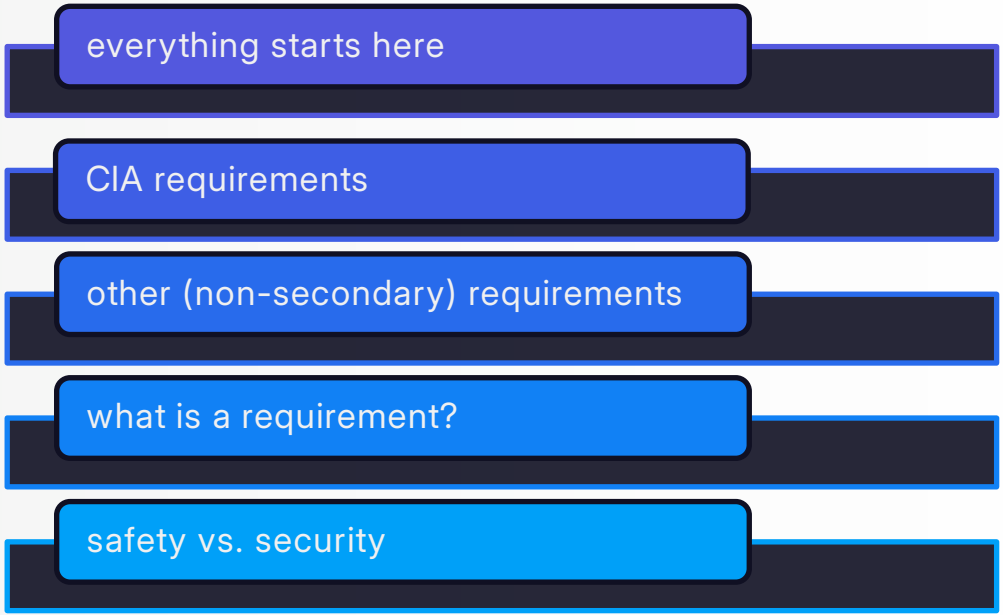
# Information security

# Information security & its requirements

# Information security



# Requirements

| | |
|---|---|
| Confidentiality | Integrity |
| Availability | Authentication |
| Non-repudiation | Accounting |

**Standard in ISO 27000**

# Cryptography vs. security

Cryptography and security differ

Cryptography traditionally deals with secrecy of information

Most real security deals with problems of fraud:

- Message modifications
- User authentication

Much of security has little to do with encryption however it might use cryptography

Almost invariably, encryption does not live alone without some form of *authentication*

# Pre-computer cryptography

an overview on old cryptography

old = before computers exist

# Scytale cipher

- Sparta, 700 BCE
- a strip of parchment was wrapped around a wooden rod, and the message was written lengthwise
- to decrypt, recipient needed a rod of **same diameter**


7

# The Egyptian pharaoh approach

**The sender**

1. Shaving a slave's head
2. Carving the secret message into the skull
3. Waiting for the hair to grow back
4. Sending the slave to the recipient

**The recipient**

a) Knows he must shave slave's head
b) After that, he could access the message

Timescales not comparable to today's


8

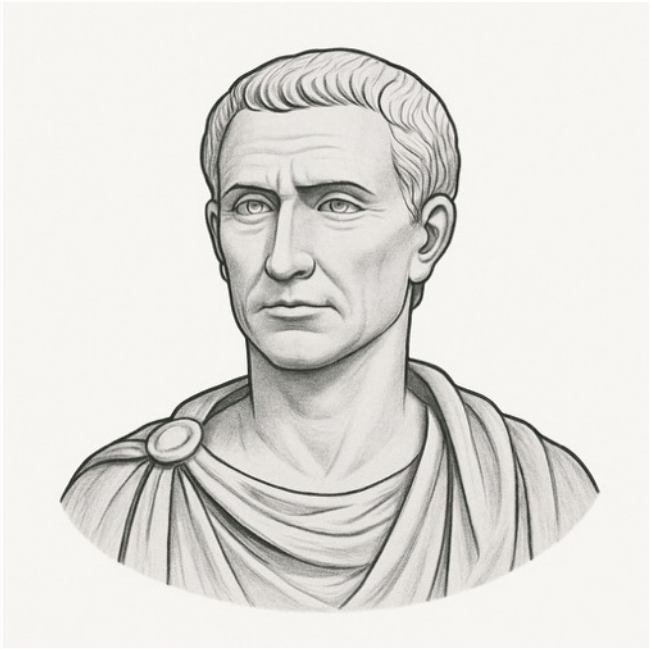
# Caesar cipher

**Shift the alphabet by a predetermined number of positions (e.g. 3)**

| A | B | C | D | E | ... | W | X | Y | Z |
|---|---|---|---|---|---|---|---|---|---|
| C | D | E | F | G | ... | Z | A | B | C |

key = 3

attack: it is sufficient to test all possible keys (26)





9

# Alphabetic substitution

| A | B | C | D | E | … | W | X | Y | Z |
|---|---|---|---|---|---|---|---|---|---|
| D | U | Z | O | L | … | Y | B | A | R |

random permutation (26! = 403 291 461 126 605 635 584 000 000)

attack: use a computer for a frequency attack

# Polyalphabetic

if long enough frequency analysis still effective

More permutations ($l = 3$)

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| A | B | C | D | E | … | W | X | Y | Z | 0 |
| D | U | Z | O | L | … | Y | B | A | R | 1 |
| B | P | Y | Q | M | | T | Z | S | N | 2 |
| W | P | Q | L | A | | S | L | F | G | 3 |

Replace symbol of text by using permutation $i + 1$, where $i$ starts by 0 and increases mod $l$

# Enigma

- one of the most advanced ciphers among the ancients (WWII)
- it encrypted each letter by passing it through a series of rotating electrical rotors, producing a different substitution with every keystroke
- broken by allied at Bletchley Park

- easily broken by a *computer* because of the limited keyspace $\approx 10^{23}$ to $10^{26}$
- see https://cryptii.com/pipes/enigma-machine

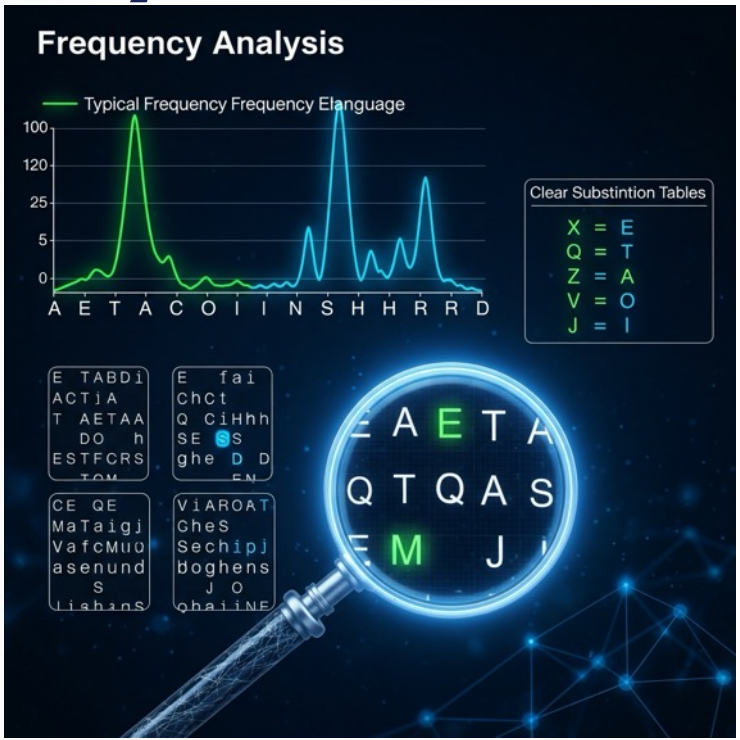
# Conclusion

- Many other examples are possible
- In all examples sender and recipient must share some information
- We call this shared information the **Key** (still today)

# Frequency analysis

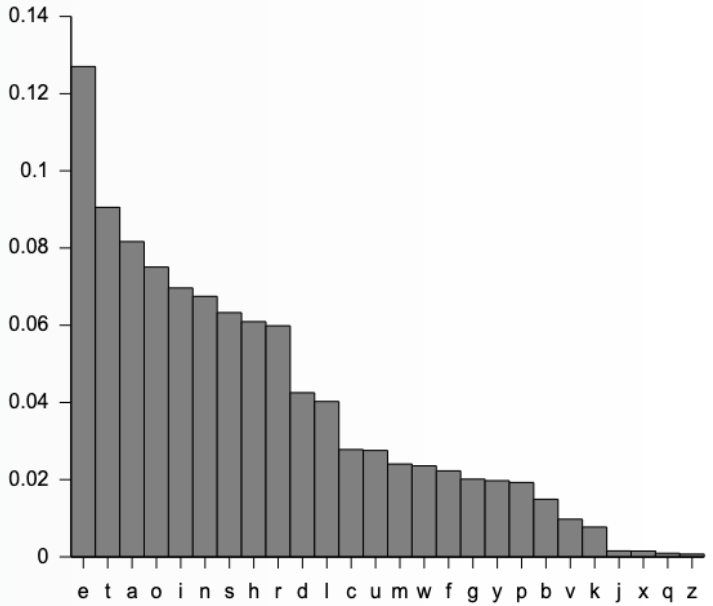
cryptoanalysis by frequency analysis

uses computers



# Frequency of letters

- Each language has its own distribution of frequencies of the letters used
- they change from language to language but are public and well-known (Wikipedia)




English

# Languages

## How to obtain frequencies

- They are typically public and available on the web
- Several online resources exist
- How to do it yourself
  - Take a large sample of text
  - Remove spaces, punctuation, and special symbols
  - Sort the remaining sequence of letters
  - Perform a simple block-based count

# Frequency analysis (simplified)

1. Clean encrypted text
2. Normalize casing (all lowercase or uppercase)
3. Compute basic frequencies
4. Count the frequency of each individual character

# Frequency analysis (simplified) (2)

5. Use known frequency profiles from major languages

| Language | Most frequent letters |
|---|---|
| English | E, T, A, O, N, I |
| French | E, A, S, I, T |
| Spanish | E, A, O, S, N |
| German | E, N, I, S, R, A |
| Italian | E, A, I, O, N |

6. Try aligning the most frequent letters of the encrypted text to each of these and see which language yields the most plausible results

# Frequency analysis (simplified) (3)

Doesn't work well on

- short messages
- polyalphabetic ciphers
- modern encryption

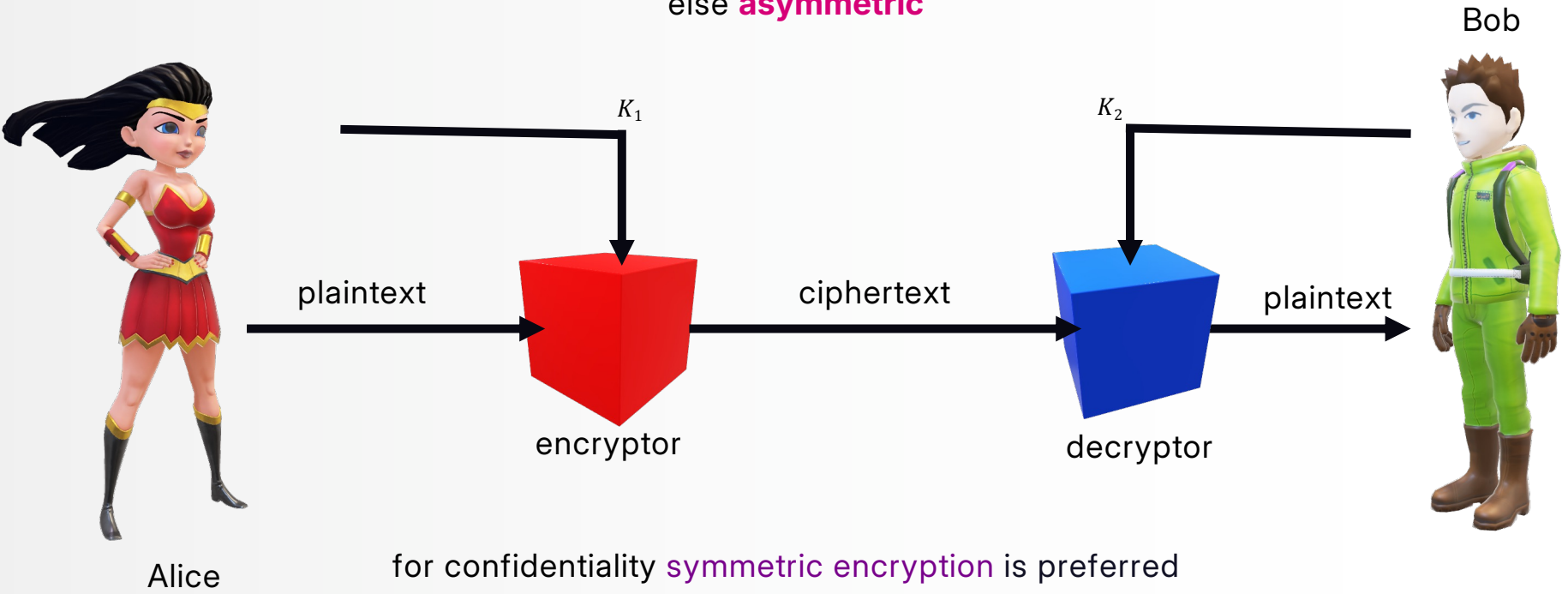
5. Look for language-specific patterns
    - common letter pairs (like TH, QU, CH, LL) or letter positions (e.g., Q followed by U is common in many languages)
    - common word lengths (e.g., 2-letter words in English: *it, is, to*)
    - repeated short patterns like *la, le, de* (Romance languages), or endings like *-en, -er* (Germanic languages)

# Model of encryption

used and accepted in all environments

# Model

if keys $K_1 = K_2$ **symmetric encryption**
else **asymmetric**




for confidentiality symmetric encryption is preferred

# What is a key?

1. (Long) string of (random) bits
2. All strings should have same probability

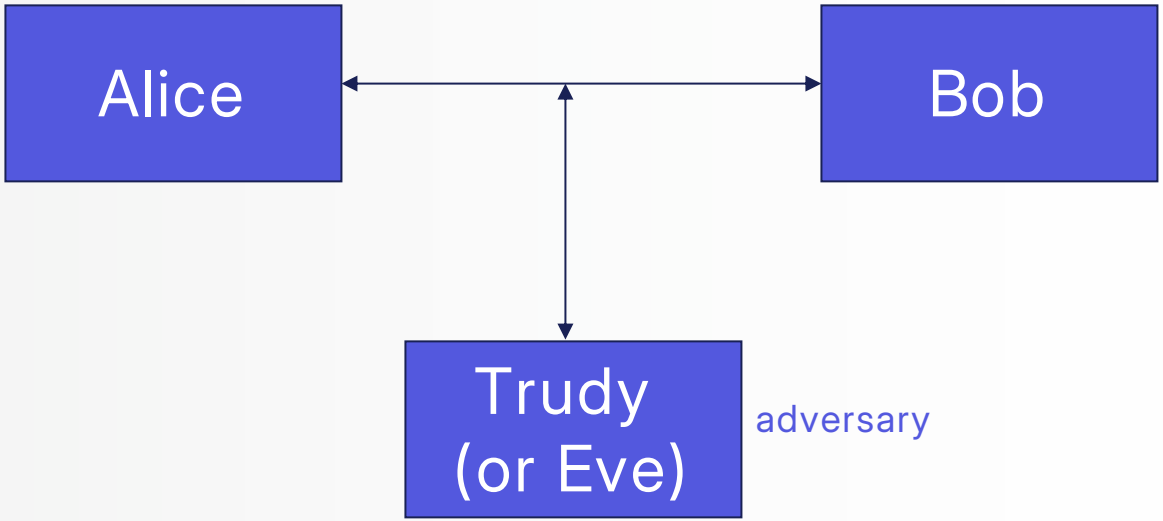

01010001110101110101010110101O...

symmetric keys must remain confidential

# Communication model


1. Two parties – Alice and Bob
2. Reliable communication line
3. Shared encryption scheme: $E$, $D$, $K_1$, $K_2$
4. Goal: send a message $M$ confidentially

# Threat (attack) model




attack to confidentiality: get partial/total information on *M*

# Adversary

## Passive

reads the exchanged messages

(no change)

## Active

can modify messages sent by Alice or Bob

can send false (fake) messages claiming that they have been sent by someone else

# Terminology

| term | meaning |
|---|---|
| **plaintext** | *information that will be encrypted* |
| **ciphertext** | *information that has been encrypted, i.e. transformed into incomprehensible text* |
| **key** | *sequence of fixed length of bits that appear random; in the symmetric case* $K_1 = K_2$ |
| **cipher** | *pair of algorithms for encryption and decryption, often denoted as (E, D)* |
| **encryptor** | *entity that applies a cipher (algorithm E), producing a ciphertext* |
| **decryptor** | *entity that applies a cipher (algorithm D), producing a plaintext* |
| **encryption** | *the operation performed by an encryptor* |
| **decryption** | *the operation performed by a decryptor* |
| **adversary** | *entity that attempts to compromise confidentiality, integrity, or availability of information systems* |

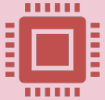
# Security goals

**Kerckhoff's principle**: a cryptosystem should be designed to be secure, even if all its details, except for the key, are publicly known

**Shannon's maxim**: the enemy knows the system

If the keys are unknown, then

- it is **hard** to obtain even partial information on the message
- it is **hard** to find the key even if we know clear text



**hard** = computationally hard: it takes long time even if the most powerful computers are available

# Adversarial model

Trudy attempts to discover information about $M$

Trudy knows the algorithms $E$, $D$

Trudy knows the message space

Trudy has at least partial information about ciphertext

Trudy does not know $K_1$, $K_2$

# Attack categories

against confidentiality and other requirements of information security

# What is in attack?

- Any **intentional** attempt to compromise the security of a computer system, network, or data by violating one or more core principles of information security, such as confidentiality, integrity, or availability
- Some attacks do not directly target information security principles but serve indirect purposes, such as enabling future compromise or gathering intelligence
- Remember, cryptography plays a fundamental role among the measures designed to uphold information security

# First attacks

| type | acronym | description |
|---|---|---|
| Eavesdropping | — | secretly listening to private conversation of others without their consent |
| Ciphertext-only attack | COA | attacker has only access to ciphertexts and tries to deduce the plaintext or key |
| Known-plaintext attack | KPA | attacker knows pairs of plaintext and ciphertext and uses them to infer the key |
| Chosen-plaintext attack | CPA | attacker can encrypt arbitrary plaintexts to gather information about the cipher |
| Chosen-ciphertext Attack | CCA | attacker can decrypt arbitrary ciphertexts of their choice, except for the target ciphertext |

# Attacks may use an oracle

- An **oracle** is a theoretical black-box function that an attacker can query to obtain outputs (e.g., ciphertexts or plaintexts), often under controlled conditions

- It is an abstract entity or mechanism that provides access to a cryptographic function without revealing its internal workings

# Selection of attacks

- **Known plaintext**: attacker has samples of both plaintext and its encrypted version (ciphertext) and is at liberty to make use of them to reveal further secret information such as secret keys

- **Chosen plaintext**: attacker has the capability to choose arbitrary plaintexts to be encrypted and obtain the corresponding ciphertexts. The goal of the attack is to gain some further information which reduces the security of the encryption scheme. In the worst case, a chosen-plaintext attack could reveal the scheme's secret key

uses an oracle

# Selection (continued)

- **Chosen ciphertext**: the cryptanalyst gathers information, at least in part, by choosing a ciphertext and obtaining its decryption under an unknown key
- **Adaptive chosen plaintext**: the cryptanalyst makes a series of interactive queries, choosing subsequent plaintexts based on the information from the previous encryptions

**both use an oracle**

# Other attacks

| type | description |
| --- | --- |
| Cryptoanalysis | Any mathematical technique used to break or weaken cryptographic algorithms |
| Brute-force attack | Attacker tries all possible keys until the correct one is found |
| Side-channel attack | Exploits physical data (timing, power consumption, EM radiation) from a device |
| Replay attack | Reuses previously captured valid messages to trick a system |
| Man-in-the-middle attack (MITM) | Attacker intercepts and potentially alters communications between two parties |

more attacks discussed next

# About brute-force

- Modern computers can quickly try exploring all possible keys (~ $10^6$ to $10^{15}$ keys/second)
  - depending on hardware
- Keys often have a fixed size: the longer the key, the stronger the security — but the slower the performance
- If a key is $n$ bits long, there are $2^n$ possible keys. The practical limit for current computers is often set around $n = 80$, but this threshold continues to rise. With 128-bit keys, even at a rate of 1 trillion keys per second, it would take over $10^{19}$ years on average to brute-force the key

# Keys and password

similar but different

# Why to make a distinction?

## Keys and passwords

- They have similarities but are different
- Both are secret
- Cryptography uses both (for now)
- They have different characteristics
- They are used in different contexts
- They are protected differently
- They have a different impact on security

# Password basics

- A password is a human-chosen secret used to authenticate a user
- It must be
  - Memorable
  - Short enough to type
  - Representable with keyboard characters

# Limitations of passwords

- Typable = predictable: constrained to the keyboard → lower entropy
- Password space = $T^n$, not $N^n$ ($T$: keyboard-typable symbols, $N>T$: all possible symbols)
- Vulnerable to
  - Brute force
  - Reuse across platforms
  - Other attacks

# Phishing attacks

- Phishing is a type of **social engineering** attack in which an adversary tricks users into revealing sensitive information, such as passwords, credit card numbers, or access credentials
- Characteristics
  - Disguised as legitimate emails, messages, or websites
  - Often creates urgency (e.g., “Your account will be locked!”)
  - Can lead to credential theft, financial loss, or malware installation
- Email, SMS, voice calls, fake login pages etc.

other vulnerabilty of passwords

# What is a cryptographic key?

- A cryptographic key is a **random bit string** used in encryption, decryption, and other cryptographic operations
- If the string is *n* bits long, then there are $2^n$ possible keys

01010110000111101…10101011

561D…AB (often represented in hexadecimal)

# Key properties

- Machine-generated
- Not typable
- Fixed-length (e.g., 128, 256 bits)
- Stored as binary, not text

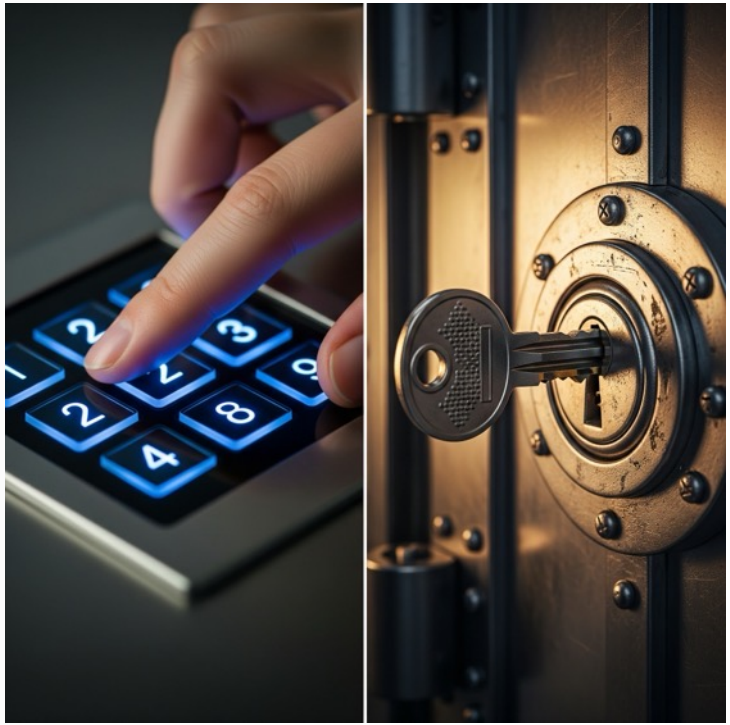
**Visual Comparison**
Password: S3cr3t!
Key: 9F4B713F8E2A... (256 bits)

# Password vs Key – Comparison Table

| feature | password | cryptographic key |
|---|---|---|
| chosen by | human | machine |
| typable | yes – must fit a keyboard | no – binary data |
| length | short (8–20 chars typically) | long (128–256 bits or more) |
| purpose | authentication | encryption / signing |
| stored as | hashed (ideally) | raw / protected |

# Analogy

- password = PIN typed into a keypad
- key = the actual metal key unlocking a vault



# Combined use

- Many systems use **passwords to unlock secret keys**
- Example
  - PGP, Signal, WhatsApp: your password decrypts a stored secret key

# Key Derivation

- Passwords are often converted into keys using "ad hoc" algorithms
  - Argon2, PBKDF2

- Still constrained by password entropy

# Why keys are more secure

- Higher entropy
- Machine-random
- Not guessable or typable

# Why passwords are still used

- Usable by humans
- Don’t require special storage
- Usability vs security tradeoff

# Summary

- **Passwords**: chosen by humans, typable, used for identity
- **Keys**: generated by machines, not typable, used for encryption
- **They’re not interchangeable** — both must be protected appropriately