An Insight to Cryptosystems & Authentication Protocols

By

Dr. Shashidhar R

Security & Blockchain Researcher,

Samsung R&D Institute India

Security Services

- **Authentication** - assurance that communicating entity is the one claimed have both peer-entity & data origin authentication
- **Access Control** - prevention of the unauthorized use of a resource
- **Data Confidentiality** –protection of data from unauthorized disclosure
- **Data Integrity** - assurance that data received is as sent by an authorized entity
- **Non-Repudiation** - protection against denial by one of the parties in a communication
- **Availability** – resource accessible/usable

Hash Functions

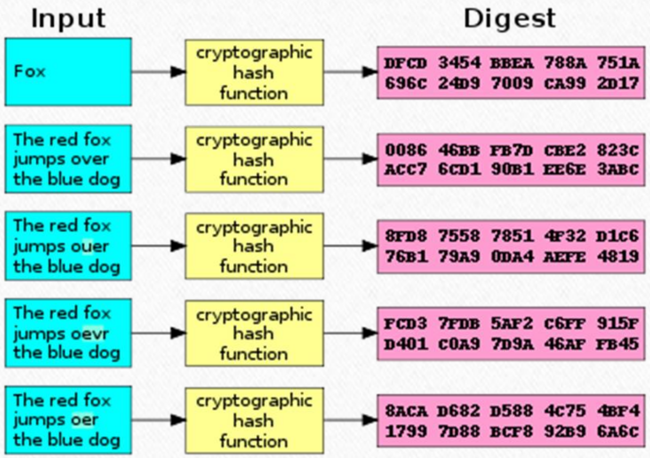
A Hash Function is a mathematical function that maps data of arbitrary size to a fixed-size output.

In Blockchain, Hash Functions are used to provide a unique and tamper-proof digital fingerprint of data.

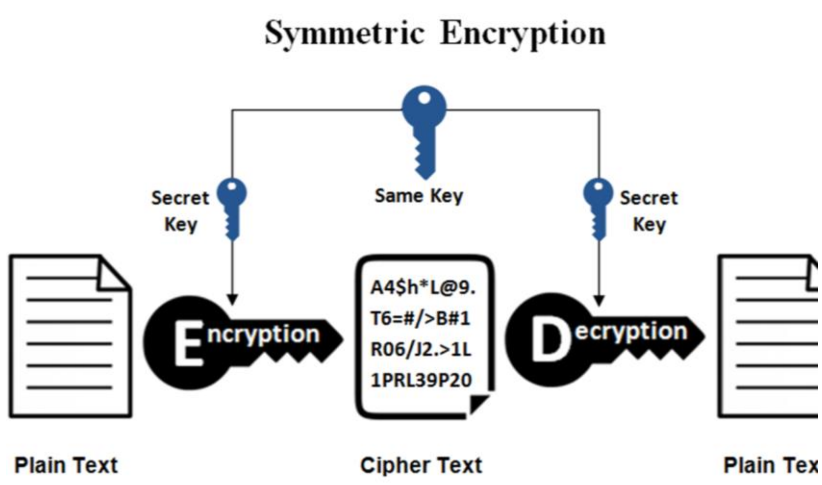
The SHA-256 (Secure Hash Algorithm 256-bit) is commonly used Hash Function in Blockchain.

In Blockchain, Hash Functions are used to create the Hash of each block's data, which includes transactions, timestamp, and a reference to the previous block's Hash.

The Hash of the current block is included in the next block, creating a chain of blocks that are linked together cryptographically.



Symmetric Cryptography



- Uses the same key for both encryption and decryption
- Fast and efficient for bulk data encryption
- Examples: AES, DES, Blowfish
- Symmetric Cryptography to secure communication
- Transactions are encrypted with a shared secret key to maintain confidentiality
- transactions are encrypted using the Advanced Encryption
  with a 256-bit key

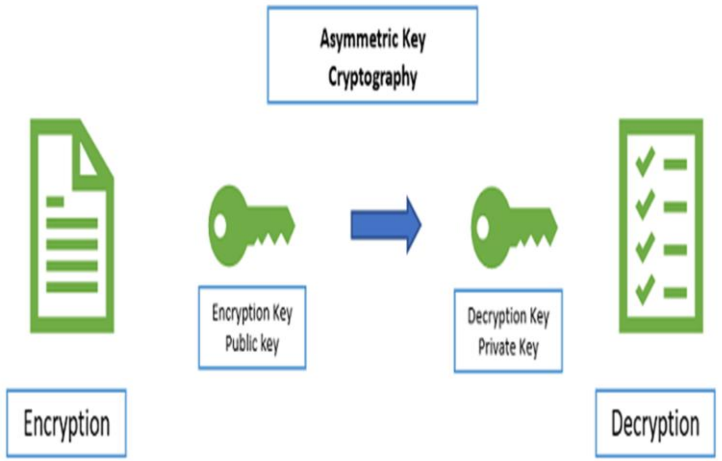
Public Key Cryptography

Public key cryptography is a type of asymmetric cryptography used in Blockchain.

It uses a pair of keys – a public key and a private key – for secure communication.

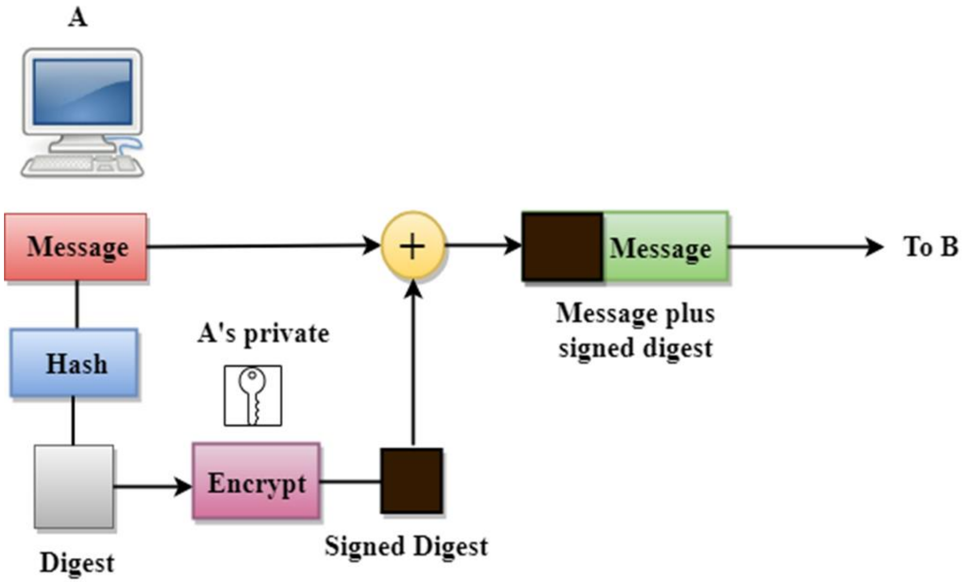
The public key is used for encryption and is available to anyone, while the private key is kept secret and used for decryption.

In Blockchain, public key cryptography is used for digital signatures.

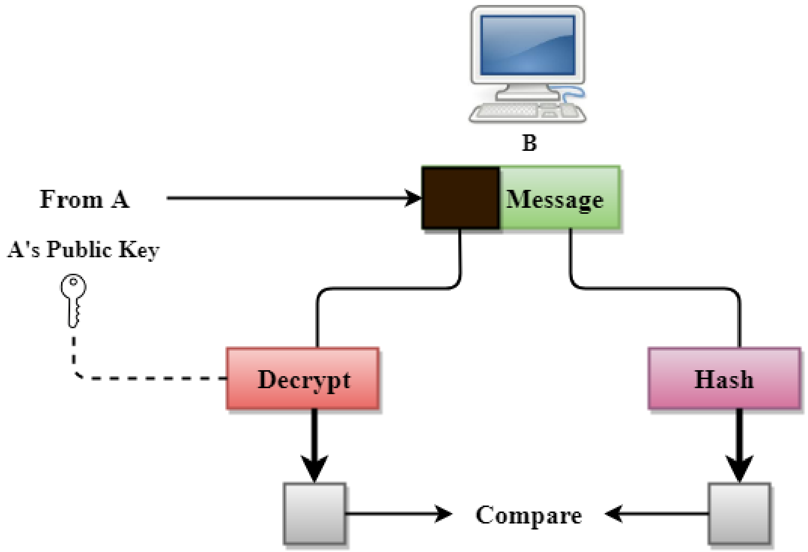


Digital Signatures Creation

- Importance of digital signatures in ensuring security and authenticity in blockchain transactions
- A digital signature is created using a private key and the message to be signed



Digital Signature Verification



- A digital signature is verified using the corresponding public key and the original message
- The advantages of using digital signatures in blockchain, such as improved security, authenticity, and non-repudiation

Message Authentication & Digital SIgnature

Outline

- Message authentication
- MAC, HMAC, DAA, CMAC
- Digital Signature
- Digital Signature Requirements
- RSA approach
- DSA Signing and Verifying
- Elgamal Fdigital SIgnature

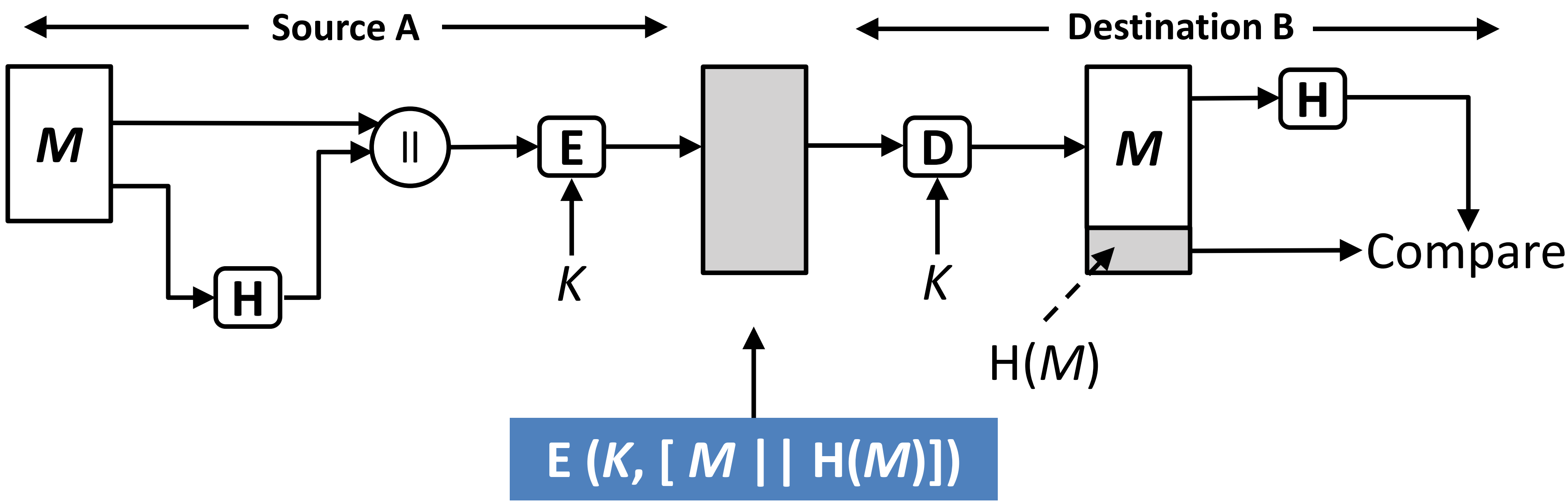
1. Message Authentication

- **Message authentication** is a mechanism or service used to verify the integrity of a message.

- Message authentication assures that data received are exactly as sent (i.e., contain no modification, insertion, deletion, or replay).

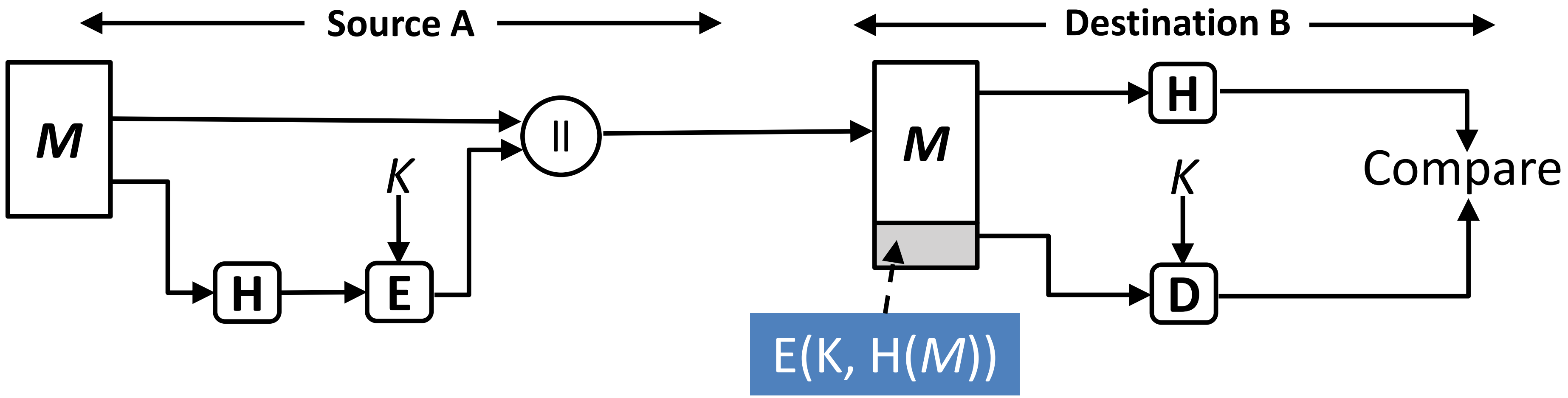
- When a hash function is used to provide message authentication, the **hash function value** is often referred to as a **message digest**.

Message authentication method - 1



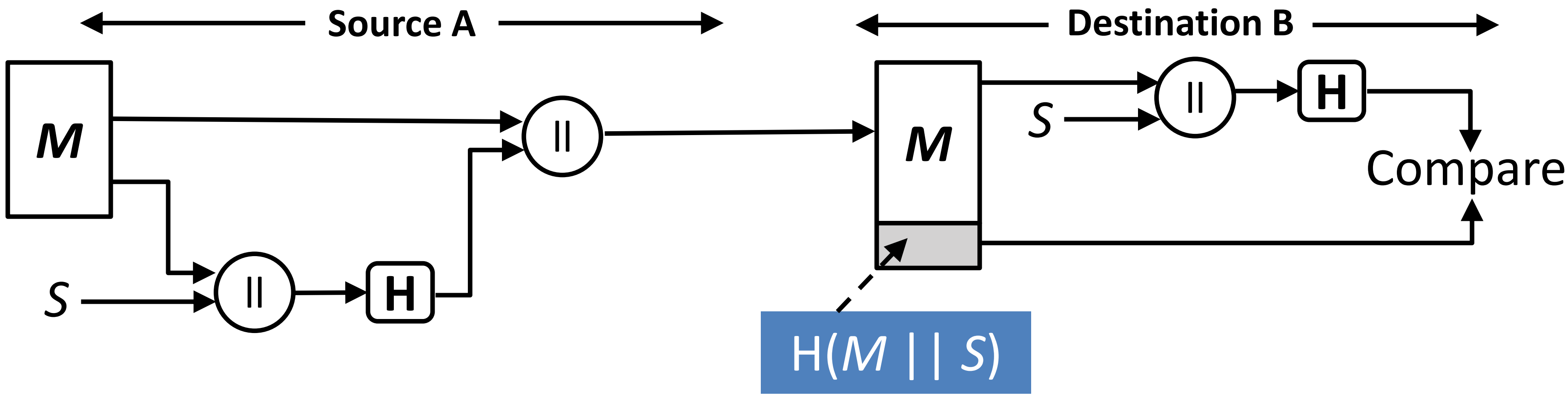
- Only A and B share the secret key, the message must have come from A and has not been altered.
- The hash code provides the structure required to achieve authentication.
- Because encryption is applied to the entire message plus hash code, confidentiality is also provided.

Message authentication method - 2



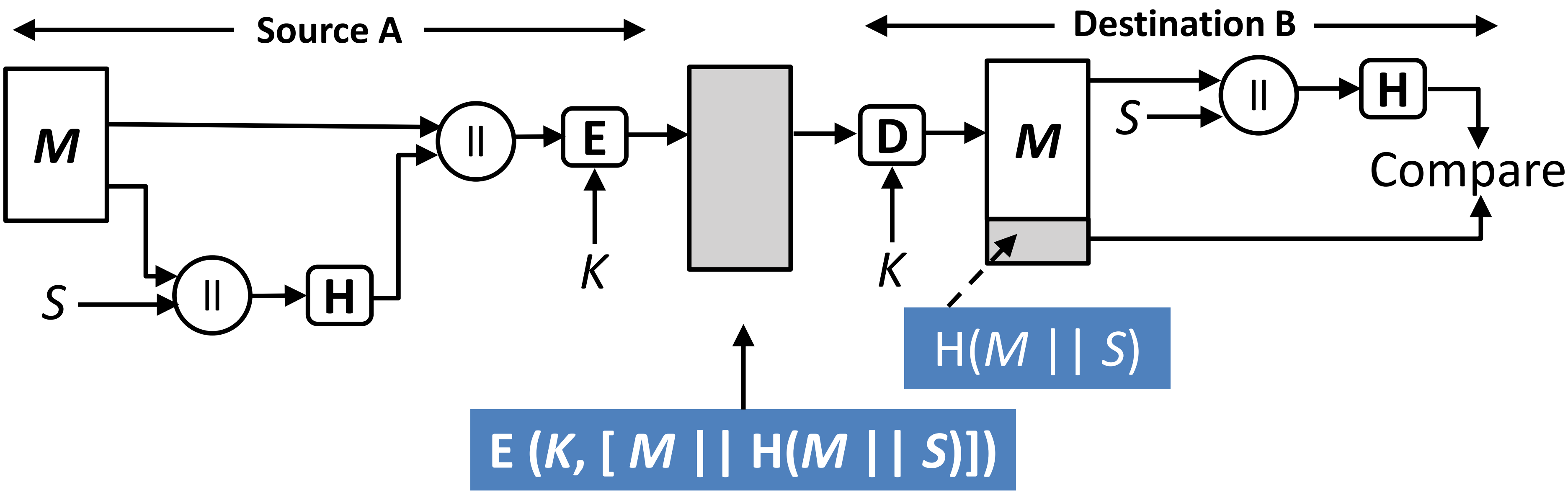
- Only the hash code is encrypted, using symmetric encryption.
- This reduces the processing burden for those applications that do not require confidentiality.

Message authentication method - 3



- It is possible to use a hash function but no encryption for message authentication.
- A and B share a common secret value S.
- A computes the hash value over the concatenation of M and S and appends the resulting hash value to M.
- Because B possesses S, it can recompute the hash value to verify.
- An opponent cannot modify an intercepted message.

Message authentication method - 4



- Confidentiality can be added to the approach of method (3) by encrypting the entire message plus the hash code.

MAC (Message Authentication Code)

- More commonly, message authentication is achieved using a **MAC** also known as **keyed hash function**.
- MACs are used between two parties that share a secret key to authenticate information exchanged between those parties.
- A **MAC** function takes as input a secret key and a data block and produces a hash value, referred to as the **MAC**.
- The combination of hashing and encryption results in an overall function that is, in fact, a MAC (Method -2 in previous slide).

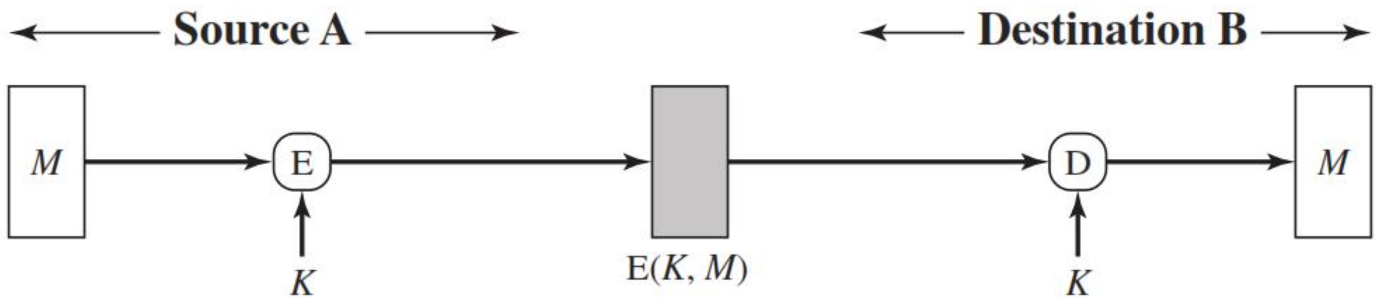
Message Authentication Requirements

1. Disclosure
2. Traffic analysis
3. Masquerade
4. Content modification
5. Sequence modification
6. Timing modification
7. Source repudiation
8. Destination repudiation

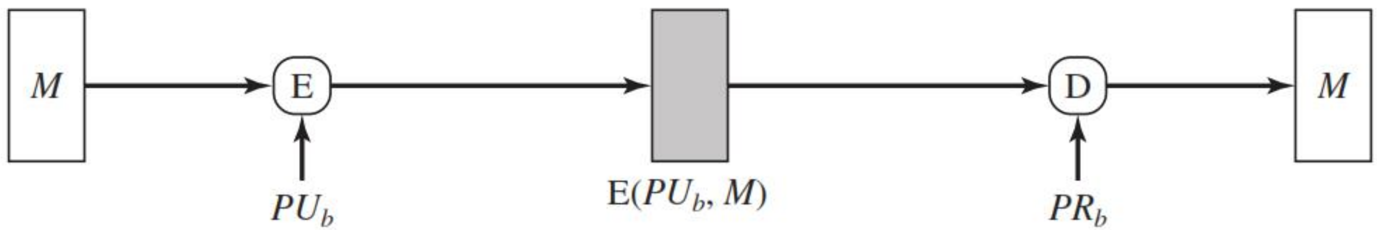
Requires Message Confidentiality

Requires Message Authentication

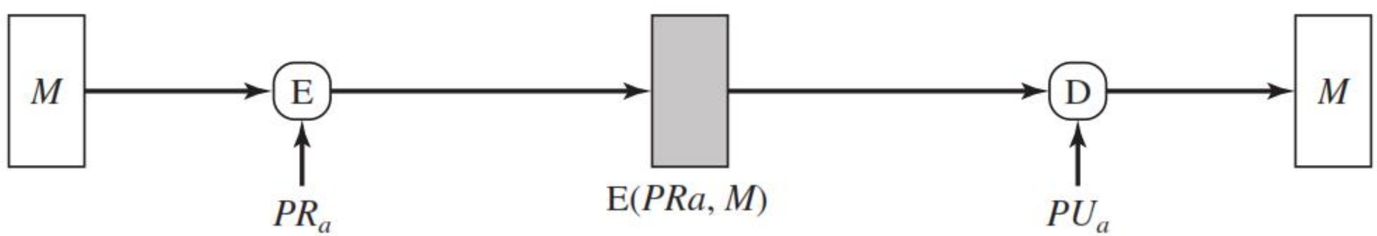
Requires Digital Signature



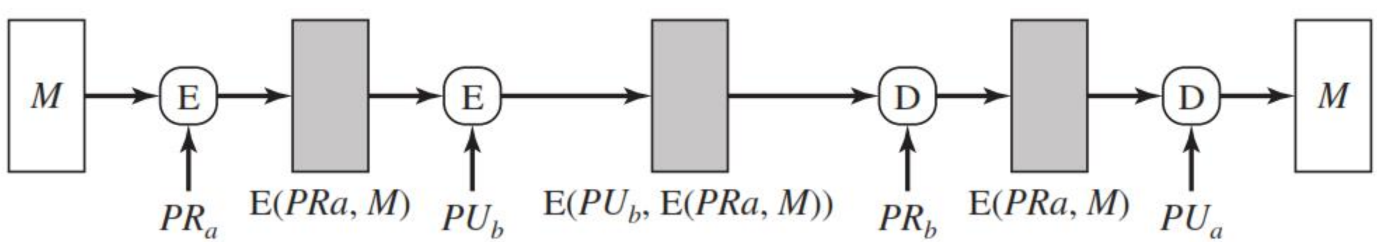
(a) Symmetric encryption: confidentiality and authentication



(b) Public-key encryption: confidentiality



(c) Public-key encryption: authentication and signature



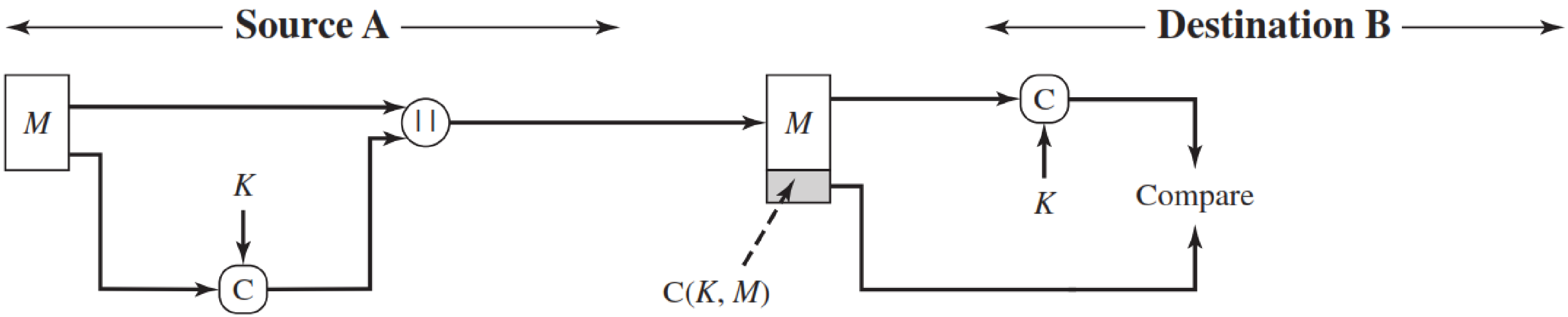
(d) Public-key encryption: confidentiality, authentication, and signature

Message Authentication Code

- An alternative authentication technique involves the use of a secret key to generate a small fixed-size block of data, known as a **cryptographic checksum** or **MAC**
- MAC is appended to the message. This technique assumes that two communicating parties, say A and B, share a common secret key K.
- When A has a message to send to B, it calculates the MAC as a function of the message and the key

$$\mathbf{MAC = C\,(\,K\,,\,M\,)}$$

Message Authentication Code



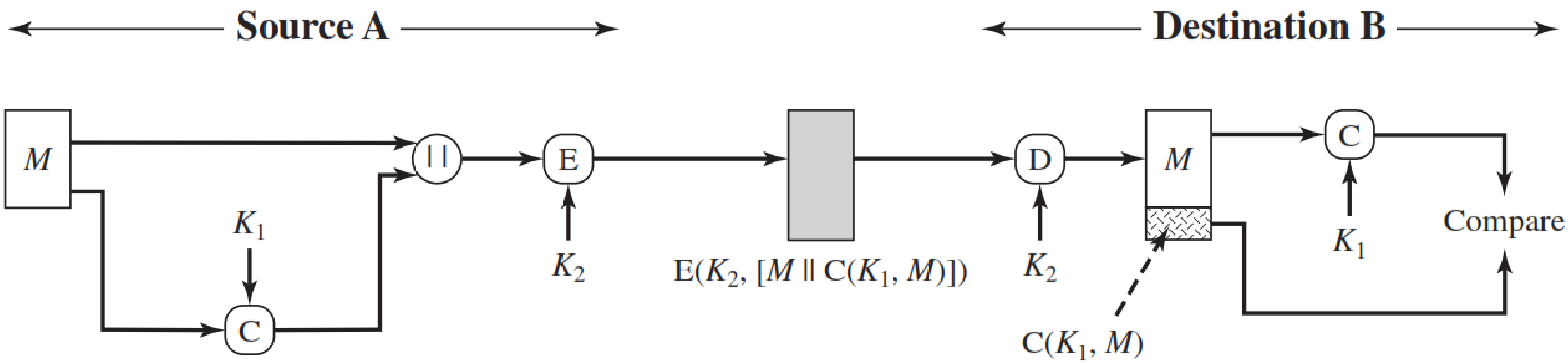
(a) Message authentication

- The receiver is assured that the message has not been altered. If an attacker alters the message but does not alter the MAC, then the receiver's calculation of the MAC will differ from the received MAC.
- Because the attacker is assumed not to Know the secret key, the attacker cannot alter the MAC to correspond to the alterations in the message.

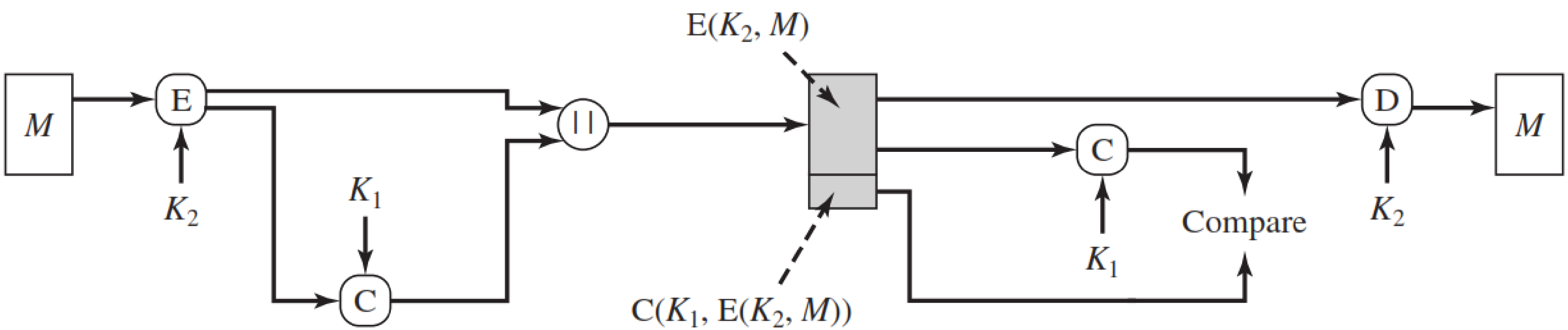
Message Authentication code - Cont…

- The receiver is assured that the message is from the alleged sender.
- Because no one else knows the secret key, no one else could prepare a message with a proper MAC.
- A MAC function is similar to encryption. One difference is that the MAC algorithm need not be reversible, as it must be for decryption.
- In general, the MAC function is a many-to-one function. The domain of the function consists of messages of some arbitrary length, whereas the range consists of all possible MACs and all possible keys.
- If an n-bit MAC is used, then there are 2^n possible MACs

Message Authentication code - Cont…



(b) Message authentication and confidentiality; authentication tied to plaintext



(c) Message authentication and confidentiality; authentication tied to ciphertext

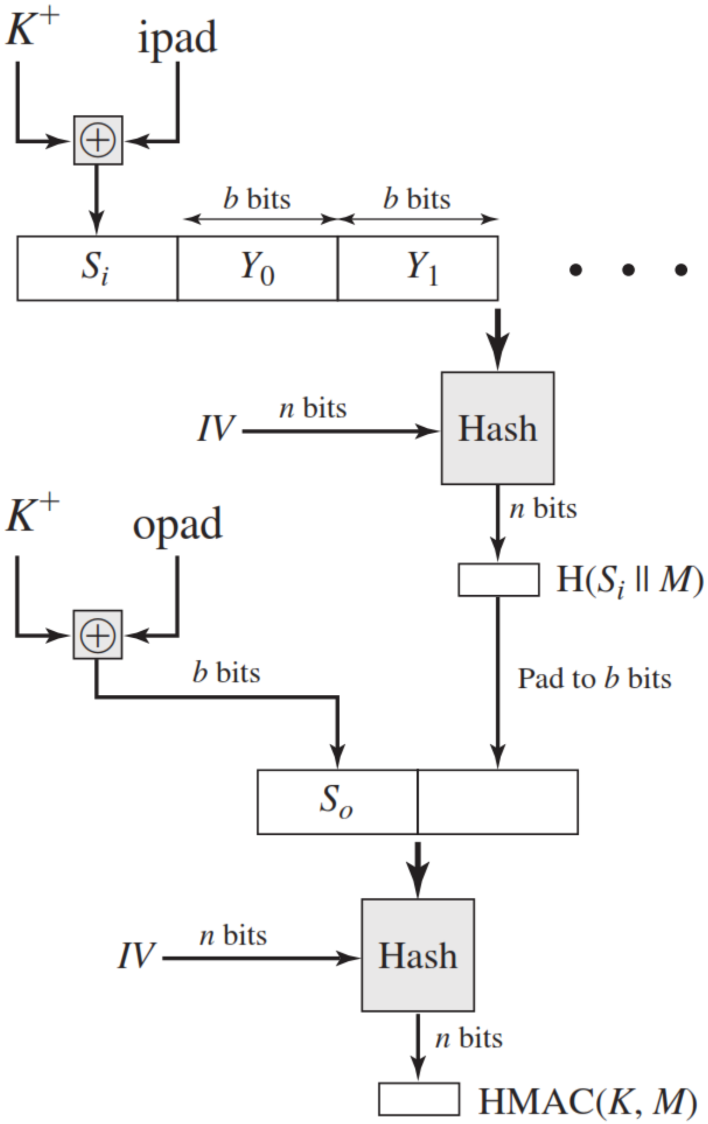
MAC Based on Hash Functions - HMAC

- Cryptographic hash functions such as MD5 and SHA generally execute faster in software than symmetric block ciphers such as DES.
- Library code for cryptographic hash functions is widely available.

Design objectives for HMAC

- To use, without modifications, available hash functions.
- To allow for easy replaceability of the embedded hash function in case faster or more secure hash functions are found or required.
- To preserve the original performance of the hash function without incurring a significant degradation.
- To use and handle keys in a simple way.
- To have a well understood cryptographic analysis of the strength of the authentication mechanism based on reasonable assumptions about the embedded hash function.

HMAC Structure



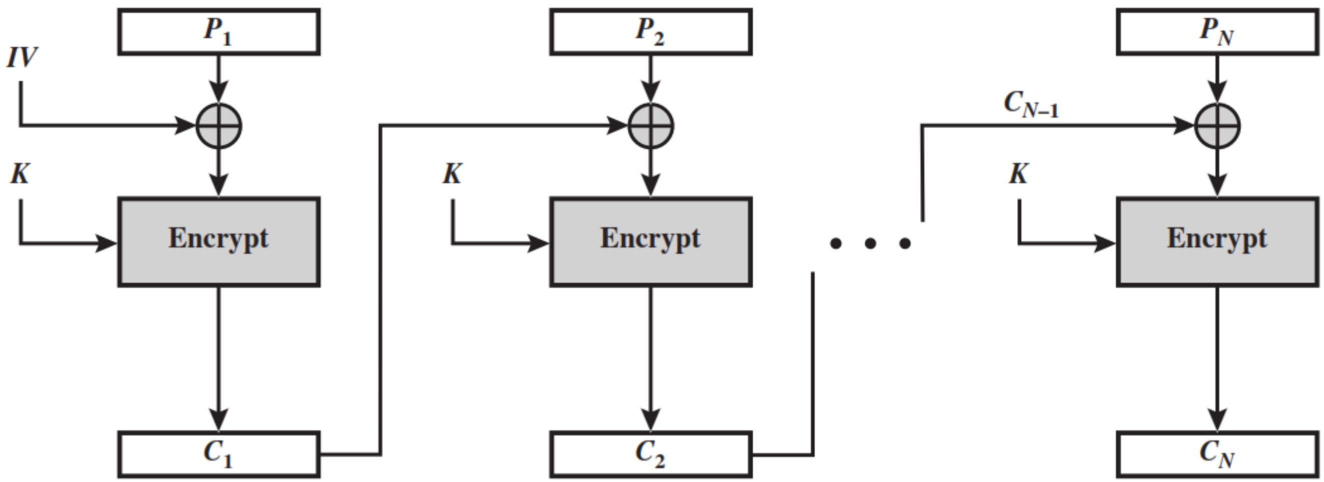
1. Append zeros to the left end of K to create a b-bit string K^+
2. XOR K^+ with ipad to produce the b-bit block Si.
3. Append M to Si.
4. Apply H to the stream generated in step 3.
5. XOR K^+ with opad to produce the b-bit block S_0.
6. Append the hash result from step 4 to S_0.
7. Apply H to the stream generated in step 6 and output the result.

HMAC Structure

- H = embedded hash function (e.g., MD5, SHA-1, RIPEMD-160)
- IV = initial value input to hash function
- M = message input to HMAC
- Y_i = i^{th} block of M
- L = number of blocks in M
- n = length of hash code produced by embedded hash function
- K^+ = K padded with zeros on the left so that the result is b bits in length
- ipad = 00110110 (36 in hexadecimal) repeated *b/8 times*
- opad = 01011100 (5C in hexadecimal) repeated *b/8 times*

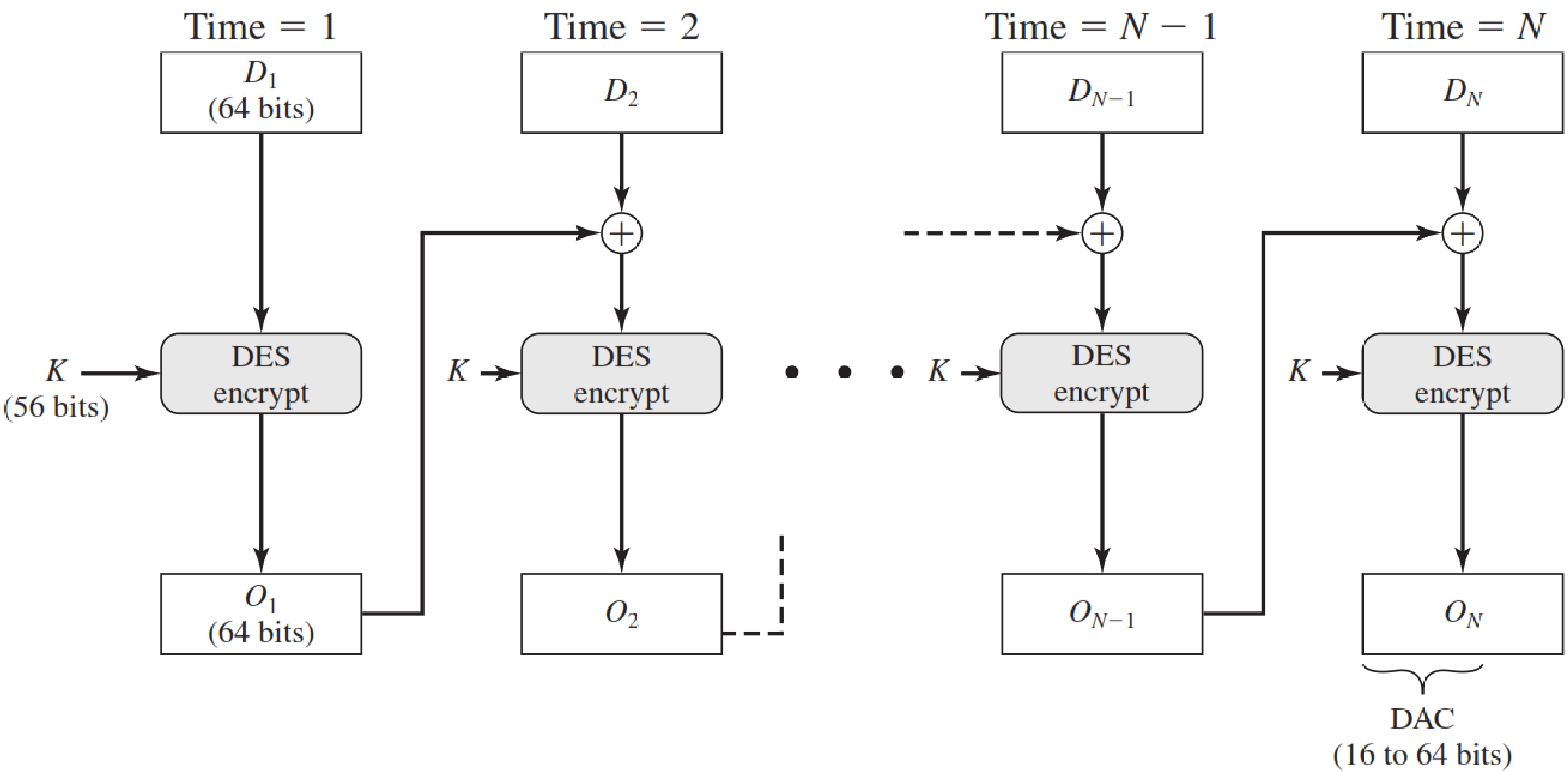
MAC based on Block Ciphers

- The **Data Authentication Algorithm** (DAA), based on DES, has been one of the most widely used MACs for a number of years.
- The algorithm can be defined as using the cipher block chaining (CBC) mode of operation of DES (Figure 6.4) with an initialization vector of zero.



(a) Encryption

Data Authentication Algorithm (DAA)



Data Authentication Algorithm (DAA)

- The data (e.g., message, record, file, or program) to be authenticated are grouped into contiguous 64-bit blocks:
- D1, D2, …. Dn. If necessary, the final block is padded on the right with zeroes to form a full 64-bit block. Using the DES encryption algorithm E and a secret key K, **a data authentication code (DAC)** is calculated as follows

$$
\begin{aligned}
O_1 &= \mathrm{E}(K, D) \\
O_2 &= \mathrm{E}(K, [D_2 \oplus O_1]) \\
O_3 &= \mathrm{E}(K, [D_3 \oplus O_2]) \\
&\cdot \\
&\cdot \\
&\cdot \\
O_N &= \mathrm{E}(K, [D_N \oplus O_{N-1}])
\end{aligned}
$$

Cipher-Based Message Authentication Code (CMAC)

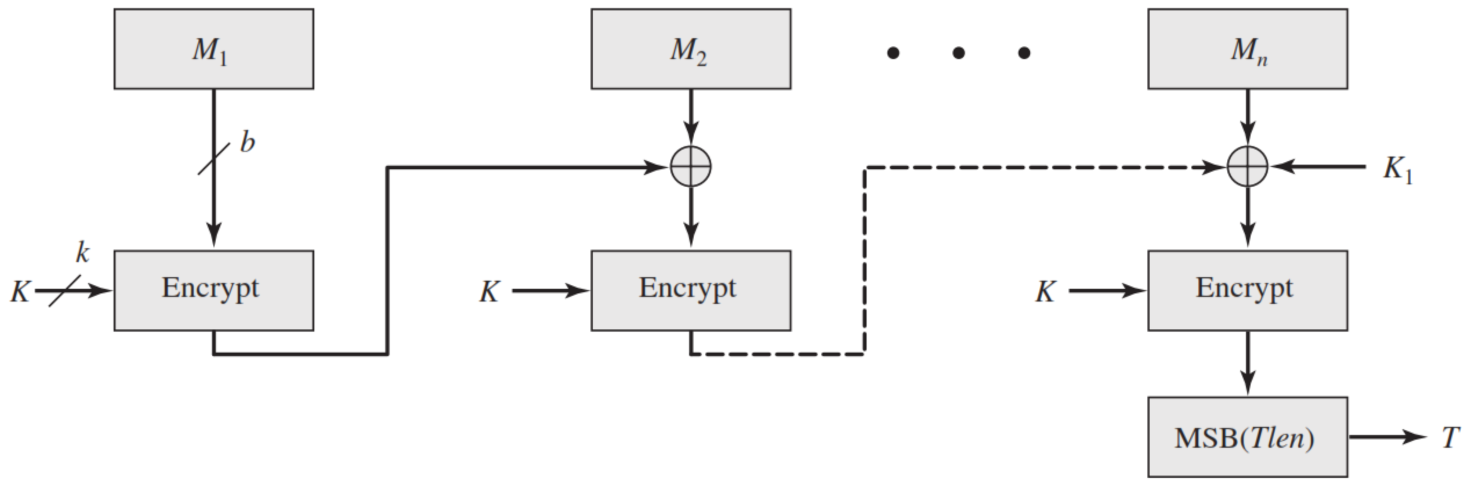
- **Cipher-based Message Authentication Code (CMAC)** mode of operation for use with AES and triple DES.
- First, let us define the operation of CMAC when the message is an integer multiple n of the cipher block length b. For AES, b = 128, and for triple DES, b = 64. The message is divided into n blocks (M1, M2,… Mn)

Cipher-Based Message Authentication Code (CMAC)

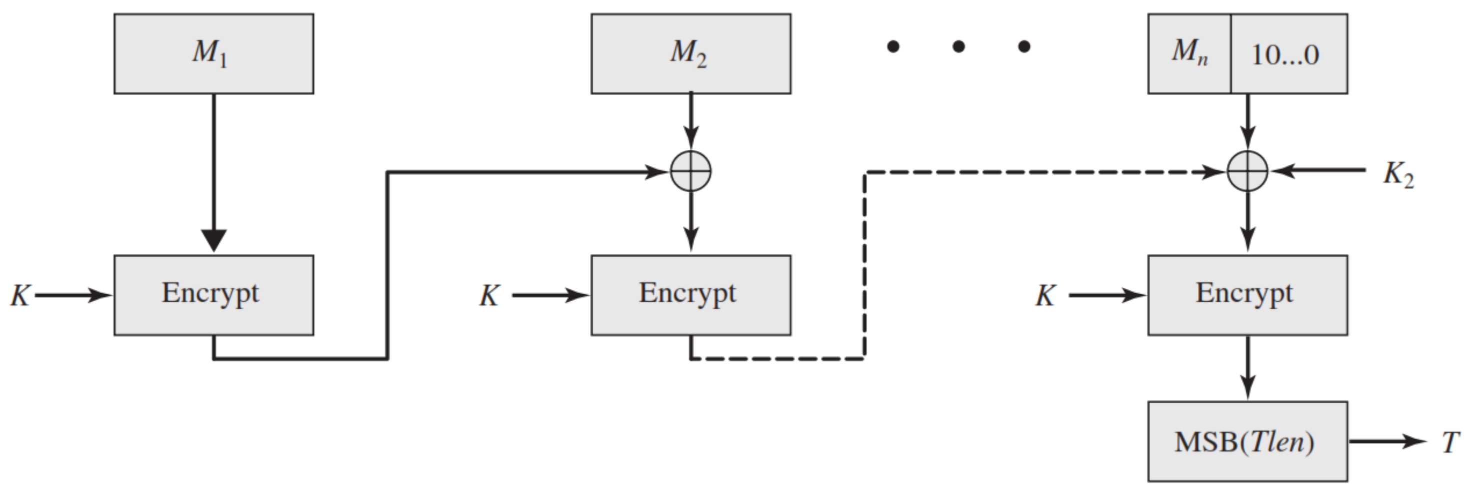
- The algorithm makes use of a k-bit encryption key K and a b-bit constant, K1.
- For AES, the key size k is 128, 192, or 256 bits; for triple DES, the key size is 112 or 168 bits.
- CMAC is calculated as follows

$$
\begin{aligned}
C_1 &= \mathrm{E}(K, M_1) \\
C_2 &= \mathrm{E}(K, [M_2 \oplus C_1]) \\
C_3 &= \mathrm{E}(K, [M_3 \oplus C_2]) \\
&\cdot \\
&\cdot \\
&\cdot \\
C_n &= \mathrm{E}(K, [M_n \oplus C_{n-1} \oplus K_1]) \\
T &= \mathrm{MSB}_{Tlen}(C_n)
\end{aligned}
$$

Cipher-Based Message Authentication Code (CMAC)



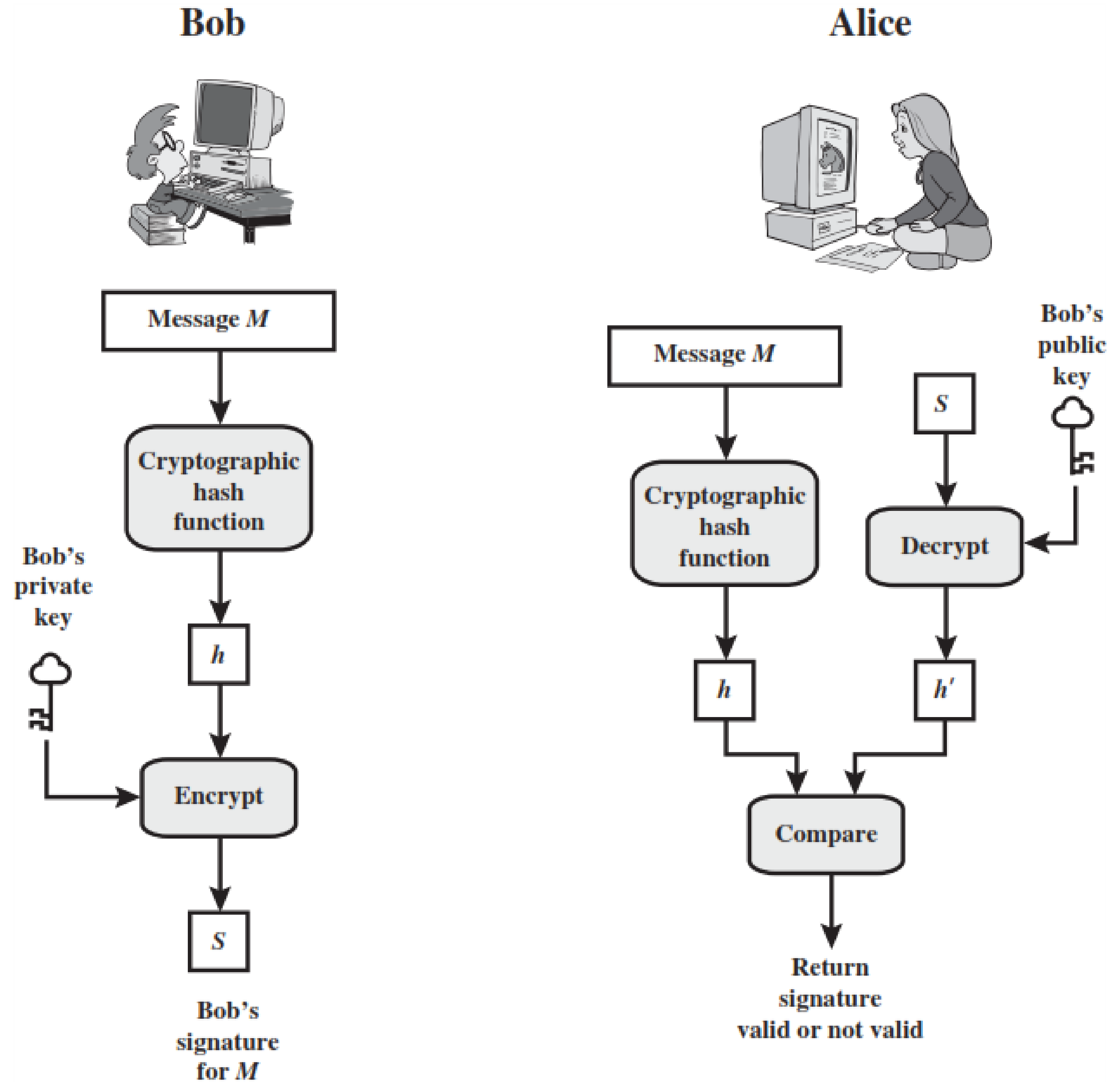
(a) Message length is integer multiple of block size



(b) Message length is not integer multiple of block size

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

Digital Signature Requirements

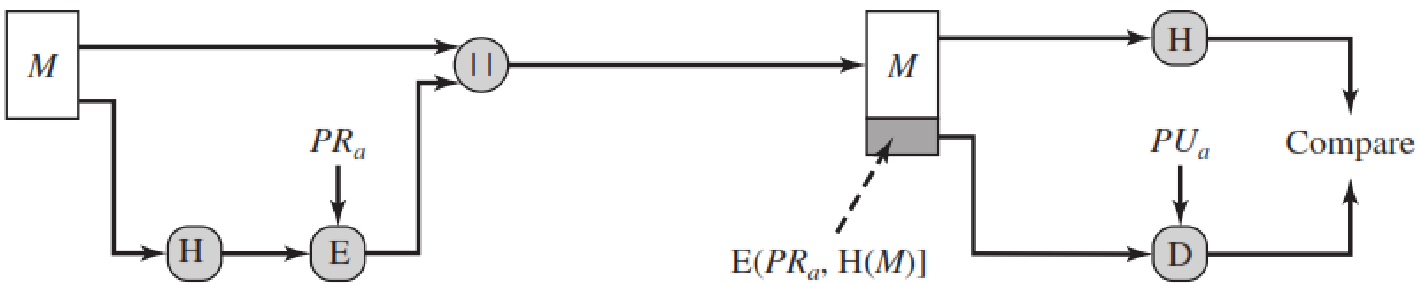
1. The signature must be a **bit pattern** that depends on the message being signed.
2. The signature must use some information **unique** to the sender to prevent both forgery and denial.
3. It must be relatively **easy to produce** the digital signature.
4. It must be relatively **easy to recognize** 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**.



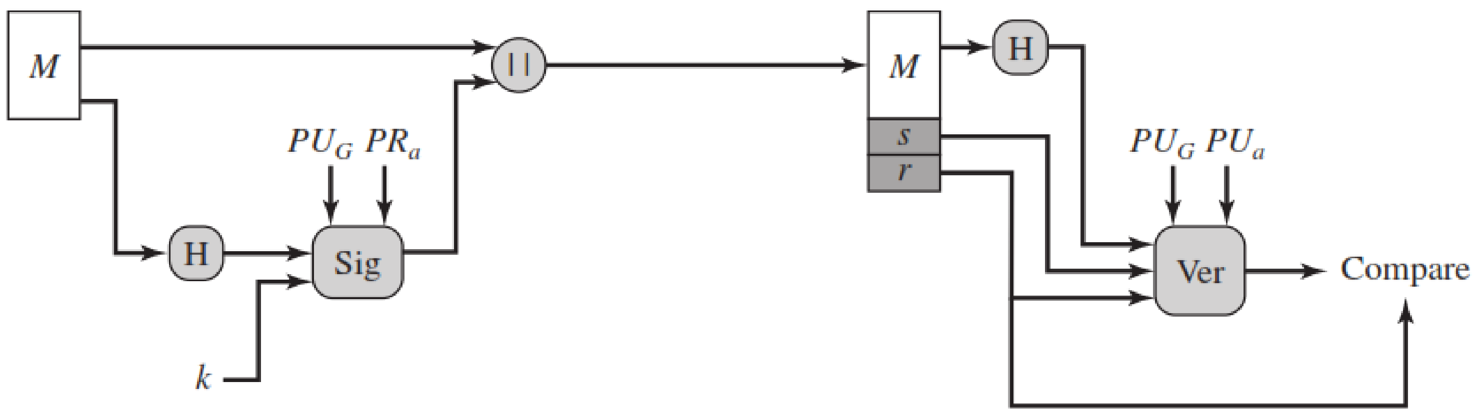
(a) RSA approach

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 $0 < k < q$

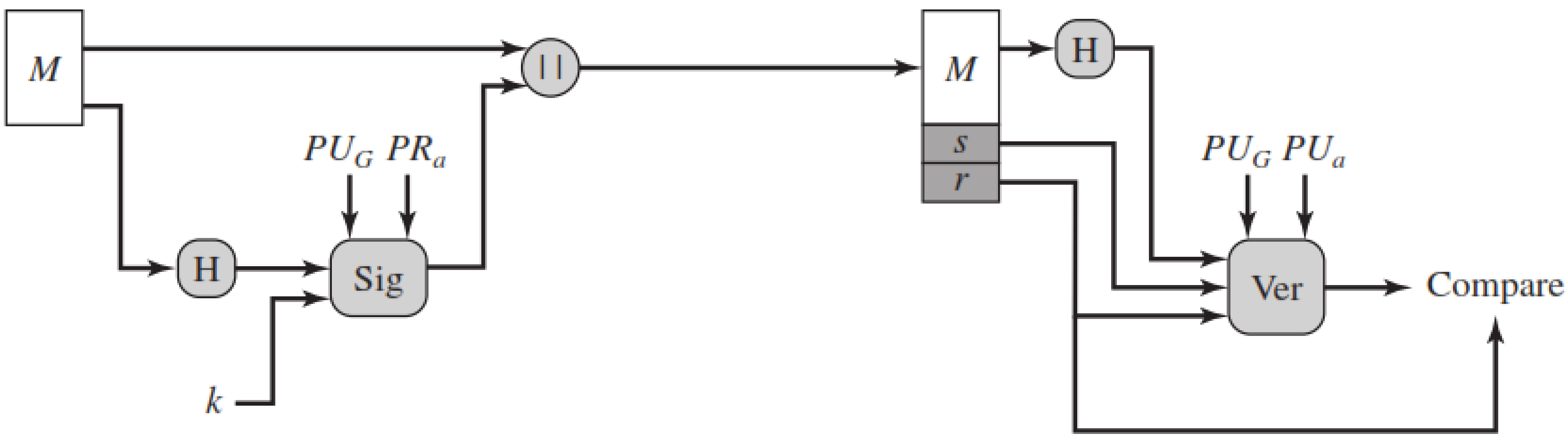
Digital Signature Algorithm

Signing

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

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

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



Digital Signature Algorithm

Verifying

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

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

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

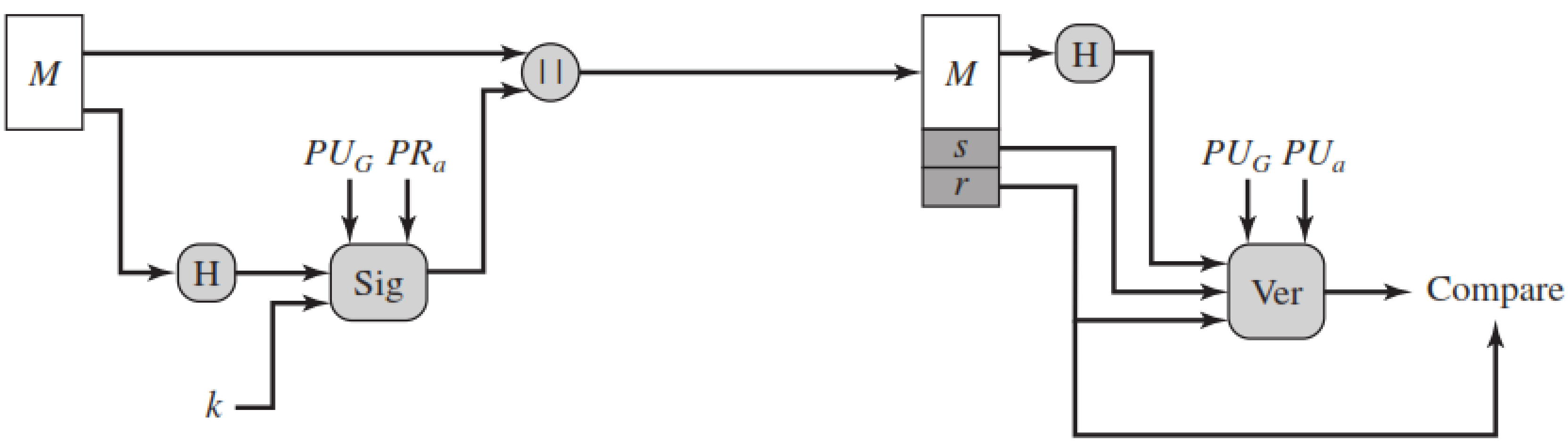
$$v = [(g^{u1}\, y^{u2}) \bmod p] \bmod q$$

$$\text{TEST: } v = r'$$

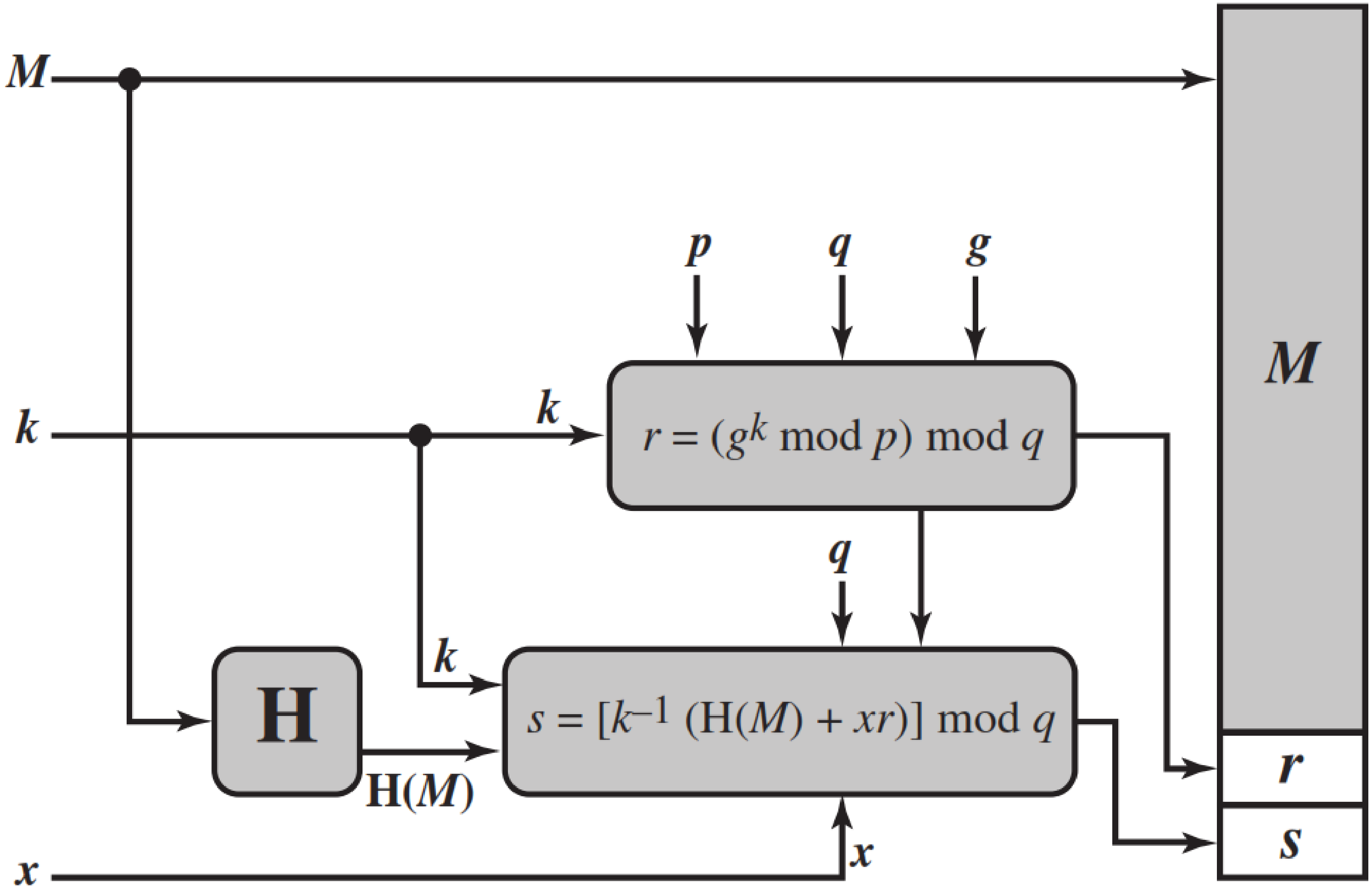
M = message to be signed

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

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

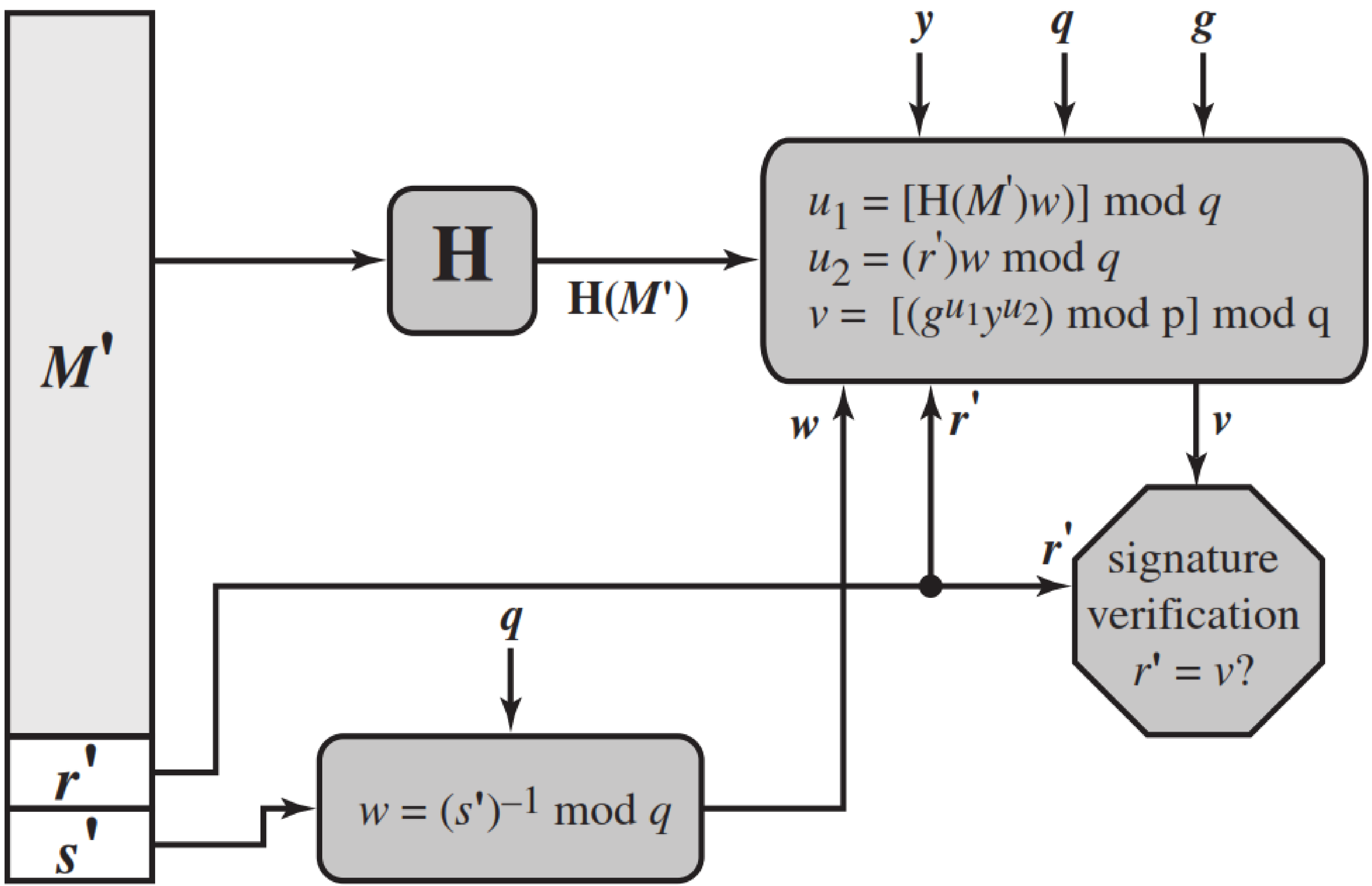


DSA Signing



(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