# Asymmetric Cryptography

SIO

deti universidade de aveiro
departamento de eletrónica,
telecomunicações e informática

João Paulo Barraca

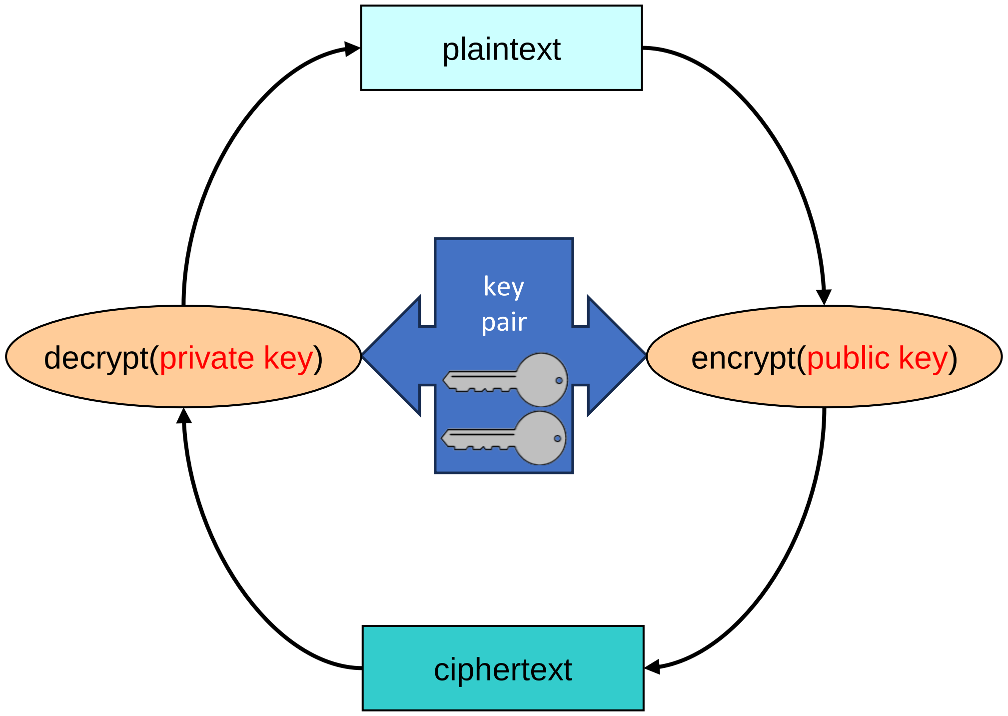
# Asymmetric (Block) Ciphers

- Use key pairs
  - **One private key**: personal, not transmittable
  - **One public key**: available to all

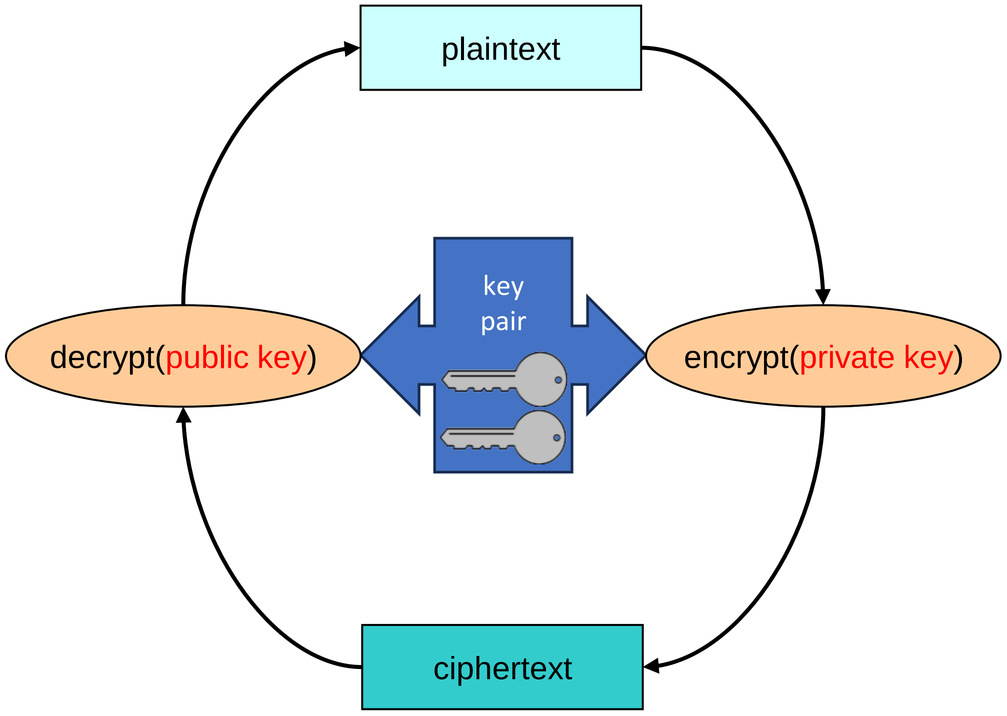
- Allow
  - Confidentiality without any previous exchange of secrets
  - Authentication
    - Of contents (data integrity)
    - Of the data origin (source authentication, or digital signature)

# Operations of an asymmetric cipher

## Confidentiality




## Authenticity

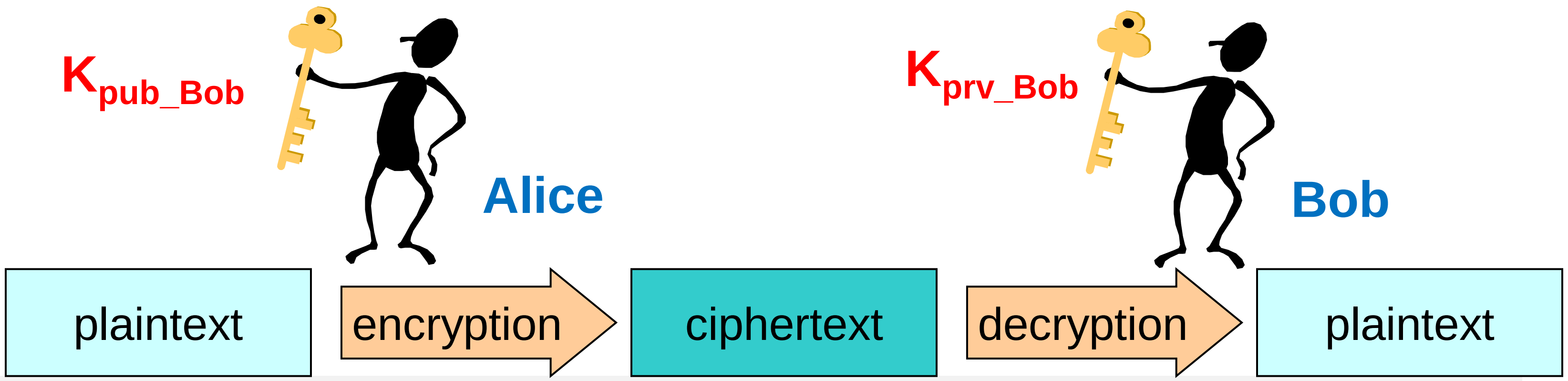


# Use cases: confidential communication

- Secure communication with a target (Bob)
  - Alice encrypts plaintext **P** with Bob's public key $\mathbf{K_{pub\_Bob}}$

    **Alice: $C = \{P\}K_{pub\_Bob}$**
  - Bob decrypts cyphertext **C** with his private key $\mathbf{K_{prv\_Bob}}$

    **Bob: $P' = \{C\}K_{prv\_Bob}$**
  - **P'** should be equal to **P** (requires checking using integrity control)
  - $\mathbf{K_{pub\_Bob}}$ needs to be known by Alice



# Use cases: authenticated communication

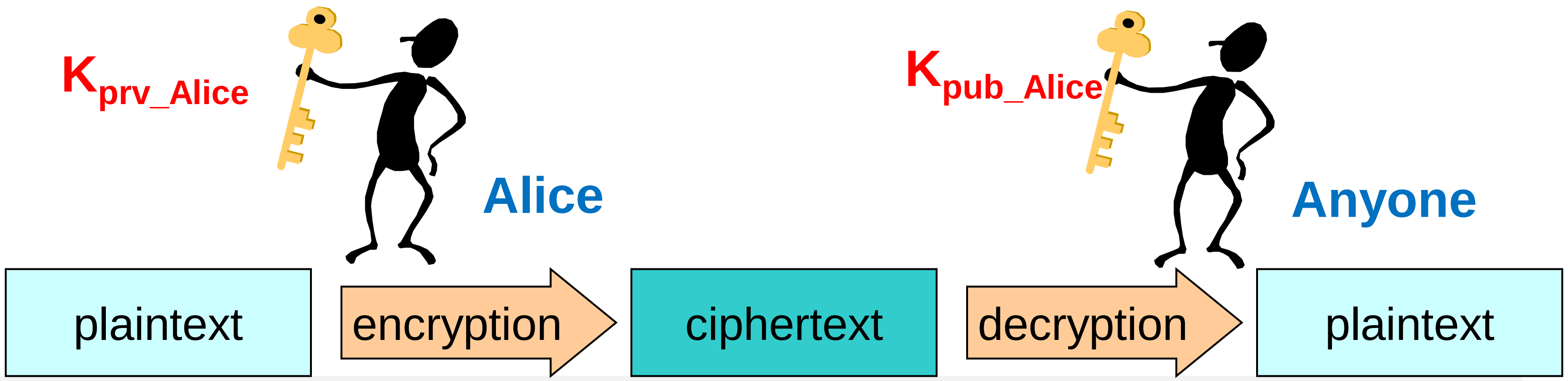
- Authenticate the communication from Alice
  - Alice encrypts plaintext **P** with her private key **$K_{prv\_Alice}$**

    **Alice: $C = \{P\}K_{prv\_Alice}$**

  - Anyone can decrypt cyphertext **C** with Alices' Public key **$K_{pub\_Alice}$**

    **Anyone: $P' = \{C\}K_{pub\_Alice}$**

  - If **P'** = **P**, then **C** is Alice's signature of **P**
  - **$K_{pub\_Alice}$** needs to be known by the message verifiers



# Asymmetric ciphers

## Issues

- Advantages
  - They are a fundamental authentication mechanism
  - They allow to explore features that are not possible with asymmetric ciphers
- Disadvantages
  - Performance: 2 or 3 orders of magnitude over AES
  - Very inefficient and memory consuming: Large keys
- Problems
  - Trustworthy distribution of public keys: how to know if the public key is the correct one?
  - Lifetime of key pairs: How to make sure that we can deal with lost/deprecated/leaked keys?

# Asymmetric ciphers

## Overview

- Approaches: complex mathematic problems
  - **Discrete logarithms** of large numbers
  - **Integer factorization** of large numbers
- Most common algorithms
  - RSA
  - ElGamal
  - Elliptic curves (ECC)
- Other techniques with asymmetric key pairs
  - Diffie-Hellman (key agreement)

# RSA

## Rivest, Shamir, Adelman, 1978

- Keys: Private: (d, n) Public: (e, n)
- Public key encryption (confidentiality) of **P**
  - $C = P^e \bmod n$
  - $P = C^d \bmod n$
- Private key encryption (authenticity) of **P**
  - $C = P^d \bmod n$
  - $P = C^e \bmod n$

**P, C are numbers!**
**Message is converted to/from numbers**

$0 \leq P, C < n$

# RSA

## Rivest, Shamir, Adelman, 1978

- Computational complexity: **Discrete logarithm** and **Integer factoring**

- Key selection
  - Large **n** (hundreds or thousands of bits)
  - **n = p × q** with **p** and **q** being large (secret) prime numbers
  - Chose an **e** co-prime with **(p-1) × (q-1)**
  - Compute **d** such that **e × d ≡ 1 (mod (p-1) × (q-1))**
  - Discard **p** and **q**
  - The value of **d** cannot be computed out of **e** and **n**
    - Only from **p** and **q**

coprime → gcd(a, b) = 1

× → multiplication

mod → modulo operation

≡ → modular congruence

a ≡ b mod n iff rem(a,n) = rem(b,n)

# Playing with RSA

- $p = 5$ $q = 11$ (prime numbers)
  - $n = p \times q = 55$
  - $(p-1) \times (q-1) = 40$
- $e = 3$ (public key = e, n)
  - Coprime of 40
- $d = 27$ (private key = d, n)
  - $e \times d \equiv 1 \pmod{40}$ -> $d \times e \bmod 40 = 1$ -> $(27 \times 3) \bmod 40 = 1$
- For a message to encrypt, $P = 26$ (notice that $P, C \in [0, n-1]$)
  - $C = P^e \bmod n$ = $26^3 \bmod 55 = 31$
  - $P = C^d \bmod n$ = $31^{27} \bmod 55 = 26$

# Hybrid Encryption

- Combines symmetric with asymmetric cryptography
  - Use the best of both worlds, while avoiding problems
  - Asymmetric cipher: Uses public keys (but it is slow)
  - Symmetric cipher: Fast (but with weak key exchange methods)

- Method:
  - Obtain $K_{pub}$ from the receiver
  - Generate a random $K_{sym}$
  - Calculate $\mathbf{C1 = E_{sym}( K_{sym}, P )}$
  - Calculate $\mathbf{C2 = E_{asym}( K_{pub}, K_{sym} )}$
  - Send **C1 + C2**
    - C1 = Text encrypted with symmetric key
    - C2 = Symmetric key encrypted with the receiver public key
      - May also contain the IV

# Randomization of asymmetric encryptions

- RSA is a deterministic algorithm: equal messages result in equal outputs

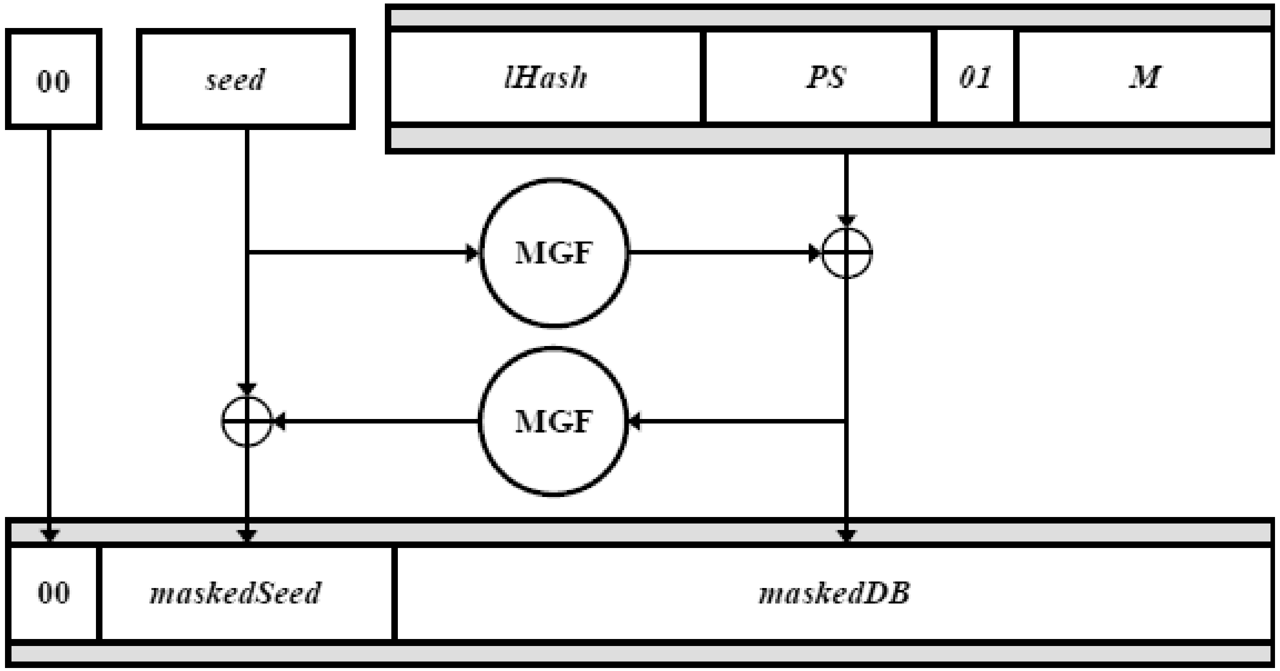
- What we need: Non-deterministic result of asymmetric encryptions
  - **N** encryptions of the same value, with the same key, should yield N different results
  - **Goal: prevent the trial & error discovery of encrypted values**

- Approaches
  - Concatenation of value to encrypt with two values
  - A fixed one (for integrity control)
  - A random one (for randomization)

# Randomization of asymmetric encryptions

## OAEP (Optimal Asymmetric Encryption Padding)

- `iHash`: digest over Label
- `seed`: random value
- PS: zeros
- M: plaintext
- MGF: Mask Generation Function
  - Similar to Hash, but with variable size



# Diffie-Hellman Key Agreement (1976)

q (large prime)
α (primitive root mod q)

| Alice | | Bob |
|---|---|---|
| $a$ = random | | $b$ = random |
| $Y_a = \alpha^a \bmod q$ | $Y_a \rightarrow$ | |
| | $\leftarrow Y_b$ | $Y_b = \alpha^b \bmod q$ |
| $K_{ab} = {Y_b}^a \bmod q$ | $K_{ab} = K_{ba}$ | $K_{ba} = {Y_a}^b \bmod q$ |

# Diffie-Hellman Key Agreement (1976)

**a = random**

$Y_a = \alpha^a \bmod q$

$K_{ac} = {Y_c}^a \bmod q$

$\xrightarrow{Y_a}$

$\xleftarrow{Y_c}$

**c = random**

$Y_c = \alpha^c \bmod q$

$K_{ca} = {Y_a}^c \bmod q$

$K_{cb} = {Y_b}^c \bmod q$

$\xleftarrow{Y_b}$

$\xrightarrow{Y_c}$

**b = random**

$Y_b = \alpha^b \bmod q$

$K_{bc} = {Y_c}^b \bmod q$

# Elliptic Curve Cryptography (ECC)

- Elliptic curves are specific functions
  - They have a generator ($G$)
  - A private key $K_{prv}$ is an integer with a maximum of bits allowed by the curve
  - A public key $K_{pub}$ is a point $(x,y) = K_{prv} \times G$
  - Given $K_{pub}$, it should be hard to guess $K_{prv}$

- Curves
  - NIST curves (15)
    - P-192, P-224, P-256, P-384, P-521
    - B-163, B-233, B-283, B-409, B-571
    - K-163, K-233, K-283, K-409, K-571

**Other curves**

- Curve25519 (256 bits)
- Curve448 (448 bits)

# ECDH: DH with ECC

ECC curve → G

| | | |
|---|---|---|
| $a$ = random | | **$b$ = random** |
| $Y_a = a\,G$ | $\xrightarrow{Y_a}$ | |
| | $\xleftarrow{Y_b}$ | $Y_b = b\,G$ |
| $K_{ab} = a\,Y_b$ | | $K_{ba} = b\,Y_a$ |

$K_{ab} = K_{ba}$

# ECC public key encryption

## Combines hybrid encryption with ECDH

- Obtain $K_{pub\_recv}$ from the receiver
- Generate a random $K_{prv\_send}$ and the corresponding $K_{pub\_send}$
- Calculate $K_{sym} = K_{prv\_send}\ K_{pub\_recv}$

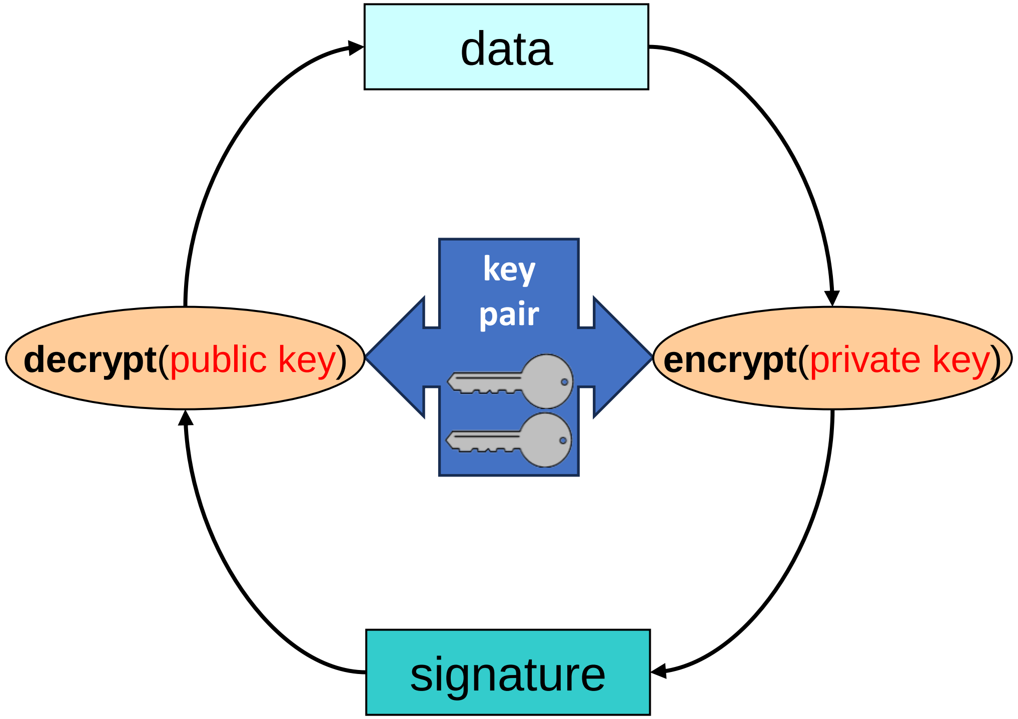
- $C = E(\ P,\ K_{sym}\ )$

- Send $C + K_{pub\_send}$
- Receiver calculates $K_{sym} = K_{pub\_send}\ K_{prv\_recv}$

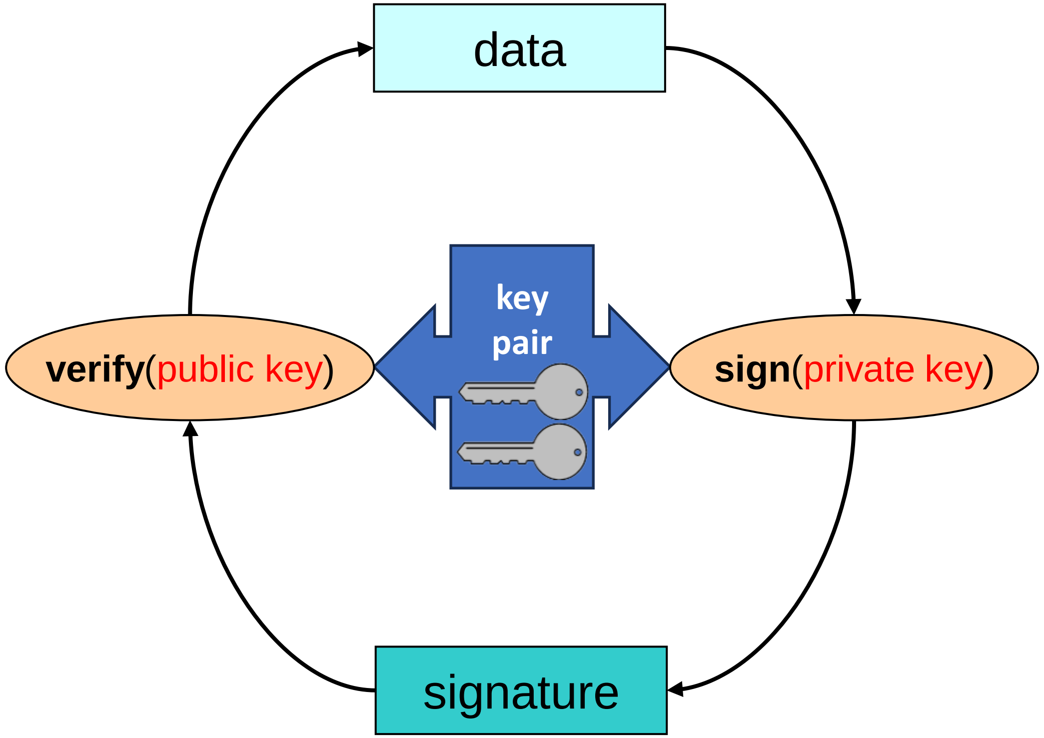
- $P = D(\ C,\ K_{sym}\ )$

# Digital signatures

## Encrypt/Decrypt (RSA)




## Sign/Verify (ElGamal, EC)



# Operations with Private Keys

- Authenticate the contents of a document
  - Ensure its integrity (it was not changed)

- Authenticate its author
  - Ensure the identity of the creator/originator

- Prevent repudiation of the encrypted payload
  - Non-repudiation
  - Genuine authors cannot deny authorship
    - Only the identified author could have generated a given payload
    - Because only the author has the private key

# Digital signatures

- Authenticate the contents of a document
  - Ensure its integrity (it was not changed)
- Authenticate its author
  - Ensure the identity of the creator/originator
- Prevent repudiation of signatures
  - Non-repudiation property
  - Genuine authors cannot deny authorship
    - Only the identified author could have generated a given signature

# Practical Considerations

- Encryption with private key is vital for authentication
  - Only the author can make it, everyone can verify it

- But… sending secure authenticated texts will require two (slow) encryptions
  - Remember: Asymmetric ciphers are slow and inefficient

- Preferred Approach: **Encrypt Hash(T), creating Digital Signatures**

# Digital Signatures

- Approaches
  - Digest function of the Text (only for performance)
  - Asymmetric encryption/decryption or signature/verification

Signing:

$$A_x(doc) = info + E(K_x^{-1}, digest(doc + info))$$

$$A_x(doc) = info + S(K_x^{-1}, digest(doc + info))$$

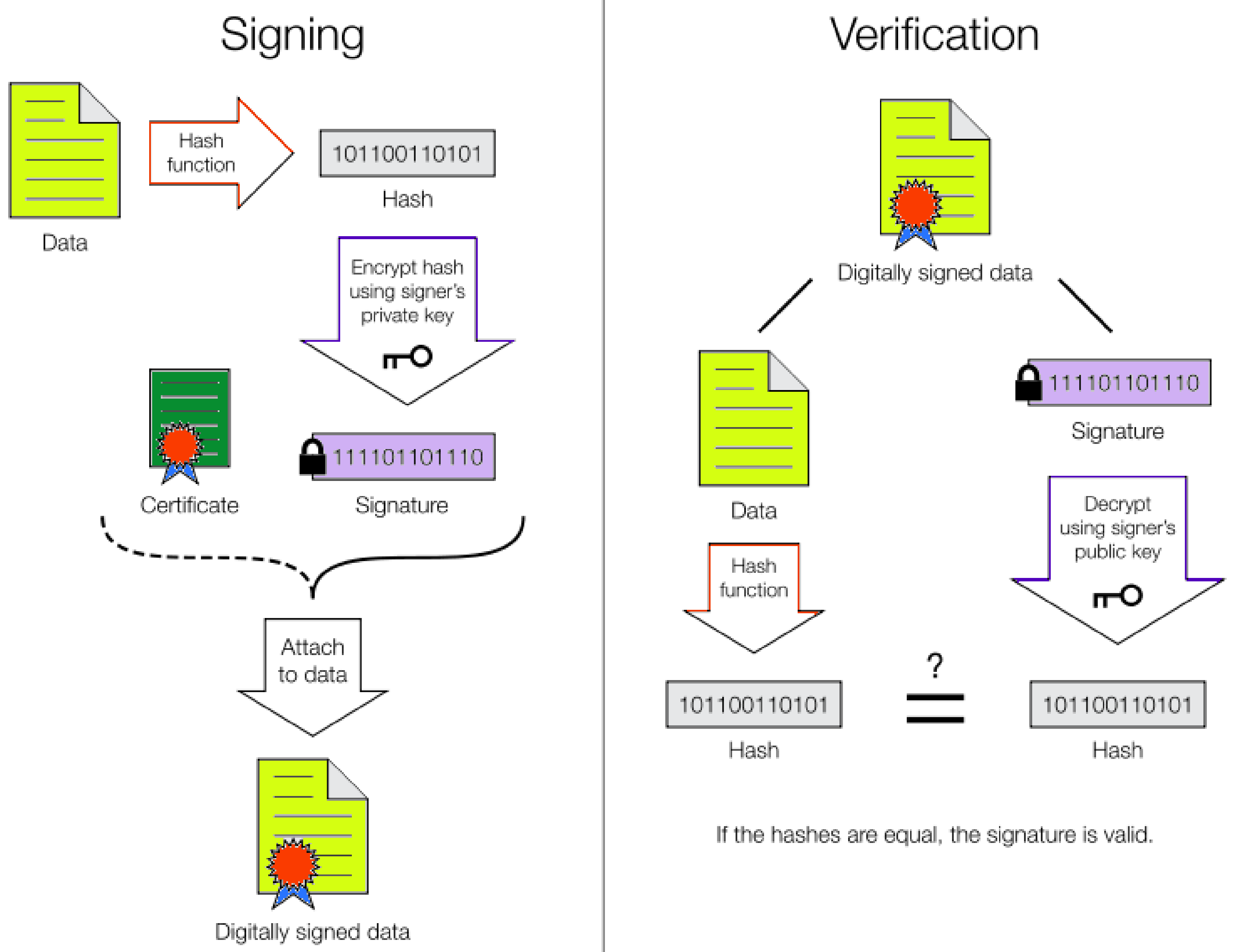
$$info = \text{signing context, signer identity, } K_x$$

Verification:

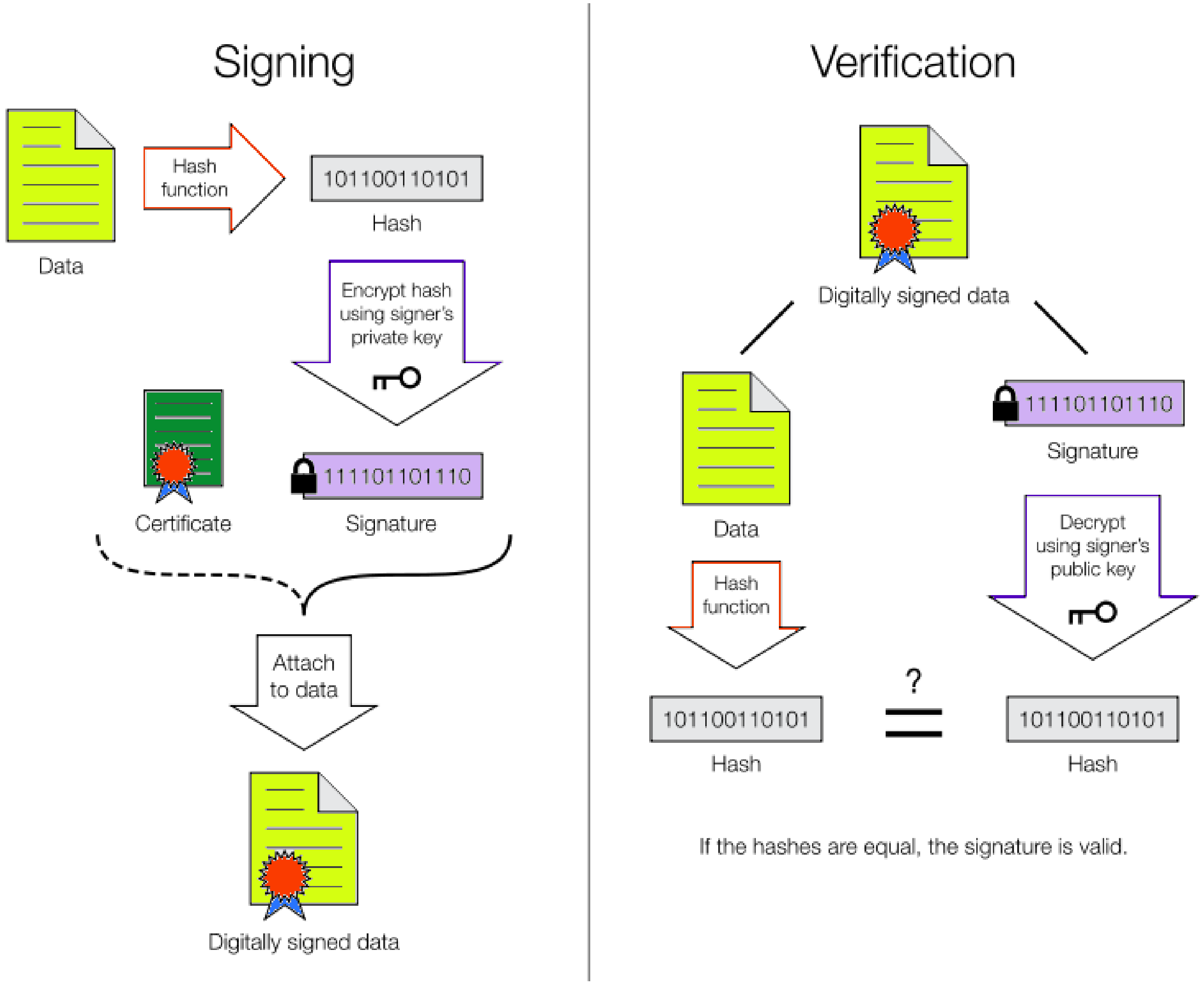
$$D(K_x, A_x(doc)) \equiv digest(doc + info)$$

$$V(K_x, A_x(doc), doc, info) \rightarrow \text{True / False}$$

# Encryption / decryption signatures



# Encryption / decryption signatures



# Digital Signature on a mail message

## Multipart content, signature w/ certificate

```
From - Fri Oct 02 15:37:14 2009
[…]
Date: Fri, 02 Oct 2009 15:35:55 +0100
From: User A <usera@domain.com>
MIME-Version: 1.0
To: User B <userb@domain.com>
Subject: Teste
Content-Type: multipart/signed; protocol="application/x-pkcs7-signature"; micalg=sha1; boundary="------------ms050405070101010502050101"

This is a cryptographically signed message in MIME format.

--------------ms050405070101010502050101
Content-Type: multipart/mixed;
 boundary="------------060802050708070409030504"

This is a multi-part message in MIME format.
--------------060802050708070409030504
Content-Type: text/plain; charset=ISO-8859-1
Content-Transfer-Encoding: quoted-printable

Corpo do mail

--------------060802050708070409030504–
--------------ms050405070101010502050101
Content-Type: application/x-pkcs7-signature; name="smime.p7s"
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename="smime.p7s"
Content-Description: S/MIME Cryptographic Signature

MIAGCSqGSIb3DQEHAqCAMIACAQExCzAJBgUrDgMCGgUAMIAGCSqGSIb3DQEHAQAAoIIamTCCBUkwggSyoAMCAQICBAcnIaEwDQYJKoZIhvcNAQEFBQAwdTELMAkGA1UEBhMCVVMxGDAWBgNV
[…]
KoZIhvcNAQEBBQAEgYCofks852BV77NVuww53vSxO1XtI2JhC1CDlu+tcTPoMD1wq5dc5v40Tgsaw0N8dqgVLk8aC/CdGMbRBu+J1LKrcVZa+khnjjtB66HhDRLrjmEGDNttrEjbqvpd2QO2
vxB3iPTlU+vCGXo47e6GyRydqTpbq0r49Zqmx+IJ6Z7iigAAAAAAAA==
--------------ms050405070101010502050101--
```

# Digital Signatures at kernel.org