Unit VIII Part 2

Public Key Cryptography and Message Authentication

- Encryption protection techniques against passive attack and active attacks are different.
- Protection against such attacks is known as message authentication

Approaches to Message Authentication

Using conventional encryption

- Symmetric encryption alone is not a suitable tool for data authentication
  - We assume that only the sender and receiver share a key, so only the genuine sender would be able to encrypt a message successfully
  - The receiver assumes that no alterations have been made and that sequencing is proper if the message includes an error detection code and a sequence number

Without message encryption

- An authentication tag is generated and appended to each message for transmission
- The message itself is not encrypted and can be read at the destination independent of the(without) authentication function at the destination
- Because the message is not encrypted, message confidentiality is not provided

- Three situation where message authentication without confidentiality is preferable.
- 1. Application where the same message is broadcast to number of destination. e.g network is now unavailable broadcast message to multiple destination.
- Another possible scenario is an exchange in which one side has a heavy load and cannot afford the time to decrypt all incoming messages. and authentication is carried out on selective basis (random checking)
- Authentication of computer program in plaintext is an attractive service . If message authentication tag were attached to the program, it could be checked whenever assurance is required of the integrity of the program.

MAC

- One authentication technique involves the use of secret key to generate a small block of data known as message authentication.(MAC)
- This technique assume that two communication parties say A and B, share a common secret key . When A has message to send to B. It calculates the message authentication code as a function of the message and the key.
- The message + code are transmitted to the intended recipient.
- The recipient perform the same calculation on received message , using same secret key , to generate a new message authentication code.
- The received code is compared to the calculated code.

- If we assume that only the receiver an sender know the identity of the secret key and if the received code matches the calculated code then
- 1. receiver is assured that message has not been altered. If attacker alter the message but does not alter the code then the reviver's calculation of code will differ from received code.
- 2. receiver is assured that it is send from the authorised sender. Because no one else knows the secret key, no one else could prepare a message with a proper code.
- 3. if the message includes a sequence number (used in TCP) then the receiver can be assured of the proper sequence, because an attacker cannot successfully alter the sequence number.

Approaches to Message Authentication (MAC)

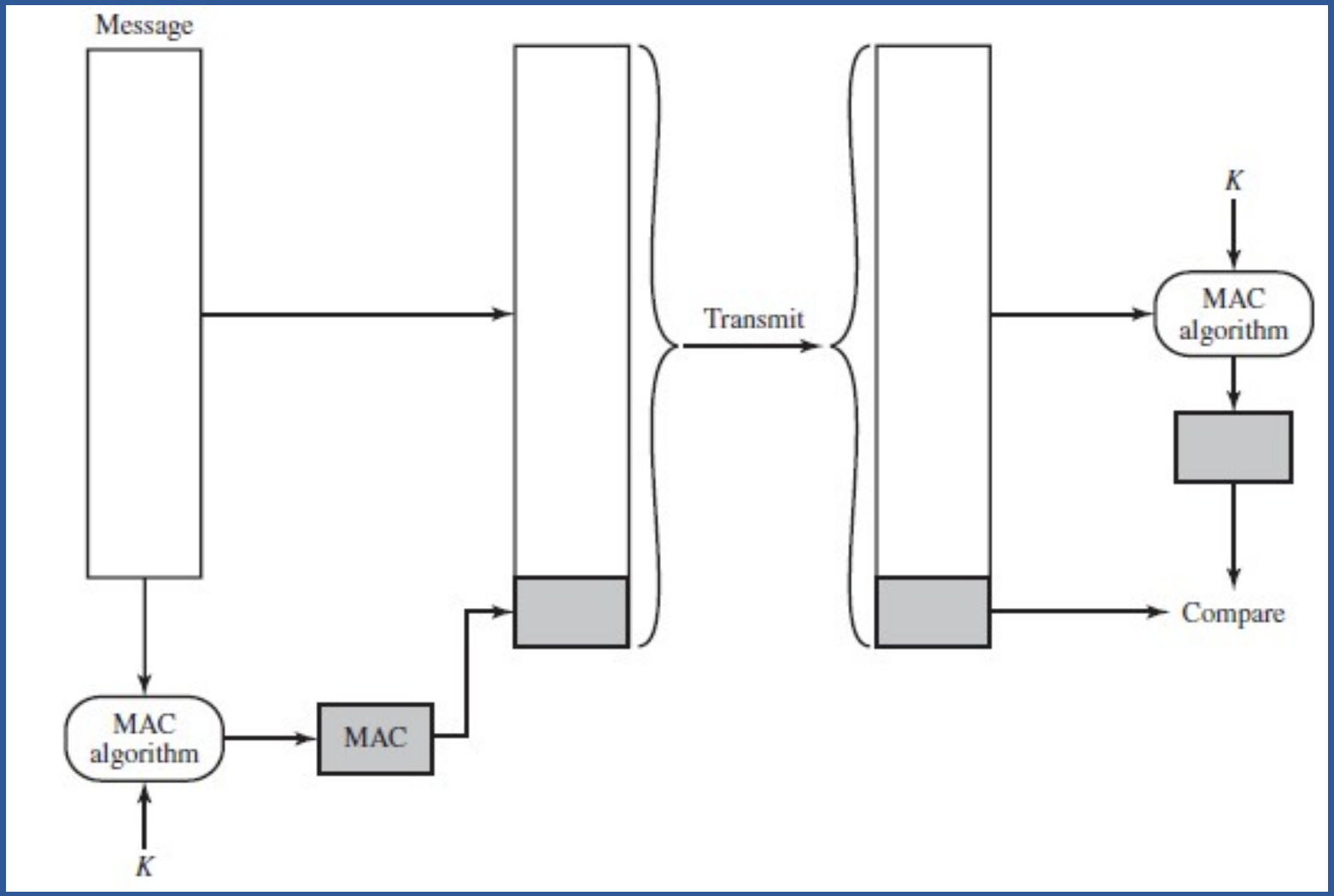


Figure 3.1 Message Authentication Using a Message Authentication Code (MAC)

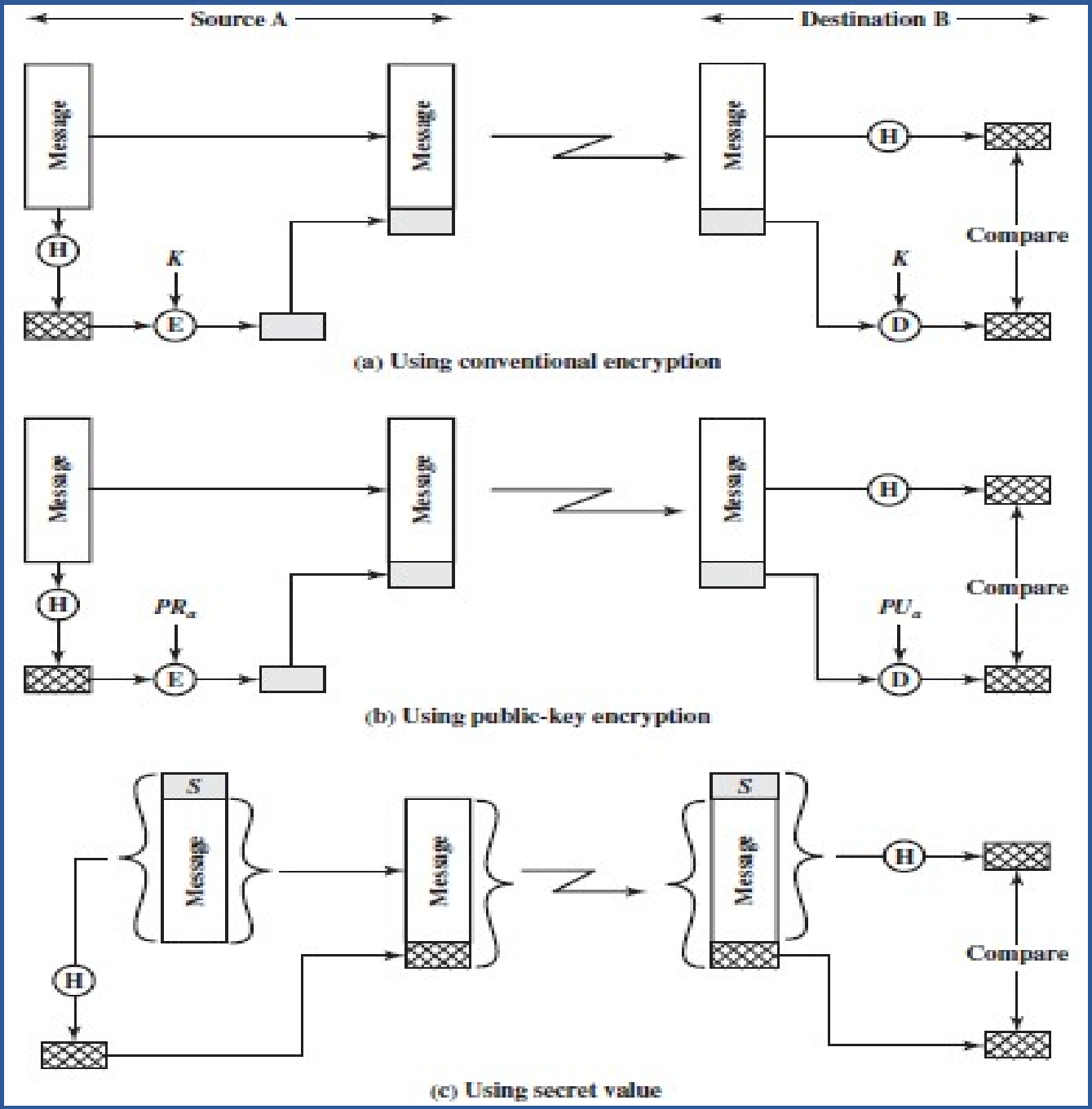
- DES is used to generate the encrypted message and the last number of bits of cipher text are used as code.
- Authentication algorithm need not be reversible. Mathamatical properties of authentication function is less vulnerable to being broken than encryption.

Hashing Technique

One-way Hash Functions

- It Accepts a variable-size message *M* as input and produces a fixed-size message digest H(*M*) as output
- Does not 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.
- Fig (a) message digest can be encrypted using conventional encryption. Sender and receiver is sharing a secrete key and authenticity is assured.

Figure 3.2 Message Authentication Using a One-Way Hash Function



- In fig(b) message digest can be encrypted using public key encryption. So it does not required to distribute the key to sender and receiver.
- In fig(c) uses the hash function but no encryption for message authentication.
- In this technique both sender and receiver share the common secret key sk .
- When A send message to b it contact the key with the message and calculate the hash function. Then it send message and hash function value to the B.
- At receiver side B already have the secret key so it recompute the hash function of secret key+message. And verify it.
- Because of key is not sent it is not possible for attacker to intercept the message. And secret key is remain secret so it is difficult for attacker to generate the false message.

Secure Hash Functions

- Is important not only in message authentication but in digital signatures
- Purpose is to produce a “fingerprint” of a file, message, or other block of data
- Most important secure hash function is **SHA**
- To be useful for message authentication, a hash function H must have the following properties:

Secure Hash Functions

- Hash Function Requirements
- The purpose of a hash function is to produce a “fingerprint” of a file, message, or other block of data. To be useful for message authentication, a hash function H must have the following properties:

1. H can be applied to a block of data of any size.

2. H produces a fixed-length output.

3. H(x) is relatively easy to compute for any given x, making both hardware and software implementations practical.

4. For any given code h, it is computationally infeasible to find x such that H(x)=h. A hash function with this property is referred to as one-way or preimage resistant.

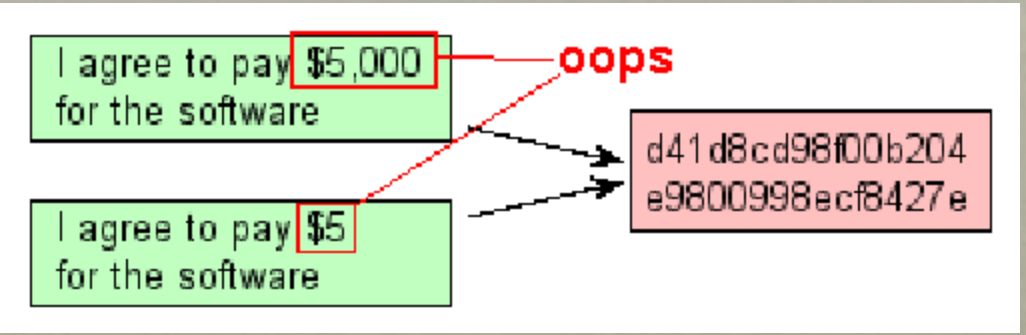
Secure Hash Functions

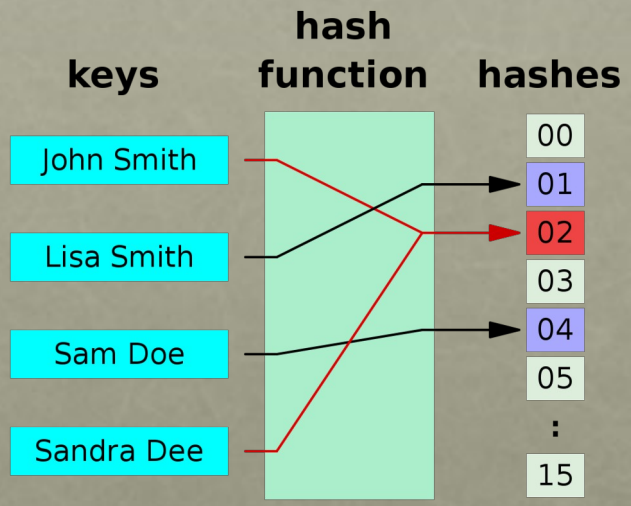
5. For any given block x, it is computationally infeasible to find y does not equal to x with H(y)= H(x). A hash function with this property is referred as weak collision resistant.

6. It is computationally infeasible to find any pair (x, y) such that H(x)= H(y). A hash function with this property is referred to as collision resistant. This is sometimes referred to as strong collision resistant.

- If hash function is not one way, attacker can easily hack secret value. And then can obtain message M.
- Second preimage property gurantees that it is impossible to find an alternative message with the same hash value as a given message.
- IF hash function statisfy first 5 properties then it is refereed as weak hash function. If sixth property is also satisfied then it is reoffered as strong hash function.

collision


I agree to pay $5,000
for the software
oops
d41d8cd98f00b204
e9800998ecf8427e
I agree to pay $5
for the software


keys
hash function
hashes
John Smith
Lisa Smith
Sam Doe
Sandra Dee
00
01
02
03
04
05
:
15

Security of Hash Functions

Security of Hash Functions

- There are two approaches to **attacking a secure hash function:**
- **Cryptanalysis**
- Involves exploiting logical weaknesses in the algorithm
- **Brute-force attack**
- The strength of a hash function against this attack depends solely on the length of the hash code produced by the algorithm
- For hash code of length n, the efforts required would be

-

Simple hash functions

Simple hash functions

- All hash functions operate using the following general principles.
- The input (message, file, etc.) 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.

- Bit-by-bit XOR operation
  - Produces simple parity
  - Known as longitudinal redundancy check
  - Effective for random data for redundancy check

	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

- Shift (Rotated XOR)
  - Set the n-bit hash value to zero
  - Process each successive n bit block of data
    - Rotate left
    - XOR the block into the hash value

6	3	6	f	6	4	6	5	7	2								
	6	3	6	f	6	4	6	5	7	2							
		6	3	6	f	6	4	6	5	7	2						
			6	3	6	f	6	4	6	5	7	2					
				6	3	6	f	6	4	6	5	7	2				
					6	3	6	f	6	4	6	5	7	2			
						6	3	6	f	6	4	6	5	7	2		
							6	3	6	f	6	4	6	5	7	2	
								6	3	6	f	6	4	6	5	7	2
6	5	3	c	a	e	8	d	a	8	e	d	b	4	2	6	0	5	2

The SHA Secure Hash function

- SHA was developed by NIST and published as a federal information processing standard (FIPS 180) in 1993
- Was revised in 1995 as SHA-1 and published as FIPS 180-1
  - The actual standards document is entitled “Secure Hash Standard”
- SHA-1 produces a hash value of 160 bits.

- In 2002, NIST produced a revised version that defined three new versions of SHA with hash value lengths of 256, 384, and 512 bits known as SHA-256, SHA-384, and SHA-512, respectively.
- Collectively, these hash algorithms are known as SHA-2 .
- These new versions have the same underlying structure and use the same types of modular arithmetic and logical binary operations as SHA-1.
-

Table 3.1
Comparison of SHA Parameters

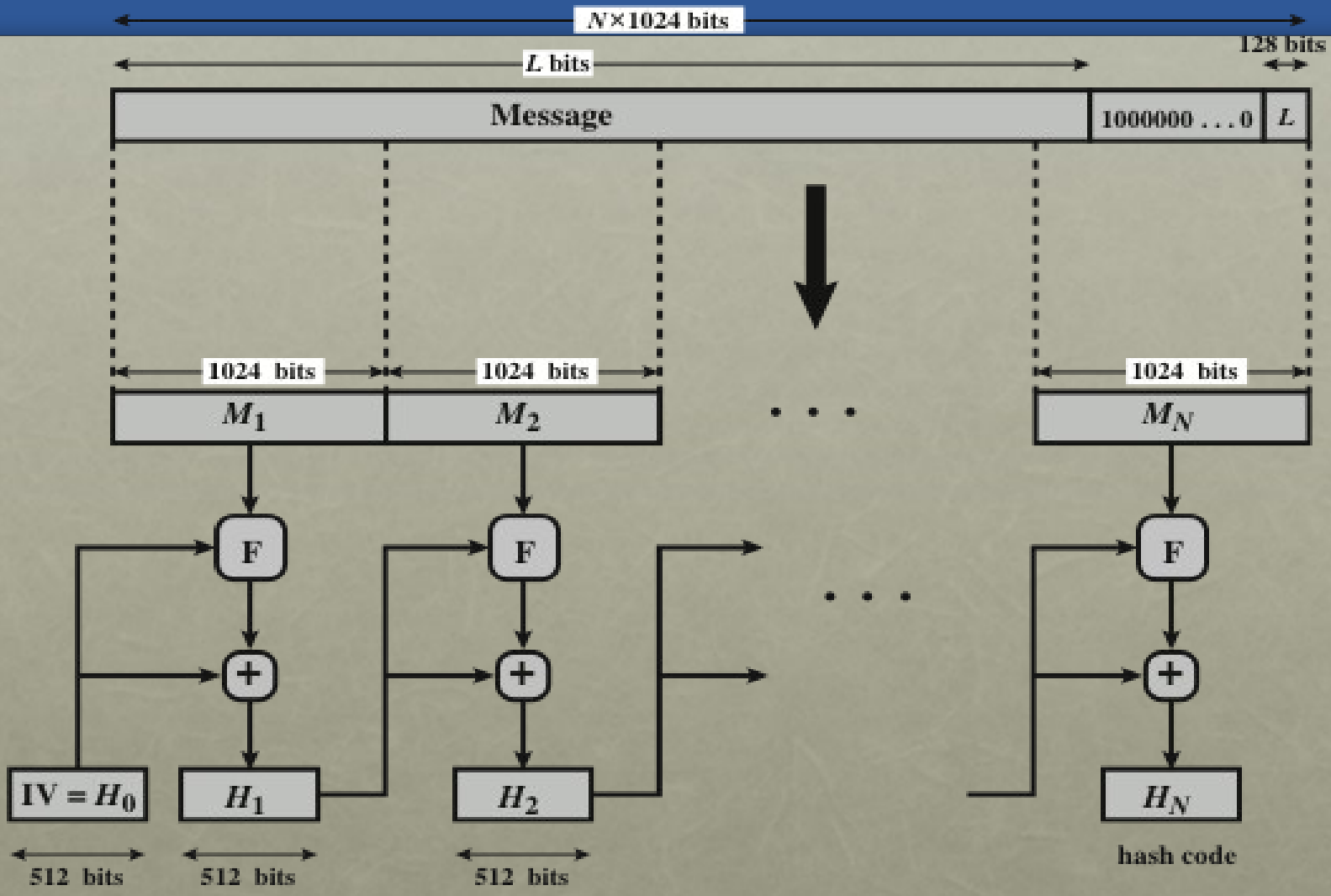
	SHA-1	SHA-224	SHA-256	SHA-384	SHA-512
Message Digest Size	160	224	256	384	512
Message Size	$< 2^{64}$	$< 2^{64}$	$< 2^{64}$	$< 2^{128}$	$< 2^{128}$
Block Size	512	512	512	1024	1024
Word Size	32	32	32	64	64
Number of Steps	80	64	64	80	80

Note: All sizes are measured in bits.

SHA 512

- The input is processed in 1024-bit blocks. and produces as output a 512-bit message digest.
- That is the message is broken into 1024-bit chunks,
- the initial hash values and round constants are extended to 64 bits,
- For each block 80 rounds are performed.
- Each round is used constant.
- Out put of one block is input to the next block

- Step 1 : Append padding bits
- Padding is always added even if message is already of the desired length. (that is length =896 modulo 1024).
- Padding bits is in the range of 1 to 1024.
- Padding bit consist of single 1 bit followed by 0s.
- Step 2 : Append Length
- A block of 128 bit is appended to the message. And contain length of original message (before padding)
- So the out put of first two steps create a message that is in multiple of 1024 bits


N×1024 bits
L bits
128 bits
Message
1000000 . . . 0
L
1024 bits
1024 bits
1024 bits
M1
M2
MN
F
F
F
IV = H0
H1
H2
HN
512 bits
512 bits
512 bits
hash code
+ = word-by-word addition mod 2^64

Step 3 : initialize hash buffer

512 bit buffer is used to hold the value of the hash function. The buffer are 64 bit registers

A,b,c...e,f , g ,h total 8 buffers (64*8 = 512)

Step 4 : process message in 1024 blocks:

- Each round takes the 512 bit buffer values and updates the content of the buffer.
- At input to the first round buffer has the value of the intermediate hash value Hi-1.
- Generate words from 0 to 79 from plain text and each word is of 64 bit that is 1024 is expanded in 80 words of 64 bit each .
- And each round words wi and constant value ki and buffer valued are being
- Process and generate 512 bit output in register abcdefgh

- Out put of 80^{th} round perform
- addition modulo 2^{64} with previous hash value and generate hash value.
-

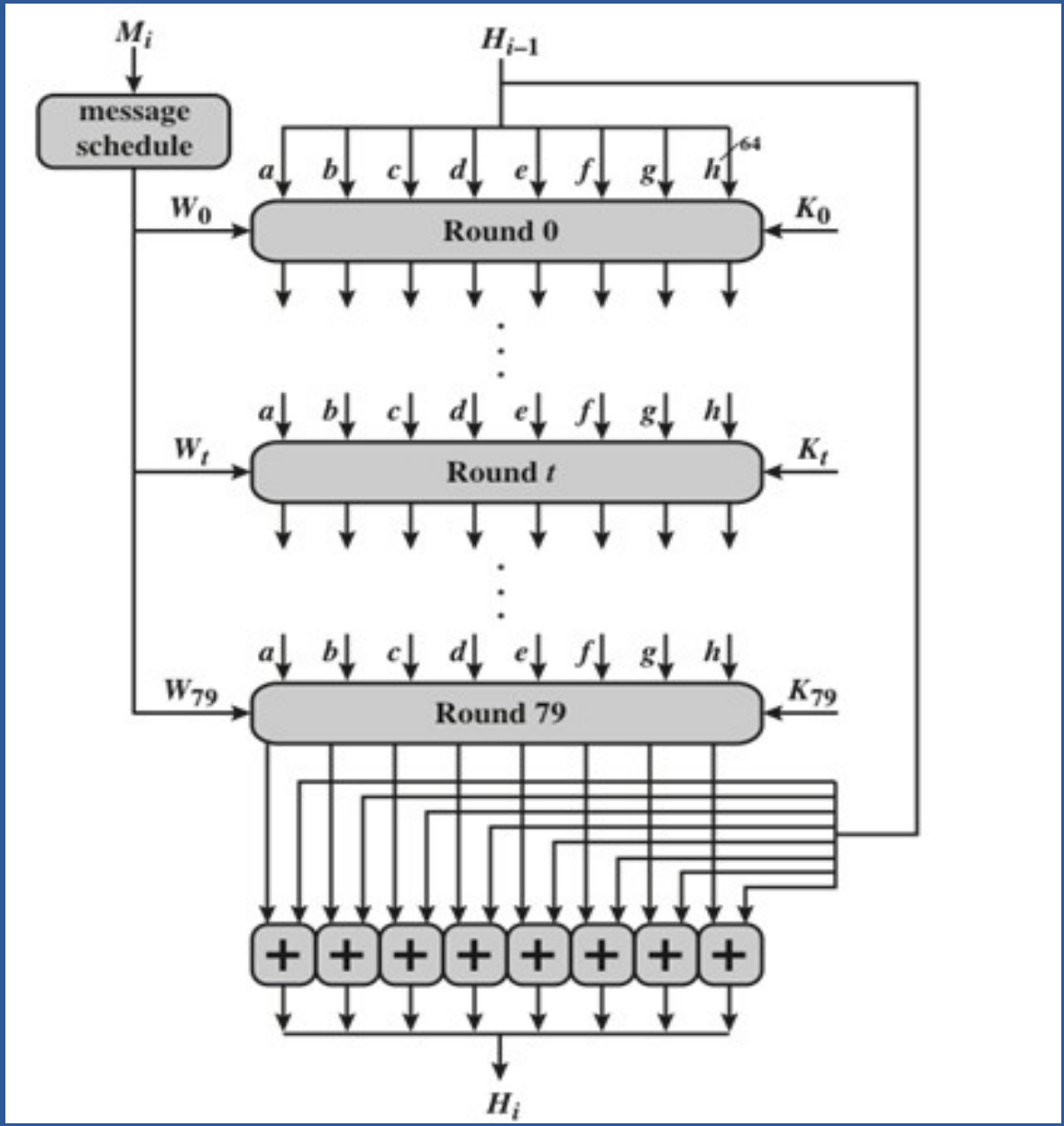


Figure 3.5 SHA-512 Processing of a Single 1024-Bit Block

Message Authentication Codes HMAC

- There has been an increased interest in developing a MAC derived from a cryptographic hash code, such as SHA-1
  - Cryptographic hash functions generally execute faster in software than conventional encryption algorithms such as DES.
  - Library code for cryptographic hash functions is widely available
  - The approach that has received the most support is HMAC

- When secret key information is included with the data that is processed by a cryptographic hash function, the resulting hash is known as an HMAC.
-

HMAC

- Has been issued as RFC 2104
- Has been chosen as the mandatory-to-implement MAC for IP Security
- Is used in other Internet protocols, such as Transport Layer Security (TLS) and Secure Electronic Transaction (SET)

HMAC Design Objectives

1) To use, without modifications, available hash functions --- in particular, hash functions that perform well in software, and for which code is freely and widely available

2) To allow for easy replace ability of the embedded hash function in case faster or more secure hash functions are found or required

3) To preserve the original performance of the hash function without incurring a significant degradation

4) To use and handle keys in a simple way

- HMAC treats the hash function as a “black box” . Which have adv that Hash function can be used as a module in implementation and code is ready to use without modification.

- If it required to be change then it will be easily replaceable by removing existing module and adding new module.

H = embedded hash function (e.g., SHA-1)

M = message input to HMAC (including the padding specified in the embedded hash function)

Y_i = ith block of M, $0 \leq i \leq (L - 1)$

L = number of blocks in M

b = number of bits in a block

n = length of hash code produced by embedded hash function

K = secret key; if key length is greater than b, the key is input to the hash function to produce an n-bit key; recommended length is $> n$

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

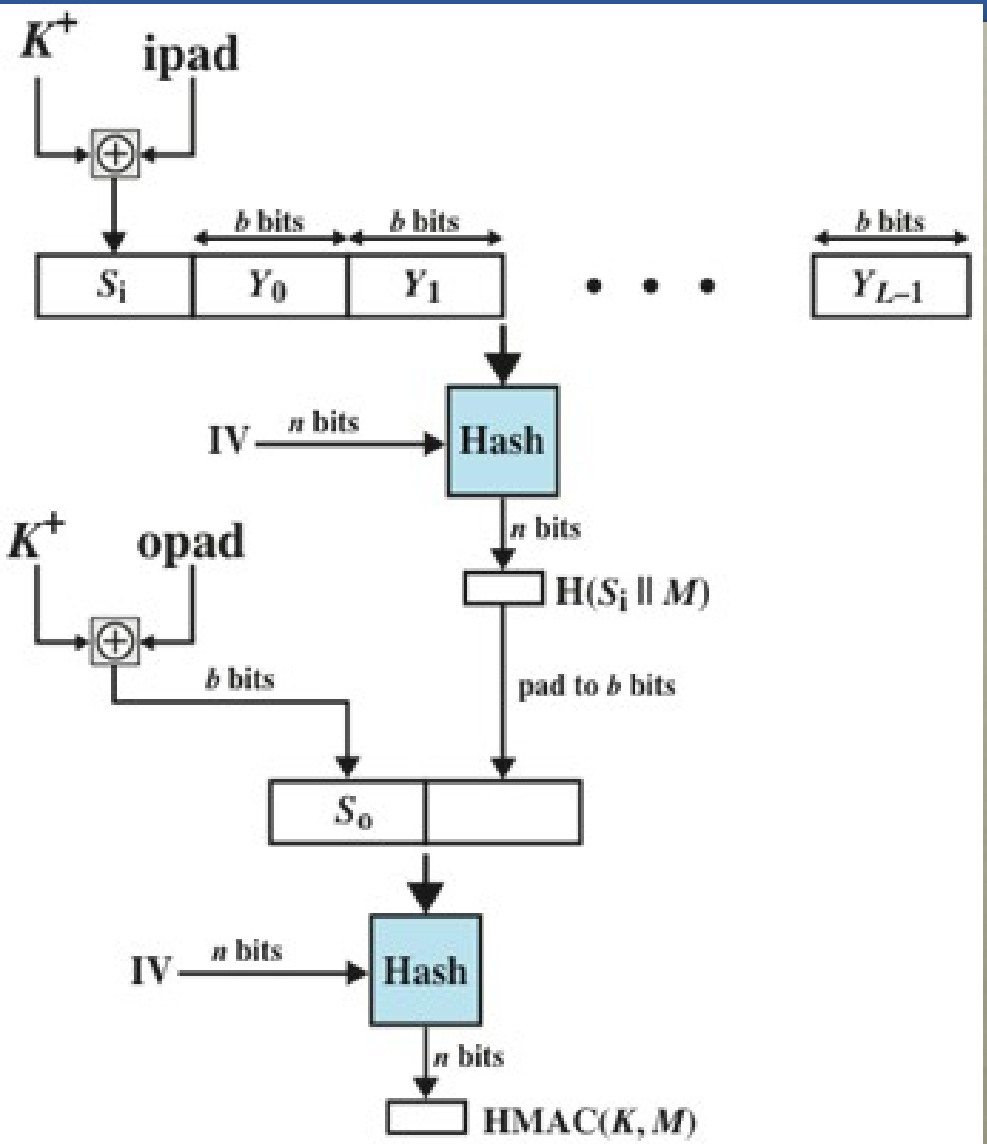
opad = 01011100 (5C in hexadecimal) repeated $b/8$ times

HMAC

- Takes a message M containing blocks of length *b* bits.
- An input signature is padded to the left of the message and the whole is given as input to a hash function which gives us a temporary message digest MD'.
- MD' again is appended to an output signature and the whole is applied a hash function again, the result is our final message digest MD.

$$\mathrm{HMAC}(K, M) = \mathrm{H}[(K^{+} \oplus \mathrm{opad}) \| \mathrm{H}[(K^{+} \oplus \mathrm{ipad}) \| M]]$$

HMAC Algorithm



HMAC overall Operations

Figure 3.6 HMAC Structure

HMAC Algorithm

1. Append zeros to the left end of K to create a b-bit string K+ (e.g., if k is of length 160 bit and b = 512 bit then 352 bits are padded with zero)
2. XOR (bitwise exclusive-OR) 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 So.
6. Append the hash result from step 4 to So.
7. Apply H to the stream generated in step 6 and output the result..

- Xor with ipad and opad results in flipping one half of the bits of K
- Si and So is pseudorandomly generated two keys from K.

MACs BASED on block cipher

-
- **Cipher Based Message Authentication Code (CMAC)**
  - Used with AES and triple DES
  - First, let us consider 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 (*M*1,*M*2, . . .,*Mn*).
  - The algorithm makes use of a *k*-bit encryption key *K* and an *n*-bit key, *K*1.
  - For AES, the key size *k* is 128, 192, or 256 bits; for triple DES, the key size is 112 or 168 bits.
  -

$$C_1 = \mathrm{E}(K, M_1)$$
$$C_2 = \mathrm{E}(K, [M_2 \oplus C_1])$$
$$C_3 = \mathrm{E}(K, [M_3 \oplus C_2])$$
•
•
•
$$C_n = \mathrm{E}(K, [M_N \oplus C_{n-1} \oplus K_1])$$
$$T = \mathrm{MSB}_{Tlen}(C_n)$$

CMAC operations

T = message authentication code, also referred to as the tag

$Tlen$ = bit length of T

$\mathrm{MSB}_s(X)$ = the s leftmost bits of the bit string X

CMAC

- If the message is not an integer multiple of the cipher block length, then the final block is padded to the right (least significant bits) with a 1 and as many 0s as necessary so that the final block is also of length b.
- The CMAC operation then proceeds as before, except that a different n-bit key K_2 is used instead of K_1.

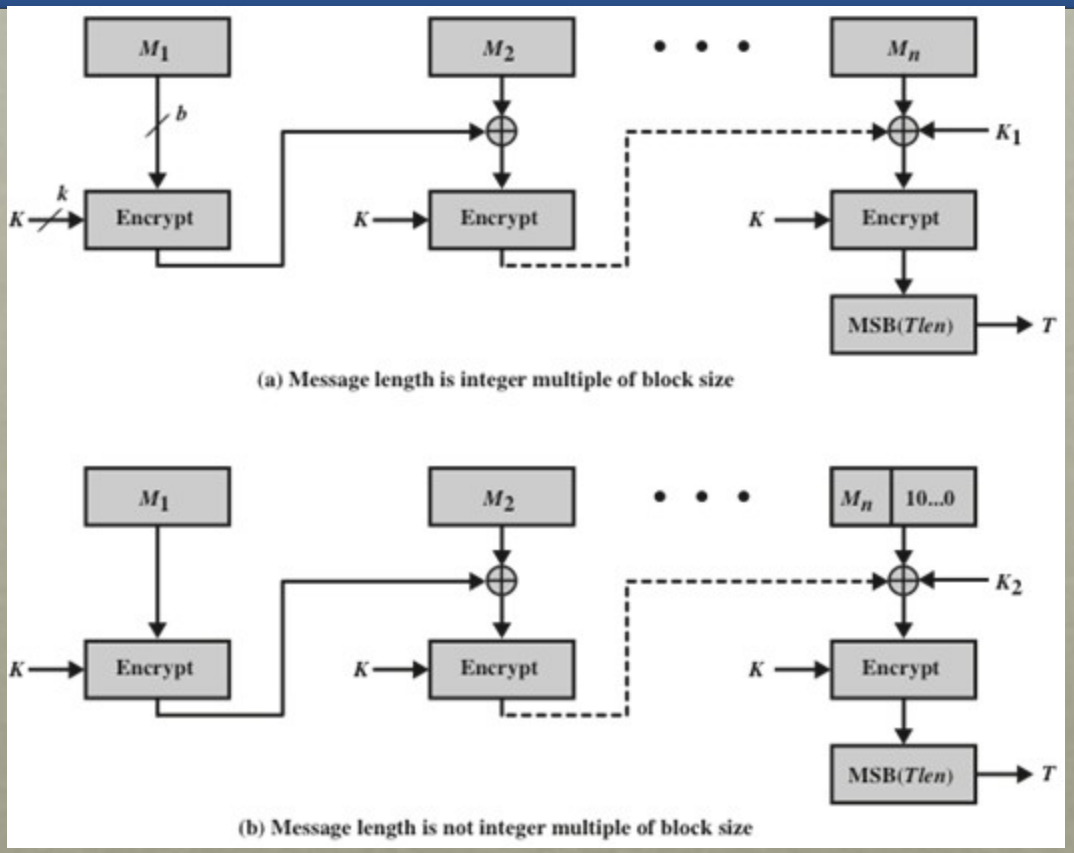
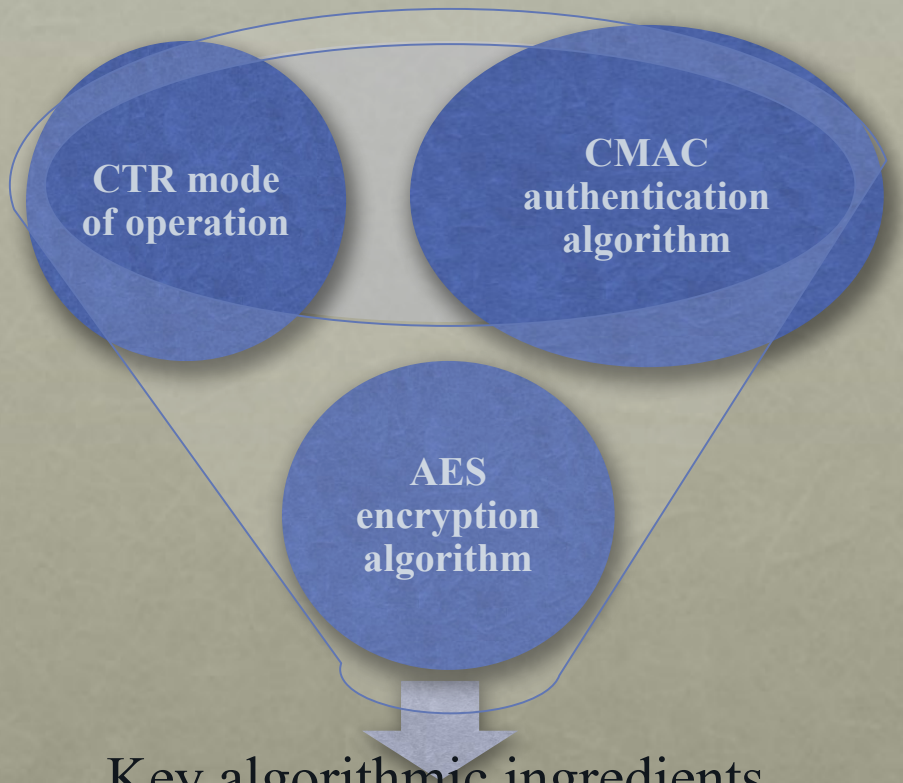


Figure 3.7 Cipher-Based Message Authentication Code (CMAC)

Counter with Cipher Block Chaining-Message Authentication Code (CCM)

- "**Authenticated encryption" is a term used to describe encryption systems that simultaneously protect confidentiality and authenticity of communications**

- A single key is used for both encryption and MAC algorithms

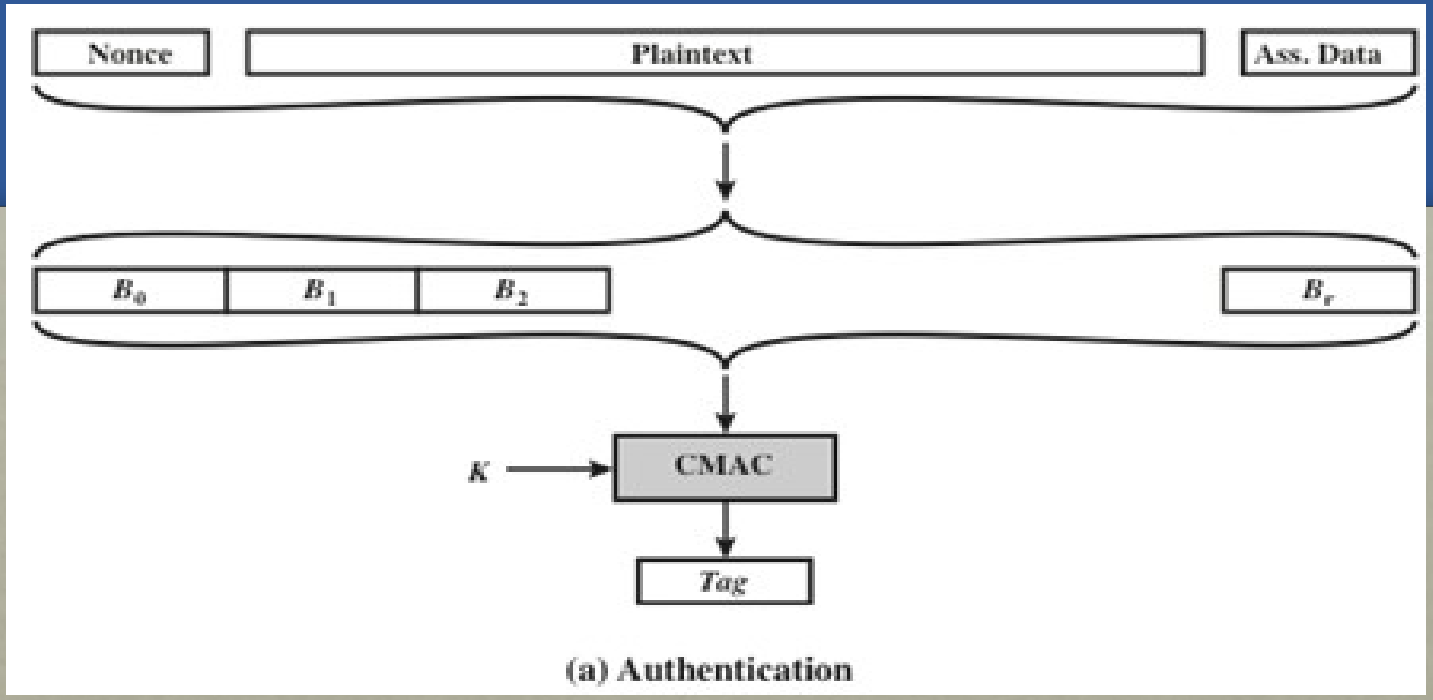


Key algorithmic ingredients for CCM

CCM

Input to CCM : Consists of 3 elements

1. Data that will be both authenticated and encrypted. This is the plaintext message P of data block.
2. Associated data A that will be **authenticated but not encrypted.** An example is a protocol header that must be transmitted in the clear form for proper protocol operation but which needs to be authenticated.
3. A nonce N (Number only once)that is assigned to the payload (part of transmitted data) and the associated data. This is a unique value that is different for every instance during the lifetime of a protocol association and is intended to prevent replay attacks and certain other types of attacks.



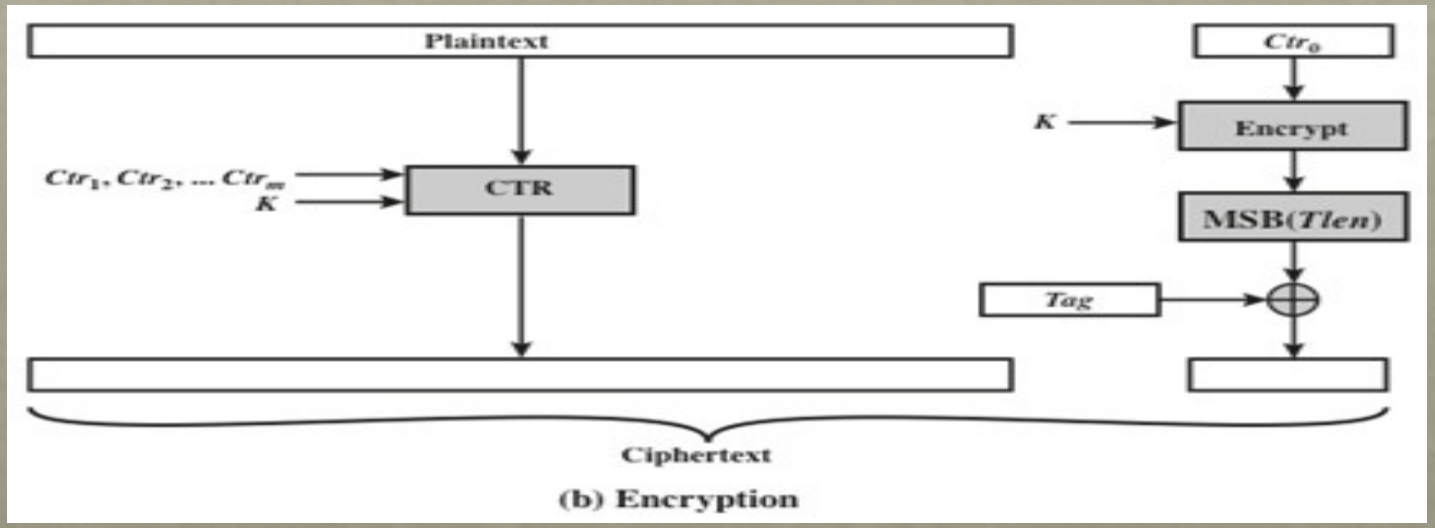


Figure 3.8 Counter with Cipher Block Chaining-Message Authentication Code (CCM)

Operations on CCM

- For (a) Authentication the input includes the nonce, associated data and plain text message. And it is formatted to sequence of blocks B0 , B1..Br.
- The first block contains the nonce plus length of nonce, length of associated data and length of plain text. After that blocks of associated data and then it is followed by plain text blocks.
- Blocks is input to the CMAC algo.which produce the MAC value.

Ccm operations

- For (b)Encryption
  - Sequence of counter is generated .
  - The authentication tag is encrypted in CTR using the single counter Ctr0.
  - The *Tlen* most significant bits of the output are XORed with the tag to produce an encrypted tag.
  - The remaining counters are used for the CTR mode encryption of the plaintext
  - The encrypted plaintext is concatenated with the encrypted tag to form the ciphertext output

Public-Key Encryption Structure

- First publicly proposed by Diffie and Hellman in 1976
- Based on mathematical functions rather than on simple operations on bit patterns
- Is asymmetric, involving the use of two separate keys

Misconceptions: for public key encryption

- Public-key encryption is more secure from cryptanalysis than conventional encryption
- Public-key encryption is a general-purpose technique that has made conventional encryption obsolete
- There is a feeling that key distribution is trivial when using public-key encryption, compared to the rather cumbersome handshaking involved with key distribution centers for conventional encryption

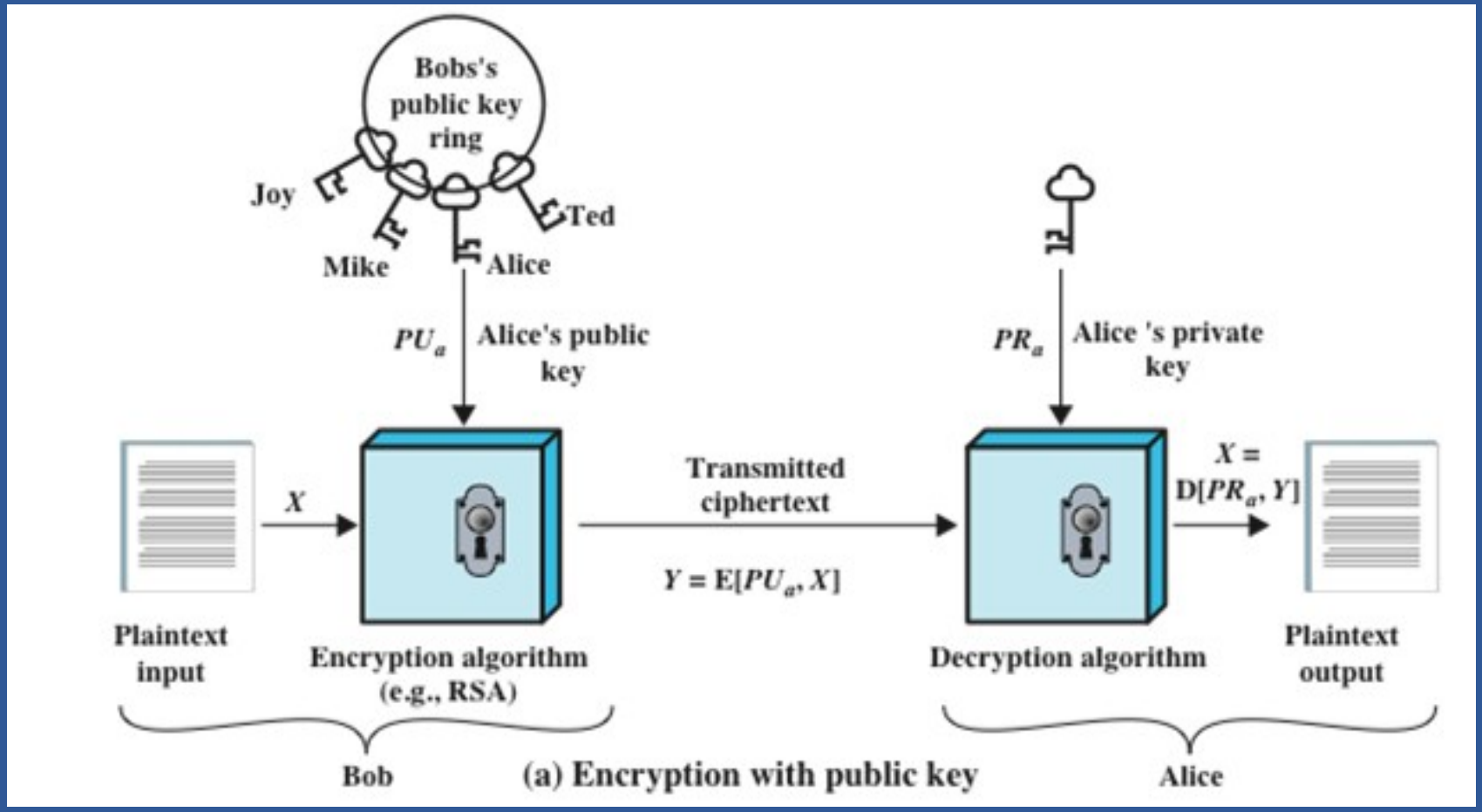


Figure 3.9 Public-Key Cryptography

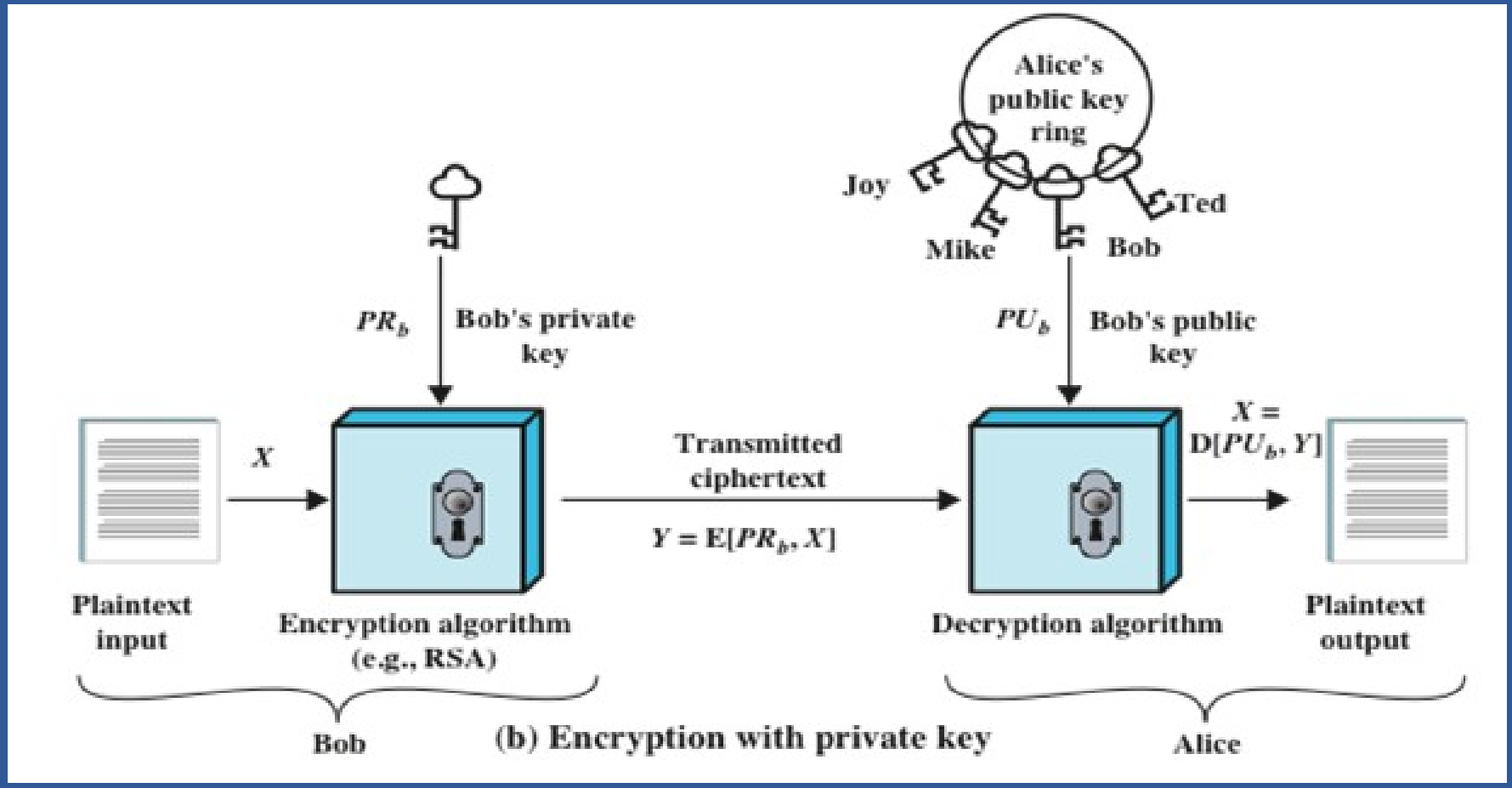


Figure 3.9 Public-Key Cryptography

- **Six element**
- Plain text
- Encryption algorithm
- Public key & private key : one is used for encryption other is used for decryption.
- Cipher text
- Decryption algorithm
- Public key is made public for other use , while the private key is known only to owner.
- **Steps for the algorithm**

1. each user generate a pair of keys for encryption and decryption
2. one key placed in public register which is become a public key. Each user maintain the public key obtain from the others.
3. if A wish to send a private message to the B. A will encrypt the message using public key of B.
4 When B receive the message , B will decrypt the message using own private key.

No other can decrypt the message because B knows only own private key.

Applications for Public-Key Cryptosystems

- Public-key systems are characterized by the use of a cryptographic type of algorithm with two keys, one held private and one available publicly
- Depending on the application, the sender uses either the sender's private key, the receiver's public key, or both to perform some type of cryptographic function

The use of public-key cryptosystems can be classified into 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 exchange	Two sides cooperate to exchange a session key

Table 3.2
Applications for Public-key Cryptosystems

Algorithm	Encryption/Decryption	Digital Signature	Key Exchange
RSA	Yes	Yes	Yes
Diffie-Hellman	No	No	Yes
DSS	No	Yes	No
Elliptic Curve	Yes	Yes	Yes

RSA

- RSA scheme is most widely accepted and implemented approach to public-key encryption
- RSA is a block cipher in which the plaintext and ciphertext are integers between 0 and n-1 for some n
- **$C = M^e \bmod n$**
- **$M = C^d \bmod n = (M^e)^d \bmod n = M^{ed} \bmod n$**
- The following requirement must meet
  1. It is possible to find values of e, d, n such that $M^{ed} \bmod n$ M for all M < n.
  2. It is relatively easy to calculate M^e and C^d for all values of M< n.
  3. It is infeasible to determine d given e and n.

Steps of RSA

1. Select two prime numbers p and q and calculating their product **n**, which is the modulus for encryption and decryption.
2. Next, we need the quantity f(n), referred to as the Euler totient of n, which is the number of positive integers less than n and relatively prime to n.
3. Then select an integer e that is relatively prime to f(n) [i.e., the greatest common divisor of e and f(n) is 1].
4. Finally, calculate d as the multiplicative inverse of e, modulo f(n). It can be shown that d and e have the desired properties.

Key Generation

Select p, q	p and q both prime, $p \neq q$
Calculate $n = p \times q$	
Calculate $\phi(n) = (p - 1)(q - 1)$	
Select integer e	$\gcd(\phi(n), e) = 1;\ 1 < e < \phi(n)$
Calculate d	$de \bmod \phi(n) = 1$
Public key	$KU = \{e, n\}$
Private key	$KR = \{d, n\}$

Encryption

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

Decryption

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

Figure 3.10 RSA Crypto System

p=17
q= 11

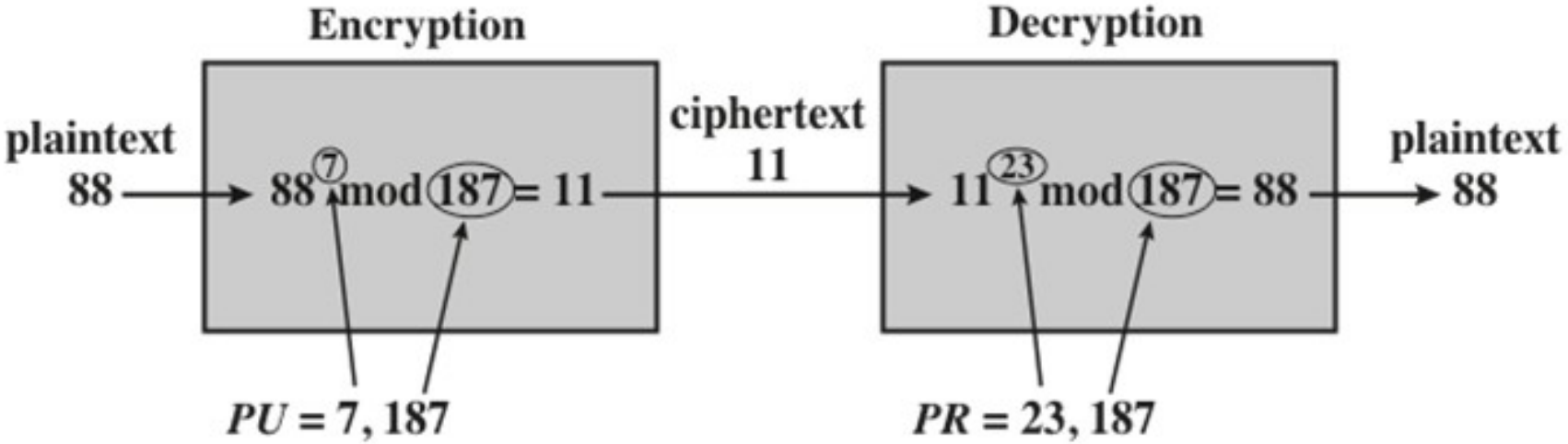


Figure 3.11 Example of RSA Algorithm

- RSA Example
- for p=13 q=17 , e = 7 then plaintext M = 10 then find public key and private key and perform encryption.
- N = 13*17 =221 , Φ(n) = (p-1)*(q-1) = 12*16 = 192
- Step 2 : select e such that gcd(192,e) =1
- Step 3 : d*e mod Φ(n) =1 that is d= **((Φ(n)*i) +1)/e** such that it gives whole number
- (192*1)+1 = 193/7 = 27.53
- (192*2)+1 = 385/7 = 55 that is d = 55.
- 55*7 mod 192 =1
- now public key is (7,221) and private key is (55,221)

- Now M = 10
- So cipher text is :
- $M^e \bmod n = 10^{7} \bmod 221 = 192$
- 192 is encrypted key
- now decript 192
- $C^d \bmod n$ 192 55 mod 221 = 10

Diffie-Hellman Key Exchange

- First published public-key algorithm
- A number of commercial products employ this key exchange technique
- Purpose of the algorithm is to enable two users to exchange a secret key securely that then can be used for subsequent encryption of messages
- The algorithm itself is limited to the exchange of the keys
- Depends for its effectiveness on the difficulty of computing discrete logarithms

Diffie-Hellman Key Exchange

- Define a primitive root of a prime number p as one whose powers generate all the integers from 1 to p. *i.e.*. if a is a primitive root of the prime number p, then the numbers

  $a \bmod p$, $a^2 \bmod p$, . . ., $ap^{-1} \bmod p$
- are distinct and consist of the integers from 1 through p 1 in some permutation
- For any integer b less than p and a primitive root a of prime number p, one can find a unique exponent i such that

  $b = a^i \bmod p$ $0 <= i <= (p- 1)$
- The exponent i is referred to as the discrete logarithm, or index, of b for the base a, mod p.

Global Public Elements

q	prime number
α	$\alpha < q$ and α a primitive root of q

User A Key Generation

Select private X_A	$X_A < q$
Calculate public Y_A	$Y_A = \alpha^{X_A} \bmod q$

User B Key Generation

Select private X_B	$X_B < q$
Calculate public Y_B	$Y_B = \alpha^{X_B} \bmod q$

Generation of Secret Key by User A

$K = (Y_B)^{X_A} \bmod q$

Generation of Secret Key by User B

$K = (Y_A)^{X_B} \bmod q$

Figure 3.12 The Diffie-Hellman Key Exchange Algorithm

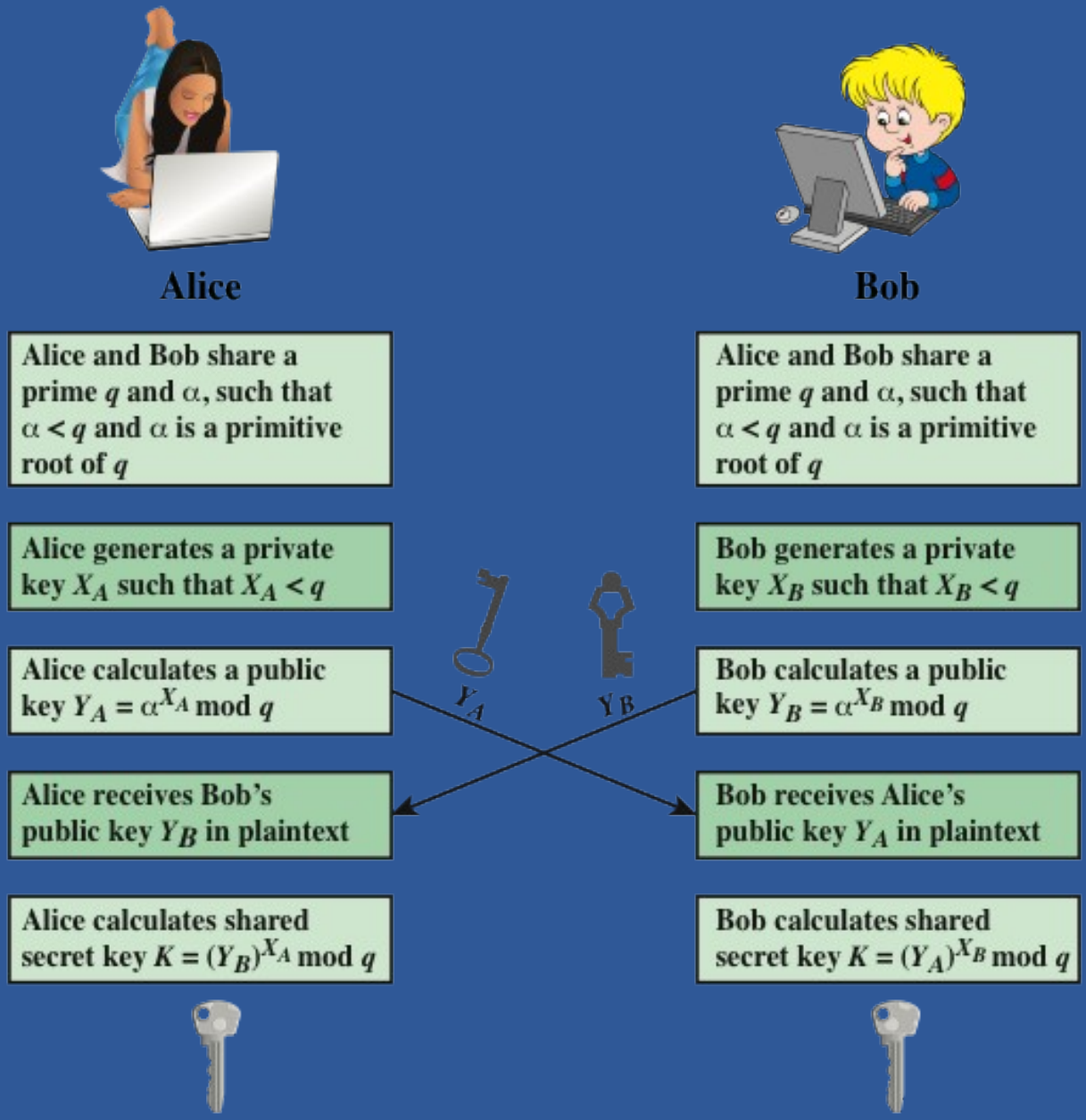


Figure 3.12 Diffie-Hellman Key Exchange

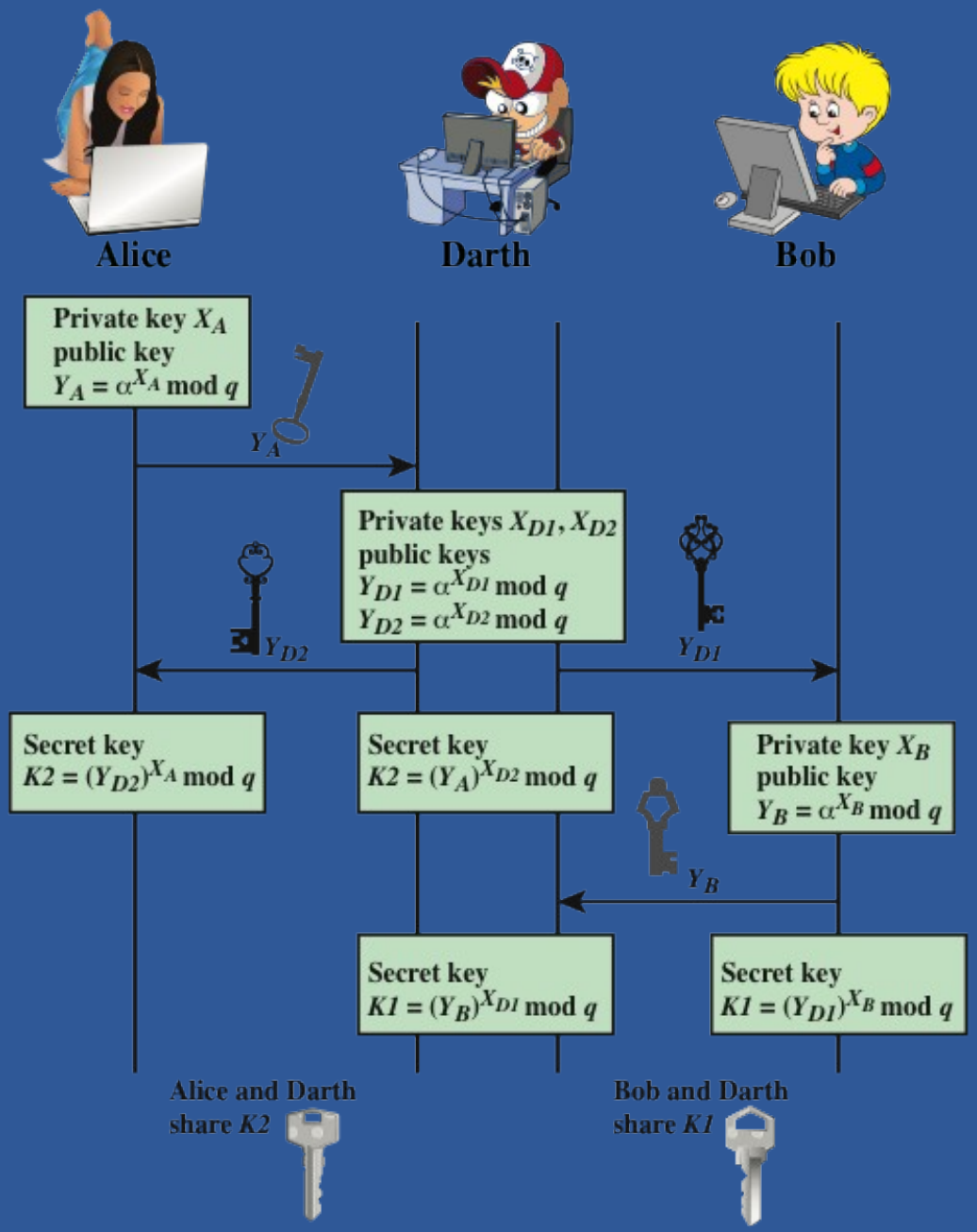


Figure 3.13 Man-in-the-Middle Attack

Digital Signature standard (DSS)

- FIPS PUB 186
- Makes use of the SHA-1 and presents a new digital signature technique, the Digital Signature Algorithm (DSA)
- Originally proposed in 1991 and revised in 1993 and again in 1996
- Uses an algorithm that is designed to provide only the digital signature function
- Unlike RSA, it cannot be used for encryption or key exchange

Elliptic-curve cryptology (ECC)

- Technique is based on the use of a mathematical construct known as the elliptic curve
- Principal attraction of ECC compared to RSA is that it appears to offer equal security for a far smaller bit size, thereby reducing processing overhead
- The confidence level in ECC is not yet as high as that in RSA

Summary

- Approaches to message authentication
  - Authentication using conventional encryption
  - Message authentication without message encryption
- Secure hash functions
  - Hash function requirements
  - Security of hash functions
  - Simple hash functions
  - The SHA secure hash function SHA-3
- Digital signatures
- Message authentication codes
  - HMAC
  - MACs based on block ciphers
- Public-key cryptography principles
  - Public-key encryption structure
  - Applications for public-key cryptosystems
  - Requirements for public-key cryptography
- Public-key cryptography algorithms
  - The RSA public-key encryption algorithm
  - Diffie-Hellman key exchange
  - Other public-key cryptography algorithms