Authentication, passwords and digital signing

Marko Hölbl

Annotation

This course introduces the concept of authentication and passwords as well as the underlying concepts and technologies. Additionally, the role of digital signing in authentication and its background concepts and technologies are presented.

Objectives

This course provides basic information about authentication, its elements and password-based authentication, and how to adequately protect passwords on both the user and the authenticator sides. The concepts of password management, multi-factor authentication, and passwordless authentications are discussed.
Moreover, information on the technical background of digital signing is presented, including hash functions, public-key cryptography and the public key infrastructure. Lastly, digital signing as a means of authentication is presented.

Keywords

passwords, authentication, digital signatures, public key infrastructure, hash functions

Date of Creation

06.01.2022

Duration

15 hours

Language

English

License

ISBN

Literature

[1] Batten, L. M. (2013). Public key cryptography: applications and attacks, John Wiley & Sons.

[2] Boonkrong, S. (2021). Authentication and Access Control: Practical Cryptography Methods and Tools, Springer.

[3] Buchmann, J., et al. (2013). Introduction to public key infrastructures, Springer.

[4] Burnett, M. (2006). Perfect password: Selection, protection, authentication, Elsevier.

[5] Grassi, P. A., et al. (2017). "**NIST** special publication 800-63b: digital identity guidelines." National Institute of Standards and Technology (**NIST**).

[6] Grimes, R. A. (2020). Hacking Multifactor Authentication, John Wiley & Sons.

CHAPTER 1

Introduction

DEFINITION

Authentication is the process of verifying the identity of someone or something.

This is done by using the information provided by the entity which needs to be checked. The authentication process in private and public computing systems, such as computer networks, often requires someone, usually the user, to use credentials issued by the system to log on. The fact that a user has a password is supposed to prove that they are genuine. The most common authentication method is the username and password combination. However, there are other options of authentication, including biometry, smart cards, one-use tokens, etc.

In most cases, authentication necessitates the presentation of credentials or an asset to substantiate the assertion that you are who you say you are. The objects of value or credentials are based on a number of distinct characteristics that demonstrate what you know, have, or are.

- **Something you know:** This could be a mental possession of yours, like a password that both the user and the authenticator are aware of. Although this is a cost-effective administrative solution, it is vulnerable to people's memory lapses and other flaws, such as the system administrators' secure storage of password files. The user can use the same password for all system logins. Passwords, passphrases, and PINs (Personal Identification Numbers) are all examples of such a factor.

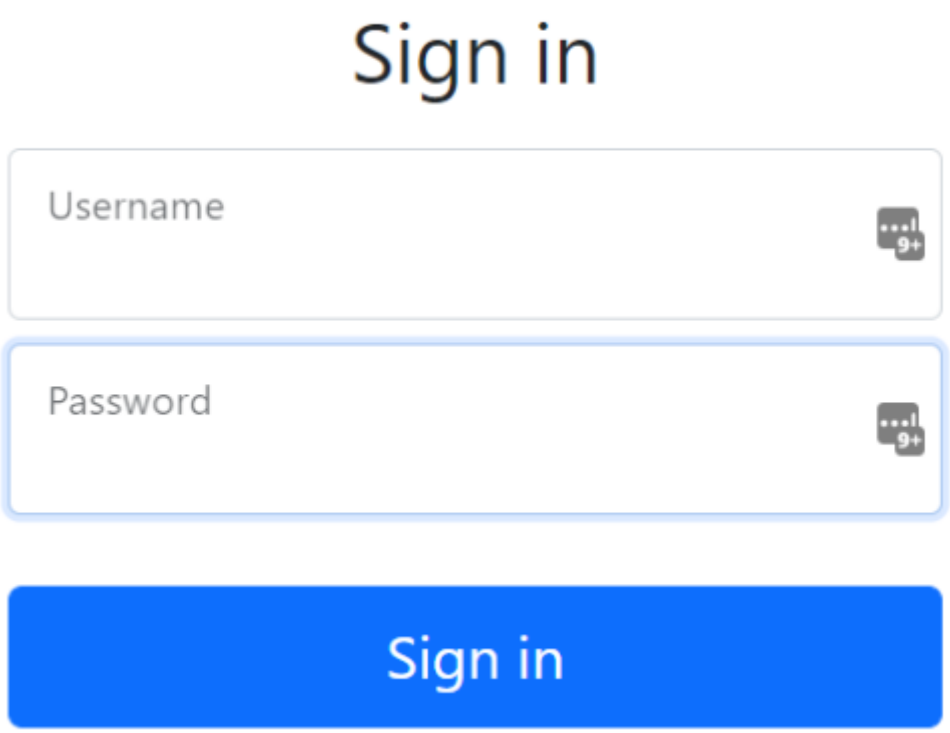


Fig. 1. Example of a user login using a username and password.

- **Something you have**: This can be any type of issued or obtained self-identification token or tag, including smart cards, hardware tokens, mobile phones and a variety of other means. Because individual physical identifications are difficult to abuse, this form is safer than the first approach (something you know). For example, losing a smart card is harder than remembering the card number.

Fig. 2. Examples of hardware tokens for the "Something you have" type of authentication.

- **Something you are**: This is a naturally acquired physical trait, like a fingerprint. This kind of authentication is mostly referred to as biometrics. Although biometrics are simple to use, the costs of obtaining biometric readers can present a problem. Fingerprints, retinal patterns, DNA patterns, and face recognition are examples of this factor.

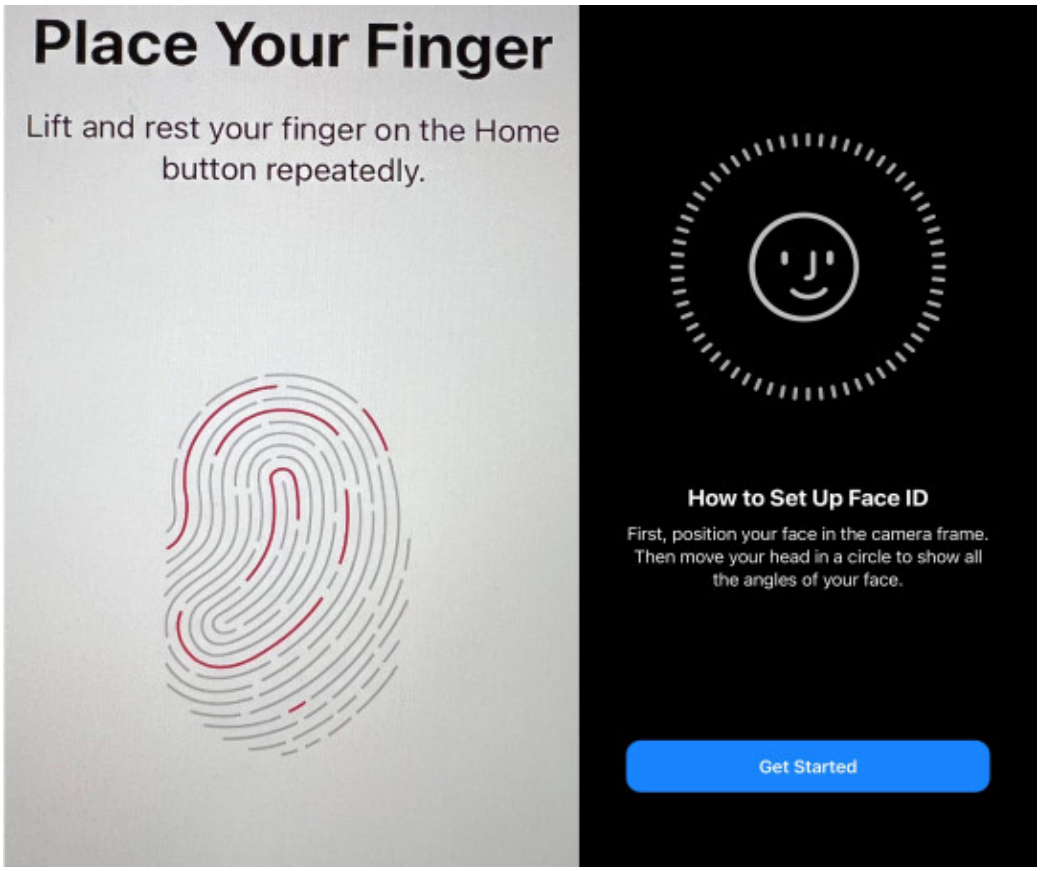


Fig. 3. Examples of biometric authentication using a smartphone.

ADVANTAGE

If a system systematically asks for numerous authentication factors, it can achieve robust security.

DISADVANTAGE

However, too frequent authentication can have the opposite effect, risking the convenience of the user.

Interaktivní prvek

Another way of looking at authentication is by means it employs and consequentially takes one of the three forms:

- **Basic authentication** involves a server. For example, the server keeps a user file with passwords, usernames, and some other essential authentication data. This is the most prevalent method of user authentication. However, it has several flaws, including forgetting and misplacing authentication information such as passwords.

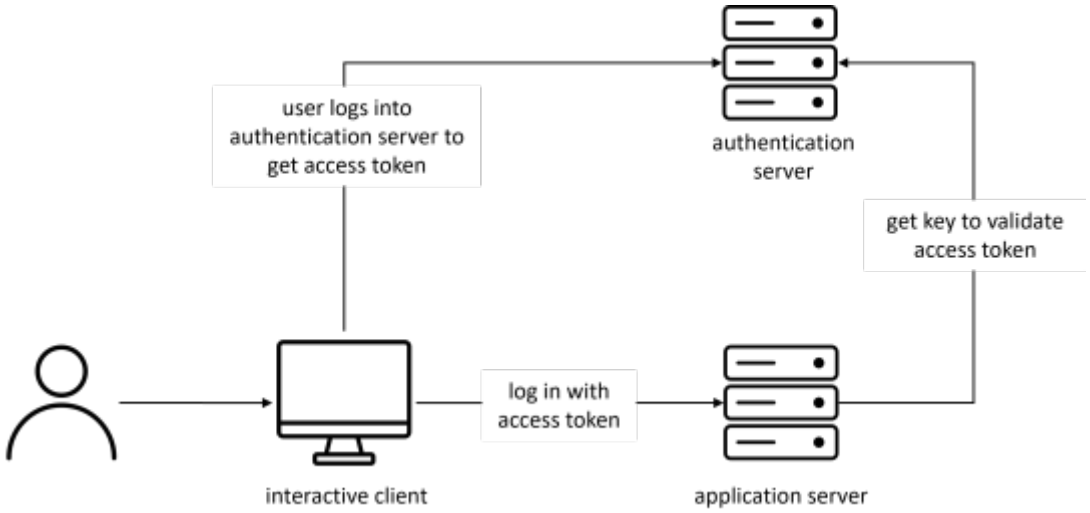


Fig. 4. Basic server-based authentication.

- **Challenge-response** is a method of authentication in which the server or another authentication system issues a challenge to the host requesting authentication and waits for a response. An example of this is using a nonce - a time-variant arbitrary number or sequence of bits that are only used once to verify that the data is only used once.
- **Centralised authentication** refers to a system in which a server authenticates, authorises, and audits network users. These three procedures are carried out in response to server activity. An example of such authentication is Kerberos.

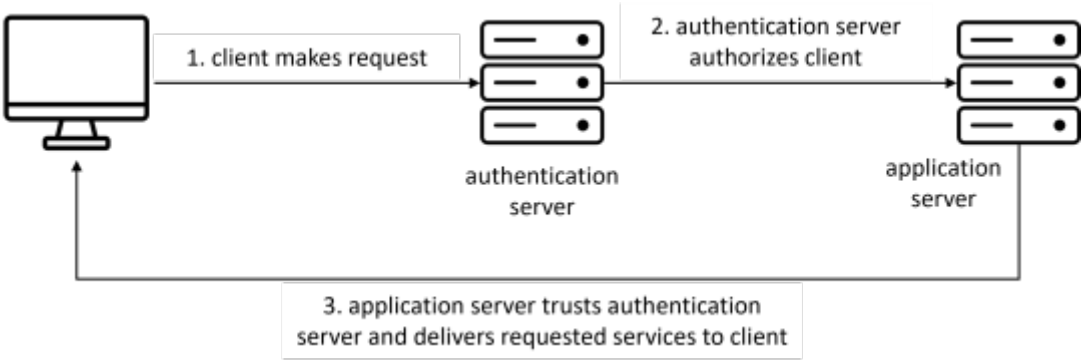


Fig. 5. Centralised authentication.

Interaktivní prvek

CHAPTER 2

Authentication Elements and Process

Authentication requires an authentication process that is based on:

- an entity of or group of entities seeking authentication;
- a distinguishing characteristic from the entity/entities being authenticated;
- an authenticator (usually a server);
- an authenticating mechanism to verify the correctness of the authenticating characteristics;
- an access control mechanism to accept or deny authentication.

The first element is frequently **individuals, processes or systems** that want to gain access to a system. If they are acting individually, they must be prepared to show the authenticator proof that they are authorised to utilise the requested system resource.

The **distinguishing characteristic** of the user is the second authentication element. We talked about these before, and they are divided into something you know, something you have and something you are. Some of these elements may not be sufficient to authenticate the entity fully, and a mix of items from several factors and trust can be utilised to improve authentication and provide stronger assurances.

The **authenticator'**s function is to positively and automatically check the credentials of the entity and determine whether or not that entity is permitted to access the requested system resource. When an authentication request is sent, the authenticator prompts for credentials to complete the authentication process. After that, the authenticator gathers the data and sends it to the authentication mechanism. A user-designated server, a ***Virtual Private Network*** (**VPN**), a firewall, a web server, an enterprise-wide dedicated server, an independent authentication service, or some other type of global identity service can function as the authenticator. Whatever is being used as an authenticator must conduct an authentication procedure that results in some sort of outcome value, such as a token that can be used to ascertain information about the authorised user later.

An overview of this authentication process is depicted in Figure 5.

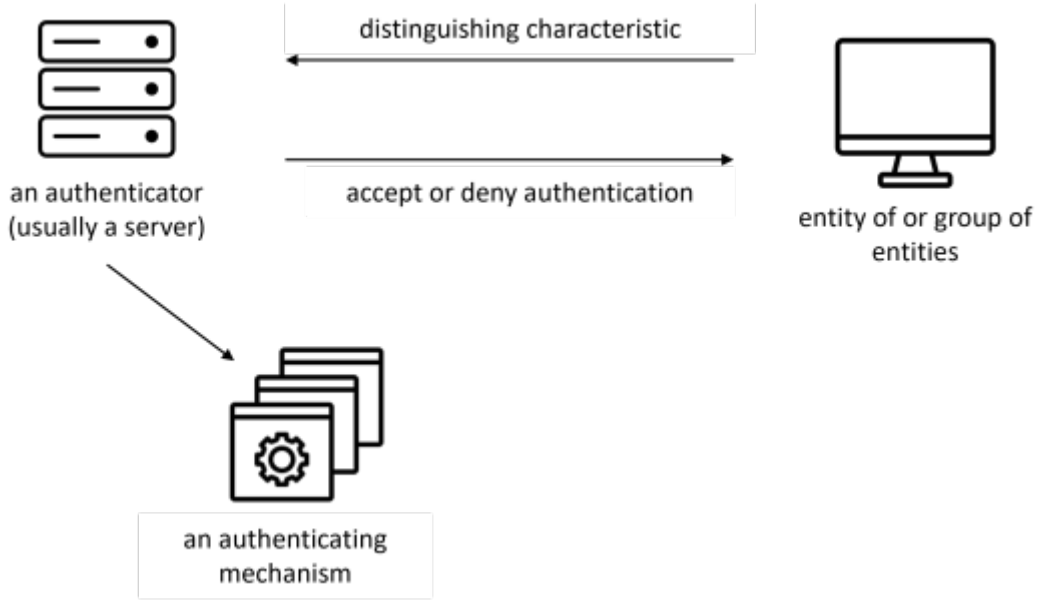


Fig. 6. The basic authentication process and its elements.

The **authentication method** is made up of three pieces that work together to ensure that the user's authenticating traits are present:

- the input,
- the transportation system and
- the verifier.

Interaktivní prvek

An input component serves as the user's interaction with the authentication mechanism. Examples are a computer keyboard, card reader, video camera, telephone, or a comparable device. The captured user identifying items are taken to a location where they will be scrutinised, analysed, and accepted or rejected. However, they must be transported for these products to arrive at this location. As a result, the system's transport section is in charge of passing data between the input component and the element that can verify a person's identity. This information is transferred via a network in modern authenticating systems, where protocols can secure it. The authentication system's final component is the verification, which is the access control mechanism.

Interaktivní prvek

2.1 Types of Authentication

We identified three factors that are utilised in a user's authentication. We also pointed out that while these factors are good, there are items in some that suffer from vulnerabilities. Table 1 shows the shortcomings of each of the factors.

Table 1. Categories of authentication

Factor	Examples	Vulnerabilities
what you know	password, PIN	can be forgotten, guessed, duplicated, easy to get in case of fraud (e.g. phishing)
what you have	tokens, smart card, one-time password sent to your phone number	can be lost, stolen, duplicated
what you are	fingerprint, face, iris	non-repudiable - it cannot be changed in case of abuse

DISADVANTAGE

We mentioned that the first two factors, "what you know" and "what you have," can generate issues for the authenticator because the information provided can be inaccurate. It can be untrustworthy because such factors are subject to a number of well-known problems, including the possibility of items being lost, forged, or easily reproduced. Knowledge can also be forgotten, and knowledge and things can be shared or stolen.

Interaktivní prvek

2.2 Multi-factor Authentication

DEFINITION

Multi-factor authentication (**MFA**) uses at least two different factors (what you know, what you are, and what you have). ***Two-factor authentication*** (**2FA**) is the same, but exactly two factors are used.

Nowadays, if **MFA** is used, it is almost always **2FA**. Typically, the first factor is a password or a PIN (something you know), and the second is usually a bank card, SMS, or a code generated by an app (what you have - i.e. your mobile device). Using fingerprints, retina scans, etc. (what you are) is an option, but it is less often used because additional hardware is required (higher cost).

Multi-factor authentication is a good way to mitigate the risk and reduce the chances of compromised credentials. For example, let us look at a password and app code combination. Even if the site itself is compromised or a password is obtained from somewhere else, the attacker cannot log in because while they can provide the appropriate username and password, they cannot provide the code generated on the mobile device. The stolen password then becomes useless (unless the attacker steals the mobile device as well, but that is not a scalable attack and, therefore, not a serious threat to most people). Meanwhile, the system administrators can still detect unsuccessful attempts to log in and ask a specific user to change their password or all of their users if their system was compromised and all the passwords leaked.

Fig. 7. Multi-factor Authentication.

CHAPTER 3

Password-Based Authentication

The password authentication technique is the most common and easiest to use. In many systems, they are usually set up by default. Reusable passwords, ***one-time passwords*** (**OTP**), challenge-response passwords, and combined approach passwords are all examples of password authentication.

Reusable Passwords

There are two forms of authentication in reusable password authentication: user and client authentication.

- **User Authentication** is the most prevalent sort of authentication, and most users are likely to be familiar with it. It is always started by the user, who submits a request to the server for authentication and authorisation to access a particular system resource. When the server receives the request, it prompts the user for a username and password. The server compares them to copies in its database when they are submitted. The authorisation is provided based on the match.
- **Client Authentication.** Typically, a user seeks authentication and subsequently authorisation to access a system or a set of system resources from the server. Authenticating users does not grant them access to any system resource they desire. User authorisation to utilise the desired resources in the amount requested and no more must be established through authentication. Client authentication is the name for this sort of authentication. It establishes users' identities and allows them to access system resources in a controlled manner.

Because these authentication methods are the most commonly used, they are also the most exploited.

DISADVANTAGE

Additionally, they are also unreliable because people forget them, write them down, share them, and, most critically, they're easy to guess since people choose simple passwords. And they are also vulnerable to surveillance and cracking. Furthermore, weak passwords (e.g., short, to simple structure) are vulnerable to today's super and strong computers, which can brute-force crack them by an exhaustive search.

One-Time Passwords

Session authentication is another name for one-time password authentication. Unlike reusable passwords, which can be used repeatedly, one-time passwords are only used once and then discarded.

ADVANTAGE

Using strong random number generators, they are generated at random. This decreases the likelihood of them being guessed. In many circumstances, they are encrypted before being sent in the clear to limit the chances of their being intercepted.

One-time passwords come in a variety of forms. Examples include S/Key and token passwords. S/Key is a one-time password creation system defined in RFC 1760.

Another example is the so-called TAN number used in Germany in the past (Figure 8). One-time passwords, while typically safer, have a number of drawbacks, including synchronisation issues caused by the elapsed time between the timestamp in the password and the system clock. The password cannot be used once these two timings are out of phase.

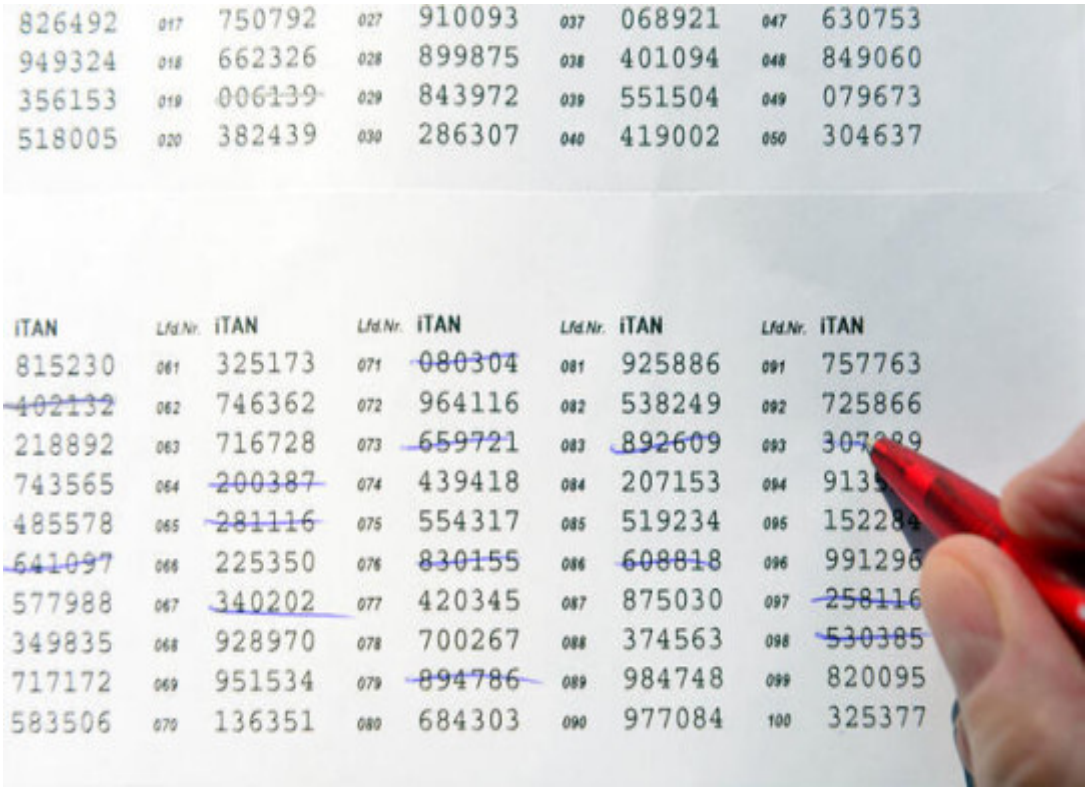


Fig. 8. TAN numbers are an example of one-time passwords (**OTP**).

Some types of one-time passwords (e.g. SMS codes and app-generated codes) are typically used as a second factor when employing **2FA**.

Challenge-Response Passwords

DEFINITION

As a password-based authentication method, challenge-response is a handshake authentication process in which the authenticator challenges the user seeking authentication. To be authenticated, the user must offer the correct response.

Depending on the system, the challenge can take many different shapes. It might be a basic request for a password, a number, a digest, or a nonce. The individual who wants to be authenticated must answer the system's challenge. Nowadays, answers are sent via a one-way function and a password token, which are known as asynchronous tokens. When the server receives the user's response, it double-checks the password.

The most common application of challenge-response authentication is in distributed systems. Despite its popularity, challenge-response authentication faces difficulties due to flaws such as user engagement and trial-and-error attacks. The issue with user engagement is the user's ability to locate the challenge on typically clattered screens. The user must then type a response rapidly.

Depending on the level of security required, the user may be required to recall the lengthy response or be compelled to write it down, after which the user must transcribe and enter it. This has the potential to be prone to errors.

INTERESTING

Some manufacturers have attempted to relieve the user's burden of remembering and typing long strings by automating the majority of the procedure, either through cut-and-paste of the challenge and responses or by a low-level automated process that limits the user's response to yes/no responses.

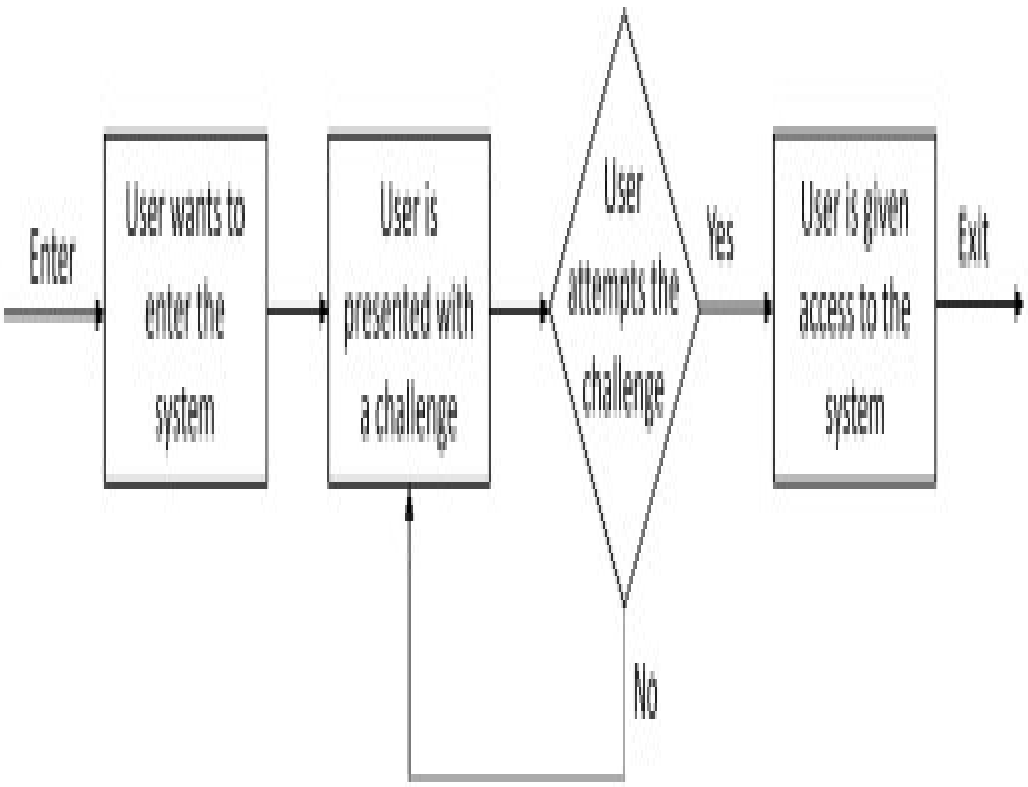


Fig. 9. The process of authentication in a challenge-response system.

It's also worth noting that challenge responses that require passwords might be misused in their most basic form because passwords are relatively easy to obtain. Passwords can potentially be intercepted if they are sent in the clear. However, because a password is not communicated in plaintext across the networks, this does not present a security consideration.

Interaktivní prvek

3.1 Password Security Problems

In cyber-attacks, data breaches are one of the most prevalent types of attacks and goals of an attacker. Therefore passwords as an authentication mechanism are becoming more and more problematic.

The uniqueness of a password is one of its most important characteristics. However, many passwords are everything but this. The most popular passwords and phrases used by people worldwide are listed in Table 2.

Table 2. Top 10 most common passwords, source: cybernews.com.

Passwords
123456
123456789
qwerty
password
12345
qwerty123
1q2w3e
12345678
111111
1234567890

DISADVANTAGE

Additionally, there are other problems regarding passwords. Many people choose to link their websites to something they can readily remember in order to generate simple, memorable combinations. However, this does not make the password unique; the opposite is true.

Cybernews investigation looked at approximately 15 billion entries and classified them into several categories and phrases. Their results show that certain password features are problematic – data related to the user. Additionally, they investigated the length of the passwords in terms of the number of characters they included. Unfortunately, the majority of the passwords used were 8 characters or less.

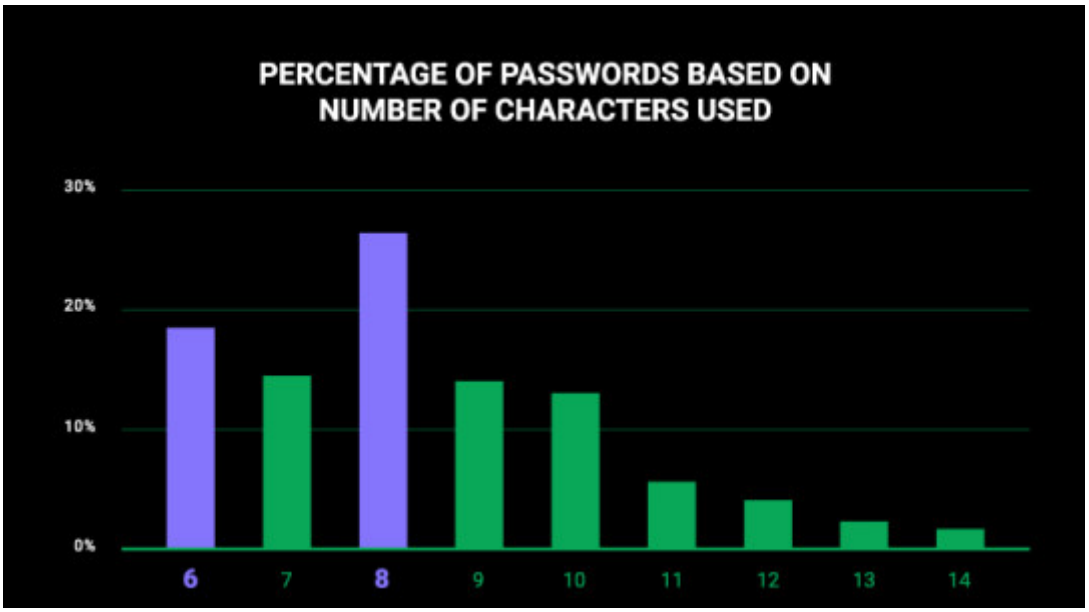


Fig. 10. Statistics on the length of passwords, source: cybernews.com.

There are more effective methods for creating a strong password. Using "heat" as a password element, for example, a simple password may be "letsgoheat" (10 characters), whereas a more difficult password might be "heatromearsenalhjamesp" (a 22-character passphrase). People also generate secure passwords with mnemonic devices, which are preferable because they are frequently long and contain random words with no logical relationship between them, making them simpler to remember for a person but more difficult for an algorithm to crack.

3.2 Attacks on Passwords

Passwords can be attacked in various ways, and generally, attacks can be classified as follows:

- Non-electronic attacks don't require technical knowledge to crack the password. Examples of such attacks include shoulder surfing, social engineering, and dumpster diving.
- Electronic type attacks require technical knowledge. Examples of such attacks include dictionary and brute-force attacks, and rainbow table attacks.

3.2.1 Non-Electronic Attacks

DEFINITION

Social engineering is a type of attack in which the attacker tries to take advantage of people's natural tendency to trust anyone. Exploiting that trust, the attacker quickly obtains the victim's credentials and uses them to get access to his account later.

Phishing, Pharming, and Whaling are just a few examples. Please note that some of them require certain technical skills (e.g. phishing).

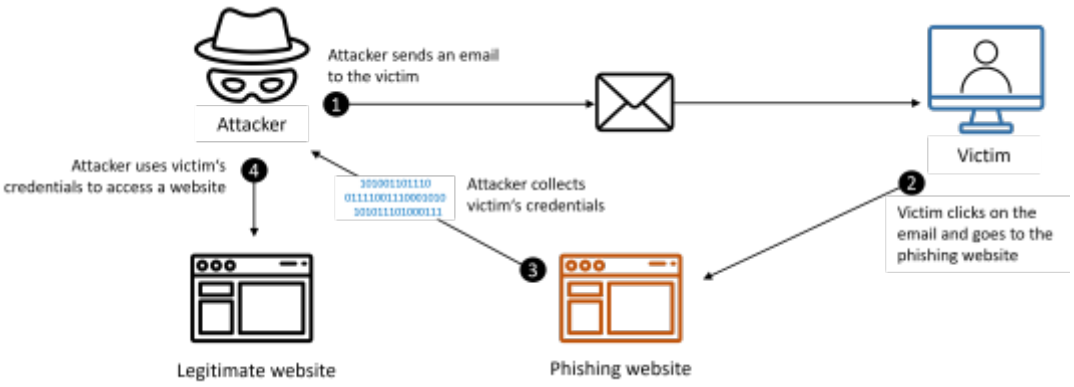


Fig. 11. A process of a phishing attack.

In a shoulder surfing attack, the attacker is standing behind you, watching you type your credentials, which he will subsequently use to access your account.

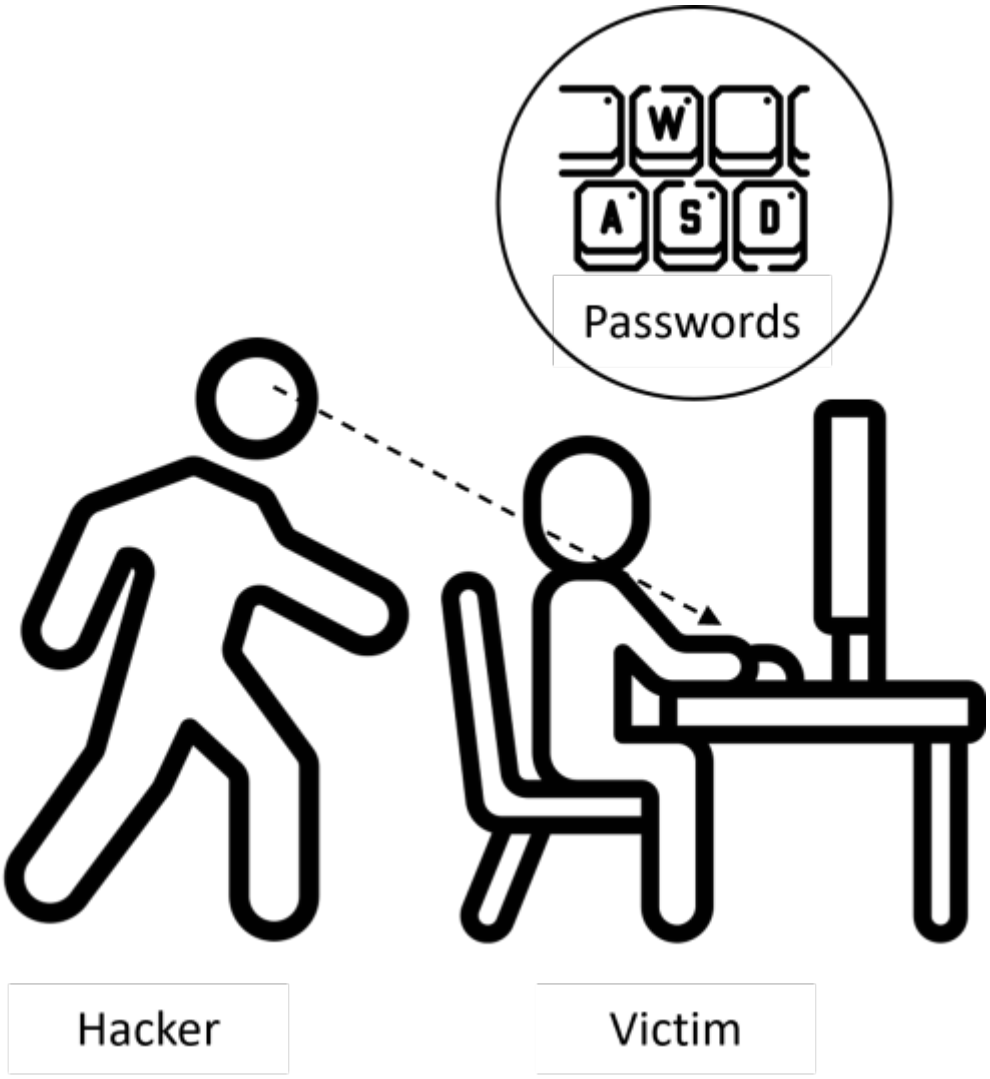


Fig. 12. Example of shoulder surfing.

The Dumpster Diving Attacker discovers something valuable in your garbage, such as your password or the pin to your credit card.

3.2.2 Electronic Attacks

DEFINITION

A **Dictionary Attack** is an attack in which an attacker attempts to get into a password-protected system by using every word in a dictionary as a type of password for that system.

It involves testing all of the strings in a list that has been pre-arranged. Historically, dictionary words were utilised in such attacks (hence the phrase dictionary attack).

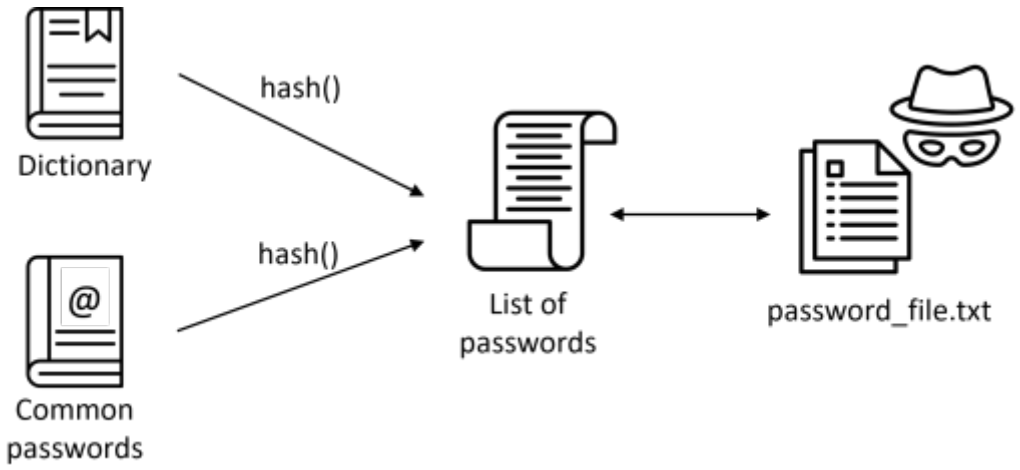


Fig. 13. How a dictionary attack works.

DEFINITION

A dictionary attack only tries the options that are thought to be the most likely to succeed.

Many people have a tendency to choose short passwords that are ordinary words or common passwords, or variants obtained, for example, by appending a digit or punctuation character. Dictionary attacks often succeed because many people have a tendency to choose short passwords that are ordinary words or common passwords or variants obtained, for example, by appending a digit or punctuation character.

Because the available lists cover most typical password formation strategies, dictionary attacks are difficult to overcome with cracking software pattern generation. A safer way is to use a password management tool or a manual method to construct a large password (15 letters or more) or a multi-word password at random.

Brute-Force Attacks

DEFINITION

Simply described, brute-forcing is a password cracking approach in which the attacker tries as many potential password combinations as possible using a set of parameters.

A website may, for example, establish a parameter requiring the password to be between 8 and 16 characters long. The password cracker could start with 00000000 in the most basic variant. Then it may try 00000001, 00000010, 00000100, and so on until it has exhausted all conceivable character combinations.

For a password of length 8 - Each field can be:

- a lowercase alphabet (26 possibilities)
- an uppercase alphabet (26 possibilities)
- a number (10 possibilities 0 through 9)

- punctuation marks or other special characters (33 possibilities)

Considering all this, you can calculate the final number of possible passwords for an 8-character password: 3,025,989,069,143,040, or around 3 quadrillions, and each one is a separate try.

You might think that someone builds a program that goes to a website, types in your username and password, pushes the login button and tries to guess your password. Then they repeat the process three quadrillion times more. However, that isn't the case. If a website takes 2 seconds to load a page, that's 2 seconds of waiting time for each attempt to get a "password incorrect" page. In other words, if the website doesn't lock the login after a specific amount of suspicious attempts, it might take up to 9 quadrillion seconds, or 287.9 million years. In reality, such an attack is carried out using leaked usernames and passwords. These were leaked as a result of a data breach (which happens more often than you think). So, a password can be disclosed in one of two ways:

- Your password is not hashed and is stored in plaintext in an extremely insecure environment. Nothing more than copying and pasting your password would be required of the reader. If your password is password1, for example, anyone reading the contents of the data breach would see password1. In this case, brute-forcing is unnecessary because the website has already handed over your information on a silver platter.
- Your password is hashed rather than stored in plaintext in a more secure environment. If the website hashed your password using the SHA-256 hash function, for example, password1 would appear as 0b14d501a594442a01c6859541bcb3e8164d183d32937b851835442f69d5c94e.

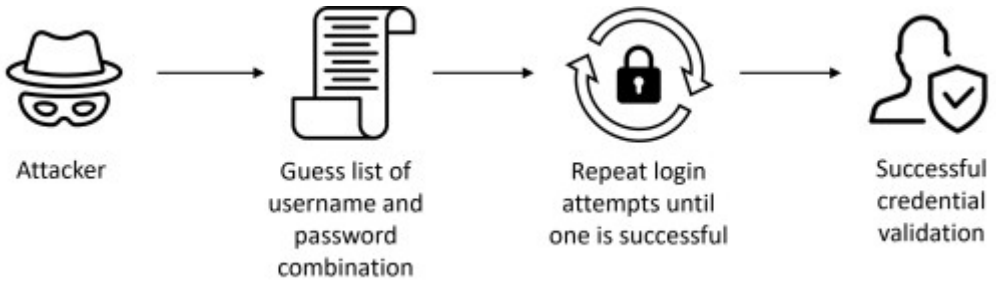
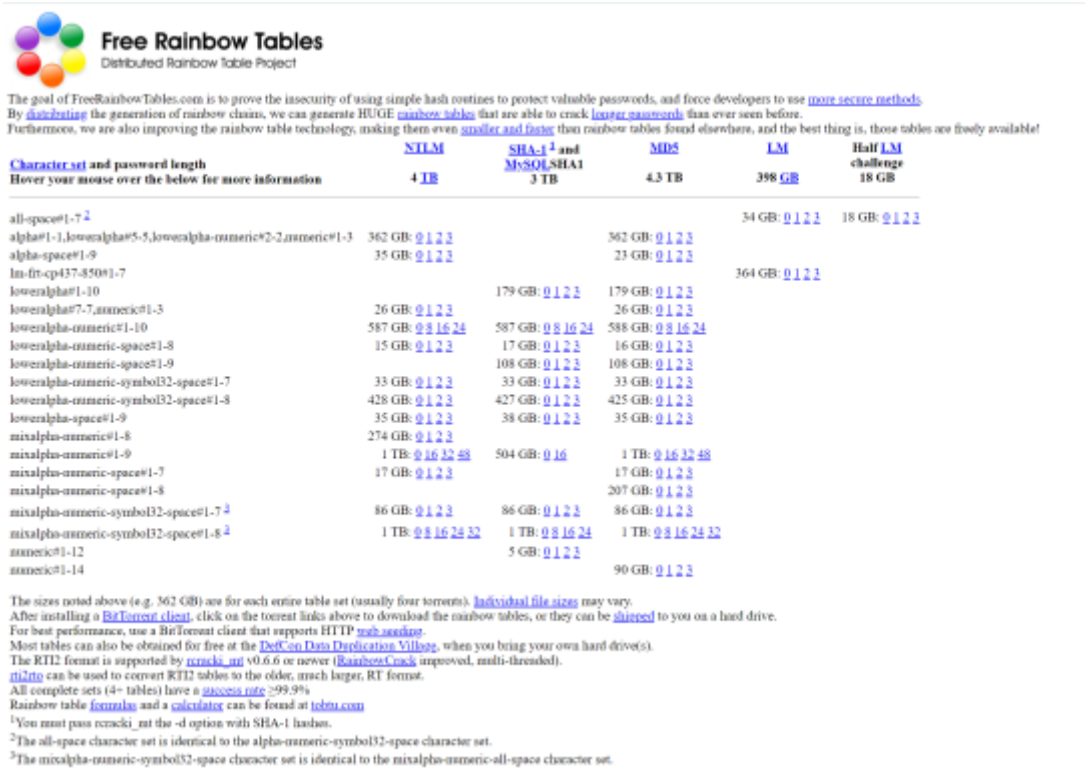


Fig. 14. Brute-force attack on passwords.

Rainbow tables are a special type of brute force password attack. It is intended to crack hash-stored passwords. Because rainbow tables are a pre-computed list of hashes of dictionary terms or previously breached passwords and stored in a database using the hash as the key, there is a time-memory trade-off. Generating a rainbow table can take a long time, but it only needs to be done once. You can look for password hashes and get the appropriate password quickly when done. To put the enormity of these databases into perspective, some rainbow tables can be 7–9TB in size.



Free Rainbow Tables
Distributed Rainbow Table Project

The goal of FreeRainbowTables.com is to prove the insecurity of using simple hash routines to protect valuable passwords, and force developers to use more secure methods.
By distributing the generation of rainbow chains, we can generate HUGE rainbow tables that are able to crack longer passwords than ever seen before.
Furthermore, we are also improving the rainbow table technology, making them even smaller and faster than rainbow tables found elsewhere, and the best thing is, those tables are freely available!

Character set and password length Hover your mouse over the below for more information	NTLM 4 TB	SHA-1 [1] and MySQL SHA1 3 TB	MD5 4.3 TB	LM 398 GB	Half LM challenge 18 GB
all-space#1-7 [2]				34 GB: 0 1 2 3	18 GB: 0 1 2 3
alpha#1-1,loweralpha#5-5,loweralpha-numeric#2-2,numeric#1-3	362 GB: 0 1 2 3		362 GB: 0 1 2 3		
alpha-space#1-9	35 GB: 0 1 2 3		23 GB: 0 1 2 3		
lm-frt-cp437-850#1-7				364 GB: 0 1 2 3	
loweralpha#1-10		179 GB: 0 1 2 3	179 GB: 0 1 2 3		
loweralpha#7-7,numeric#1-3	26 GB: 0 1 2 3		26 GB: 0 1 2 3		
loweralpha-numeric#1-10	587 GB: 0 8 16 24	587 GB: 0 8 16 24	588 GB: 0 8 16 24		
loweralpha-numeric-space#1-8	15 GB: 0 1 2 3	17 GB: 0 1 2 3	16 GB: 0 1 2 3		
loweralpha-numeric-space#1-9		108 GB: 0 1 2 3	108 GB: 0 1 2 3		
loweralpha-numeric-symbol32-space#1-7	33 GB: 0 1 2 3	33 GB: 0 1 2 3	33 GB: 0 1 2 3		
loweralpha-numeric-symbol32-space#1-8	428 GB: 0 1 2 3	427 GB: 0 1 2 3	425 GB: 0 1 2 3		
loweralpha-space#1-9	35 GB: 0 1 2 3	38 GB: 0 1 2 3	35 GB: 0 1 2 3		
mixalpha-numeric#1-8	274 GB: 0 1 2 3				
mixalpha-numeric#1-9	1 TB: 0 16 32 48	504 GB: 0 16	1 TB: 0 16 32 48		
mixalpha-numeric-space#1-7	17 GB: 0 1 2 3		17 GB: 0 1 2 3		
mixalpha-numeric-space#1-8			207 GB: 0 1 2 3		
mixalpha-numeric-symbol32-space#1-7 [3]	86 GB: 0 1 2 3	86 GB: 0 1 2 3	86 GB: 0 1 2 3		
mixalpha-numeric-symbol32-space#1-8 [3]	1 TB: 0 8 16 24 32	1 TB: 0 8 16 24	1 TB: 0 8 16 24 32		
numeric#1-12		5 GB: 0 1 2 3			
numeric#1-14			90 GB: 0 1 2 3		

The sizes noted above (e.g. 362 GB) are for each entire table set (usually four torrents). Individual file sizes may vary.
After installing a BitTorrent client, click on the torrent links above to download the rainbow tables, or they can be shipped to you on a hard drive.
For best performance, use a BitTorrent client that supports HTTP web seeding.
Most tables can also be obtained for free at the DefCon Data Duplication Village, when you bring your own hard drive(s).
The RTI2 format is supported by rcracki_mt v0.6.6 or newer (RainbowCrack improved, multi-threaded).
rti2rto can be used to convert RTI2 tables to the older, much larger, RT format.
All complete sets (4+ tables) have a success rate ≥99.9%
Rainbow table formulas and a calculator can be found at tobtu.com

[1] You must pass rcracki_mt the -d option with SHA-1 hashes.
[2] The all-space character set is identical to the alpha-numeric-symbol32-space character set.
[3] The mixalpha-numeric-symbol32-space character set is identical to the mixalpha-numeric-all-space character set.

Fig. 15. Sizes of rainbow tables from freerainbowtables.com.

Interaktivní prvek

Interaktivní prvek

3.2.3 Password Attack Tools

Password hacking tools are becoming increasingly popular these days, so we'll go over a number of them. For the most part, password cracking tools are used to test the strength of a password or launch a hostile attack. There are numerous online and offline tools dedicated only to breaking passwords. Remote login interfaces, such as SSH and RDP services are targeted by online assaults. On the other side, offline attacks occur after files have been exfiltrated. After that, the passwords are immediately targeted.

Some of the tools available are Hashcat, John the Ripper or THC Hydra.

Hashcat is a cross-platform password recovery program that works with both GPU and CPU. Hashcat was created in 2009 by MIT and is recognised for supporting a wide range of hashing algorithms such as "LM Hash, NT Hash, MD4, MDS, and many others." When it was first developed, this program supported four different sorts of attacks.

- Dictionary attacks: over 14 million passwords, starting with the most popular and ending with the least common. It will guess a password, hash it, and compare the hash to the password it is attempting to crack.
- Combinator attacks: similar to dictionary attacks, but instead of using two-word lists as "dictionaries," it creates a new word list with every word joined with every other word.
- Masks attacks: for example, if you know your account's password is 9 characters long and ends with a digit, you know it will need a $52*10^9$ combination to guess the password, which will take approximately 4 years. However, if you know that the password begins with an uppercase letter and ends with a number, the time will be cut in half.
- Rule-based attack: hashcat can specify what kind of password to try based on how your victim creates a password.
- Brute force attack: it will attempt everything until it finds something "which will normally take a long time because it will try every possible combination."

Video 1

CHAPTER 4

Different Aspects of Password Security

In this chapter, we will cover different aspects of password security, which can be divided into:

- User-centred
- Server centred.

These aspects include guidelines for security passwords, 2-factor authentication, adequate password storage on the server-side etc.

4.1 Secure Password Guidelines and Good Practices

Passwords are the prevalent authentication method because they are the easiest for developers to implement and users to understand and use. However, some conceptual weaknesses are associated with the use of passwords (e.g. poorly selected, easily guessed, etc.). The U.S.A.'s ***National Institute of Standards and Technology*** (**NIST**) regularly updates its recommendations for creating and managing passwords. In one of the recent shifts in the paradigm of password security, they have suggested that users focus on password length over complexity (combinations of special characters, numbers, small or capital letters) because complex passwords are hard to remember. Consequently, users tend to achieve complexity in predictable ways (e.g. adding a number 1 at the end of a password). One way to achieve length is with nonsensical passphrases where words are in a sequence that has no meaning. For the same reason, **NIST** no longer recommends strict character composition rules when creating a password. However, they do recommend regularly comparing passwords (or at least any new passwords) to a list of compromised passwords to identify already revealed and weak passwords. Nevertheless, the recommended minimum length of a password is 12 characters. While regular changes of passwords were recommended in the past, this is no longer the case, as this makes users less likely to remember their passwords after changes and instead start to use the same passwords with small alterations.

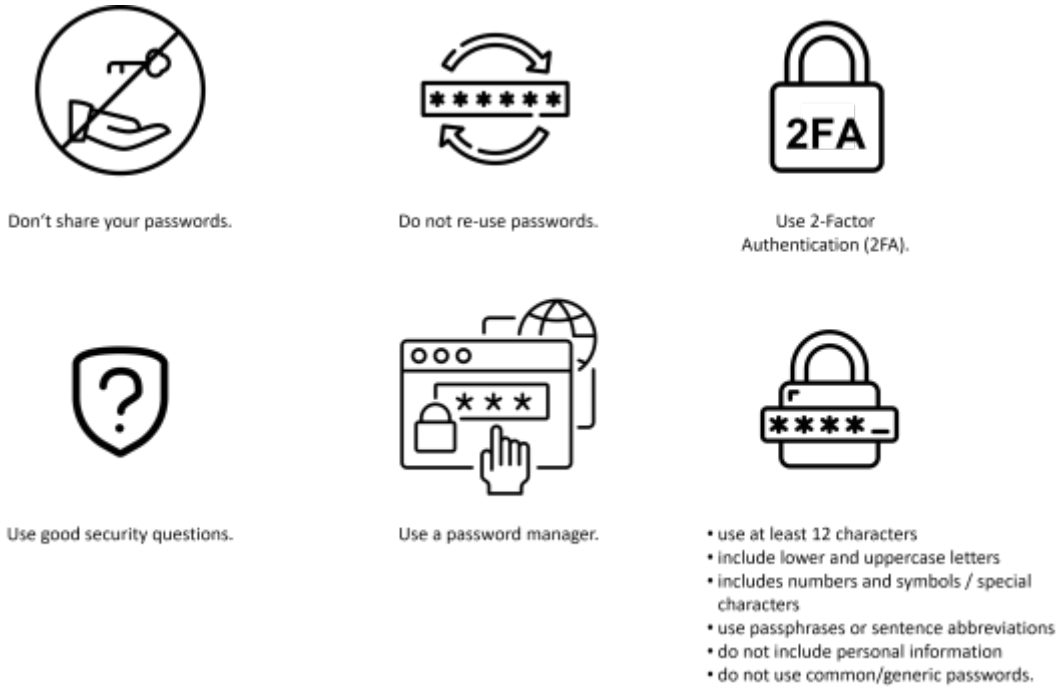


Fig. 18. Best Practices for using Passwords.

For a secure, self-created password, the following requirements should be taken into consideration:

- use at least 12 characters
- include lower and uppercase letters
- includes numbers and symbols / special characters

- use passphrases or sentence abbreviations
- do not include personal information
- do not use common/generic passwords.

Keep in mind that a password should NOT include personal info such as date of birth, a pet's name, your name, or your email address.

Although you can use technical measures to make sure users choose strong passwords, it is impossible to control what users do with those passwords. They can write it on paper next to their computer, share it with other people, or use it for other accounts. The latter is especially dangerous, as using the same password across multiple accounts will compromise all of them if any of the services using the same passwords are breached. This means that if you use the same password for accessing your library and email account, and somebody breaches the security of the library (which should be a lot easier than the servers of a large email service provider – e.g. Google) and steals the password, they can use that information to gain access to your email account. Educating the users is the only real solution to prevent such bad practices. Therefore, users should be educated (on how) to create strong passwords, not write them anywhere accessible, and never reuse the same password.

There are other good practices when it comes to protecting passwords on the user's side:

- Use different passwords for different scenarios
- Use a passphrase
- Use a password manager
- Use two-factor authentication (**2FA**)

While it may appear innocent, using the same password on many websites is a perilous business. Breaches of personal information on consumer websites are getting more widespread. If your information is stolen from a social media site, and you use the same password on your online banking app and a few online shopping sites, the attacker will gain free reign to all of these sites. There are apps and software that can actually alert you when your passwords have been part of a data breach. If your information has been leaked, Google's Password Manager can, for instance, alert you.

Interaktivní prvek

The next good practice is to utilise more than one word. A passphrase is a word sequence that looks like a sentence but should not have any meaning. Your passphrase should not include easily sourced personal information, just as a password should not. You may even utilise generators to create a random string of words for user.

Interaktivní prvek

Finally, one can use different online services to check whether your password was breached. One of the most well-known services of this type is the website haveibeenpwned.com. However, various other web services like this one exist.

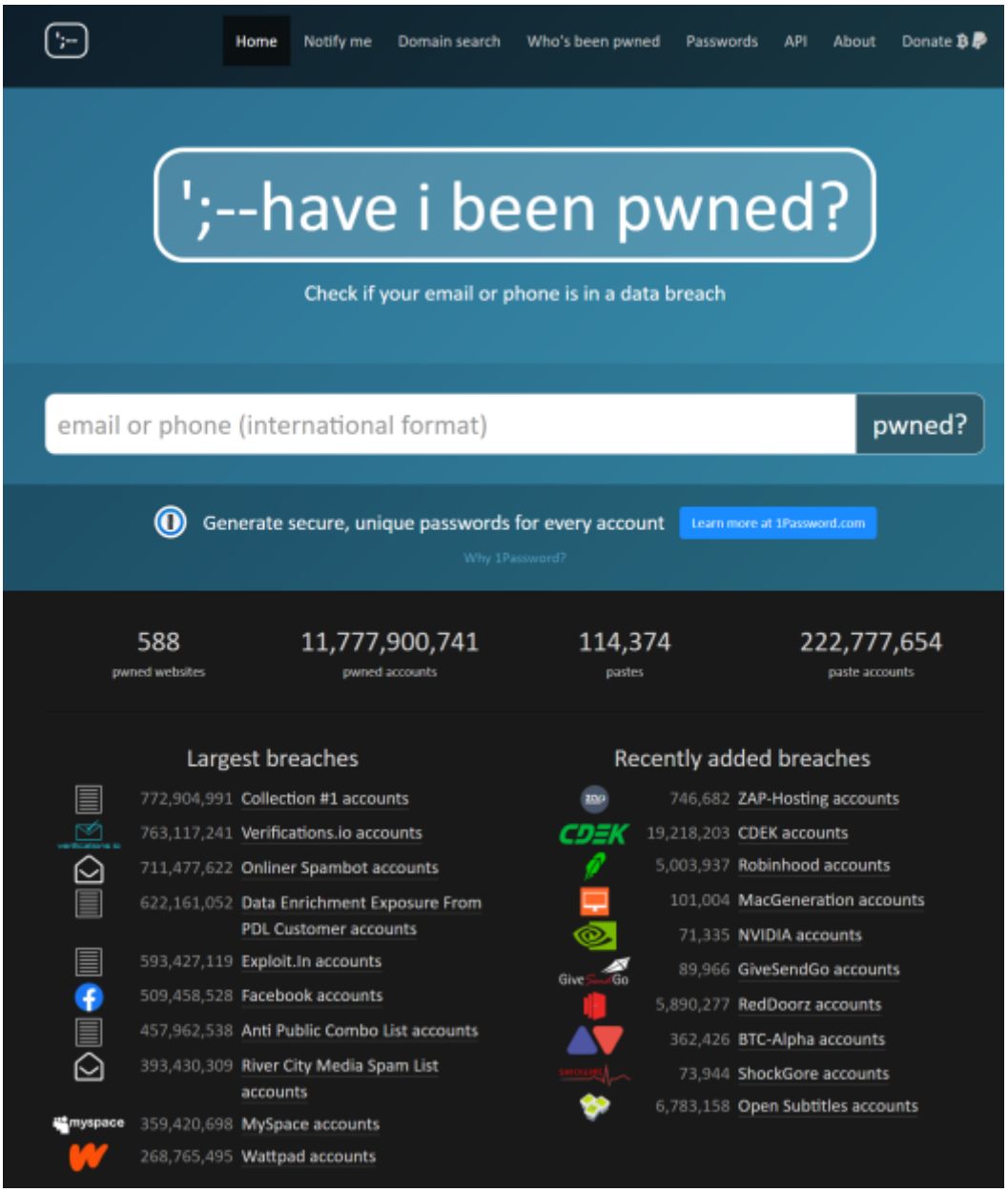


Fig. 19. The "';--have i been pwned?" web services.

4.2 Password Managers

Apart from 2-factor authentication, using a password manager is a good way to keep passwords secure. This option works almost universally, improves the security of your passwords, and makes the login procedure more convenient than less so.

Good password managers, on the other hand, encrypt your data thoroughly. Even if an attacker obtains access to the data file, they must first decode it before extracting any useful information. In most circumstances, this is extremely tough and not worth the effort. Compared to the alternative, using a password manager regularly ensures that you always have a list of accounts that can be updated as needed.

Attackers are deterred by storing data locally rather than in the cloud. Obtaining the data of a single person or family requires a significant amount of effort. You may pick where your password data is stored with good password managers.

Many people have systems, and they tend to share things in common, just like common passwords. These tactics are well-known among attackers. That knowledge has been baked into cracking algorithms, and as a result, systems may be more harmful than beneficial. Let's pretend your system is excellent for the sake of argument. Even so, there is still a risk. As additional breaches expose your password, the likelihood of your system being hacked grows. Once your system has been breached, the attacker can use it to guess your login credentials on other websites. Finally, utilising your system instead of a password manager is actually slower.

Long passphrases are often preferable to passwords. However, many websites do not support them (character and length restrictions). Furthermore, while they are easier to remember than passwords, they do not address the issue of people using the same password on various sites or relying on a system.

However, as the master password for your password manager, which is the key that unlocks all of the password data, using a passphrase is recommended. This enables you to choose and memorise an extremely long master password with ease.

If you consider storing your passwords in a browser, there are a number of reasons why this solution is bad. Browsers don't take password security as seriously as they should because they don't require a master password to access. All you need to do is be logged in to the computer. When utilising computers other than your own, this method is inconvenient. Browser passwords are only effective within the browser. That isn't a viable option these days, as your password is frequently required for both web and mobile apps.

Password managers provide the ability to store more than simple passwords, which is quite useful. You can use a password manager to save and complete credentials, such as credit cards, which can also be done with browsers. You can also keep other sensitive information in the password manager, such as licenses, identities, bank account numbers, and any other information.

So, consider your password manager to be a digital vault that you can carry about with you.

ADVANTAGE

Additionally, password managers also offer other advantages related to usability:

- It's integrated with the areas where you need to use passwords, and it's quick and simple to create, update, and fill passwords.
- It's compatible with many platforms, and passwords are synced.
- It performs admirably in a wide range of circumstances and designs and is usually well-maintained.
- Takes security seriously, taking efforts (strong end-to-end encryption) to ensure that data is secure even if a breach occurs.

In order for a password manager to be effective, you must remember a few key points:

- It should be used on all of your websites. Everywhere. Exceptions only add to the system's vulnerability and complexity (less likely to succeed).
- For each site, create a unique password. If you have the opportunity, make them as long as possible. A minimum of 20–30 characters should be used. The harder it is to crack a password, the longer it is (exponentially harder, actually). Because the password manager will fill them in for you most of the time, you won't have to input them.
- Not every website fully supports password managers, and passwords must occasionally be copied to the clipboard before being pasted into a login form. This isn't very safe because it could expose your password in a variety of ways. After a short period of time, most password managers will automatically remove password data from the clipboard
- Some services continue to impose absurd password requirements, like passwords being at least 12 characters long. Here password managers come in handy since they generate such passwords at random, which is about as secure as I can get given the constraints.
- Make your master password as long as possible, and use a password that is tough to guess. It's also a good idea to change your master password regularly to reduce the danger of it being leaked or intercepted by a key logger.
- If you wish to exchange passwords, have the other person create their own vault and use the password manager's sharing feature to share individual passwords with them.

Some of the popular password managers include:

- LastPass
- Dashlane
- LogMeOnce
- 1Password

- Keeper
- KeePass

Some of them come for free; you need to pay for some. And some are free, but you need to pay for advanced features.

Video 2

4.3 2-Factor Authentication (2FA)

DEFINITION

2FA is an additional layer of protection that verifies that anyone attempting to access an online account is who they claim to be.

The user must first provide their username and password. They will then be requested to submit another piece of information before they can receive access.

Withdrawing money from an ATM is a good example of two-factor authentication. The transaction can only be completed with the correct combination of a bank card (something you have) and a PIN (personal identification number, something you know).

The majority of websites offer the possibility of SMS-based verification. However, mobile devices are gaining ground for **2FA**.

ADVANTAGE

The advantages are evident:

- Because it employs mobile devices that are (typically) carried all the time, no additional tokens are required.
- Dynamically generated passcodes are safer to use than fixed (static) login information since they are continually changing.

DISADVANTAGE

However, there are also disadvantages:

- Inconvenience - whenever authentication is required, users must have a charged mobile phone and in range of a cellular network. Access is often impossible without backup plans if the phone is unable to display messages, such as if it is damaged or shuts down for an update or due to temperature extremes (e.g. winter exposure). It's possible that text messages won't arrive right away, causing additional delays in the authentication procedure due to copy-pasting or putting it in manually.

Please note that SMSs are not as safe as you might expect. SMS text transmissions to mobile phones are insecure and vulnerable to interception. As a result, third parties can steal and use the token. Mobile phone **2FA** is often bypassed during account recovery. Modern smartphones are used to check email and receive text messages. In most cases, email is always logged in. Because the phone may receive the second factor, if the phone is lost or stolen, all accounts for which the email is the key can be hacked. As a result, smartphones merge the two criteria into one. If a user's phone is stolen, the criminal may be able to obtain access to the user's accounts. Hackers can gain access to mobile phone networks by cloning SIM cards. If a device does not support SMS, two-factor authentication via a voice call is a viable option

for practically everyone else.

The Authy mobile App

Suppose you have a smartphone or other mobile device. In that case, you can acquire your two-factor authentication code without using SMS or voice calls by downloading and installing one of the many popular two-factor authentication apps directly to your device. This is a far more secure way to log in using two-factor authentication. Apps like Authy and Google Authenticator produce a **TOTP** (***Time-based One-Time Passcode***) right inside the app.

ADVANTAGE

Even if an attacker managed to persuade your mobile service provider to do a SIM switch, they would still be unable to access your authentication codes. The information required to produce those codes is stored on your actual device rather than on the SIM card.

You'll want to set up your first **2FA** accounts now that you've installed Authy on your phone. This is done by scanning a QR Code (given by the site where you want to secure an account) using the app. You'll probably start protecting other accounts once you've captured your initial code and protected your first account.

Fig. 20. Scanning a QR code in the Authy mobile app.

Now you must choose between keeping all of your **2FA** tokens on a single device and backing them up to the cloud.

DISADVANTAGE

If you go with the first option, then lose, upgrade, or have your device stolen, you'll have to convince every service where you've activated **2FA** to turn it off. After that, when you change your phone, you'll need to get back into your account and manually re-enable **2FA** on each service.

ADVANTAGE

This is why Authy lets you back up your **2FA** tokens to their safe cloud storage, which is only accessible by you, so you can always restore your accounts if you lose, steal, or replace an out-of-date device.

They ask you to set a backup password when you backup **2FA** tokens to the cloud, and they utilise this password to encrypt your data and then sync it to their cloud service. Your data is extremely secure in their cloud platform because they never physically store your password — but it's critical that you remember it.

Fig. 21. Using the Authy mobile app.

After that, it is strongly advised that you install Authy on another device. The app will automatically sync tokens to each device you have Authy installed on if you have synchronised them to the Authy cloud. You might also download the browser-independent Authy Desktop app if you only have one mobile device.

Interaktivní prvek

Video 3

4.4 Aspects of Secure Password Storage (on the Server-Side)

Passwords need to be adequately protected on the authenticator's side (usually the server). Many data breaches are reported daily, and therefore one cannot rely on the security of authenticator's systems. Thus passwords cannot be stored in plaintext and should be stored in a protected manner. However, we will first briefly introduce some concepts needed for understanding safe password storage.

4.4.1 Hashed Password Storage

DEFINITION

A cryptographic hash function takes an input (or message) and returns a fixed-length alphanumeric string.

The string is known as a hash value, a message digest, a digital fingerprint, a digest, or a checksum.

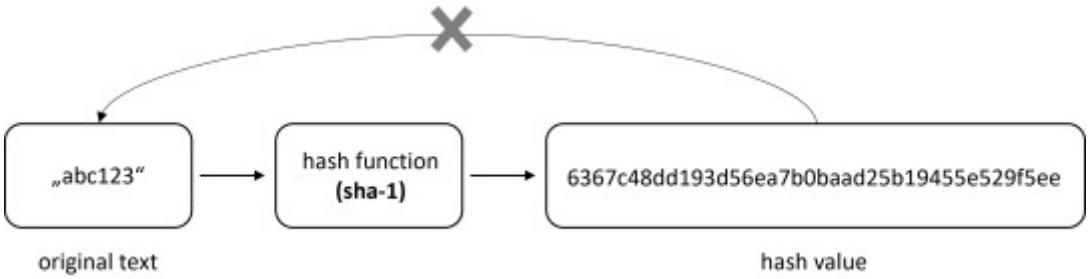


Fig. 22. How hashing works.

Figure 22 depicts the process of hashing. We start with the word "abc123" and use the SHA-1 hash function to get a fixed-size alphanumeric output, which we call a hash value. We won't be able to recover our original input text using this hash value. We cannot reverse a hash value to find the original content since hash functions are one-way and thus irreversible. If the same material is passed through the same hash function, the output/hash result should be the same. So instead of saving the password in plain text, we can hash it with the hashing function and save the hash value.

user_name	password
john	abc123
sam	abc123
alice	xyz456

→

user_name	hash password
john	6367c48dd193d56ea7b0baad25b19455e529f5ee
sam	6367c48dd193d56ea7b0baad25b19455e529f5ee
alice	0772dbe339a885eb2ed73c1fe842d2ef6e9003a3

Fig. 23. Protection of stored passwords using hashing.

When a user tries to log in to the system, the hash function is used to hash the user's password and compare it to the hash value stored in the table. We can allow the user to log in to the system if both hash values are the same. In Figure 23, both john and sam have the same password, "abc123," and their hash

values are the same after applying the hash algorithm. Consider the case where john has access to the database and can see the hash password. Then john may see that his password hash value is the same as sam's password hash value. As a result, john will be able to use sam's credentials to log into the system. To get around this, we can use a technique called salting.

4.4.2 Salted Hashing

Our goal with salted hashing is to make the password hash value unique. Thus, the system generates a random collection of characters called salt. When the user types in a plain text password, the produced random set of characters is appended to it. Then we used the hashing function to extract the hash value from the inserted text (salted hash). In this instance, each user's salt value must be saved.

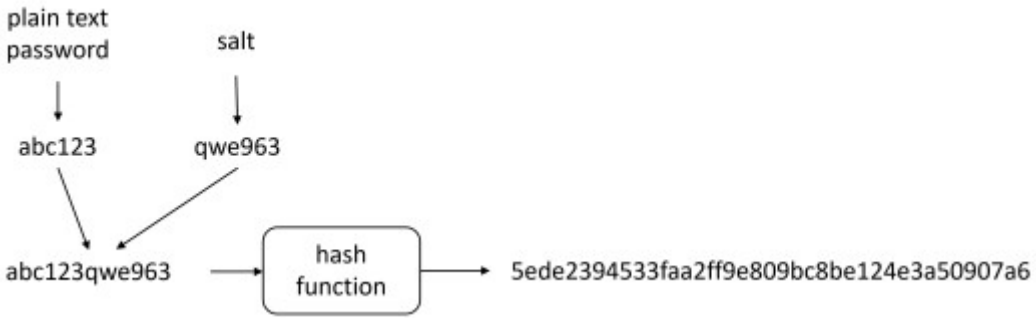


Fig. 24. The process of salted hashing.

Even though john and sam have the same password, their hash values are different (see Figure 25).

Interaktivní prvek

During the login process, the system retrieves the appropriate user's salt value from the database, appends it to the input password, passes it through the hashing function, and compares the resulting hash value to the hash value recorded in the table. The user is successfully authenticated if both hash values match.

user_name	salt_value	salted_hash_password
john	qwe963	5ede2394533faa2ff9e809bc8be124e3a50907a6
sam	hjk521	6367c48dd193d56ea7b0baad25b19455e529f5ee
alice	asd753	0772dbe339a885eb2ed73c1fe842d2ef6e9003a3

Fig. 25. An example of a table storing salted and hashed passwords.

The minimum protection of the stored passwords should include hashing and the use of salts. Additionally, **NIST** suggests locking a user out of the system if they use an incorrect password too many times (e.g., after three unsuccessful tries user cannot try again for one minute), allowing emojis, ASCII, and Unicode characters in passwords, and allowing copy-and-paste functions in the password fields to

make using password managers and multi-factor authentication more convenient.

Interaktivní prvek

CHAPTER 5

Passwordless Authentication

Given all the problems and weaknesses of passwords, the idea of authentication without passwords is not new. As the name implies, passwordless authentication allows a user to log in or acquire access without inputting a password or answering security questions. Passwordless authentication reduces the need for hazardous passwords and their administration while also improving the security of user accounts by reducing their vulnerability to assaults. There are different passwordless authentication mechanisms such as proximity badges, physical tokens, USB devices/ keys, magic links, biometric recognition, mobile applications, etc. Most of these techniques are commonly utilised in multi-factor authentication to increase security. However, some of these solutions can be employed as a first-factor authentication system.

These elements are typically divided into two categories:

- Examples of ownership elements are smartphones, **OTP** tokens, smart cards, or hardware tokens ("something the user has").
- Fingerprints, retinal scans, face or voice recognition, and other biometric identifiers are examples of inherence factors ("what the user is").

Passwordless authentication is often confused with ***Multi-factor Authentication*** (**MFA**) because both use a variety of authentication factors. However, while **MFA** is used as an additional layer of security on top of password-based authentication, passwordless authentication does not require a memorised secret and typically uses just one highly secure factor to authenticate identity, making it faster and easier for users.

Passwords are difficult to remember, and the requirements are becoming increasingly complex. Different sites may have different password policies, so a password generated for one site might not work on another. Remembering a password generated for a modern enhanced password policy is often challenging.

Like ***Fast Identity Online*** (**FIDO**), different standards come into play here. Although passwordless authentication and **FIDO** technology have existed for some time, online services and identity providers have yet to use them on a large scale. Passwordless authentication will become the future of authentication, thanks to the incorporation of biometric capabilities into most modern mobile devices and laptops.

ADVANTAGE

Passwordless authentication enhances the end-user experience by eliminating password fatigue. The user no longer needs to create a lengthier, more secure password and can receive unified access to all programs by simply connecting to a USB device or scanning their fingerprint.

5.1.1 Fast Identity Online

Fast Identity Online (**FIDO**) is a set of open-source authentication protocols established by the **FIDO** Alliance to do away with passwords. To implement safe authentication, **FIDO** protocols use basic public key cryptography algorithms. The private keys will never leave the security device, and all conversations will be encrypted.

Fig. 26. Example of **FIDO** based authentication.

The **FIDO** alliance has issued three sets of standards.

- **UAF** (***Universal Authentication Framework***): The passwordless authentication option is included in the **FIDO UAF** protocol. Users who use this protocol should sign a challenge provided by the **FIDO** server using one or more security factors available in their security/digital device.
- **U2F** (***Universal Second Factor***): The second-factor authentication option is provided by the **FIDO U2F** protocol. To establish their identity, users must supply two pieces of evidence. This has been renamed to CTAP1 with the launch of the FIDO2 protocol.
- FIDO2: The **FIDO** Alliance's newest set of specifications is known as FIDO2.

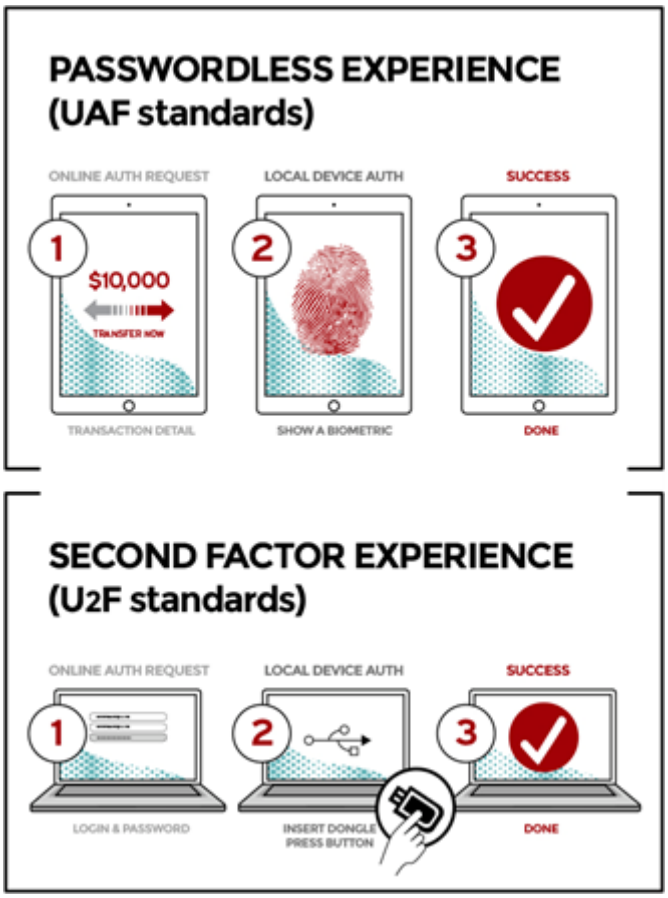


Fig. 27. The **UAF** and **U2F** (CTAP1) standards for passwordless authentication.

5.1.2 FIDO2 and Webauthn

The FIDO2 specification consists of:

- W3C Web Authentication (WebAuthn) standard and
- the **FIDO** Client to Authenticator Protocol 2 (CTAP2).

FIDO2 allows users to use ordinary devices to effortlessly authenticate to internet services in both mobile and desktop contexts. WebAuthn is a standard online API for **FIDO** authentication that is incorporated into platforms and browsers. CTAP2 is a CTAP version that allows users to use external and built-in authenticators to offer passwordless, two-factor, or multi-factor authentication. The WebAuthn API is a tool for creating and managing public key credentials. An overview of the FIDO2 authentication method is shown in Figure 28.

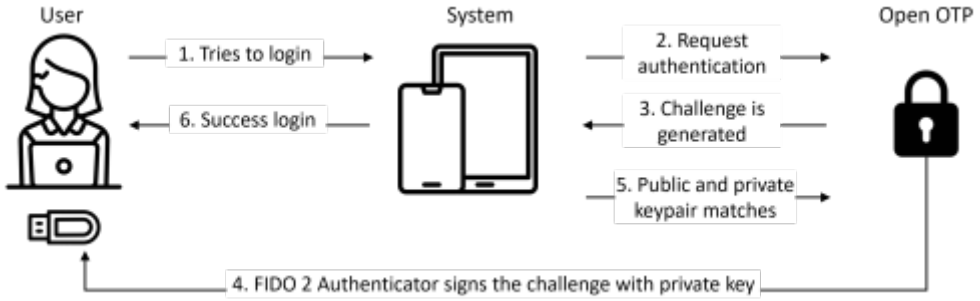


Fig. 28. The FIDO2 Authentication Method.

Interaktivní prvek

CHAPTER 6

An Introduction to Digital Signing

DEFINITION

A mathematical system for checking the validity of digital messages or documents is known as a digital signature.

A genuine digital signature gives a recipient a good reason to believe that the message was created by a known sender (authenticity) and that it was not altered in transit if the prerequisites are met (integrity).

The main goals a digital signature strives to achieve are:

- **Authentication**: Digital signatures are bound to a specific user via their private key. As a result, they can determine who owns the private key used to sign the original data/message (e.g. document, email, or file). See below for more on private and public keys.
- **Integrity**: A hashing technique is used in digital signatures to ensure that a message is not tampered with. See below for more about hashing.

So digital signatures are one of the ways to authenticate an entity, but we first need to clarify some concepts before it can be shown how a digital signature can be used in authentication.

6.1 Public-Key Cryptography

To understand digital signatures, we must first explain asymmetric cryptography, often referred to as public-key cryptography. In contrast to classical (symmetric) encryption, which uses only one key for encryption, asymmetric encryption employs a pair of keys. Encryption is the process of encoding information, as depicted in Figure 29.

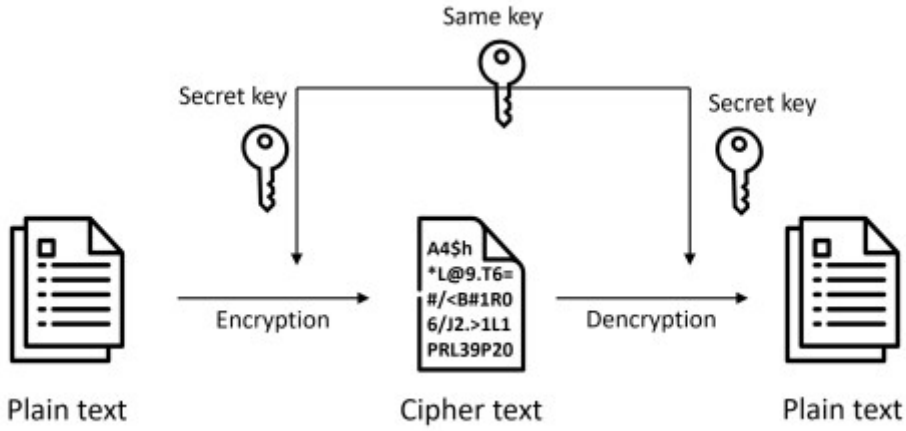


Fig. 29. Symmetric Encryption

Imagine you wish to send someone an encrypted message using classical encryption. In this scenario, both parties must agree on a single key. You can't transmit it to each other because then someone may see it, and they'll be able to see all of your messages.

DEFINITION

> On the other hand, asymmetrical encryption employs a pair of keys, a public and private key that belong together mathematically. Only the linked private key can decrypt what is encrypted with the public key.

Now, if someone wants others to send them encrypted messages, they just publish their public key for everybody to see. They simply use their private key to decrypt messages encrypted with their public key, as messages encrypted with their public key can only be decrypted with their private key. This is useful because we don't have to worry about sharing the public key in a secure manner.

In a nutshell, for two parties to communicate securely using asymmetric encryption, the process is as follows:

- The public keys are exchanged between the two parties.
- Person 1 encrypts the message they wish to send using person 2's public key and sends it to person 2.
- Person 2 decrypts the message with their private key.

This process is depicted in Figure 30.

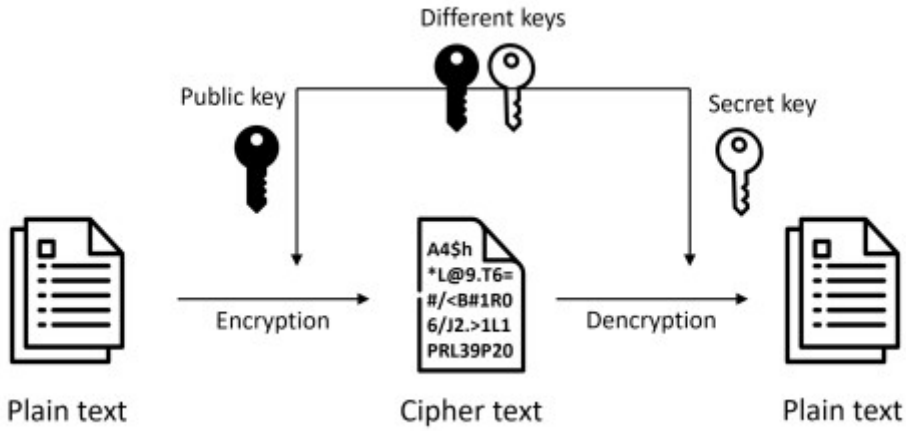


Fig. 30. The process of asymmetric (public-key) encryption.

DEFINITION

Digital signatures function by signing (encrypting) anything with a private key, which is then validated by the public key that is connected with it. So the keys in the key pair are used oppositely.

This is because the signer is the only person with access to the private key used to make the signature. Therefore, you can be certain it was this person who signed. Anybody can use the public key to verify (i.e. successfully decrypt the message) that the public key owner created the message.

Interaktivní prvek

6.2 The Process of Digital Signing

As already mentioned in digital signing, a key pair is used, consisting of public and private keys. Cryptographic key pairs are used to encrypt (locking) and decrypt (unlocking) source data in the same manner as physical keys are used for locking and unlocking. Private keys are kept secure and confidential because if someone learns another person's private key, they can sign source data as that person. On the other hand, public keys are supposed to be shared with everyone. The data encrypted by the private key can only be decrypted with the public key, revealing the original data.

DEFINITION

However, the process of digital signing includes asymmetric cryptography and hash functions.

These two building blocks are combined to form the actual signing process as follows:

1. Using a hashing method, the sender calculates the hash of the source content they want to deliver.
2. The sender encrypts the calculated hash with their private key to create a digital signature.
3. Then, the content and the digital signature can be sent to the recipient.
4. After the recipient receives the messages, the receiver uses the sender's publicly available public key to decrypt the sender's encrypted digital signature. If successful, the sender's identity as the private key owner used to encrypt the file is confirmed.
5. The receiver then obtains the original content from the received message and generates a hash of this content.
6. The content is validated as identical to what the sender supplied if the receiver's calculated hash matches the senders. If the hashes don't match, the content has been tampered with, and thus the signature is not valid.

A graphical representation of the process is given in Figure 31.

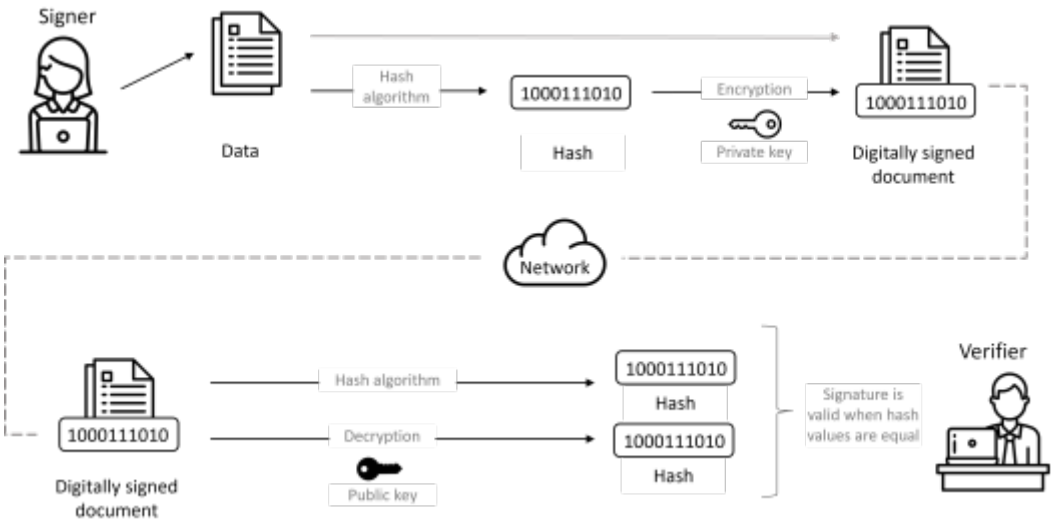


Fig. 31. A schematic representation of the creation and verification of a digital signature.

Because the sender's public key is publicly available, anyone can decrypt the encrypted content the sender sends. As a result, this encryption method only verifies integrity, not confidentiality.

One would ask why we generate a hashed value of the data before signing it? Simply because it makes the signature much smaller and the process of creation and verification of the digital signature faster since only the hashed values are compared rather to the entire data/document. Note that it works because hash algorithms always produce a value of a set length.

ADVANTAGE

As you may have guessed, digital signatures provide several advantages, including the following:

- they boost security and confidence because they can't be reverse-engineered or falsified;
- provide non-repudiation to the source data encryptor;
- ensure the integrity of the transmitted data.

DISADVANTAGE

However, there are also drawbacks to digital signatures, such as:

- the fact that there is no way to revoke signatures (source data trust) after they have been distributed, thereby making judgments irreversible;
- the usage of public keys renders secrecy impossible. Anyone with the public key can verify the signature.

Interaktivní prvek

Interaktivní prvek

CHAPTER 7

The Public Key Infrastructure

We have already learned about the concepts of digital signing and public-key cryptography. As it turns out, we need something more for the whole concept to function properly in real-life. We need something called the ***Public Key Infrastructure*** (**PKI**).

DEFINITION

> **PKI** is a set of technologies, processes and entities that allows for safe communication over insecure public networks.

For example, **PKI** is what adds the S to HTTPS, and if you're viewing this content in a web browser, you're presumably using one to ensure it's from a trusted source. PKIs enable regulated access to systems and resources, data protection, and transaction accountability by establishing the identity of individuals, devices, and services.

PKI is being utilised in a variety of applications, including allowing communication security in the internet of things (IoT) and digital document signing. **PKI**, which is based on asymmetric cryptography, is commonly used to set up secure electronic communications such as online shopping, banking, and emails and communications between users and the websites to which they connect using HTTPS. **PKI** enables strong authentication, data encryption, and digital signatures for people, services, and objects by providing digital identities. These security methods provide secure access to physical and digital resources, secure communication between people, services, and things, and the digital signing of papers, transactions, or other data.

7.1 Components of Public Key Infrastructure

PKI is made of the following components:

- a ***certificate authority*** (**CA**)
- a ***registration authority*** (**RA**)
- a ***validation authority*** (**VA**)
- digital certificates

And, of course, ***public-key cryptography*** (**PKC**).

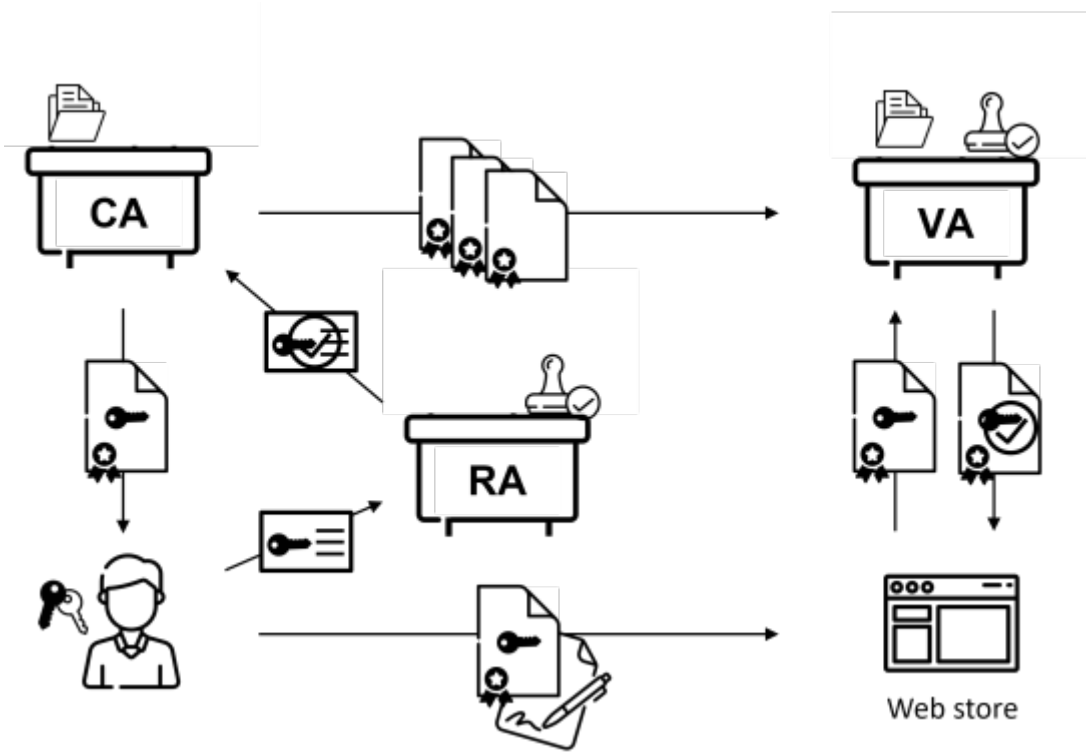


Fig. 32. Components of the Public Key Infrastructure (**PKI**).

Certificate Authority

DEFINITION

A certificate authority, often known as a certification authority (**CA**), is a company that creates and distributes digital certificates.

A digital certificate verifies that the named subject of the certificate owns a public key. Others (relying parties) can trust signatures and assertions about the private key that matches the certified public key. A **CA** serves as a trusted third party, trusted by both the certificate's subject (owner) and the party relying on the certificate.

The signing of certificates used in HTTPS, the secure surfing protocol, is one of the most prevalent uses for certificate authorities. Another popular application is for national governments to issue identity cards that may be used for digital signing or eGovernment.

Registration Authority

DEFINITION

In public key infrastructures, a Registration Authority (**RA**) is a function for certificate enrolment. It is in charge of receiving certificate signing requests from individuals, servers, things, and other applications, whether for initial enrolment or renewals. These requests are verified by the Registration Authority and forwarded to a Certificate Authority (**CA**).

A Registration Authority also handles certificate lifecycle management. Consider the case of revocation. The **RA** includes business logic to accept requests, including methods for verifying the requester's origin and the party who should have the certificate.

For accessibility and security concerns, a Registration Authority is normally separated from a Certificate Authority. The **RA** can be accessed through a user-friendly GUI or by APIs and standard protocols that are easy to integrate.

Validation Authority

PKI certificates are validated by a **PKI** Validation Authority (**VA**). Access to ***Certificate Revocation Lists*** (**CRL**), ***Online Certificate Status Protocol*** (**OCSP**), and **CA** chain certificate downloads are examples of certificate validation services. Because certificates can be issued and revoked, it is vital to verify the authenticity of a certificate before trusting it. The Validation Authority is tasked with resolving this issue.

The issuing Certificate Authority is responsible for providing certificate status updates to the Validation Authority per the established policy. You rely on each connected certificate authority to issue a list of revoked digital certificates when using CRLs (**CA**).

Digital Certificate

A digital certificate is a type of electronic identification for individual entities or organisations, similar to an ID. It includes information such as identification, a serial number, and expiration dates. We can also see the certificate authority's digital signature, which ensures the certificate's authenticity and the certificate holder's public key within the information. For example, **PKI** allows authenticated connections and, if combined with other cryptographic approaches, also secures connections between two communicating machines since the identities of the two parties can be confirmed using digital certificates. Almost all certificates issued nowadays comply with the X.509 standard.

Certificates come in many types:

- **Code signing certificates**: The code has been validated as coming from the developers and has not been modified, making the software trustworthy. It's used to sign software releases and validate software from the seller or developer to confirm that it's legitimate.

- **Email certificates:** The S/MIME protocol can be used to safeguard and validate emails, allowing the sender to establish authorship and prevent tampering.
- **Document signing certificates:** Adobe, Microsoft, and other software programs should be used to sign documents to ensure that they are unaltered and trustworthy. This certificate is nearly always used when you see a digital signature on a document.
- **TLS (HTTPS) certificates:** Used for secure HTTPS connections.

Fig. 33. Example of a digital certificate in Microsoft Windows.

Interaktivní prvek

7.2 The Hierarchical structure of the Public Key Infrastructure

A hierarchy of CAs that sign and issue digital certificates or credentials is common in **PKI**. Sub-CAs are given the power to sign digital certificates for devices by each **CA**. End devices communicate digital certificates at the bottom, which are allowed by the sub-**CA** above them, which generated and signed them. These are sometimes referred to as device certifications. Sub-CAs that create device certificates have their own certificate, which is authorised by the digital signature of the **CA** above them, and so on. **PKI** eventually reaches the root, which serves as the foundation for this particular **PKI** ecosystem domain.

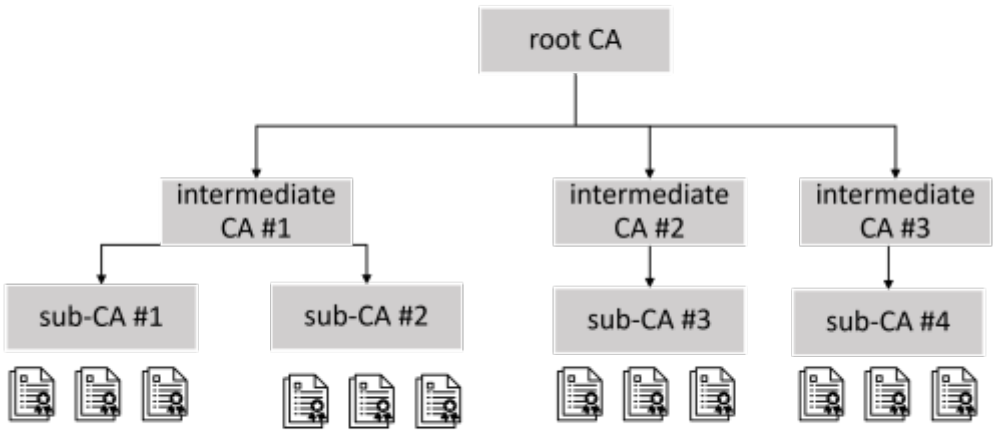


Figure 34. An example of a **PKI** hierarchy.

We can allow for specific revocation or access denial levels in the case of a leak or compromise of a private key in the ecosystem by placing **PKI** ecosystems in hierarchies.

Anyone can revoke the certificate of any **PKI** element, from the device to a high-level **CA**, depending on the nature of the security breach. However, that revocation of a certificate also revokes anything below the element in the hierarchy.

Furthermore, this demonstrates why **PKI** implementations are organised in tree-like hierarchies. This design allows the ecosystem owner to undertake selective damage control in the event of a breach. This is why issuing device certificates from the root **CA** is not a good idea because it limits flexibility. If something goes wrong in this instance, we may have to invalidate and revoke the entire **PKI** and all

deployed devices in the field. As a result, device certificates are virtually always issued by sub-CAs underneath the root certificate authority.

7.3 Digital Certificate Lifecycle

The lifecycle of a digital certificate starts with its creation and can be briefly explained as follows:

- **Certificate Enrolment:** The Certificate Authority (**CA**) receives a request for a certificate from an entity. A person, a device, or even a few lines of code might be considered an entity.
- **Certificate Issuance:** The **RA** must verify the applicant's identity, which is usually done via credentials or by relying on the identity of another **RA** who has already verified the applicant.
- **Certificate Validation:** The server checks with the **CA** every time the certificate is used to authenticate to ensure that it is still valid and hasn't expired or been revoked.
- **Certificate Revocation:** When certificates are first issued, they have an expiration date that is stated. When that date passes, the **CA** puts the certificate on the Certificate Revocation List (**CRL**), a form of a blacklist that tells the server not to trust certain certificates.
- **Certificate Renewal:** CAs can be configured to automatically renew certificates when they reach their expiration date, albeit they usually require re-verification of identity.

Interaktivní prvek

Certificate Authorities and Chain of Trust

The term "chain of trust" refers to the relationship between a digital certificate and a trusted **CA**. To be trusted, a certificate must be traceable back to the trusted root from whom it was issued, which means that all certificates in the chain — server, intermediate, and root — must be properly trusted.

In Figure 35, we can see that for google.com server GTS-**CA** 1C3 is a bottom-level **CA**. GTS-Root R1 is a mid-level **CA**. R1 top root **CA** for GlobalSign. In this method, a trust chain can be established.

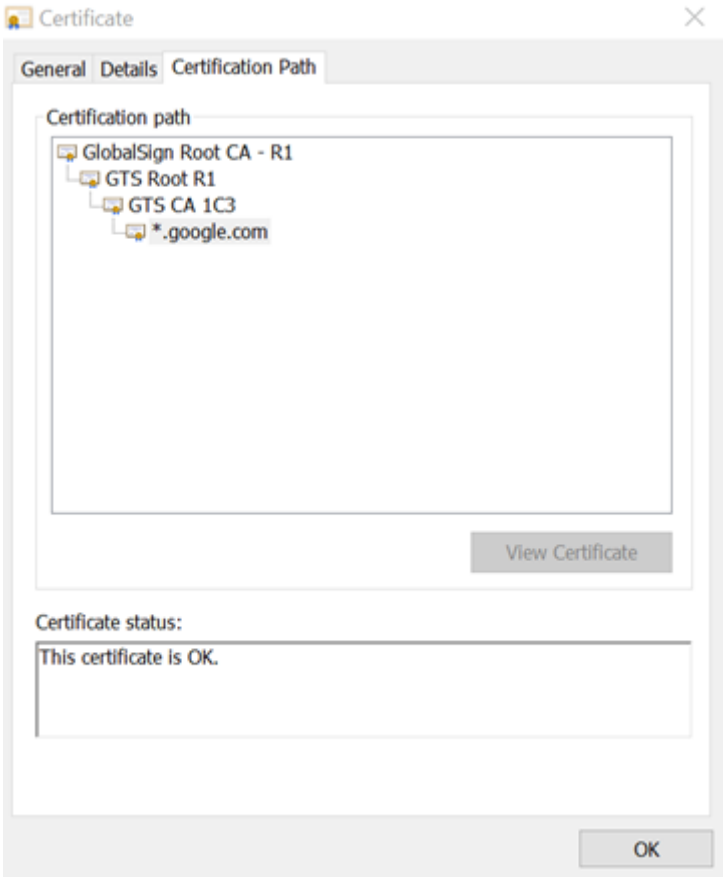


Fig. 35. Example of a Chain of Trust.

There are 3 parts to the chain of trust:

- A **root certificate** is a digital certificate that belongs to the Certificate Authority that issued it. For instance, most browsers come with it pre-installed, and it's saved in a "trusted storage." The Certificate Authorities keep a close eye on the root certificates. For example, GlobalSign Root CA- R1 is a root **CA**.
- **Intermediate certificates** are like branches on a tree, and the root certificate is like the tree trunk. They serve as a link between the protected root certificates and the publically issued server certificates. There will always be at least one intermediate certificate in a chain, but there may be more.
- **The server certificate** is the one that has been granted to a certain domain (in this case, www.google.com).

7.4 Authentication using PKI and PKC

In the digital world, public key infrastructure (**PKI**) is a system for authenticating people and devices. One or more trusted parties digitally sign documents verifying that a specific cryptographic key belongs to a specific user or device. The key can then be used as the user's identity in digital networks.

Digital certificates can also be used in **2FA** or in passwordless authentication.

When a user attempts to authenticate their identity to a server, the server creates random data and sends it to the user. The user then encrypts the data using their private key and returns it back to the server. The server decrypts the data using the user's digital certificate's public key, and if the decrypted data matches the received data, the server knows the user is who they say they are. This is the basic process of using PKI+**PKC** for authentication purposes.

CHAPTER 8

Test

The most secure way to store passwords is:

- ◯ encryption
- ◯ hashing
- ◯ plaintext
- ◯ slated hashing

Salting passwords makes it harder for an attacker to attack, as it makes the dictionary attack specific to each:

- ◯ user
- ◯ attacker
- ◯ device
- ◯ password

What is the most prominent disadvantage of the "what you are" characteristic?

- ◯ it is non-reputable
- ◯ it can be lost
- ◯ is can be forgotten
- ◯ in can be stolen

What is multi-factor authentication?

- ◯ Authentication that uses at least two different factors for authentication.

- ◯ Authentication that uses exactly two different factors for authentication
- ◯ Authentication that uses exactly one different factors for authentication
- ◯ It is the same as 2-factor authentication (2FA)

In a challenge-response authentication method who is presented with the challenge?

- ◯ the user
- ◯ the server
- ◯ the program
- ◯ the authenticator

How long should passwords be according to current guidelines and best practices?

- ◯ 6 characters
- ◯ 4 characters
- ◯ 7 characters
- ◯ at least 12 characters

What does a hash functions do?

- ◯ makes a hashtag
- ◯ calculate a unique data identifier
- ◯ creates passwords
- ◯ prevent authentication

Which is the standard for passwordless authentication?

- ◯ FIDO2
- ◯ FIBA
- ◯ UFI

- ◯ UPA

In what sequence does the process of digital signing use keys?

- ◯ the senders' private key and the recipient's public key
- ◯ the sender's private key and the sender's public key
- ◯ the recipient's private key and the recipient's public key
- ◯ the sender's public key and the recipient's private key

If public key cryptography is used for encryption purposes, which key is used to encrypt the data?

- ◯ the recipient's public key
- ◯ the sender's public key
- ◯ the recipient's private key
- ◯ the sender's private key

What is the sequence of keys used in asymmetric encryption?

- ◯ the senders' private key and the recipient's public key
- ◯ the sender's private key and the sender's public key
- ◯ the recipient's private key and the recipient's public key
- ◯ the sender's public key and the recipient's private key

What is the correct sequence of steps in the process of digital signing?

- ◯ hash, sign and send
- ◯ sign, hash and send
- ◯ encrypt, hash and send
- ◯ hash, code and send

Which are the components of PKI (public-key infrastructure)?

- ◯ CA, MA, LA, digital signature
- ◯ CA, RA, PA, digital signature
- ◯ CA, RA, PKC, digital certificate
- ◯ CA, RA, PKC, digital signature

What is the main task of a CA?

- ◯ issuing certificates
- ◯ issuing digital signatures
- ◯ checking digital signatures
- ◯ verifying an individuals' identity

How is PKI implemented?

- ◯ as a tree structure
- ◯ as a client-server system
- ◯ in hardware
- ◯ sequentially

How can digital certificates be used for authentication?

- ◯ they serve as the main authentication factor
- ◯ they serve as a 2nd authentication factor
- ◯ they can't be
- ◯ to sign documents

What is the aim of authentication?

- ☐ to check one's identity
- ☐ to identify someone
- ☐ to check what someone can access
- ☐ is an essential part of cybersecurity

What are the most common authentication methods?

- ☐ username and password
- ☐ face scan
- ☐ email and password
- ☐ RSA SecureID

What are the main authentication categories?

- ☐ how you look
- ☐ what you know
- ☐ who you are
- ☐ what you have

Examples of authentication using the "what you have" principle include:

- ☐ a smartphone
- ☐ a password
- ☐ a fingerprint
- ☐ a smart card

The basic authentication process includes:

- ☐ a server
- ☐ authentication data

- [ ] a user
- [ ] a keyword

The parts of an authentication method are:

- [ ] an input
- [ ] a verifier
- [ ] a computer
- [ ] a transportation system

Which elements can be used for 2-factor authentication?

- [ ] a SMS message
- [ ] an identification token
- [ ] a smartphone app
- [ ] a username

Current industry standards for secure password storage include:

- [ ] encryption
- [ ] hashing
- [ ] storing in plaintext
- [ ] salted hashing

Password salting technique makes what kind of attack more hard?

- [ ] dictionary attacks
- [ ] brut-force attacks
- [ ] server attacks
- [ ] user device attacks

What are vulnerabilities of the "what you know" authentication technique?

- ☐ it can be forgotten
- ☐ it can be lost
- ☐ it can be duplicated
- ☐ it can be denied

What are vulnerabilities of the "what you have" authentication technique?

- ☐ it can be forgotten
- ☐ it can be lost
- ☐ it can be duplicated
- ☐ it can be denied

What are examples of password-authentication?

- ☐ one-time passwords
- ☐ reusable passwords
- ☐ structured passwords
- ☐ credentials

What are non-electronic attacks on passwords?

- ☐ shoulder surfing
- ☐ social engineering
- ☐ dictionary attack
- ☐ brute-force attack

What are electronic attacks on passwords?

- ☐ phishing
- ☐ social engineering
- ☐ dictionary attack
- ☐ brute-force attack

Which of the following are password cracking tools?

- ☐ John the cracker
- ☐ John the ripper
- ☐ Hydra
- ☐ Hybrid

What should passwords not include?

- ☐ different type of characters
- ☐ words associated with you
- ☐ birth dates
- ☐ special characters

What functionalities do password manager typically include?

- ☐ auto-complete
- ☐ password generation
- ☐ password evaluation
- ☐ password deprecation

What are the disadvantages of 2FA?

- ☐ inconvenience
- ☐ higher security

- ☐ Concerns about privacy
- ☐ stronger authentication

What are the main goals a digital signatures?

- ☐ assuring authentication
- ☐ assuring integrity
- ☐ assuring confidentiality
- ☐ assuring authorization

In asymmetric encryption which kind of keys are used?

- ☐ a public key
- ☐ a secret key
- ☐ a privacy key
- ☐ a private key

What building blocks are included in the process of digital signing?

- ☐ hash functions
- ☐ symmetric encryption algorithms
- ☐ key exchange algorithms
- ☐ asymmetric encryption algorithms

What is the correct sequence of steps in the process of the verification of a digital signature?

- ☐ get hash from signature, get hash from data, compare
- ☐ get hash from data, get hash from signature, compare
- ☐ compare, get hash from data, get hash from signature
- ☐ compare, get hash from signature, get hash from data

Which are the components of PKI (public-key infrastructure)?

- ☐ CA
- ☐ PA
- ☐ digital signature
- ☐ digital certificate

What are typical component of digital certificates?

- ☐ expiration date
- ☐ issuer
- ☐ length
- ☐ digital signature

What are the main parts of the chain of trust in PKI?

- ☐ A root certificate
- ☐ an intermediate certificate
- ☐ a digital signature
- ☐ an administration certificate