Digital Signatures

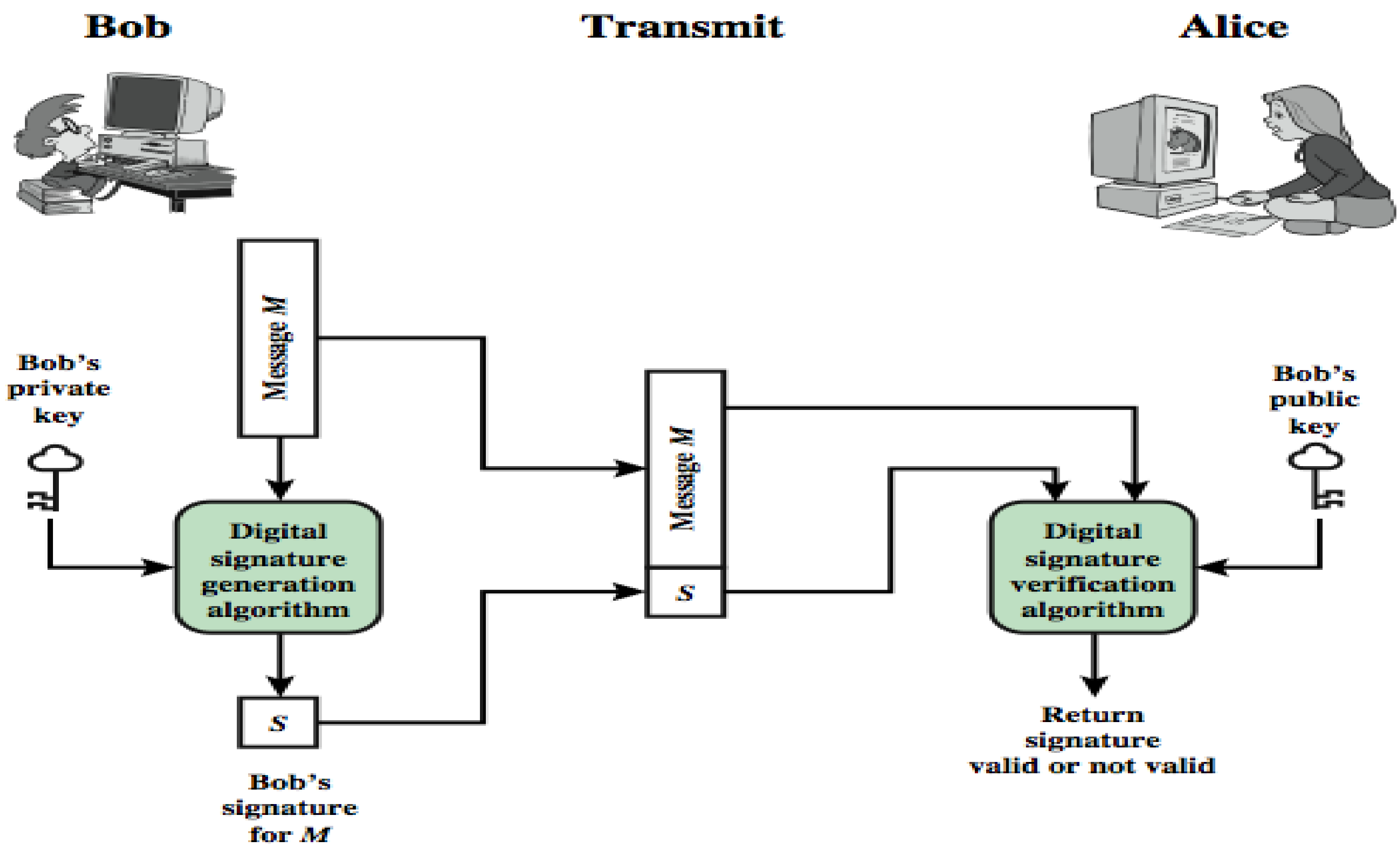
- The most important development from the work on public-key cryptography is the digital signature.
- Message authentication protects two parties who exchange messages from any third party. However, it does not protect the two parties against each other denying creation of a message.
- A digital signature is similar to the handwritten signature, and provides a set of security capabilities that would be difficult to implement in any other way. It must have the following properties:
  - It must verify the author and the date and time of the signature
  - It must to authenticate the contents
  - It must be verifiable by third parties, to resolve disputes

Thus, the digital signature function includes the authentication function.

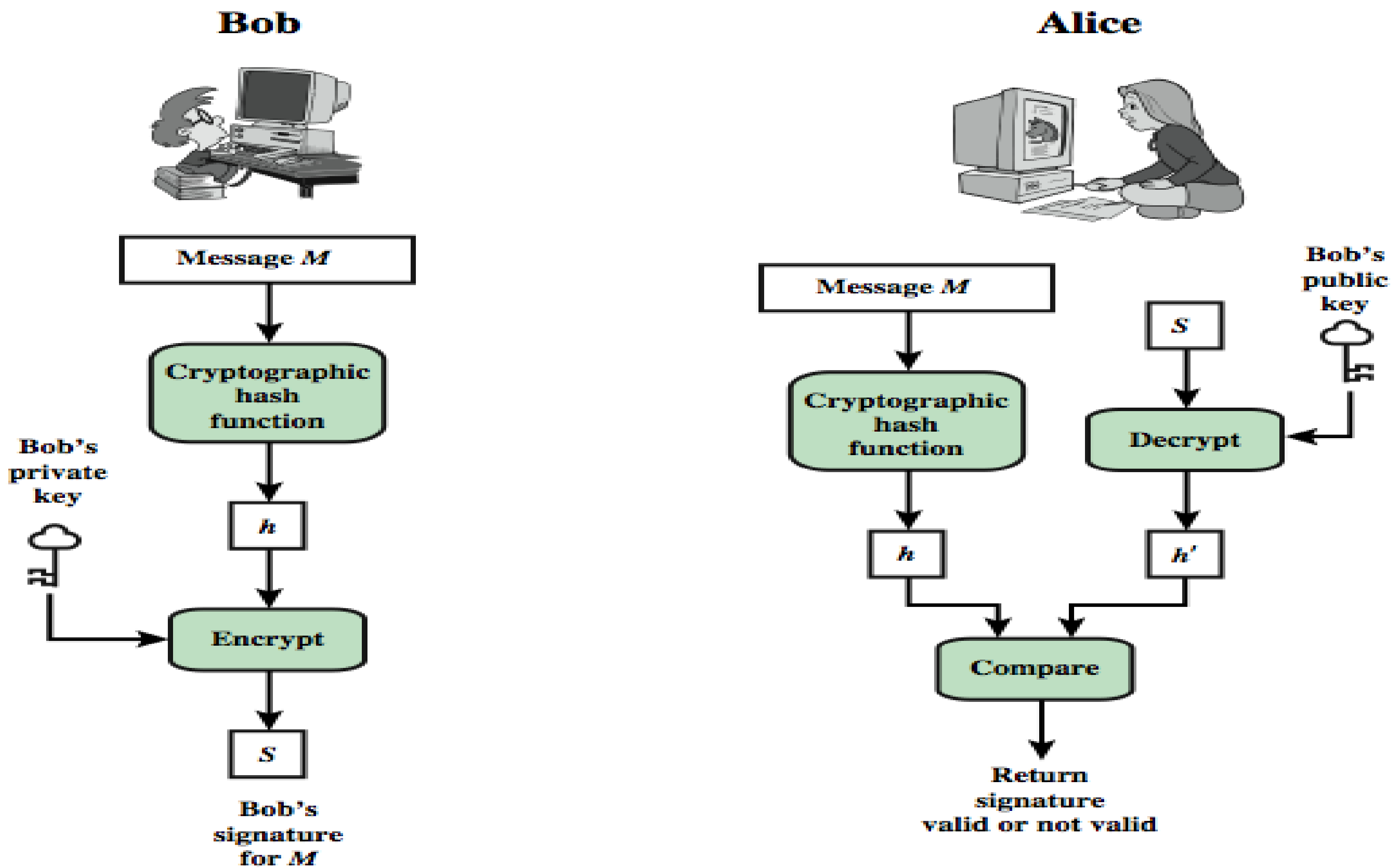
Digital Signatures

- 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 and Forgeries

- attacks
  - key-only attack
  - known message attack
  - generic chosen message attack
  - directed chosen message attack
  - adaptive chosen message attack
- break success levels
  - total break
  - selective forgery
  - existential forgery

Types of attacks:

Note: A denotes the user whose signature is being attacked and C denotes the attacker.

- **Key-only attack**: C only knows A's public key.
- **Known message attack**: C is given access to a set of messages and signatures.
- **Generic chosen message attack**: C chooses a list of messages before attempting to breaks A's signature scheme, independent of A's public key. C then obtains from A valid signatures for the chosen messages. The attack is generic because it does not depend on A's public key.
- **Directed chosen message attack**: Similar to the generic attack, except that the list of messages is chosen after C knows A's public key but before signatures are seen.
- **Adaptive chosen message attack**: This means the C may request signatures of messages that depend on previously obtained message-signature pairs.

Break success levels:

- **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 at least one message. C has no control over the message. ~~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 Scheme

- Direct Digital Signatures involve the direct application of public-key algorithms involving only the communicating parties (sender & receiver)
- assumed receiver has sender's public-key
- A digital signature may be formed by encrypting the entire message with the sender's private key, or by encrypting a hash code of the message with the sender's private key.
- Confidentiality can be provided by further encrypting the entire message plus signature using either public or private key schemes.
- can encrypt using receivers public-key
- important that sign first then encrypt message & signature
- security depends on sender's private-key

ElGamal Digital Signatures

- It's 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)
- The global elements of ElGamal are a prime number q and a, which is a primitive root of q.
- each user (e.g., A) generates their key
  - chooses a secret key (number): 1 < xA < q-1
  - compute their public key: yA = axA mod q

1. Generate a random integer X_A, such that $1 < X_A < q - 1$.
2. Compute $Y_A = \alpha^{X_A} \bmod q$.
3. A's private key is X_A; A's pubic key is $\{q, \alpha, Y_A\}$.

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: S1 = ak mod q
  - compute K-1 the inverse of K mod (q-1)
  - compute the value: S2 = K-1(m-xAS1) mod (q-1)
  - signature is:(S1,S2)

To sign a message M, user A first computes the hash $m = \mathrm{H}(M)$, such that m is an integer in the range $0 \le m \le q - 1$. A then forms a digital signature as follows.

1. Choose a random integer K such that $1 \le K \le q - 1$ and $\gcd(K, q - 1) = 1$. That is, K is relatively prime to $q - 1$.
2. Compute $S_1 = \alpha^K \bmod q$. Note that this is the same as the computation of C_1 for Elgamal encryption.
3. Compute $K^{-1} \bmod (q - 1)$. That is, compute the inverse of K modulo $q - 1$.
4. Compute $S_2 = K^{-1}(m - X_A S_1) \bmod (q - 1)$.
5. The signature consists of the pair (S_1, S_2).

- any user B can verify the signature by computing
  V1 = am mod q
  V2 = yAS1 S1S2 mod q
  signature is valid if V1 = V2

Any user B can verify the signature as follows.

1. Compute $V_1 = \alpha^m \bmod q$.
2. Compute $V_2 = (Y_A)^{S_1}(S_1)^{S_2} \bmod q$.

   The signature is valid if $V_1 = V_2$. ~~Let us demonstrate that this is so. Assume~~

ElGamal Signature Example

- use field GF(19) q=19 and a=10
- Alice computes her key:
  - A chooses xA=16 & computes yA=1016 mod 19 = 4
- Alice signs message with hash m=14 as (3,4):
  - choosing random K=5 which has gcd(18,5)=1
  - computing S1 = 105 mod 19 = 3
  - finding K-1 mod (q-1) = 5-1 mod 18 = 11
  - computing S2 = 11(14-16.3) mod 18 = 4
- any user B can verify the signature by computing
  - V1 = 1014 mod 19 = 16
  - V2 = 43.34 = 5184 = 16 mod 19
  - since 16 = 16 signature is valid

For example, let us start with the prime field GF(19); that is, $q = 19$. It has primitive roots {2, 3, 10, 13, 14, 15}, as shown in Table 2.7. We choose $\alpha = 10$.

Alice generates a key pair as follows:

1. Alice chooses $X_A = 16$.
2. Then $Y_A = \alpha^{X_A} \bmod q = \alpha^{16} \bmod 19 = 4$.
3. Alice's private key is 16; Alice's pubic key is $\{q, \alpha, Y_A\} = \{19, 10, 4\}$.

Suppose Alice wants to sign a message with hash value $m = 14$.

1. Alice chooses $K = 5$, which is relatively prime to $q - 1 = 18$.
2. $S_1 = \alpha^K \bmod q = 10^5 \bmod 19 = 3$ (see Table 2.7).
3. $K^{-1} \bmod (q - 1) = 5^{-1} \bmod 18 = 11$.
4. $S_2 = K^{-1}(m - X_A S_1) \bmod (q - 1) = 11\,(14 - (16)(3)) \bmod 18 = -374 \bmod 18 = 4$.

Bob can verify the signature as follows.

1. $V_1 = \alpha^m \bmod q = 10^{14} \bmod 19 = 16$.
2. $V_2 = (Y_A)^{S_1}(S_1)^{S_2} \bmod q = (4^3)(3^4) \bmod 19 = 5184 \bmod 19 = 16$.

Thus, the signature is valid because $V_1 = V_2$.

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 2n-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 aq = 1 mod p
- (a,p,q) are global parameters for all

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.

- each user (e.g., A) generates a key
  - chooses a secret key (number): 0 < s < q
  - compute their public key: v = a-s mod p

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 Signature

- user signs message by
  - choosing random r with 0<r<q and computing x = ar mod p
  - concatenate message with x and hash result to computing:

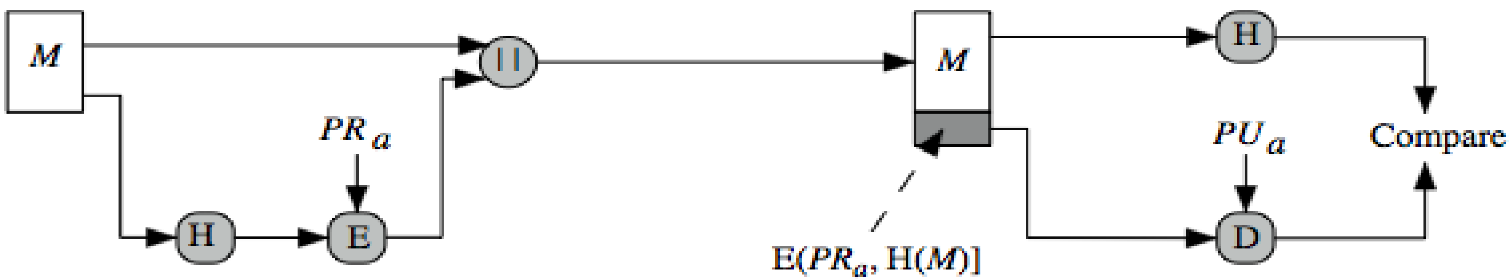
  e = H(M || x) Choose a random integer r with $0 < r < q$ and compute $x = a^r \bmod p$

  - computing: y = (r + se) mod q
  - signature is pair (e, y)
- any other user can verify the signature as follows:
  - computing: $x' \equiv a^y v^e$ mod p
  - verifying that: H (M || x') = H (M || x).

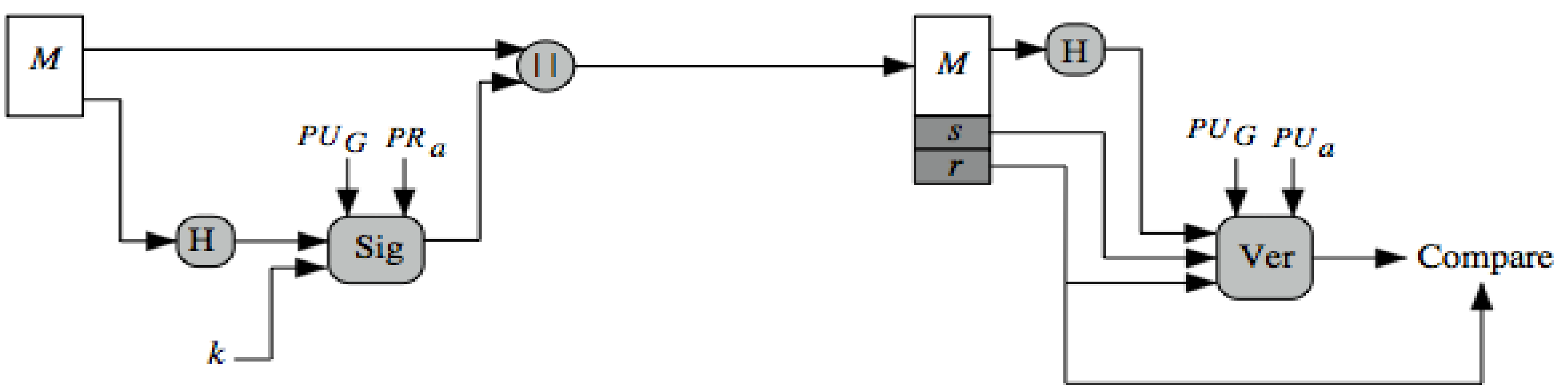
Digital Signature Standard (DSS)

- US Government 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 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 2L-1 < p < 2L

    p prime number where $2^{L-1} < p < 2^L$

  - choose g = h(p-1)/q

    g= $h^{(p-1)/q} \bmod p$
- users choose private & compute public key:
  - choose random private key: x<q
  - compute public key: y = gx mod p

    $y = g^x \bmod p$

DSA Signature Creation

- to sign a message M the sender:
  - generates a random signature key k, k<q
  - k must be random, be destroyed after use, and never be reused
- then computes signature pair:
  - r = (gk mod p)mod q
  - s = [k-1(H(M)+ xr)] mod q
- sends signature (r,s) with message M

Signing

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

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

Signature $= (r, s)$

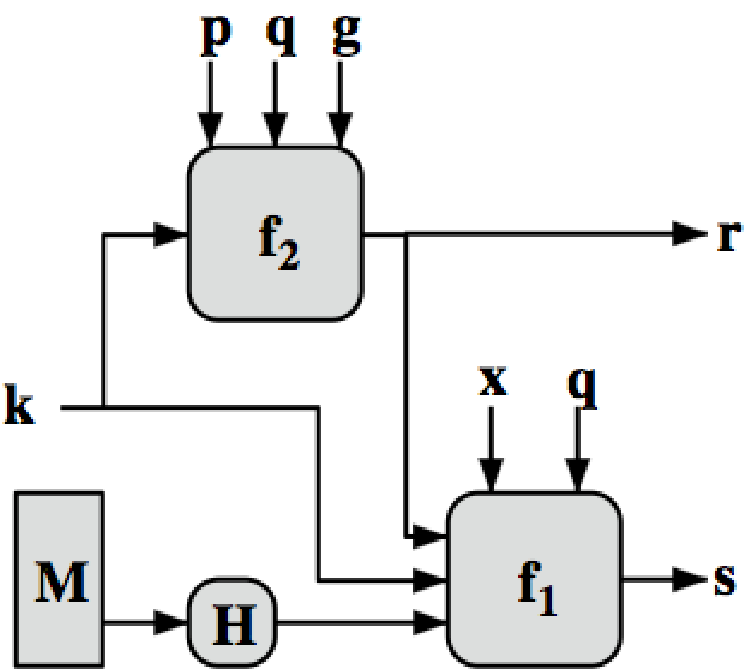
DSA Signature Verification

- having received M & signature (r,s)
- to verify a signature, recipient computes:
  - w = s-1 mod q $w = (s')^{-1} \bmod q$
  - u1= [H(M)w]mod q
  - u2= (rw)mod q
  - v = [(gu1 yu2)mod p]mod q

$$u_1 = [\mathrm{H}(M')w)] \bmod q$$
$$u_2 = (r')w \bmod q$$
$$v = [(g^{u_1}y^{u_2}) \bmod \mathrm{p}] \bmod q$$

- if v=r then signature is verified
- ~~see Appendix A for details of proof why~~

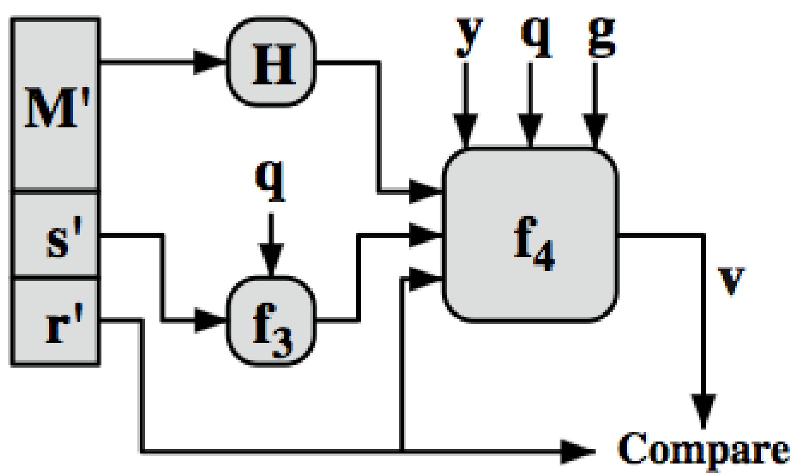
DSS Overview



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

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

(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