Computer Network Security

Dr. Nadine ZBIB
Assistant Professor
College of Science and Information Systems

Public-key cryptography and message authentication

1. Approaches to Message Authentication
2. Secure Hash Functions and HMAC
3. Public Key Cryptography Principles
4. Public Key Cryptography Algorithms
5. Digital Signatures
6. Key Management

Authentication

"A party should be able to verify the identity and some other information of other party."

- Requirements - must be able to verify that:
    1. Message came from a valid source,
    2. Contents have not been altered,
    3. Sometimes, it was sent at a certain time or sequence.

- Protection against active attack (falsification of data and transactions)

Approaches to Message Authentication

- Authentication Using Conventional Encryption
  - Only the sender and receiver should share a key
- Message Authentication without Message Encryption
  - An authentication tag is generated and appended to each message
- Message Authentication Code
  - Calculate the MAC as a function of the message and the key. MAC = F(K, M)

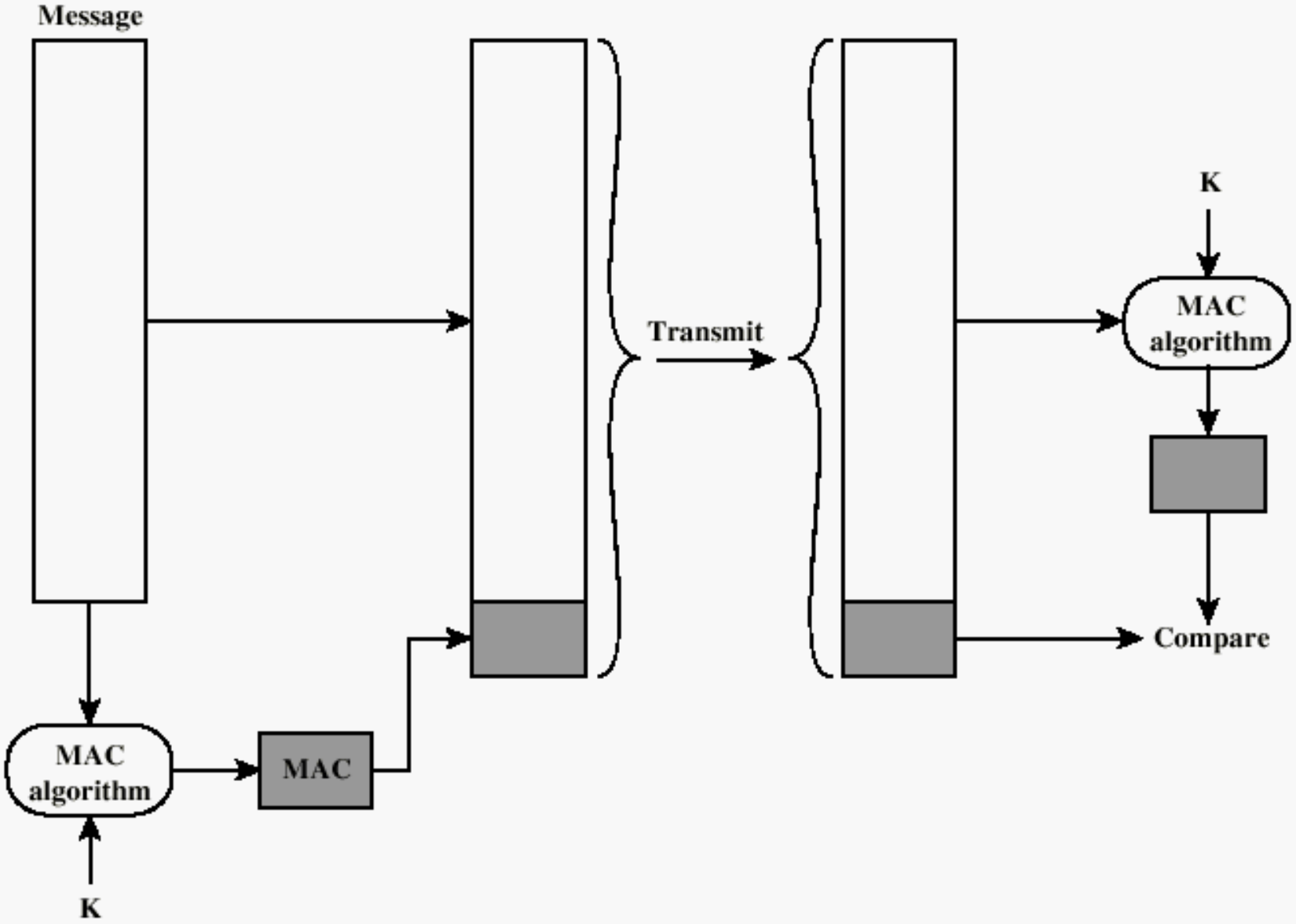


Figure 3.1 Message Authentication Using a Message Authentication Code (MAC)

- One authentication technique involves the use of a secret key to generate a small block of data, know as **MAC,** that's is appended to the message.
- this technique assumes that sender and receiver share a common secret key K_{AB}
- when A has a message to send to B, it calculates the MAC as a function of the message and the key: $MAC_M = F(K_{AB}, M)$
- **The message + MAC** are transmitted to the receiver, the recipient performs the same calculation on the received message, using the same key, to generate a new MAC

The received code is compared to the calculated code (Fig 3.1), if we assume that only the receiver and sender know the identity of the secret key, and if the received code matches the calculated code, then:

1. The receiver is assured that the message has not been altered.
2. The receiver is assured that the messages is from the alleged sender.
3. If the message includes a sequence number, then the receiver can be assured of the proper sequence, because an attacker cannot successfully alter the sequence number.

Hash Functions

- condenses arbitrary message to fixed size

  `h = H(M)`

  - No secret key needed
  - usually assume hash function is public
  - hash used to detect changes to message
- want a cryptographic hash function
  - computationally infeasible to find data mapping to specific hash (one-way property)
  - computationally infeasible to find two data to same hash (collision-free property)

One-way HASH function

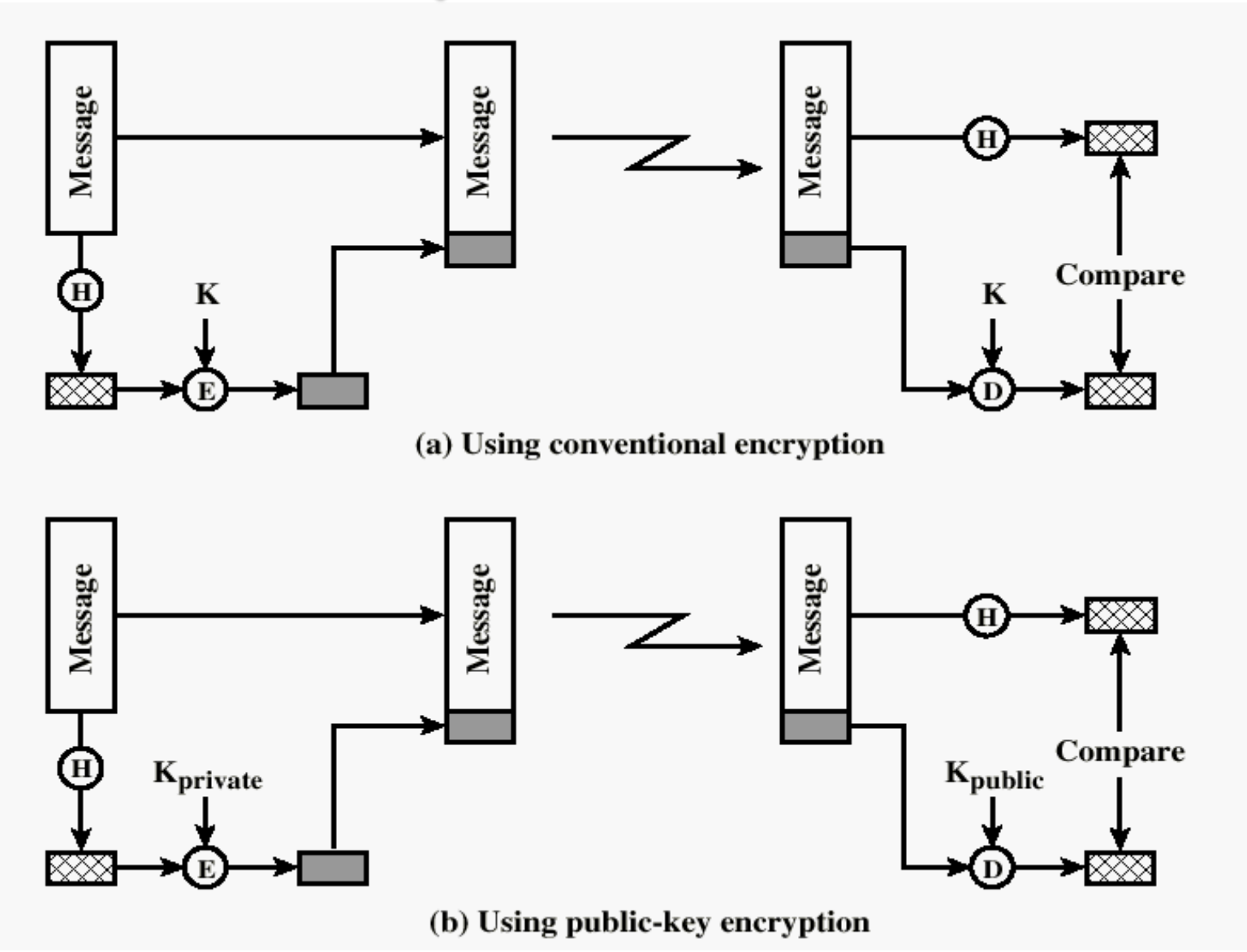
- An alternative to the MAC is the one-way hash function.

- as with the MAC, a hash function accepts a variable size message M as input and produce a fixed size message digest H(M) as output.

- Unlike the MAC, a hash function does not also take a secret key as input.

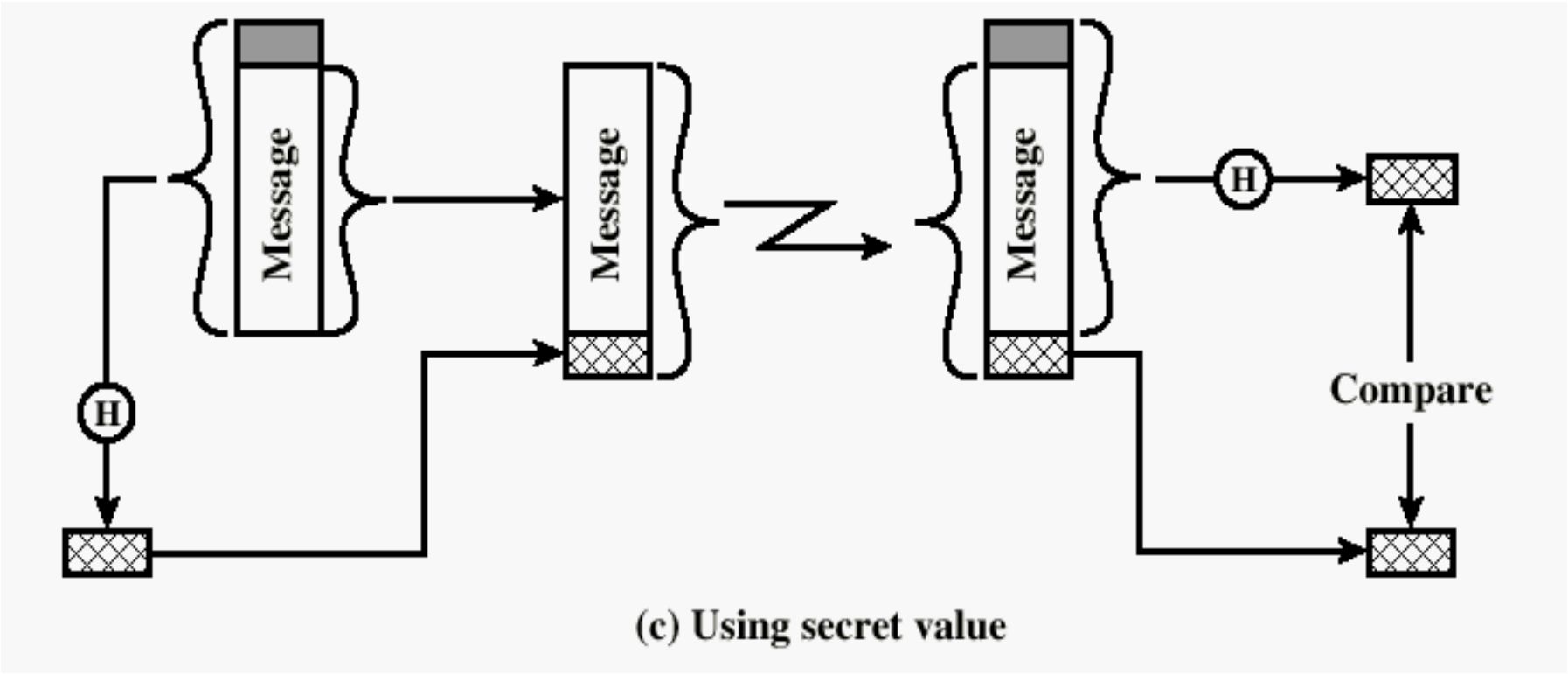
- To authenticate a message, the message digest is sent with the message in such a way that the message digest is authentic.

One-way HASH function



One-way HASH function

- Secret value is added before the hash and removed before transmission.



(c) Using secret value

HASH Functions

- Purpose of the HASH function is to produce a "fingerprint of a file/message.

- Properties of a HASH function H:
  1. H can be applied to a block of data at any size
  2. H produces a fixed length output
  3. Easy to implement: H(x) is easy to compute for any given x.
  4. One-way property: For any given block h, it is computationally infeasible to find x such that H(x) = h.

HASH Functions...

- Properties of a HASH function H:...

4. Weak Collision Resistance:

   For any given block x, it is computationally infeasible to find y such that H(y) = H(x).

5. Strong Collision Resistance:

   It is computationally infeasible to find any pair (x, y) such that H(x) = H(y)

$$y \neq x$$

Simple Hash Function

- The input (message, file...) is viewed as a sequence of n-bit blocks,
- The input is processed one block at a time in an iterative fashion to produce an n-bit hash function.

- One of the simplest hash functions is the bit-by-bit exclusive XOR of every block.

Simple Hash Function

	bit 1	bit 2	• • •	bit *n*
block 1	b_{11}	b_{21}		b_{n1}
block 2	b_{12}	b_{22}		b_{n2}
	•	•	•	•
	•	•	•	•
	•	•	•	•
block *m*	b_{1m}	b_{2m}		b_{nm}
hash code	C_1	C_2		C_n

Figure 3.3 Simple Hash Function Using Bitwise XOR

- One-bit circular shift on the hash value after each block is processed would improve

Simple Hash Function

- Ci = ith bit of the hash function, 1<=i<=n
- M = number of n-bot blocks in the input
- Bij = ith bit in the ij block
- ⊕ = XOR operation

$$Ci = bi1 + bi2 + \ldots + bim$$

Secure Hash Algorithm (SHA-1)

- Developed by NIST in 1993.

- It accepts as input messages with maximum length of less than 2^64.
- Produces an output of 160 bits.
- Input is processed in 512 bit blocks.

Secure Hash Algorithm (SHA-1)

- **Steps in Processing**
- **Step 1: Append padding bits:**

-Message is padded with bits so that it is a multiple of 512.

- The padding consists of a single 1 bit followed by necessary number of 0s.

-Padding is done even if the message if of desired length.

Secure Hash Algorithm (SHA-1)

- **Steps in Processing…**
- **Step 2: Append the length:**

-A 64 bit block is appended to the message.

-Treated as unsigned 64 bit integer.

-Denotes the length of the message (before padding).

Secure Hash Algorithm (SHA-1)

- **Steps in Processing…**
- **Step 3: Initialize the buffer:**

-A 160 bit buffer is used to hold intermediate and final results.

-Used to chain the output of a stage to the input of next stage.

-This buffer is initialize with a pre-determined value.

Secure Hash Algorithm (SHA-1)

- **Steps in Processing…**
- **Step 4: Process 512 bit blocks:**

-Each 512 bit block goes through 4 rounds of processing.

-Each round has 20 steps.

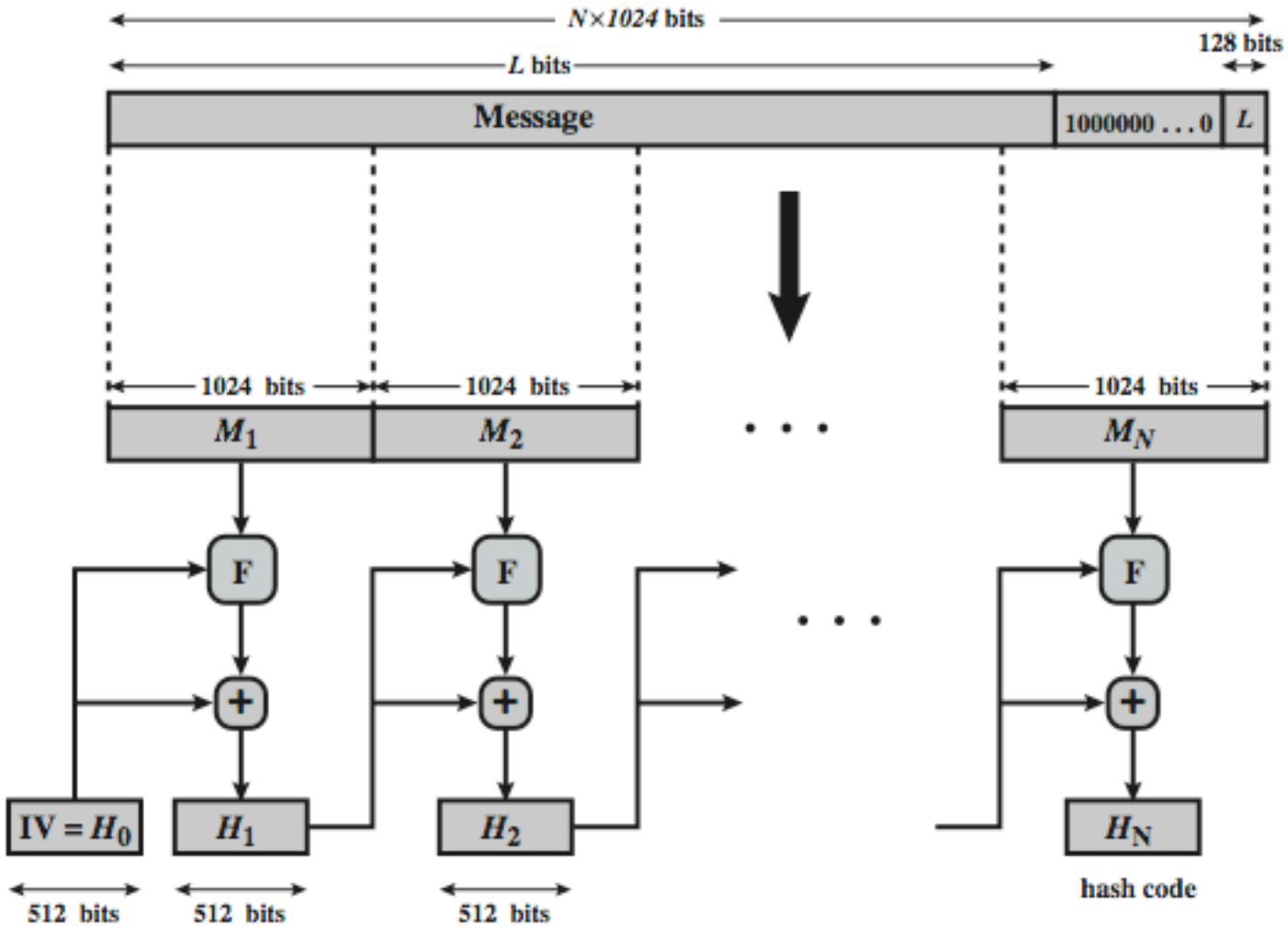
. **Step 5: Output:**

-After all blocks are processed, the 160 bit output of the last block.

SHA-512 Compression Function

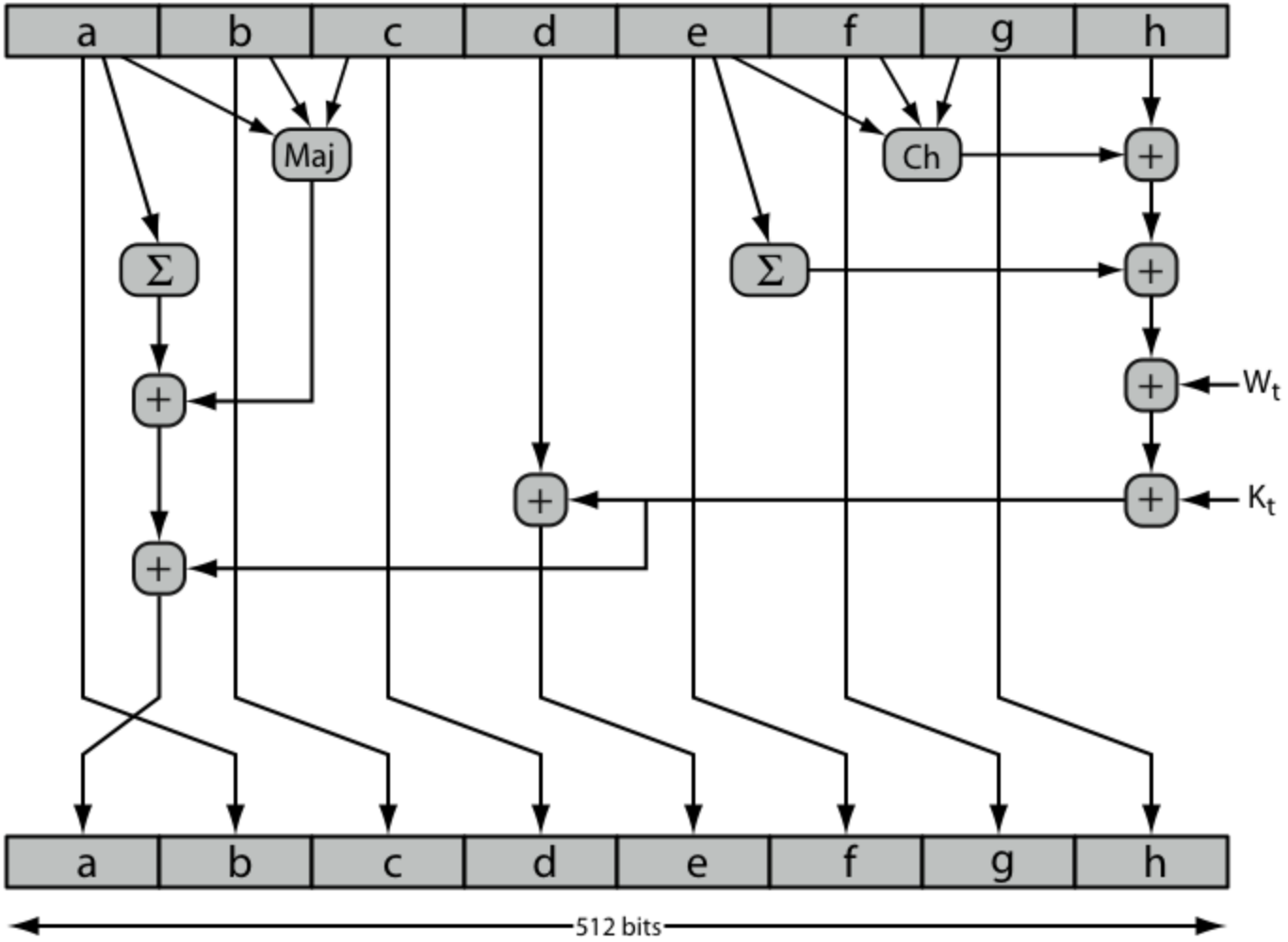
- heart of the algorithm
- processing message in 1024-bit blocks
- consists of 80 rounds
  - updating a 512-bit buffer
  - using a 64-bit value Wt derived from the current message block
  - and a round constant based on cube root of first 80 prime numbers

SHA-512 Overview



F is a compression function that varies;

SHA-512 Round Function



Other Secure HASH functions

	SHA-1	MD5	RIPEMD-160
Digest length	160 bits	128 bits	160 bits
Basic unit of processing	512 bits	512 bits	512 bits
Number of steps	80 (4 rounds of 20)	64 (4 rounds of 16)	160 (5 paired rounds of 16)
Maximum message size	$2^{64}-1$ bits	∞	∞

HMAC

- Use a MAC derived from a cryptographic hash code, such as SHA-1.
- **Motivations:**
  - Cryptographic hash functions executes faster in software than encryption algorithms such as DES
  - Library code for cryptographic hash functions is widely available

HMAC..

- **Design principles:**

-Built around an existing hash function (like MD5, SHA-1, etc.)

  o An existing Hash function can be used.

  o Hash function can be replaced easily.

-Uses a secret key (keyed hash).

HMAC..

- **Operation:**

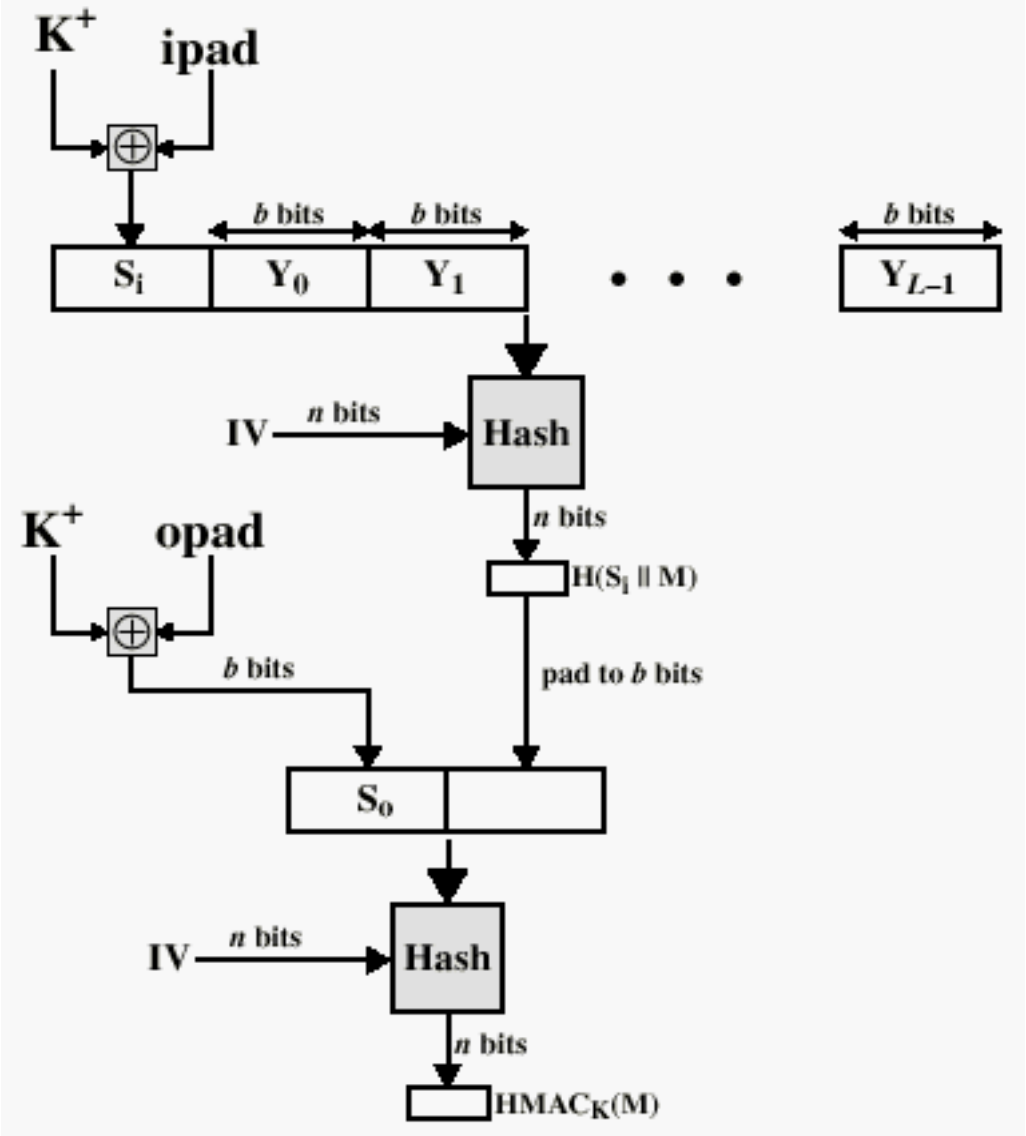
1. Append zeros to the left of key K to create a b-bit K+.
2. Bitwise XOR K+ with ipad to produce a b-bit block.
3. Append the message M to this block.
4. Hash the data generated in Step 3.

HMAC..

- **Operation:…**

5. Bitwise XOR K+ with opad to produce a b-bit block.
6. Append the hash result from step 4 to this block.
7. Hash the data generated in Step 6 to produce the final hash output.

HMAC Structure



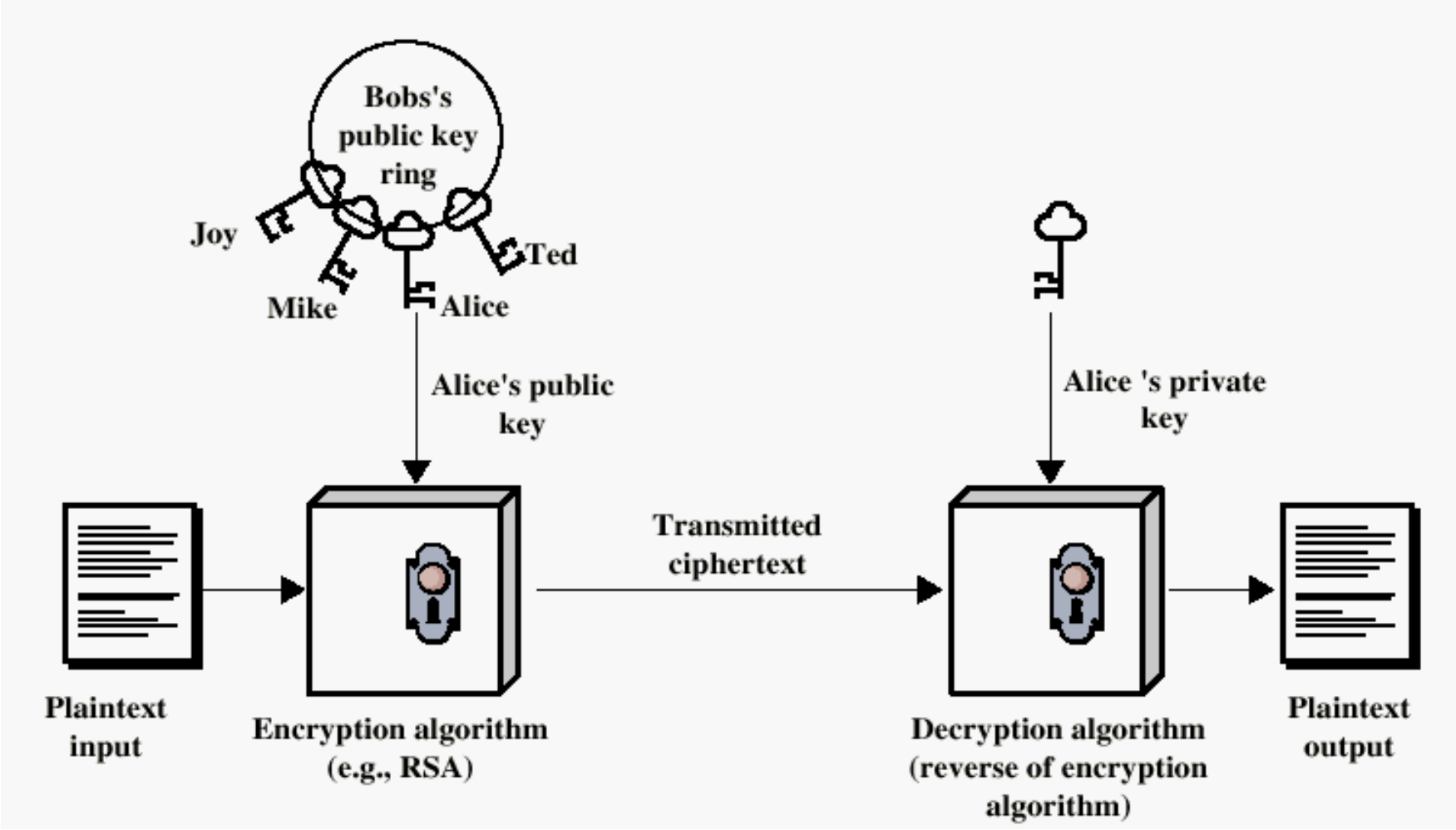
HMAC Security

- proved security of HMAC relates to that of the underlying hash algorithm
- attacking HMAC requires either:
  - brute force attack on key used
  - birthday attack (but since keyed would need to observe a very large number of messages)
- choose hash function used based on speed verses security constraints

Public-Key Cryptography Principles

- The use of two keys has consequences in: key distribution, confidentiality and authentication.
- The scheme has six ingredients (see Figure 3.7)
  - Plaintext
  - Encryption algorithm
  - Public and private key
  - Ciphertext
  - Decryption algorithm

Encryption using Public-Key system



Applications for Public-Key Cryptosystems

- Three categories:
  - **Encryption/decryption:** The sender encrypts a message with the recipient's public key.
  - **Digital signature:** The sender "signs" a message with its private key.
  - **Key echange:** Two sides cooperate two exhange a session key.

Requirements for Public-Key Cryptography

1. Computationally easy for a party B to generate a pair (public key KU_b, private key KR_b)
2. Easy for sender to generate ciphertext: $C = E_{KUb}(M)$
3. Easy for the receiver to decrypt ciphertect using private key:

$$M = D_{KRb}(C) = D_{KRb}[E_{KUb}(M)]$$

Requirements for Public-Key Cryptography

4. Computationally infeasible to determine private key (KR_b) knowing public key (KU_b)
5. Computationally infeasible to recover message M, knowing KU_b and ciphertext *C*
6. Either of the two keys can be used for encryption, with the other used for decryption:

$$M = D_{KRb}[E_{KUb}(M)] = D_{KUb}[E_{KRb}(M)]$$

Public-Key Cryptographic Algorithms

- RSA and Diffie-Hellman
- **RSA** - Ron Rives, Adi Shamir and Len Adleman at MIT, in 1977.
  - RSA is a block cipher
  - The most widely implemented
- **Diffie-Hellman**
  - Echange a secret key securely
  - Compute discrete logarithms

The RSA Algorithm – Key Generation

1. Select p,q — p and q both prime
2. Calculate $n = p \times q$
3. Calculate $\Phi(n) = (p-1)(q-1)$
4. Select integer e — $\gcd(\Phi(n), e) = 1; 1 < e < \Phi(n)$
5. Calculate d — $d = e^{-1} \bmod \Phi(n)$
6. Public Key — KU = {e,n}
7. Private key — KR = {d,n}

Example of RSA Algorithm

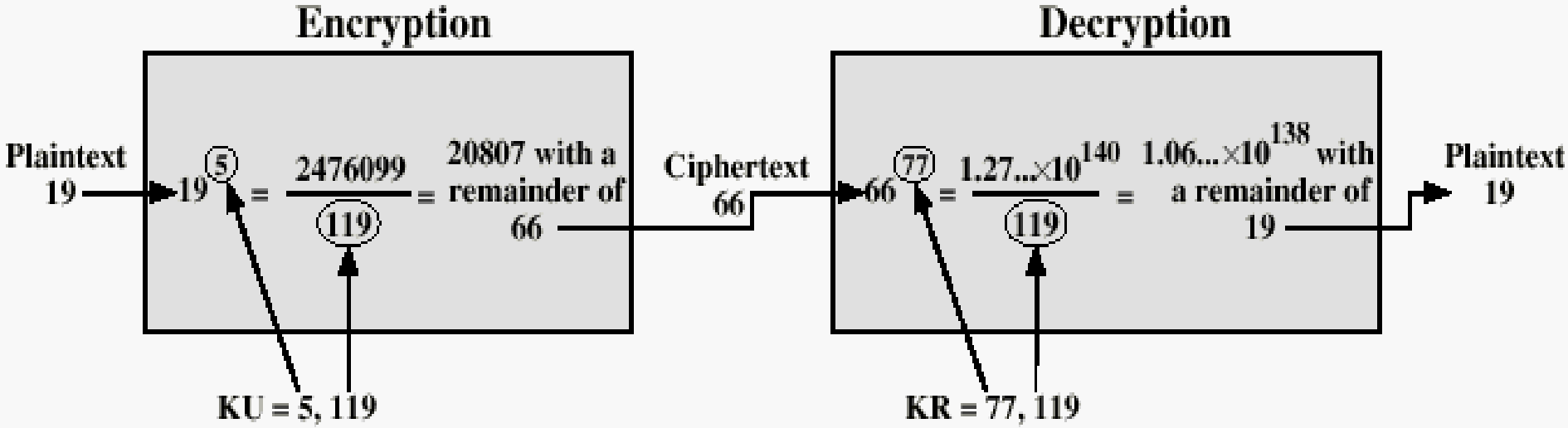


Figure 3.9 Example of RSA Algorithm

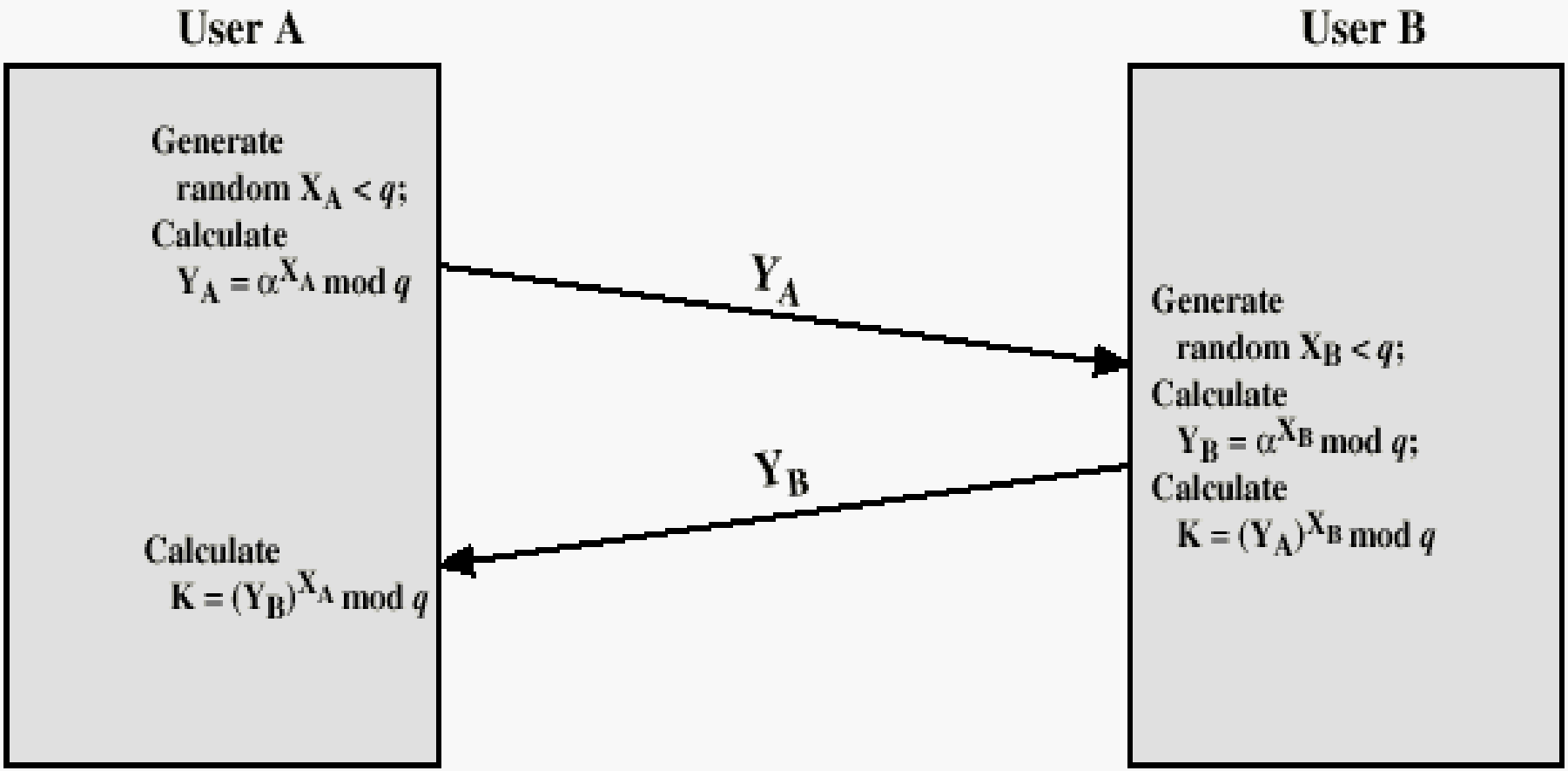
The RSA Algorithm - Encryption

- Plaintext: $M<n$
- Ciphertext: $C = M^e \pmod{n}$

The RSA Algorithm - Decryption

- Ciphertext: C
- Plaintext: $M = C^d \pmod{n}$

Diffie-Hellman Key Echange



Other Public-Key Cryptographic Algorithms

- Digital Signature Standard (DSS)
  - Makes use of the SHA-1
  - Not for encryption or key echange
- Elliptic-Curve Cryptography (ECC)
  - Good for smaller bit size
  - Low confidence level, compared with RSA
  - Very complex

Key Management
Public-Key Certificate Use