Digital Signatures

Signature requirements

Requirements of a signature scheme:

- Authenticity (certainty of the sender)
- Integrity (immutability of the signed message)
- Non-repudiation (non-deniability by the signer)
- Confidentiality (*optional*)

Example of signature

Types of signatures

Autograph signature:

- The signer writes his or her signature on a paper document
- The authenticity of the signature is verified through graphical comparison of the signature with a signature specimen

Biometric or **graphometric signature**:

- The signer signs a digital document by writing on a digital surface
- An electronic device records the graphometric features of the signature (graphics, speed, pressure...)
- The authenticity of the signature is verified by comparing the graphometric features of the signature with those of a signature template

Digital signature:

- The signer computes some digital data representing his or her signature for a digital document
- The signature is obtained through an asymmetric cryptographic primitive, using the signer's private key
- The authenticity of the signature is verified through the same asymmetric cryptographic primitive, using the signer's public key

From asymmetric cryptography to digital signatures

In any asymmetric encryption scheme:

- Alice generates her private key (SK_A) and the corresponding public key (PK_A)
- Bob (anyone) uses Alice's public key (PK_A) to encrypt messages addressed to Alice
- Alice is the only one who can decrypt those messages by using her private key (SK_A)

From asymmetric cryptography to digital signatures (2)

Let us suppose that Alice has a message (m) to be digitally signed:

1. Alice generates her private key (SK_A) and the corresponding public key (PK_A)

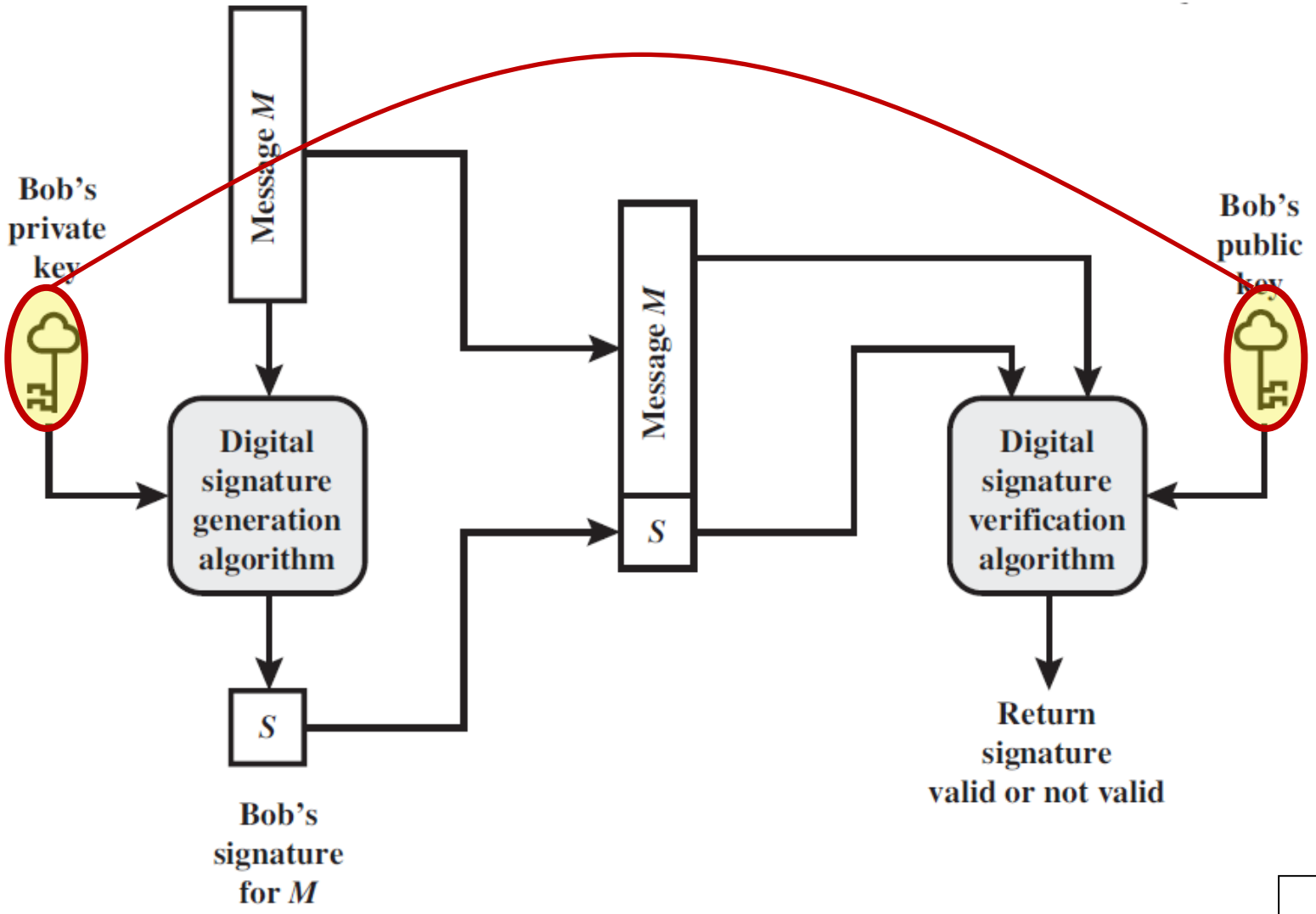
2. Alice «decrypts» the message m with her private key (SK_A) and obtains d_A

– d_A is univocally tied to m and only Alice (who owns SK_A) can compute it
– d_A therefore represents a signature of m by Alice, and can be appended to m

3. Anyone who receives (m, d_A) can «encrypt» d_A (through Alice's public key PK_A) and check whether the result coincides with m

4. If so, the signature is accepted as valid, otherwise it is discarded

General digital signature scheme



RSA signature

Key generation:

- Alice generates two large prime numbers, p and q, and calculates $n = pq$
- Alice chooses the exponent e such that $1 < e < \varphi(n) = (p - 1)(q - 1)$ and GCD $(e, \varphi(n)) = 1$
- Alice computes d such that $de \equiv 1 \pmod{\varphi(n)}$
- Alice publishes $PK_A = (e, n)$ and keeps $SK_A = (d, p, q)$ secret

Signing procedure:

- Alice signs the message m by computing $y \equiv m^d \pmod{n}$ and publishing (m, y) (then the message is not confidential)

Verification procedure:

- Bob computes $z \equiv y^e \pmod{n}$ and checks that $z = m$

RSA signature (2)

The signature cannot be used by Eve on another message (m_1), as $y^e \neq m_1 \pmod{n}$

Eve needs a new signature y_1 such that $y_1^e \equiv m_1 \pmod{n}$

This is equivalent to decrypt RSA, that can only be done by Alice

Eve could try to fix a forged signature y_1, and obtain the message as $m_1 \equiv y_1^e \pmod{n}$, but the probability that m_1 makes sense (and, moreover, has the intended meaning) is too low

ElGamal signature scheme

Key generation:

- Alice chooses a large prime number p, a primitive root α and a secret number a between 1 and $p - 2$
- Alice computes $\beta \equiv \alpha^a \pmod{p}$ and publishes (p, α, β)

Signing procedure:

- Alice chooses a secret random number k coprime to $p - 1$
- Alice computes $r \equiv \alpha^k \pmod{p}$ and $s \equiv k^{-1}(m - ar) \pmod{p - 1}$
- The signed message consists of (m, r, s)

ElGamal signature scheme (2)

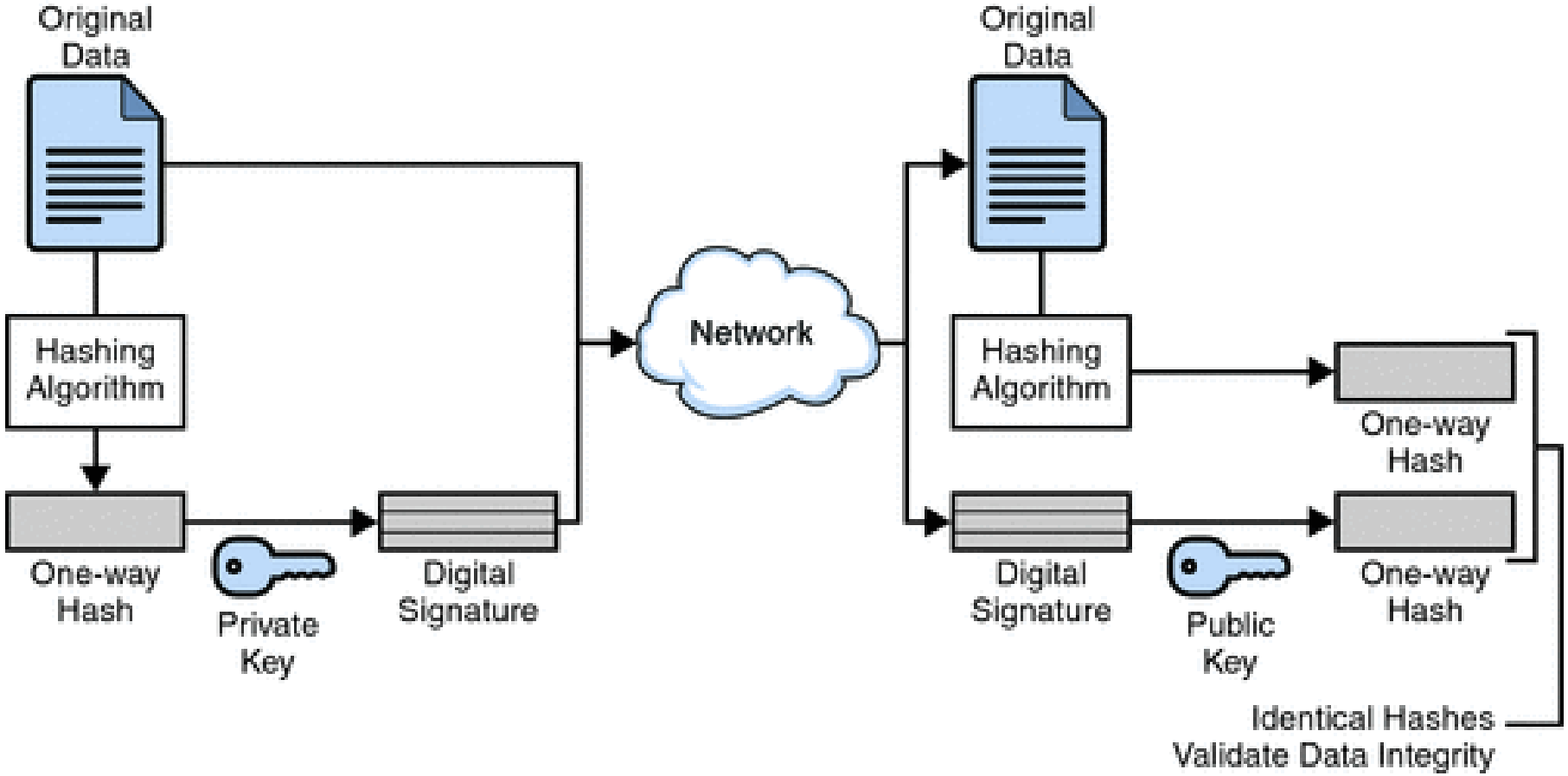
Verification procedure:

- Bob computes $v_1 \equiv \beta^r r^s \pmod{p}$ and $v_2 \equiv \alpha^m \pmod{p}$
- Bob considers the signature valid if and only if $v_1 \equiv v_2 \pmod{p}$

Hash and sign

- If we use digital signature schemes naively, they produce a signature that is as long as the signed message, or even more
- This makes digital signatures practically unusable
- A solution to this problem is that of computing the digital signature not on the message itself, but on a hash digest of the message
- This approach is called **hash-and-sign**

Hash-and-sign (2)



Hash-and-sign (3)

- A public hash function h is chosen

1. Alice generates her private-public keypair (SK, PK)
2. Alice computes the message digest $d = h(m)$, which is much shorter than m and has a fixed length, therefore it is more convenient to be signed
3. Alice computes her signature of the message digest through the decryption function, $d_A = D_{SK}(d)$, and appends it to the message: (m, d_A)
4. Bob receives (m, d_A) and:
    1. Computes a first local copy of the message digest from the message: $d' = h(m)$
    2. Computes a second local copy of the message digest from the signature, through the encryption function: $d'' = E_{PK}(d_A)$
    3. Compares the two local copies of the message digest
5. If $d' = d''$, then Bob accepts the signature, otherwise he discards it

Hash-and-sign (4)

In order for Eve to use an intercepted signature on a different message (m'), she needs that $sig(h(m')) = sig(h(m))$, that is, $h(m') = h(m)$

She would hence need to find a **hash collision**

Without the hash function, it would be needed that $sig(m') = sig(m)$, which is clearly impossible for the bijective nature of the asymmetric primitive

Digital Signature Algorithm

The Digital Signature Algorithm (DSA) is FIPS standard for digital signatures proposed by the National Institute of Standards and Technology (NIST) in August 1991 for use in the Digital Signature Standard (DSS), and finally adopted in 1994. It was later revised in 1996.

DSA is a hash-and-sign signature scheme

The signature is applied to a 160-bit message digest

Digital Signature Algorithm (2)

Alice chooses a 160-bit prime number q and another prime number p such that $q|(p-1)$

For the system to be sure, the problem of discrete logarithm must be difficult to solve for p (in the first version, p was 512-bit long)

If g is a primitive root modulo p and $\alpha \equiv g^{(p-1)/q} \pmod{p}$, then $\alpha^q \equiv 1 \pmod{p}$

Alice chooses a secret number a between 1 and $q-2$ and calculates $\beta \equiv \alpha^a \pmod{p}$

Alice publishes (p, q, α, β) and keeps a secret

Digital Signature Algorithm (3)

Signing procedure:

- Alice draws a random number k between 1 and $q - 2$
- Alice computes $r \equiv (\alpha^k \pmod{p}) \pmod{q}$
- Alice computes $s \equiv k^{-1}(m + ar) \pmod{q}$
- Alice sends the signature (r, s) together with the message m

Verification procedure:

- Bob computes $u_1 \equiv s^{-1}m \pmod{q}$ and $u_2 \equiv s^{-1}r \pmod{q}$
- Bob computes $v \equiv (\alpha^{u_1}\beta^{u_2} \pmod{p}) \pmod{q}$
- Bob verifies that $v = r$