

Asymmetric Cryptography

SIO

deti universidade de aveiro
departamento de eletrónica,
telecomunicações e informática

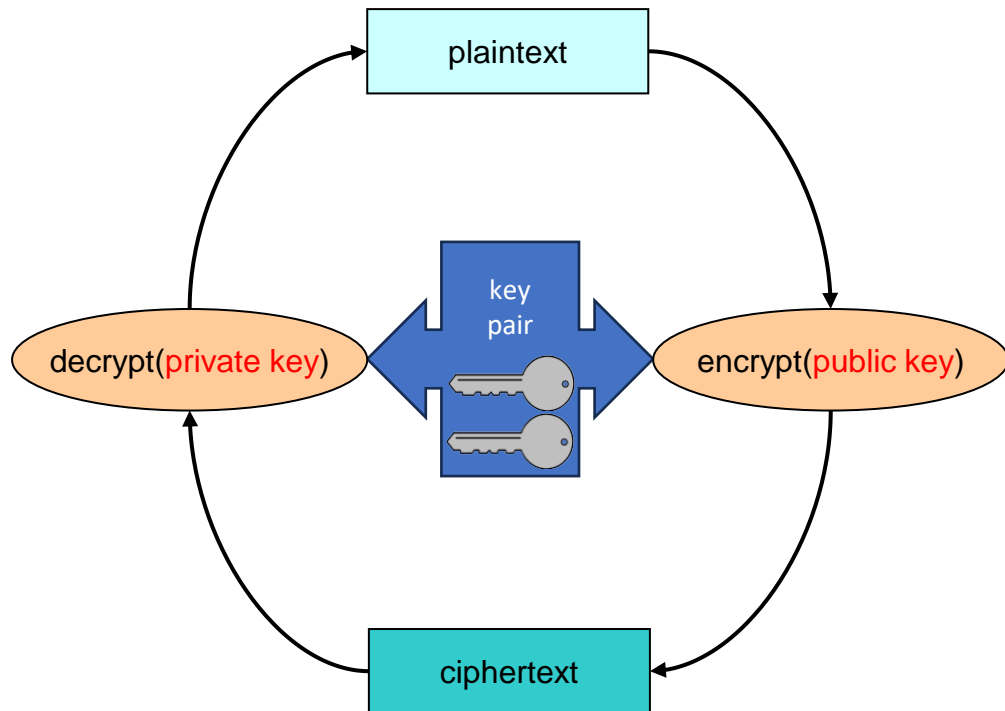
João Paulo Barraca

Asymmetric (Block) Ciphers

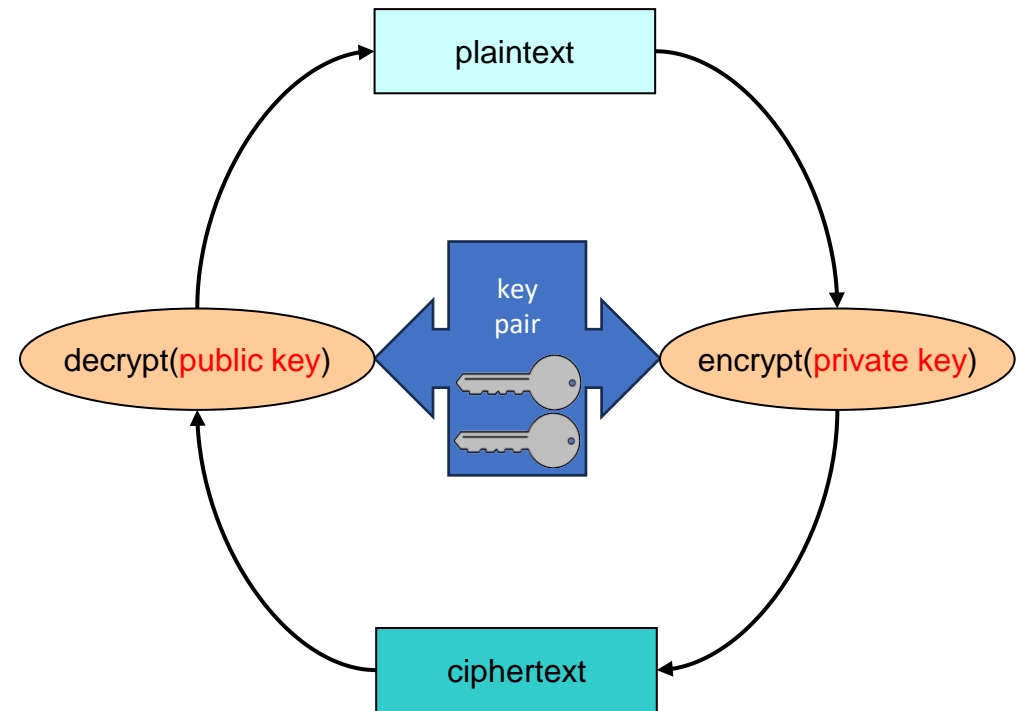
- Use key pairs
 - **One private key:** personal, not transmittable
 - **One public key:** available to all
- Allow
 - Confidentiality without any previous exchange of secrets
 - Authentication
 - Of contents (data integrity)
 - Of the data origin (source authentication, or digital signature)

Operations of an asymmetric cipher

Confidentiality

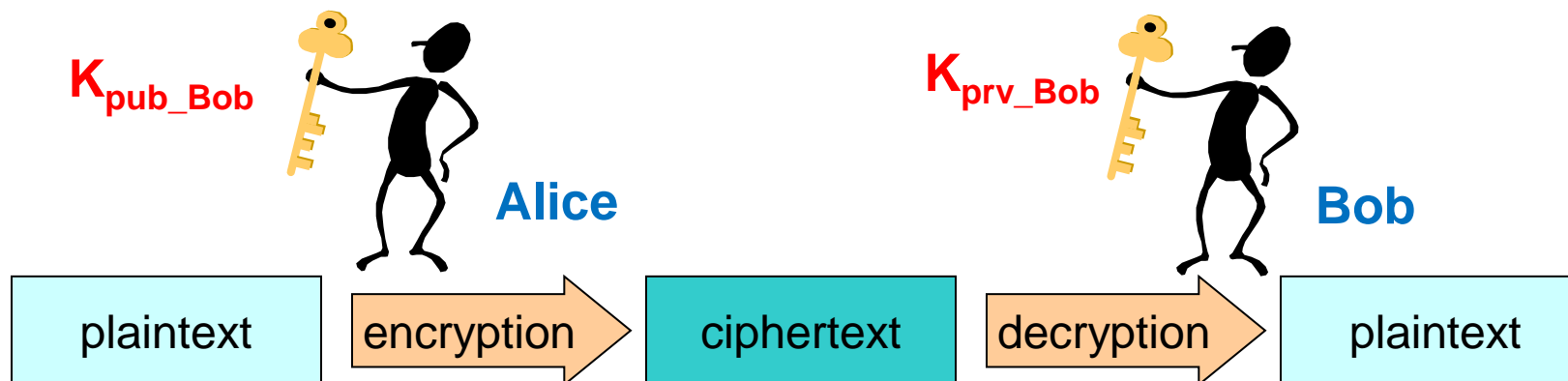


Authenticity



Use cases: confidential communication

- Secure communication with a target (Bob)
 - Alice encrypts plaintext **P** with Bob's public key $K_{\text{pub_Bob}}$
Alice: $C = \{P\}_{K_{\text{pub_Bob}}}$
 - Bob decrypts cyphertext **C** with his private key $K_{\text{prv_Bob}}$
Bob: $P' = \{C\}_{K_{\text{prv_Bob}}}$
 - **P'** should be equal to **P** (requires checking using integrity control)
 - $K_{\text{pub_Bob}}$ needs to be known by Alice



Use cases: authenticated communication

- Authenticate the communication from Alice

- Alice encrypts plaintext **P** with her private key $K_{\text{prv_Alice}}$

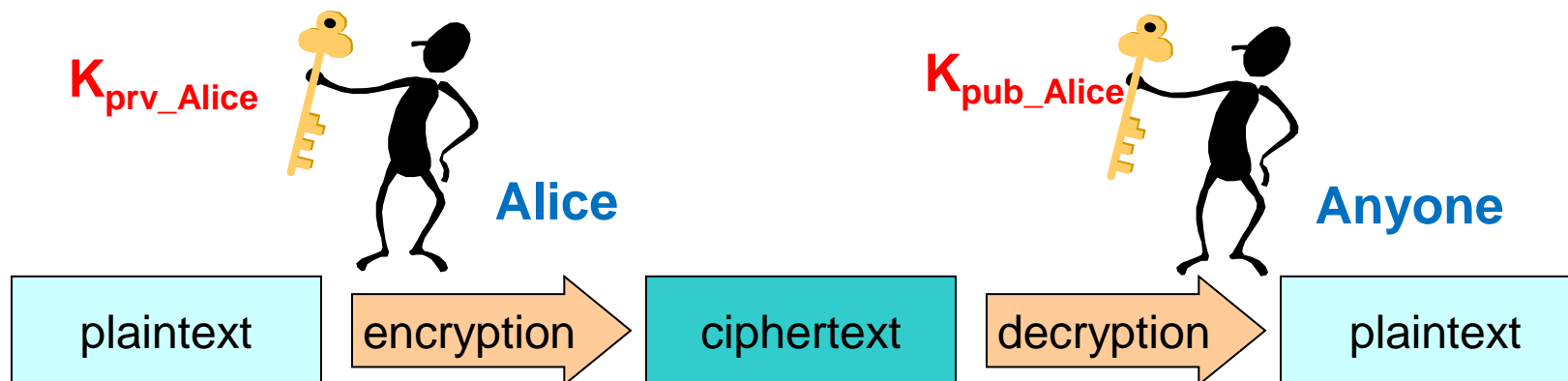
$$\text{Alice: } C = \{P\}K_{\text{prv_Alice}}$$

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

$$\text{Anyone: } P' = \{C\}K_{\text{pub_Alice}}$$

- If $P' = P$, then **C** is Alice's signature of **P**

- $K_{\text{pub_Alice}}$ 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)
 - **$n = p \times q$** with **p** and **q** being large (secret) prime numbers
 - Chose an **e** co-prime with **$(p-1) \times (q-1)$**
 - Compute **d** such that **$e \times d \equiv 1 \pmod{(p-1) \times (q-1)}$**
 - Discard **p** and **q**
 - The value of **d** cannot be computed out of **e** and **n**
 - Only from **p** and **q**

coprime $\rightarrow \gcd(a, b) = 1$

$\times \rightarrow$ multiplication

mod \rightarrow modulo operation

$\equiv \rightarrow$ modular congruence

$a \equiv b \pmod n$ iff $\text{rem}(a,n) = \text{rem}(b,n)$

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)
 - $e \times d \equiv 1 \pmod{40}$ \rightarrow $d \times e \pmod{40} = 1$ \rightarrow $(27 \times 3) \pmod{40} = 1$
- For a message to encrypt, $P = 26$ (notice that $P, C \in [0, n-1]$)
 - $C = P^e \pmod{n} = 26^3 \pmod{55} = 31$
 - $P = C^d \pmod{n} = 31^{27} \pmod{55} = 26$

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 $C1 = E_{sym}(K_{sym}, P)$
 - Calculate $C2 = E_{asym}(K_{pub}, K_{sym})$
 - Send **C1 + C2**
 - C1 = Text encrypted with symmetric key
 - C2 = Symmetric key encrypted with the receiver public key
 - May also contain the IV

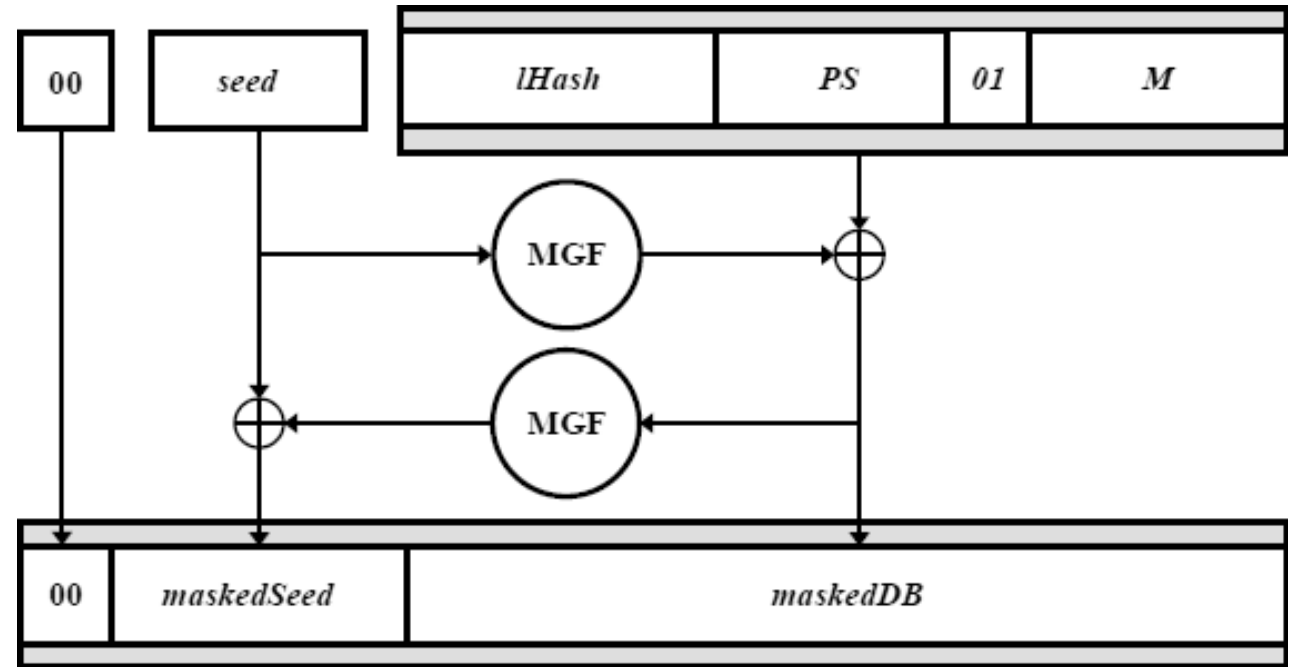
Randomization of asymmetric encryptions

- 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)

- *i*Hash: 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)



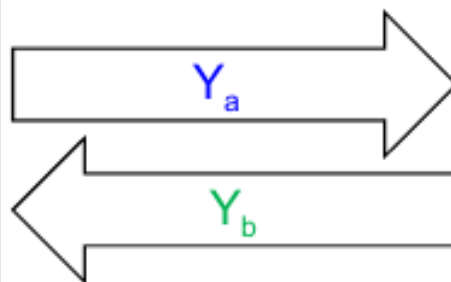
q (large prime)
 α (primitive root mod q)



a = random

$$Y_a = \alpha^a \text{ mod } q$$

$$K_{ab} = Y_b^a \text{ mod } q$$



$$K_{ab} = K_{ba}$$

b = random

$$Y_b = \alpha^b \text{ mod } q$$

$$K_{ba} = Y_a^b \text{ mod } q$$

Diffie-Hellman Key Agreement (1976)



a = random

$$Y_a = \alpha^a \text{ mod } q$$

$$K_{ac} = Y_c^a \text{ mod } q$$

c = random

$$Y_c = \alpha^c \text{ mod } q$$

$$K_{ca} = Y_a^c \text{ mod } q$$

$$K_{cb} = Y_b^c \text{ mod } q$$

b = random

$$Y_b = \alpha^b \text{ mod } q$$

$$K_{bc} = Y_c^b \text{ mod } q$$

Y_a

Y_c

Y_b

Y_c

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 $(x,y) = K_{\text{prv}} \times G$
 - 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 \rightarrow G



a = random

$$Y_a = a G$$

$$K_{ab} = a Y_b$$

b = random

$$Y_b = b G$$

$$K_{ba} = b Y_a$$

$$K_{ab} = K_{ba}$$

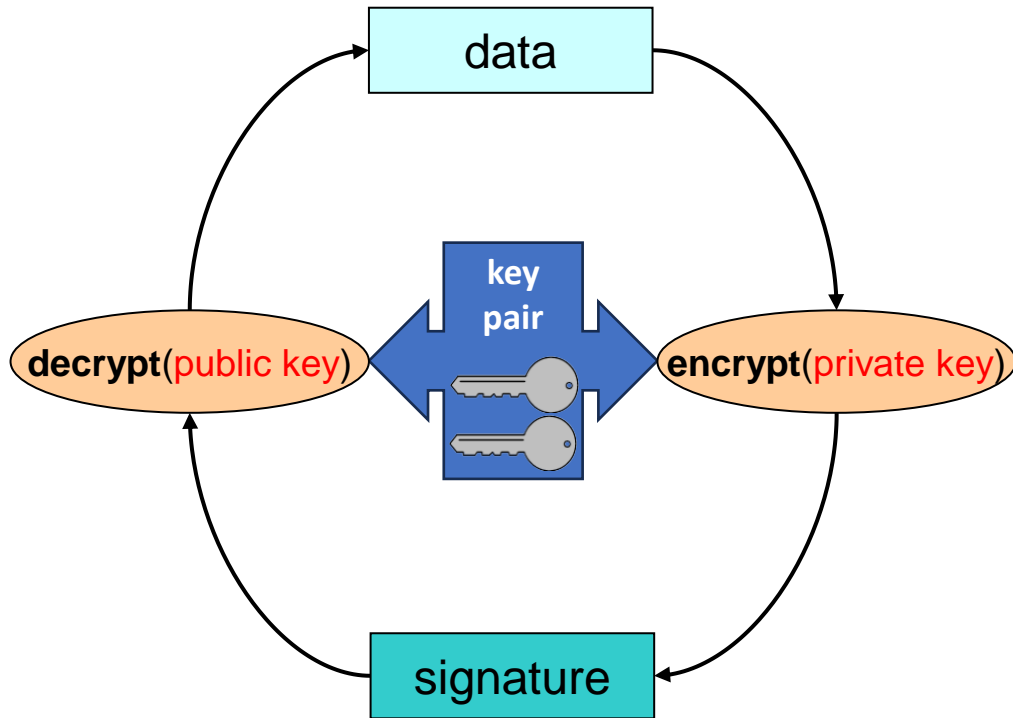
ECC public key encryption

Combines hybrid encryption with ECDH

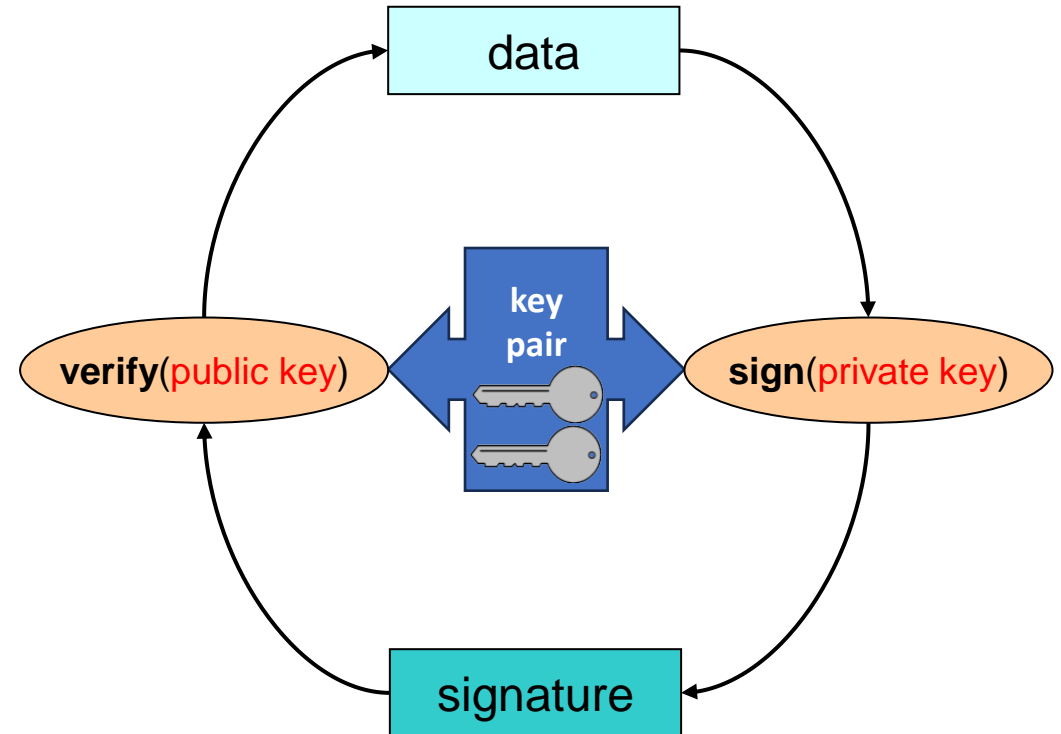
- Obtain $K_{\text{pub_recv}}$ from the receiver
- Generate a random $K_{\text{prv_send}}$ and the corresponding $K_{\text{pub_send}}$
- Calculate $K_{\text{sym}} = K_{\text{prv_send}} K_{\text{pub_recv}}$
- $C = E(P, K_{\text{sym}})$
- Send $C + K_{\text{pub_send}}$
- Receiver calculates $K_{\text{sym}} = K_{\text{pub_send}} K_{\text{prv_recv}}$
- $P = D(C, K_{\text{sym}})$

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:

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

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

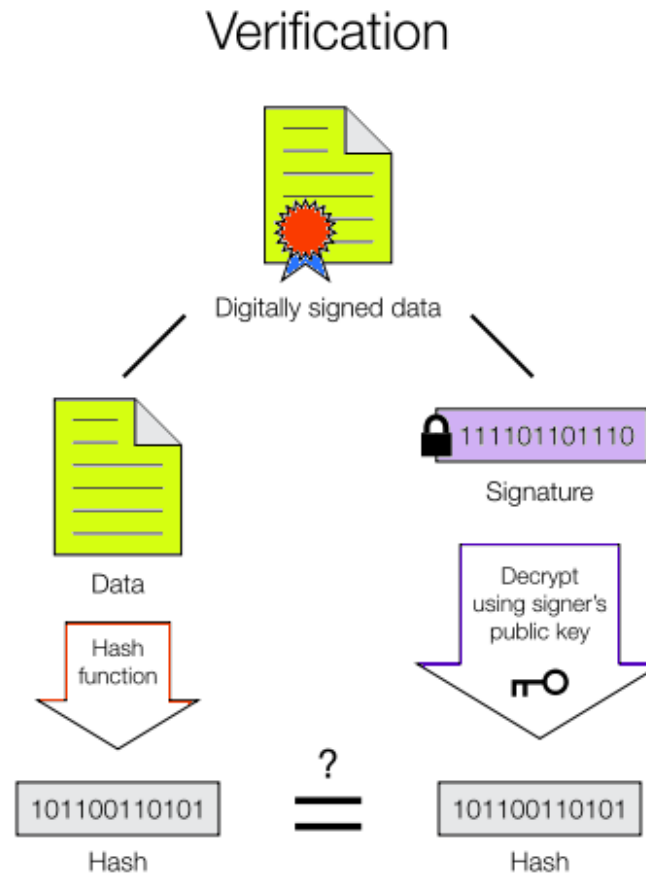
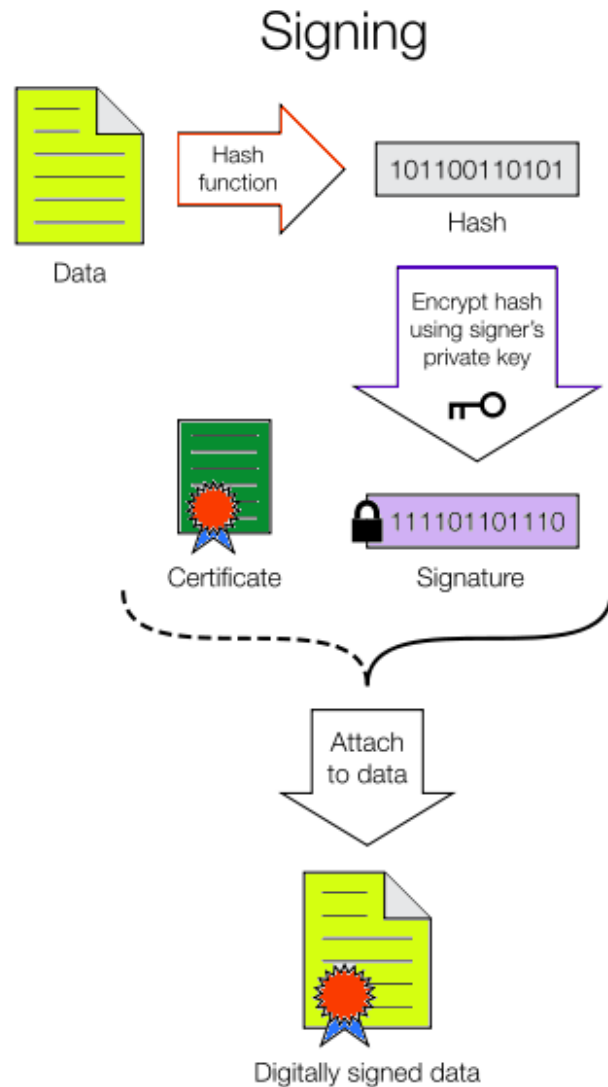
info = signing context, signer identity, K_x

Verification:

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

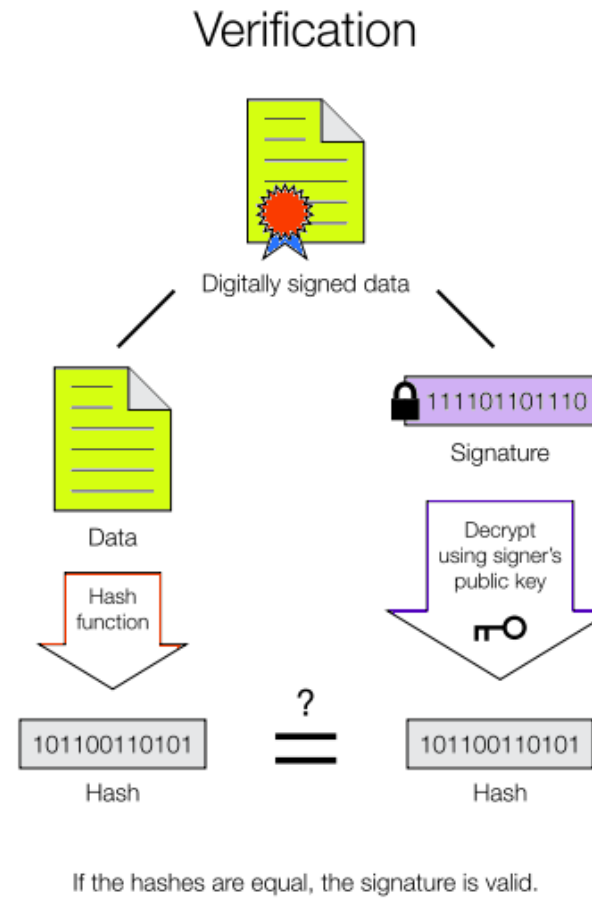
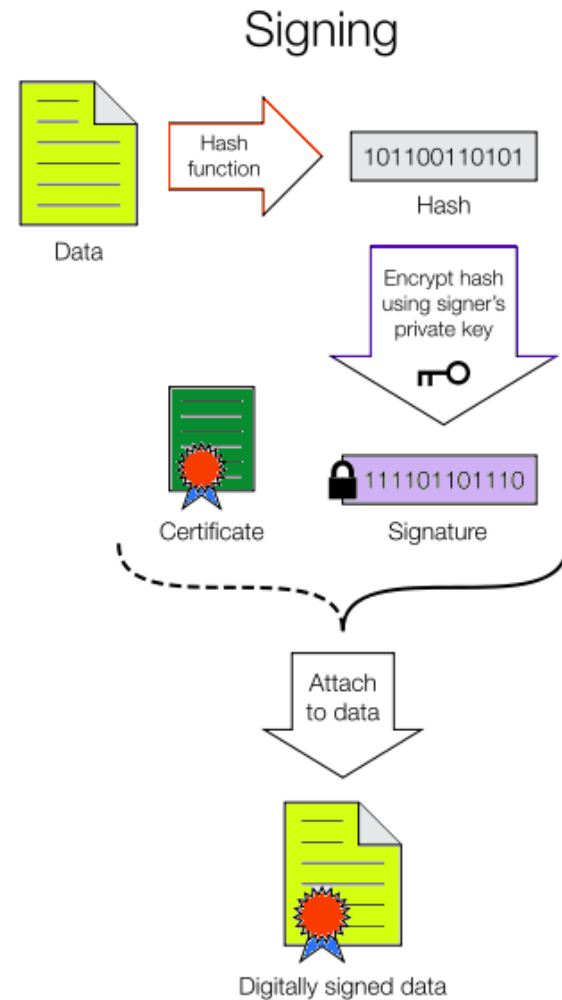
$$V(K_x, A_x(\text{doc}), \text{doc}, \text{info}) \rightarrow \text{True} / \text{False}$$

Encryption / decryption signatures



If the hashes are equal, the signature is valid.

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
```

```
MIAGCSqGSIB3DQEHAqCAMIACAQExCzAJBgUrDgMCGGUAMIAGCSqGSIB3DQEHAQAoIIamTCCBUkwgSyoAMCAQICBAcnIaEwDQYJKoZIhvcNAQEFBQAwdTElMAkGA1UEBhMVCVVMxGDAwBgNV
[...]
KoZIhvcNAQEBBQAEgYCOFks852BV77NVuww53vSx01XtI2JhC1CDlu+tcTPoMD1wq5dc5v40Tgsaw0N8dqgVLk8aC/CdGMbRBU+J1LKrcVZa+khnjjtB66HhDRLrjmEGDNttrEjbbvdpd2Q02
vxB3iPTLU+vCGXo47e6GyRydqTpbq0r49Zqmx+IJ6Z7iigAAAAA==
-----ms050405070101010502050101--
```

Digital Signatures at kernel.org

File Name	Date	Time	Size
patch-6.7.9.xz	06-Mar-2024	15:09	703K
patch-6.7.xz	08-Jan-2024	06:00	8M
patch-6.8.1.xz	15-Mar-2024	19:04	5992
patch-6.8.10.xz	17-May-2024	10:24	730K
patch-6.8.11.xz	25-May-2024	14:46	740K
patch-6.8.12.xz	30-May-2024	07:59	878K
patch-6.8.2.xz	27-Mar-2024	05:24	241K
patch-6.8.3.xz	03-Apr-2024	13:44	374K
patch-6.8.4.xz	04-Apr-2024	18:39	366K
patch-6.8.5.xz	10-Apr-2024	14:49	461K
patch-6.8.6.xz	13-Apr-2024	11:27	498K
patch-6.8.7.xz	17-Apr-2024	09:38	537K
patch-6.8.8.xz	27-Apr-2024	15:28	583K
patch-6.8.9.xz	02-May-2024	14:54	643K
patch-6.8.xz	10-Mar-2024	21:45	7M
patch-6.9.1.xz	17-May-2024	10:28	3336
patch-6.9.10.xz	18-Jul-2024	11:57	603K
patch-6.9.11.xz	25-Jul-2024	08:15	647K
patch-6.9.12.xz	27-Jul-2024	09:48	652K
patch-6.9.2.xz	25-May-2024	14:54	16K
patch-6.9.3.xz	30-May-2024	07:55	151K
patch-6.9.4.xz	12-Jun-2024	09:49	263K
patch-6.9.5.xz	16-Jun-2024	12:04	306K
patch-6.9.6.xz	21-Jun-2024	12:54	388K
patch-6.9.7.xz	27-Jun-2024	12:04	465K
patch-6.9.8.xz	05-Jul-2024	07:53	521K
patch-6.9.9.xz	11-Jul-2024	11:08	572K
patch-6.9.xz	13-May-2024	05:20	7M
sha256sums.asc	10-Oct-2024	11:05	102K

```
-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA256

3f6690efe8dc49751e33fbcc45d35fa9048f75e03bfeaaa2a28e1037ca8d85cf ChangeLog-6.0
7eb504c0d87687a37753fbfe13e54ea979648b987e6f3f49c7f8f947bee7df3e ChangeLog-6.0.1
38a40e43b4daeb3de10ad21c414c5e969f600ed4105883cab885392ada0b24c2 ChangeLog-6.0.2
1252dbe12a2bfc4320bc8721ffa6ac9755d21b355456189453949c3e34b40a30 ChangeLog-6.0.3
ed8991c1d0c78cb907af07648eacd889a8d05ce2c752fefbac52faa7a5e76e3c ChangeLog-6.0.4
a135968b2ba483877b1e0d6c29f022df2ea2202b83b2d7a6367b1d218c402822 ChangeLog-6.0.5
23982b4a283f50f9eff4cdfc5a92e3cf188373e928fd529a0355b2f03e591 ChangeLog-6.0.6
685098787f5099393813af01dbf42be53d4cf66439101819e9d6812f9ea18b0d ChangeLog-6.0.7
75ab6be0d282b450c847e4fd8d16a900c55b02dd1c2d4d367a0d72d6fa3ea6c0 ChangeLog-6.0.8
40c049dfd11dea11d06d9fa38268e6a4f1c46168ff6afa374d8977db75e4bc15 ChangeLog-6.0.9
ec14449d5d5f11d0c80cf1c1c33f2628333e1c4cf00779cd1dec66fcb934626 ChangeLog-6.0.10
d1aec42501f371cb0d46e428c56fd1b9e785a3b7ad884f641505486a1721a517 ChangeLog-6.0.11
5a7cc6b10574bf4ee627977173f6de69c150ed3a7ff039b1cc2ab2e9aea3045f ChangeLog-6.0.12
05a014d458f50fda29cb92bffb99f7ff13506d245d21124e4b836e28f0b8197a ChangeLog-6.0.13
44728440fadf4711f85d4bbc59cf43614f720c6d1aa7c9235c9497371ca55536 ChangeLog-6.0.14
05b2597d94fe9674d5c575bb008953d1b548939a55f12709fdc6b0ce69abc211 ChangeLog-6.0.15

2b269f51babfd89937206aa0fcac6f93c94cdf2f24d7e54ebc304a93cf9e4929 patch-6.11.2.xz
4c808f6dd8814ab55a343649a2e2b925895b7f97044d15fa3424e5cf69349c3e patch-6.11.3.xz
-----BEGIN PGP SIGNATURE-----
Version: GnuPG v2.0.22 (GNU/Linux)

iQIcBAEBCAAGBQJnB7TpAAoJEGMtOgZYnaax9wUP/izFOfkROcdC0YdH5pmlNQmQ
cBhqieYbwVm8lIOgUndjvcRRe/k7JaZKA73w7yGr456QGSIBu2jOvgytmdbQ2QSG
i6u1LF6npRNO41qwb5cHd1L2UTEef0qDgqtBEnvSAWHgozoopfj/2VhfdJ5H4n/n
tVjGHQeXm00EVWV0rOhLkFR31YvRMBQwNDcB79Hxazd7qpCL6/yT1K5S8wUQe6B6
Nt5m9dgdjR3WN0x1u9Wrg5akC3sSE3881P+TR/g4KhvhUzWzWMAPDHyXdv50/QEE5
0rr5XowtKkHFc2DENGf/b9egx0ojdy637JtV4kK2FbquRTAwgDNVvZv3p8M95KG2
v+8XEKpPtEThuvDI259Rfk6Q0D+aI/4uMnVGK1RzpfBz0Q9qwJGrWUF9nj1mT6Ud
GV0DegxenKfwoe8dUxR61HntFHL1T1oXiZ3YZk9XqS18N51+uUydkuGkAWlchQZo
6qIS2dvVMmHnqC99rSxaYr/Qnn5WooCTd+u4iflmcqT7ss5aBnI5hgWN30vSwkZk
mJuPK8gCHBaPrXfb9G0d6esDBFP+szNBbpUn4K1nHKmrMQT7prGfDTmJlFckIax
xTuHfjGgFZeGeg6BIGEZ37Mh5k+GmzntOK/RxUS55iytkXZOs2rYHvuO69KIJWCb
uda1IuRrgOZ4hoWw1whD
=s9Ry
-----END PGP SIGNATURE-----
```