Digital Signature Schemes

Department of Computer Engineering

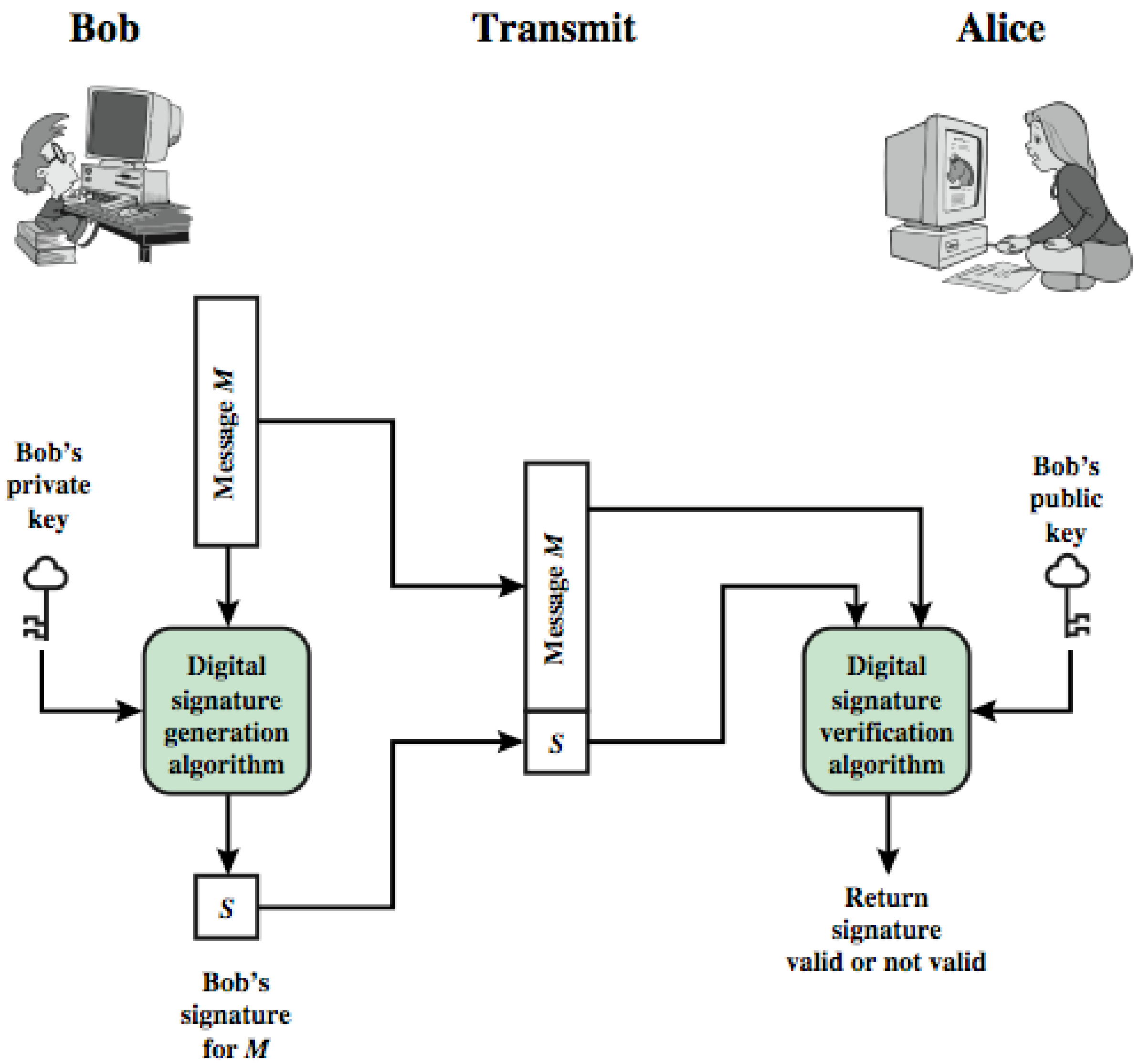
CRYPTOGRAPHY
CS-531

Faculty-
Vivek Kumar Anand

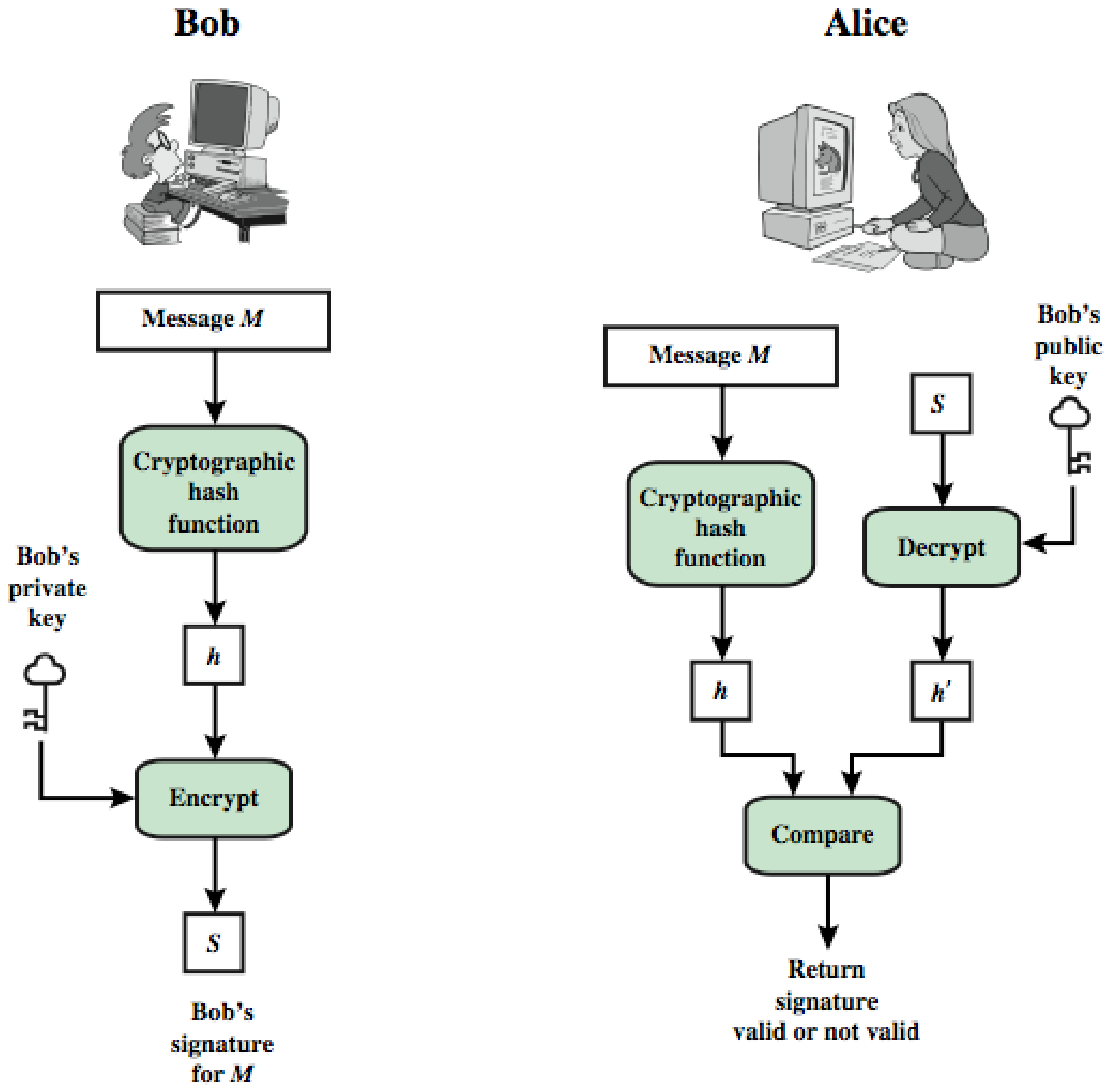
Digital Signatures

- have looked at message authentication
  - but does not address issues of lack of trust
- digital signatures provide the ability to:
  - verify author, date & time of signature
  - authenticate message contents
  - be verified by third parties to resolve disputes
- hence include authentication function with additional capabilities

Digital Signature Model



Digital Signature Model



Attacks

- C = Attacker, A = victim
- **Key-only attack:** C only knows A's public key.
- **Known message attack:** C has set of messages, signatures.
- **Generic chosen message attack:** C obtains A's signatures on messages selected without knowledge of A's public key.
- **Directed chosen message attack:** C obtains A's signatures on messages selected after knowing A's public key.
- **Adaptive chosen message attack:** C may request signatures on messages depending upon previous message-signature pairs.

Forgeries

- **Total break:** C determines A's private key.
- **Universal forgery:** C finds an efficient signing algorithm that provides an equivalent way of constructing signatures on arbitrary messages.
- **Selective forgery:** C forges a signature for a particular message chosen by C.
- **Existential forgery:** C forges a signature for a particular message not chosen by C. Consequently, this forgery may only be a minor nuisance to A.

Digital Signature Requirements

- must depend on the message signed
- must use information unique to sender
  - to prevent both forgery and denial
- must be relatively easy to produce
- must be relatively easy to recognize & verify
- be computationally infeasible to forge
  - with new message for existing digital signature
  - with fraudulent digital signature for given message
- be practical save digital signature in storage

Direct Digital Signatures

- involve only sender & receiver
- assumed receiver has sender’s public-key
- digital signature made by sender signing entire message or hash with private-key
- can encrypt using receivers public-key
- important that sign first then encrypt message & signature
- security depends on sender’s private-key

ElGamal Digital Signature

- signature variant of ElGamal, related to D-H
  - so uses exponentiation in a finite (Galois)
  - with security based difficulty of computing discrete logarithms, as in D-H
- use private key for encryption (signing)
- uses public key for decryption (verification)
- each user (eg. A) generates their key
  - chooses a secret key (number): $1 < x_A < q-1$
  - compute their **public key**: $y_A = a^{x_A} \bmod q$

ElGamal Digital Signature

- Alice signs a message M to Bob by computing
  - the hash $m = H(M)$, $0 <= m <= (q-1)$
  - chose random integer K with $1 <= K <= (q-1)$ and $\gcd(K,q-1)=1$
  - compute temporary key: $S_1 = a^k \bmod q$
  - compute K^{-1} the inverse of K mod (q-1)
  - compute the value: $S_2 = K^{-1}(m-x_A S_1) \bmod (q-1)$
  - signature is:(S_1,S_2)
- any user B can verify the signature by computing
  - $V_1 = a^m \bmod q$
  - $V_2 = y_A^{S1} S_1^{S2} \bmod q$
  - signature is valid if $V_1 = V_2$

ElGamal Signature Example

- use q=19 and a=10
- Alice computes her key:
  - A chooses x_A=16 & computes $y_A=10^{16} \bmod 19 = 4$
- Alice signs message with hash m=14 as (3,4):
  - choosing random K=5 which has gcd(5,18)=1
  - computing $S_1 = 10^5 \bmod 19 = 3$
  - finding $K^{-1} \bmod (q-1) = 5^{-1} \bmod 18 = 11$
  - computing $S_2 = 11(14-16*3) \bmod 18 = 4$
- any user B can verify the signature by computing
  - $V_1 = 10^{14} \bmod 19 = 16$
  - $V_2 = 4^3.3^4 = 5184 = 16 \bmod 19$
  - since 16 = 16 signature is valid

ElGamal Signature Example

- use q=71 and a=7
- Alice computes her key:
  - A chooses x_A=16 & computes $y_A=7^{16}$ mod 71 = 19
- Alice signs message with m=15 :
  - choosing random K=31 which has gcd(31,70)=1
  - computing $S_1 = 7^{31}$ mod 71 = 11
  - finding K^{-1} mod (q-1) = 31^{-1} mod 70 = 61
  - computing S_2 = 61(15-16*11) mod 70 = 49
- any user B can verify the signature by computing
  - $V_1 = 7^{15}$ mod 71 = 23
  - $V_2 = 19^{11}.11^{49}$ mod 71 = 23
  - since 23 = 23 signature is valid

Schnorr Digital Signatures

- also uses exponentiation in a finite (Galois)
  - security based on discrete logarithms, as in D-H
- minimizes message dependent computation
  - multiplying a 2*n-bit* integer with an *n-bit* integer
- main work can be done in idle time
- have using a prime modulus *p*
  - *p–1* has a prime factor *q* of appropriate size
  - typically *p* 1024-bit and *q* 160-bit numbers

Schnorr Key Setup

- choose suitable primes *p* , *q*
- choose *a* such that $a^q = 1 \bmod p$
- (a,p,q) are global parameters for all
- each user (eg. A) generates a key
  - chooses a secret key (number): $0 < s < q$
  - compute their **public key**: $v_A = a^{-s} \bmod q$

Schnorr Signature

- user signs message by
  - choosing random r with 0<r<q and computing $x = a^r \bmod p$
  - concatenate message with x and hash result to computing: $e = H(M \| x)$
  - computing: $y = (r + se) \bmod q$
  - signature is pair (e, y)
- any other user can verify the signature as follows:
  - computing: $x' = a^y v^e \bmod p$
  - verifying that: $e = H(M \| x')$

The first part of this scheme is the generation of a private/public key pair, which consists of the following steps.

1. Choose primes p and q, such that q is a prime factor of $p - 1$.
2. Choose an integer a, such that $a^q = 1 \bmod p$. 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.

A user with private key s and public key v 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).

Any other user can verify the signature as follows.

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

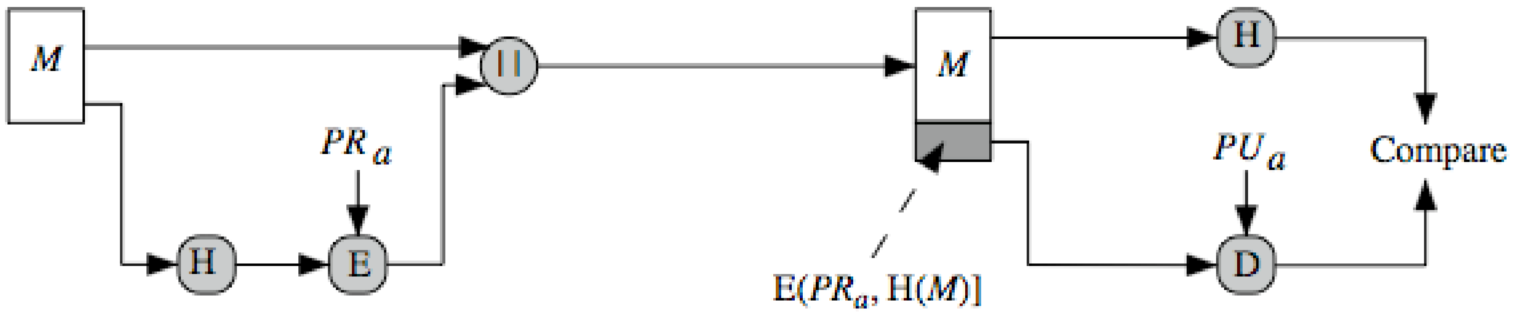
$$x' \equiv a^y v^e \equiv a^y a^{-se} \equiv a^{y-se} \equiv a^r \equiv x \pmod p$$

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

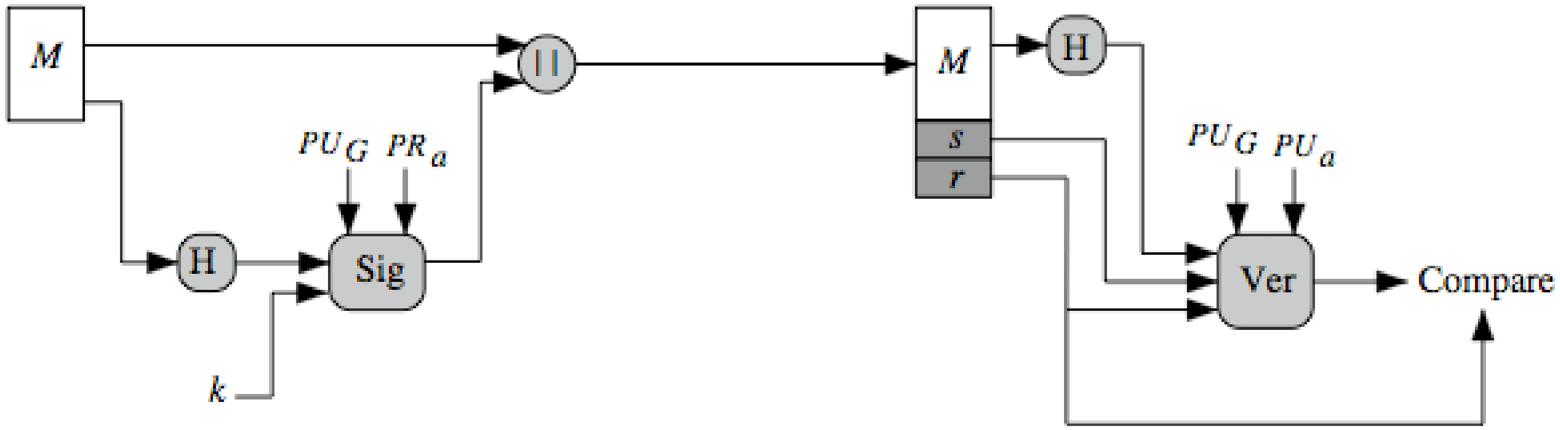
Digital Signature Standard (DSS)

- US Govt approved signature scheme
- designed by NIST & NSA in early 90's
- published as FIPS-186 in 1991
- revised in 1993, 1996 & then 2000
- uses the SHA hash algorithm
- DSS is the standard, DSA is the algorithm
- FIPS 186-2 (2000) includes alternative RSA & elliptic curve signature variants
- DSA is digital signature only unlike RSA
- is a public-key technique

DSS vs RSA Signatures



(a) RSA Approach



(b) DSS Approach

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
- variant of ElGamal & Schnorr schemes

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
  - Number, 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:

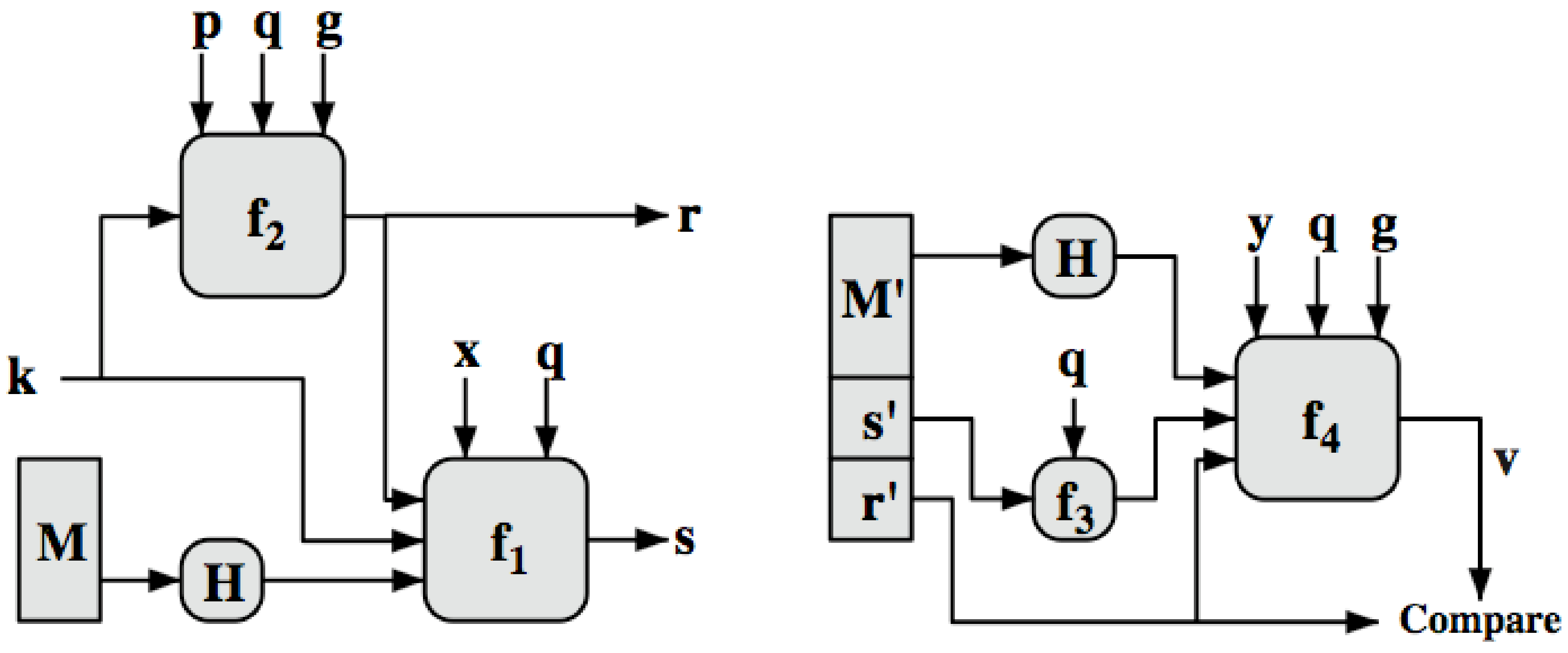
  $w = s^{-1} \bmod q$

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

  $u2= (rw)\bmod q$

  $v = [(g^{u1}\ y^{u2})\bmod p\ ]\bmod q$
- 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

Global Public-Key Components

p prime number where $2^{L-1} < p < 2^L$
for $512 \le L \le 1024$ and L a multiple of 64;
i.e., bit length L between 512 and 1024 bits
in increments of 64 bits

q prime divisor of $(p-1)$, where $2^{N-1} < q < 2^N$
i.e., bit length of N bits

g $= h(\mathrm{p}-1)/\mathrm{q}$ is an exponent mod p,
where h is any integer with $1 < h < (p-1)$
such that $h^{(p-1)/q} \bmod p > 1$

User's Private Key

x random or pseudorandom integer with $0 < x < q$

User's Public Key

$y = g^x \bmod p$

User's Per-Message Secret Number

k random or pseudorandom integer with $0 < k < q$

Signing

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

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

Signature $= (r, s)$

Verifying

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

$u_1 = [H(M')w] \bmod q$

$u_2 = (r')w \bmod q$

$v = [(g^{u1}y^{u2}) \bmod p] \bmod q$

TEST: $v = r'$

M	= message to be signed
H(M)	= hash of M using SHA-1
M', r', s'	= received versions of M, r, s

Figure 13.3 The Digital Signature Algorithm (DSA)

Summary

- have discussed:
  - digital signatures
  - ElGamal & Schnorr signature schemes
  - digital signature algorithm and standard