Cryptography and Computer Security (CSS)

Lecture # 9

INTRODUCTION
TO
CSS COURSE
(CS401)

Unit III
Digital Signatures

- Digital Signature
- Digital Signature properties
- Requirements and security
- Various digital signature schemes (Elgamal and Schnorr)
- Digital Signature algorithm / Digital Signature Standard

Cryptographic Goals

Cryptographic goals

Confidentiality

Symmetric-key ciphers:

- Block ciphers
- Stream ciphers

Public-key ciphers

Data integrity

Arbitrary length hash functions

Message Authentication codes (MACs)

Digital signatures

Authentication

Entity authentication

Authentication primitives

Message authentication

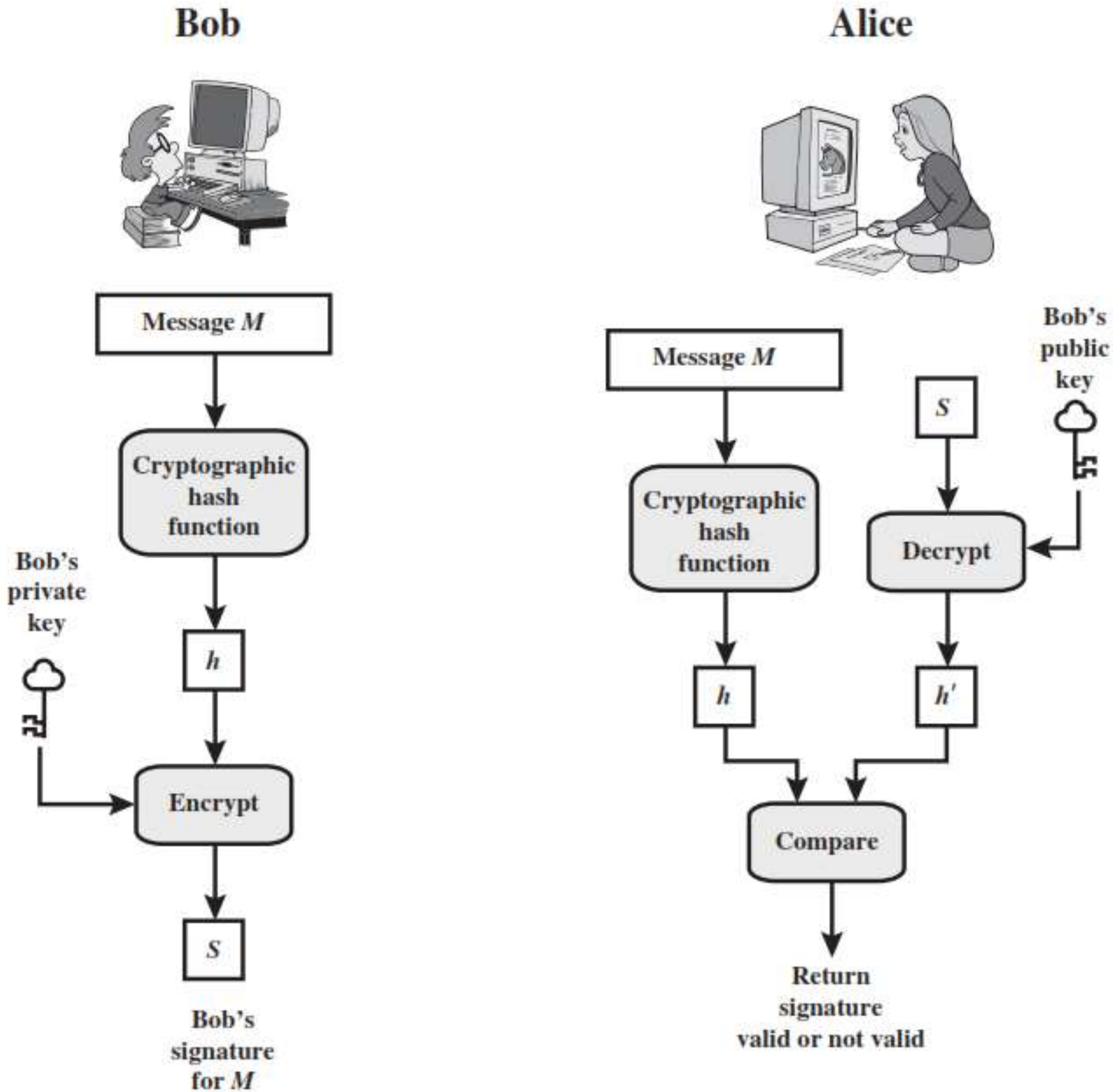
- MACs
- Digital signatures

Non-repudiation

Digital signatures

Digital Signature

- A **digital signature** is an authentication mechanism that enables the creator of a message to attach a code that acts as a **signature**.
- Typically the **signature** is formed by taking the hash of the message and encrypting the message with the creator’s private key.
- The **signature** guarantees the source and integrity of the message.
- The **digital signature standard (DSS)** is an NIST standard that uses the secure hash algorithm (SHA).


Bob
Alice
Message M
Cryptographic hash function
h
Bob's private key
Encrypt
S
Bob's signature for M
Message M
Cryptographic hash function
h
S
Bob's public key
Decrypt
h'
Compare
Return signature valid or not valid

Hash code, MAC and Digital Signature

Hash Code

- A **hash** of the message, if appended to the message itself, only protects against accidental changes to the message, as an attacker who modifies the message can simply calculate a new hash and use it instead of the original one. So this **only gives integrity**.

MAC

- A message authentication code (MAC) (sometimes also known as keyed hash) protects against message forgery by anyone who doesn't know the secret.
- This means that the receiver can forge any message – thus we have both **integrity** and **authentication** (as long as the receiver doesn't have a split personality), **but not non-repudiation**.

Hash code, MAC and Digital Signature

Digital Signature

- A **digital signature** is created with a private key, and verified with the corresponding public key of an asymmetric key-pair.
- Only the holder of the private key can create this signature, and normally anyone knowing the public key can verify it.

Attacks and Forgeries

- **Key-only attack:** C only knows A’s public key.
- **Known message attack:** C is given access to a set of messages and their 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; the same attack is used against everyone.
- **Directed chosen message attack:** Similar to the generic attack, except that the list of messages to be signed is chosen after C knows A’s public key but before any signatures are seen.
- **Adaptive chosen message attack:** C is allowed to use A as an “oracle.” This means the A may request signatures of messages that depend on previously obtained message–signature pairs.

Attacks and 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 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

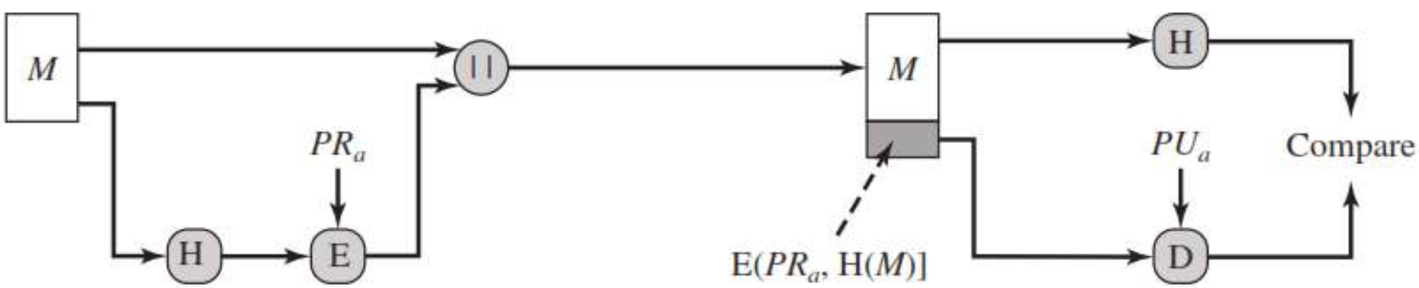
1. The signature must be a **bit pattern** that depends on the message being signed.
2. The signature must use some information **unique** to the sender to prevent both forgery and denial.
3. It must be relatively **easy to produce** the digital signature.
4. It must be relatively **easy to recognize** and **verify** the digital signature.
5. It must be computationally **infeasible to forge** a digital signature, either by constructing a new message for an existing digital signature or by constructing a fraudulent digital signature for a given message.
6. It must be **practical to retain a copy** of the digital signature in storage.

Digital Signature Standard / DSA

- The **DSS** uses an algorithm that is designed to provide only the digital signature function.
- Unlike RSA, it cannot be used for encryption or key exchange.

RSA Approach

- In the **RSA** approach, the message to be signed is input to a hash function that produces a **secure hash code** of fixed length.
- This hash code is then encrypted using the sender's private key to form the **signature**.
- Both the message and the signature are then transmitted.
- The recipient takes the message and produces a **hash code**.



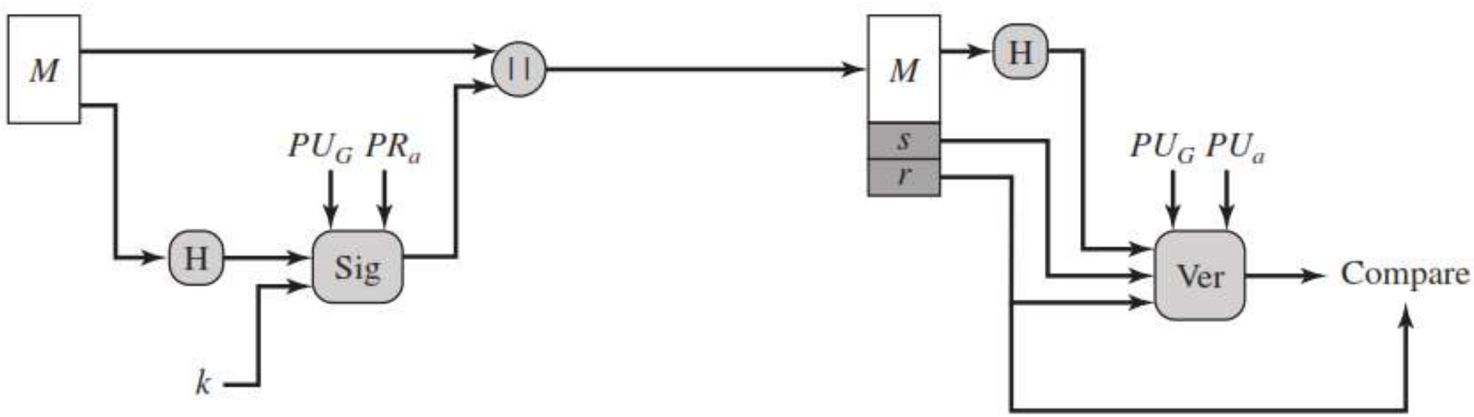
(a) RSA approach

RSA Approach

- The recipient also decrypts the signature using the sender's public key.
- If the calculated hash code matches the decrypted signature, the signature is accepted as valid.
- Because only the sender knows the private key, only the sender could have produced a valid signature.

DSA Approach

- The **hash code** is provided as input to a **signature function** along with a random number ***k*** generated for this particular signature.
- The signature function also depends on the sender's private key **(*PRa*)** and a set of parameters known to a group of communicating principals.
- We can consider this set to constitute a global public key **(*PU*)**
- The result is a signature consisting of two components, labelled ***s*** and ***r***.



(b) DSA approach

DSA Approach

- At the receiving end, the hash code of the incoming message is generated.
- This plus the signature is input to a **verification function**.
- The verification function also depends on the global public key as well as the sender's public key ***(PUa)***, which is paired with the sender's private key.
- The output of the verification function is a value that is equal to the signature component ***r*** *if the signature is valid.*
- *The signature* function is such that only the sender, with knowledge of the private key, could have produced the valid signature.

Digital Signature Algorithm

Global Public-Key Components

p prime number where $2^{L-1} < p < 2^{L}$
for $512 \leq L \leq 1024$ and L a multiple of 64;
i.e., bit length of 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 $= h(p - 1)/q \bmod p$,
where h is any integer with $1 < h < (p - 1)$
such that $h^{(p-1)/q} \bmod p > 1$

Digital Signature Algorithm

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 $0 < k < q$

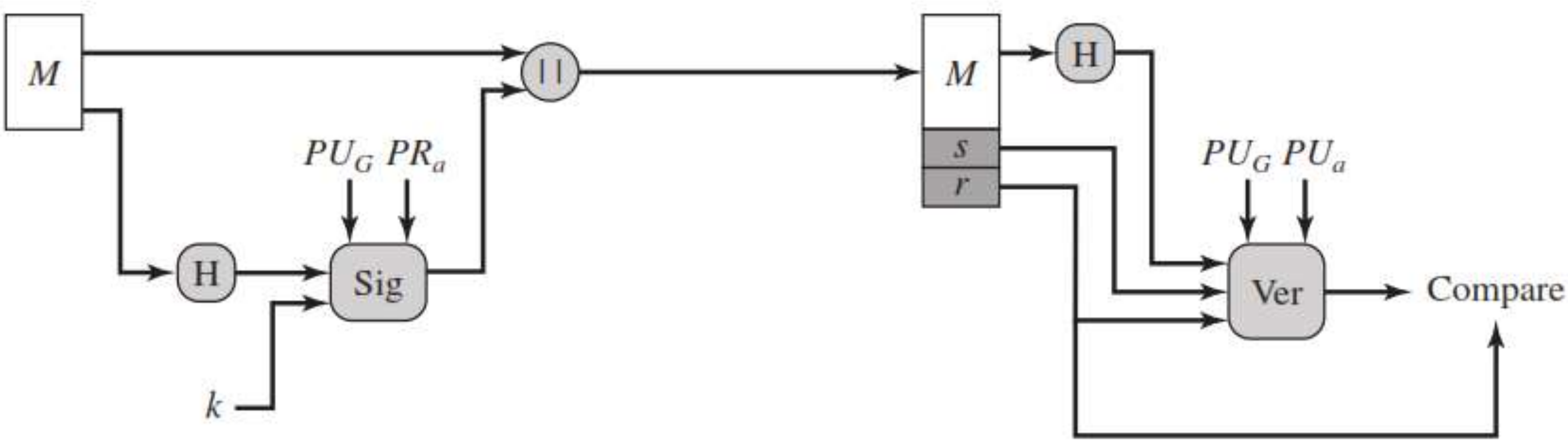
Digital Signature Algorithm

Signing

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

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

$$\text{Signature} = (r, s)$$



Digital Signature Algorithm

Verifying

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

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

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

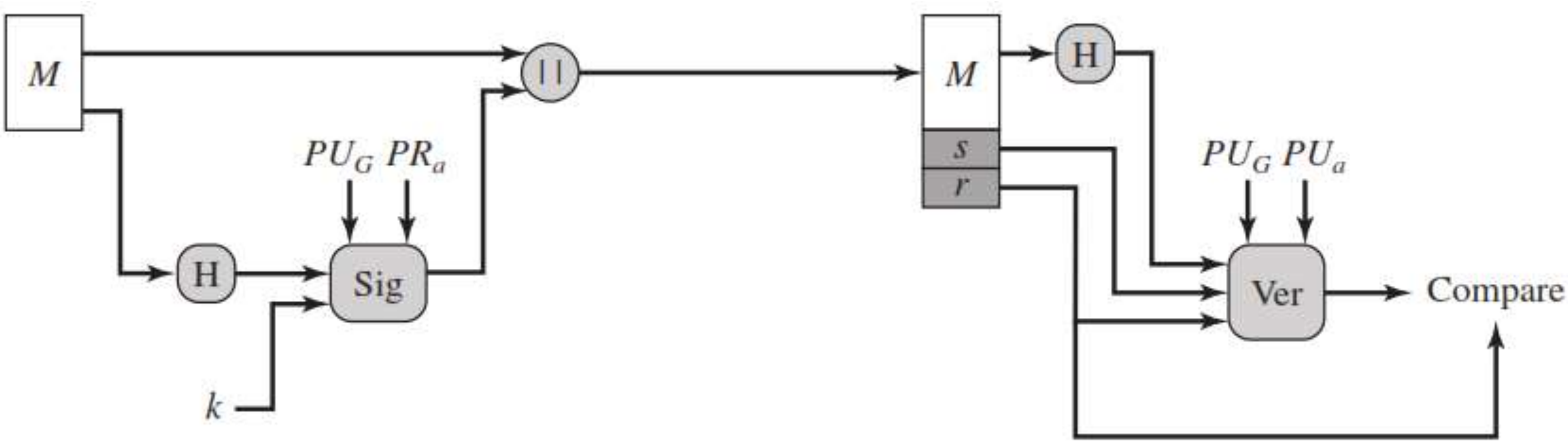
$v = [(g^{u1}\, y^{u2}) \bmod p] \bmod q$

TEST: $v = r'$

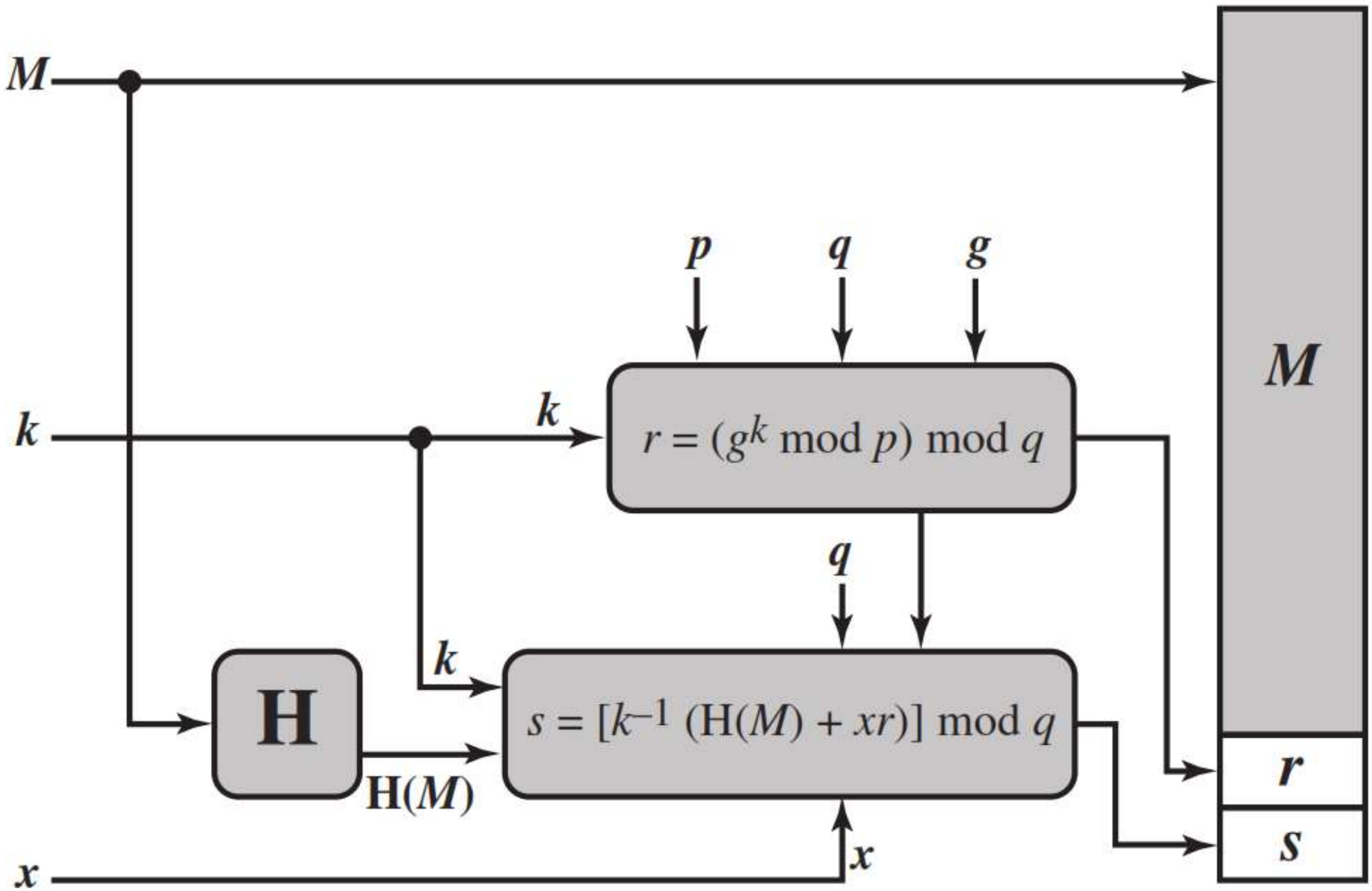
M = message to be signed

$H(M)$ = hash of M using SHA-1

M', r', s' = received versions of M, r, s

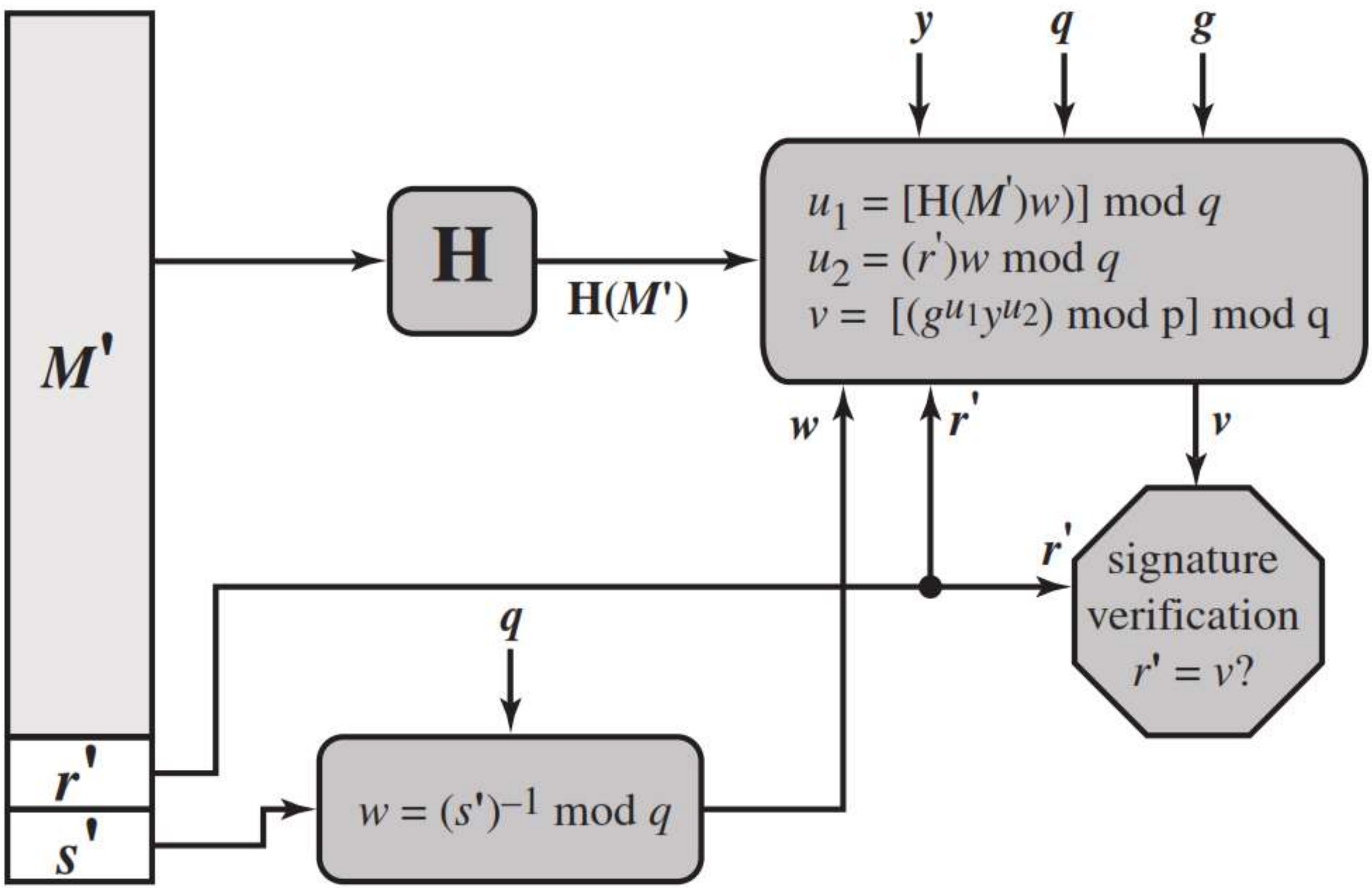


DSA Signing



(a) Signing

DSA Verifying



ElGamal Digital Signatures

- Uses 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
- 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**: $v_A = a^{-sA} \bmod q$

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