Digital Signature Schemes

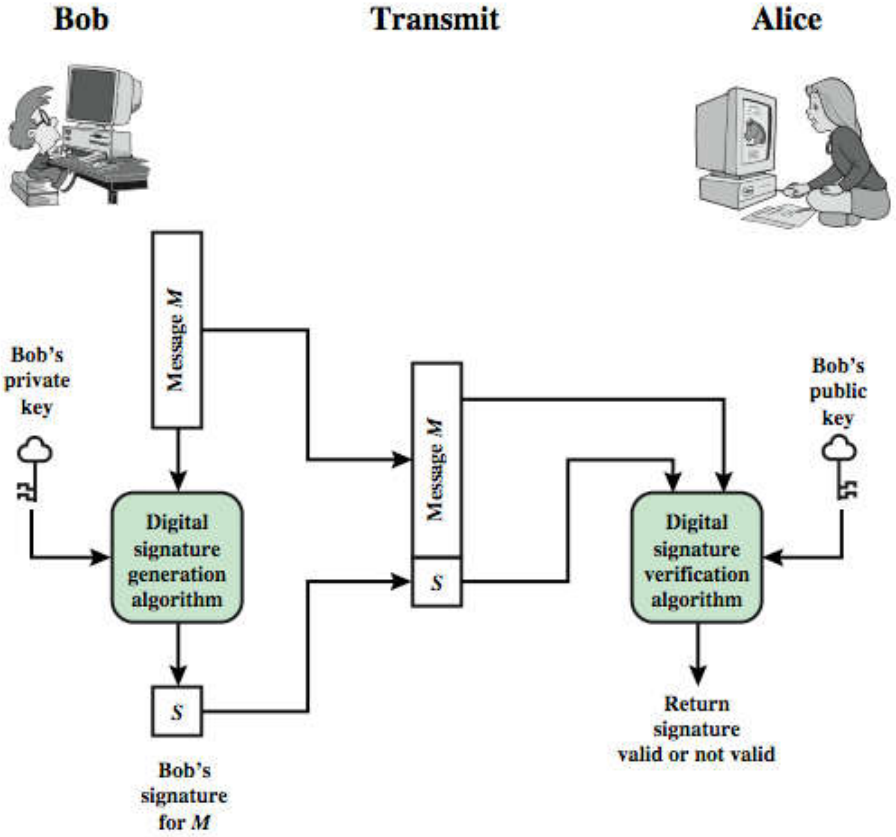
Dr. Ayad Ibrahim, 2018-2019

Outline

- Introduction
- Advantage of Digital signatures
- Adversary Goal
- RSA-signature
- Attacks on RSA-signature
- Hashed-RSA
- Shnorr Signature
- DSA algorithm

Introduction

- Digital signature schemes allow a signer *S* who has a public key *pk* to "**sign**" a message such that any other party who knows *pk* can verify the signature.



Services of digital signature

1. **Authentication**: verify that the message originated from S.
2. **Integrity**: ensure message has not been modified in any way.

- Signature schemes can be viewed as the public-key *counterpart* of **message authentication** codes.

Advantages of digital signature over MAC

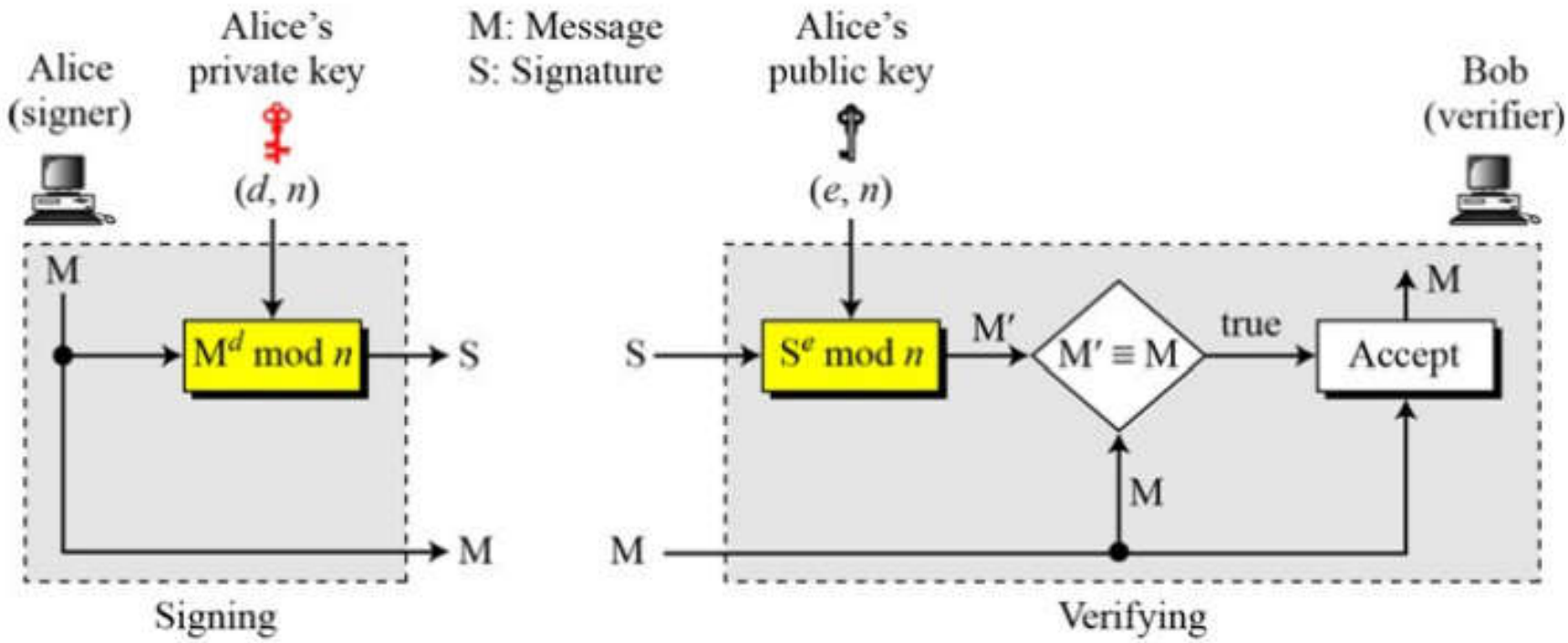
- The sender sign message once for all recipients.
- Third party can verify the legitimate signature on *m* with respect to S's public key.
- *Non-repudiation*: a valid signature on a message is enough to convince the judge that S indeed signed this message.
- Message authentication codes have the advantage of being roughly 2-3 orders of magnitude more *efficient* than digital signatures.

Adversary Goal

- **Existential forgery**

"Given a public key *pk* generated by a signer S, we say an adversary outputs a forgery if it outputs a message *m* along with a valid signature on *m*, such that *m* was not previously signed by S"

RSA Signatures



$$M' \equiv M \pmod{n} \quad \rightarrow \quad S^e \equiv M \pmod{n} \quad \rightarrow \quad M^{d \times e} \equiv M \pmod{n}$$

Attacks of RSA-signature

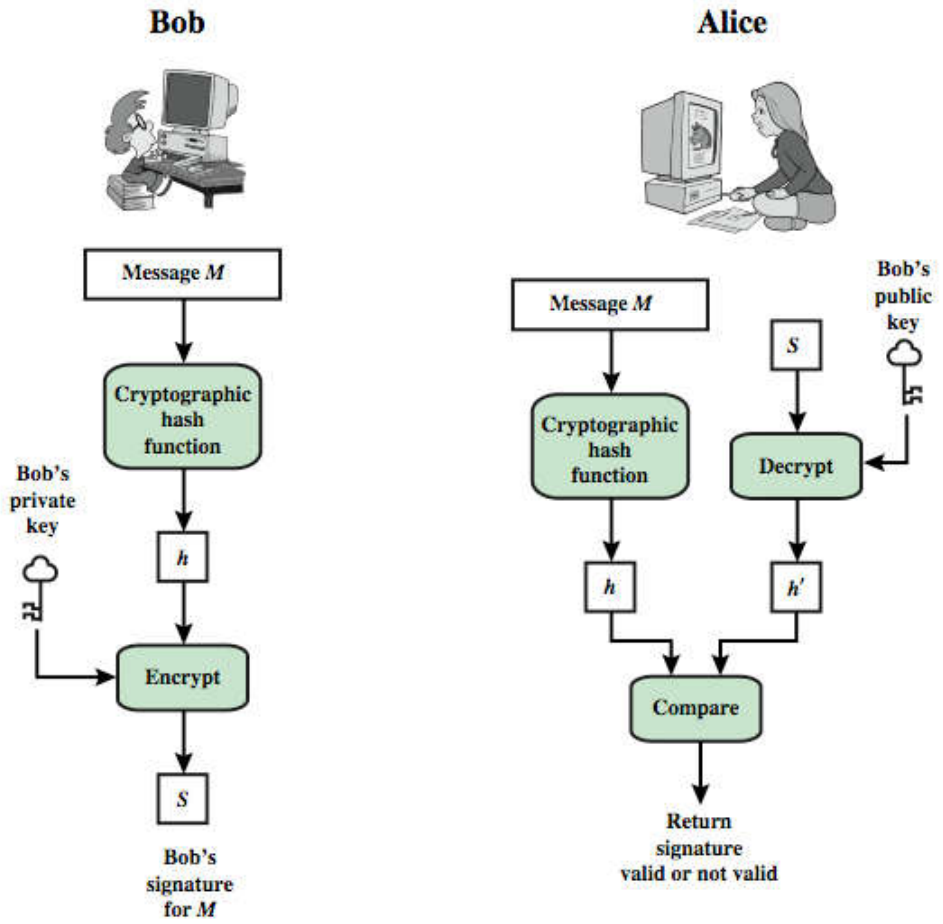
- The attack works as follows: given public key *pk* = <*N*, e>, choose arbitrary $\sigma \in \mathbb{Z}_N^*$ and compute $m = \sigma^e \mod N$; then output the forgery (*m*, σ).

- The adversary can chooses a random $m1 \in \mathbb{Z}_N^*$, sets *m2* := *[m/m1 mod N]*, and then obtains signatures $\sigma 1, \sigma 2$ on m_1 and m_2, respectively.
- We claim that $\sigma := \sigma 1 . \sigma 2 \mod N$ is a valid signature on *m*.
- This is because:

$$\sigma^e = (\sigma_1 \cdot \sigma_2)^e = (m_1^d \cdot m_2^d)^e = m_1^{ed} \cdot m_2^{ed} = m_1 m_2 = m \bmod N,$$

Hashed-RSA

- The basic idea is to take modify the textbook RSA signature scheme by applying some function H to the message before signing it.



Discrete Logarithm(s) (DLs)

- Fix a prime *p*.
- Let a, b be nonzero integers (mod p).
- The problem of finding x such that $a^x \equiv b \pmod{p}$ is called the discrete logarithm problem.
- Suppose that n is the smallest integer such that $a^n \equiv 1 \pmod{p}$, i.e., n=**ord**(a).
- By assuming $0 \le x < n$, we denote $x=L_a(b)$, and call it the discrete log of b w.r.t. a (mod p)
- Ex: p=11, a=2, b=9, then $x=L_2(9)=6$

Schnorr's Signature

- Schnorr assumes the discrete log problem is difficult in prime order groups.
- Key generation

1. Choose primes p and q, such that q is a prime factor of $p - 1$.
2. Choose an integer a, such that $\alpha^q = 1 \bmod p$. The values a, p, and q comprise a global public key that can be common to a group of users.
3. Choose a random integer s with $0 < s < q$. This is the user's private key.
4. Calculate $v = a^{-s} \bmod p$. This is the user's public key.

Schnorr's Signature

- Signing

A user with private key and public key generates a signature as follows.

1. Choose a random integer r with $0 < r < q$ and compute $x = a^r \bmod p$. This computation is a preprocessing stage independent of the message M to be signed.
2. Concatenate the message with x and hash the result to compute the value e:

$$e = \mathrm{H}(M \| x)$$

3. Compute $y = (r + se) \bmod q$. The signature consists of the pair (e, y).

Schnorr's Signature

- Verification

1. Compute $x' = a^y v^e \bmod p$.
2. Verify that $e = \mathrm{H}(M \| x')$.

To see that the verification works, observe that

$$x' \equiv a^y v^e \equiv a^y a^{-se} \equiv a^{y-se} \equiv a^r \equiv x \pmod{p}$$

Hence, $\mathrm{H}(M \| x') = \mathrm{H}(M \| x)$.

Digital Signature Algorithm (DSA)

- creates a 320 bit signature
- with 512-1024 bit security
- smaller and faster than RSA
- a digital signature scheme only
- security depends on difficulty of computing discrete logarithms

DSA Key Generation

- have shared global public key values (p, q, g):
  - choose 160-bit prime number q
  - choose a large prime p with $2^{L-1} < p < 2^{L}$
    - where L= 512 to 1024 bits and is a multiple of 64
    - such that q is a 160-bit prime divisor of $(p-1)$
  - choose $g = h^{(p-1)/q}$
    - where $1<h<p-1$ and $h^{(p-1)/q} \bmod p > 1$
- users choose private & compute public key:
  - choose random private key: $x<q$
  - compute public key: $y = g^{x} \bmod p$

DSA Signature Creation

- to **sign** a message `M` the sender:
  - generates a random signature key `k, k<q`
  - nb. `k` must be random, be destroyed after use, and never be reused
- then computes signature pair:

$$r = (g^k \bmod p) \bmod q$$

$$s = [k^{-1}(H(M) + xr)] \bmod q$$

- sends signature `(r,s)` with message `M`

DSA Signature Verification

- having received M & signature `(r,s)`
- to **verify** a signature, recipient computes:

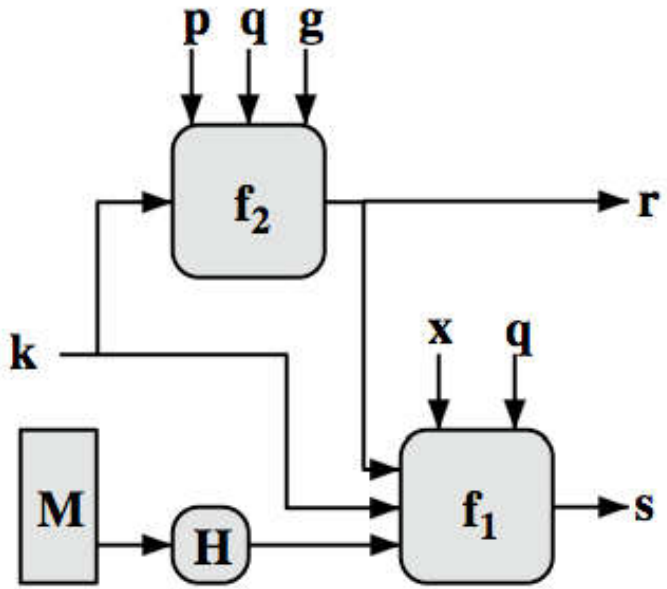
  $w = s^{-1} \bmod q$

  $u1 = [H(M)w] \bmod q$

  $u2 = (rw) \bmod q$

  $v = [(g^{u1}\ y^{u2}) \bmod p] \bmod q$
- if $v=r$ then signature is verified

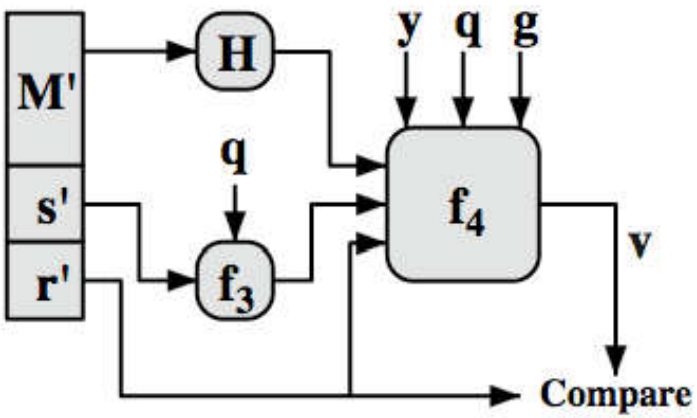
DSS Overview



$s = f_1(H(M), k, x, r, q) = (k^{-1} (H(M) + xr)) \bmod q$

$r = f_2(k, p, q, g) = (g^k \bmod p) \bmod q$

(a) Signing



$w = f_3(s', q) = (s')^{-1} \bmod q$

$v = f_4(y, q, g, H(M'), w, r')$

$= ((g^{(H(M')w) \bmod q} y^{r'w \bmod q}) \bmod p) \bmod q$

(b) Verifying

Correctness of DSA

$$s = k^{-1}(H(m) + xr) \bmod q$$

Thus

$$\begin{aligned} k &\equiv H(m)s^{-1} + xrs^{-1} \\ &\equiv H(m)w + xrw \pmod{q} \end{aligned}$$

Since g has order $q \pmod{p}$ we have

$$\begin{aligned} g^k &\equiv g^{H(m)w} g^{xrw} \\ &\equiv g^{H(m)w} y^{rw} \\ &\equiv g^{u_1} y^{u_2} \pmod{p} \end{aligned}$$

Finally, the correctness of DSA follows from

$$\begin{aligned} r &= (g^k \bmod p) \bmod q \\ &= (g^{u_1} y^{u_2} \bmod p) \bmod q \\ &= v \end{aligned}$$

- Thanks for listening