Cryptography Background

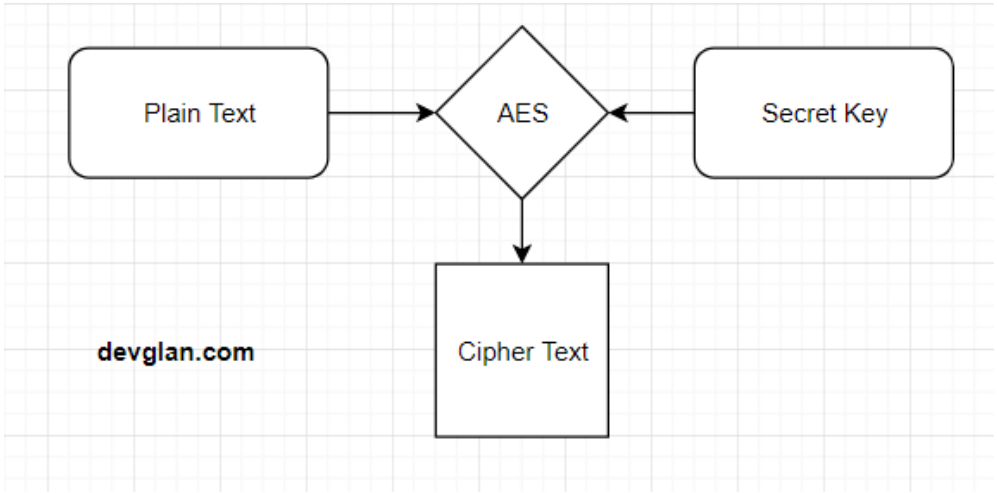
Επικ. Καθηγητής Χριστόφορος Νταντογιάν

dadoyan@ionio.gr

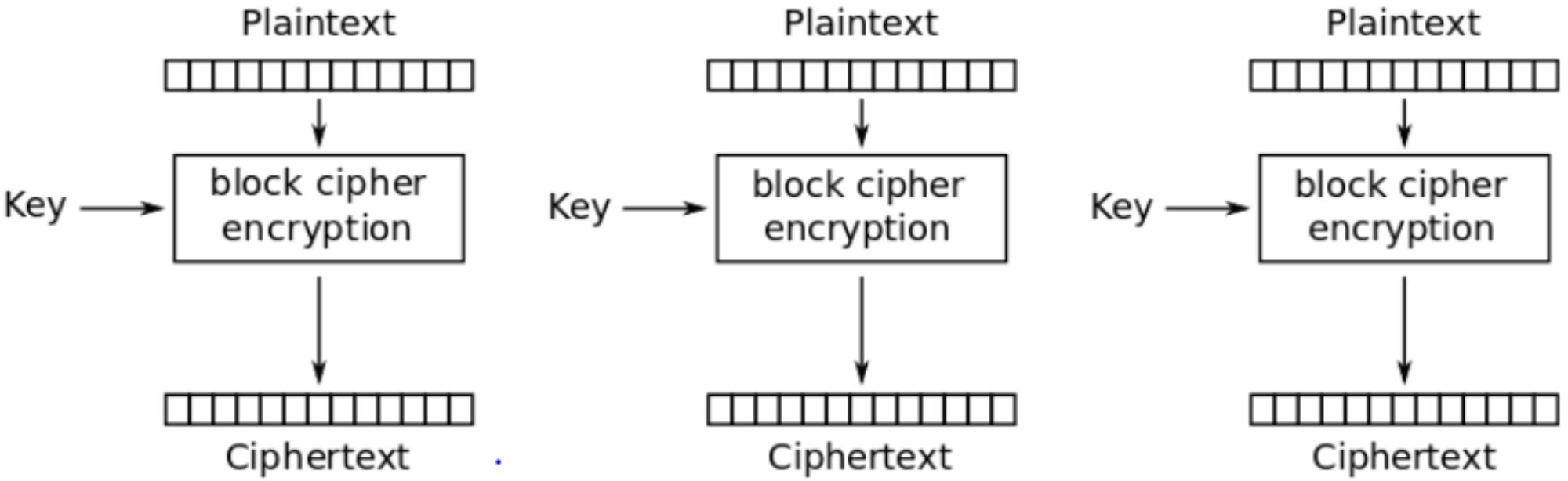
Block Ciphers

Advanced Encryption Standard (AES)

- AES has been adopted by the U.S. government and is now used worldwide.
- For AES has a block size of 128 bits, and three different key lengths: 128, 192 and 256 bits.
- How do we encrypt messages that are longer than the block size?
  - Use of modes (not only for AES but for any symmetric encryption)

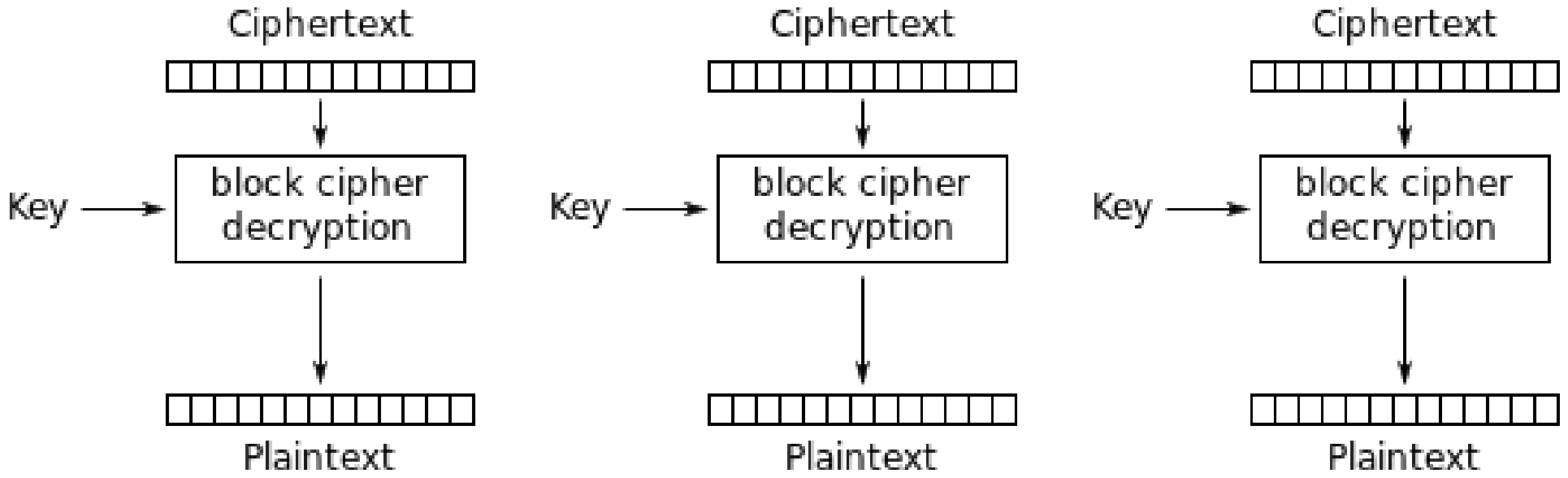


ECB mode encryption



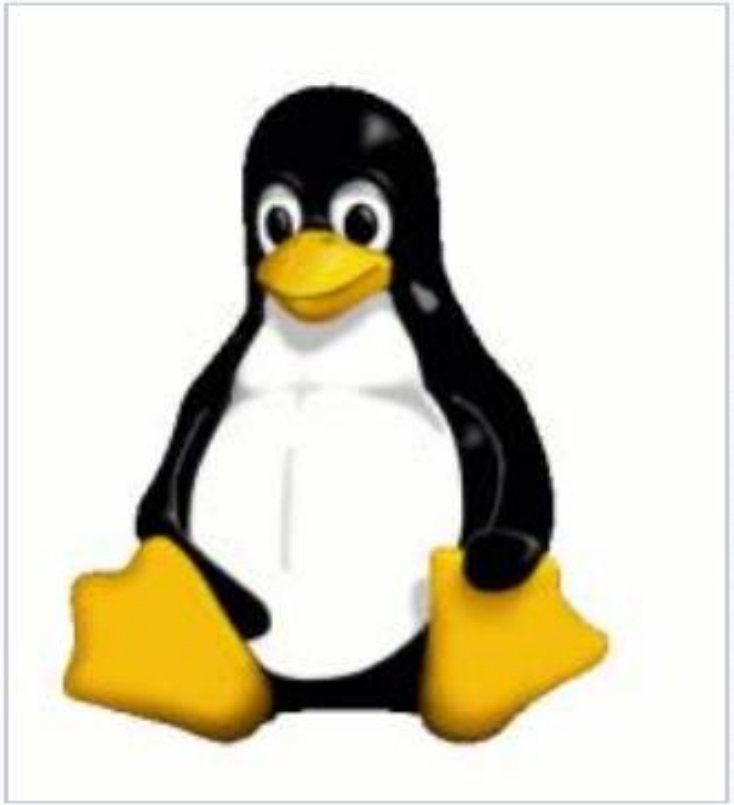
Electronic Codebook (ECB) mode encryption

ECB Decryption

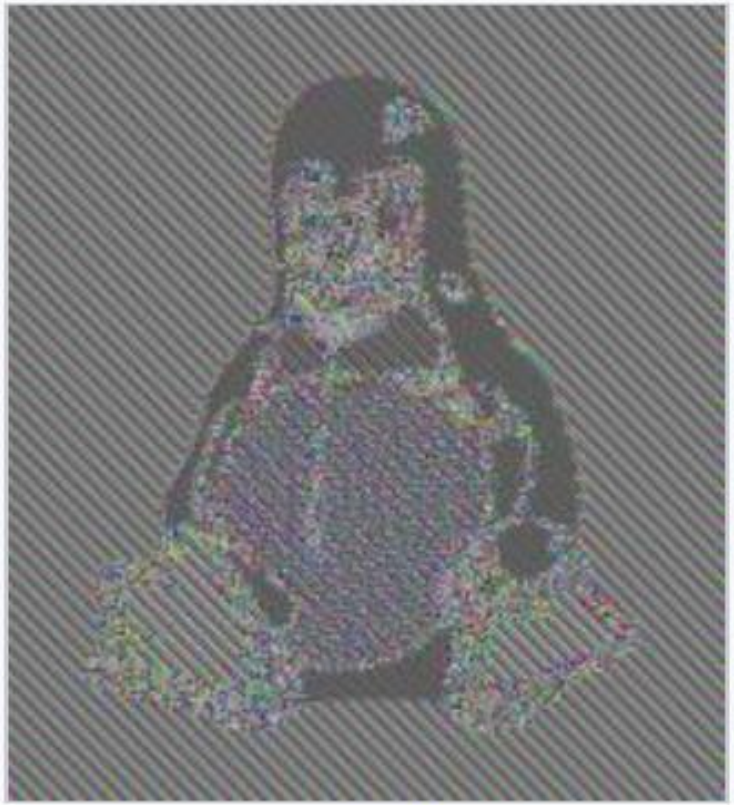


Electronic Codebook (ECB) mode decryption

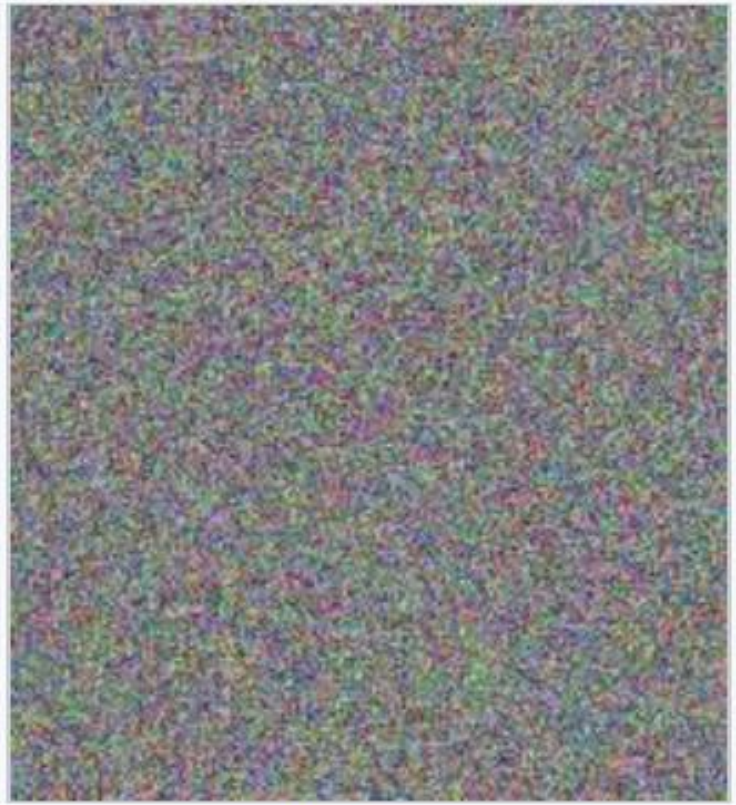
Insecure ECB



Original image



Encrypted using ECB mode



Modes other than ECB result in pseudo-randomness

Two properties for a secure cipher

Confusion

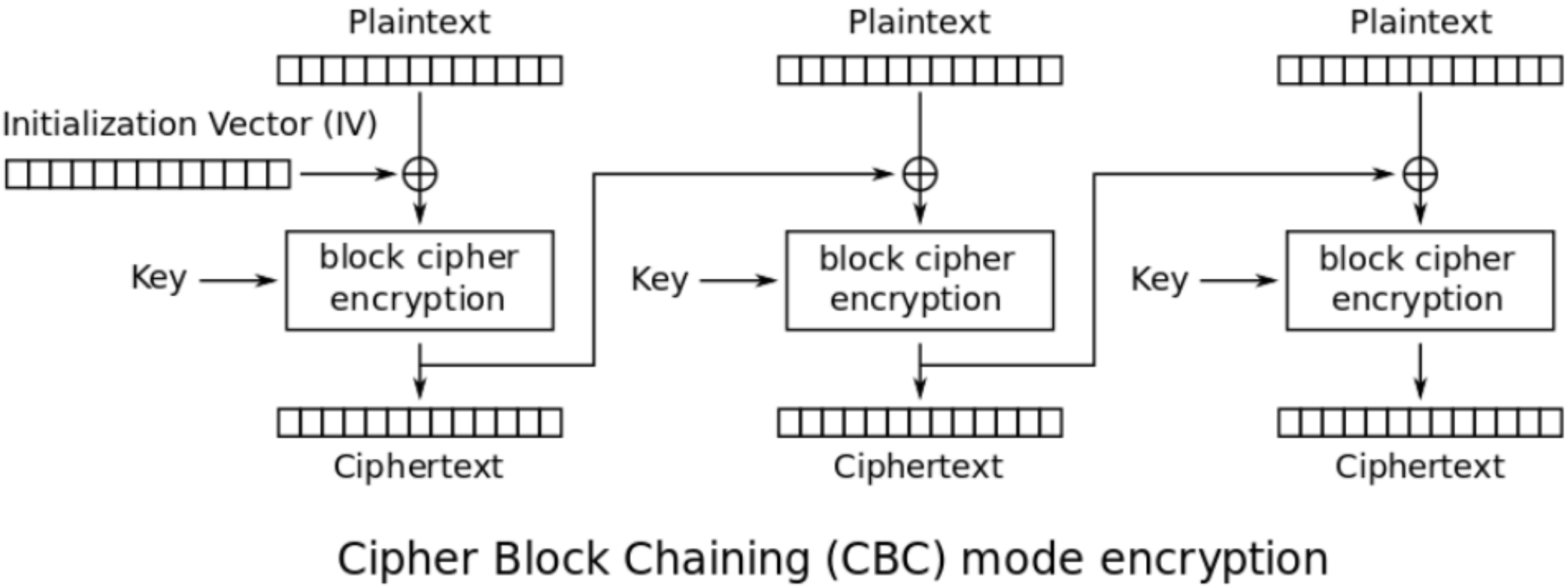
- The property of confusion hides the relationship between the ciphertext and the key.
- Confusion means that each binary digit (bit) of the ciphertext should depend on several parts of the key, obscuring the connections between the two.
- This property makes it difficult to find the key from the ciphertext and if a single bit in a key is changed, most or all the bits in the ciphertext will be affected.

Two properties for a secure cipher

Diffusion

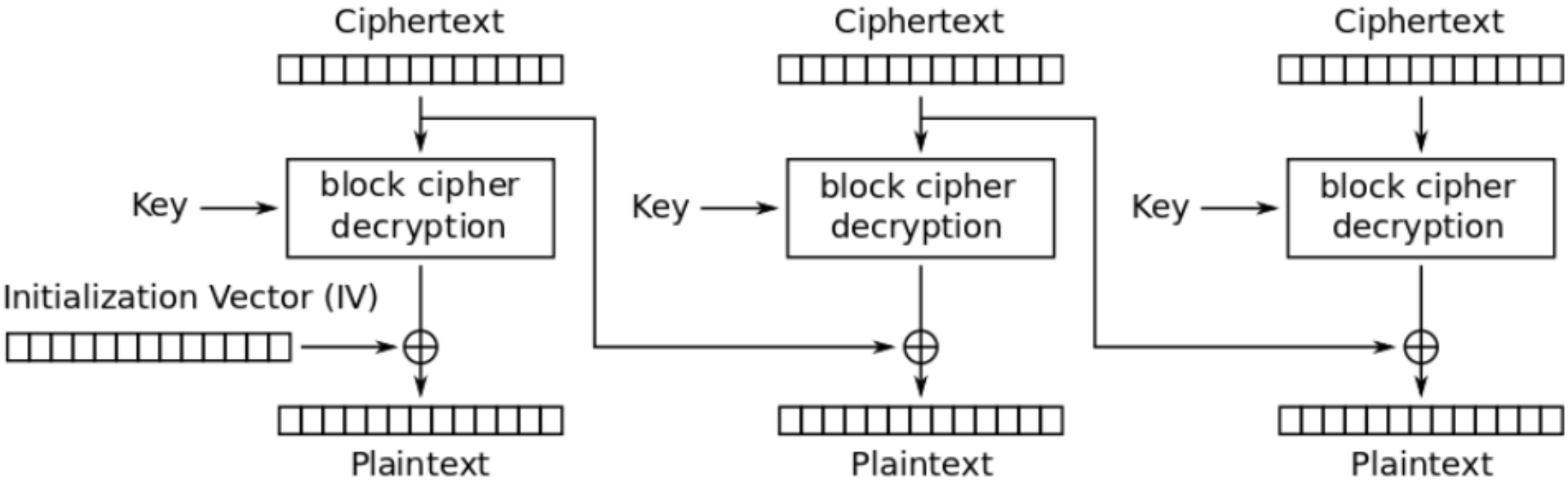
- The idea of diffusion is to hide the relationship between the ciphertext and the plain text.
- Diffusion means that if we change a single bit of the plaintext, then (statistically) half of the bits in the ciphertext should change, and similarly, if we change one bit of the ciphertext, then approximately one half of the plaintext bits should change.

CBC mode encryption



Cipher Block Chaining (CBC) mode encryption

CBC mode decryption

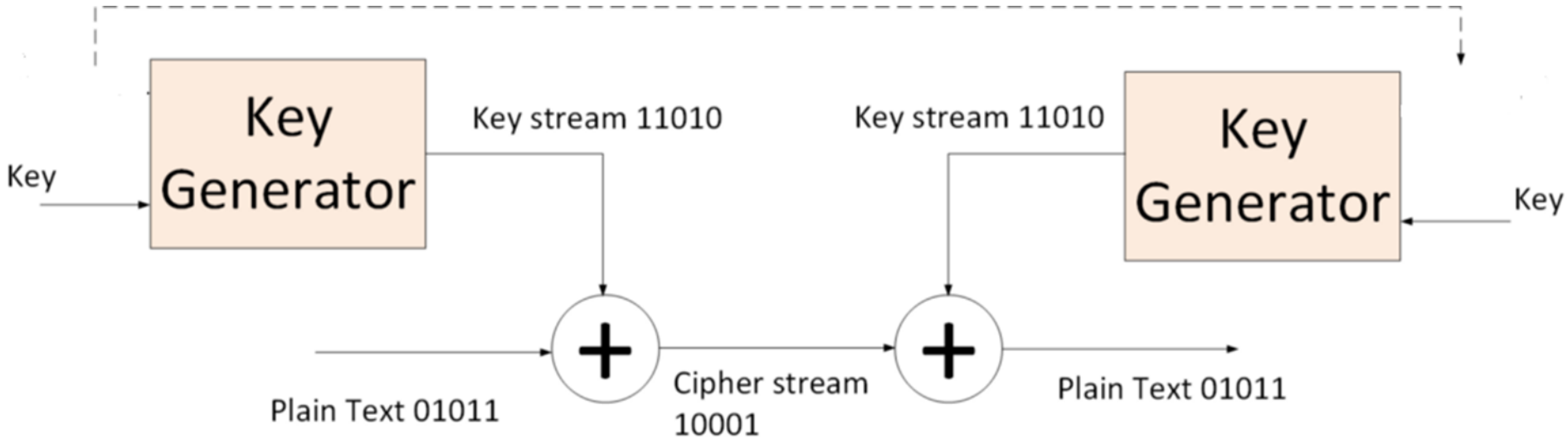


Cipher Block Chaining (CBC) mode decryption

Stream Ciphers

Stream Ciphers

- Generate a keystream as long as the plaintext



Stream ciphers

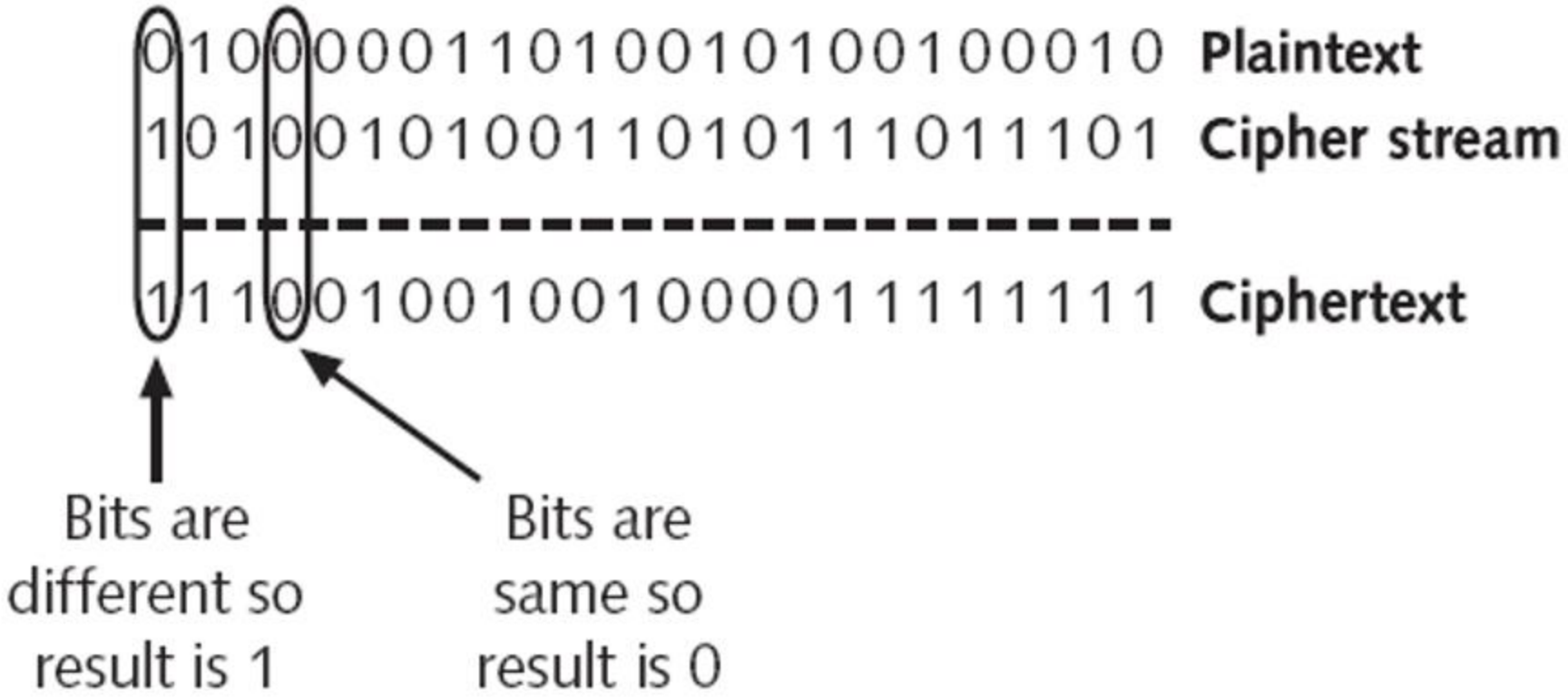


Figure 11-10 Creating ciphertext with XOR

Stream ciphers

- Simpler hardware implementation!
- Speed
- Used when the size of the plaintext comes in quantities of unknowable length → No padding
- Error do not propagate
- RC4, Chacha, Salsa20, Acorn

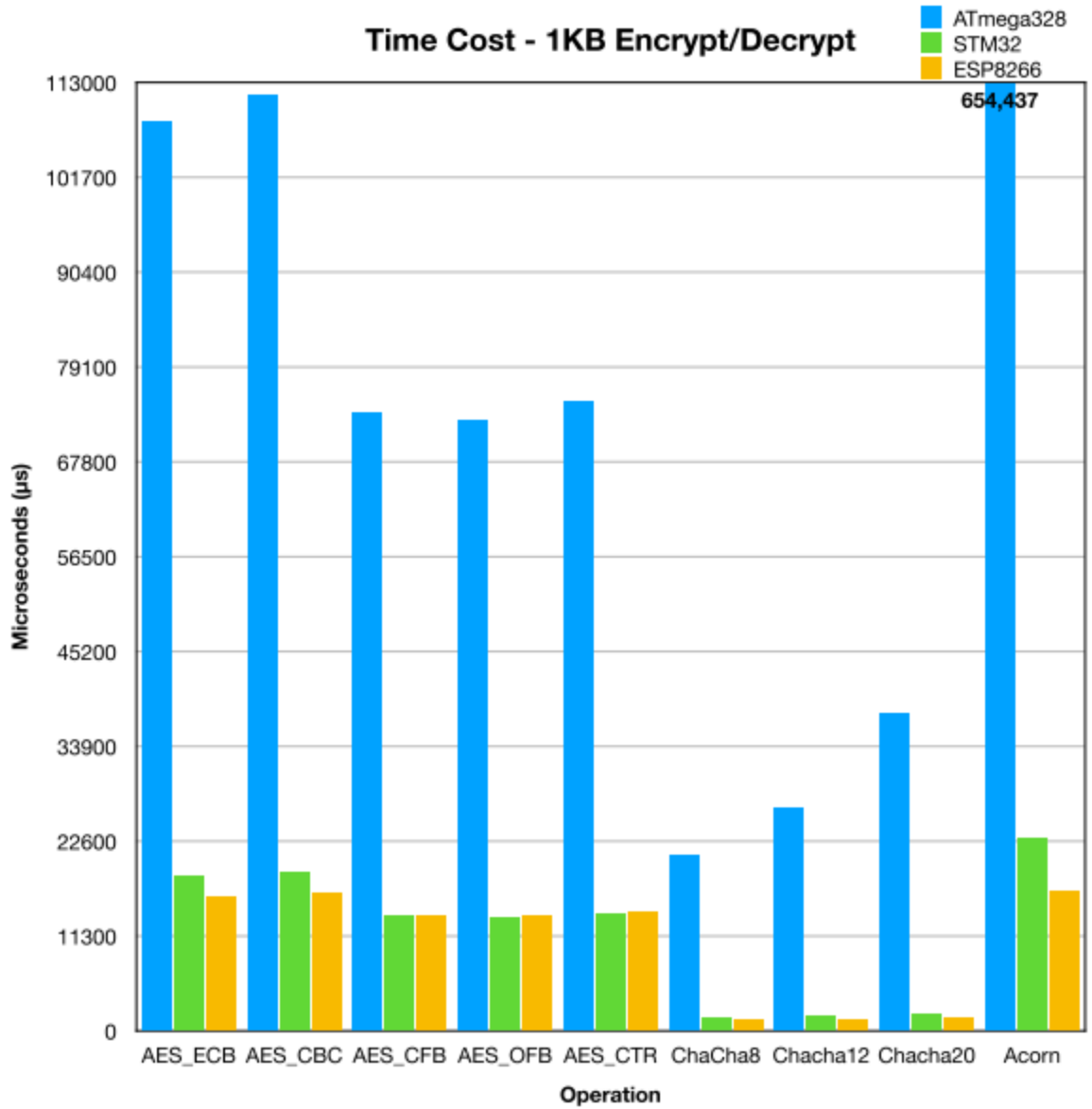


FIGURE 13. Time cost comparison of encryption and decryption for 1KB of data.

Hash functions

- A **hash function** is any function that can be used to map data of arbitrary size to fixed-size values.
  - must be deterministic

- MD5 → 128 bits

- SHA1→ 160 bits

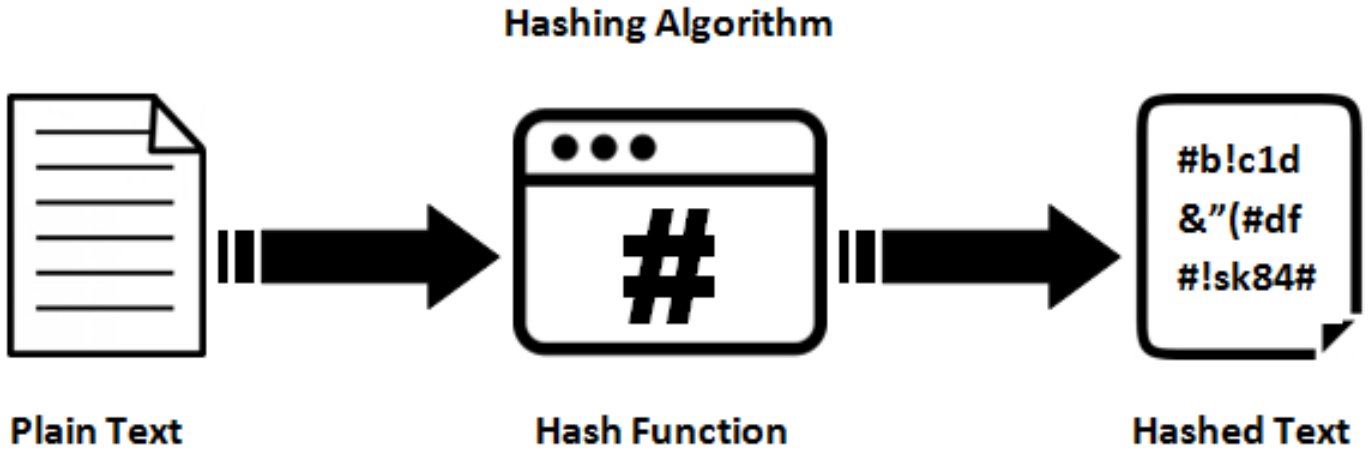
- SHA256 → 256 bits

Hashing algorithms

- MD5 (not recommended)
- SHA-1 (not recommended)
- SHA-2 (multiple versions for different size outputs, 348, 256, 512, etc),
- SHA-3,
- SHA256
- Bcrypt
- Scrypt
- Argon2

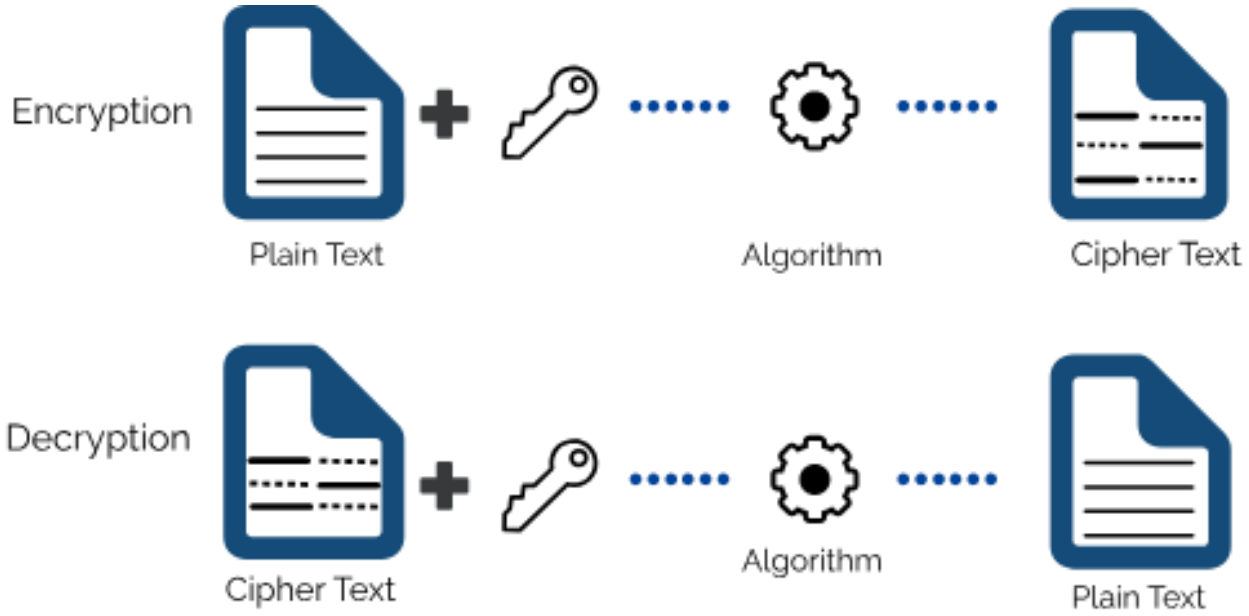
Hashing

- It’s a one way function!!!!
- No keys



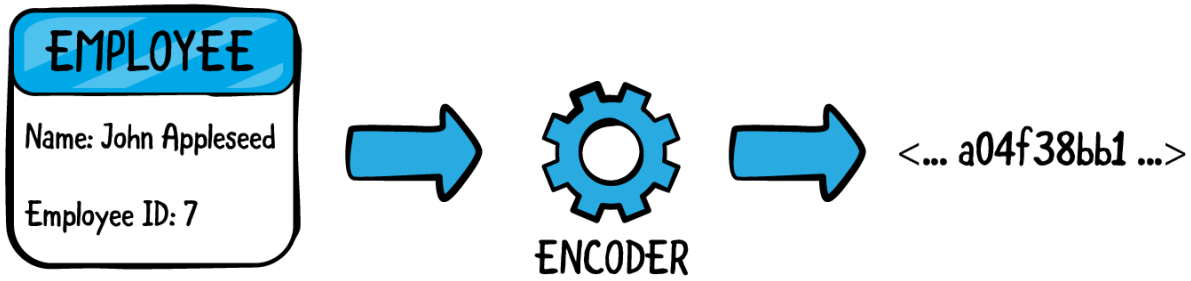
Encryption

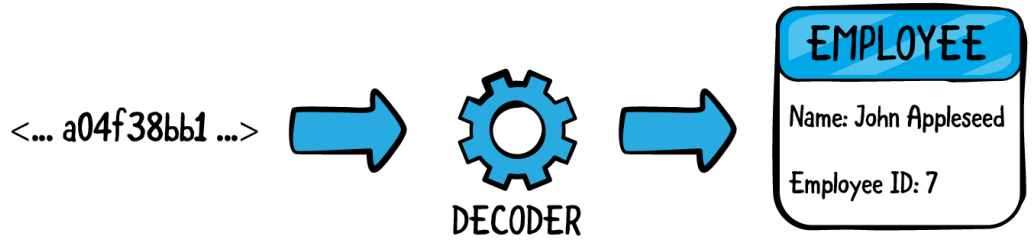
- Its not a one way function! You can reverse (decryption).
- Use of keys



Encoding

- Its not a one way function (you can reverse it)
- No keys
- No security purposes
- ASCII, UNICODE,
- base64
- Morse code

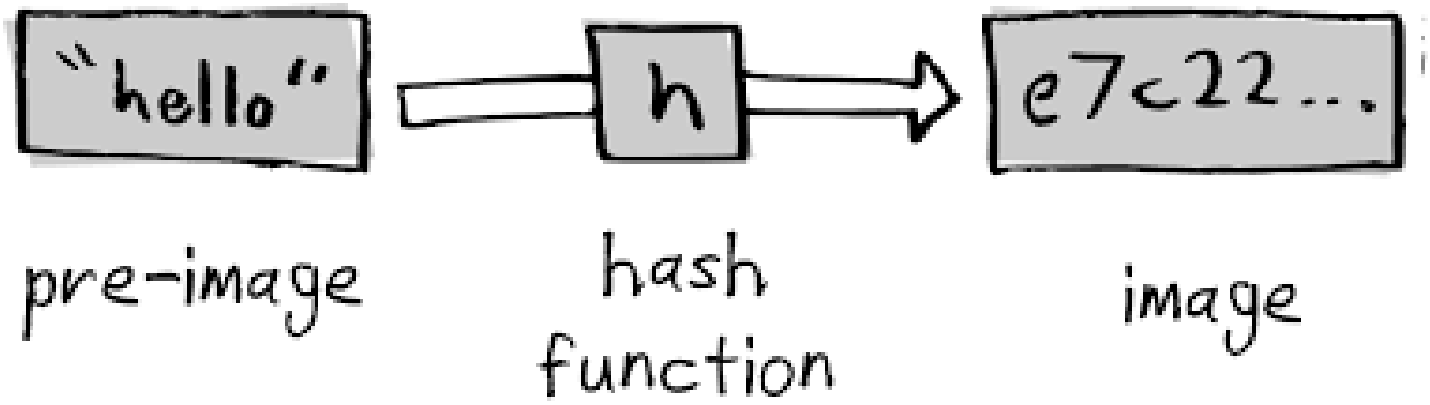




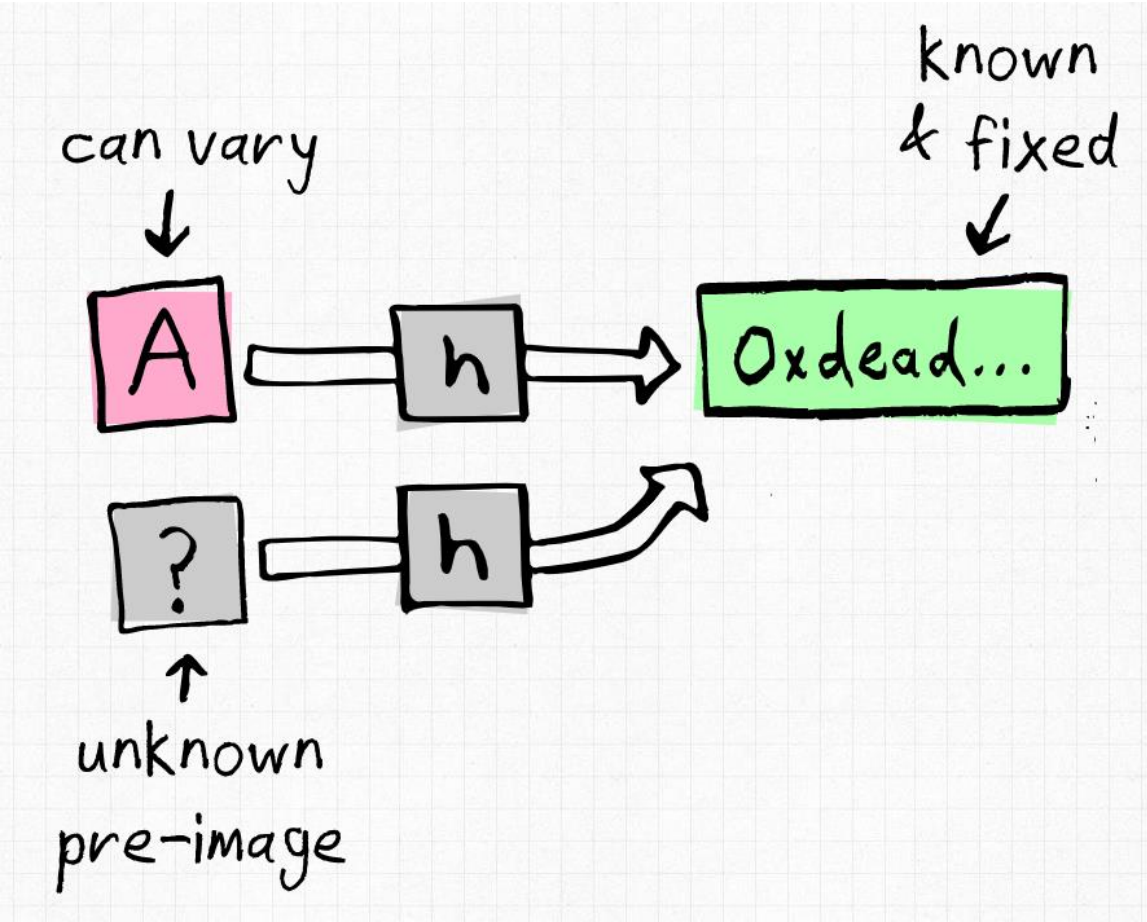
Properties

1. A small change to a message should change the hash value so extensively that the new hash value appears uncorrelated with the old hash value (**avalanche effect**)
2. Assuming H(x)=y, then given y, it is impossible to find a x such as H(x)=y (first preimage, also known as one-way functions)
3. Given x and H(x)=y, is impossible to find another x' such as H(x')=y (second preimage)
4. it is infeasible to find two different messages x, x' with the same hash value, that is H(x)=y=H(x')

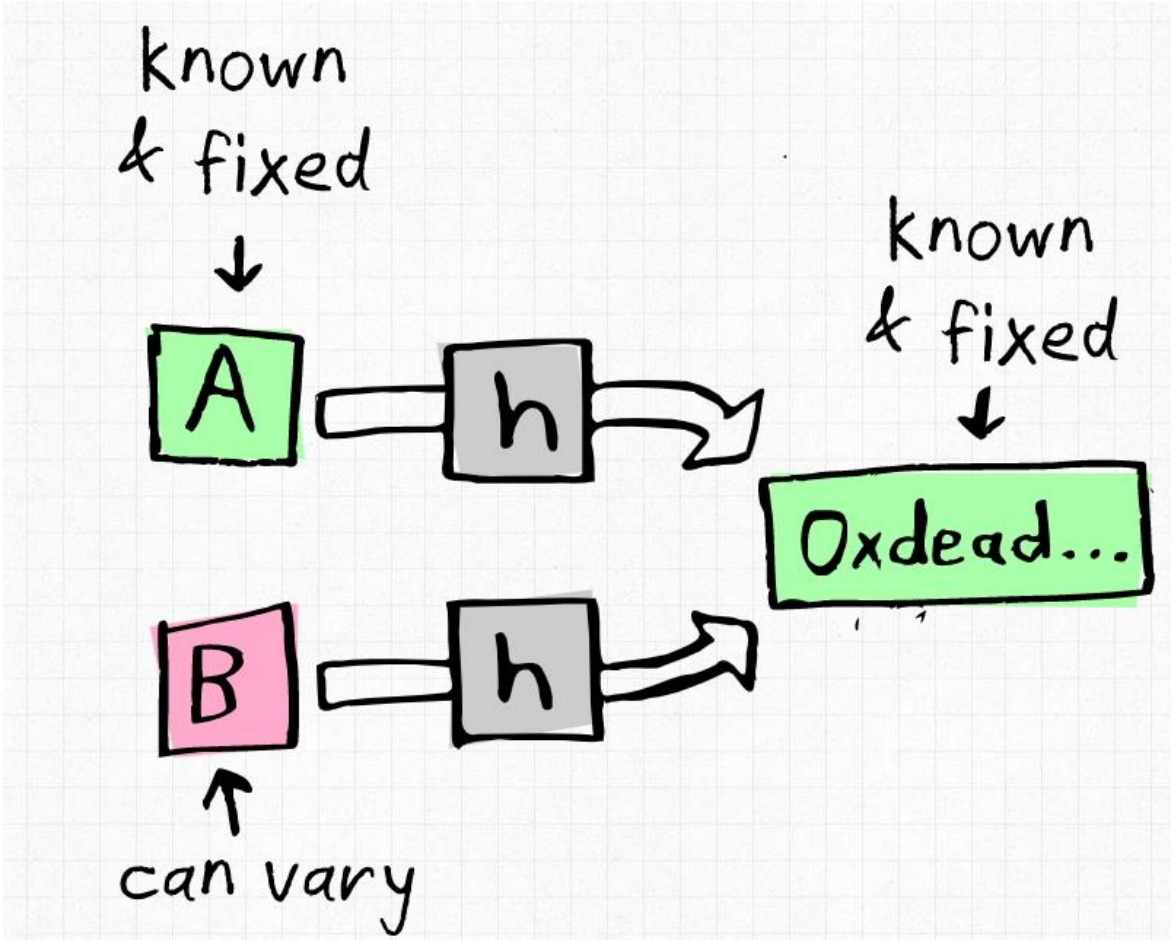
A hash function



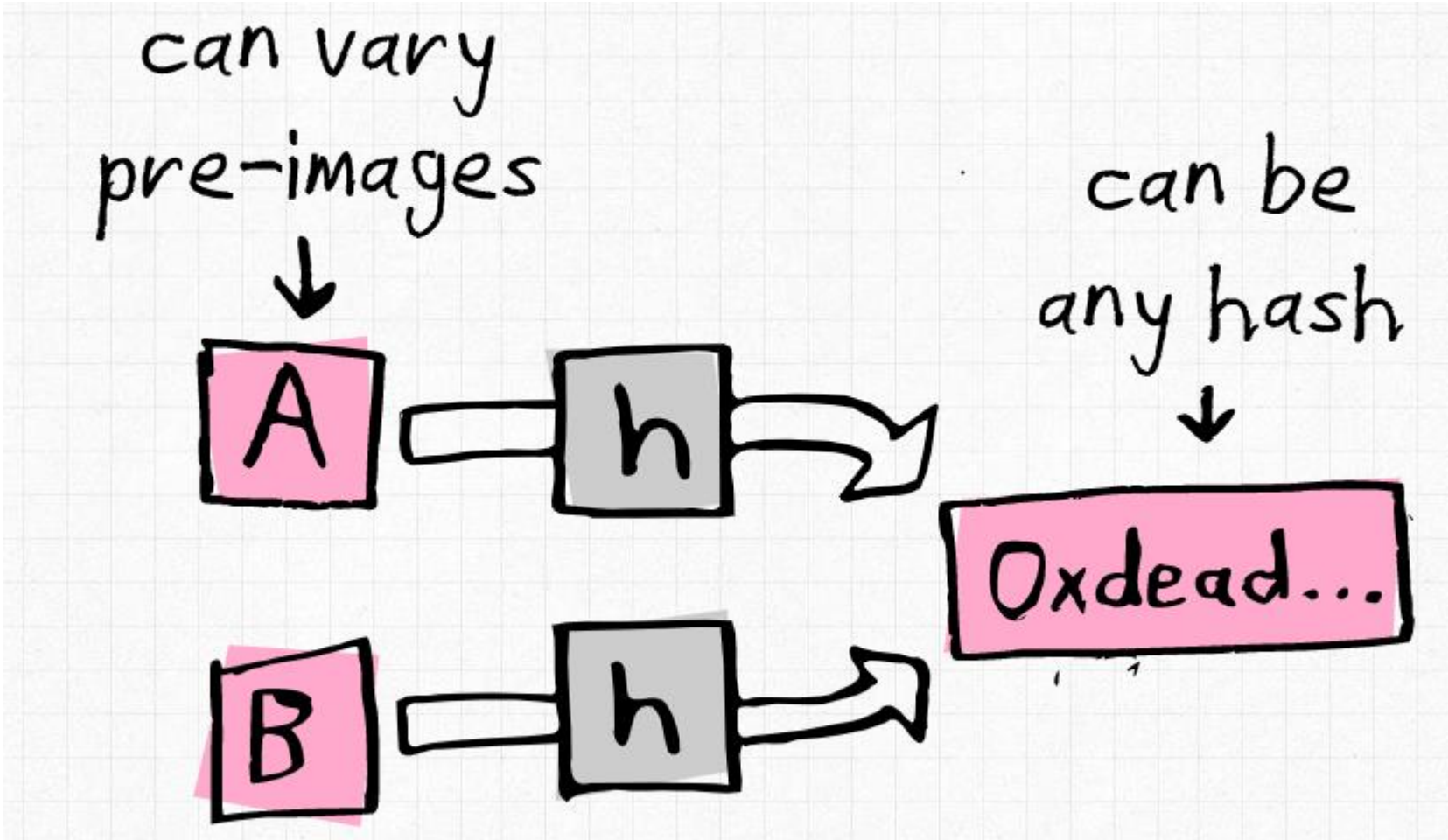
First preimage attack



Second preimage attack

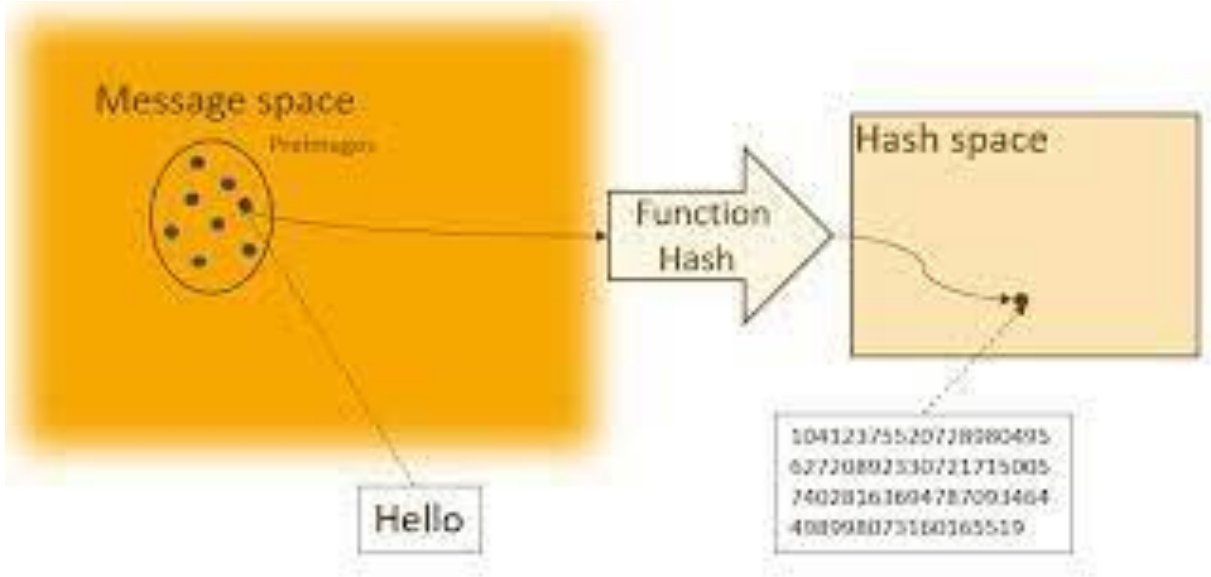


Collision attacks



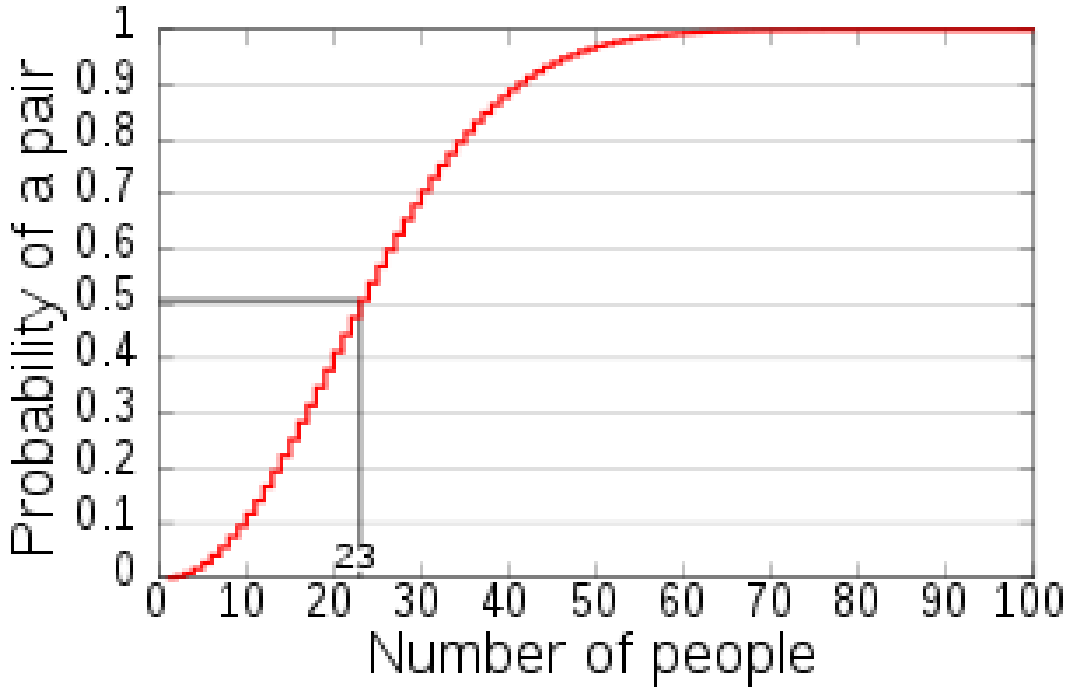
Collision Probability

- If the output of a hash function is n bits, then how many different inputs we have to hash in order to have a collision with probability 100%?



Birthday paradox

- Birthday paradox concerns the probability that, in a set of n randomly chosen people, some pair of them will have the same birthday.



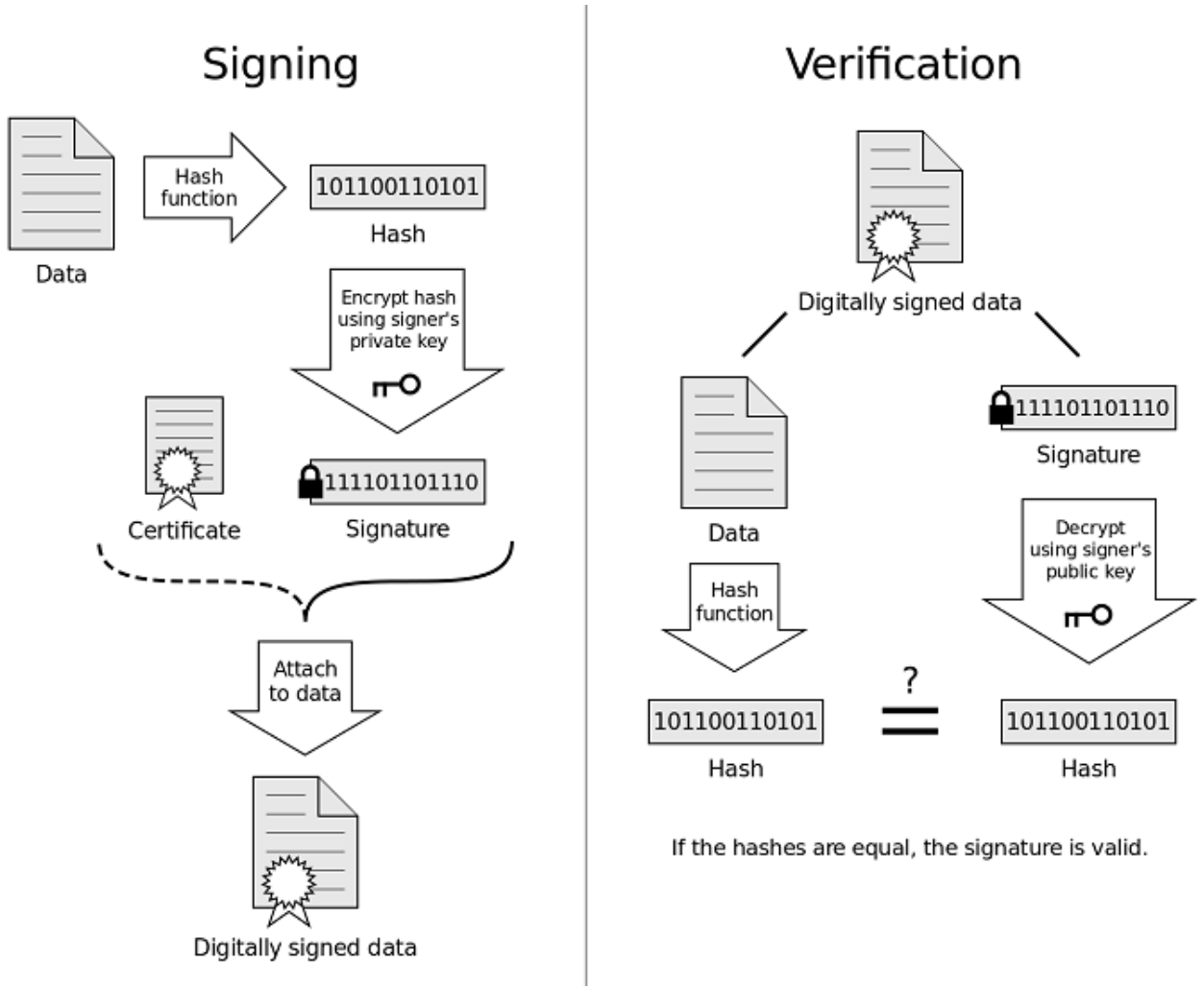
Birthday attack

- 2^n+1 is the classical worst case complexity of preimage (or second preimage) resistance security → Brute force attack (assuming no collusions) → $2^n/2 = 2^{n-1}$ on average

- With a birthday attack, it is possible to find a **collision** of a hash function in roughly $\sqrt{2^n} = 2^{n/2}$

- Demo

Certificates and collisions

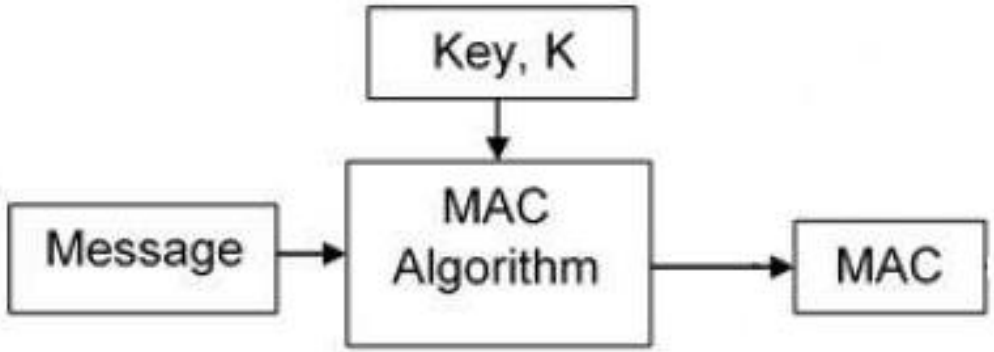


MAC

Message Authentication Code (MAC)

- Also known as tag or MIC (Message Integrity Code)

- A short piece of information used to authenticate a message—in other words, to confirm that the message came from the stated sender (its authenticity) and has not been changed (integrity)



How to implement a MAC?

- MAC algorithms can be constructed from other cryptographic primitives, like cryptographic hash functions → HMAC

- Hash functions takes 1 input, how can we modify them to take 2 inputs, that is the message m and the key k?

- Append m to k, thus we have one input m+k
  - Message 43432 and key 545453, then h(43432545453) is a hmac!

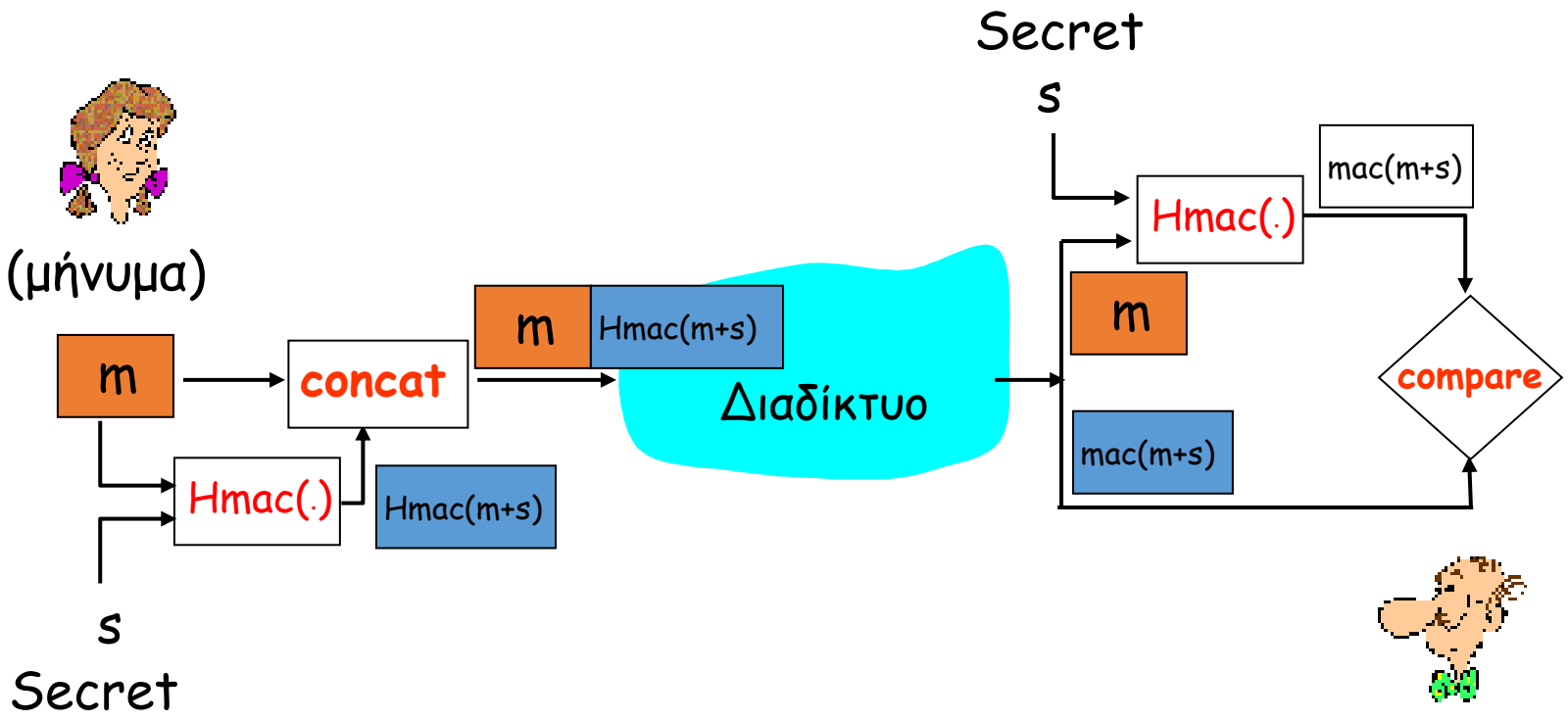
HMAC

- HMAC-MD5
- HMAC-SHA1
- HMAC-SHA256

Q: What is the key size?

A: It should be equal to the output of the hash function. Thus, HMAC-MD5 should use 128 bits while HMAC-SHA1 should use 160 bits

MAC for integrity and message authentication



Caution !!

- The previous figure shows how to generate a MAC
- However, the figure has an erroneous description of how a HMAC can be implemented
- Question: How to implement a HMAC as a develop?

Hash Message Authentication Codes (HMAC)

MAC key *K*, plaintext *M*

MAC=H(*K* || M)? MAC=H(M || *K*)?

There are weaknesses with *all* of the above (**length extension attacks**).

HMAC = H(*K* || H(*K* || M))

Basic Lesson in cryptography

- NEVER IMPLEMENT YOUR OWN CRYPTOGRAPHY!
- USE READY TO USE FUNCTIONS AND LIBRARIES FROM YOUR PROGRAMMING LANGUAGE
- Cryptographic schemes are usually broken through implementation not the underlying mathematical assumptions

Any other way to Implement MAC?

- Hash functions are not the only way to implement a MAC

- We can use encryption algorithms to convert them to MAC!

1. CBC-MAC
2. Poly1305

CBC-MAC

- Note that similar to HMACs, CBC-MAC require two parameters, a key and a text to produce its MAC

- CBC-MAC is applied **in WPA2 standard (CCMP protocol)** for generating the MAC of the messages (we will revisit this)

- Other drawbacks?
  - https://blog.cryptographyengineering.com/2013/02/15/why-i-hate-cbc-mac/

Crypto Hygiene

- Q: Do I need to use different keys for encryption and MAC?
- A: The answer is yes! It's always a good idea to use separate keys for separate functions.

- Q: Do client and server need to use different symmetric keys (i.e., one key for client to server encryption and one key for server to client encryption)?
- A: No, but some protocols do this (e.g., SSL/TLS)

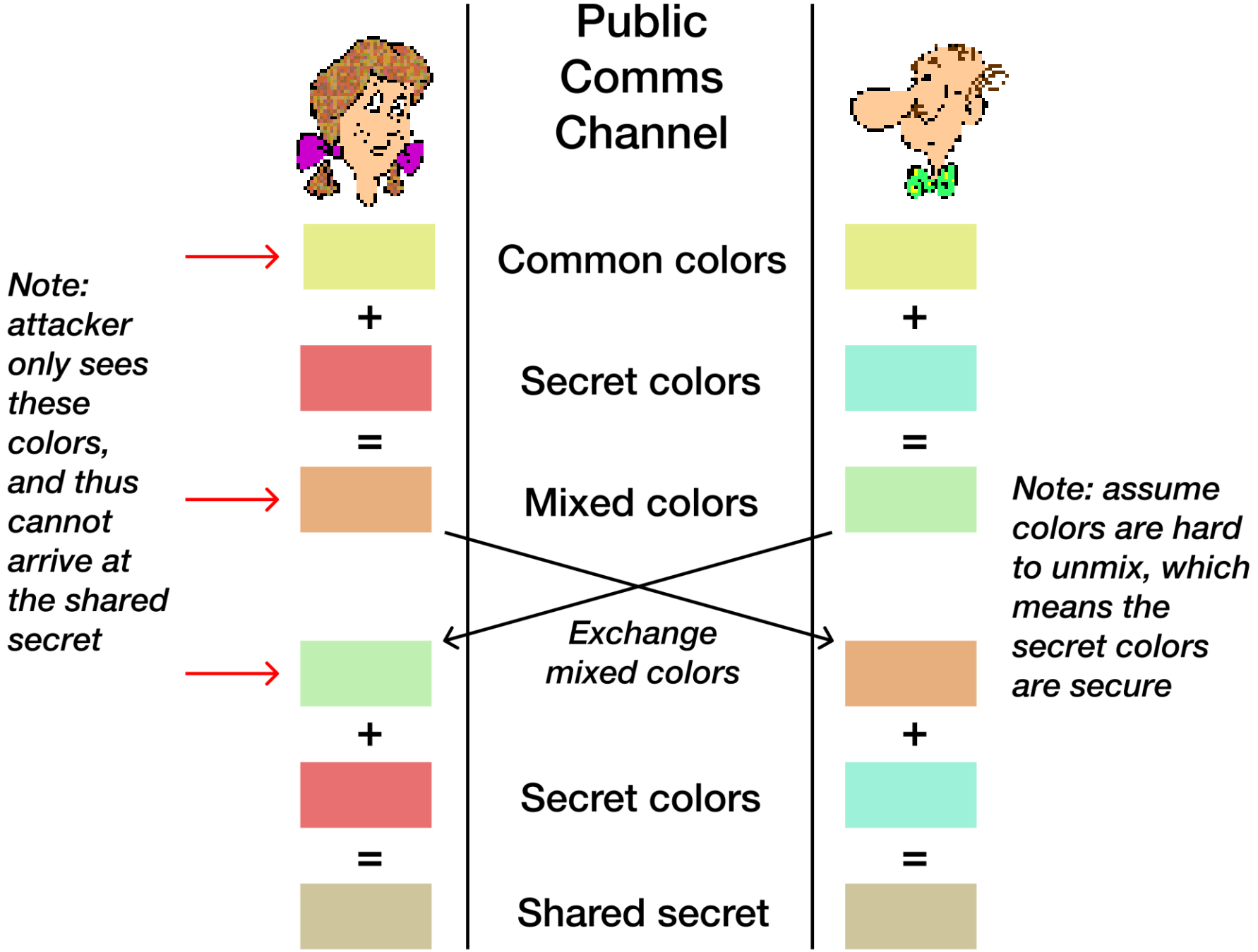
KEY AGREEMENT WITH DIFFIE-HELLMAN

Key agreement between Alice and Bob

1. Physical/Manually

2. Public key cryptography with certificates

3. **Public key cryptography with Diffie-Hellman key agreement (magic protocol)**


Public
Comms
Channel
Common colors
Secret colors
Mixed colors
Exchange
mixed colors
Secret colors
Shared secret
Note:
attacker
only sees
these
colors,
and thus
cannot
arrive at
the shared
secret
Note: assume
colors are hard
to unmix, which
means the
secret colors
are secure

1. Alice & Bob agree upon 2 large prime number n and g. These numbers are public
2. Alice chooses a large random number x (private) and calculates $A= g^x \bmod n$ and sends it to Bob
3. Bob chooses a large random number y (private) and calculates $B=g^y \bmod n$ and sends it to Alice
4. Alice computes $K_A=B^x \bmod n= g^{xy} \bmod n$
5. Bob computes $K_B= A^y \bmod n = g^{xy} \bmod$



n=11, g=7

x=3

y=6

$A=g^x \bmod n=7^3 \bmod 11=2$

$B=g^y \bmod n=7^6 \bmod 11=4$

B=4

A=2

$K_A=B^x \bmod n=4^3 \bmod 11=9$

$K_B=A^y \bmod n=2^6 \bmod 11=9$

$K_A=K_B$

K1=K2

Man in the middle attack in Diffie-Hellman







n, g

Generate x

Compute $A=g^x \bmod n$

$A=g^x \bmod n$

n, g

Generate z

Compute $M=g^z \bmod n$

$M=g^z \bmod n$

n, g

Generate y

Compute $B=g^y \bmod n$

$B=g^y \bmod n$

$K_{ZB}=B^z \bmod n=g^{yz} \bmod n$

$K_{ZB}=M^y \bmod n=g^{yz} \bmod n$

$M=g^z \bmod n$

$K_{ZA}=M^x \bmod n=g^{xz} \bmod n$

$K_{ZA}=A^z \bmod n=g^{xz} \bmod n$

K_{ZA} K_{ZA}

K_{ZB} K_{ZB}

Man-in-the-middle attack

- The above attack is called Active man-in-the-middle attack
- There is also passive man-in-the-middle
- This is one of the most powerful attacks in network protocols, it can occur and depending on the protocol several attacks can be achieved.

How can we secure Diffie-Hellman?

- Identify the problem: Which security principle is broken?
- Answer: Integrity
- Propose two different solutions to eliminate the man in the middle attack describe previously!
- Answer: 1) MAC or 2) digital signatures
- Which one is used in practice and why?
- Answer: Digital signatures in TLS 1.3 !!! MAC require a symmetrical key (otherwise there is no reason to run Diffie Hellman)