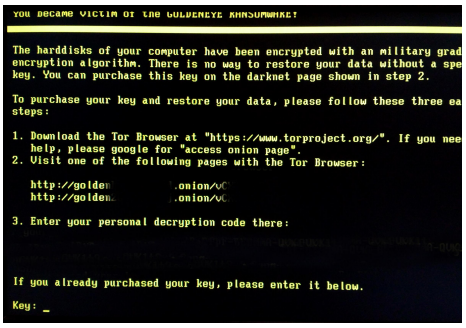


Figure 4.1: src: BlueBreezeWiki, Goldeneye-ransomware-161212, CC BY-SA 3.0

Figure 4.2: src: Motormille2, Ransomware-pic, CC BY-SA 4.0

or leaked. Many companies have faced this situation where an attacker wanted money to not publish sensitive data.

- Should I pay to get my files back?

→ Absolutely not!

How can you be sure that after paying, the attacker will actually fulfill their promise and provide you with the decryption key and tools? How do you know that someone who has already encrypted your files and demanded money for it won't use that decryption tool to install spyware on your system?[10] What if they say, "Well, I've also planted malware, so you need to pay for that too!" Would you trust someone who is an attacker and has already harmed you?

Moreover, this leads to the proliferation of ransomware. If people pay, at-

[10] Sometimes, the decryption tool they provide you is actually just another piece of malware! You can't trust someone who is an extortionist!

That's why companies that fall victim to ransomware, even if they manage to obtain a decryption tool, first test it on a specific computer to ensure it isn't malware. They simulate the environment on a series of computers that resemble normal, real systems, rather than using an isolated computer. This is because some types of malware check for certain system parameters (like system sensors and files) to see if they are running in an isolated environment. If they detect that they are in an isolated environment, they may not perform any harmful actions to avoid detection.

tackers will continue to use this method. If people stop paying, attackers will stop using this method. Why? Because they see that they can make money this way, so they are more likely to continue. If they knew that no one would pay the ransom, why would they waste their time creating ransomware when it wouldn't generate any income?

This is serious. We have something called *ransomware as a service*. Some will create ransomware and sell it to others so they can infect computers. They will take a percentage of the ransom. This is a serious problem. It is like a business.

The best way to counter ransomware is to have backups.

• Well, I have a backup! So they can do nothing!

→ First, your backup needs to be reliable. It shouldn't be the case that you just say, "I have some copies of my files on my hard drive, and I call that a backup!" because that could also be infected! Second, this is not enough. The attacker can still threaten to publish your data.

Your sensitive data should be encrypted. So if they steal it, they can't read it. **Some Tips About backup**

- Any digital device might stop working at any moment, so backups must be regular.
- Backups should be regular. It is not enough to back up once a month.
- Data should be encrypted before backup.
- Backups should be stored in different places. For example, one at your home and one at your office. If your home is burned, you can still access your data.
- Backups should be in different formats. For example, one on a hard drive and one in cloud storage.
- If you're running a business, you should be able to recover your data in a short time. For example, if your data is deleted, you should be able to recover it in a few hours. (The backup power lines are different. This way, if one area loses power, another area will still have electricity.)

What to do if you're infected with ransomware? Do NOT trust random decryption tools. Many videos on YouTube recommend downloading a tool and running it to get your files back. But many of them are actually malware! Search for recommendations from security experts and antivirus companies.

No More Ransom[11] is an initiative that brings together law enforcement agencies and cybersecurity companies to assist individuals and organizations affected by ransomware attacks. By providing valuable resources, including decryption tools and guidance on how to respond to such incidents, No More Ransom aims to empower victims to recover their data without paying ransoms.

[11] `https://www.nomoreransom.org/`

- How do they can decrypt the data?

→ Many ransomwares use bad cryptographic implementations. This means that the encryption is not strong enough, and it can be decrypted. Some of them use weak keys. Some don't clear encryption keys from memory. This way the key can be recovered. This is the reason why sometimes we suggest not to turn off your computer. If you turn it off, the key will be cleared from memory. (As RAM is volatile memory, it loses its data when the power is turned off.)

A Talk to Those Who Create Ransomware

If you are a ransomware creator, I want to talk to you. You may think that you are doing a good job. You may think that you are making money. But you are not! You are harming people.

Ransomware can destroy files that represent years of hard work and dedication for individuals and businesses. This loss can have devastating consequences, not just financially but also emotionally, as people may lose irreplaceable memories, important documents, or critical data. Additionally, in sectors like healthcare, ransomware attacks can have life-threatening consequences. For example, if a hospital's systems are locked, patients may not receive the care they need, and lives could be lost.

Crimes like creating ransomware leave a lasting mark in law enforcement databases. No matter how clever you think you are at hiding your tracks, the reality is that law enforcement agencies are continuously improving their methods and technologies to track down cybercriminals. Eventually, the evidence will catch up with you, and you will be held accountable for your actions.

Even if you use cryptocurrencies and Tor, it's still possible for law enforcement to track you down. Just as there are search engines like Google, there are also powerful tools specifically designed to locate cybercriminals. This is because organizations and individuals dedicated to tracking down cybercriminals—beyond just law enforcement—have financial incentives to do so. When they successfully monitor and identify you, they often receive increased funding for their efforts. They can analyze even the smallest clues to trace your activities and ultimately identify you. Engaging in cybercrime is a risky endeavor, and the chances of being caught are higher than you might think. You have to live every day in fear of being caught. The golden age of cybercrime is over. Right now, you might be around 18 or 19 years old and feel invincible, not fully understanding the consequences of your actions. But when you reach 40, you'll realize the impact of those choices and how they can affect your life in ways you never anticipated. I will show you how people are caught in the future. (Section Tor)

4.5 Trojan

One of the legendary wars of the Greeks was the siege of a city called Troy, which lasted for ten years. Eventually, they came up with a clever plan. They built a massive wooden horse and hid a few soldiers inside it. The rest of the army pretended to retreat. The people of Troy, believing they had won, thought the giant wooden horse was a gift from the Greeks and brought it into their city. Later that night, the soldiers hidden inside the horse quietly emerged and opened the gates, allowing the Greek army to enter and conquer Troy.

In the world of computers, a Trojan horse operates in a similar way. It often appears as a legitimate program, tricking users into thinking it's safe. For example, it might look exactly like Firefox, but it secretly performs malicious actions in the background. Essentially, a program is created that mimics Firefox, but with additional harmful code embedded within it, allowing it to carry out destructive tasks without the user's knowledge. That's why we advise downloading programs from official sources. If a program has a checksum (which we discussed in the hash section), make sure to verify it to ensure that the file hasn't been tampered with.

Reading List

- GhostBuster-Detecting arbitrary persistent and stealthy software[12]
- Forged Memory-A scary development in rootkits[13]

4.6 Keystroke Logger (Keylogger)

Imagine there's a program that records everything you type and sends it to a hacker. This poses significant risks, especially when you're entering sensitive information. For instance, if you type in your password, that password gets recorded and sent to the attacker. This is extremely dangerous! It could even capture your bank password, leading to serious financial consequences. These types of programs are known as keyloggers.

Keyloggers primarily target physical keyboards, which is why, during online payments, websites connected to payment gateways often provide a virtual keyboard with random numbers. They recommend using this virtual keyboard because if you have a keylogger that is monitoring your physical keyboard, it won't be able to capture what you enter on the virtual keyboard. (This doesn't mean that virtual keyboards are 100% secure. They can also be monitored by malware, but it's less likely.)

Don't use keyboard software that you're not sure about. They can be keyloggers. For example, if you're using software that changes the color of your keyboard, it might be a keylogger.

[12] https://www.schneier.com/blog/archives/2005/02/ghostbuster.html
[13] https://www.schneier.com/blog/archives/2011/05/forged_memory.html

4.7 Adware

Adware is a type of malware that displays unwanted advertisements on your computer. It's like a person who follows you around all day, constantly showing you ads. It's very annoying! Adware can also slow down your computer, making it difficult to work efficiently. It can consume a lot of your internet bandwidth, which can be a problem if you have a limited data plan.

Sometimes ads are fake, and if you click on them, you might be redirected to a malicious website.

How can you know you're infected? Seeing ads that you didn't see before is a sign of adware. Also, if you notice a lot of pop-up windows or changes in your browser, like a new homepage or a new search engine, that could indicate an infection.

4.8 Spyware

Spyware can capture a wide range of personal information, not just photos or screenshots of your chats. It can track your location, record your conversations through the microphone, take pictures using your front camera, and essentially invade your privacy in numerous ways. Any activity that violates your personal space and intrudes into your private matters can be considered spying.

Attention! Having access to certain features does not necessarily mean that an app is spying on you. For example, a photography app obviously needs access to the camera; without that permission, it wouldn't be able to take photos at all.

However, if a calendar app requests access to your camera, it raises a red flag. Why would a calendar app need to access the camera? This is where we need to be cautious about granting unnecessary permissions to applications.[14]

That's why developers should clearly explain in their app descriptions why they need specific permissions. For instance, if an app requires camera access for scanning QR codes, it should state that explicitly. Transparency about permissions helps users make informed decisions and protects their privacy.

You might think that since an app is reputable, it's safe to grant it all permissions. But even well-known companies have violated people's privacy in the past. This brings us back to the principle of "my data, my ownership." Your data belongs to you, and there's no reason to give it away unnecessarily.

For example, if you don't plan to scan a QR code, why should you grant camera access? Allowing access just for the sake of it means that the camera could sit idle, potentially being misused later.

[14] Doctor Web: Android Trojan controlled via Telegram spies on Iranian users: `https://news.drweb.com/show/?i=11331`

4.9 Botnet

Sometimes, you might become infected with malware that effectively turns your computer into a "zombie."

What does that mean? Think of a zombie as a creature that lacks independent thought and is controlled by someone else. It simply follows commands without any reasoning. Similarly, when your computer is infected with malware, it can be manipulated to perform actions you don't want it to.

On a more technical note, the servers that the malware connects to for receiving commands and reporting back to the hacker are known as "Command & Control Servers (C&C)." Typically, the victim's computer connects to these servers and informs them that it is compromised, asking, "What should I do now?" The server then sends back instructions, telling the infected computer what actions to take.

So, what characteristics should these servers have? When a program connects to a server, monitoring tools can track which website it connects to. This poses a significant risk for malware developers. Just as there are thieves, there are also police, along with malware analysts, security researchers, and law enforcement agencies. These security researchers analyze network packets to find ways to stop malware and trace it back to its original creator.

If a malware developer directly embeds their personal server address in the code or connects to their own server, it becomes easy for authorities to track them down. They could be questioned about why commands are being sent from their server to infected computers, instructing them to perform certain actions.

Additionally, antivirus software will block that server, and services like Google Safe Browsing or DNS providers may also prevent access to it. This is where malware developers employ various techniques to obscure their server's identity and avoid detection. They might use a series of *compromised* servers, known as a "botnet," to relay commands. This way, even if one server is blocked, the malware can still receive commands from another server in the botnet.

If the C&C server always has a fixed address, it would quickly be detected and blocked by antivirus software and web browser protections. Therefore, the C&C address needs to change frequently. However, if the malware developer includes a list of addresses in the code, those would also be discovered and blocked. So, the address must be dynamic and constantly updated. But where do they get this changing address from?

One approach is to create a specific website where the current C&C server address is posted and regularly updated. The malware would then only need to include this single website address in its code, and it would go to that site to retrieve the latest C&C server address.

However, you might wonder how this is different from directly embedding the C&C server address in the code. In essence, it's not much different, as that website could also be blocked easily.

This is where malware developers employ a clever technique. They might

place the address of the website in the comments of a popular post, like one about Britney Spears. The malware would then search through the comments to find the C&C server address.

But wait! It's not that simple! You might ask how the malware knows which address to extract. This is typically done using a predefined format or pattern that the malware is programmed to recognize. For example, the malware could be designed to look for specific keywords or structures in the comments that indicate the presence of the C&C server address. This way, even if the address changes frequently, the malware can still locate it by searching for the right cues in the comments.

In this method, the malware doesn't directly place the address in the comments (because the page admin would likely delete it). Instead, it hides the address within a text that, when a specific hash (not a cryptographic hash, but think of it more like a hash table) is calculated, results in the value 183.

For example, the comment text might be:

```
#2hot make loved to her, uupss #Hot #X
```

The custom hash for this text is 183.

The victim's computer will then go through the comments under the Britney Spears post and calculate the hash for each one (using that specific hash function). If the hash equals 183, the comment is found. Now, how do we convert this text into a link?

This is where regular expressions (Regex) come into play. To briefly explain, Regex is used to search for a string within another string. For example, if we write:

```
gr(e|a)y
```

We're looking for strings that start with "gr," followed by either "e" or "a," and ending with "y." This would find both "grey" and "gray."

The Regex used here is:

```
(?:\u200d(?:#|)(\w))
```

This searches for the Unicode character `u200d` (known as a "zero-width joiner," which is used to separate special characters without creating a space) or the characters `@` or `#`. After that, it looks for a character that is part of the set `[A-Za-z0-9_]`.

Now, let's see how this works with the comment text:

```
#2hot make loved to her, uupss #Hot #X
```

In the computer, it appears as:

```
#2hot ma<200d>ke lovei<200d>d to <200d>her, <200d>uupss <200d>#Hot
<200d>#X
```

The malware is supposed to look for the characters `u200d`, `@`, or `#`. So, when it searches through the text:

```
#2hot ma<200d>ke lovei<200d>d to <200d>her, <200d>uupss <200d>#Hot
<200d>#X
```

It finds the `u200d` characters and takes the letters that follow them. This results in:

```
2kdhuHX
```

The malware also initially adds the following text on the victim's computer:

```
http://bit.ly/
```

Now, let's see the complete link:

```
http://bit.ly/2kdhuHX
```

Congratulations! Your link has been created! This is how they cleverly obfuscate and hide information.

- Russian APT Groups Continue Their Stealthy Operations[15]
- Satellite Turla: APT Command and Control in the Sky[16]
- Using Google Docs as a C2 proxy with a headless browser[17]

4.10 Denial-of-service attack (DoS) & Distributed Denial-of-service attack (DDoS)

One type of attack that can be launched against a server is a DoS/DDoS attack, which stands for Denial of Service/Distributed Denial of Service. As the name suggests, it aims to deprive users of services.

Imagine you are an employee responsible for assisting various clients. When someone approaches you, you greet them and ask how you can help. Now, if someone wants to disrupt your performance, they could send a thousand people to surround you, and each one would just say hello and shake your hand before leaving. This would create such a crowd that you wouldn't be able to attend to your actual clients.

In a DDoS attack, a similar scenario occurs. When a computer communicates with HTTPS websites, it first sends a request to the server, essentially saying, "Hello, are you there?" The server responds, "Yes, I am here." This initial exchange is known as a handshake, much like a greeting.

Now, imagine if millions of "hello" requests are sent to the server. The server has to respond to each one, saying, "Yes, I am here." This flood of requests

[15] `https://www.infosecinstitute.com/resources/threat-intelligence/russian-apt-groups-continue-stealthy-operations/`

[16] `https://securelist.com/satellite-turla-apt-command-and-control-in-the-sky/72081/`

[17] `https://frereit.de/google_doc2/`

Figure 4.3: src: This work is in the public domain in the United States because it is a work prepared by an officer or employee of the United States Government as part of that person's official duties under the terms of Title 17, Chapter 1, Section 105 of the US Code.

overwhelms the server. It becomes so busy responding to these greetings that it has no time to perform any other tasks. As a result, the website becomes extremely slow and may even become inaccessible. This means that legitimate customers cannot access the service, and you might also struggle to maintain a connection with your server! Have you noticed that when a sale or promotion is active on a website, the site tends to slow down? This is precisely the reason behind it.

DDoSing is a very harmful act and is considered a crime! Engaging in such activities can lead to legal consequences, including being taken to court and facing lawsuits. It's important to understand that this behavior is not only unethical but also illegal. For instance, the FBI has arrested individuals and shut down websites that provide DDoS services and tools under the "Computer Fraud and Abuse Act."

Sometimes, you may receive messages demanding a ransom, threatening to launch a DDoS attack against you if you don't pay, for example, $500 by tomorrow. They might even say, "To prove we're serious, we'll launch a small DDoS attack at 4 PM today so you know what to expect."

It's crucial to remember: never pay the ransom, whether it's in the case of ransomware or these types of threats. Instead, report the incident to the police. They can provide assistance and help you deal with the situation. Paying the ransom only encourages further criminal behavior and does not guarantee that the attackers will follow through on their promises. Always seek help from law enforcement in these situations.

In fact, this is an underground business where infected computers are bought and sold! For example, the person who controls these botnets, which have infected numerous computers, might say, "Pay me $20,000, and I'll launch a DDoS attack on that target for you!"

4.10.1 CAPTCHA

With the advancement of technology, robots have been developed to replicate the tasks that humans perform and carry them out on their behalf. However, this has led to a problem: these robots can execute malicious actions, such as brute force attacks or DDoS attacks. To combat this issue, researchers created a mechanism known as CAPTCHA.

As the name suggests, CAPTCHA is designed to distinguish between humans and computers. Whenever suspicious activity is detected—such as receiving ten requests in one second or repeated actions that resemble robotic behavior—a CAPTCHA test is implemented to determine whether the request is coming from a human or a bot. Tests like "I am a human" or "I am not a robot" are all forms of CAPTCHA.

The idea is to leverage the differences between human cognition and robotic processing. For instance, humans can read and interpret the characters in a CAPTCHA, while bots struggle to do so. Therefore, these tests are used to validate requests. [18]

Now, what happens when your computer is used for attacks like DDoS? Typically, the security systems of websites will notice that your computer is sending requests to many different sites. This indicates suspicious activity, especially if your IP address has been involved in multiple recent attacks (for example, if your IP was part of eight out of ten recent attacks). As a result, they may conclude that your device has become part of a botnet and block your access. This is a clear sign that you may be infected when websites start implementing CAPTCHAs for your requests.

You may have noticed this when using Tor or a VPN. Because these services provide anonymity and many users share the same range of IP addresses, websites may mistakenly think you are a bot, prompting them to present a CAPTCHA. So, whenever you encounter a CAPTCHA, don't panic.

The story behind the creation of reCAPTCHA is quite fascinating. The developers realized that there are countless books and documents that have not been digitized, and hiring people to type them all out would be prohibitively expensive and time-consuming. Instead of employing typists, reCAPTCHA proposed a clever solution: they would separate words into pairs and present them to people on the internet to type. This way, they could collect the typed results and combine them, effectively digitizing millions of images containing text.

How do they ensure that the typed words are correct? They give the same two words to multiple users—typically around ten—and the correct result is determined by the majority's input. By aggregating the results from many users, they can accurately reconstruct the text.

Now, Google is using a similar approach with its Google Street View images. They extract elements like postal codes and license plates[19] and present them

[18] See images: `https://commons.wikimedia.org/w/index.php?search=captcha&title=Special%3AMediaSearch&fulltext=Search`

[19] see image: `https://techcrunch.com/wp-content/uploads/2012/03/`

to users for typing. The typed information is then incorporated into Google Maps. This demonstrates how technology can be leveraged effectively: instead of randomly typing words, millions of books can be transformed into text files, and street maps can be constructed. It's truly impressive how innovative ideas like these can lead to such significant advancements!

How can you bypass CAPTCHA? There are services that solve CAPTCHAs for you. They use AI and machine learning to do so. Not only that, but there are centers where people solve CAPTCHAs for you. Most of them are from countries where the minimum wage is low, like India.

4.11 Backdoor

We have discussed it before. So we will not repeat it here.

4.12 Fileless malware

Fileless malware is a type of malware that doesn't rely on files to infect a system. Instead, it resides in the system's memory, making it more difficult to detect and remove. This is because **some** (not all) antivirus prevention mechanisms are based on scanning files, not memory.

That's why NSA[20] and Kaspersky[21][22] recommend to restart the device weekly. Many devices have built-in feature to do it automatically. Because when you restart the device, the memory is cleared and the malware is removed.

4.13 Coinminer

We know that mining cryptocurrencies requires significant computational power. Some thieves think, "Why should we invest in all this hardware to mine when we can exploit other people's computers to do the mining for us?"

In this scenario, malware is used to hijack your CPU or GPU for mining purposes. Inexperienced and greedy malware developers often consume a large portion of your CPU and GPU resources. This leads to your computer being overworked, causing it to slow down and become less responsive. As the hardware works harder, it generates more heat, which in turn causes the fans to spin faster. You might notice these signs and start to suspect that your computer has been compromised.

However, more sophisticated malware developers take a different approach. Instead of consuming a large amount of processing power, they only use a small

recaptcha-collection.jpg

[20] NSA Mobile Device Best Practices: https://media.defense.gov/2021/Sep/16/2002855921/-1/-1/0/MOBILE_DEVICE_BEST_PRACTICES_FINAL_V3%20-%20COPY.PDF

[21] https://www.kaspersky.com/blog/how-to-protect-from-pegasus-spyware/43453/

[22] https://arstechnica.com/information-technology/2023/06/clickless-ios-exploits-infect-kaspersky-iphones-with-never-before-seen-malware/

fraction—say, 3%—to avoid detection. This way, your computer doesn't overheat, the fans don't run constantly, and you remain unaware of the mining activity. Additionally, they often name the malware something similar to well-known processes, like "Microsoft Service," so that if you open the Task Manager to check which programs are running, you won't easily recognize the malicious software.

But one important thing to keep in mind is that websites can also engage in this practice. When you visit a website, it might use your computer's processing power to mine cryptocurrencies. This can happen in two ways:

- **Transparent Mining:** The website informs you that they are using your computational power for mining, explaining that it helps them earn money and allows you to support their services. They will ask for your permission and approval.
- **Covert Mining:** In this scenario, the website does not inform you and secretly uses your resources for mining. This is considered malicious behavior and is unethical.

Note: If the website is mining, you might notice in the Task Manager that, for example, Firefox is consuming 90% of the CPU. This is certainly a red flag (though it may not necessarily indicate malware; it could simply be a bug in Firefox itself).

To prevent this kind of resource exploitation, you can take several steps:

- **Use Browsers with Built-in Protection:** Popular browsers like Firefox have features that include a cryptocurrency miner blocker. While this is a good start, it may not be sufficient on its own.
- **Script Blocking:** A more effective method is to use script-blocking extensions like *NoScript*. This extension allows you to control which scripts run on the websites you visit, effectively blocking unwanted mining scripts and other potentially harmful content.[23]
- Your device runs out of storage

Some cracked software also contains coin miners. Surely, the person who cracks the software needs money to live. Why should they work for you for free? Some may include malware in the cracked software.

Other signs of infections may be:

- You cannot access antivirus websites. (Malware does this to prevent you from downloading an antivirus.)

[23]However, it's important to note that using script-blocking extensions like NoScript may impact your web browsing experience. Some useful elements on websites, such as interactive features or certain content, might also be blocked along with the unwanted scripts. This can lead to a less functional or more cumbersome experience on some sites.

You may need to manually allow specific scripts or features on trusted websites to ensure they work properly. Balancing security and usability is key, so it's essential to be mindful of which sites you choose to allow scripts on while maintaining your online safety.

- Unintentional changes (deletion of files, UI changes, etc.)
- More internet charges[24]
- Battery drain[25]

4.14 What to Do?

One way that apps generate revenue is by entering into agreements with companies or individuals. They might say, "When someone installs our app, if they also install your app, pay us, for example, 1 cent for each installation." The other party agrees to this arrangement. Therefore, it's important to be cautious when installing apps and to check the boxes you select to avoid installing any additional programs. An app might also violate your privacy, and the more apps you have, the higher the likelihood of encountering bugs. For instance, if you have 200 apps, the chances of a security breach are greater compared to having just 10. It's clear that with 200 apps, each one could potentially have its own vulnerabilities!

Don't open or download files sent via email. Also, make sure to disable automatic downloads. Sometimes, just downloading certain files can cause issues. There might be a vulnerability leading to malicious actions. That's why we always recommend staying updated! The sooner vulnerabilities are patched, the sooner the risk of exploitation is eliminated.

In your phone's settings, disable the installation of apps from unknown sources. Also, look for device admin apps. Normally, there should not be

[24] Some malware consumes a lot of internet bandwidth for downloading things or uploading and stealing your data.

[25] When you become infected with malware, a portion of your hardware is dedicated to the operation of that malware. This is because malware uses your hardware resources to function, much like a car that works harder the more passengers it has. When part of your hardware is occupied by a process, what happens? Naturally, that process requires energy, which means your battery will drain faster than usual. You might suddenly notice that your battery is depleting more quickly after a few days. (Not every instance of battery drain is related to malware! For example, a 5-6 year old phone battery will naturally drain faster than when it was new. In the context of malware, "draining faster" means you notice a sudden decrease in battery life over a short period.)

When hardware is used more intensively, what happens? The device heats up, causing the fan to spin faster to cool it down. Since a portion of your system's capabilities is allocated to the malware, your device's overall performance decreases, making it less able to handle its regular tasks, which can lead to freezing or lagging.

Well, if there were no malware and your phone was still draining battery quickly or overheating, it's worth considering what might be causing this. One possibility is that we know from previous discussions that apps can have issues that are resolved through updates, and no app is perfect. It's possible that the developer didn't optimize the app well for your specific device, leading to poor performance.

If your phone is overheating, it could be due to an app that isn't compatible with your device or an unstable update that doesn't work well with your system. In such cases, you can uninstall the problematic app and wait for a new, improved version to download after it has been updated.

any app there. Additionally, check for accessibility software. Many of them can read everything that appears on the screen. Disable those you don't want.

Imagine a hacker trying to hack into your system. To do this, they would typically need to provide you with a malicious link or a harmful file. Since you are aware that you shouldn't click on every link or open every .exe or .apk file, how would they manage to get through your defenses?

Malware developers have come up with a sneaky tactic: they take advantage of the fact that many users have file extensions hidden. They think, "If someone can't see file extensions, we can create a misleading filename that makes it look harmless, like an image." For instance, they might name a file `screenshot.jpg.exe`. If you have file extensions hidden, you would only see the part that looks like an image (the name): `screenshot.jpg` and assume it's safe to open, which could lead you to fall for their trick. That's why it's essential to enable file extension visibility on your device![26]

One of the tricks hackers use is to provide you with a legitimate-looking app, but when you try to open it, you receive a message saying that a certain program, like Google Play Services, is outdated. It prompts you to click a button to install the latest version. Be aware that these are actually malware, disguised with fake icons to resemble well-known apps, trying to convince you that they are the real deal. Always make sure to update your apps only from official sources (e.g., Google Play Store).

Virustotal is a wonderful website where you can upload suspicious files for analysis. An important note is to not upload sensitive files, as *Virustotal* sends the data to multiple parties.

If you get infected, search for *Safe Mode*, *emergency kits*, and *second opinion malware scanners*. If you suspect that an app is malicious, it's best to uninstall it while in Safe Mode (not in normal mode). Why Safe Mode? Because third-party apps are disabled in this mode. This means that if you try to uninstall a problematic app in normal mode, it might not be completely removed, or you may not be able to uninstall it at all.

4.15 The Dark Side of Connectivity: Security Risks in the Internet of Things

The Internet of Things (IoT) is a network of interconnected devices that communicate with each other. These devices can be anything from smart home devices like smart TVs, smart refrigerators, and smart thermostats to smart cars, smart watches, and even smart toys. The IoT is a double-edged sword. On the one hand, it makes our lives easier and more convenient. On the other hand, it poses a significant security risk. The more devices we connect to the internet, the more vulnerable we become to cyberattacks.

For example, I wake up in the morning, and with a click on my phone, it tells my water heater to start because I want to take a shower. When I'm done,

[26] If you don't know where to enable it, just search for it!

it automatically sends a message to my oven to preheat for breakfast. When I start my car, it notifies my smart door to unlock because I'm heading out. After I leave, the door informs my smart vacuum cleaner to start cleaning the house so that when I return, everything is clean.

In short, in our quest to showcase progress and convenience, we are connecting everything we see to the internet. This is a very dangerous trend. It increases our attack surface. Additionally, security is often a secondary concern, no one is really interested in it. Employers might say, "What? You need a month to enhance security? Forget it! We need to get the product to market before our competitor does. Just skip that month and focus on adding new features, like making the lights change color. We need to cut costs to be able to sell our product at a lower price than our competitors." IoT devices must be cheap to be affordable for everyone. This means that the manufacturers cut costs in every possible way.

Many IoT devices are hard to update, and some are not updated at all. Many use default passwords for all users.[27] For example, if you buy a smart camera, the password might be "admin" for everyone. This is a huge security risk. If a hacker knows the default password, they can easily access your device. This is why it's crucial to change the default password to a strong, unique one. How many of you have changed the default password on your router? I bet 90% of you haven't. :)

If you're a developer, please, please don't implement a system with a fixed default password.[28]

Many IoT devices have a minimal OS. Think about what can go wrong. Many of these OSs have little to no security features.

In summary, not much attention is paid to security. No one cares about the security protocols of these IoT devices. Now, think about what problems this could create.

- Imagine I write malware to hack your washing machine and make it run with the door open, creating a mess in your home. :)
- Or I create ransomware that infects your front door. Then I demand $10 to unlock it. You're in a hurry, so instead of calling a repairman and paying $50 for their service, you say, "Okay, fine!" and pay up. :)
- I could hack your TV and listen in on all the conversations happening in your home through its microphone.

One of the things I've noticed is that in the future, there might be microchips injected into the body that continuously release necessary medications over a month. Now, imagine in the distant future, a company wants to create a luxury drug and connect it to the internet or a communication protocol that monitors

[27] Many VPNs also do this!

[28] UK Becomes First Country to Ban IoT Devices with Default Passwords: https://www.bitdefender.com/en-us/blog/hotforsecurity/uk-becomes-first-country-to-ban-iot-devices-with-default-passwords

the body in real-time. That communication protocol will likely be full of vulnerabilities. It could be hacked, and they might demand a ransom, saying, "If you don't pay, we'll release the entire dosage of the medication all at once!"

If you're working in the IoT field, you need to be very careful about security. Explain to your employer why security is important and convince them of its significance. If you're involved in research, focus on the security of IoT protocols. If you're a user, be careful about the devices you connect to the internet. Don't connect everything you see to the internet. Don't buy a smart device just because it's smart. Think about whether you really need it. Use a firewall to block unwanted traffic.

Reading List

- Spying with your robot vacuum cleaner: eavesdropping via lidar sensors[30]
- Keyhole Imaging[31]

4.16 Sandbox

Sometimes, we need to open files or programs that we're unsure about in terms of their safety or functionality. However, if these files contain malware or harmful activities, opening them on our main system could put our computer at risk. To mitigate this, we can use a sandbox to create a relatively isolated environment. This way, if something malicious occurs, it remains separate from our computer.

Think of it like having a room where you can open unknown items, but instead of doing it in the main room, you use a special box within that room. If something goes wrong, any harmful actions take place inside the box rather than in the main area of your computer.

In a sandbox, we restrict the program's access. We isolate it from other processes and resources. For example, we might allow it to access only a specific part of the computer's memory, permit only certain actions, or limit its access to specific files. By doing this, we minimize its permissions, preventing it from accessing the main computer and interfering with other processes, thus keeping our system safe!

So, what is the purpose of a sandbox, and who might need it?

Frequent Web Surfers: For those who spend a lot of time browsing the internet, a sandbox can be invaluable. The web is a common entry point for malware, as it allows various content to enter your device. If someone frequently visits untrustworthy sites, they can use a sandbox to isolate their browsing activities.

Users Opening Various Files: People who receive numerous emails or download different files can also benefit from a sandbox. Emails can be a gateway for malicious files, so it's wise to open email attachments and use email programs within an isolated environment.

Malware Analysts and Security Professionals: There are individuals whose job is to analyze malware behavior. If they were to work with malware

on their main system, they would risk getting infected. By using a sandbox, they can safely study malware without compromising their primary system. However, it's important to note that for malware analysis, the sandbox must be properly configured; otherwise, there is still a risk of infection.

In summary, a sandbox is a useful tool for anyone who interacts with potentially harmful content, providing an extra layer of security and isolation.

• Is a sandbox completely safe? Does having a sandbox mean we won't get infected at all?

→ Think of it this way: is a seatbelt completely safe? Yes, it prevents many injuries, but does wearing a seatbelt make you invulnerable? No! The same principle applies in the world of computers. There's a saying: "No system is safe." This means that no system is entirely secure. Even the most secure systems can have vulnerabilities that can be exploited. Software also has its flaws, and there are always potential entry points for attacks. Nothing is 100% foolproof!

It's important to configure sandboxes correctly, especially for malware analysis or when running programs of uncertain safety. Otherwise, issues can arise. It's like trying to keep a snake in a box; if the box isn't secure, the snake will escape! If there are gaps at the edges, the snake will find a way out.

There are various sandbox programs available. For general use, the standard settings are usually sufficient. However, for those looking to perform professional tasks like malware analysis, it's essential to customize the settings according to their specific needs.

• Great! We can analyze malware using this method. For example, we can run a program we're unsure about in a sandbox. If it doesn't perform any malicious actions, we can conclude that the program is completely safe and not malware!

→ It's not that simple! While it's true that antivirus programs often use sandboxes to run suspicious applications and see if they exhibit harmful behavior, malware has its own tricks. For instance, some malware tries to detect whether it's running in a sandbox.

Think about it: a computer with no files, no internet connection, and nothing else—doesn't that seem suspicious? Doesn't it look like a sandboxed environment? Malware can check to see if it's running in a sandbox to avoid detection by antivirus software or analysts. Some sophisticated malware, especially those created by professionals or used in state-sponsored attacks (where one government tries to hack another or specific targets), performs initial checks to see if it's in a sandbox. If it is, it may refrain from executing its malicious actions to avoid being discovered.

Take the Stuxnet virus, for example. It spread to thousands of computers but didn't activate in those systems because it was designed to target specific nuclear facilities. It would check if it was in the right environment before carrying out its malicious tasks.

Therefore, for those looking to analyze a company's malware, their sandbox environment should closely resemble the company's actual systems. This means creating files that mimic the company's files but contain false information and

configuring the isolated system settings to match the main systems.

There's also a concept called a honeypot. These are systems set up with misleading information to lure hackers into attempting to breach them, allowing for their capture. It's like placing a jar of honey to attract insects; they come for the food but end up getting trapped. In this case, it's a trap designed to catch hackers!

- Advanced Browser Security with Firejail - A Hands On Guide[29]
- Sandboxie (Windows sandbox app)[30]
- Windows Sandbox (Official solution for Windows)
- Dangerzone[31][32] (Take potentially dangerous PDFs, office documents, or images and convert them FIREJAILto safe PDFs)
- Qubes OS (A reasonably secure operating system)[33]

Reading List

- Safe PDF viewers[34]
- Mozilla Firefox sandbox[35][36]
- EdgeSpot detects PDF samples tracking users who use Google Chrome as local PDF viewer[37]
- Improving Malicious Document Detection in Gmail with Deep Learning[38]
- Chrome 0-day exploit CVE-2019-13720 used in Operation WizardOpium[39]
- Snap and Flatpak sandboxing method
- SELinux & AppArmor

[29] https://firejail.wordpress.com/2021/11/11/advanced-browser-security-with-firejail-a-hands-on-guide/

[30] https://sandboxie-plus.com/

[31] Yes PDF and other files can also be dangerous. (e.g. Malware in Image: https://blog.reversinglabs.com/blog/malware-in-images)

[32] https://dangerzone.rocks/

[33] https://www.qubes-os.org/

[34] https://malwaretips.com/threads/safe-pdf-viewers.109756/, archived: https://web.archive.org/web/20220526230914/https://malwaretips.com/threads/safe-pdf-viewers.109756/

[35] https://hacks.mozilla.org/2021/12/webassembly-and-back-again-fine-grained-sandboxing-in-firefox-95/

[36] https://hacks.mozilla.org/2021/05/introducing-firefox-new-site-isolation-security-architecture/

[37] https://blog.edgespot.io/2019/02/edgespot-detects-pdf-zero-day-samples.html

[38] https://security.googleblog.com/2020/02/improving-malicious-document-detection.html

[39] https://securelist.com/chrome-0-day-exploit-cve-2019-13720-used-in-operation-wizardopium/94866/

4.17 Virtual Machine

A virtual machine is like creating a new computer on your main operating system. You can install a new operating system on this virtual machine.

For example, let's say your CPU has 6 cores, your storage is 512 GB, and you have 16 GB of RAM. You can allocate resources from your main system to create a virtual machine. You might decide to set aside 2 of those 6 CPU cores, 50 GB of your 512 GB storage, and 6 GB of RAM. From that point on, these resources function as if they belong to a separate computer, allowing you to treat it like an independent system. This is done using software known as hypervisors, such as VMware or VirtualBox (the latter being open-source!).

This provides better[40] virtualization compared to regular sandbox apps. I recommend having a virtual machine and doing your web browsing in it.

If the main operating system is Arch-based with a KDE environment, and the virtual machine is Debian-based with a GNOME environment, is this setup more secure than having both systems use the same GNOME environment? I think that if there's a vulnerability in Debian, it might not exist in Arch. So, one could become compromised while the other remains safe. It's like having two different filters instead of two identical pieces of fabric.

4.18 Anti-virus

An important note is that an anti-virus is not a magical solution that will protect you from all threats. It's like a seatbelt in a car; it can prevent many accidents, but it doesn't make you invulnerable. Always look at them as second layer of defense. The first layer is you! You should always be cautious and aware of what you're doing.

4.18.1 Signature-based detection

Have you ever noticed that some people identify someone's nationality based on their face, like Chinese or Japanese? In computers, we do something similar with malware. They check it and notice that a certain element is common among a series of malicious programs. That common element can be considered a shared feature. So, whenever we see it, we can say that it's malware from that family. This means we create a signature based on certain binary patterns, and whenever we see that pattern, we say it's malware. This is called signature-based detection. It is like a fingerprint.

Signatures can be created based on:

Byte sequence

Researchers often extract a unique sequence of bytes from the malware binary that rarely appears in other programs. This could be a specific function call, a

[40] No app is completely bug-free!

distinctive piece of code, or a payload-embedded data structure. For example, the binary:

```
0x68 0x65 0x6C 0x6C 0x6F 0x20 0x65 0x78 0x61 0x6D 0x70 0x6C 0x65 0x2E 0x63 0x6F 0
```

is the hex representation of the string "hello example.com". This may indicate that the malware is trying to connect to a specific domain (C&C server) to download additional payloads or exfiltrate data.

In addition to signature detection, antivirus programs look for which company has signed the file. If the file is signed by a company that is known to be legitimate, it is considered safe. However, this creates a problem. What if the private key of the company is stolen? Then the attacker can sign the malware with the company's key, and the antivirus will consider it safe. There have been cases where the private key of a company was stolen, and the attacker signed the malware with it and attacked big companies.

Furthermore, certain sequences of bytes representing sequence of operations, API calls with certain parameters can be used to create a signature.

What is the advantage of signature-based detection?

- It is easy to implement.
- It is fast.
- Can be used when scanning a file. (Before executing it)

What is the disadvantage of signature-based detection?

- It is not effective against new malware.
- It is not effective against polymorphic malware. (Malware that changes itself to avoid detection)

4.18.2 Behavioral-based detection

Behavioral-based detection is a technique that detects malware based on its behavior. It looks at what the malware does. For example, if a program tries to access the webcam without the user's permission, it is likely to be malware. Malware often employs various techniques to hide its presence and evade detection. Here are some suspicious behaviors specifically related to hiding malware:

- **Code Injection:** Malware often injects its code into legitimate processes to avoid detection. This technique allows the malware to run in the context of a trusted process, making it harder to detect.
- **Process Hollowing:** This technique involves replacing the memory of a legitimate process with malicious code. By doing this, the malware can execute its code in the context of a trusted process.

- **Steganography:** Some malware uses steganography to hide its code within other files, such as images or audio files. This technique allows the malware to evade detection by security tools that only scan for known executable file types.
- **Encrypted Payload:** If a malware sample contains encrypted payloads[41] and decrypts them at runtime, it may indicate malicious behavior. Think about why? Because many antivirus programs scan the file statically, meaning they analyze the file without executing it. If the payload is encrypted, the antivirus program won't be able to detect it.
- **Running Certain Commands:** Some malware may execute specific commands to manipulate system behavior or gather information.
- **Delayed Execution:** Some malware may delay its execution to avoid detection during initial installation. This can involve waiting for a specific event or time before activating its malicious functions.
- **Environmental Checks:** Malware often performs checks to determine if it is running in a sandbox or virtual environment. If it detects that it is being analyzed, it may alter its behavior to avoid detection. Some malware is designed to harm other countries; if it infects its own country, it will be caught by its own country's security systems. For example, malware from Russia will check whether the system it is trying to harm has the Russian language or timezone. If it does, it will not perform any malicious actions.
- **Self-Destruction:** Some malware is designed to delete itself if it detects that it is being analyzed or if certain conditions are met, making it harder to study and understand.
- **Manipulating System Logs:** Malware may attempt to erase or modify system logs to cover its tracks and prevent detection of its activities.
- **Unusual Network Activity:** Unusual outbound connections to unknown IP addresses or domains, especially if they occur at odd hours or in large volumes.

4.18.3 Anti-virus bypass methods

TODO...

Reading List

- Writing YARA Rules[42] (A tool to create signatures)

[41] A payload is the part of the malware that performs the malicious actions.
[42] `https://yara.readthedocs.io/en/stable/writingrules.html`

4.19 End of the Chapter Questions

1. A program access to many files at once and after that, entropy of the files increases. What is the program doing? Might it be malicious?

1.a. Encryption produce random outputs (high entropy). The program is likely doing encryption (May be ransomware)

5.0 Network

5.1 Look-alike Domain

Imagine you're trying to access your bank's website, and you type in the URL. However, you make a typo and end up on a site that looks exactly like your bank's website. You enter your username and password, and the site accepts them. You think you've logged in, but in reality, you've just given your credentials to a malicious actor.

This is a common tactic used by cybercriminals to steal your information. They create websites that look identical to legitimate sites, tricking you into entering your sensitive data. This is known as a look-alike domain attack.

Sometimes they use tricks like offering a discount to get you not to check the URL carefully and just click on the link.

- You can't be phished with a one-time password!

→ That's not true!

- To determine if a page is fake, enter incorrect card information; if you don't get an error, it's legitimate!

→ That's not a valid approach!

Please don't get caught up in these incorrect methods! The best way is to check the domain (though there are conditions, such as using a reliable browser, which I'll explain later). It's all about the website's domain. For example, in Iran, all payments must be made through the Shaparak system (`shaparak.ir`). If you see a website that doesn't use the Shaparak system, it's likely a scam. Always check the domain!

5.1.1 How to determine the domain of a website?

The domain is the last part before the `.com`, `.ir`, or .org, etc. and also the part just before the last dot. Or, to put it another way, it's the last segment before the character "/" and the part just before the final dot.

What does this mean? Let's look at some examples:

`https://www.eff.org/about`

The domain is `eff.org`. Why? Because it ends with `eff.org` followed by the character `/`.

`https://www.eff.org/about/contact`

The domain here is still `eff.org`.

`https://supporters.eff.org/donate/join-eff-4`

The segment just before the last dot. (The last dot is highlighted in red which is the dot after `supporters`.)

When we say the Shaparak domain, we mean something like this:

`https://sep.shaparak.ir/`

This is the Shaparak domain.

The domain of the bank payment gateway page must end with `shaparak.ir`. Here are some examples:

`https://www.bmi.shaparak.ir/`

In each example, I highlighted the domain in green and the dot from which the domain starts in red. It is almost always something like `.shaparak.ir`. Anything else is a phishing attempt:

`https://www.khariid.ir/`

I said it must be Shaparak, not `khariid` or any other domain! So this is phishing!

`https://www.shaparak.xyz/`

I said it should be `shaparak.ir`, not `.xyz`! So this is phishing.

`https://www.bmi-shaparak.ir/`

I said it should be something like `.shaparak.ir`. That's it! Here, a hyphen was used instead of a dot, which makes it phishing.

`https://www.bmi.shapalak.ir/`

In this case, a lowercase `l` was used instead of `r`, which is phishing. You might not pay attention to the address and read it incorrectly.

`https://www.bmi.shaqarak.ir/`

Again, a technique was used to confuse us. A `q` was placed instead of a `p`, which is phishing.

`https://www.bmi.snaparak.ir/`

Here, an `n` was used instead of an `h`, which is phishing.

`https://www.shaparak.lr/`

Here, a lowercase `l` that looks like a capital `I` was used instead of an `i`. That's why they always say not to click on links. It looks like Shaparak, but when you read it, you might think it's a capital `I` when it's actually a lowercase `l`. Never, ever, ever click on links. Instead, type the address letter by letter into your browser.

1. Never download programs from untrusted sources. Hackers often try to lure you into installing software from unreliable sources, such as links sent via messages. For example, they might send a text saying, "A complaint has been registered against you in the government system. Follow the link below for more information," encouraging you to click!

Do not click on such links under any circumstances. Similarly, avoid clicking on links in messages claiming "free internet," "tax reduction tricks," and so on. Malicious software can easily steal the one-time password sent to you via SMS or other formats.

• Some banking apps are not available on the Google Play Store. What should we do?

→ Download them from the official website of the bank. (Pay attention to the bank's domain!) Ask someone knowledgeable about what the bank's official domain is. You can also contact the bank's customer service.

Sometimes, malicious software is sent to you via files through WhatsApp or other applications. Under no circumstances will any government agency send you a message through WhatsApp. For instance, someone might falsely pose as a postal worker and claim, "I have brought a notification letter from the judiciary, and since you weren't home, I had to open it and provide you with the tracking code so you can follow up online using the link below."

They will give you a phishing link. When you enter the site, it will ask you to pay a fee of $2. While $2 may not seem like a lot, you may not realize that the actual amount being charged is $200.

2. Always check that the beginning of the website address starts with HTTPS. If it starts with HTTP, it indicates a phishing attempt. However, just because it starts with HTTPS doesn't necessarily mean it's safe.

You can also enable the HTTPS-Only mode in your browser settings to ensure that you only visit secure websites. You will receive a warning if you try to visit an insecure site.

Some people use their main card, which has a significant amount of money, for online purchases. However, this is completely wrong. Instead, do the following:

- One bank account should hold a significant amount of money but should not have a debit card or online banking password.

- Another account can have a debit card but no online banking password, and it should contain a small amount of money for everyday expenses like buying fruits and groceries.

- A third account should have an online banking password and a small balance for online purchases.

This way, if you fall victim to skimmers (devices that copy your banking information and later use it to withdraw from your account) in the real world, you won't lose much since that account doesn't hold a lot of money! Yes, scams can happen in the real world too!

Similarly, if you get scammed during an online purchase, since your card is separate and only a small amount of money is in that account, you likely won't lose much!

Whenever you need to make a larger purchase, transfer the amount you want to spend to the card you use for online shopping. This way, there will always be just enough money on that card for your purchases, but not a large amount!

4. Do not make payments on public computers, such as those in internet cafes! There is a very high likelihood that their systems are filled with malware (viruses)! I say this seriously! Public computers, including university systems, are often infected. Instead, use your personal phone for payments whenever possible.

5. Use a reliable and updated browser. Good browsers include Firefox and Brave. Chrome is also good, but not for privacy. It's good for security, but not for privacy. Safari is also a good browser. However, I recommend Firefox and Brave. They are both open-source and have good privacy settings.

These browsers are secure and provide good protection against attacks. Additionally, for example, Firefox (not Brave) makes it very easy for you to find the domain. For instance, look at the images below! Firefox highlights the domain for us and dims the rest of the website address, making it significantly easier to identify the domain: The domain in figure 5.1 is `listings-94915329.com` and

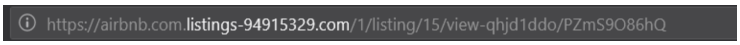


Figure 5.1: Firefox's domain highlighting feature

not `airbnb.com`.

Or look at this image: The domain in figure 5.2 is `eff.org`.

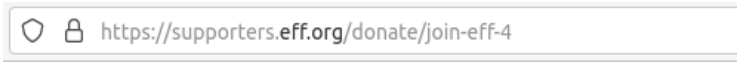


Figure 5.2: Firefox's domain highlighting feature

6. Use Private Window or Incognito Mode. In Private Window mode in Firefox and Brave, some (not all) malicious capabilities related to certain things

(like extensions) are disabled or not allowed to use Private Window at all! So, it might help you **a bit**.[1]

7. Avoid installing unnecessary extensions. Extensions can be dangerous. For example, they can read your browsing history, see and steal your passwords, and more.

You can also keep the browser you use for payments separate from others. For example, you might use Firefox exclusively for online transactions.

8. Using a virtual keyboard can be helpful. As we mentioned earlier, keyloggers primarily target physical keyboards (though not always).

9. Pay attention to the information in the SMS containing the one-time password. When the one-time password arrives, it will indicate the amount and the company involved, for example, "An amount of X for Y company." Make sure that the amount matches the website and that the name of the company you are making the payment to is correct and consistent!

(Note that the one-time password generation apps provided by banks are more secure than SMS-based one-time passwords. Many phishing attempts occur because malware (viruses) can steal the SMS containing the one-time password. This is why we advise against installing apps from sources other than the Google Play Store and other places. Not only that, SMS is an inherently insecure protocol.)

10. Use antivirus software. It can help you detect malware. However, it is not a panacea. It is not a complete solution. It is just a tool to help you. It is like a seatbelt. It can prevent many injuries, but it doesn't make you invulnerable. The same applies to antivirus software. It can help you, but it is not a complete solution.

11. On Samsung phones, go to: `Settings => Accessibility => Installed Services`
`Settings => Biometrics and security => Device admin apps`
Check to ensure that no suspicious apps are enabled in this section. If they are, they can perform various actions, such as reading and stealing your dynamic password SMS.

On Xiaomi phones, go to: `Settings => Privacy Protection => Special permissions => Device admin apps`

12. Most phishing sites have a short lifespan. This means that after a few hours or days, they get added to a blacklist, and both antivirus software and browsers will warn you that the site is phishing. Therefore, one way to check is by looking at the domain age. For example, the payment domain of a certain bank should not have been created just a month ago. You can check the domain age by searching for the site's WHOIS information.[2]

13. Never click on links. Instead, type the address directly into your browser. Have you noticed that sometimes when you enter a website, it says something like:

[1] Private Window or Incognito Mode won't hide your IP address or search history from your ISP (Internet Service Provider) or the websites you visit.

[2] However, some hackers, especially those who are more professional, can buy old domains to avoid this check.

"Click here to see more content." "Linux is actually a kernel."

In the first case, by clicking on the word "here," and in the second case, by clicking on the word "Linux," you are taken to a website where the link is hidden beneath the words "here" and "Linux." If you want to see the link to that website, you can hover your mouse over it and look at the bottom of the page to see the actual address. Additionally, you can right-click on the link and select "Copy link address" to see the actual address. This way, you can see where the link will take you before clicking on it.

In addition to that, phishing can occur through QR codes. For example, you might see a QR code on a poster that says, "Scan this code to get a discount." When you scan the code, it takes you to a phishing site. So, be careful with QR codes as well. Always use tools that allow you to see the actual address before clicking on it.

Moreover, when you type the address yourself, you can avoid phishing with Unicode characters.[3] We have different characters that look the same. For example, the letter `a` can be written in different ways (different languages have similar characters that can be used to trick you. You might think you're clicking on the right website when in fact you're not).

14. See *Phishing campaigns are using AMP URLs to avoid detection*[4]

15. See `https://web.archive.org/web/20220429195039/https://www.youtube.com/watch?v=vPJ6irUDmHI`

This is a very important topic. Therefore, I will pose a few questions. Please think about them before reading the answers below.

I have entered the bank payment gateway. Which addresses do you think, if present on my payment page, would indicate that it is not a phishing attempt?

1) `https://www.sep-shaparak.ir/`
2) `https://www.sep.shapatak.ir/`
3) `https://www.sep.shaparak.ir.khariid.ir/`
4) `https://www.sep.shaprak.jf/`
5) `https://www.sep.shparak.ir/`
6) `https://www.sep.shaparak.ir.shaparak.jr/`
7) `https://www.sep.shaprak.ir/`
8) `https://www.sep.pardakht-shaparak.ir/`
9) `https://www.sep.shaparak-es.ir/`
10) `https://www.sep.shaparak.ir.xyz/`
11) `https://www.sep.shapraak.ir/`
12) `https://www.sep.shaparak3.ir/`
13) `https://www.pytnxjwsafe.ir/`
14) `http://www.shaparak.ir.snaparak.ir/pay/hsju26`
15) `https://www.pardakht-hesab.ir/shaparak.ir`
16) `https://www.shaparak-ir.ir/`
17) `https://www.shaparakir.ir/`

[3] Phishing with Unicode Domains: https://www.xudongz.com/blog/2017/idn-phishing/ -¿ However, the address bar of browsers is now fixed and shows the associated ASCII characters.

[4] `https://www.threatdown.com/blog/phishing-campaigns-are-using-amp-urls-to-avoid-detection/`

18) https:/shaparak.ir.kdsfk.sfjdsalfsaf.whierlw.asovlwervn.gnbv.xsulgvgf.com/
19) https://www.sep.shaparak.ir/

Answer

Only the last one is not phishing; the rest are all phishing attempts. Let's review them one by one (again, to help you become more familiar with domains, I will highlight the domain in green and the dot from which the domain starts in red):

1. We said it should be something like .shaparak.ir, but this is .shaparak-ir. A hyphen?! That's incorrect!

https://www.sep-shaparak.ir/

2. shapatak?? It should be shaparak, but they replaced the letter r with t to trick us!

https://www.sep.shapatak.ir/

3. This is khariid.ir, not shaparak! Notice that the last part before .ir, .com, or .info is the domain!

https://www.sep.shaparak.ir.khariid.ir/

4. Why jf? It should be .ir instead!

https://www.sep.shaprak.jf/

5. shparak.ir? Where did the a after sh go? It should say shaparak.ir!

https://www.sep.shparak.ir/

6. Again, pay attention to the domain. The domain of the site is shaparak.jr! .jr? It should be .ir, meaning shaparak.ir!

https://www.sep.shaparak.ir.shaparak.jr/

7. Why shaprak.ir? Where did the a after p go?! Look! The a is missing after p, so it's incorrect! It should be shaparak.ir, not shaprak.ir!

https://www.sep.shaprak.ir/

8. The domain is pardakht-shaparak.ir. They used a hyphen to trick us into not noticing!

https://www.sep.pardakht-shaparak.ir/

9. The domain is shaparak-es.ir. What is es-?! The domain should be shaparak.ir, that's it!

https://www.sep.shaparak-es.ir/

10. Why .ir.xyz? That's incorrect!

https://www.sep.shaparak.ir.xyz/

11. Why are there two as after the r? Why shapraak.ir? Two as are a mistake!

https://www.sep.shapraak.ir/

12. Shaparak 3? We never had a number! What kind of domain is shaparak3? That's incorrect!

https://www.sep.shaparak3.ir/

13. What kind of domain is this? It's not even shaparak! So it's completely wrong! What does pytnxjwsafe.ir even mean?

https://www.pytnxjwsafe.ir/

14. This has several issues: One: Why is it http and not https? The fact that it's http indicates that it's phishing! That's enough to realize it's a scam! Two: Why is the domain snaparak.ir? Why isn't it shaparak? They used a technique to replace the letter h with n!

http://www.shaparak.ir.snaparak.ir/pay/hsju26

15. The domain is pardakht-hesab.ir, and it's not shaparak at all! - Oh, but what about that shaparak.ir at the end? + That's after the character / and doesn't matter!

https://www.pardakht-hesab.ir/shaparak.ir

16. The domain is shaparak-ir! Did you notice that a dot is different from a hyphen? Here, they tried to trick us with a hyphen!

https://www.shaparak-ir.ir/

17. The domain is shaprakir.ir! It's not shaparak.ir!

https://www.shaparakir.ir/

18. This one is deliberately long to fit in the browser's address bar on mobile devices and not show completely, making you mistakenly think it's really shaparak, when it is not!

https:/shaparak.ir.kdsfk.sfjdsalfsaf.whierlw.asovlwervn.gnbv.xsulgvgf.com/

19. This one is completely correct! It is something like sep.shaparak.ir. The domain is entirely valid and is shaparak.ir.

https://www.sep.shaparak.ir/

See figure 5.3 The designer of this site was really creative. They placed a fake image to make you think you're on the Shaparak site, but in reality, you didn't pay attention to the address bar of your browser and only saw the fake image.

Sometimes instead of text, an image containing text is sent to your email. This method bypassed many spam filters which operate based on text.

Figure 5.3: A phishing site

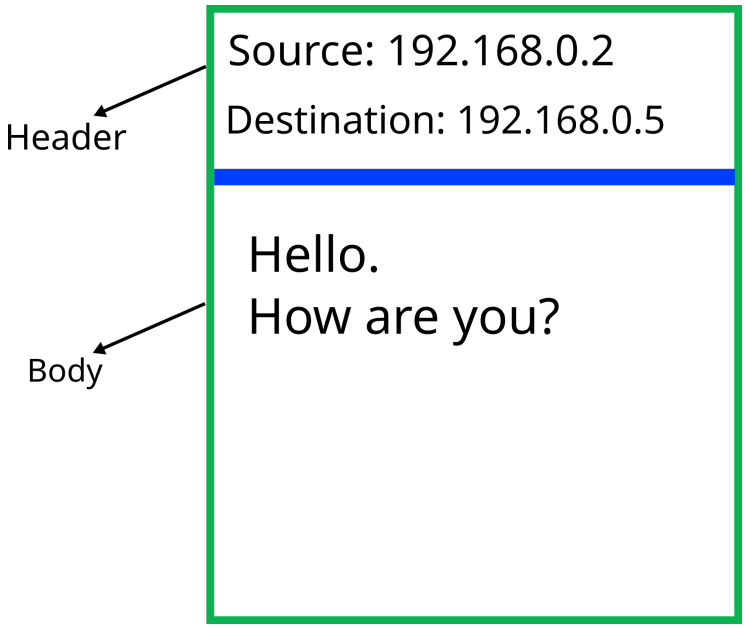


Figure 5.4: A packet

5.2 DNS

The internet is a space where everyone is connected. We also know that if I want to send a message to someone in the physical world, I need to go to the post office, write a letter, and specify the sender's and recipient's addresses. In the world of the internet, it's quite similar. Everyone has an address. If I want to connect to Google, I find Google's address and send a message to that address! This address is called an IP address, or Internet Protocol Address.[5]

Each packet has a header and a body. The header contains the sender's and recipient's addresses and other information about the packet like the protocol used, parameters to establish the connection, the size of the packet, and so on. (Remember this! We will come back to it later.) The body contains the actual data being sent. The header is like the envelope of a letter, while the body is like the content of the letter itself.

But here's the thing! Remembering the addresses of all websites is really hard, right? Also, addresses on the internet look something like this:

```
61.156.22.11
```

It's quite difficult to memorize all those numbers for every site, isn't it? Well, here they tell us that you don't need to memorize numbers! Instead, just

[5]That's why it's said that it's better for no one to know your IP address, because it's like knowing your home address!

remember the English names. You can write something like `signal.org`, and I will find out what the address behind `signal.org` is! In other words, we assign a series of names to it, and later we go find it ourselves.

• How does it find that?

→ It's like a phone book. There are certain places that connect these alphabetical addresses to their numerical addresses. It's something like this:

Alphabetical Address	Numerical Address
`signal.org`	`123.123.123.123`
`google.com`	`12.34.124.123`
`eff.org`	`45.123.11.14`

Table 5.1: A simple example of a DNS table

Note: To obtain the IP address of a server (if it is not behind proxies), you can use the command `ping`. For example:

```
ping signal.org
```

This is why a request called DNS is sent, asking for the IP address of a site. For example, "I want to access the Mozilla site. Please give me its IP." The other side accepts the request and sends it back.

• Who provides us with this IP address for Mozilla?

→ We'll get to that later!

However, there's a problem! This DNS request is sent in plain text, meaning it's transmitted in a completely normal format using English letters, without any encryption. This means that others (anybody in the same network as I am and ISP and routers) can see which website I intended to visit or which server I wanted to connect to. It's important to note that it's not just about accessing a website; sometimes, an application may need to communicate with a server as well.

This situation poses a threat to individuals' privacy because ISPs and other parties can see which sites I wanted to access and even alter that request.[6]

We can use other DNS resolvers that encrypt the DNS communication. (e.g. DNS over HTTPS) For example, we can use `9.9.9.9` or `1.1.1.1`.

When you use HTTPS, the body of the request is encrypted.

But should we use any provider?

No! For example, if we use Google, we're essentially giving them a list of all the sites we visit. Let's take a look at one of these providers.

The information I gathered about Quad9 comes from their website and is licensed under the CC BY-NC SA 4.0 License (Attribution-NonCommercial-

[6] One of the methods used for filtering and banning websites is manipulating the DNS request, resulting in us receiving an incorrect IP address. For example, in Iran, we might receive the IP address of a filtering page, such as `10.10.34.35`, which leads us to a filtering page instead of the intended website. (Peyvandha page: `https://fa.wikipedia.org/wiki/%D8%B5%D9%81%D8%AD%D9%87_%D9%BE%DB%8C%D9%88%D9%86%D8%AF%D9%87%D8%A7`)

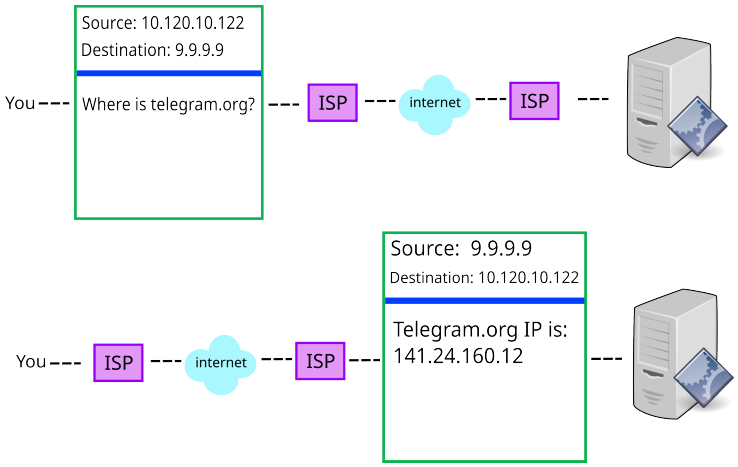


Figure 5.5: A DNS request

ShareAlike 4.0 International). You can find more details on their official website at `https://www.quad9.net`!

Additionally, these DNS resolvers block requests that direct you to websites containing malware or harmful content. This way, they provide a level of protection against malware, helping to keep you safer online:

> Quad9 routes your DNS queries through a secure network of servers around the globe. The system uses threat intelligence from more than a dozen of the industry's leading cybersecurity companies to give a real-time perspective on what websites are safe and what sites are known to include malware or other threats. If the system detects that the site you want to reach is known to be infected, you'll automatically be blocked from entry - keeping your data and computer safe.

You may ask how much of an impact does it have? → A study has shown that:

> DNS firewalls could have mitigated one-third of the incidents we studied[7]

Also we have:

> Quad9 commits to obey the law in any country in which it operates. Therefore, it will only operate in countries with a rule of law compatible with Quad9's ethics and moral duty to protect users. If a government were to use national law to attempt to force Quad9 to act

[7] `https://www.globalcyberalliance.org/reports_publications/measuring-the-economic-value-of-dns-security/`

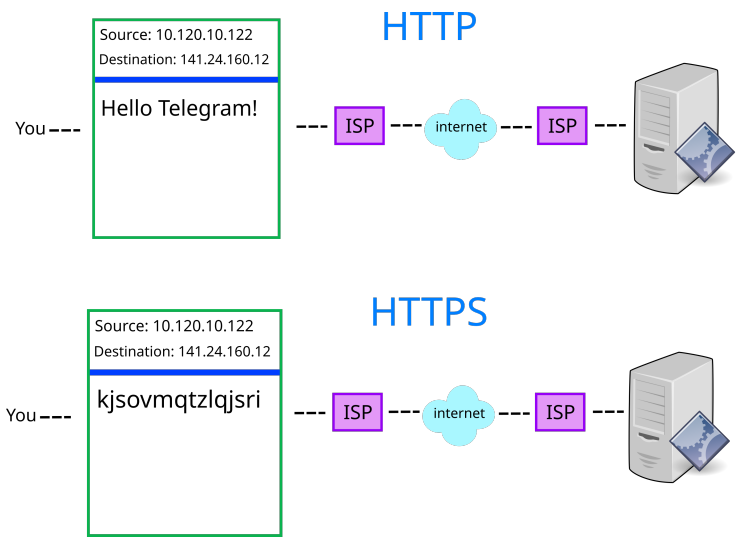


Figure 5.6: Using a encrypted DNS over HTTPS

> in a way that would harm users (such as collecting information that might de-anonymize an at-risk individual), Quad9 would withdraw from operations in that country. This does not mean users within that country would be prevented from using the Quad9 service (unless the country itself prevented them); the service will operate from locations in nearby countries.

One of the good things I noticed in the FAQ section on the Quad9 website is that they addressed a question that many people might have: how do they benefit from this service? This is something that should always be considered in discussions about privacy. They provided an answer to this concern:

> Why do threat intelligence (TI) providers share their data with Quad9, and what do they get out of it?
>
> Quad9 gives anonymized telemetry back to the TI providers only for the malicious domains they share with Quad9. This telemetry never includes the source IP information of the user.

Also:

> When your devices use Quad9 normally, no data containing your IP address is ever logged in any Quad9 system. Connections can employ encryption if your system supports it, and the entire Quad9 platform has been designed to be GDPR-compliant from the first public announcement in 2017.

Don't forget to change your DNS settings on your router as well; Your router, OS, and browser.

• Does this prevent filtering?
→ No! It doesn't prevent filtering. Eventually, the packets will have the IP address of the filtered site in their header. ISP will see that and filter it. Blocking users from accessing a site is done in several ways. One of them is filtering the DNS request. The other is dropping the packets that have the IP address of the filtered site in their header.

5.3 VPN

Many of us may have used VPNs with names like:

- Super VPN
- Super Fast VPN
- Super Fast Unlimited VPN
- Turbo Fast VPN
- Free VPN
- Free Proxy

But we may not realize that almost all of these VPNs are insecure!

5.3.1 Why should I be careful when choosing a VPN?

For example, let's learn about *Betternet*.

It had an IP leak issue, which is one of the worst problems a VPN can have![8]

> Two VPN apps (Flash Free VPN [18] and Betternet [19]), which combined have more than 6M installs, have the highest number of embedded tracking libraries: 11 and 14 respectively.[9]

Is that all? No! What could be worse than this?! Yes, 13 antivirus programs have identified Betternet as malware.

Some VPNs/software are created by governments to spy on their citizens.[10]

[8] https://restoreprivacy.com/vpn/reviews/betternet/
[9] https://research.csiro.au/isp/wp-content/uploads/sites/106/2016/08/paper-1.pdf
[10] See "Tech Tuesday: Eye Spy - The Dangers of Legal Malware": https://blackpointcyber.com/resources/blog/eye-spy-the-dangers-of-legal-malware/

5.3.2 Why should we even use a VPN?

Generally speaking, when you visit a website, the government, your ISP (Internet Service Provider), and the website itself can automatically gather the following information: (Of course, they can learn other things through various methods, but these are the most obvious ones!)

If the website is using HTTP and you are not connected to a VPN, they can see everything! Literally, everything! It's as if they are sitting right next to you, watching everything you do. They can even alter the information you see on your screen!

In the case of an HTTPS site: They can see the address of the website (not the exact page on the website) you want to visit. (We discussed this in detail in the DNS section.)

But the big question is, how can you tell if the connections within an app to its website are HTTP or HTTPS? Normally, you can't!

• So why is it bad for websites to see where we want to go?

→ Because people's browsing habits are unique and specific![32] Let me clarify this for you. Think about your daily routine in the real world. For example, you usually wake up at 7 AM, then go to work at 8 AM. You do certain tasks at work, and finally, you come home around 5 PM, have something to eat, do a few things, and go to bed at a certain time.

Now, let's say you put on a mask and change your clothes. But does that make you anonymous? No! Everyone knows that the person whose life follows that routine is you. So, if someone is familiar with your lifestyle, even if you go to Austria, they can still figure out that the person who wakes up at 7 AM and does certain things is you.

The same goes for browsing the web. Our browsing habits are unique. For instance, if I visit a specific website every morning, even if I change my laptop and internet connection tomorrow, my browsing habits will still reveal that the person visiting those particular sites is me!

There are very few people in the world who browse the web like I do, visiting the same sites and going to specific websites at certain times of the day. You can see this in detail in the research from Mozilla.[32]

See the image below:

In the case of HTTP without using a VPN: Explanation: The government, your ISP, and anyone else on your network (like Wi-Fi) can read your messages! They can see the exact content of your messages.

In the case of HTTP with a VPN: Your message is placed in a secure box that no one can read! This box is sent to the computers of the VPN provider. From there, since things are not filtered, you connect to the website without any restrictions. Essentially, the VPN takes your data and communicates with that website on your behalf. Since there are no filters on that end, we can think of it as a "filter breaker"—something that bypasses censorship! However, from the VPN provider's computers onward, everything is as it was before.

It prevents interference in your privacy (spying) by the government and your

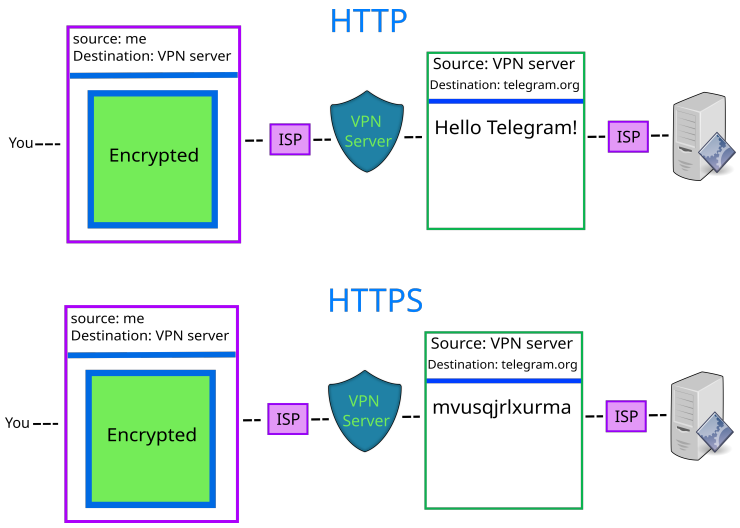


Figure 5.7: How packets look like when using VPN

ISP. When they try to read your message, they'll say, "Oh! We can't read what's in that box!" (Of course, this is true only if you use a suitable VPN, which we will explain later.)

Better Explanation: Your message is encrypted on your device, and then this encrypted message leaves your device and goes to the VPN server. Along the way between you and the VPN provider's servers, because the data is encrypted, it prevents interference from the ISP and some attacks like man-in-the-middle attacks! (Not all types, but some!) Now, at the VPN servers, this encryption is decrypted, and the message continues on its way to the destination you want!

However, the VPN must be trustworthy! Otherwise, just as the ISP and the government could alter our messages or steal our information, a VPN could do the same! While it prevents ISP interference, what if the VPN itself is untrustworthy? That's why we're providing guidance on which VPNs are reliable and good!

Now, what if the website is HTTPS?

In the case of an HTTPS website without using a VPN: **Explanation:** The government and your ISP cannot read the message. This is because HTTPS acts like that secure box.[11] However, they can still know which website we are visiting!

In the case of HTTPS with a VPN: The government won't see which website we want to visit! Once the data reaches the VPN server and is decrypted, the HTTPS encryption still protects our data until it reaches the website! It can see the amount of data you passed out and how much time you're connected to VPN though.

[11] The government can launch various attacks. However, by default, it is secure, and they cannot read the messages.

From us to the VPN server:

```
VPN_Encryption(HTTPS(plain_text))
```

From the VPN server to the desired website:

```
HTTPS(plain_text)
```

Now, if you use an untrustworthy VPN, the information we mentioned can be accessed by the VPN provider, just like the information you would give to your ISP and the government.

While a VPN prevents the ISP and the government from seeing what you are doing and is often used to connect to HTTP sites to prevent ISP spying, what happens if the VPN itself is untrustworthy and decides to act maliciously? That's why it's crucial to choose a reliable VPN!

5.3.3 What Data Can They Collect

The information you typically provide when installing the app includes various device details. This encompasses hardware information, unique device identifiers, and other related data.

- What to do?

→ You can use apps that collect less information, and experts believe these options are better.

→ Some providers offer a file called a config file. You can use this file to establish your VPN connection through other applications. This way, you can utilize clients that either do not collect data or collect significantly less.

5.3.4 When choosing a VPN (Or generally an app), what should you pay attention to?

An important note is that you should not rely only on one factor. For example, if a VPN is open-source, it doesn't necessarily mean it's good. You need to consider multiple factors together. Here are some of the most important ones:

Don't fool for reviews!

Many people are unaware that reviews can be manipulated. There are services that allow you to buy fake reviews. For example, you can pay a few dollars to get 1000 positive reviews written by real people. Those people may be infected with malware. In addition to that, there are centers in countries with low wages where people are hired to write fake reviews. They are paid a few cents for each review they write. So, if you see a VPN with 1000 positive reviews, it doesn't necessarily mean it's good. It could be that the company paid for those reviews. Look for other indicators of trustworthiness.

What version to use?

It's best to use the official version of the VPN. For example, if you want to use ProtonVPN, download it from the official website. Don't download it from other sources. If you download it from a third-party source, you might be downloading a modified version that collects data.

Additionally, what version should you use? Alpha, beta or stable version? Alpha is the most unstable version, followed by beta, and then stable. Programmers use alpha and beta versions to test their software. If you use an alpha version, you might encounter bugs or security problems (as program testing is still ongoing).

The creators should be trustworthy

Do they have a good track record when it comes to privacy? For instance, if a company with a history of selling data has developed this VPN, you can't trust them.

What if the company has no history at all? Sometimes, you might notice an unknown VPN suddenly gaining popularity overnight. This can be suspicious.

Perhaps the goal is to gather information from users. For example, if a government is blocking VPNs, but one VPN is still operational and everyone is talking about it, promoting it, or if someone messages you saying: "Hi, I've been following you, and you're a great person. I found a VPN that connects easily and rarely disconnects. You should install it." You should have a red flag in your mind that this VPN might have been created with the intention of collecting data.

• But wait, a well-known TV network is advertising it! A network that operates against a certain government. Or someone with 200,000 followers is promoting it. They're an influencer! It's impossible that they're in relation with those who want to collect information.

→ Most TV networks and influencers will promote anything if it's somewhat popular or if you pay them well. Or they bring in an expert who might not be credible and promotes bad things. The network itself may lack the knowledge to determine whether the information being presented is accurate.

Every government, especially dictatorial regimes, has certain individuals who may appear to oppose them on the surface. However, their main goal is to engage in media manipulation and gather information about people.

These individuals enter the scene during critical moments to divert attention from specific things. In fact, some of those with a large following accept offers to collaborate with security organizations for financial gain. Or they may cooperate in order to be able to continue their work.

Transparency and Open-source Nature are Essential.

The service must be clear about what it does, what it claims, and how it operates. This transparency in applications is tied to being open-source. However,

simply being open-source is not enough; it must be verified. Open-source means the code is accessible, but it doesn't necessarily mean it's good!

It should be validated by experts, meaning that independent organizations or researchers have conducted security assessments. Additionally, bug bounty programs[12] can be a good indicator of a service's reliability.

A good VPN provides clear information about the protocols and algorithms it uses[13] and does not hide this information. In contrast, unreliable VPNs typically do not explain the algorithms they employ or how they are implemented. They may simply claim, "Trust us for your privacy; we are great." That's fine, but I need to see how you implement your service! Where is your code?

A reputable provider offers a transparent picture of what they have done and what they plan to do. For example, you can see Proton VPN's 2023 roadmap.[14]

They explain what they have accomplished. If you visit their website, you can see what they plan to do in the future. This way, they present a transparent picture of their operations, which is a sign of professionalism. (Not every company that does this can be labeled as professional, but it is certainly a good practice that others should follow. It allows them to create an accurate representation of their past activities and a more organized vision for the future by outlining their goals. Additionally, customers feel more satisfied when they see the improvements that have been made and the new features that are set to be added. As a customer, I can understand that this company is making an effort and has accomplished certain things.)

Country of Origin

It's important to consider that while the individuals or team behind a VPN may be trustworthy, they might be legally obligated to comply with certain regulations. For instance, in some countries, the government may require the VPN provider to retain user data or share it under specific circumstances.

• Can the VPN provider do anything about this?

→ Sometimes, they can! For example, they can choose not to store any data. If there's no data retained, then even if the government requests it or a hack occurs, there's nothing to leak!

This is often referred to as a "No-log policy."[15][16] (However, it's worth noting that some providers falsely claim to have a No-log policy while actually storing

[12] We even have a profession called "bug bounty hunter." For example, if I have knowledge about application security and I discover a bug, I report it according to the company's guidelines. They might say, "Great job! You didn't exploit the bug and instead reported it to us so we can enhance our security. Here's a $100 reward." For instance: ProtonMail Bug Bounty Program. However, be aware that you can't just go out and find bugs randomly! You need to have the necessary knowledge and not engage in unauthorized activities. Follow the established rules; otherwise, if you exploit a bug, you could be committing illegal hacking and end up in jail!

[13] e.g. `https://protonvpn.com/secure-vpn/strong-protocols`

[14] `https://protonvpn.com/blog/proton-vpn-roadmap-spring-2023/`

[15] Mullvad VPN was subject to a search warrant. (Customer data not compromised)

[16] OVPN wins court order

data.[17] This is why it's crucial to choose a provider that is genuinely trustworthy!)

• So, if they don't store data, the government can't do anything, right?

→ Not necessarily!

Group 1: In some countries, you can't easily set up services that protect privacy to the extent that they can prevent government access. If you can, you often have to cooperate, and this cooperation sometimes doesn't even have a specific legal framework! This means they can seize data based solely on accusations, without a court trial.

These countries also share their data with their allies. For example, if countries X and Y are friends, and you use a VPN from country X, your own country, Y, can request information about a specific individual.

Group 2: In some countries, like the United States, you are allowed to set up these services, but sometimes you have to log who is using them. If they want information about specific individuals, they can obtain the logged data with a court order. If you refuse to cooperate, you might have to shut down your business. You may face some difficulties, but compared to countries in Group 1, it's much easier to separate yourself and say, "I won't cooperate," and you can bring media attention to the issue so that the government can't pressure you too much. However, in Group 1 countries, this usually doesn't happen. The pressure is often so intense that you're forced to comply, and you don't have the freedom to publicize the situation.

For example, a notable case is Lavabit, which faced pressure to hand over information about Edward Snowden. They stated they were willing to shut down their service rather than provide the data.[18]

Group 3: In some countries like Switzerland and Sweden (and European Union), which have strong privacy protections, the government doesn't force you to log data. Generally, the government can't exert pressure. They can only request logs in specific cases, and this requires approval from a court and a judge within the country. They can't simply request logs based on suspicion; they must (at least mostly) prove that you are a criminal. Only then can they obtain logs to track you down, as was the case with ProtonMail.[19] Typically, these countries have better privacy protections and respond to requests from other countries with caution:

> In general, Swiss authorities do not assist foreign authorities from countries with a history of human rights abuses.[20]

That's why the origin country is important.

• Why do I hear more about privacy violations in Europe than in certain other countries?

[17] https://cyberinsider.com/vpn-logs-lies/

[18] https://web.archive.org/web/20130809031439/https://lavabit.com/

[19] https://restoreprivacy.com/protonmail-logs-users/

[20] https://web.archive.org/web/20230121203459/https://proton.me/legal/transparency

→ Because in countries with freedom of speech, these issues are taken very seriously. People have the right to defend their rights and make them public, and there are even researchers who actively look for instances of privacy violations to address them. In contrast, in other countries, people don't have the right to speak out about these issues, and there isn't enough freedom for investigative journalists.

NOTE

When news about theft, corruption, and accountability for wrongdoers diminishes or disappears, it indicates a high level of corruption in that place, as there is no sensitivity to holding wrongdoers accountable. Corruption never truly disappears; it exists everywhere. However, when there are no reports of high-ranking officials being dismissed, it suggests that everyone is colluding and engaging in corrupt practices together.

Privacy Policy

Another thing we should consider while looking for a desired program is their privacy policy. A privacy policy is a document that explains how the company will handle your data, including how it uses, stores, and shares your data. It's important to read this document carefully. Generally speaking, if a company has a very complicated privacy policy, it's a red flag. Some companies make their privacy policy so complicated that you can't understand it. This is a sign that they are trying to hide something. A good privacy policy is clear and easy to understand.

• Do they always tell the truth in their privacy policy?

→ Not always! But if they don't, you can sue them.

Protocol in Use

In a VPN, there are various connection methods (protocols) available. However, not all of these methods are secure or reliable. Good protocols are:

- OpenVPN
- Wireguard

example of a bad protocol is PPTP.

• I see some protocols doesn't work in countries like Iran and China. Why is that?

→ Remember when we talked about the structure of packets? (Figure 5.4) The header of the packet contains information about the protocol used. Bits used in the header of OpenVPN packets are different from those used in Wireguard packets. So, if the government wants to filter a specific protocol, they can do so

by filtering the packets that have that specific header. They can fingerprint the packets and say, "If I see packets with this type of header, I will drop them." This is another method of filtering and blocking access.

Encryption

For apps, you should pay attention to how they protect your data. What algorithms for symmetric encryption, asymmetric encryption, hash functions, and key exchange mechanisms do they use?

Plaintext data should be encrypted before being sent over the network. The encryption key must not leave the device. The key should be generated on the device and not transmitted to the server.

Moreover, data should be stored encrypted on the device. For example, some apps (e.g., messaging apps, note-taking apps) store messages in plain text on the device. This is a security risk because if someone gains access to your device, they can read your messages through the app's database (which is stored in plaintext in different files on the device). Unfortunately, many apps we use, even those that claim to be secure, store data in plaintext on the device. The security of the data starts from the device.

• If the data is encrypted on the device, how can we enable features like search? Many features require access to the data.

→ There are solutions out there. See how Tuta does that.[21]

Desktop app or Browser Extension or web apps?

Desktop apps can be more secure than browser extensions in some cases. As they have more access to the system, they can provide more security features (e.g., through the enforcement of security features).

Browser extensions are limited in terms of what they can do. They can't access the system as much as desktop apps can. They can only access the browser and the data that the browser provides.

Another advantage is that open-source desktop apps are mostly tested and analyzed by experts. You can download them and use them. But how can you be sure that the JavaScript code you get is the same JavaScript code others get? The service may collaborate with security agencies to provide you with a specific code that uses weak encryption.

On the other hand, desktop apps can collect much more data.

What login method should I use?

It's best to use a method that doesn't require you to provide your email address. If you use your email address, they can link your account to other services. For example, if you use the same email address for your VPN account and your

[21] `https://tuta.com/blog/first-search-encrypted-data`

email account, they can link the two accounts together. If you use a different email address for each service, it's harder for them to link the accounts.

If you use a username and password, it becomes more difficult.[22]

Although biometrics may seem convenient, it is sometimes not recommended. Some biometric options are not very secure. Most face recognition systems can be fooled with a photo.

You can be forced to unlock your device with your fingerprint or face. In some countries, police can force you to unlock your device with your fingerprint or face, but they are not allowed to force you to unlock your device with a password.

Additionally, if biometric data is stolen, it can't be changed. You can't change your fingerprint or face. That's why I recommend not enabling biometrics for cheap IoT devices, as they are more likely to have less security.

A disadvantage of using a password is that it might get leaked through security cameras. It is really hard to always look at your environment and make sure no one is watching you. Eventually, there will be a situation where you inadvertently enter your password in front of a camera.

[22]But not impossible, as many people use similar usernames for different services. There are OSINT tools that can help you find usernames on different platforms.

6.0 Fingerprint

While many resources focus on trackers, cookies, and other methods of tracking, we little see discussions about fingerprinting. For this, I preferred to dedicate a separate section to this topic.

In this section, we will learn that privacy-based tools are not necessarily good! They might set us apart from others, which can be suspicious. For example, if only 10 people are using a specific privacy-focused system, it clearly makes us identifiable! They may not know what we are doing, but it's obvious that we might be engaging in something considered "bad." Therefore, not every anonymity-based operating system is beneficial! Instead, it should not uniquely display our fingerprint on the web!

6.1 What is Fingerprinting?

In the real world, you stand out from others based on certain characteristics. In other words, these traits help identify you. For example, if a friend were to describe you, they might say: "Yeah, he's of average height. His skin tone is light brown. He has black eyebrows and an average nose. His eyes are brown and large. His hair is black. He usually wears green clothes and typically has white or black pants. He wears glasses."

So, people recognize you by these features. You know how, when they want to identify a criminal, they use facial recognition? Well, when you go online, your device sends and shares certain information that can describe you. There used to be a saying: "On the internet, nobody knows you're a dog." This means that no one knows if a dog is sitting behind the computer or a person. However, we need to revise this thinking! You are not anonymous on the internet! When you enter a website, your device automatically tells that site (or makes some traces based on your behavior), "I'm a laptop with a Windows operating system, from Tehran. I'm browsing with Firefox version X, I have these fonts installed, my timezone is X, these extensions are installed (with the help of investigating the behavior of the browser based on some elements), my window size is 1920x1080[1],

[1] See also: *How CSS Alone Can Help Track You*: `http://web.archive.org/web/20230424015823/https://matt.traudt.xyz/posts/2016-09-04-how-css-alone-can-help-track-you/`

and my hardware information is this." So, if you visit that site again tomorrow, it will recognize you because of this information. In fact, finding you online is quite easy!

With this information, they can track you on the internet, and even if you change your IP address a hundred times, they can still figure out who you are. This raises serious questions about anonymity on the internet.

• Why is this information sent about me? Why does it say I'm using a certain browser and that my operating system is Windows?
→ I'll provide the answer in the User-Agent section.

• Why wasn't the internet designed from the beginning in a way that doesn't require this information to be sent?
→ Unfortunately, the internet wasn't built with privacy and security as its top priorities! In fact, things are usually created first, and then security measures are added later. Initially, workers were just doing their jobs, and then they realized, "Oh! We need something called safety gloves and helmets." Cars were introduced first, and then seat belts came along, followed by airbags. There were no airbags from the very beginning! Essentially, we first develop a new technology or feature, and later we realize, "Oh! This part needs to be secured!" So, we think about securing it afterward.

• Is there a way to reduce the amount of data sent, or at least send false or misleading data so that they don't know what my device really is? For example, can I pretend I'm using Linux and Brave instead of Windows? Can't we trick websites by sending incorrect data?
→ Yes, there are some solutions for that. We can discuss them in detail later.

6.2 Narrowing Down: How Characteristics Lead to Identification

How can this data and fingerprinting lead to my identification? Let's create a scenario in the real world to illustrate how you can be identified:

- Which country are you from? Let's assume you're from Iran. This narrows it down from 8 billion people to 85 million.
- Now, what is your height? This might limit the circle of identification to, say, 20 million people.
- What is your skin color? This could further narrow it down to around 5 million.
- What is the color of your eyes? This might reduce the number to about 4 million.

- Are you wearing glasses? If you are, that could narrow it down to approximately 500,000 people.
- What color do you usually wear? This might limit it to around 50,000 individuals.

This process continues, with each characteristic further narrowing the number of potential matches until you are identified! There are thousands of other factors that can also play a role, and some of these factors can distinguish you very quickly.

On a computer, you have specific characteristics that serve as your "fingerprint" or, more accurately, your features (device characteristics and behavior patterns). When these elements are combined, they tighten the identification loop around you, making it easier to recognize you.

6.3 Does This Fingerprinting Have Any Positive Uses?

6.3.1 Detecting Suspicious Behavior

Imagine you always log into your bank account using your laptop. Now, if one day someone cracks your account (the term "hacking" is often used, but "cracking" is more accurate) and tries to access it, the bank's identification mechanisms, designed with automated programs, will check and realize, "Wait! Someone is trying to log in from a new device that you've never used before." This prompts the bank to suspect that someone might be trying to access your account from a different location. As a result, they may block your account and notify you of the incident, asking if it was you who attempted to log in.

For instance, when you log into your Google account from a new device, Google wants to ensure that it's really you. It uses the fingerprint of the new device to determine that this is a new device that hasn't been used before. Therefore, it might not be you at all! This is why they want to verify that the right person is accessing your account.

Additionally, if a website requires each person to have only one account, fingerprinting techniques, including cross-browser and cross-device fingerprinting, can help prevent the creation of multiple accounts by the same individual.

Furthermore, fingerprinting can be used to detect bots. For example, if a website receives a large number of requests from a single device in a short period, it might suspect that a bot is trying to access the site. In this case, the website can block the bot from accessing the site. This also helps to prevent DDoS attacks, as the fingerprint of requests from a bot is different from a real person's browser.

Moreover, fingerprinting can be used to detect whether a real person has clicked on ads or whether a bot has clicked on them. This is important for advertisers, as they want to ensure that their ads are being seen by real people and not bots.

Nevertheless, fingerprinting can also be used for malicious purposes. For example, if a website wants to track you, it can use fingerprinting to identify you and monitor your activities. This is why it's important to be aware of fingerprinting and take steps to protect your privacy.
• Is it accurate enough to identify me?
→ Yes[33].

6.4 Device Fingerprinting

What device are you using? What is the name of the device? Are you on a desktop or a mobile phone? What operating system are you using? Windows, GNU/Linux, BSD-based, Mac, Android, iOS, or something else? All of these can help narrow down your identification!

What is the name of the device you are using to connect to the internet? This name can be seen by your ISP. Unfortunately, the default name of each device is often the exact model of your smartphone! For example, for a Samsung A73, it appears as: SM-A736. This means the ISP can see that SM-A736 is connected.[2]
• Should we change it?
→ Changing it would create a unique fingerprint. Because most people use default names, changing it would make you stand out. While SM-A736 might blend in with others, the downside is that if someone knows your device model (or the apps you're using), they can also understand its vulnerabilities! Knowing the exact model of your phone can be the first step toward potential risks. The truth is, I can't give a definitive answer. What do you think?

What is your time zone? Because it can likely help determine your geographical area.
• What is a time zone?
→ The Earth is divided into different sections to synchronize time across regions. This allows me to understand what time it is in places like Switzerland, for example. We have a reference point known as Greenwich Mean Time (GMT). For instance, Iran is 3 hours and 30 minutes ahead of Greenwich (UTC+3:30).

It's not always necessary for someone to know your time zone directly! They can infer it from your activity patterns. For example, I can determine where you are based on the times you post tweets. Generally, people sleep at night and are awake during the day. By monitoring your activity, I can see when you are online and active, which helps me estimate your time zone.

[2]
• Where can I find out what this name is?
→ Let's follow a logical approach. The device name should typically be in the settings! So, let's go to the settings. You'll see a series of options. The device name is usually found under "About Device." There, you can see the name of your device.

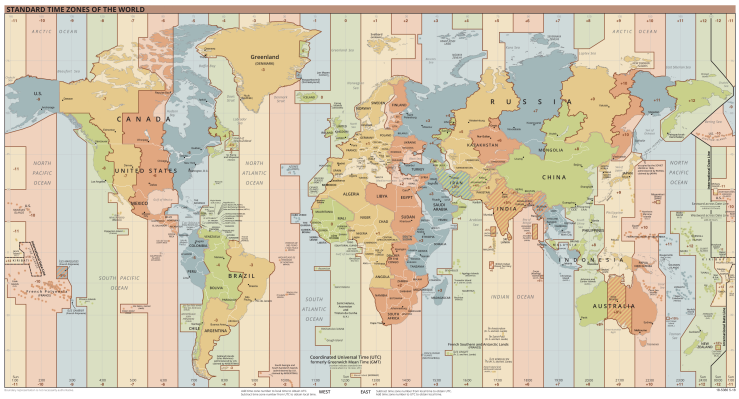


Figure 6.1: Time Zones src: This work is in the public domain in the United States because it is a work prepared by an officer or employee of the United States Government as part of that person's official duties under the terms of Title 17, Chapter 1, Section 105 of the US Code.

For instance, if I notice that you're not online on Fridays and Thursdays, it might indicate that you're Iranian, as those days are typically weekend days in Iran, and you might not be working then. Conversely, if you are more active on Thursdays and Fridays, it could suggest that those are your days off, giving you more time for online activities.

Additionally, there might be certain times of the year when you are less active, perhaps due to celebrations like the New Year. All of these patterns can provide clues about your geographical location and time zone without you explicitly stating it.

For this, Tails OS (an operating system that focuses on privacy and anonymity) sets the time zone to GMT.[3] What is installed on your operating system? This includes both applications[4] and various other elements like languages, fonts, settings you've configured, and even your wallpaper! Yes, even your wallpaper can serve as a fingerprint![5]

Warning

If you believe you are a target and need to remain anonymous, keep in mind that every additional item or customization can serve as a fingerprint! Almost anything can contribute to your unique identification, even things you might not consider significant.

This method can lead to your identification across different browsers because the information about your device remains consistent.

[3] `https://tails.boum.org/doc/first_steps/desktop/time/index.en.html`
[4] Some Android apps gather the list of installed apps and use it to track you.
[5] https://fingerprint.com/blog/how-android-wallpaper-images-threaten-privacy/

6.5 Browser Fingerprinting

6.5.1 User-Agent (HTTP Header)

The Mozilla website[6] provides a very comprehensive yet simple explanation of what a User-Agent (UA) is. In summary, a browser sends a string of text to a website to describe itself, ensuring that it receives the most compatible version of the site. This allows the website to display correctly, ensuring that buttons work properly and the layout looks good. For example, the browser might say, "I am Firefox version X on Windows."

While this is beneficial for user experience, it poses privacy issues. This information can serve as a distinguishing feature that sets you apart from others. Each piece of data contributes to creating a unique fingerprint for you, making it easier for websites to identify and track you across sessions.

• Why should we differentiate and say that if you come in with Firefox, the site looks one way, and if you come in with Brave, it looks a bit different? Why not just provide the same version for everyone?

→ You're right! It would be better if that were the case. However, browsers have some differences, and that creates issues. Because of these variations, sometimes it's necessary to make certain adjustments to ensure compatibility.

There have been various efforts to standardize different aspects of web development, but when things vary between browsers, achieving that standardization becomes challenging! Each browser may interpret and render web content differently, which is why developers often need to make specific changes to accommodate these differences.

6.5.2 Accept Header

The Accept header is another part of the HTTP request that the browser sends to the server. These headers specify the types of content the browser can accept. For example, the browser might say, "I can accept mp3, mp4, and webm files." This information can also contribute to your fingerprint. See also: Accept-Encoding[7], Accept-Language[8].

6.5.3 How You Type

How you type can also serve as a fingerprint. For example, the speed at which you type and the time intervals between keystrokes can be used to identify you. I highly recommend seeing Whonix's article on this topic.[9]

[6] https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/User-Agent
[7] https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Accept-Encoding
[8] https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Accept-Language
[9] https://www.whonix.org/wiki/Keystroke_Deanonymization

6.5.4 Extensions and add-ons

The extensions and add-ons you have installed can also serve as a fingerprint. For example, if you have an ad-blocker with specific settings, this can help identify you. Because your browser behaves differently from others, it can be used to distinguish you from other users.

For example, there are some extensions that block trackers based on your behavior. They watch what websites you visit. If they want to gather information or make a connection, they allow it. However, if a website wants to make a connection and you have not visited that website (or that category of websites), it blocks the connection.

Nevertheless, this behavior can be used to identify you. For example, if you have an extension that blocks connections to a specific website, this can be used as a fingerprint and can leak your browsing behavior. See [34] and EFF Privacy Badger changes[10].[11]

6.6 Solutions

First, I should note that there is no perfect solution! However, we can take some steps to reduce the effectiveness of fingerprinting. The Internet is not designed to be private, and it's not easy to be completely anonymous. Still, we can take steps to lessen the effectiveness of fingerprinting (some may cause compatibility issues). Here are some solutions:

6.6.1 Blending In

Think about a world where every person is just like any other person. They all have the same height, weight, hair color, eye color, and even the same face! In such a world, it would be impossible to identify anyone! This is the concept of blending in. If everyone is the same, no one stands out, and no one can be identified.

To achieve this, we need to make our fingerprint similar to others. For instance, the Tor Browser[12] is one of the best tools for blending in. It's designed to make all users look the same.[13] People using the Tor Browser seem to be using the same browser, with the same settings, extensions, and even the same operating system! This makes it difficult to distinguish one user from another (not impossible, but difficult!).

[10] https://www.eff.org/deeplinks/2020/10/privacy-badger-changing-protect-you-better

[11] See also:
DataSpii: The catastrophic data leak via browser extensions: https://securitywithsam.com/2019/07/dataspii-leak-via-browser-extensions/
Avast and AVG collect and sell your browsing history: What you need to know: https://www.tomsguide.com/news/avast-avg-data-collection

[12] https://www.torproject.org/

[13] https://blog.torproject.org/browser-fingerprinting-introduction-and-challenges-ahead/

The Tor Browser is slow. If you want a regular browser with similar features, you can use the Mullvad Browser[14]. It is designed in partnership with the Tor Project and Mullvad VPN. Generally, it is the Tor Browser without the Tor network. It is designed for better privacy (not anonymity).

6.6.2 Randomizing

Randomizing is another method to reduce the effectiveness of fingerprinting. For example, you can use a tool that changes your fingerprint every time you visit a website. This way, you won't always appear as the same user. You can also randomize your time zone, language, and other elements that contribute to your fingerprint. See how the Brave Browser implements this. [15]

Warning

Some people might use test websites to check if their fingerprint is unique. However, if that website is malicious, it can store your fingerprint and track you across the web! Be careful when using such websites. The EFF[a] tool is a better option if you want to check your fingerprint.

[a] `https://coveryourtracks.eff.org/`

Cover Your Tracks from EFF gives you an output like thsis:

```
only one in X browsers have the same fingerprint as yours
```

Generally speaking, the smaller the number, the better your fingerprint is. This is because it means we need to check fewer browsers to find one that is similar to yours. Conversely, if the number is high, it means that your fingerprint is so unique that we need to check many browsers to find a similar one. You're not blending in.

6.6.3 Browser Isolation

If your browsing data is separated, it becomes harder to link different browsing data to one another. For instance, use one browser for your school stuff, another for work stuff, and so on. However, this doesn't prevent device fingerprinting (device-specific information). Additionally, don't use the email or phone number you used in one browser in another browser, as this can link your browsing data to one another. For your privacy, use fake email addresses and phone numbers. Phone numbers are dangerous; they are directly linked to you.

As a result, some people have different devices/OSs for different tasks.

[14] `https://mullvad.net/en/browser`

[15] `https://brave.com/privacy-updates/4-fingerprinting-defenses-2.0/`
`https://github.com/brave/brave-browser/wiki/Fingerprinting-Protections`

6.6.4 Reading List

- A history of Fingerprinting protection in Firefox[16]
- Demo: Disabling JavaScript Won't Save You from Fingerprinting[17]
- Mitigating Browser Fingerprinting in Web Specifications[18]

[16] https://www.ghacks.net/2018/03/01/a-history-of-fingerprinting-protection-in-firefox/
[17] https://fingerprint.com/blog/disabling-javascript-wont-stop-fingerprinting/
[18] https://w3c.github.io/fingerprinting-guidance/

7.0 Web Browsing

We've reached the web browsing section. Just like in the previous parts, where I first introduced malware and then discussed countermeasures, in this section, I want to start by outlining the threats before explaining how to deal with them.
• Why do you do this?
→ Because you need to understand what poses a threat to you. If I simply list a few things to memorize, like what to do and what not to do, you'll quickly forget them because you won't know the reasons behind them. For example, why shouldn't I do a certain action? Why should I avoid accessing an HTTP site? If you don't understand the reasoning, it will slip your mind. Moreover, new and more complex threats are emerging every day, and it's not always someone else's job to teach you. You need to be your own teacher so that when new threats arise, you can learn how they operate and how to counter them.

In the future, I won't be here to guide you, so I'm teaching you how to fish now, so you can catch bigger fish on your own later.

7.1 Web Browser

A browser is software that allows you to access the internet. It's like a car that takes you to different places on the internet. If your car had a camera to watch you while you drive, you wouldn't like it, right? Similarly, browsers can watch you while you browse the internet. They can track you, record your activities, and even send this information to others. This is why it's important to choose a browser that respects your privacy.

I recommend:

- *Firefox* and *Brave* for regular use
- *Mullvad* for better privacy (better privacy comes with some compatibility issues for some websites)
- *Tor Browser* for anonymity[1]

[1] If your anonymity is vital for you, the Tor Browser alone is not enough. Use anonymity-focused operating systems like Tails OS.

7.1.1 Chromium-based or Non Chromium based?

Chromium is an open-source project that serves as the foundation for many browsers, including Google Chrome, Microsoft Edge, and Brave. These browsers are created based on changes to the code of Chromium. They modify some functionalities and add new features to create a unique browser. However, the core of these browsers is still Chromium. While Chromium is open-source, Google Chrome and Microsoft Edge are not.

I don't have enough knowledge about the security of browsers. Some say the sandbox of Chromium is better than that of Firefox. Others argue that Firefox is better because some of its code is written in Rust. Of course, none of these can be a complete reason. We should consider everything. If you can help in this section, please let me know.[2]

Here, I want to talk about why you should also use Firefox-based browsers.

Currently, all major browsers, except for Firefox and Safari, are based on Chromium. The entire web is becoming dominated by Chromium[3][4], which is concerning. This trend reduces innovation and creativity, putting the web in the hands of a single entity.

Mozilla's revenue primarily comes from its user base. For instance, Google pays Mozilla to make its search engine the default in Firefox.

- Why?

→ This allows Google to maintain its market share, keep users within its ecosystem, and gather more data about individuals. Remember when we discussed how important information and presence in an ecosystem are? In 2021, Google paid Apple around $15 billion to ensure that its search engine is the default for Safari, Mac, and Siri![5]

For Mozilla, this figure is about $400 to $450 million annually from 2021 to 2023[6], which accounts for roughly 90% of its revenue. If Mozilla's user base declines, its income will decrease, leading to potential layoffs and a decline in the organization. This would be detrimental to the web, as we would lose out on the valuable technologies that Mozilla brings to the table. It would also mean losing an organization that prioritizes our privacy.

You can customize Firefox using the settings available on the *about:config* page. Alternatively, you can use browsers like *Mullvad* that are configured out of the box. *Librewolf* is another Firefox-based browser that is worth mentioning. The Tor Browser is also Firefox-based.

[2] Might be good: `https://madaidans-insecurities.github.io/firefox-chromium.html`
[3] `https://gs.statcounter.com/browser-market-share`
[4] `https://data.firefox.com/dashboard/user-activity`
[5] `https://www.forbes.com/sites/johanmoreno/2021/08/27/google-estimated-to-be-paying-15-billion-to-remain-default-search-engine-on-safari/`
[6] `https://www.pcmag.com/news/mozilla-signs-lucrative-3-year-google-search-deal-for-firefox`

7.1.2 Extensions and addons

You can use extensions to enhance your browser's privacy. *NoScript* is a fantastic extension that lets you control which scripts run on a website. While it can be a bit annoying at first, it significantly improves your privacy.

uBlock Origin is another great extension that blocks contents you don't like (such as ads and trackers). I personally don't like to block ads, as I believe in supporting websites that provide valuable content. Without ads, many websites would struggle to survive as you don't pay for their services.

An interesting idea to hide real browsing history is to have a lot of fake browsing history. This way, it becomes difficult to identify which browsing history is real and which is fake. See *Adnauseam*[7] and *TrackMeNot*[8]. (I am not aware about their effectiveness, but the idea is interesting.)

7.1.3 Search Engines

Most search engines log your searches for warrant requests[9] or tracking purposes. This includes keywords you typed but didn't hit enter. It's important to choose a search engine that respects your privacy.

OK that's enough for now.

- Why so little information?

→ For more up to date and more precise information, I recommend you to read their documents.[10]

7.1.4 Reading List

- PrivacyTests.org (Tests different browsers for privacy)[11]

7.2 Deepweb and Darkweb

7.2.1 What is Deepweb?

The deep web is the part of the internet that is not indexed by search engines. It includes things like your emails, bank accounts, and other private information that you don't want to be publicly accessible. The deep web is not inherently bad; it's just not indexed by search engines. It's like a private room in your house that only you have access to.

[7] `https://adnauseam.io/`

[8] `https://www.trackmenot.io/`

[9] *Exclusive: Government Secretly Orders Google To Identify Anyone Who Searched A Sexual Assault Victim's Name, Address Or Telephone Number*: `https://www.forbes.com/sites/thomasbrewster/2021/10/04/google-keyword-warrants-give-us-government-data-on-search-users/`

[10] Brave, Firefox

[11] `https://privacytests.org/`

The dark web is a term for a part of the deep web that is intentionally hidden. It is used for illegal activities, such as selling drugs, weapons, stolen data, and hacking tools. The website addresses on the dark web are made up of random characters, making them difficult to find by brute force. People use special browsers like Tor to access the dark web. You need to get the address of the website you want to visit from a trusted source, as there are many scams on the dark web.

- Is most of the dark web illegal?

→ No.[35]

7.3 The Onion Router (Tor)

As I showed in figure **??**, everything on the internet is transferred via packets. These packets are like letters; they have a sender and a receiver. Everyone in the middle can see the packet. However, when we talk about anonymity, we want to hide the packet content, sender, and receiver. When using a VPN, the packet content is hidden from the ISP. However, the VPN provider can see:

- Packet content
- Sender
- Receiver

We can't change the sender and receiver for the entire communication (otherwise, communication can't be done!). So we somehow need to split this data into different parties, ensuring that no party has total access to all the information.

7.3.1 Tor comes in

According to a *Top Secret* document leaked from NSA, Tor is:

> The King of High-Secure, Low-Latency Internet Anonymity. There are no contenders for the throne in waiting.[12][13]
>
> We sill never be able to de-anonymize all Tor users all the time.
>
> With manual analysis we can de-anonymize a very small fraction of Tor users, however, no success de-anonymizing a user in response to a TOPI request/on demand.[14]

[12] https://www.scribd.com/doc/173441295/Tor-The-king-of-high-secure-low-latency-anonymity

[13] https://www.theguardian.com/world/interactive/2013/oct/04/tor-high-secure-internet-anonymity

[14] https://web.archive.org/web/20190228101152/https://edwardsnowden.com/wp-content/uploads/2013/10/tor-stinks-presentation.pdf

See how the wonderful guide "How Does Tor Really Work? The Definitive Visual Guide (2023)" explains Tor[15][16]. (It is required to read the guide before continuing this section.)

The primary idea is to let each node know only partial information:
Node 1:

- Nothing about data (as data is encrypted three times)
- Knows the sender (Because it gets the data from the sender)
- No information about the reciever (As it only pass data to node two)

Node 2:

- Nothing about data (as data is encrypted two times)
- No inforamtion about the sender (Because it gets the data from node 1)
- No information about the reciever (As it only pass data to node three)

Node 3:

- Knows the data (As it is the last node and needs to send the data to the reciever). However, if you use HTTPS, the data is encrypted and node 3 can't see the data. Only metadata is visible.
- No information about the sender (Because it gets the data from node 2)
- Knows the reciever (As it pass the data to the reciever)

This way, no node has access to the three pieces of information we discussed.
• What if all three nodes are controlled by the same entity?
→ This is a valid concern. If all three nodes are controlled by the same entity, they can see all the information. However, the probability of this happening is very low. The Tor network consists of thousands of nodes, and it's highly unlikely that all three nodes will be controlled by the same entity.

We don't even need all three nodes; only the first and the last one are enough. The first node knows the sender, and the last node knows the receiver. To be able to map the packet in the first node to the corresponding packet in the third node, an attacker can use timing, packet size, and other features. This is called a *Correlation Attack*.

Nodes of the Tor network are called *Relays*. There are three types of relays: *Guard Relay*, *Middle Relay*, and *Exit Relay*. All relays are public[17] and anyone can run a relay.

[15] `https://skerritt.blog/how-does-tor-really-work/`
[16] `https://support.torproject.org/https/`
[17] Except for bridges, which we will get to later.

Some malicious actors run exit nodes to monitor and change unencrypted traffic (mainly[18] HTTP trafic)[19]. However, Tor regularly checks the exit nodes to see if they are doing something bad. If they are, they are removed from the network.[20][21] This is why it is important to use HTTPS.

7.3.2 Timing Attacks

One drawback of using Tor is that your ISP can see that you are using it. This is because the Tor nodes are public. Additionally, the fingerprint of a Tor connection is different from that of a regular HTTPS connection. Think about a situation where you're a journalist. You're the only one using Tor in your area, and you publish a news article related to your area. Although you are anonymous and nobody sees who published that, you can easily be de-anonymized. This is because the ISP sees that, for example, at 12:03, someone in area A connects to Tor, and at 12:05, a document related to that area is published. So it is highly likely that you are the one who published that document. This is called a *Timing Attack*.

Additionally a security organizations might even collaborate with your ISP to drop your connection and see if that anonymous individual goes offline as well. If they do, it becomes clear that you were the one hiding behind that anonymous persona. When your power goes out or your internet connection drops, that person will also disconnect, revealing the link between you two. They could also perform a DoS attack on that Tor node or on you, and if you get disconnected, it might indicate that it was you.

See also "Correlation Counting Attacks" from " Hitchhiker's Guide"[22].

Timing attacks are serious. There is no way to avoid them unless more people start using Tor. If more people use Tor, it becomes difficult to identify who published that document. This is the exact reason that Tor was made public by the US Navy; they wanted to hide their communication in the crowd. The more people use Tor, the more secure it is. This is why it is important to use Tor even if you don't need it—to help others.

Traffic correlation, will never be completely solved. If a global collaboration between countries occurs, in theory, it would be possible to identify individuals.

Antoher scenario is that if I want to find out who someone is on Tor, I might send them a video link from a well-known site. If that person accesses the site without using Tor, their IP address gets exposed. Since it's a popular site, they might not think it could be a trap to gather their information.

When someone sends you a link, don't click on it right away. It becomes very obvious that you were the one who accessed the website just seconds after receiving the message.

[18] There are other unencrypted traffics also

[19] `https://arstechnica.com/information-technology/2007/09/security-expert-used-tor-to-collect-government-e-mail-passwords/`

[20] `http://gitlab.torproject.org/legacy/trac/-/wikis/doc/ReportingBadRelays`

[21] `https://web.archive.org/web/20220416013422/https://matt.traudt.xyz/posts/2019-10-17-you-want-tor-browser-not-a-vpn/`

[22] `https://anonymousplanet.org/guide/`

Bomb In Harvard

The story goes that a Harvard student wanted to have his final exam canceled. He sent an email to the university using a disposable email service called "Guerrillamail" and claimed that there was a bomb in the university. The email was sent through Tor:

> shrapnel bombs placed in science center, sever hall, emerson hall, thayer hall, 2/4. guess correctly. be quick for they will go off soon.[23]

Keep in mind that when you send an email, the sender's IP address is included in the header.

• How should I know that?

→ That's why I say that without sufficient knowledge, if you take action, the chances of making a mistake are high! If you understand how email works, you would realize this. At the very least, you could have read the privacy policy and documentation of such email services!

In fact, this means that the IP address from Tor was sent and received by the university. So far, everything might seem normal since it's a Tor IP, and there shouldn't be a problem.

That's true, but here's the catch! The FBI noticed that an email had been received from Tor. They suspected that the sender might be at the university at the time the email was sent. The FBI looked into the university's Wi-Fi network to see who was connected to Tor at that time. One of those individuals was named "Eldo Kim." The FBI approached him and asked if he was the one who sent the email, and he confessed that it was indeed him.

As we mentioned, when you connect to Tor, it's evident that you are using it, but they can't tell where you went or what you did. In this case, the FBI wasn't sure if this person had actually sent the email because the Tor network obscures that information. If many people had been using Tor at that time, the FBI wouldn't have been able to make everyone confess.

He could also have not used Tor in the university. As almost all universities, do log the activities of their students, they could have seen that he was using Tor. Additionally, it is a high chance that you would get de-anonymized if you use Tor in a network that you want to connect to that same network. (e.g., using Tor in university to send the message to the university).

• Should we use VPN with Tor? To hide the fact that we are using Tor?

→ Generally speaking, No. VPNs are not for anonymity. They may record your activities and help deanonymize you.[24]

Since the fingerprint of a Tor connection is different from HTTPS and Tor relays are public, some countries are able to block Tor. However, there are some

[23] https://www.justice.gov/usao-ma/pr/harvard-student-charged-bomb-hoax

[24] Might be helpful: https://gitlab.torproject.org/legacy/trac/-/wikis/doc/TorPlusVPN, https://tails.net/support/faq/index.en.html#vpn, https://matt.traudt.xyz/posts/2016-11-12-vpn-tor-not-net-gain/

Tor relays that are private and not public. These are called *Bridges*. They are mostly used in countries that block Tor. If a country blocks Tor, you can use bridges to access it. However, if the country knows the IP addresses of the bridges, it can block them. This is why bridges are not public. For example, meek-google is a bridge that makes Tor traffic look like Google traffic. This is why it is hard to block.[25] Or, see Snowflake.[26] Or, see "Roger Dingledine - The Tor Censorship Arms Race The Next Chapter - DEF CON 27 Conference" from "DEFCONConference" YouTube channel[27] which describes how Tor is blocked in different countries.

mmmmmm

7.3.3 Dangerous behaviors lead to de-anonymizing

In this section, we will explore a number of actions you should avoid while using Tor, as they can lead to the exposure of your true identity. Unfortunately, for years, the conditions for using Tor have not been clearly communicated; instead, it has simply been promoted. This misleading advertising causes users to engage with Tor without proper knowledge, leaving them unaware that they can still be identified!

Don't use Brave, Firefox, or any other browser instead of the Tor Browser for anonymity

You might think about using Firefox while routing your data through Tor and assume that it's as if your browser is Tor, believing you are anonymous. While it's true that your data passes through Tor, many issues can arise. These browsers are not designed for anonymity! You can easily expose yourself in various ways.

One of the smallest issues that can occur is a DNS leak. This means that while your regular data is routed through Tor, your DNS requests are sent normally, revealing your original IP address!

These browsers do not have certain anonymity features enabled. For instance, some settings may inadvertently contribute to creating a unique fingerprint for you. Therefore, it's best to use only the Tor Browser, avoiding unofficial versions or other browsers.

One day, while browsing the Tutanota website, I noticed a suggestion that if you encounter an error with the Tor Browser, you could use another browser that is proxied through Tor.

> If you are getting this warning on the Tor browser, you can Launch another browser, using the running Tor instance as a proxy.[28]

This is a very dangerous recommendation! Especially coming from Tutanota, a service that focuses on protecting people's privacy! It really needs a warning

[25] https://blog.torproject.org/how-use-meek-pluggable-transport/
[26] https://support.torproject.org/censorship/what-is-snowflake/
[27] https://www.youtube.com/watch?v=ZB8ODpw_om8
[28] https://web.archive.org/web/20230823193652/https://tutanota.com/blog/best-encryption-with-kdf

stating that this practice is not suitable for maintaining anonymity, and if you want to stay anonymous, you should avoid it!

For example, the Brave browser displays a message when you open a window that routes data through Tor, stating:

> With Tor connectivity, it becomes more difficult [29] for sites to see your true IP address and for network observers to see what sites you visit. However, if your personal safety depends on remaining anonymous, use the Tor Browser instead.

Tor itslef says:

> We strongly recommend against using Tor in any browser other than Tor Browser. Using Tor in another browser can leave you vulnerable without the privacy protections of Tor Browser.[30]

Download OR Opening Files

Absolutely avoid downloading any files unless it is absolutely necessary!
• Well, I'll download it, but I'll scan it with antivirus software before opening it! Or I'll even check it on VirusTotal to make sure it's not malware!
→ First of all, no antivirus software can detect all malware. Second, sometimes your system may have vulnerabilities that can be exploited just by downloading a file. In the past, there have been vulnerabilities that allowed infections simply through file downloads.

Third, why are you downloading a file from Tor in the first place? Surely it's to remain anonymous. However, scanning it with antivirus software could actually lead to your identification.

• How so?
→ Remember when I mentioned that most antivirus programs send file hashes or sometimes the files themselves to the company for analysis? It can be determined that someone downloaded a specific file from Tor, and that file was later scanned by an antivirus program. This could lead to your identification.

• Alright! I'll open the file in a virtual machine and a sandbox environment!
→ The Tor Browser can only keep the data passing through it anonymous. However, sometimes when you open certain files, they may establish a connection to a website and send or receive information, which can lead to your real IP being exposed outside the browser. It doesn't matter if it's in a sandbox or elsewhere because the connection is made outside of Tor, revealing your IP.

Or the file could even be malicious, like a harmful image, and opening it could lead to your exposure! [31] Tor Project says[32]:

[29] Not impossible
[30] `https://support.torproject.org/tbb/tbb-9/`
[31] interesting: `https://blog.reversinglabs.com/blog/malware-in-images`
[32] `https://support.torproject.org/`

> Tor Browser will warn you before automatically opening documents that are handled by external applications. **DO NOT IGNORE THIS WARNING**. You should be very careful when downloading documents via Tor (especially DOC and PDF files, unless you use the PDF viewer that's built into Tor Browser) as these documents can contain Internet resources that will be downloaded outside of Tor by the application that opens them. This will reveal your non-Tor IP address. If you must work with files downloaded via Tor, we strongly recommend either using a disconnected computer, or using dangerzone[33] to create safe PDF files that you can open. Under no circumstances is it safe to use BitTorrent[34] and Tor together, however.

Don't use Tor browser alone!

If a connection is made outside of Tor, your real IP address can be exposed. Here, there are some OSs that are designed to prevent any connection outside of Tor. All data passes through Tor. Additionally, no software is free of bugs[35]. If a website can exploit a vulnerability in your browser, it can de-anonymize you. This is why we prefer to use sandbox environments. Read about Qubes OS, Tails OS[36], and Whonix.

Don't use Windows or MacOS

Windows and macOS are not designed for privacy. No matter how much effort you put into disabling data collection features, there is still telemetry and other features that can leak your information. Additionally, they are closed source. This means that you can't see what is happening in the background. There are more chances that there are backdoors in these OSs.[37] This is why we prefer to use open source operating systems. Read about Qubes OS, Tails OS, and Whonix for anonymity.

Don't Install any other add-ons or change any settings

As I mentioned in the fingerprinting section, any change can be used to identify you. This is why we prefer to use the Tor Browser as it is. It is designed to make all users look the same. If you change something, you are not blending in.

The only change you're allowed is to change the protection level ("Standard", "Safer", "Safest") in the Tor Browser.

It is also obvious that you should not use an outdated version of the Tor Browser.

---

[33] `https://dangerzone.rocks/`

[35] Firefox, Google Chrome

[36] Tails is mostly used in situations where no data should remain on disk after the session.

[37] Why did I say more? Because no one has read all lines of Linux.

Do NOT do illegal things

Doing bad things is wrong! Don't do it! Would you want to harm yourself or others? No! So, don't engage in such actions! Even if it seems funny or engaging, it's not worth it! You will understand this when you get older and face years in prison. It's not worth it! You can't be anonymous forever. You will be caught. It's just a matter of time.

Security organizations regularly monitor websites that are known for illegal activities by using vulnerabilities[38] or dealing with the creator of the website. Some are even created by them as a honeypot.

Some schenarios

Never share any information that you think might identify you. This includes providing your phone number, address, country, or even the continent you live on.

Scenario 1: For example, imagine you are a journalist working on a story. You might inadvertently write something like: "Today, I was passing by a certain place when an incident occurred." This essentially reveals where you live, and they could check the cameras to see who was there at that time!

Well, I won't describe myself! I'll write: "Today, I was informed that..."

Eventually, you might slip up. For instance, if I want to find you, I might write: "By the way, a certain car came by, but it didn't fire any shots."

Then, if you see I'm mistaken, you might correct me by saying: "No! That car came by, but it fired a couple of shots."

In doing so, you inadvertently reveal that you were the one who witnessed the scene! And whether you were there or not, they will definitely check the cameras and so on.

Scenario 2: For instance, suppose you want to leak a sensitive document, but you don't realize that only a limited number of people had access to that specific document. Once it gets leaked, they will know that three people were aware of it. They can then check who has accessed that document recently and track you down.

Keep in mind that company logs are usually very precise. They clearly show which individual accessed which document, when they accessed it, or even if they copied or sent it! So, they can trace you from that as well!

Not only that, some companies change the document slightly for each person they want to share it with. This way, they can identify who leaked the document.[39]

Scenario 3: For example, suppose you upload a screen recording showing that you visited a certain website and performed specific actions, and you share this video on Tor, unaware that you accessed the website using a regular internet connection! This means I can check the logs of that website to see who

[38] FBI Admits It Controlled Tor Servers Behind Mass Malware Attack: `https://www.wired.com/2013/09/freedom-hosting-fbi/`

[39] How Apple catches leakers: From color changes to comma placement: `https://9to5mac.com/2023/05/11/how-apple-catches-leakers/`

accessed that particular section. I could potentially find your real IP address.
• Well, you might not know when I accessed that website. Which day's log do you want to retrieve?
→ First of all, the IPs of exit nodes are public and identifiable, so I just need to filter out the IPs associated with Tor. Additionally, with the video you recorded, I might be able to determine your operating system and apply filters to narrow down your identity even further!

• How can you tell what operating system I'm using?
→ If you're on Windows or Mac, it's obvious. That leaves BSD-based and Linux-based systems. I can often tell which operating system you're using based on certain elements, like the wallpaper, since most operating systems come with customized wallpapers that are included with each new version.

• I can even figure out whether you're using a desktop computer or a laptop.
→ How can you do that?
→ If there's a battery icon in the taskbar, I know it's a laptop (or at least a portable device). Also, you might not have noticed that at some point, the website or the taskbar at the top or bottom of your screen displayed the date or time... that makes it much easier! I can check the logs for that specific day or time.

Scenario 4: They post a Photoshop of a person doing something bad to shame you. You respond to them, saying that the picture is fake, using a fake account registered on the same device or network as your other identity. This is a big mistake! You are linking your two identities together.

Don't use a session for two purposes

Scenario 1: For example, if you log into a website using two different accounts, those accounts can be linked together, revealing that both belong to you. To prevent this, Tor has a feature called "New Identity," which effectively changes the circuit to avoid such connections.

Remember, I've always emphasized that one of the most important things is to read the documentation of the service you are using to understand how to work with it and grasp its nuances! For instance, Tails states that when you use the "New Identity" feature, only the Tor Browser changes, while the operating system retains the previous circuit, which is not ideal. Therefore, it's better to restart Tails.[40]

Be aware that engaging in risky behaviors, such as logging into a website with one identity and then using "New Identity" to log in again shortly after with a new identity, can raise red flags. It's clear that you might be the same person who just changed your connection. For example, if ten people are logged

[40] `https://tails.boum.org/doc/about/warnings/identity/index.en.html#contextual`

in through Tor, and one person logs out, another person logs in right after.

So, keep in mind that while you may have changed your identity, with a little attention, it can be deduced that the person who was active ten minutes ago and did certain actions is the same person returning after ten minutes to perform similar or even different actions.

Scenario 2: You might enter a website using Tor and send the link to someone else within the same Tor network through a secure chat room. Everything happens within Tor. After you send the link, that person logs in, and if I am the owner of the website, I would notice that another Tor IP address entered immediately after yours. This suggests that they are likely a friend or associate of yours, or at least connected to you in some way. Therefore, I can link you two together.

This likelihood increases significantly for websites with fewer visitors, as the chances of two people being on my site at the same time, both using Tor and both navigating to the same page, are quite low!

Use Tor consistently

Even if you don't use Tor even once, you can be de-anonymized. This is further discussed in the "Silk Road" section.

Check configurations every time after an update

After an update, the settings might be changed. So, it is important to check the settings after an update.

Be careful about people and cameras around you

If you are in a public place, there might be cameras that can see you.

7.3.4 Reading List

- Hitchhiker's Guide on Digital Footprint[41] (Highly recommended)
- Hitchhiker's Guide on IRL and OSINT [42]
- Tails OS documentation[43]
- Whonix documentation[44]
- Search for blogs of people who are behind Tor[45]

[41] `https://anonymousplanet.org/guide/#your-digital-footprint`
[42] `IRLandOSINT`
[43] `https://tails.net/doc/`
[44] `https://www.whonix.org/wiki/Documentation`
[45] `https://www.torproject.org/about/people/`, e.g. `https://matt.traudt.xyz/`

7.4 Metadata

Each file on a computer has a certain structure. The computer interprets the data and knows how to show it to you. For instance, for a photo, the data indicates the color of each pixel. Alongside this data, there is metadata. Metadata includes additional information about the file. For instance, for a photo, metadata includes the date and time the photo was taken, the camera model, the location, and even the name of the person who took the photo. This information is stored in the file itself. When you share a photo, you also share this metadata. This is why it is important to remove metadata before sharing a file.

For instance, John McAfee was hiding from the police. During an interview with a journalist, the journalist took a photo of him and shared it on Twitter. The police were able to track him down by analyzing the metadata in the photo. The metadata revealed the location where the photo was taken, leading to his arrest.[46]

Any file you share can contain metadata. This includes photos, videos, documents, and even PDFs. For example, one technique that the police can use to locate a stolen camera is to check the metadata of suspicious photos to see if the camera in question can be identified. This is possible because we know that photos taken with any camera include information about the camera's specifications, model, and the location of where the photo was taken in their metadata. [47]

Think about whether you want to know whether a student created a project by themselves or copied it from the internet. You can check the metadata of the file to see when the file was created. If the file was created before the project was assigned, it's likely that the student copied it from the internet. Or you can check who created the file. If the file was created by someone else, it's likely that the student copied it from the internet.

Metadata can be found in various contexts. For instance, the main content of an email consists of its text and data, but when we say that a service collects email metadata, we refer to information such as the email's subject, sender and recipient details, sending and receiving times, email size, and so on.

It's important to note that metadata might not seem very relevant at first glance (like the time a message is sent), but when combined, it can reveal a wealth of information. For example, if you message someone every day, that person is likely a close friend or family member, or perhaps a work colleague.

This can lead to the creation of a network showing how close you are to certain individuals. For instance, if a terrorist consistently sends messages to someone before carrying out their plans, that person is likely an accomplice. Even without the actual content of the messages, a lot can be inferred from the metadata.

For example, consider an individual involved in a covert operation. While

[46] https://www.npr.org/sections/thetwo-way/2012/12/04/166487197/betrayed-by-metadata-john-mcafee-admits-hes-really-in-guatemala

[47] https://web.archive.org/web/20110220205639/http://www.stolencamerafinder.com/

their activities may be hidden, we can observe that every time this person enters a specific area, an email is sent to a particular address. From this temporal correlation, we can deduce that it is likely this individual who is sending the email.

Well, we've talked a lot about metadata. Is it really that impactful? If you haven't realized its significance through our discussion so far, let me draw your attention to two quotes:

> We kill people based on metadata. –Micheal Hayden[48]

> Metadata absolutely tells you everything about somebody's life. If you have enough metadata you don't really need content.[49] –Stewart Baker[50]

7.4.1 What to do?

Remember, metadata is stored in the binary of the file. Therefore, we can use tools to delete or change that binary.

The first point is that the more complex a file is, the less likely it is that its metadata can be removed. Some types of metadata cannot be deleted at all. Therefore, it's advisable to use simpler formats like `.txt`, which contain minimal metadata.[51]

The second point is that even if you remove metadata from a file, it's possible that the file itself contains information that can be used to identify you. For instance, a photo of you in your room can be used to identify you, even if the metadata is removed.

LibreOffice

In LibreOffice, you can go to `Tools => Options... => LibreOffice` to specify which metadata should be saved. Keep in mind that, due to the complexity of document files, removing metadata from their components can be a bit challenging. Therefore, it's best to delete the metadata from each individual file and image used in your document beforehand.

Linux Tools

- `mat2 file_name` =¿ For most common file formats
- Metadata Cleaner[52] (uses mat2)
- `exiftool -all= file_name`

[48] Retired United States Air Force four-star general and former Director of the National Security Agency, Principal Deputy Director of National Intelligence, and Director of the Central Intelligence Agency: from WikipediaCreative Commons Attribution-ShareAlike 4.0 License

[49] `https://www.wired.com/2015/03/data-and-goliath-nsa-metadata-spying-your-secrets/`

[50] former General Counsel of the National Security Agency

[51] `https://tails.boum.org/doc/sensitive_documents/index.en.html`

[52] `https://metadatacleaner.romainvigier.fr/`

In discussions about privacy and security, always be cautious about trusting others! The person may not have sufficient knowledge. So, ask them why they claim that mat2 and Metadata Cleaner can effectively remove metadata. Is it really effective? What's their source? Are they just talking nonsense?

I learned about these two tools on the Tails website.[53]

Android

Scrambled Exif removes image metadata.

7.4.2 Reading List

- What is Metadata (with examples)[54] A very good website to see real examples of metadata

7.4.3 Silk Road

The story revolves around a website called Silk Road, which was one of the largest drug marketplaces on the dark web. The FBI had been searching for its creator for a long time. However, as you may know, the Tor network concealed both the identities of its users and the real IP addresses of its servers.

In an effort to find the creator, the police began searching the internet for old discussions about the website. They hoped that by connecting with people who were among the first to talk about Silk Road, they might get closer to identifying its creator.

They came to a post[55] by "Altoid" on a website about mushrooms:

> I came across this website called Silk Road. It's a Tor hidden service that claims to allow you to buy and sell anything online anonymously. I'm thinking of buying off it, but wanted to see if anyone here had heard of it and could recommend it. I found it through silkroad420.wordpress.com, which, if you have a tor browser, directs you to the real site at http://tydgccykixpbu6uz.onion. Let me know what you think...

They get suspicious. They searched about "Altoid". They got several results:

Result 1

> What an awesome thread! You guys have a ton of great ideas. Has anyone seen Silk Road yet? It's kind of like an anonymous amazon.com. I don't think they have heroin on there, but they are selling

[53] https://tails.boum.org/doc/sensitive_documents/metadata/index.en.html#index2h1

[54] https://dataedo.com/kb/data-glossary/what-is-metadata

[55] https://www.shroomery.org/forums/showflat.php/Number/13860995

> other stuff. They basically use bitcoin and tor to broker anonymous transactions. It's at http://tydgccykixpbu6uz.onion. Those not familiar with Tor can go to silkroad420.wordpress.com for instructions on how to access the .onion site.[56]

Result 2

> IT pro needed for venture backed bitcoin startup
>
> ... If interested, please send your answers to the following questions to rossulbricht at gmail dot com[57]

Result 3

> How can I connect o a Tor hidden service using cURL in PHP?[58]

The email behind this post was 'rossulbricht@gmail.com'; he later changed his username on StackOverflow to 'Frosty'.

The pieces of the puzzle are starting to come together. Ross, who is interested in Silk Road, is looking for someone to help with his business who is an expert in Bitcoin. Meanwhile, our main character is facing some issues with the hidden service...

At one point, when they suspected Ross Ulbricht, an undercover FBI agent was investigating him and stumbled upon his YouTube channel. He noticed that Ross's YouTube username was familiar: 'ohyeaross.'[59] The agent recalled that whenever he chatted with the admin of Silk Road, that person would write 'Oh yea' instead of 'Oh Yeah,' omitting the final 'h.' This became another reason for them to be more certain that Ross was indeed the admin of the site.

NOTE: Try to avoid unnecessary chatting, as your writing style might reveal your true identity. This is especially important in a world where artificial intelligence can scan the entire web for writing patterns similar to yours, potentially leading to your identification in the clear net. If it's essential to communicate, make sure not to use your real-world writing style at all. How you code is also a part of your writing style. See also [36].

Due to a problem with the website's configuration, the IP address of the main Silk Road server had been exposed. The FBI agents had gone to the server and noticed that several admin IPs had connected without using Tor[60]. In fact, they discovered that one user had accessed it from a coffee shop.

Can you guess where that coffee shop was? That's right, it was near Ross's home! It seems that the puzzle is becoming more and more complete.

Although the evidence is strong, they still wanted to catch Ross in the act. They wanted to catch him while he was logged in to the website.

[56] `https://web.archive.org/web/20150319080002/https://bitcointalk.org/?topic=175.70\protect\leavevmode@ifvmode\kern+.2777em\relaxwap2`

[57] `https://bitcointalk.org/index.php?topic=47811.msg568744#msg568744`

[58] `https://stackoverflow.com/questions/15445285/how-can-i-connect-to-a-tor-hidden-service-using-curl-in-php`

[59] `https://www.youtube.com/@ohyeaross/about`

[60] Which is something wrong they did.

One day, while Ross was sitting in a café and doing his work, the police saw that the admin was logged in. Subsequently, they staged a fake fight behind him. When he turned to see what was happening, one person grabbed his laptop while another apprehended him. This is how our story's Ross was arrested.

Warning

The lesson here is that don't try to start something that strongly requires anonymity until you have gained enough knowledge. Even a small mistake in the beginning can lead to your downfall.

When the police go bad!

Remember when we were discussing privacy and its importance, and we asked who watches the watchers? We mentioned that those in power might abuse it. This is a perfect real-life example! The police go rogue, and their eyes are drawn to all that money they can seize![61]

• Why are all the examples you give about people who did something illegal and got caught by the FBI? We're not doing anything wrong, so we're safe!

→ No! Look, security organizations certainly won't announce, "Hey, look, we just arrested a journalist who was criticizing us!" They won't say that. But when they catch bad people, they make a big deal out of it, saying, "Hey everyone, look, we're ensuring your safety!" This way, they can justify asking for more funding and power.

It's very important not to share sensitive information with anyone. For example, Ross confided in his girlfriend, and she, in turn, told her friend. There was even a time when she was upset with Ross and posted on Facebook that people would find it very interesting to know he was running a drug website. This small detail could have easily exposed Ross.

Additionally, towards the end, Ross had become overconfident, thinking that since they hadn't been able to catch him for such a long time, they would never be able to do so.

You can read the full story in the book 'American Kingpin: The Epic Hunt for the Criminal Mastermind Behind the Silk Road.'[37]. If you're a Persian speaker, I highly recommend you to listen to ChannelBPodcast.[62]

[61] Stealing bitcoins with badges: How Silk Road's dirty cops got caught: https://arstechnica.com/tech-policy/2016/08/stealing-bitcoins-with-badges-how-silk-roads-dirty-cops-got-caught/

[62] 1: https://channelbpodcast.com/archives/3209
2: https://channelbpodcast.com/archives/3220
3: https://channelbpodcast.com/archives/3289
4: https://channelbpodcast.com/archives/3453

7.4.4 Reading List

- Thirteen Years of Tor Attacks[38]
- Speculative Tor Attacks[63]
- Multi-hop VPN[64]
- Combating Traffic Correlation Attacks - Decoy Traffic Mode[65]

f

7.5 Behavior on Social Media

When we want to join a social media platform, forum, website, or email service, the first step is to register. However, there are some important points we should keep in mind during this process.

7.5.1 Registeration

Only Register When Necessary

Don't register unless you have to. There's no need to sign up for every site you visit. Some websites may not be secure and could misuse your cookies, IP address, email address, and your password to compromise your other accounts (Or attackers may hack into their account and use the information). This is especially true for many Iranian websites, which often lack proper security measures.

Avoid Using Personal Information for Registration

When registering, avoid using your birth date, city name, or similar personal information in your ID, username, or account name. For example, names like Alex1994 or AlexLondon2000 can be easily traced back to you.

Try to choose a username that is a bit longer and difficult to guess, making it harder for someone to use brute-force methods to access your account. Often, people looking to find you on other social media platforms will try names similar to yours or variations of your existing accounts using scripts (automated codes that perform tasks).

[63] https://www.whonix.org/wiki/Speculative_Tor_Attacks
[64] Proton VPN: https://protonvpn.com/support/secure-core-vpn/
[65] http://windscribe.com/blog/combating-traffic-correlation-attacks-decoy-mode/

Password

Never use the same password for different accounts. If a website does not securely store your password (e.g., not using strong hash algorithms) and its data is compromised, attackers can gain access to your other accounts that share the same password.

Change your passwords regularly. Because if a website is hacked and your password get leaked, changing it likely (not always. As many websites, doesn't log out other devices logged in) stops the attackers from accessing your account. However, if you don't change it, they can continue to access your account even after the website has been hacked.

While we recommend changing your passwords every few months, this doesn't mean you should get lazy and revert to using old passwords. For instance, some people might change their GitHub password to the same one they used for their Stack Overflow account, or they might just add a letter or two at the end. This is a mistake. If your password has been leaked on another site, attackers can use automation to test variations of it by simply adding extra characters.

For example, if your password is "AlexJia," they might try "AlexJia2003" as a variation. Just adding numbers or letters to the end of an old password is not a secure approach. Make sure to change your password completely and create a new one.

Don't get too creative[66].[67] There are extensive lists of passwords based on popular songs, well-known words from various languages, and different dialects. The best approach is to use a completely random and long password.

If you find it hard to remember such passwords, consider using a password manager. This way, you only need to remember one random password, and with a little practice, you'll be able to memorize it in just a few days.

If you are a developer, please avoid implementing unnecessary restrictions, such as disallowing special characters in passwords. Such limitations reduce the entropy of the password, making it less secure.

Don't use websites provide feadback for your password. For example, some websites offer a service where you can check if your password has been leaked or is it strong enough. This is a bad idea because you are sharing your password with them, and they might not be trustworthy. Even if they claim to be secure, you can't be sure. It's better to use a password manager that doesn't require you to share your password with them.

[66] e.g., mir is not more secure than amir

[67] See *Bad Idea!!! - Lyrics, Quotes & Phrases as PASSWORDS*: `https://passwordbits.com/bad-idea-lyrics-quotes-phrases-as-passwords/#more-1439`

Bibliography

[1] Niels Ferguson, Bruce Schneier, and Tadayoshi Kohno. chapter 23, pages 323–326. John Wiley & Sons, Ltd, 2015.

[2] Daniel Genkin, Adi Shamir, and Eran Tromer. RSA key extraction via low-bandwidth acoustic cryptanalysis. Cryptology ePrint Archive, Paper 2013/857, 2013.

[3] Alon Shakevsky, Eyal Ronen, and Avishai Wool. Trust dies in darkness: Shedding light on samsung's TrustZone keymaster design. Cryptology ePrint Archive, Paper 2022/208, 2022.

[4] Vincent Rijmen and Bart Preneel. Cryptanalysis of mcguffin. In *Fast Software Encryption: Second International Workshop. Leuven, Belgium, 14-16 December 1994, Proceedings*, volume 1008 of *Lecture Notes in Computer Science*, pages 353–358. Springer, 1994.

[5] Connie J Mulligan, Edward B Quinn, Dima Hamadmad, Christopher L Dutton, Lisa Nevell, Alexandra M Binder, Catherine Panter-Brick, and Rana Dajani. Epigenetic signatures of intergenerational exposure to violence in three generations of Syrian refugees. *Scientific Reports*, 15(1), 2 2025.

[6] Meta removes Iran-based fake accounts targeting Instagram users in Scotland, 1 2022.

[7] Louise Matsakis. How the West Got China's Social Credit System Wrong. 7 2019.

[8] Ron Wyden. DNI Clapper tells Wyden the NSA does not collect data on millions of Americans, 3 2013.

[9] CNBC. Yahoo Security officer confronts NSA Director — CNBC, 2 2015.

[10] Niels Ferguson, Bruce Schneier, and Tadayoshi Kohno. *Hash Functions*, chapter 5, pages 77–88. John Wiley & Sons, Ltd, 2015.

[11] Dan Boneh, Henry Corrigan-Gibbs, and Stuart Schechter. Balloon hashing: A memory-hard function providing provable protection against sequential attacks. Cryptology ePrint Archive, Paper 2016/027, 2016.

[12] P h c. phc-winner-argon2/argon2-specs.pdf at master · P-H-C/phc-winner-argon2.

[13] Ilya Mironov et al. Hash functions: Theory, attacks, and applications. *Microsoft Research, Silicon Valley Campus*, 10, 2005.

[14] Bruce Schneier. *Applied Cryptography: Protocols, Algorithms, and Source Code in C*. John Wiley & Sons, 2 edition, 1996.

[15] Alex Biryukov and Dmitry Khovratovich. Related-key cryptanalysis of the full AES-192 and AES-256. Cryptology ePrint Archive, Paper 2009/317, 2009.

[16] John T. Kohl. The use of encryption in kerberos for network authentication. In Gilles Brassard, editor, *Advances in Cryptology — CRYPTO' 89 Proceedings*, pages 35–43, New York, NY, 1990. Springer New York.

[17] Niels Ferguson, Bruce Schneier, and Tadayoshi Kohno. *Block Cipher Modes*, chapter 4, pages 63–76. John Wiley & Sons, Ltd, 2015.

[18] D. Dolev and A. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–208, 1983.

[19] Martin Gardner. Mathematical games. *Scientific American*, 237(2):120–124, 8 1977.

[20] Brian P. Hayes. The Magic Words are Squeamish Ossifrage. *American Scientist*, 82(4):312–316, 7 1994.

[21] Bruce Schneier. Security pitfalls in cryptography. 1998.

[22] Python Software Foundation. secrets — generate secure random numbers for managing secrets. Accessed: 2024-12-06.

[23] Python Software Foundation. Pep 506 – adding a secrets module to the standard library, 2015. Accessed: 2024-12-06.

[24] Daniel J. Bernstein. *Introduction to post-quantum cryptography*, pages 1–14. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[25] Ivan Damgård, Helene Haagh, Rebekah Mercer, Anca Nițulescu, Claudio Orlandi, and Sophia Yakoubov. Stronger security and constructions of multi-designated verifier signatures. Cryptology ePrint Archive, Paper 2019/1153, 2019.

[26] Elaine Barker. *Recommendation for key management:*. May 2020.

[27] Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

[28] Santiago Cuéllar, Bill Harris, James Parker, Stuart Pernsteiner, and Eran Tromer. Cheesecloth: Zero-knowledge proofs of real-world vulnerabilities, 2023.

[29] Peter Gutmann. Secure deletion of data from magnetic and solid-state memory. In *Proceedings of the sixth USENIX security symposium, san jose, CA*, volume 14, pages 77–89, 1996.

[30] Sriram Sami, Yimin Dai, Sean Rui Xiang Tan, Nirupam Roy, and Jun Han. Spying with your robot vacuum cleaner: eavesdropping via lidar sensors. In *Proceedings of the 18th Conference on Embedded Networked Sensor Systems*, SenSys '20, page 354–367, New York, NY, USA, 2020. Association for Computing Machinery.

[31] C. Metzler, D. Lindell, and G. Wetzstein. Keyhole Imaging: Non-Line-of-Sight Imaging and Tracking of Moving Objects Along a Single Optical Path. *IEEE Transactions on Computational Imaging*, 2021.

[32] Sarah Bird, Ilana Segall, and Martin Lopatka. Replication: Why we still can't browse in peace: On the uniqueness and reidentifiability of web browsing histories. In *Sixteenth Symposium on Usable Privacy and Security (SOUPS 2020)*, pages 489–503. USENIX Association, August 2020.

[33] Yinzhi Cao, Song Li, and Erik Wijmans. (cross-)browser fingerprinting via os and hardware level features. In *Network and Distributed System Security Symposium*, 2017.

[34] Artur Janc, Krzysztof Kotowicz, Lukas Weichselbaum, and Roberto Clapis. Information leaks via safari's intelligent tracking prevention, 2020.

[35] Daniel Moore and Thomas Rid and. Cryptopolitik and the darknet. *Survival*, 58(1):7–38, 2016.

[36] Farkhund Iqbal, Hamad Binsalleeh, Benjamin C.M. Fung, and Mourad Debbabi. Mining writeprints from anonymous e-mails for forensic investigation. *Digital Investigation*, 7(1):56–64, 2010.

[37] Nick Bilton. *American kingpin: The epic hunt for the criminal mastermind behind the Silk Road*. Penguin, 2018.

[38] B Evers, J Hols, E Kula, J Schouten, M Den Toom, RM Van der Laan, and JA Pouwelse. Thirteen years of tor attacks. *Computer Science, Delft University of Technology*, 2016.