Plan of Talk

- ElGamal Cryptosystem
- ElGamal Digital Signature
- Digital Signature Standard

ElGamal Cryptography

- public-key cryptosystem related to D-H
- so uses exponentiation in a finite (Galois)
- with security based difficulty of computing discrete logarithms, as in D-H
- 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 Message Exchange

- Bob encrypt a message to send to A computing
  - represent message M in range $0 \le M \le q-1$
    - longer messages must be sent as blocks
  - chose random integer k with $1 \le k \le q-1$
  - compute one-time key $K = y_A^k \bmod q$
  - encrypt M as a pair of integers (C_1, C_2) where
    - $C_1 = a^k \bmod q$; $C_2 = KM \bmod q$
- A then recovers message by
  - recovering key K as $K = C_1^{xA} \bmod q$
  - computing M as $M = C_2 K^{-1} \bmod q$
- a unique k must be used each time
  - otherwise result is insecure

ElGamal Example

- use field GF(19) $q=19$ and $a=10$
- Alice computes her key:
  - A chooses $x_A=5$ & computes $y_A=10^5 \bmod 19 = 3$
- Bob send message $m=17$ as $(11,5)$ by
  - chosing random $k=6$
  - computing $K = y_A^k \bmod q = 3^6 \bmod 19 = 7$
  - computing $C_1 = a^k \bmod q = 10^6 \bmod 19 = 11$;
    $C_2 = KM \bmod q = 7.17 \bmod 19 = 5$
- Alice recovers original message by computing:
  - recover $K = C_1^{xA} \bmod q = 11^5 \bmod 19 = 7$
  - compute inverse $K^{-1} = 7^{-1} = 11$
  - recover $M = C_2 \ K^{-1} \bmod q = 5.11 \bmod 19 = 17$

ElGamal Digital Signatures

- 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 $K \bmod (q-1)$
  - 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
  - $V_1 = a^m \bmod q$
  - $V_2 = y_A^{S_1} S_1^{S_2} \bmod q$
  - signature is valid if $V_1 = V_2$

ElGamal Signature Example

- use field GF(19) $q=19$ 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 $\gcd(18,5)=1$
  - computing $S_1 = 10^5 \bmod 19 = 3$
  - finding $K^{-1} \bmod (q-1) = 5^{-1} \bmod 18 = 11$
  - computing $S_2 = 11(14-16.3) \bmod 18 = 4$
- 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

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): $0 < s_A < q$
  - compute their **public key**: $v_A = a^{-s_A} \bmod q$

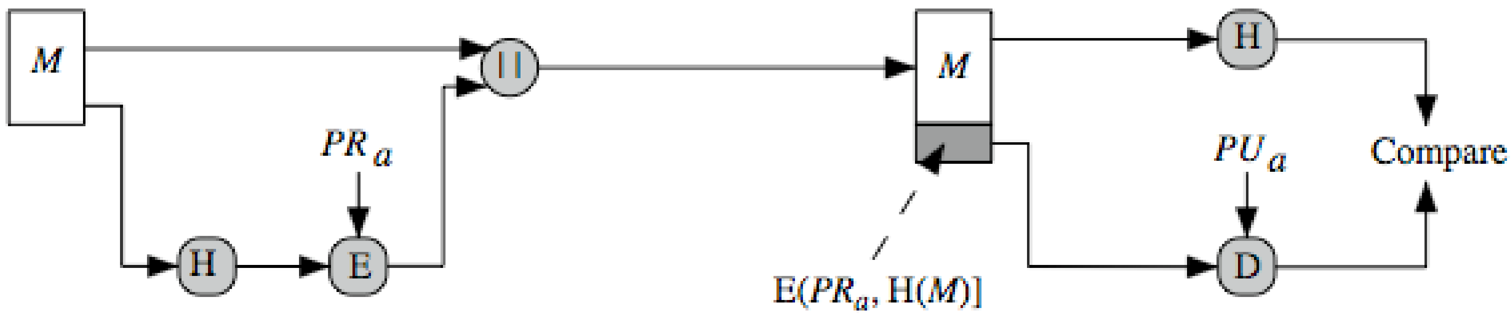
Schnorr Signature

- 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: $e = H(M \| x')$

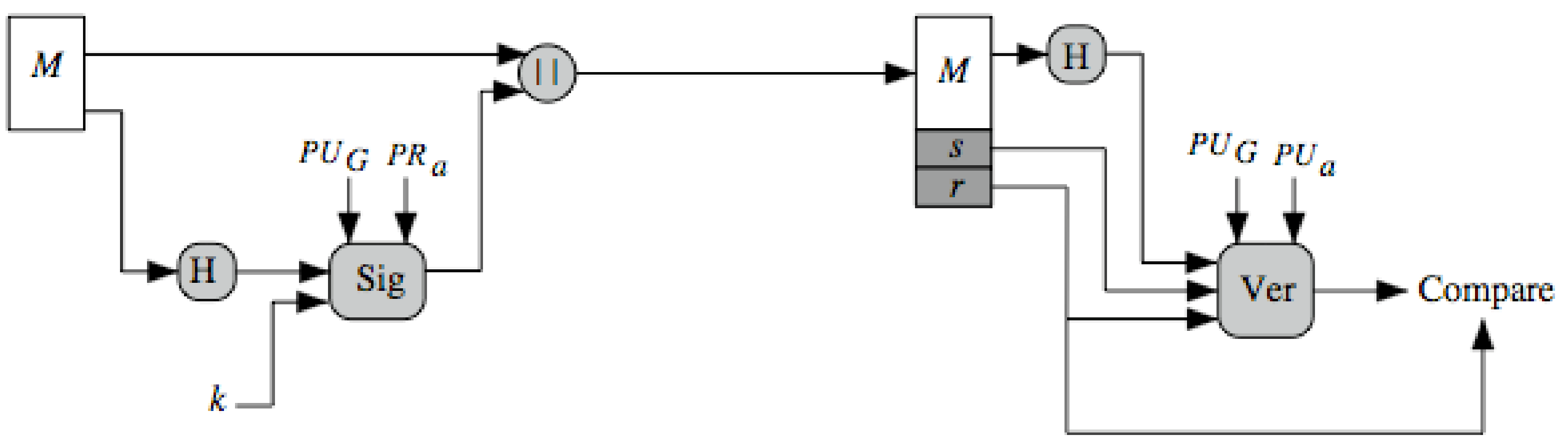
Digital Signature Standard (DSS)

- 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
- It is a 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`
  - nb. k must be random, be destroyed after use, and never be reused
- to **sign** 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:

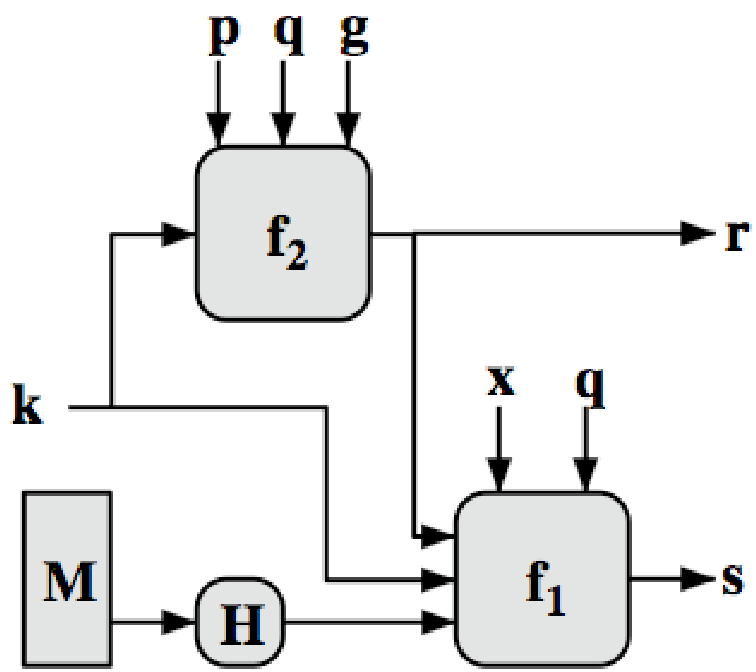
  $w = s^{-1} \bmod q$

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

  $u2 = (rw) \bmod q$

  $v = [(g^{u1} y^{u2}) \bmod p] \bmod q$
- if `v=r` then signature is verified
- see Appendix A for details of proof why

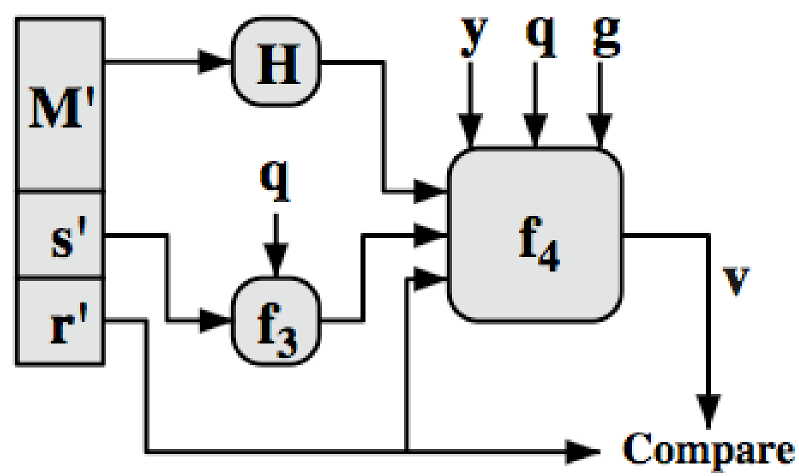
DSS Overview



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

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

(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