Digital Signature Algorithm (DSA)

INDIAN INSTITUTE OF INFORMATION TECHNOLOGY SRI CITY

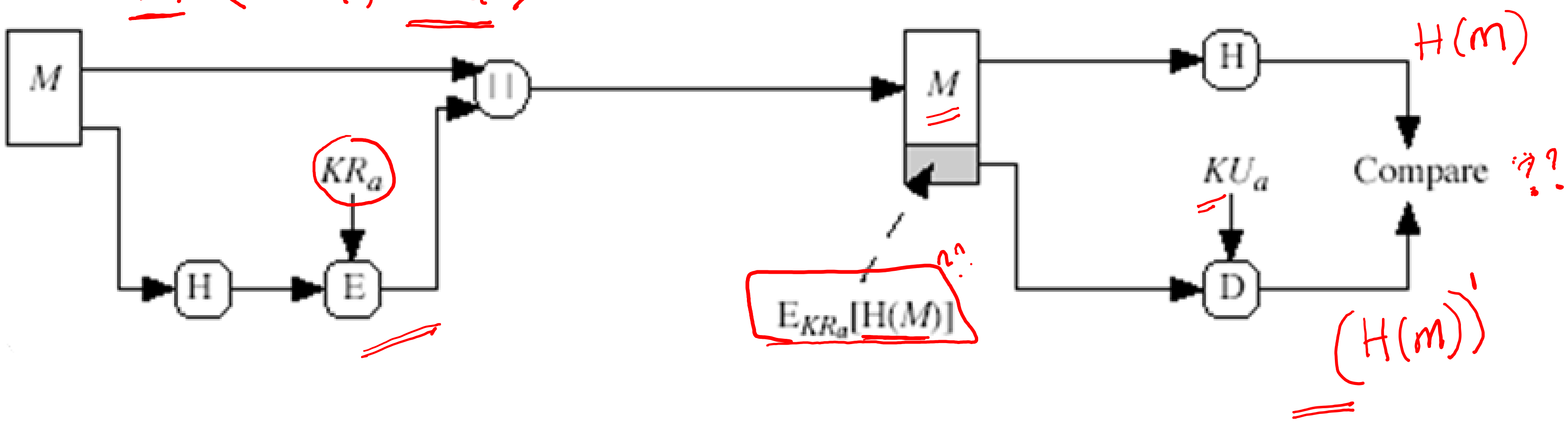
CHITTOOR, INDIA

Digital Signature

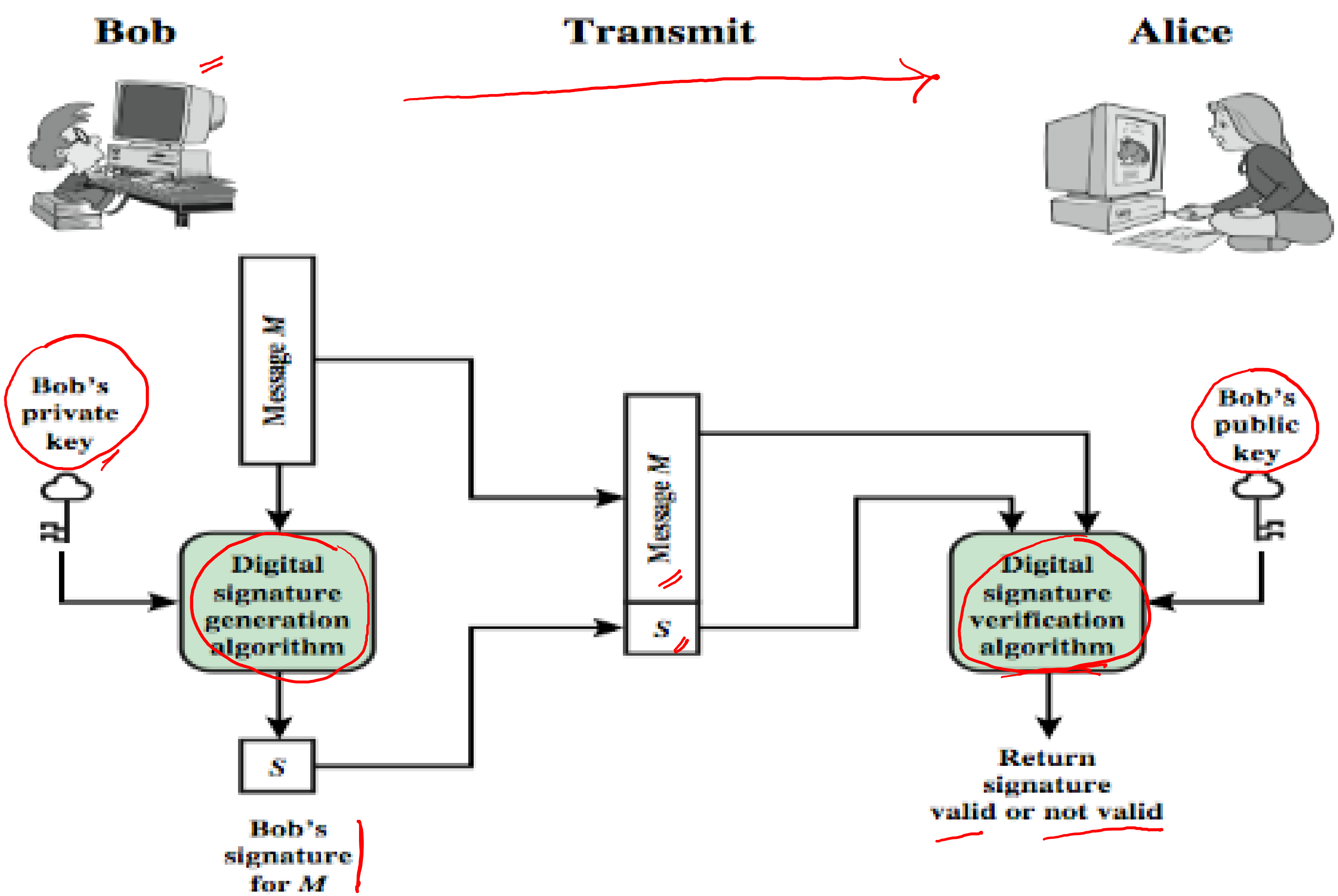
- Usually assume that the hash function is public and not keyed
  - eg. MAC which is keyed
- Hash is used to detect changes to message
- Most often to create a digital signature

A $\langle KR_a, KU_a \rangle$

B



Digital Signature Model



Digital Signature Requirements

- Information unique to sender
  - to prevent both forgery and denial

A – (m, s)

B – (m, s')

- Relatively easy to produce.

- Relatively easy to recognize & verify.

- Computationally infeasible to forge

- **Practical:** save digital signature in storage

Direct Digital Signatures

(Involve only sender & receiver)

A $\xrightarrow{(S, m)}$ B

(Signer) (Verifier)

- Assumed receiver has sender's public-key
- Digital signature made by sender signing entire message or hash with private-key and encrypt using receivers public-key.
- Important that sign first then encrypt message & signature.
- Security depends on sender's private-key

ElGamal Digital Signatures

- Signature variant of ElGamal, related to D-H
  - uses exponentiation in a finite field (Galois)
  - with security based-on difficulty of computing DLP
- 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**: $y_A = g^{x_A} \bmod q$

Alice's Set up:
secret key : $1 < x_A < q-1$
public key: $y_A = g^{x_A} \bmod q$

ElGamal Digital Signature

- Alice signs a message M to Bob as follows:
  - Compute $m = H(M)$, $\boxed{0 <= m <= (q-1)}$
  - Chose K with $1 <= K <= (q-1)$ and $\gcd(K, q-1) = 1$
  - Compute temporary key: $S_1 = g^K \bmod q$
  - Compute K^{-1} the inverse of $K \bmod (q-1)$
  - Compute signature: $S_2 = K^{-1}(m - x_A S_1) \bmod (q-1)$
  - Signature is: (S_1, S_2)

$K S_2 = m - x_A S_1 \Rightarrow \boxed{m = K S_2 + x_A S_1} \bmod q-1$

$V_2 = y_A^{S_1} \times S_1^{S_2} = g^{x_A S_1} \times g^{K S_2} \pmod q$

- Any user B can verify the signature as follows:
  - Compute $V_1 = g^m \bmod q$
  - Compute $V_2 = y_A^{S_1} S_1^{S_2} \bmod q$
  - Verify validity of $V_1 = V_2$ (valid if equality holds)

$$\boxed{g^m = g^{K S_2 + x_A S_1}}$$

ElGamal Signature Example

$m = H(m)$

Use field GF(19) $q=19$ and $g=10$

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 follows:

- choosing random $K=5$ which has $\gcd(18,5)=1$

CPQ-08

ElGamal Signature Example

Use field GF(19) $q=19$ and $g=10$

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 as follows:

- choosing random $K=5$ which has $\gcd(18,5)=1$
- computing $S_1 = 10^5 \bmod 19 = 3$
- finding $K^{-1} \bmod (q-1) = 5^{-1} \bmod 18 = 11$
- computing $S_2 = 11(14-16x3) \bmod 18 = 4$

Any user B can verify signature (3, 4) on 14 as follows:

- $V_1 = 10^{14} \bmod 19 = 16$
- $V_2 = 4^3 x 3^4 = 5184 = 16 \bmod 19$
- since $16 = 16$ signature is valid

Schnorr Digital Signatures

- Uses exponentiation in a finite field (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
- Use a prime modulus p
  - $p-1$ has a prime factor q of appropriate size
  - typically p 1024-bit and q 160-bit numbers

Note: Use 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 g such that $g^q = 1 \bmod p$
- (g,p,q) are global parameters for all

- Each user, say A, generates a key pair
  - Choose **secret key**: $0 < s < q$
  - Compute **public key**: $v = g^{-s} \bmod p$

Schnorr Signature

- **User signs message as follows:**
  - Choose random r with $0<r<q$
  - Compute $x = g^r \bmod p$
  - Compute: $e = H(M||x)$
  - Compute: $y = (r + se) \bmod q$
  - Signature : $(\mathbf{e},\mathbf{y})$
- **Any user can verify the signature as follows:**
  - Computing: $x' = g^y v^e \bmod p$
  - Verifying that: $e = H(M || x')$

$g^{(r+se)} \; (g^{-s})^e$

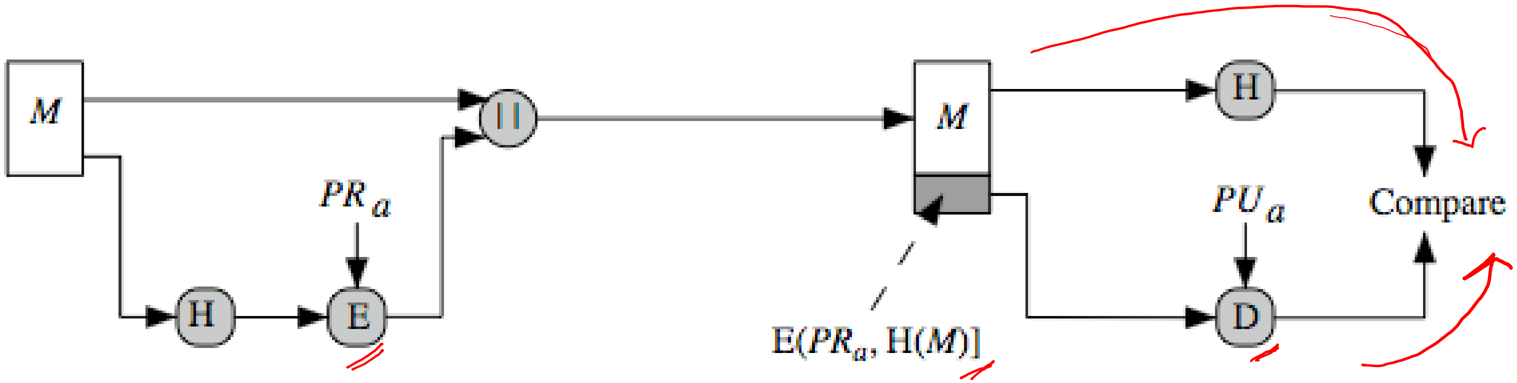
$x' \stackrel{?}{=} g^r$

$e \stackrel{?}{=} H(m \| x')$

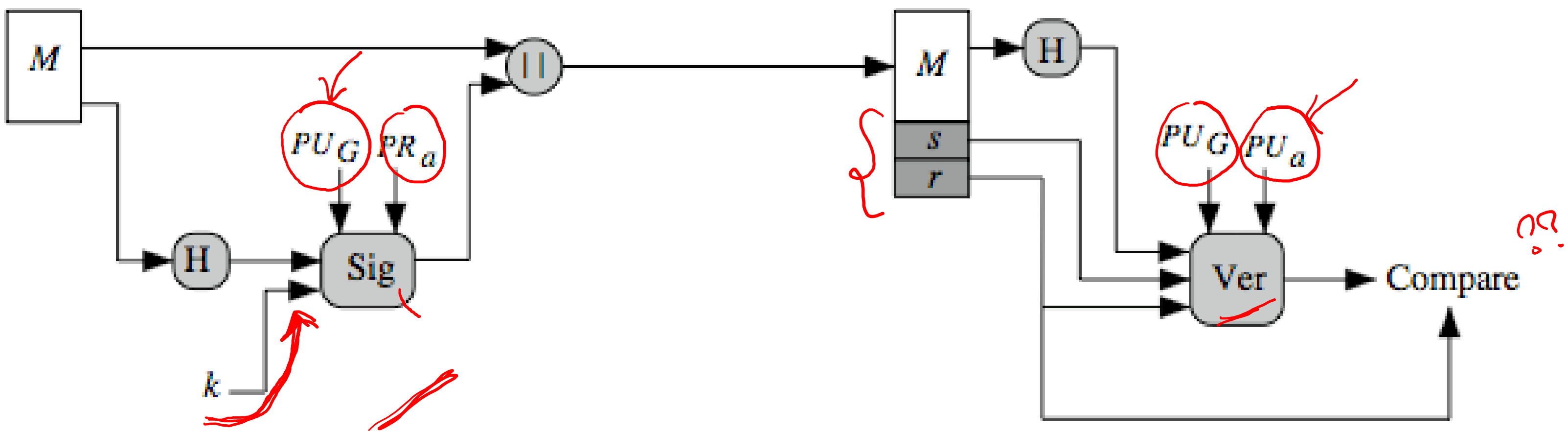
Digital Signature Standard (DSS)

- 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, and DSA is the algorithm
- FIPS 186-2 (2000) includes alternative RSA & elliptic curve signature variants
- **DSA is digital signature only, but not public-key technique like RSA.**

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 and digital signature scheme only
- Security depends on difficulty of computing DLP
- Variant of ElGamal & Schnorr schemes

DSA Key Generation

- **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: $y = g^x \bmod p$

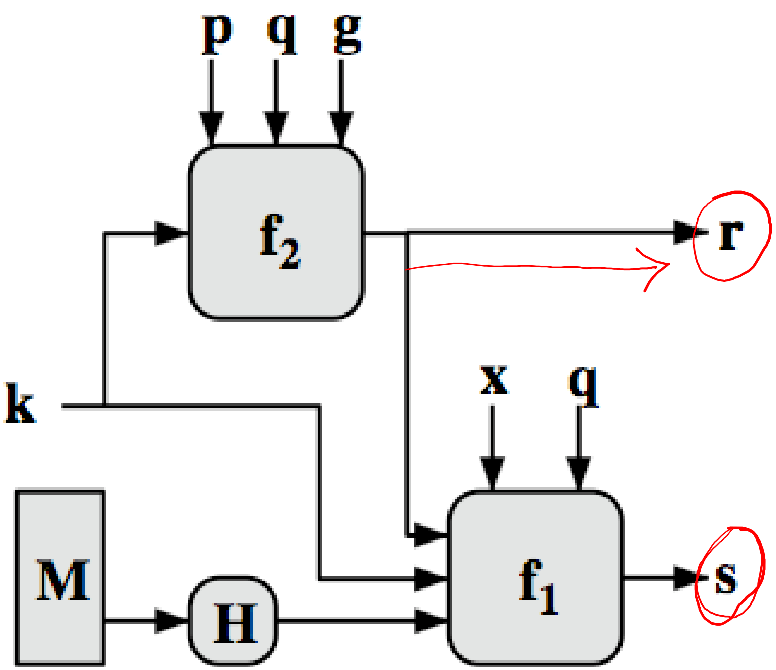
DSA Signature Creation

- To **sign** a message M the sender:
  - generates a random signature key k, $k<q$
  - Note. **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

- 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

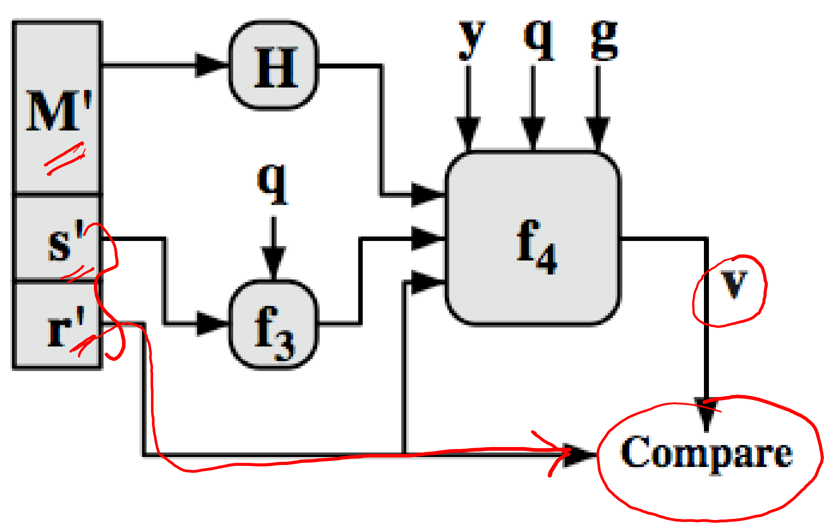
DSS Overview



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

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

(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 ?

THANK YOU