# Symmetric Cryptography

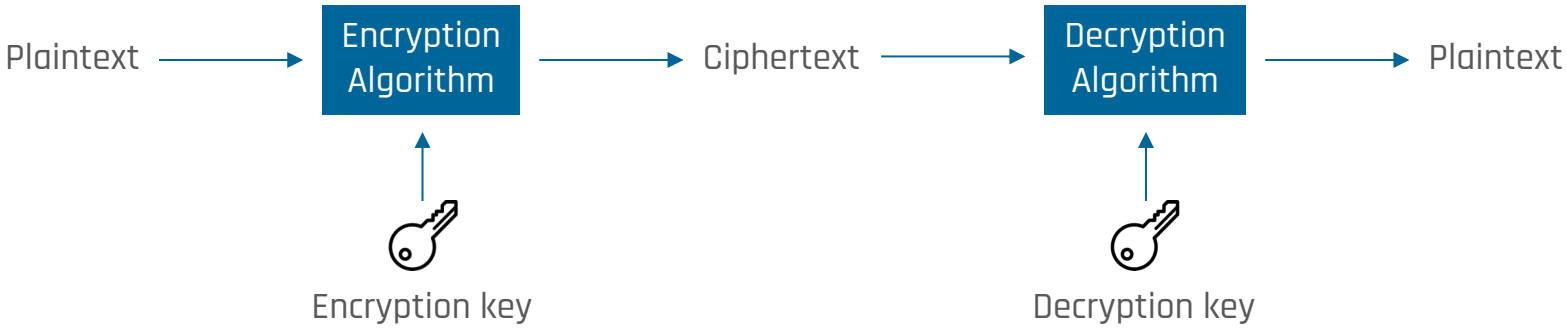
## Introduction to Security (192.019)

Mauro **Tempesta**

**Security** & **Privacy** Research Unit (192-06)
https://**secpriv.wien**

# Cryptography – Classical Definition

Cryptography is the science of techniques for encrypting and decrypting data



# Cryptography – Modern Definition

> Cryptography is the practice and study of techniques for **secure communication** in the presence of **adversarial behavior**
> — Ronald L. Rivest

Encryption algorithms

Hash functions

Message authentication codes

Digital signatures

# Symmetric vs. asymmetric cryptography

## Symmetric cryptography

- Cryptographic operations use the same key (e.g., encryption and decryption)
- The key must remain secret and communicated in a secure way to the partner
- Every pair of communication partners needs a different key
- Very good performance

## Asymmetric cryptography

- Cryptographic operations use different keys (e.g., one key for encryption, one for decryption)
- Only one of the keys remains secret, the other one is publicly available
- Each person needs a keypair and the public keys of the communication partners
- Rather slow

# Classic Cryptography

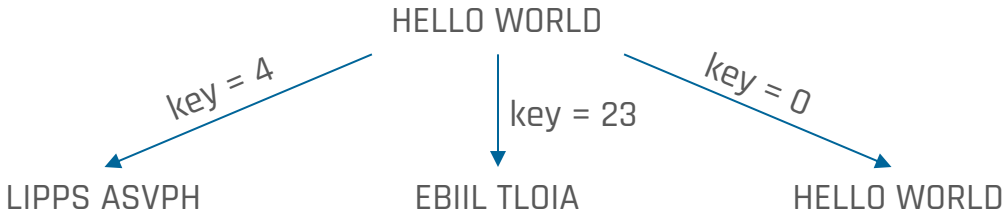
# Types of Classical Ciphers

- **Substitution ciphers**
  - Symbols in the plaintext are substituted with other (typically different) symbols
    - Simple vs. polygraphic: operates on one or multiple symbols
    - Monoalphabetic vs. polyalphabetic: one or multiple possible substitutions depending on the position of the symbols in the plaintext
- **Transposition ciphers**
  - Plaintext symbols are preserved, but their order is permuted

- Classical ciphers alone are nowadays insecure
  - Still, substitution and transposition are the fundamental transformations employed in modern symmetric ciphers

# Shift Cipher (or Caesar Cipher)

- Reportedly used by Julius Caesar for his military correspondence
- Each letter of the plaintext is replaced by a letter some fixed number of positions further down the alphabet
  - After the last letter, counting starts again from the beginning of the alphabet
  - The fixed number of positions is the cryptographic key of the scheme



# Security Analysis of the Shift Cipher

- The **key space** (set of possible keys) size is alphabet length – 1
  - Only 25 keys for the Latin alphabet
  - An attacker can just try all possible keys to recover the plaintext (**brute-force attack**)

Brute-forcing LIPPS ASVPH:
Key 1 -> KHOOR ZRUOG
Key 2 -> JGNNQ YQTNF
Key 3 -> IFMMP XPSME
Key 4 -> HELLO WORLD
...

# Simple Monoalphabetic Substitution

- Each plaintext symbol is replaced with a (possibly) different symbol in the ciphertext
  - The key defines the **mapping** from plaintext to ciphertext symbols
  - Possible to use a different alphabet in the ciphertext

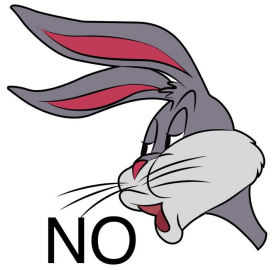
| | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Key | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z |
| | Q | M | O | Z | C | L | A | R | E | K | U | N | V | B | H | Y | I | F | G | X | P | S | D | W | J | T |

| | |
|---|---|
| Plaintext | SUBSTITUTION |
| Ciphertext | GPMGXEXPXEHB |

# Security Analysis of the Monoalphabetic Substitution

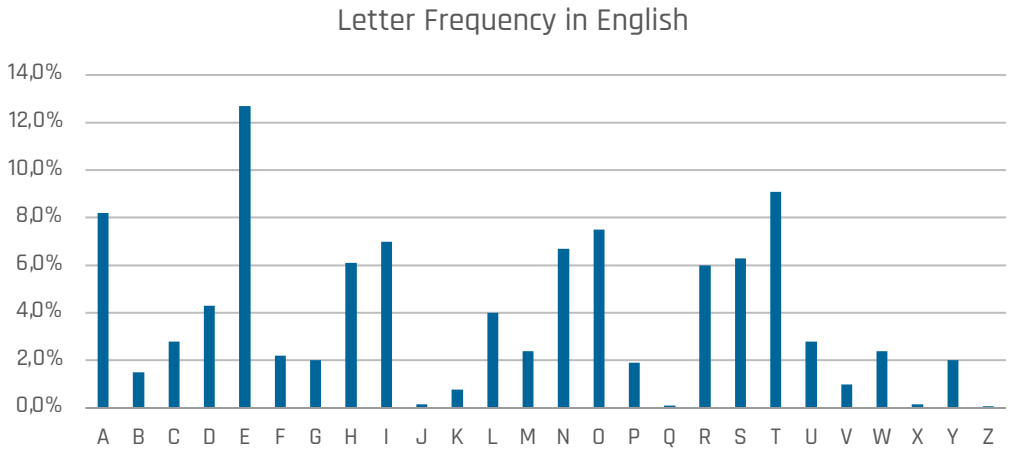
- A brute-force attack is in principle always possible…
  - The number of possible keys is the factorial of the alphabet size
  - Latin alphabet: $26! \approx 4{,}032 \cdot 10^{26}$ keys
  - If we can try one billion keys per second, a brute-force would require $1{,}278 \cdot 10^{10}$ years!

**Does this mean that this encryption algorithm is secure?**



# Statistical Cryptanalysis of the Monoalphabetic Substitution

- Letters appear with **different frequencies** in natural languages
- The monoalphabetic substitution **does not hide** the frequency distribution of the letters in the plaintext



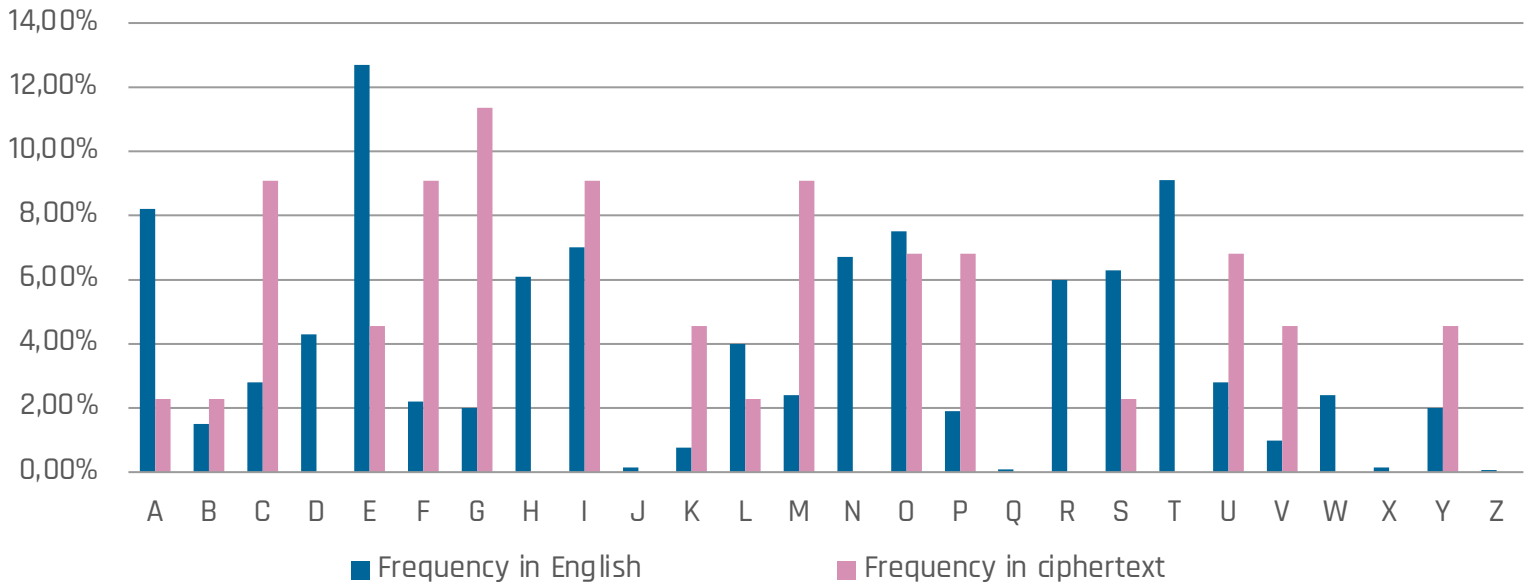
# Statistical Cryptanalysis of the Monoalphabetic Substitution

- Analysis of letter frequency distribution can be used to decrypt **sufficiently long** ciphertexts written in a known language
  - Frequency distribution in short texts may not be particularly meaningful
  - Trial-and-error/guessing can also lead to good results!

- Possible strategies if whitespaces are preserved in the ciphertext:
  - Identify **recurrent patterns** and substitute them with words occurring often in the target language (prepositions, articles, …)
    - A single-letter word in English is likely to be the article “A”
    - A three-letters word could correspond either to “THE” or “NOT”

# Example - Cracking a Ciphertext

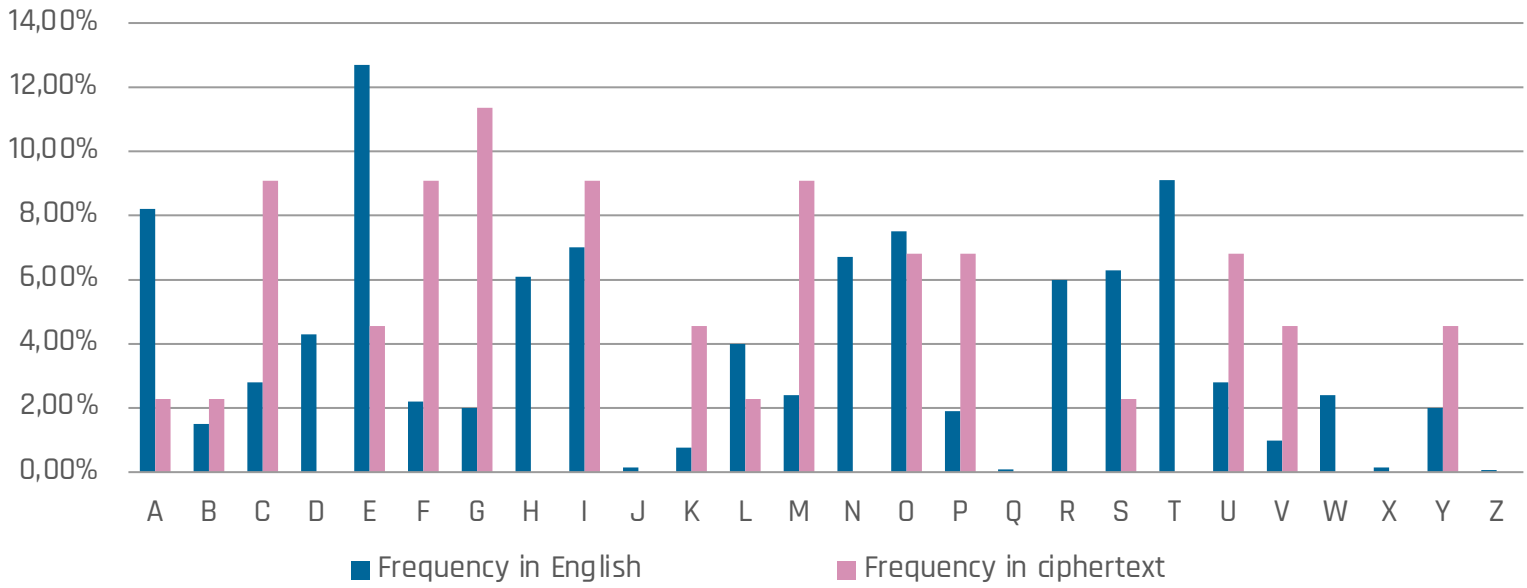
Ciphertext `MUG PVFPMIMVMIKC YISUGN YOC FG GOPIBE FNKAGC FE UOCL`



# Example - Cracking a Ciphertext

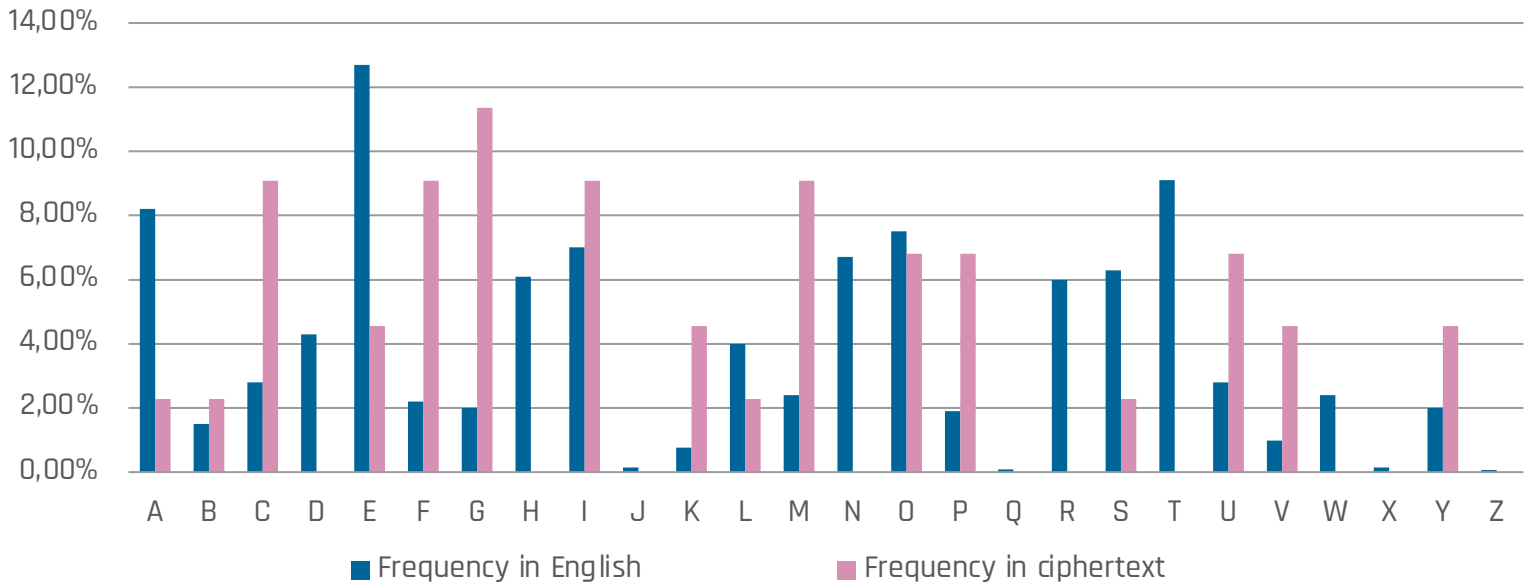
Ciphertext MUG PVFPMIMVMIKC YISUGN YOC FG GOPIBE FNKAGC FE UOCL

Plaintext T E T T T E E E E E



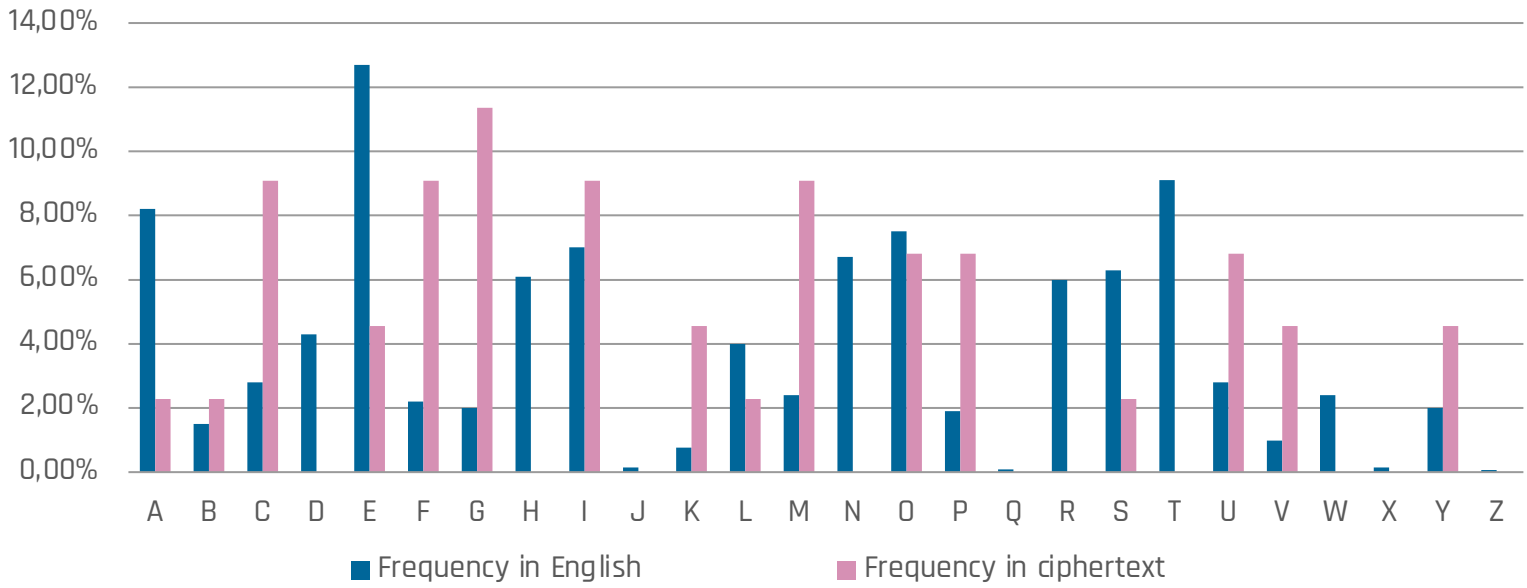
# Example - Cracking a Ciphertext

| | |
|---|---|
| Ciphertext | MUG PVFPMIMVMIKC YISUGN YOC FG GOPIBE FNKAGC FE UOCL |
| Plaintext | THE T T T HE E E E H |



# Example - Cracking a Ciphertext

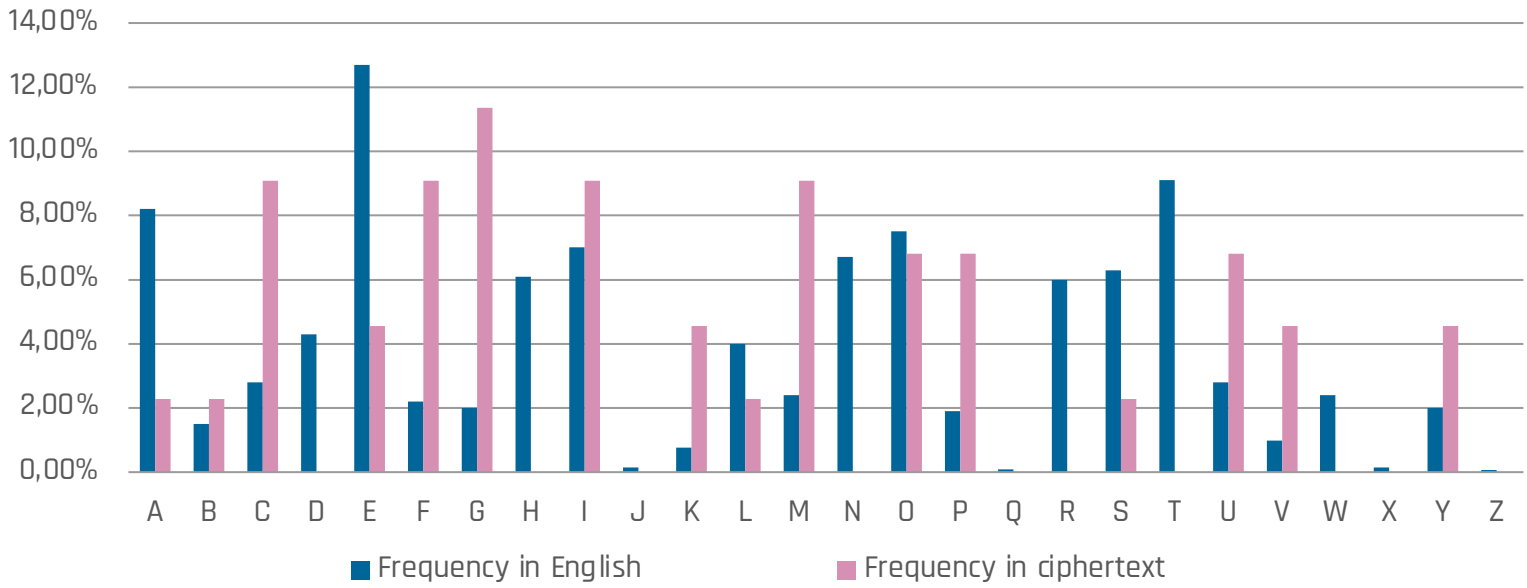
| | |
|---|---|
| Ciphertext | MUG PVFPMIMVMIKC YISUGN YOC FG GOPIBE FNKAGC FE UOCL |
| Plaintext | THE B T T T HE BE E B E B H |



# Example - Cracking a Ciphertext

Ciphertext MUG PVFPMIMVMIKC YISUGN YOC FG GOPIBE FNKAGC FE UOCL

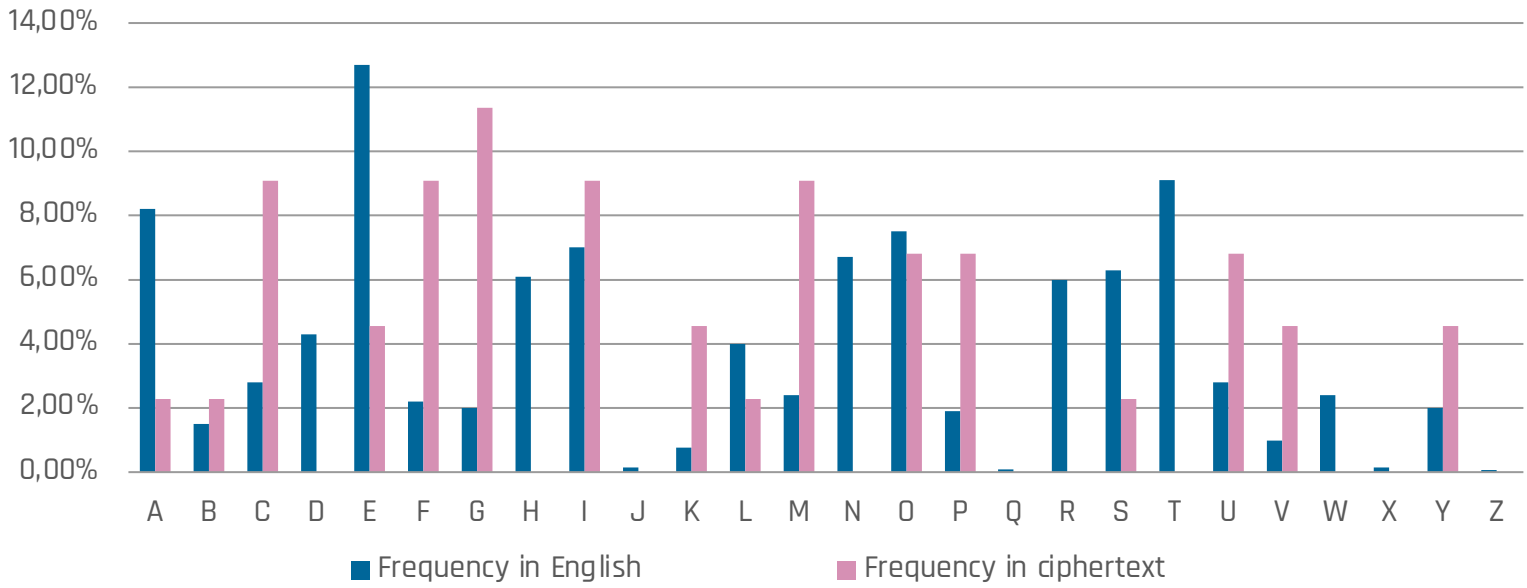
Plaintext THE B T T T HE BE E Y B E BY H



# Example - Cracking a Ciphertext

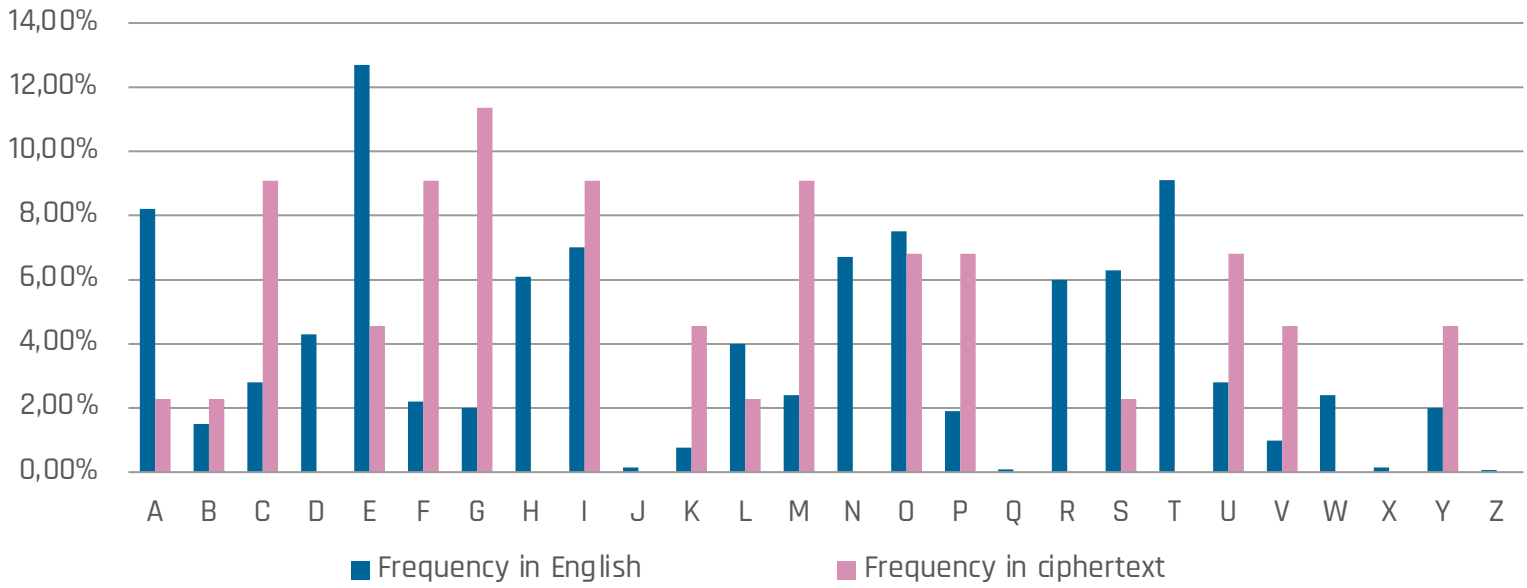
Ciphertext MUG PVFPMIMVMIKC YISUGN YOC FG GOPIBE FNKAGC FE UOCL

Plaintext THE B T T T N C HE CAN BE EA Y B EN BY HAN



# Example - Cracking a Ciphertext

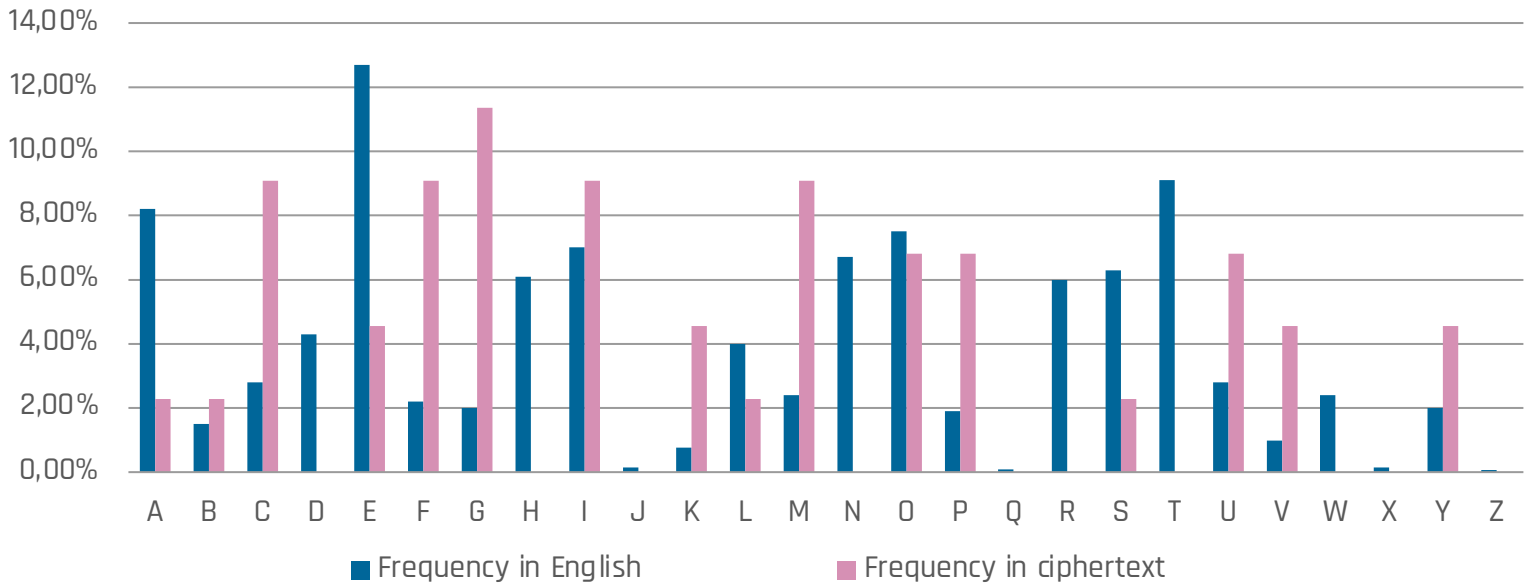
| | |
|---|---|
| Ciphertext | MUG PVFPMIMVMIKC YISUGN YOC FG GOPIBE FNKAGC FE UOCL |
| Plaintext | THE B T T T N C HE CAN BE EA Y B EN BY HAND |



# Example - Cracking a Ciphertext

Ciphertext MUG PVFPMIMVMIKC YISUGN YOC FG GOPIBE FNKAGC FE UOCL

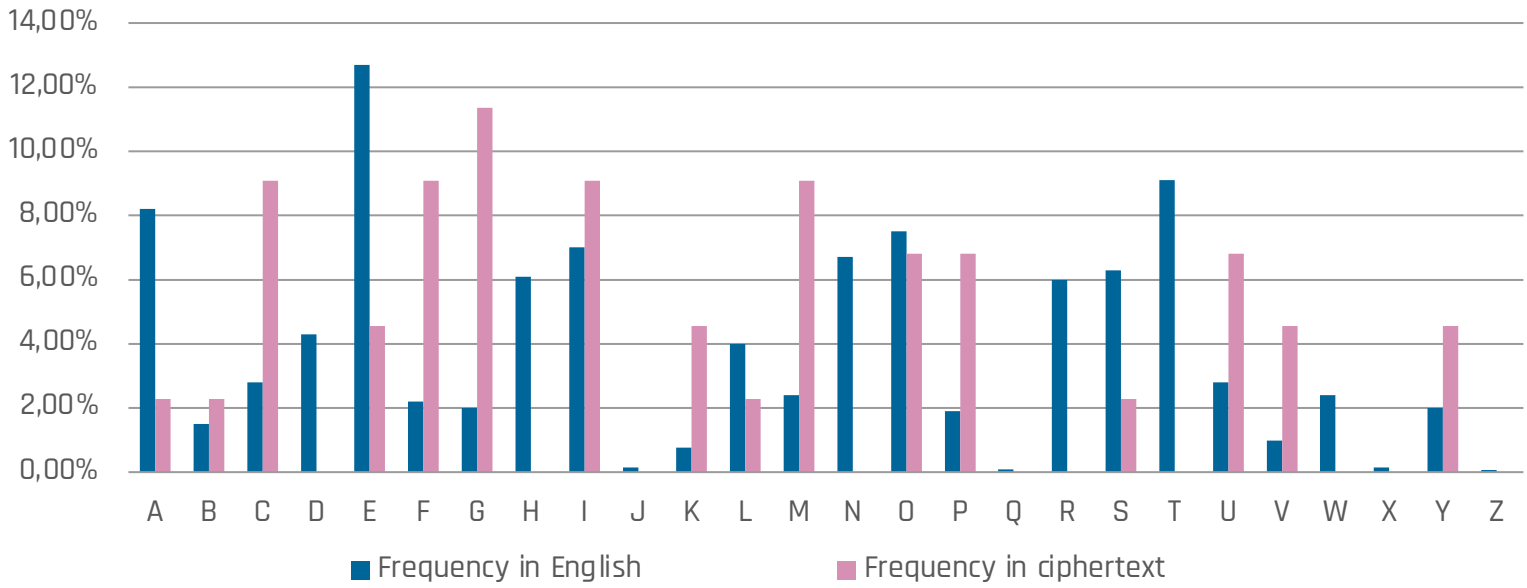
Plaintext THE B TIT TI N CI HE CAN BE EA I Y B EN BY HAND



# Example - Cracking a Ciphertext

Ciphertext MUG PVFPMIMVMIKC YISUGN YOC FG GOPIBE FNKAGC FE UOCL

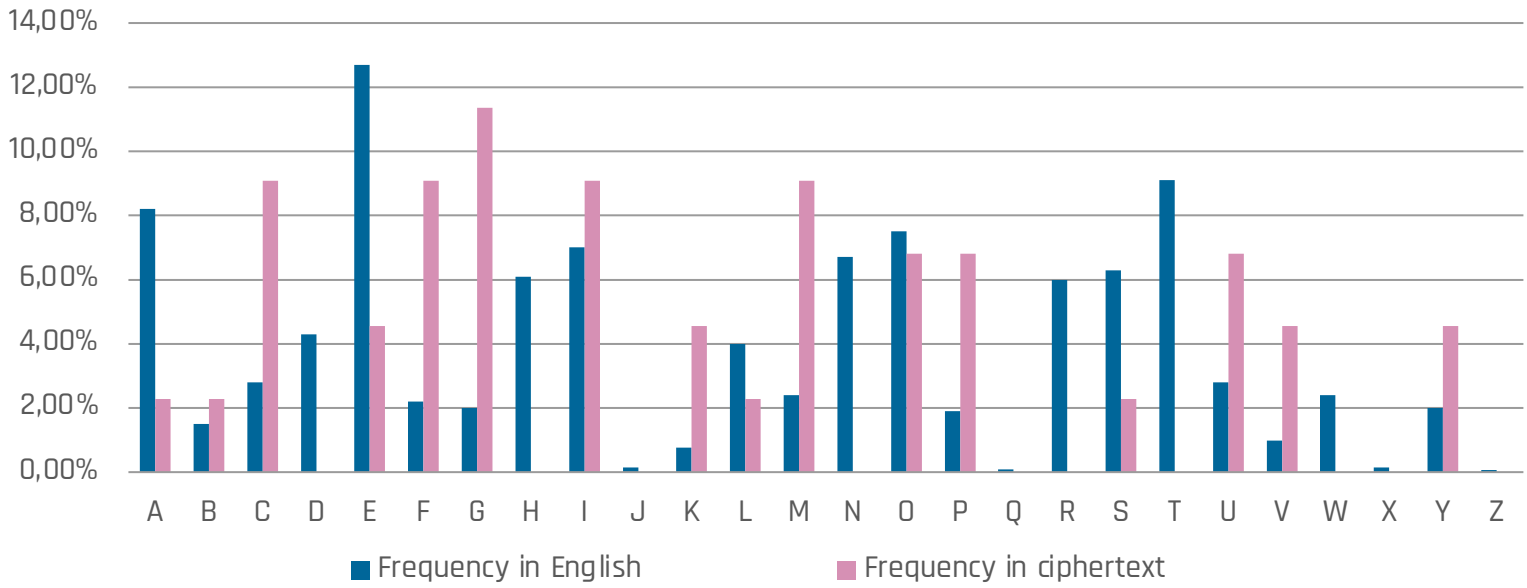
Plaintext THE S BSTIT TI N CI HE CAN BE EASILY B EN BY HAND



# Example - Cracking a Ciphertext

Ciphertext MUG PVFPMIMVMIKC YISUGN YOC FG GOPIBE FNKAGC FE UOCL

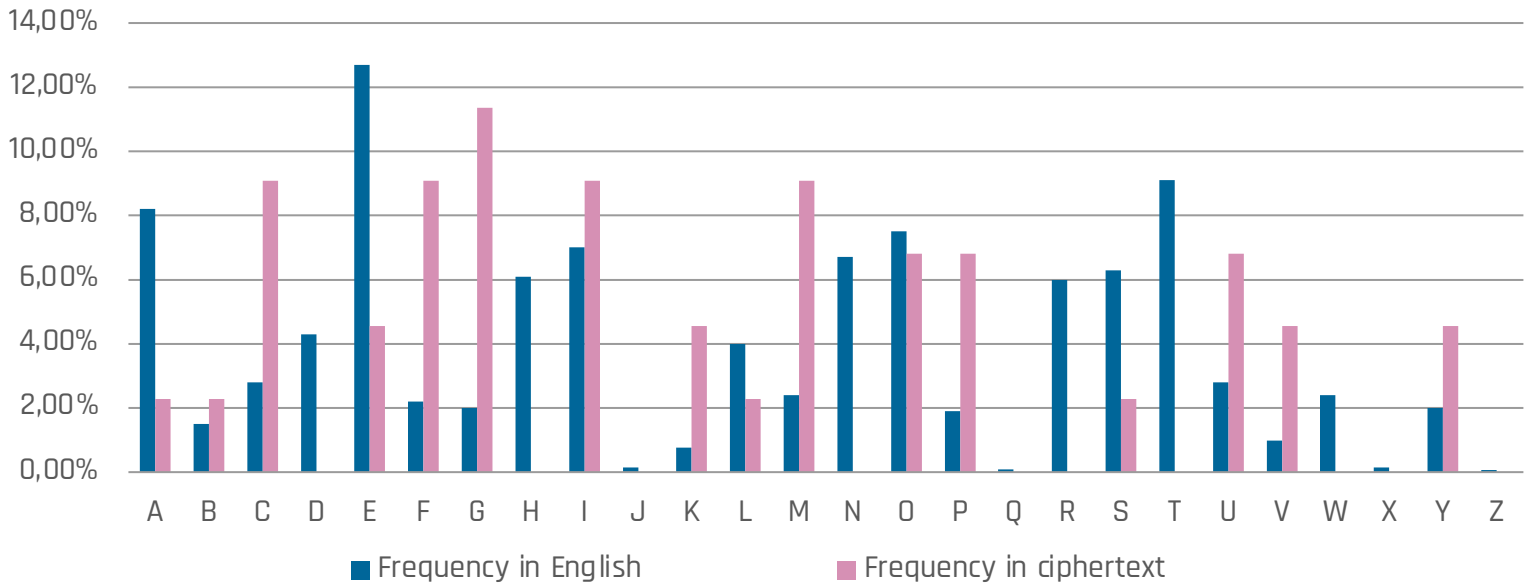
Plaintext THE SUBSTITUTION CI HE CAN BE EASILY B O EN BY HAND



# Example - Cracking a Ciphertext

Ciphertext MUG PVFPMIMVMIKC YISUGN YOC FG GOPIBE FNKAGC FE UOCL

Plaintext THE SUBSTITUTION CIPHER CAN BE EASILY BROKEN BY HAND



# Vigenère Cipher – A Polyalphabetic Substitution Cipher

- The Vigenère cipher **hides** the letter frequency by performing different substitutions depending on the letter's position in the plaintext
- The key is a sequence of numbers (of arbitrary length)
  - Every plaintext letter is assigned a number from the key
  - The key is cyclically repeated if shorter than the plaintext
  - Each number represents the size of a shift in the alphabet, which is used for encryption

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Plaintext | S | U | B | S | T | I | T | U | T | I | O | N |
| Key | 8 | 6 | 2 | 8 | 6 | 2 | 8 | 6 | 2 | 8 | 6 | 2 |
| Ciphertext | A | A | D | A | Z | K | B | A | V | Q | U | P |

# Vigenère Cipher – A Polyalphabetic Substitution Cipher

- The Vigenère cipher **hides** the letter frequency by performing different substitutions depending on the letter's position in the plaintext
- The key is a sequence of numbers (of arbitrary length)
  - Every plaintext letter is assigned a number from the key
  - The key is cyclically repeated if shorter than the plaintext
  - Each number represents the size of a shift in the alphabet, which is used for encryption

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Plaintext | S | U | B | S | T | I | T | U | T | I | O | N |
| Key | 8 | 6 | 2 | 8 | 6 | 2 | 8 | 6 | 2 | 8 | 6 | 2 |
| Ciphertext | A | A | D | A | Z | K | B | A | V | Q | U | P |

Same plaintext letter, possibly different ciphertext letters

# Vigenère Cipher – A Polyalphabetic Substitution Cipher

- The Vigenère cipher **hides** the letter frequency by performing different substitutions depending on the letter's position in the plaintext
- The key is a sequence of numbers (of arbitrary length)
  - Every plaintext letter is assigned a number from the key
  - The key is cyclically repeated if shorter than the plaintext
  - Each number represents the size of a shift in the alphabet, which is used for encryption

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Plaintext | S | U | B | S | T | I | T | U | T | I | O | N |
| Key | 8 | 6 | 2 | 8 | 6 | 2 | 8 | 6 | 2 | 8 | 6 | 2 |
| Ciphertext | A | A | D | A | Z | K | B | A | V | Q | U | P |

Same ciphertext letter, possibly different plaintext letters

# Security of the Vigenère Cipher

- Brute-force attacks are typically infeasible
  - Given a key length $l$, the number of possible keys is $26^l$ (Latin alphabet)
  - When $l$ is unknown, all possible values must be tried (up to plaintext length $L$)

- Statistical analysis can be quite effective for longer texts
  1. First the key length is determined
     - Kasiski test: searches for repeating patterns
     - Friedman test: uses a statistical measure known as coincidence index
  2. The ciphertext is split in different groups, each one containing the letters encrypted with the same key element
     - Each group corresponds to the encryption with a shift cipher and can be broken with frequency analysis

# Columnar Transposition

- The key is a list of numbers (of length $l$) containing each number from 1 to $l$
- The plaintext is entered line by line into a matrix with $l$ columns
  - An element of the key is assigned to each column
- The ciphertext is obtained by reading the matrix column by column
  - The key elements assigned to the columns determine in which order they are read

| | |
|---|---|
| Key | 3, 1, 2, 5, 4 |
| Plaintext | `THIS IS A TRANSPOSITION` |
| Ciphertext | `HSROOI ASNTITPI  STSANI` |

```
3 1 2 5 4
T H I S
I S   A
T R A N S
P O S I T
I O N
```

# Security of the Columnar Transposition

- Brute-force attacks are effective only for short keys
  - The number of possible keys is the factorial of the key length

- For larger keys, heuristic optimization methods (e.g., Hill climbing) can be applied
  - Key idea: Decryption with partially correct keys produce partially correct texts

# Perfect Secrecy

# Formal Definition of Symmetric Encryption Schemes

A **symmetric encryption scheme** with key space $\mathcal{K}$, message space $\mathcal{M}$ and ciphertext space $\mathcal{C}$ is a triple of algorithms $(Gen, Enc, Dec)$ where:

- The **key generation algorithm** $Gen$ takes no input and returns a randomly selected key $k \in \mathcal{K}$
- The **encryption algorithm** $Enc$ takes a key $k \in \mathcal{K}$ and a message $m \in \mathcal{M}$ and produces a ciphertext $c \in \mathcal{C}$
- The **decryption algorithm** $Dec$ takes a key $k \in \mathcal{K}$ and a ciphertext $c \in \mathcal{C}$ and produces a plaintext $m \in \mathcal{M}$ (or an error)

# Correctness of a Symmetric Encryption Scheme

The symmetric encryption scheme $(Gen, Enc, Dec)$ is **correct** if for all keys $k \in \mathcal{K}$ and messages $m \in \mathcal{M}$, the following property holds:

$$Dec\big(k, Enc(k, m)\big) = m$$

# Perfect Secrecy

- Information-theoretic definition proposed by Claude Shannon
  - Only-ciphertext attacker model: given a ciphertext, the attacker tries to find the corresponding plaintext or the encryption key

- An encryption scheme provides perfect secrecy if a ciphertext does not reveal any information about the corresponding plaintext
  - By observing the ciphertext, an attacker does not learn anything about the plaintext that was not already known before

# Formal Definition of Perfect Secrecy

- Let M, C, K be random variables denoting the value of the plaintext, ciphertext and key (for a given probability distribution) with following assumptions:
  - K is uniformly distributed over the key space $\mathcal{K}$
  - M and K are independent

An encryption scheme $(Gen, Enc, Dec)$ provides **perfect secrecy** if for every probability distribution over $\mathcal{M}$, every message $m \in \mathcal{M}$ and every ciphertext $c \in \mathcal{C}$, the following property holds:

$$\Pr[M = m \mid C = c] = \Pr[M = m]$$

# Perfect Ciphers Require Keys as Long as Messages

Let $(Gen, Enc, Dec)$ be a cipher providing perfect secrecy and let $\Pr[C = c] > 0$ for every ciphertext $c \in \mathcal{C}$. Then $|\mathcal{K}| \geq |\mathcal{M}|$.

## Proof

- By definition of perfect secrecy and Bayes theorem we have $\Pr[C = c | M = m] = \Pr[C = c]$ for every $m \in \mathcal{M}$ and $c \in \mathcal{C}$
- If we fix $m$, we know that there exists a key $k \in \mathcal{K}$ such that $Enc(k, m) = c$
  - If not, $\Pr[C = c | M = m] = 0 \neq \Pr[C = c]$ (hypothesis violation)
- Since $Enc$ is a function and $m$ is fixed, we have $|\mathcal{K}| \geq |\mathcal{C}|$
- Since $Enc$ is injective, we have $|\mathcal{C}| \geq |\mathcal{M}|$

$$|\mathcal{K}| \geq |\mathcal{M}|$$

# The One-Time-Pad

- First cipher that was proved to offer perfect secrecy (1949)
  - Plaintexts, ciphertexts and keys are bitstrings
  - The key is randomly generated and is as long as the plaintext
  - Must be newly generated for every message!

$$Enc(k, m) = k \oplus m$$
$$Dec(k, c) = k \oplus c$$

Bitwise XOR

| $\oplus$ | 0 | 1 |
|---|---|---|
| 0 | 0 | 1 |
| 1 | 1 | 0 |

| | |
|---|---|
| Plaintext | 01000100 11110010 01101001 |
| Key | 11010010 11100110 00011011 |
| Ciphertext | 10010110 00010100 01110010 |

# Correctness and Perfect Secrecy of the One-Time-Pad

## Correctness

$$Dec\big(k, Enc(k,m)\big) = k \oplus Enc(k,m) = k \oplus k \oplus m = 0 \oplus m = m$$

## Perfect Secrecy

$$\begin{aligned}&\Pr[M = m \mid C = c]\\ &= \Pr[M = m \mid K = m \oplus c]\\ &= \Pr[M = m]\end{aligned}$$

Only the key $m \oplus c$ can produce the observed ciphertext

Due to independence of M and K

# Problems of the One-Time-Pad

- Requirements on the key are a deal-breaker
  - True randomness of the key is difficult to achieve
  - Key storage requires as much space as the ciphertext itself
  - Key must be communicated in a secure way to the receiver and used only once

- No integrity: changes to the ciphertext are reflected to the plaintext

- Key reusage completely breaks the cipher
  - If $c_1 = Enc(k, m_1)$ and $c_2 = Enc(k, m_2)$, then $c_1 \oplus c_2 = m_1 \oplus m_2$
    - If both plaintexts are in a natural language, they can typically be recovered using heuristics
    - If one plaintext is known to the attacker, the other one can be easily computed

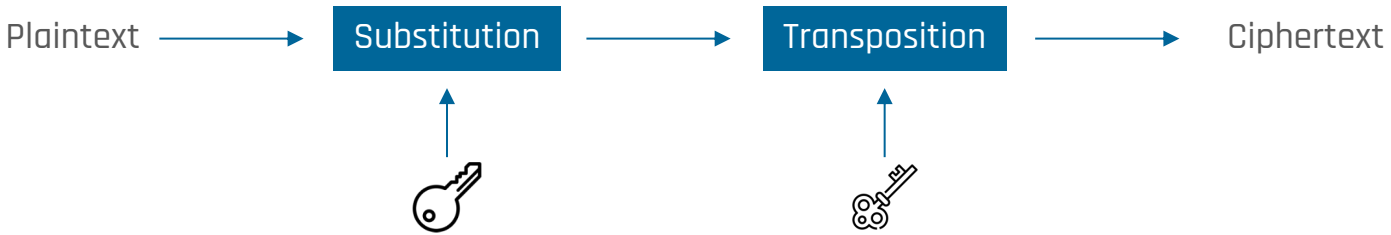
$$m_1 = c_1 \oplus c_2 \oplus m_2$$

# Modern Cryptography

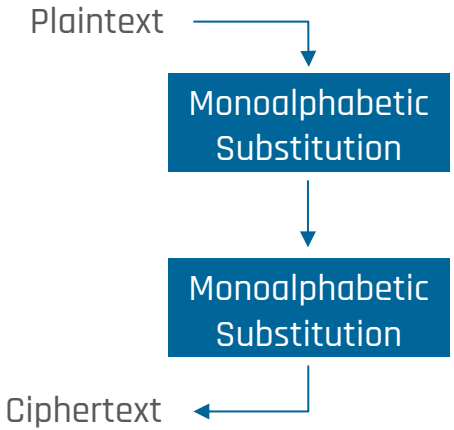
# Product Ciphers

A **product cipher** is a cipher that consists of the combination of multiple, possibly iterated transformations

- Goal is to produce a cipher stronger than its individual components
  - Modern ciphers consist of the iterative combination of substitutions, transpositions and modular arithmetic (e.g., AES)



# Not All Combinations are Good!



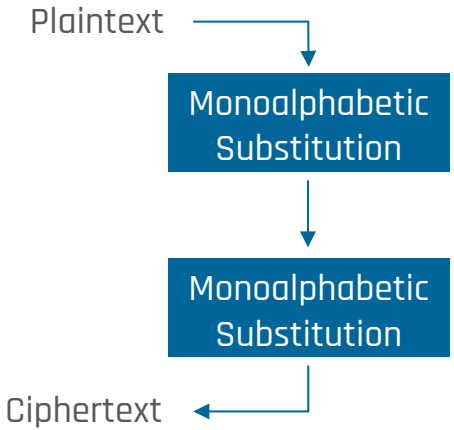

| A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Q | M | O | Z | C | L | A | R | E | K | U | N | V | B | H | Y | I | F | G | X | P | S | D | W | J | T |

| A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| D | H | S | X | N | Z | R | M | B | Q | W | F | U | V | J | G | Y | T | A | E | C | K | O | P | I | L |

Equivalent to


| A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Y | | | | | | | | | | | | | | | | | | | | | | | | | |

# Not All Combinations are Good!



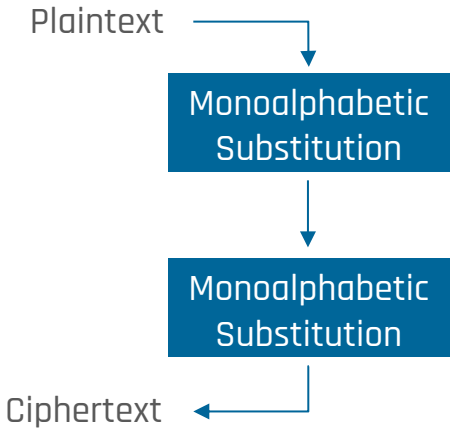

| A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Q | M | O | Z | C | L | A | R | E | K | U | N | V | B | H | Y | I | F | G | X | P | S | D | W | J | T |

| A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| D | H | S | X | N | Z | R | M | B | Q | W | F | U | V | J | G | Y | T | A | E | C | K | O | P | I | L |

Equivalent to


| A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Y | U | | | | | | | | | | | | | | | | | | | | | | | | |

# Not All Combinations are Good!




| A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Q | M | O | Z | C | L | A | R | E | K | U | N | V | B | H | Y | I | F | G | X | P | S | D | W | J | T |

| A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| D | H | S | X | N | Z | R | M | B | Q | W | F | U | V | J | G | Y | T | A | E | C | K | O | P | I | L |

Equivalent to


| A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Y | U | J | L | S | F | D | T | N | W | C | V | K | H | M | I | B | Z | R | P | G | A | X | O | Q | E |

# Confusion and Diffusion

- **Confusion** – No relationship between plaintext and ciphertext should be recognizable
  - Distribution of bytes in the plaintext should be concealed in the ciphertext
  - Changes to the plaintext should have unpredictable repercussions to the ciphertext

- **Diffusion** – Every bit of the plaintext and the key should spread over the ciphertext and influence as many of its bits as possible
  - Changing one bit in the plaintext/key should trigger the change of about half of the ciphertext bits (**avalanche effect**)

# Block vs. Stream Ciphers

## Block Ciphers

- Plaintext is split into blocks of **equal length**
- **Padding algorithms** are used to fill the last block if shorter than the block length
- A **mode of operation** determines how the encryption is applied to each block
- The same key is used for the encryption of every block


## Stream Ciphers

- A sequence of pseudorandom bytes (**keystream**) is derived from the **key**
  - Sometimes already processed parts of plaintext or ciphertext are used
- The keystream is combined with the plaintext (typically via **bitwise XOR**) to produce the ciphertext

# Stream Ciphers

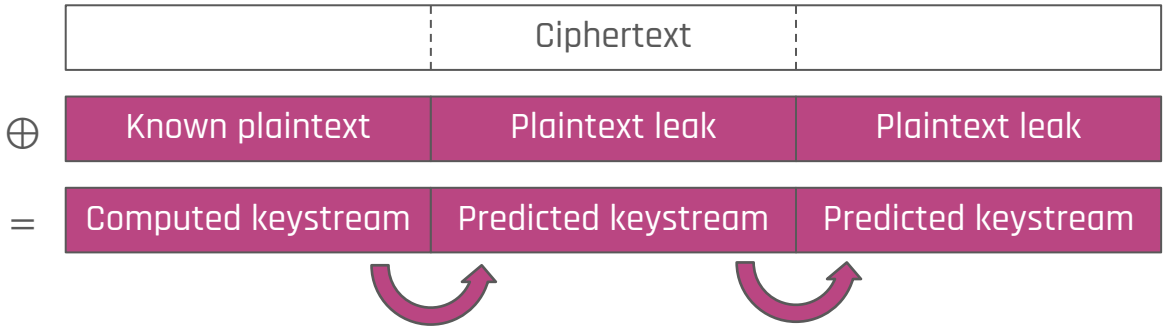
# Stream Ciphers

- An approximation of the One-Time-Pad
  - The key is used as **seed** to initialize the **state** of a **pseudorandom generator** (PRG)
  - The PRG generates the **keystream** which is combined with the plaintext via XOR

- PRG state is updated periodically (e.g., before the generation of each output)
  - **Synchronous ciphers**: the next state depends only on the current state
  - **Self-synchronizing ciphers**: the next state (also) depends on previous plaintexts/ciphertexts

# Security of Stream Ciphers

- No perfect secrecy, actual security depends on the PRG
  - Next PRG output should not be predictable from the previous ones
  - Otherwise, the cipher can be broken if part of the plaintext is known/guessable



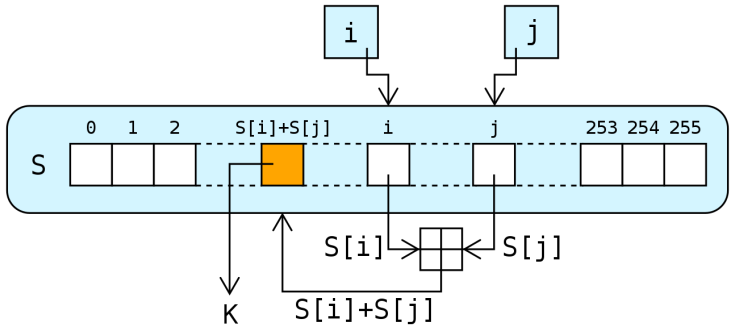
# Malleability

An encryption algorithm is **malleable** if it is possible to transform a ciphertext into another one that decrypts to a message bearing some relation with the original plaintext

- Generally undesired, because an attacker can **modify** messages in a **controlled way**
  - Used in homomorphic schemes to perform computations over encrypted values
- Stream ciphers are **malleable**
  - Given $c = Enc(k, m)$, one can compute $Enc(k, m \oplus t) = c \oplus t$

# RC4 – A (Nowadays Broken) Stream Cipher

- Developed by Ron Rivest in 1984
  - PRG state size 2064 bits
    - A permutation S of all 256 possible bytes
    - Two indexes `i`, `j`
  - Key size between 40 and 2048 bits
    - Used to generate the initial permutation of bytes




```
i := 0, j := 0
while GeneratingOutput:
    i := (i + 1) % 256
    j := (j + S[i]) % 256
    swap S[i] and S[j]
    t := (S[i] + S[j]) % 256
    K := S[t]
    output K
endwhile
```

# Block Ciphers

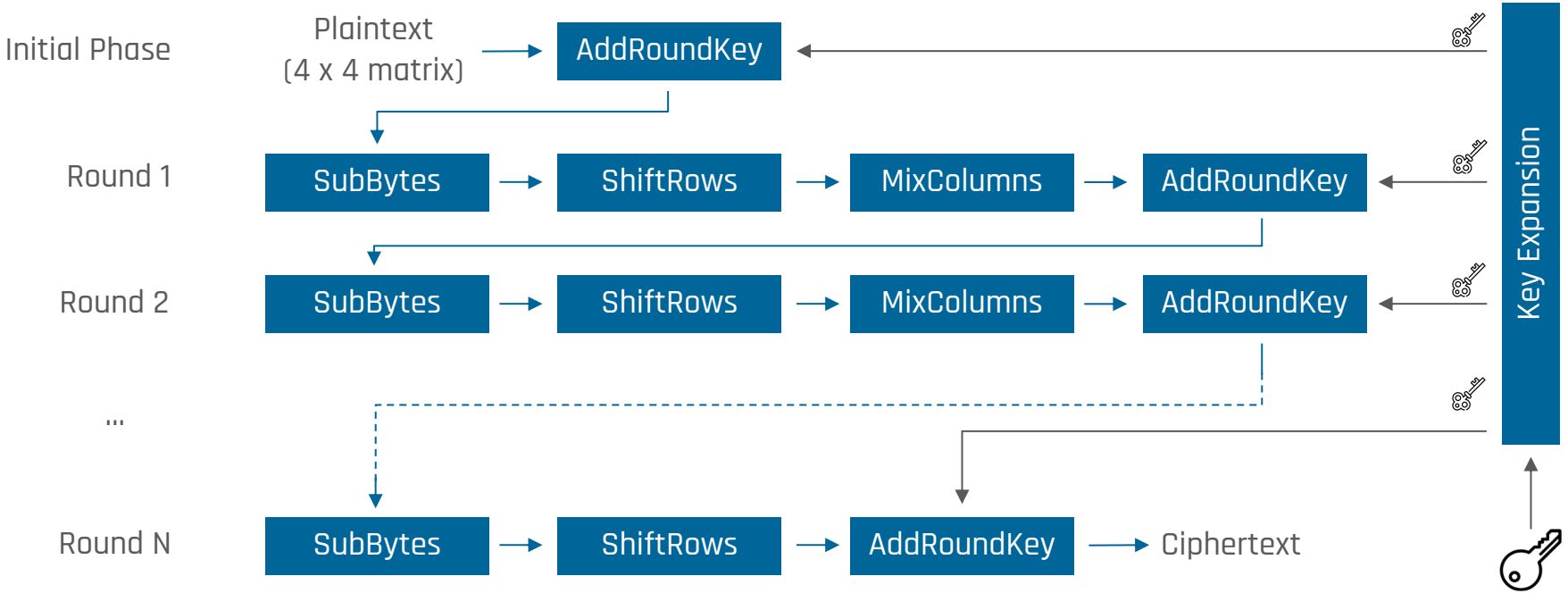
# AES - Overview

- Most used **block cipher** for data encryption
  - Variation of the Rijndael algorithm developed by Joan Daemen und Vincent Rijmen
  - Standardised in 2001 after a five-year process

- Characteristics
  - Block size: 128 bits
  - Variable key size: 128, 192, 256 bits
  - Good performance both in hardware and software
    - Native support in most of the recent CPU architectures
  - Product cipher consisting of multiple rounds
    - Each round consists of non-linear substitutions, permutations, combination with a subkey

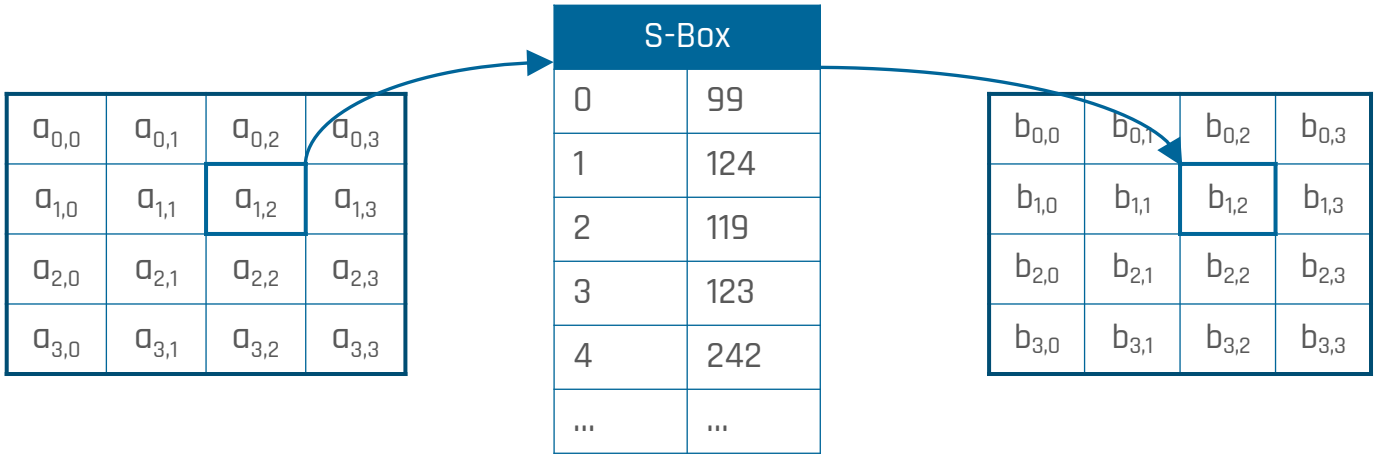
| Key size | # rounds |
|---|---|
| 128 | 10 |
| 192 | 12 |
| 256 | 14 |

# Overview of AES Encryption



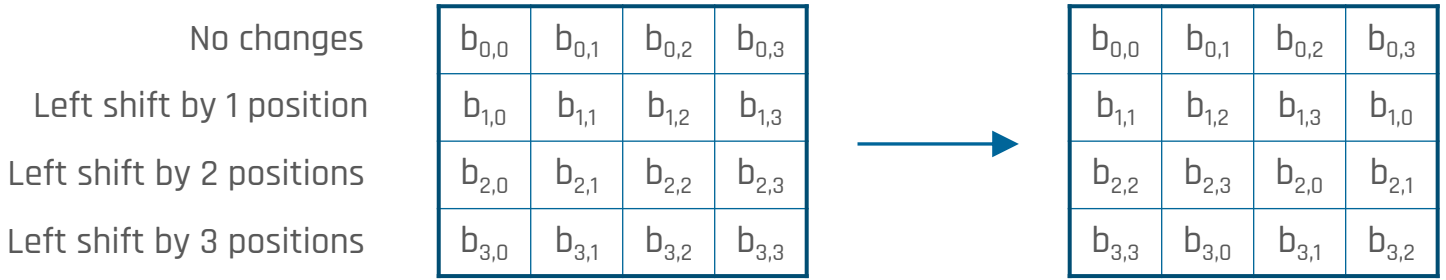
# The SubBytes Step

- Every byte in the current state is substituted with another byte according to a fixed **substitution table** (S-Box)
  - This transformation introduces **non-linearity** in the cipher
  - Provides resistance against **differential** and **linear cryptanalysis**



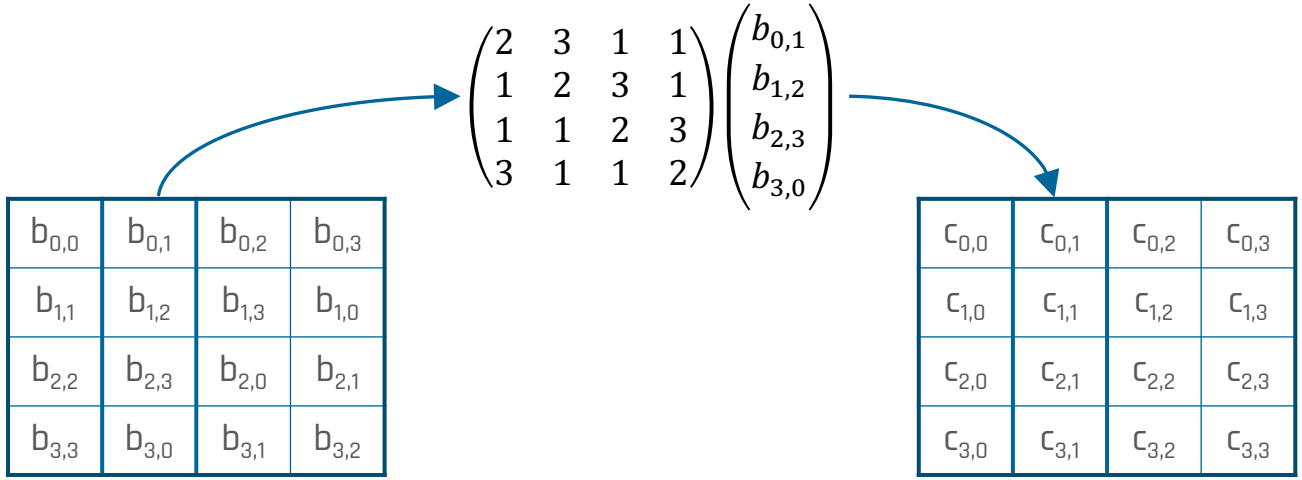
# The ShiftRows Step

- State elements are **left shifted** by an offset that depends on the line
  - Each column of the output state contains an element **from each column** of the initial one
  - Prevents that columns are **encrypted separately**



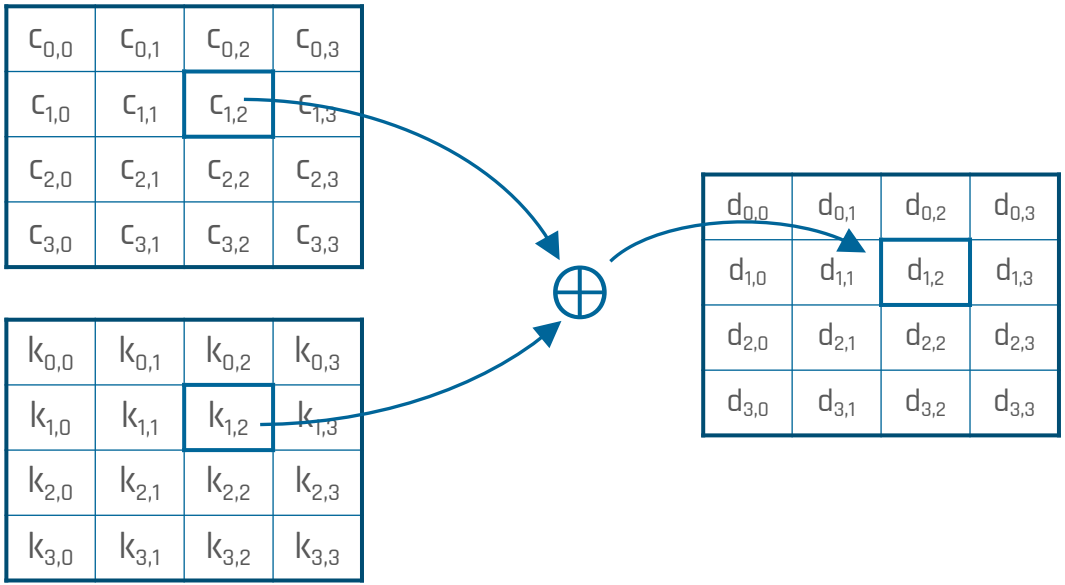
# The MixColumn Step

- Each column of the state is multiplied with a **fixed matrix**
  - In this way, each input byte affects **all** output bytes
  - Contributes to **diffusion** (together with ShiftRows)



# The AddRoundKey Step

- The **subkey** (128 bit) for the current round is **XORed** with the current state
  - Subkeys are derived from the **original key** using XOR, cyclic shifts and substitutions



# Modes of Operation

A **mode of operation** is an algorithm that determines how to use a block cipher to process messages larger than a block

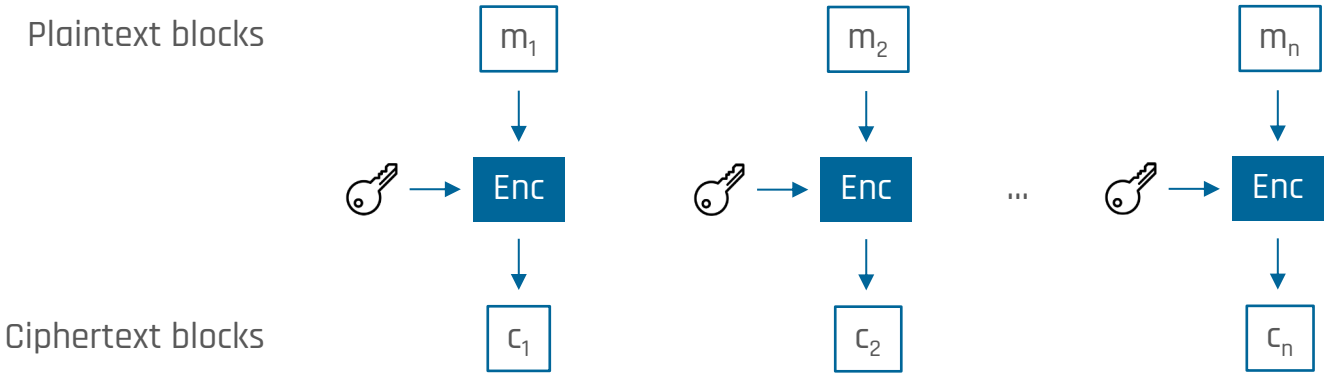
- Classic modes of operations: ECB, CBC, CFB, OFB, CTR
  - Focus exclusively on data confidentiality
  - Additional mechanisms required for integrity and authenticity (e.g., MACs)

- Authenticated modes of operation: GCM, CCM, EAX
  - Provide simultaneously confidentiality, integrity and authenticity

# Relevant Characteristics of Modes of Operation

- **Parallelization** – Possibility to parallelize encryption or decryption of message blocks
- **Random read access** – Possibility to access to the content of a specific message block without decrypting all preceding blocks
- **Error propagation** – Impact of a change in the ciphertext to the decrypted plaintext
  - Only relevant for classical modes
  - Authenticated modes should **abort decryption** in case of **integrity violations**

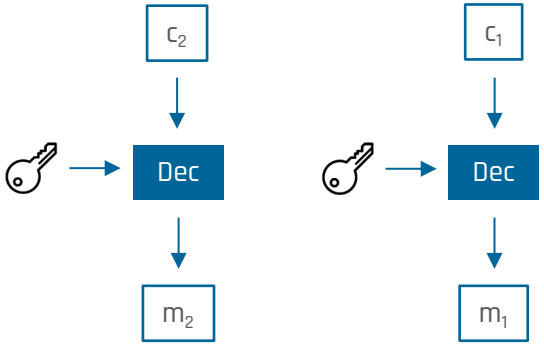
# Electronic Code Book Mode (ECB)

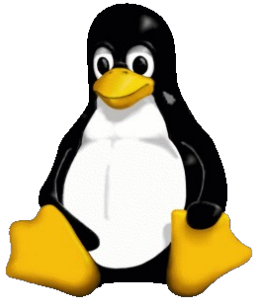



- Plaintext blocks are processed independently from each other
  - Parallelization (both encryption and decryption) and random read access are possible
  - A bit flip in a ciphertext block produces random errors exclusively in the corresponding plaintext block

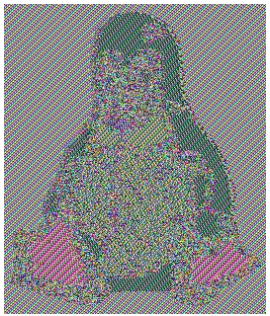
# Issues of ECB

- Patterns in the plaintext are **visible** in the ciphertext
- **Strong malleability**: new ciphertexts (for a given key) can be generated by piecing together blocks from previously observed ciphertexts



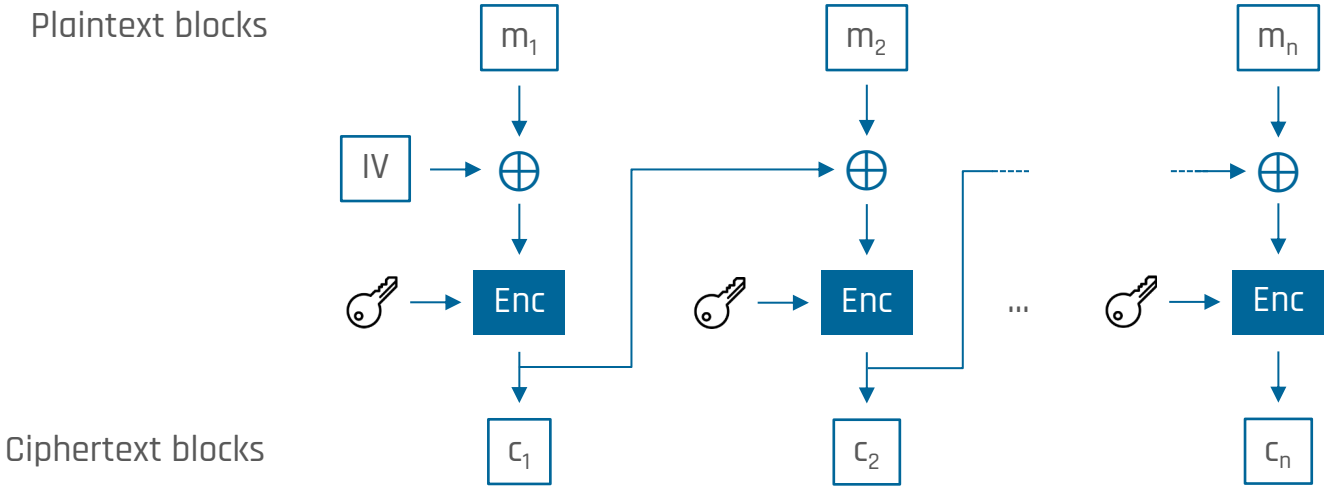



Encryption with ECB



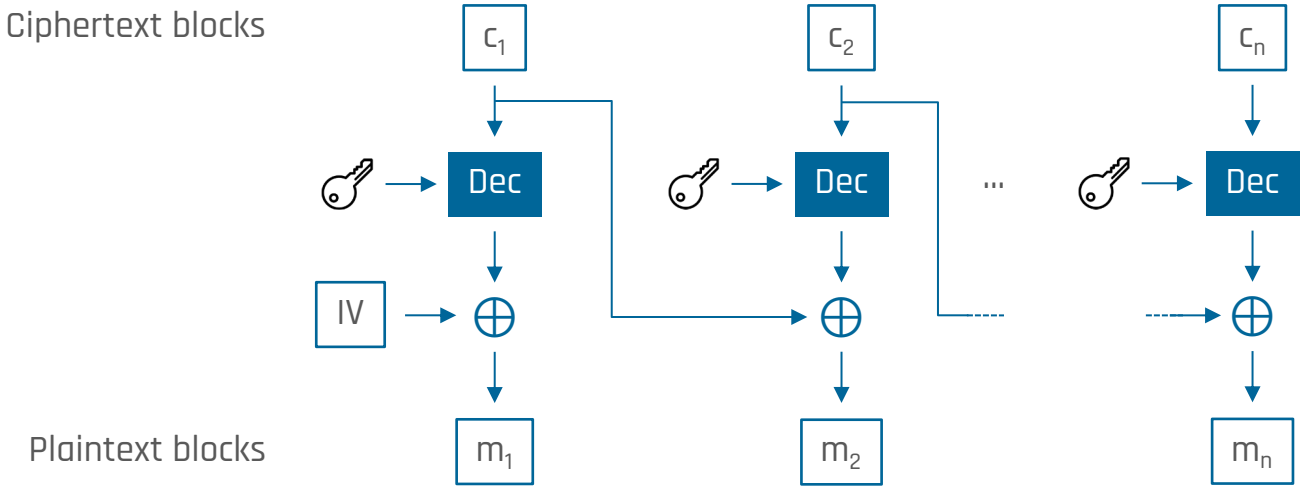
**ECB should be avoided**, unless the message to be encrypted is not larger than a block!

# Cipher Block Chaining Mode (CBC)



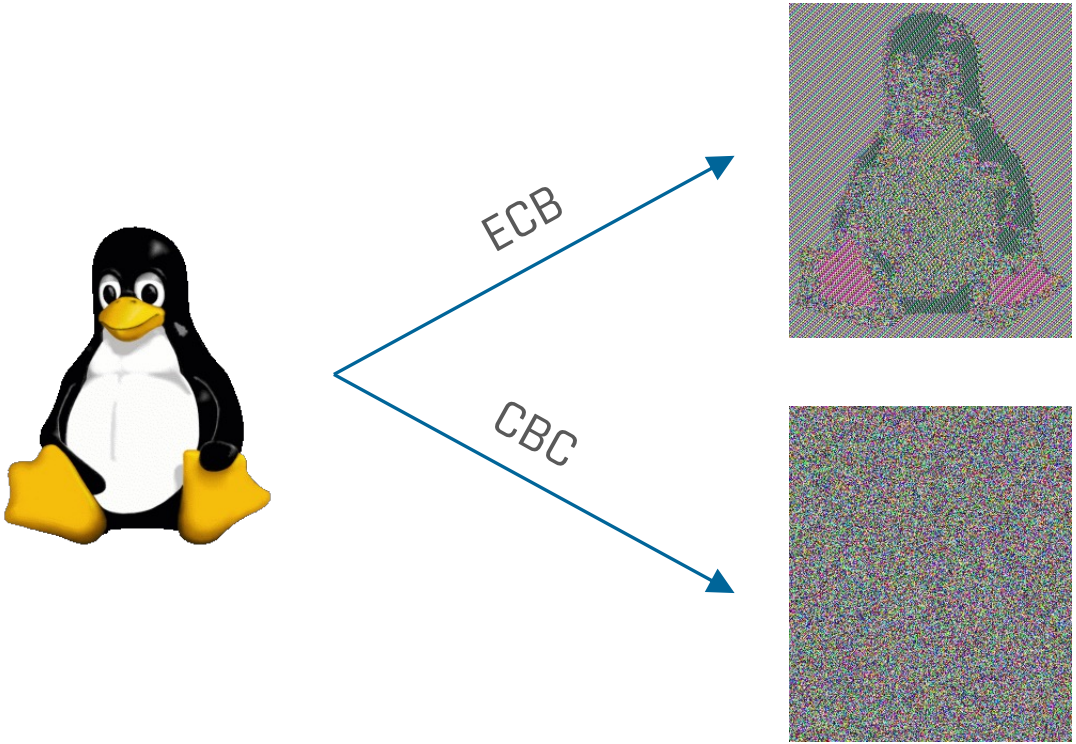

- Each message block is XORed with the encryption of the previous block before being encrypted
  - The first block is XORed with a randomly generated initialization vector (IV), typically prepended to the ciphertext
  - Encryption is non-deterministic: the same pair (plaintext, key) might produce different ciphertexts

# Cipher Block Chaining Mode (CBC)

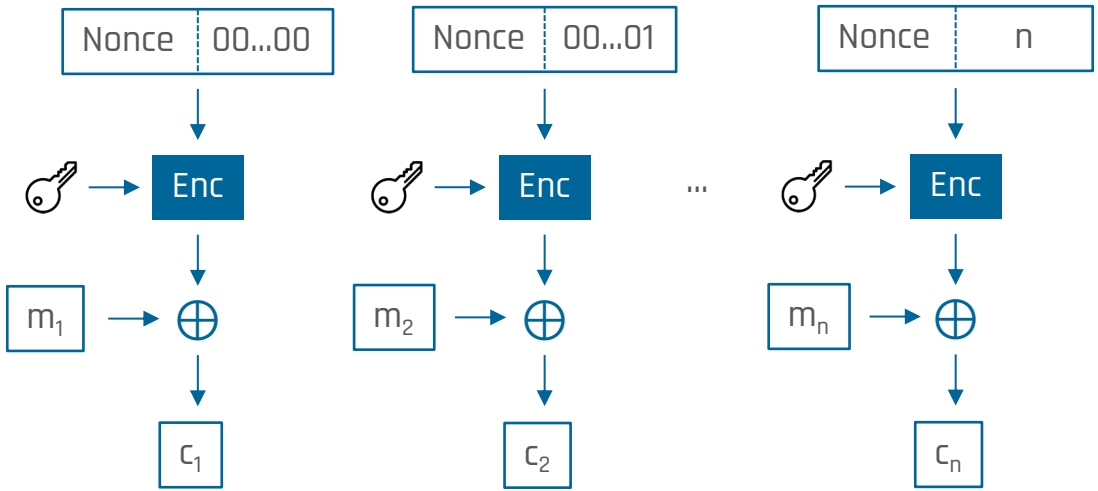



- Parallelisation of decryption and random read access are possible
  - No parallel encryption due to dependency from the previous ciphertext block
- A bit flip in a ciphertext block produces random errors in the corresponding plaintext block and a bit flip at the same position in the next plaintext block
  - A bit flip in the IV causes a flip of the corresponding bit in the first plaintext block

# CBC Reveals No Patterns

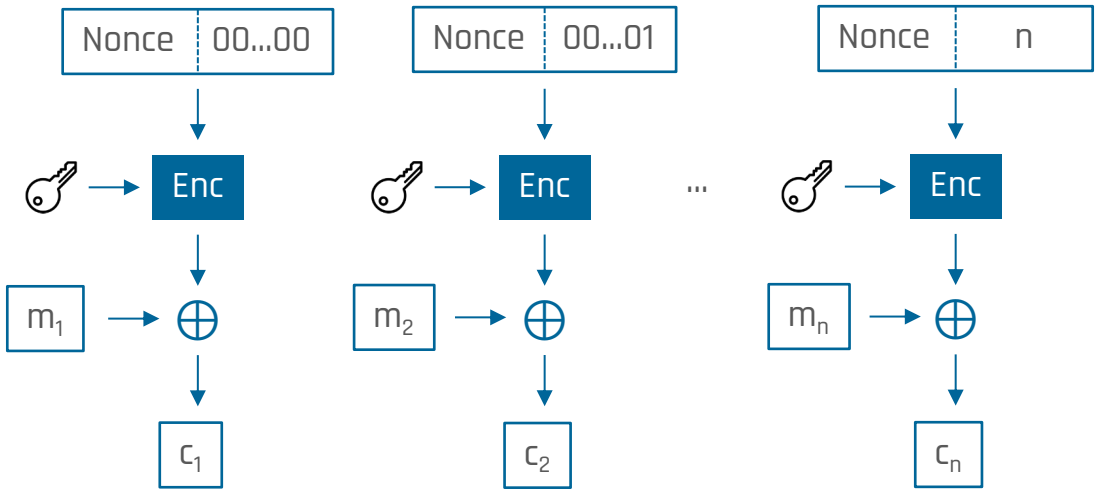


# Counter Mode (CTR)




- This modes implements a **stream cipher** starting from a block cipher
  - The block cipher is used to produce the keystream
  - The input to the block cipher consists of a **random nonce** and a counter
    - The random nonce must be communicated to the decrypting party (e.g., prepended to the ciphertext)

# Counter Mode (CTR)



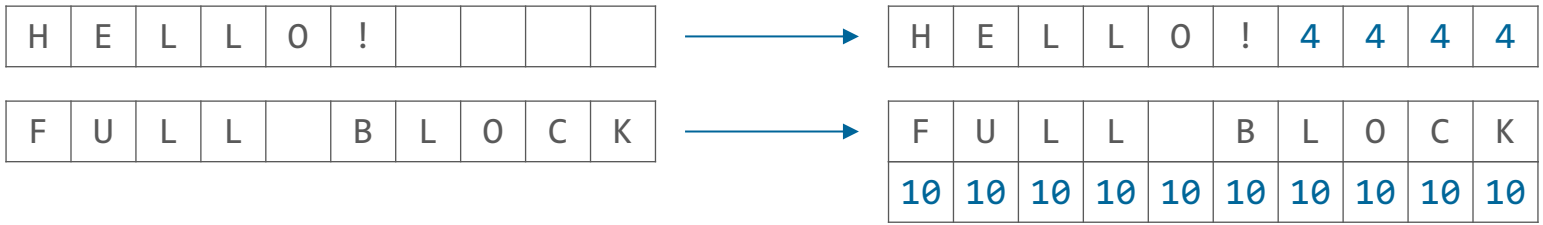

- Provides parallelization of encryption/decryption and random read access
- A bit flip in a ciphertext block causes a bit flip at the corresponding position in the plaintext
  - A bit flip in the nonce produces unpredictable errors in all decrypted blocks
- The nonce cannot be reused, otherwise the keystream is the same (similar problems as in the One Time Pad)

# Padding

- Padding algorithms are used to fill the last block of a plaintext if its length is not a multiple of the block size

- PKCS#7 is a standard padding algorithm for block ciphers
  - If N bytes are missing in the last block, add N bytes with value N
  - If the last block is full, add a new block where all bytes match the block size

**Examples (block size 10)**



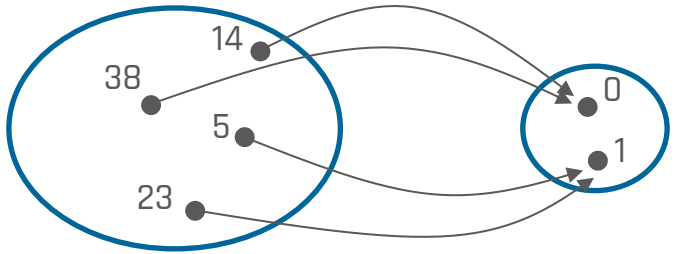
# Hash Functions

# Hash Functions

A **hash function** is any function that maps inputs of **arbitrary** size to outputs of a **fixed** size

- Desired properties of hash functions
  - Very fast to compute
  - Minimizes the number of **collisions**
    - Collisions arise when different inputs are mapped to the same output
    - Necessarily present due to infinite number of possible inputs!

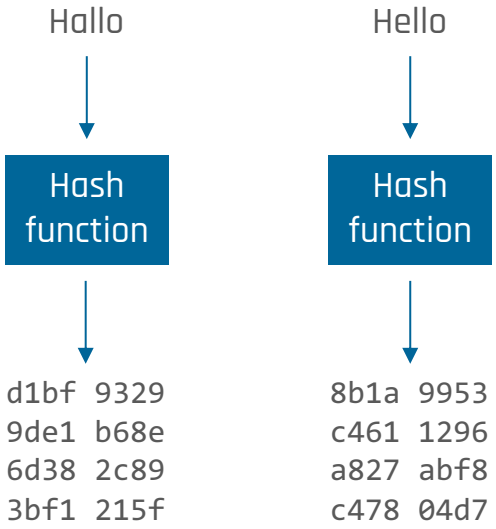
$$h(x) = x \bmod 2$$




Input space: $\mathbb{N}$ Output space: $\{0, 1\}$

# Cryptographic Hash Functions

- Special type of hash functions satisfying the following properties
  - **One-way**: easy to compute the hash for a given input, **infeasible** to compute an input that maps to a given hash value
  - **Uniformity**: the probability of obtaining a particular $n$-bit output for a random input should be $2^{-n}$
  - **Diffusion**: each bit of the input message should affect as many output bits as possible
  - **Collision resistance**



# Many Flavours of Collision Resistance

- **Weak collision resistance**
  - Given a message $m$ and a hash function $h$, it is infeasible to find another message $m'$ such that $h(m) = h(m')$
- **Strong collision resistance**
  - Given a hash function $h$, it is infeasible to find two different messages $m, m'$ such that $h(m) = h(m')$
- **Resistance against near-collisions**
  - Given a hash function $h$, it is infeasible to find two different messages $m, m'$ such that their hashes $h(m), h(m')$ differ only in a few bits

# Overview of Famous Hash Functions

| Hash function | Year | Hash length | Secure? |
|---|---|---|---|
| MD5 | 1991 | 128 bits | **No** (efficient attacks for finding collisions published in 2005) |
| SHA-1 | 1995 | 160 bits | **No** (first public collision in 2017, chosen-prefix attacks are practical) |
| SHA-2 | 2002 | 224, 256, 384, 512 bits | **Yes** (current attacks work only on reduced versions with less rounds) |
| SHA-3 | 2012 | 224, 256, 384, 512 bits | **Yes** |

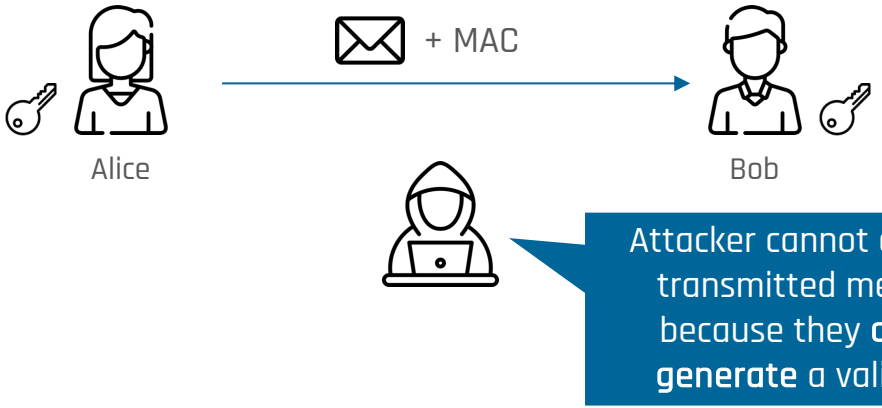
# Usage of Hash Functions

- **Integrity of messages and files**
  - Some websites provide the hash value of downloadable files to ensure their integrity
  - The hash must be communicated in a **secure way** to protect against **adversarial changes**!
- **Digital signatures**
  - The signature of a document is computed over the hash of the document to speed up the computation
- **Password storage**
  - Password hashes are saved instead of passwords
  - Hinders password recovery if the hashes are leaked

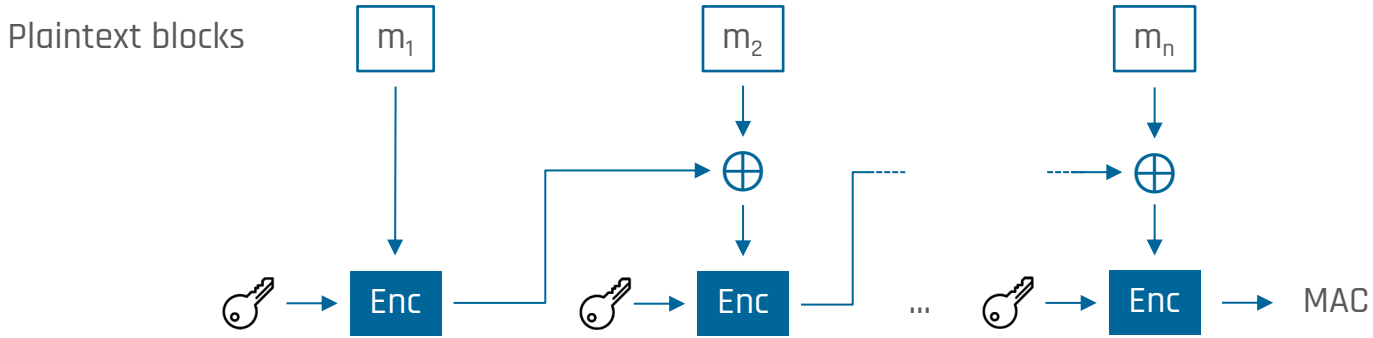
# Message Authentication Codes

# Message Authentication Codes (MAC)

- MACs are used to guarantee the **integrity** and **authenticity** of a message
  - Based on block ciphers or hash functions
  - **Shared key** needed for generating / verifying the MAC



# CBC-MAC

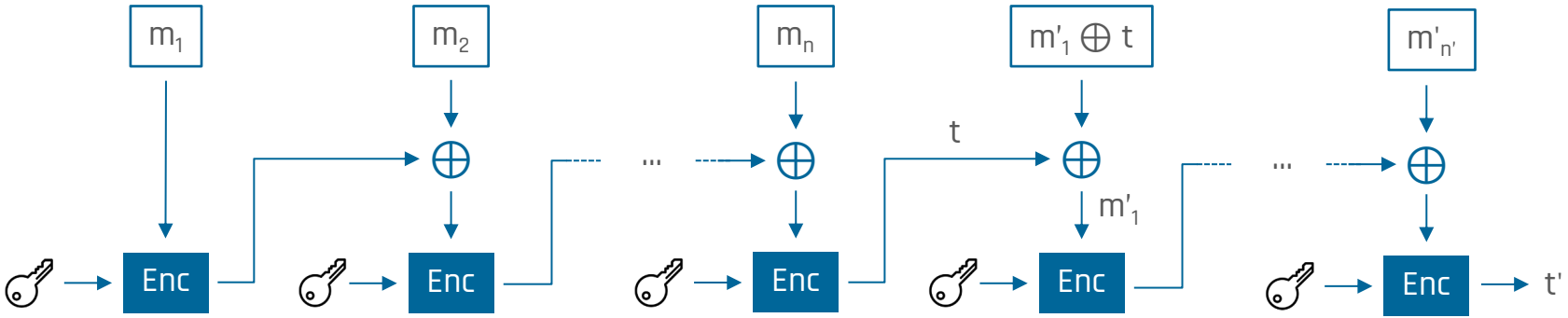



- Based on block cipher encryption in **CBC mode** (with null initialisation vector)
  - MAC is the block resulting from the **last encryption**
- Secure **only** for **fixed-length messages**

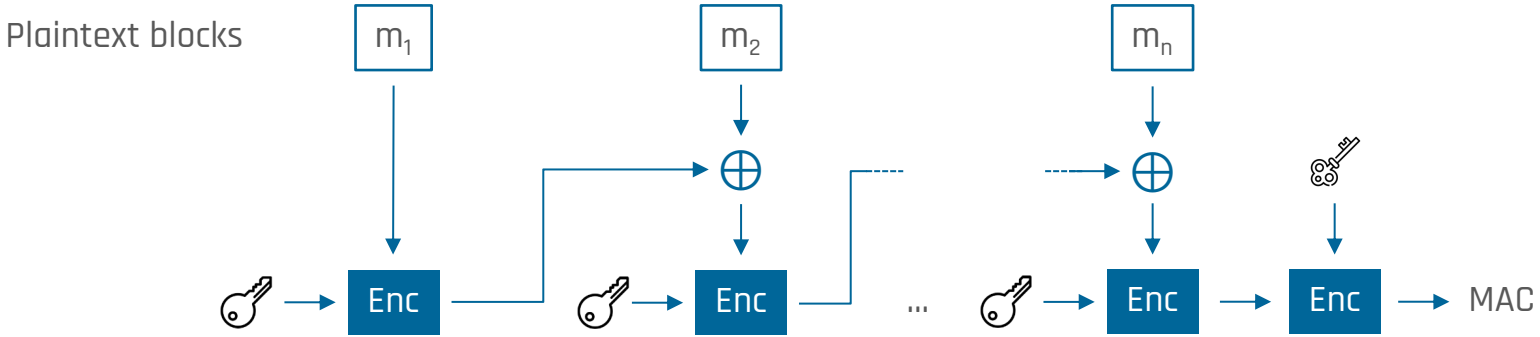
# Attack with Variable Message Size

- Let $t = CBCMAC(k, m)$ and $t' = CBCMAC(k, m')$
  - Both MACs and messages are known to the attacker
- Let $m'' = m \parallel (m'_1 \oplus t) \parallel m'_2 \parallel \cdots \parallel m'_{n'}$
  - The MAC of $m''$ is equal to $t'$

Concatenation of $m$ with $m'$ but the first block of the latter is XORed with $t$



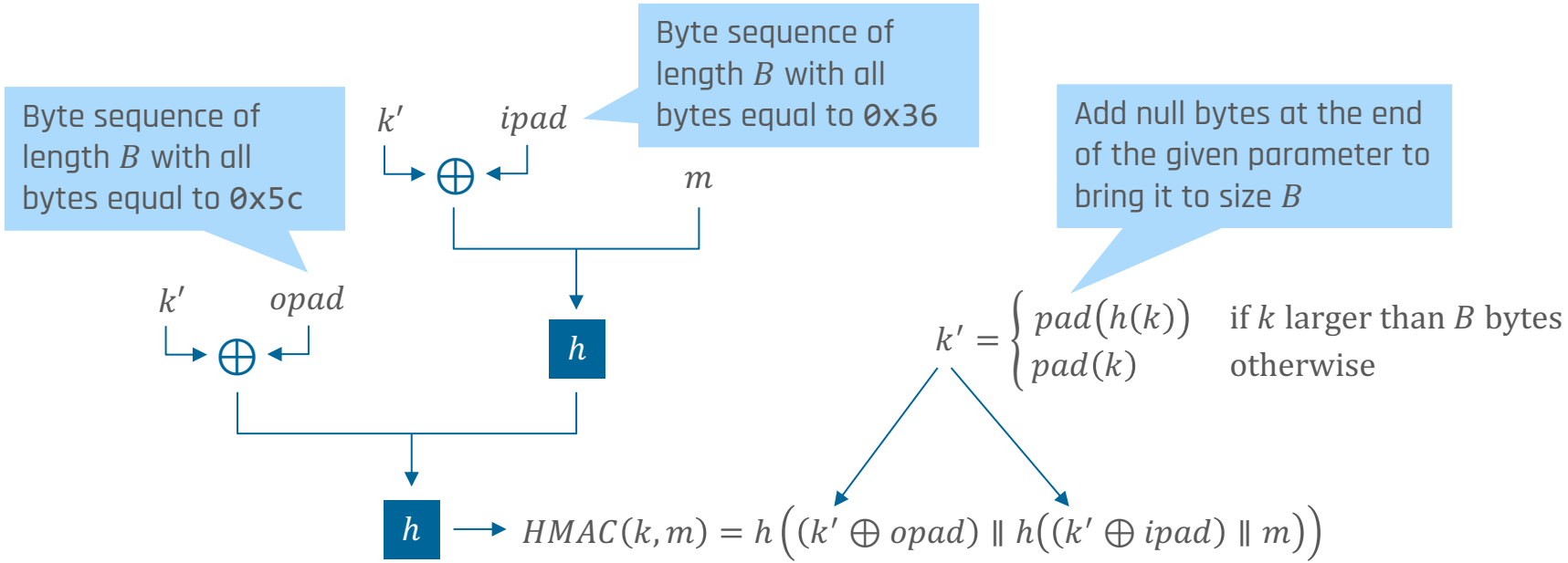
# Encrypted CBC-MAC




- Variant of CBC-MAC in which the encryption of the last block (MAC of the CBC-MAC scheme) is **encrypted** under a different key
  - Can be securely used also with **variable-length messages**

# HMAC

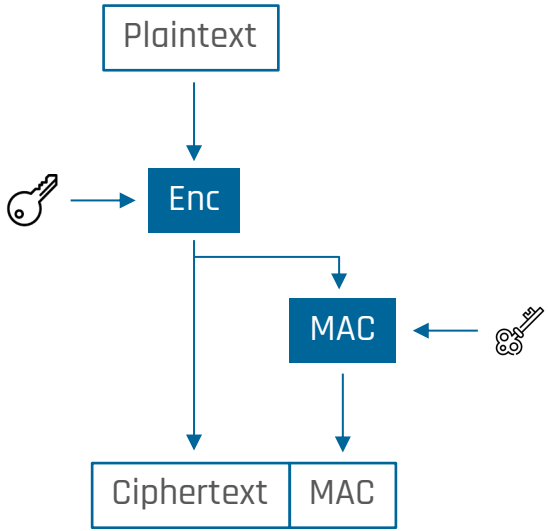
- MAC based on a **hash function** $h$ (with block size $B$) and a **secret key** $k$



# Authenticated Encryption

# Encrypt-then-MAC

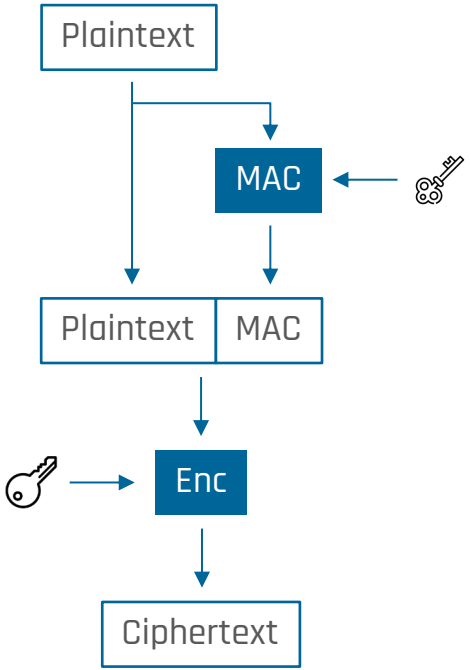
- First the plaintext is **encrypted**, then a MAC is computed over the **ciphertext**
  - Used in the IPSec protocol
- Approach is secure for **any combination** of secure encryption and MAC algorithms
  - The keys used for encryption and MAC computation must be **different**



# MAC-then-Encrypt

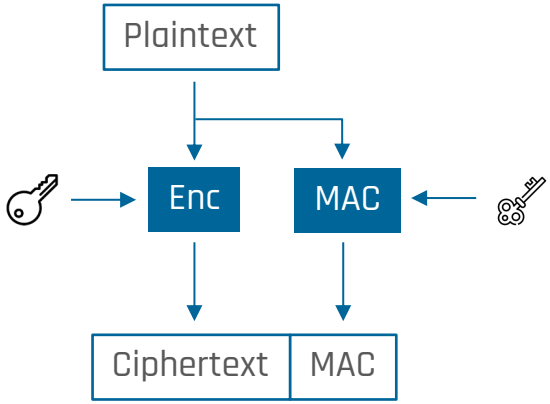
- The MAC is computed over the **plaintext**, then plaintext and MAC are **encrypted together**
  - Used in SSL / TLS (until version 1.2)

- Not secure for **all possible combinations** of secure ciphers and MACs
  - Secure with block ciphers in **CBC mode** or with **stream ciphers**
  - Cause of catastrophic **padding-oracle attacks** in TLS (POODLE and Lucky13)



# Encrypt-and-MAC

- The plaintext is **encrypted**, the MAC is also computed over the **plaintext**
  - Used in the SSH protocol
- Not secure for **all possible combinations** of secure ciphers and MACs
  - Differently from MAC-then-Encrypt, also **not secure** with standard stream ciphers
  - Still, possible to get it right!



# Authenticated Encryption Modes

- The best way to achieve authenticated encryption nowadays is to use standardised, **authenticated encryption modes**
  - Provide simultaneously confidentiality, integrity and authenticity
  - Such modes include GCM, CCM, CWC, OCB, EAX

- Starting from version 1.3, TLS supports **exclusively** cipher suites making use of authenticated encryption modes