Modern Cryptography and Its Applications

Yanwei Yu

E-mail: ywyu@ustc.edu.cn

Grading Scheme

- Labs 50%
  - 5 lab

参与八强淘汰！

- Midterm Test / Paper: 10%

- Final Exam: 40%, half-open

1 Introduction

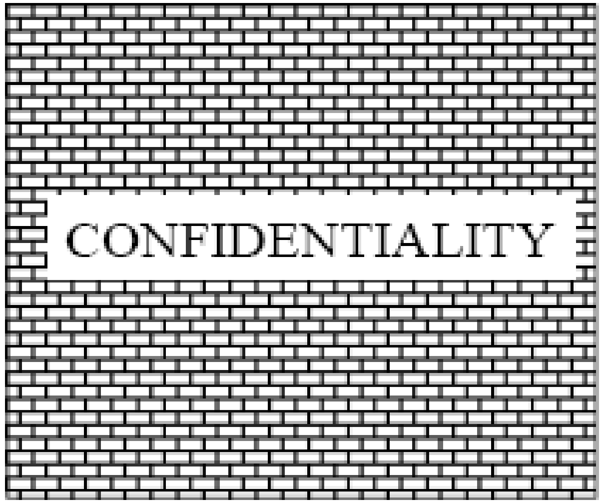
- **OSI Security Architecture focuses**
  - **security attacks**
  - **Services**
  - **mechanisms**

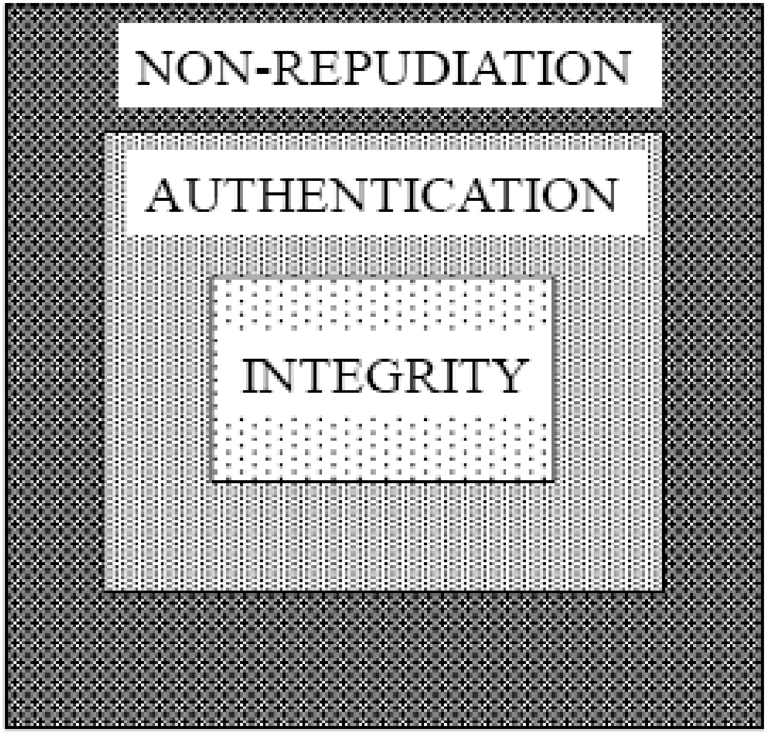
Table 1.4 Relationship Between Security Services and Mechanisms

SERVICE \ MECHANISM	Encipherment	Digital signature	Access control	Data integrity	Authentication exchange	Traffic padding	Routing control	Notarization
Peer entity authentication	Y	Y			Y			
Data origin authentication	Y	Y						
Access control			Y					
Confidentiality	Y						Y	
Traffic flow confidentiality	Y					Y	Y	
Data integrity	Y	Y		Y				
Nonrepudiation		Y		Y				Y
Availability				Y	Y			

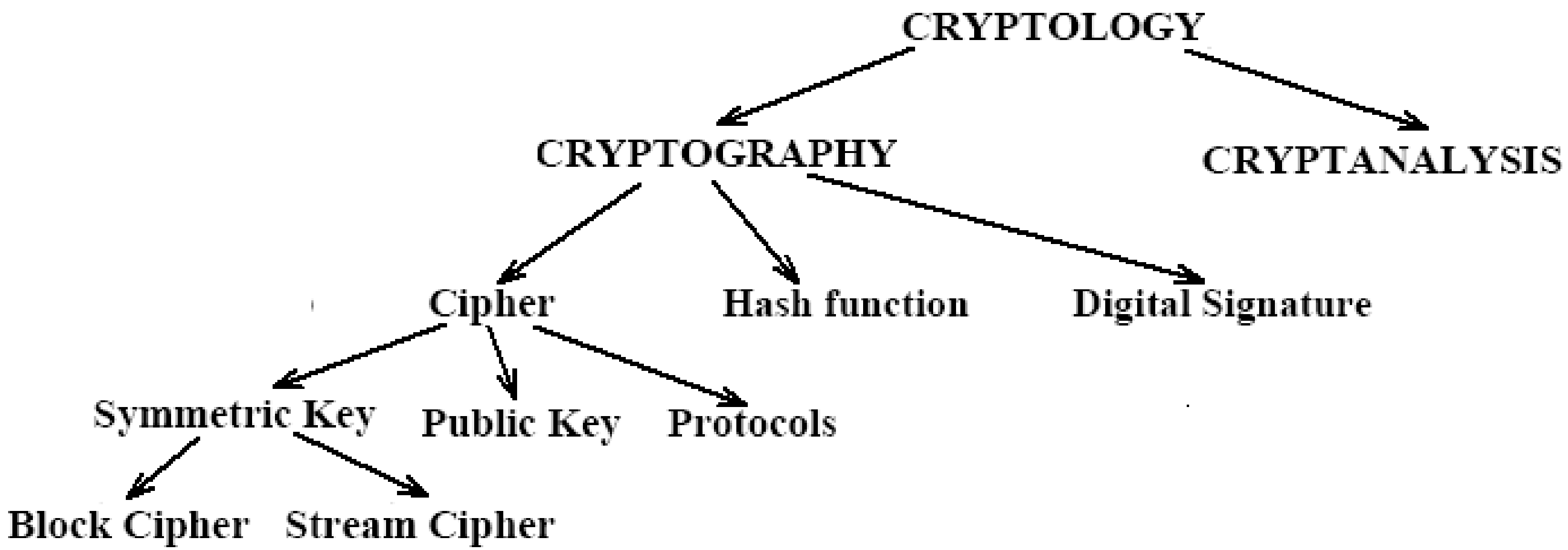
no single <u>mechanism</u> that will support all <u>services</u> required

Relations among security services



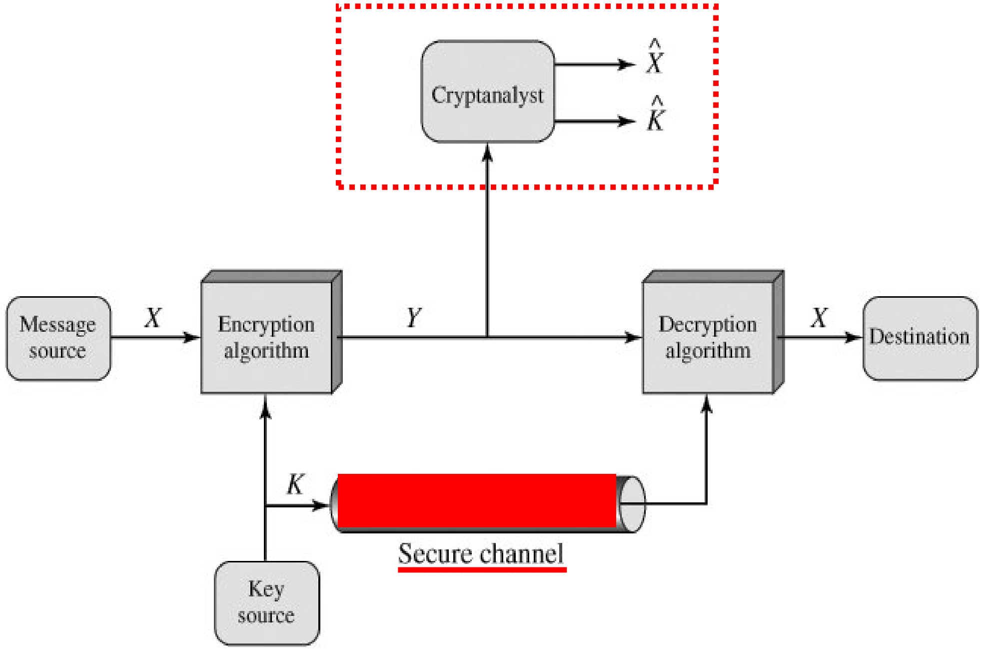


- **Basic security mechanism**



2 Classical Encryption Techniques

- Basic Terminology
  - Cryptology=Crypto(secret)-log(word)
  - Cryptography=Crypto(secret)-graph(write)
    - symmetric encryption(block / stream), asymmetric encryption
  - Cryptanalysis (codebreaking)
    - cryptanalytic attack: ciphertext only, known plaintext, chosen plaintext
    - brute-force(穷举) attack



- **Five Basic Elements**
  - **plaintext, ciphertext, key**
  - **Encipher, decipher**
- **two requirements for secure use of symmetric encryption:**
  - **a strong encryption algorithm**
  - **a secret key known only to sender / receiver**
- ***Kerckhoffs' principle***
  - **assume encryption algorithm is known**
  - **implies a secure channel to distribute key**

- unconditional security
  - **no matter how much computer power or time is available, the cipher cannot be broken since the ciphertext provides insufficient information to uniquely determine the corresponding plaintext**
- computational security
  - **given limited computing resources, the cipher cannot be broken**
    - **cost needed for calculations exceeds ciphertext value**
    - **time needed for calculations exceeds valid lifetime of ciphertext**

- **Classical Cryptology：**

  Transposition / Permutation -> Substitution

  – **No key -> have key**

  – **Substitution:**

    - **Mon-alphabetic Substitution(单表替换)**
      – **Caesar Cipher:** $c_i = f(m_i) = m_i + 3 \bmod 26$, NO KEY!!
      – **Shift Cipher：** $c_i = f(m_i) = m_i + k \bmod 26$, 26 Keys
      – **general Monalphabetic substitution ciphers:** 26! Keys
    - **Poly-alphabetic Substitution(多表替换): Vigenere cipher**

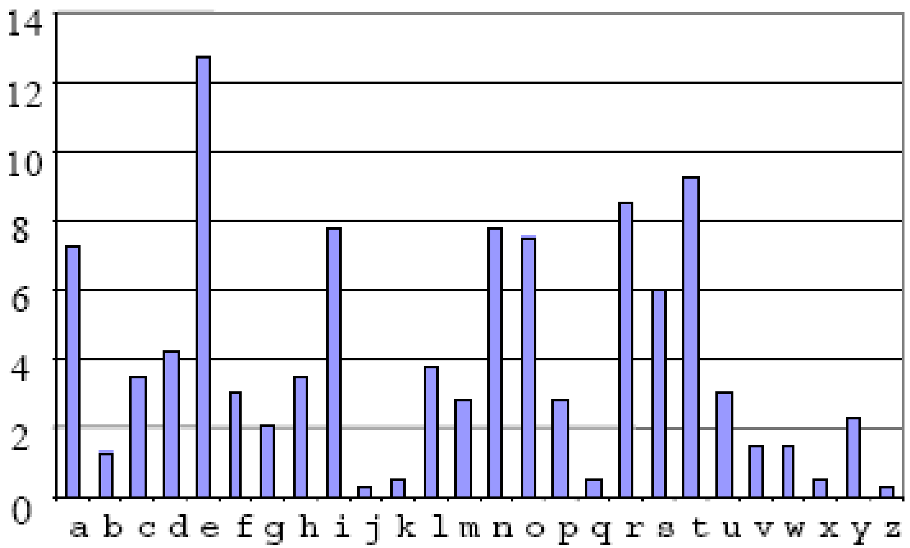
$$c_i = f_{i \bmod d}(m_i) = m_i + k_{i \bmod d} \bmod 26$$

$$m_i = f^{-1}_{i \bmod d}(m_i) = m_i - k_{i \bmod d} \bmod 26$$

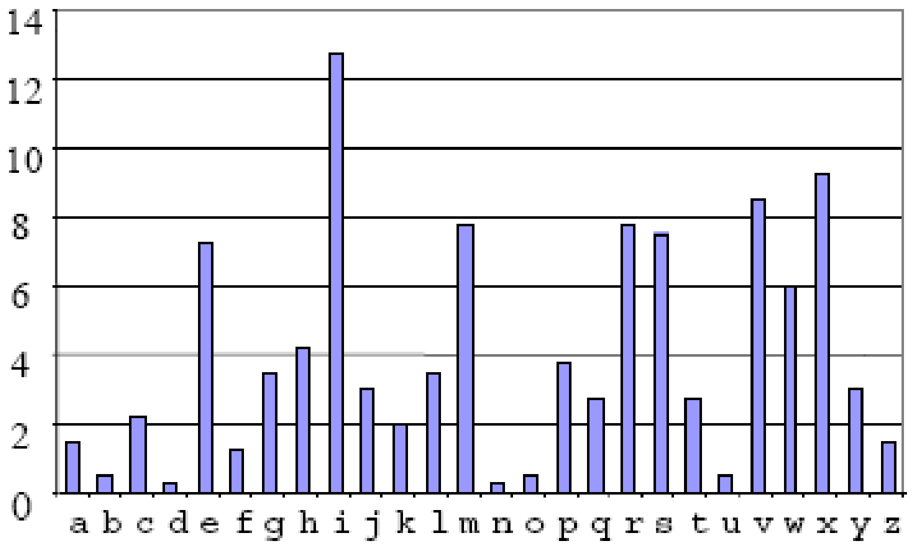
    - **Combined with product: Rotor machines**

$$\text{Key} = k_0, k_1, \ldots, k_{d-1}$$

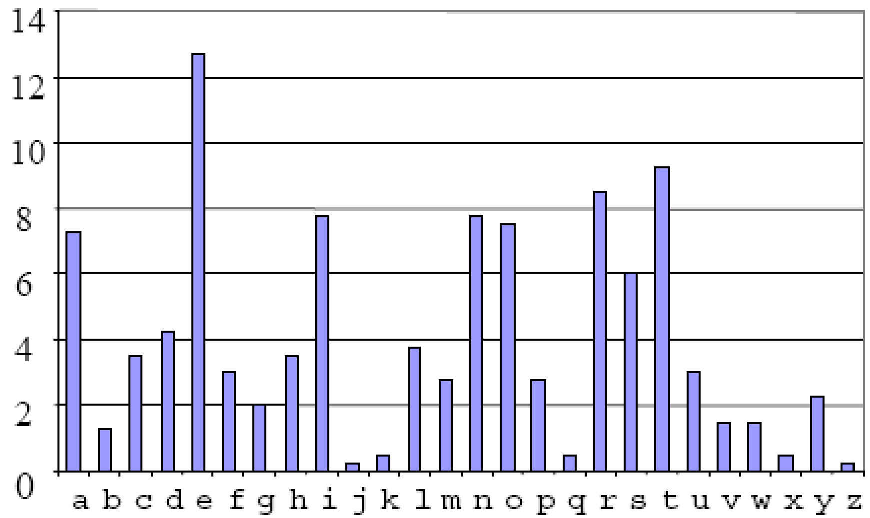
Number of keys for a given period $d = (26)^d$



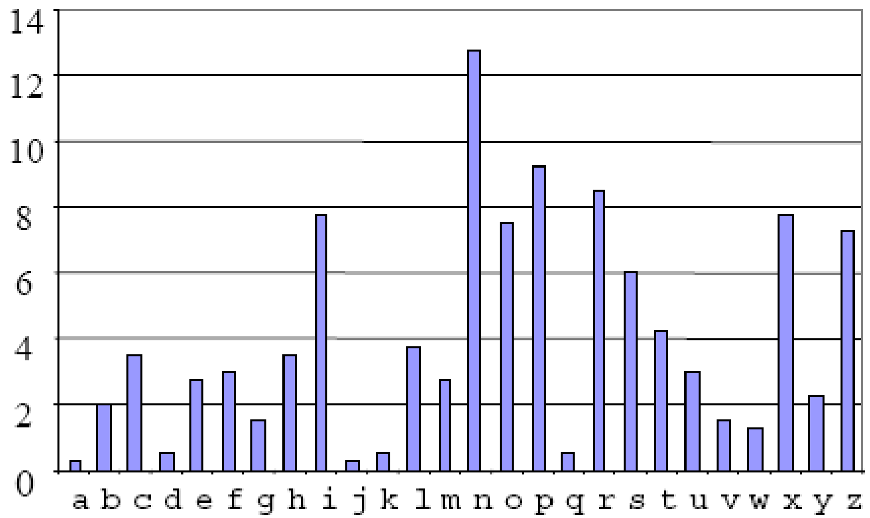
Character frequency in a long English plaintext



Character frequency in the corresponding ciphertext for a shift cipher



Character frequency in a long English plaintext



Character frequency in the corresponding ciphertext for a general monoalphabetic substitution cipher

3 Block Ciphers and the Data Encryption Standard

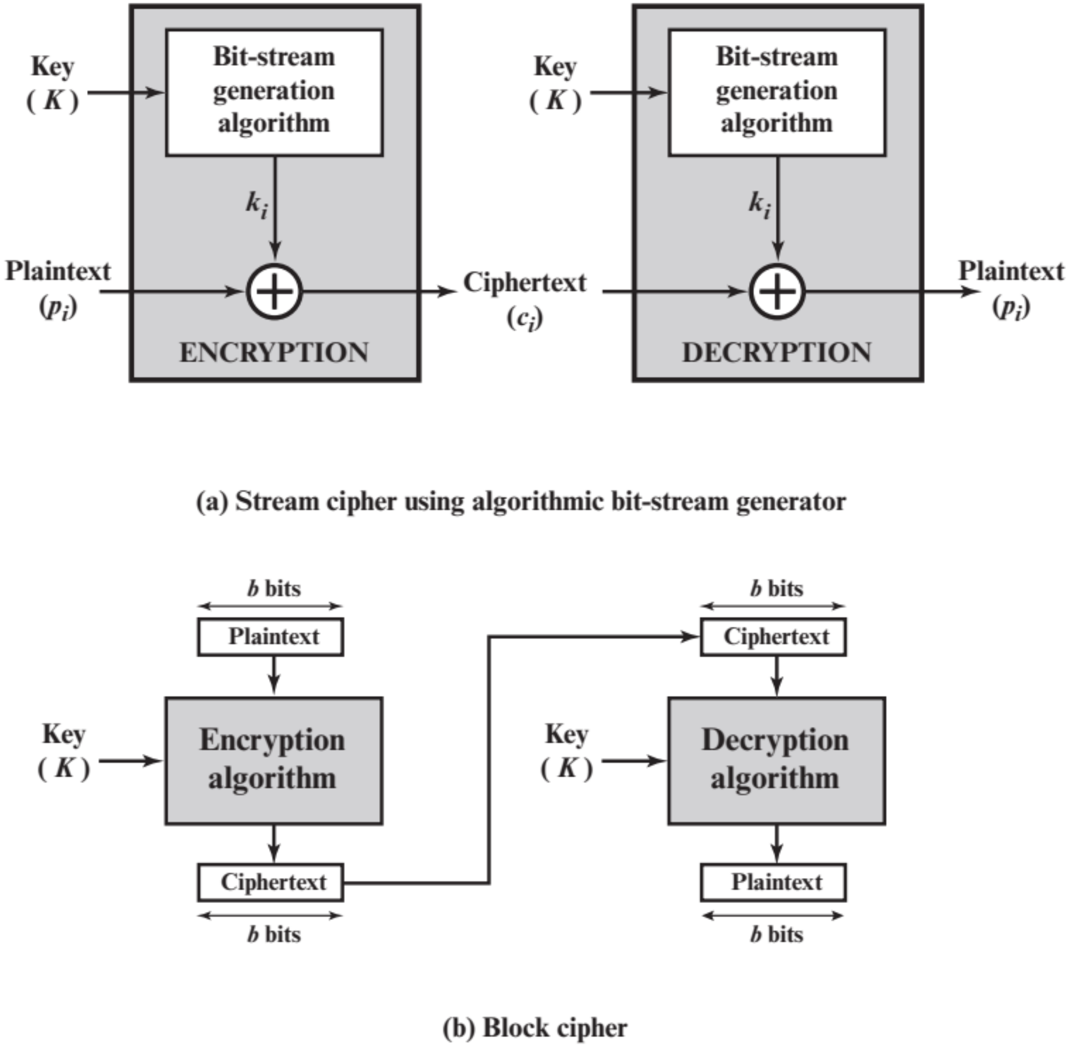


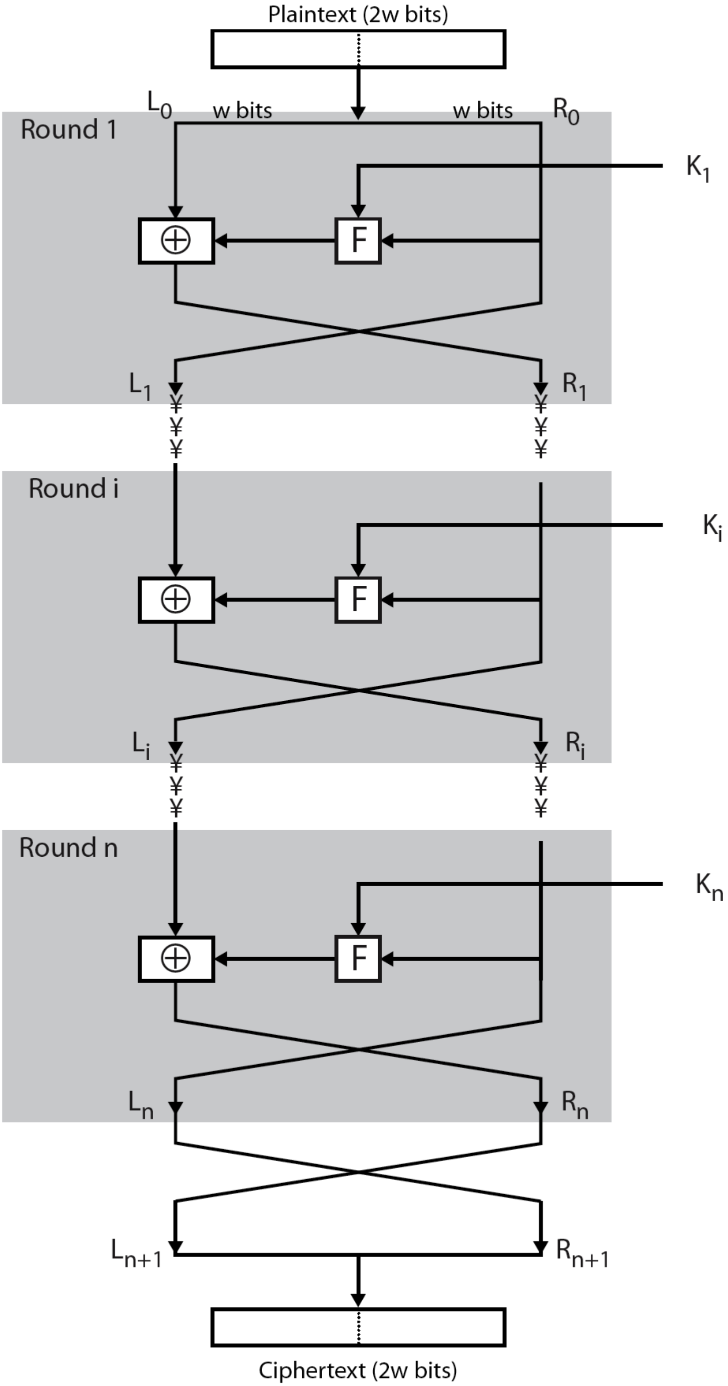
Figure 4.1 Stream Cipher and Block Cipher

- Computationally secure ciphers based on the idea of confusion and diffusion
- Diffusion(扩散) – spreading influence of one plaintext letter to many ciphertext letters
  - E.g. through the use of permutations and linear substitutions
- Confusion(混淆) – makes relationship between ciphertext and key as complex as possible
  - E.g. through the use of non-linear substitutions

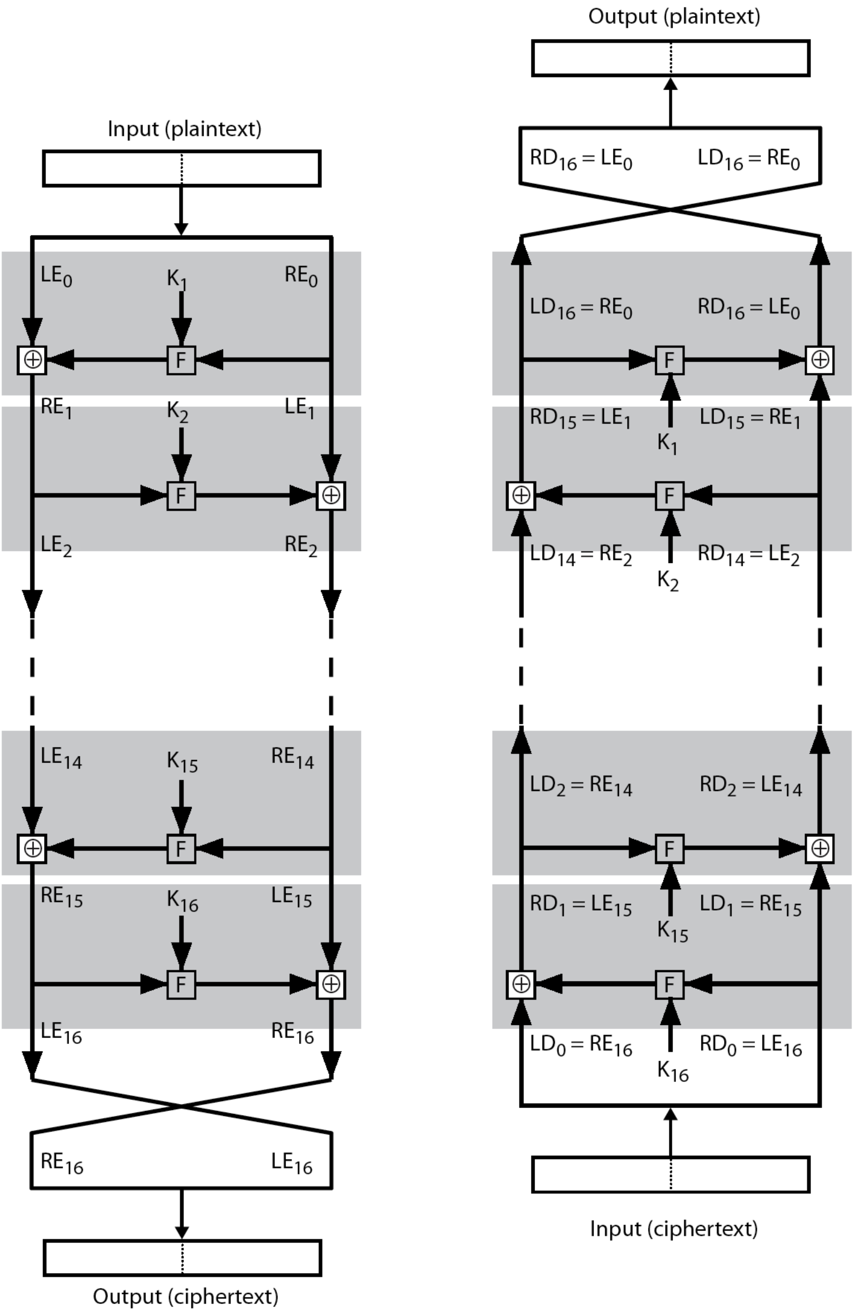
- **Feistel cipher(对合&不可逆)**
  - **process through multiple rounds**
  - **For each round,**
    - **partitions input block into two halves**
    - **perform a substitution on left data half based on the round function of last right half & subkey**
    - **then have permutation swapping halves**

$$L_i = R_{i-1}$$
$$R_i = L_{i-1} \oplus F(R_{i-1}, K_i)$$

- **Feistel Cipher Design Elements**
  - **block size, key size, number of rounds**
  - **subkey generation algorithm**
  - **round function (no invertible requirements)**
  - **fast software en/decryption**
  - **ease of analysis**

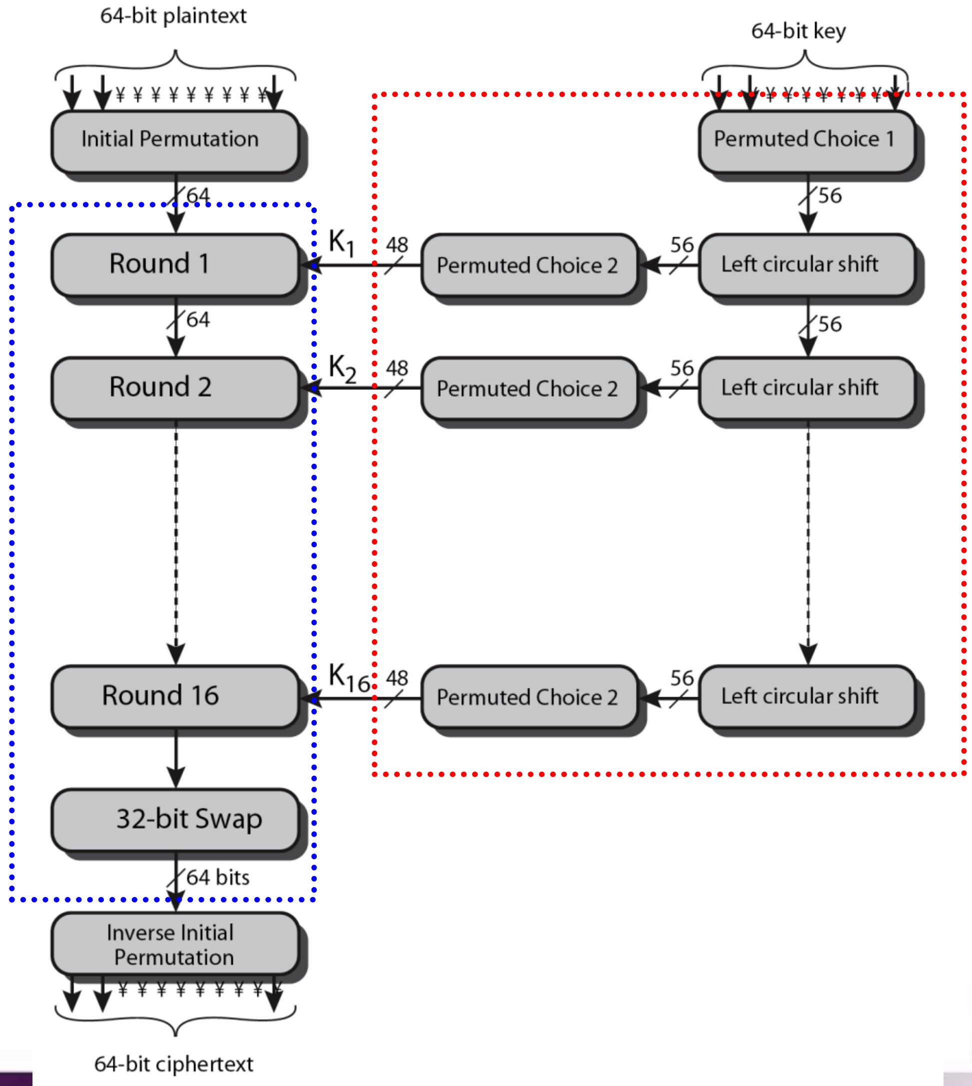


- **Feistel Cipher Encryption & Decryption**
- 对合 & 不可逆



DES Encryption Overview

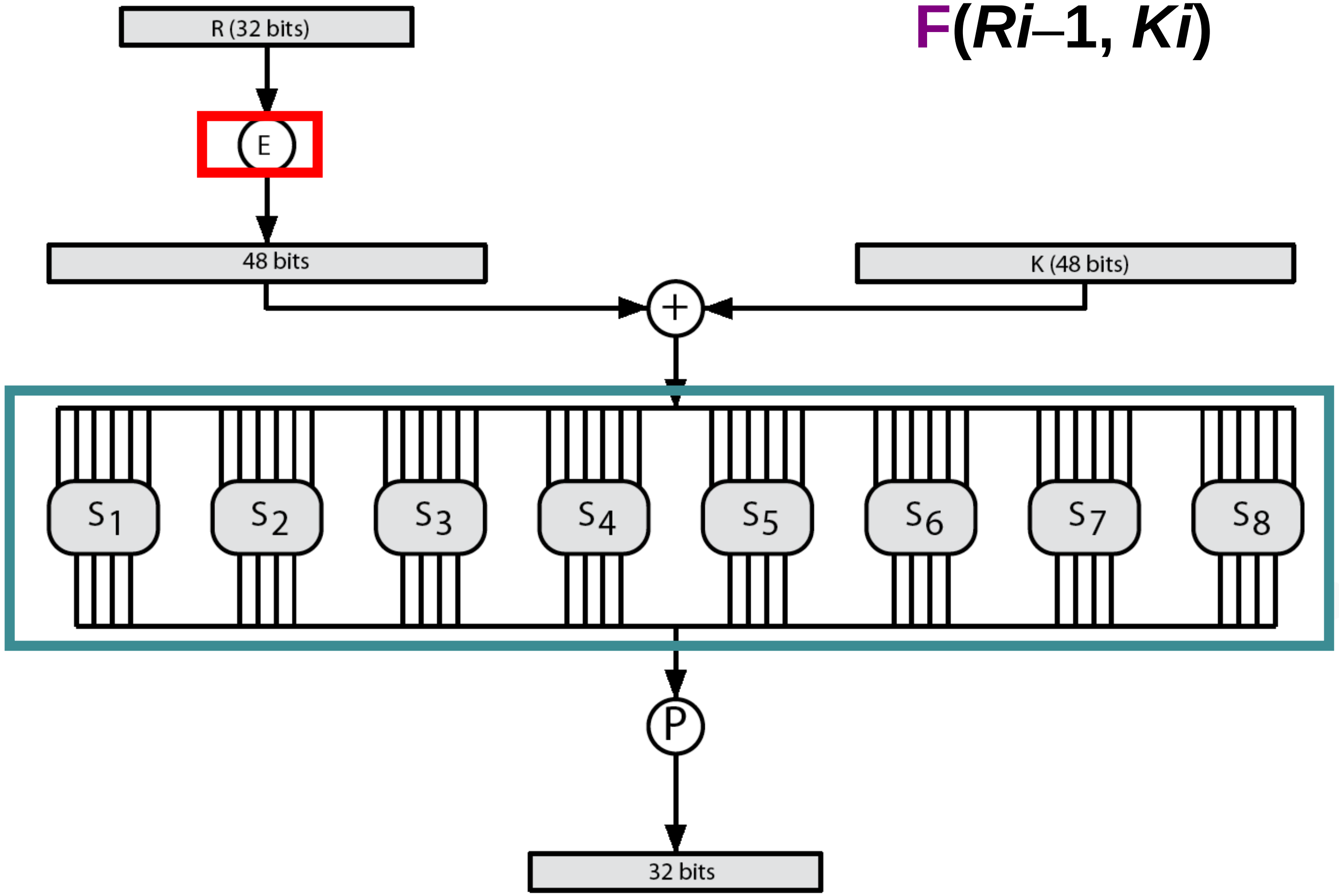
- **明/密文分组长度为64bits，密钥长度为56bits**
- **是Feistel结构**
- exhibits strong **valanche**(雪崩)
- brute-force attack to DES: 平均情况下，$O(2^{55})$



DES Round Structure

- **F takes 32-bit R half and 48-bit subkey:**
  - ✓ expands 32-bit R to 48-bits using perm E
  - ✓ adds to subkey using XOR
  - ✓ passes through 8 S-boxes to get 32-bit result
  - ✓ finally permutes using 32-bit perm P

F(*Ri*–1, *Ki*)



4 triple-DES and AES

- **Need a replacement for DES**
  - **potential vulnerability of DES to a brute-force attack: $O(2^{55})$**
  - **$O(2^{54})$ under the chosen plaintexts**

    (if $C=E_K(P)$, then $\overline{C}=E_{\overline{K}}(\overline{P})$)
- **Two alternatives**
  - **design a completely new algorithm: AES**
  - **use multiple encryption with DES and multiple keys to preserve the existing investment in software and equipment: Triple-DES**

- **Double-DES? 112 bits key**

  "meet-in-the-middle" attack: $O(2^{56})$

  - Assume pair (P,C), have $C=E_{K2}E_{K1}(P)$
    - $X = E_{K1}(P) = D_{K2}(C)$
    - attack by encrypting P with all keys and store
    - then decrypt C with keys and match X value
  - two blocks of known plaintext-ciphertext will succeed against double DES
- **Triple-DES**
  - use three stages of encryption
  - cost of the known-plaintext "meet-in-the-middle" attack: $O(2^{112})$
  - Two forms:
    - use two keys with E-D-E sequence
    - use Three Keys with E-D-E sequence

Groups, Rings, and Fields

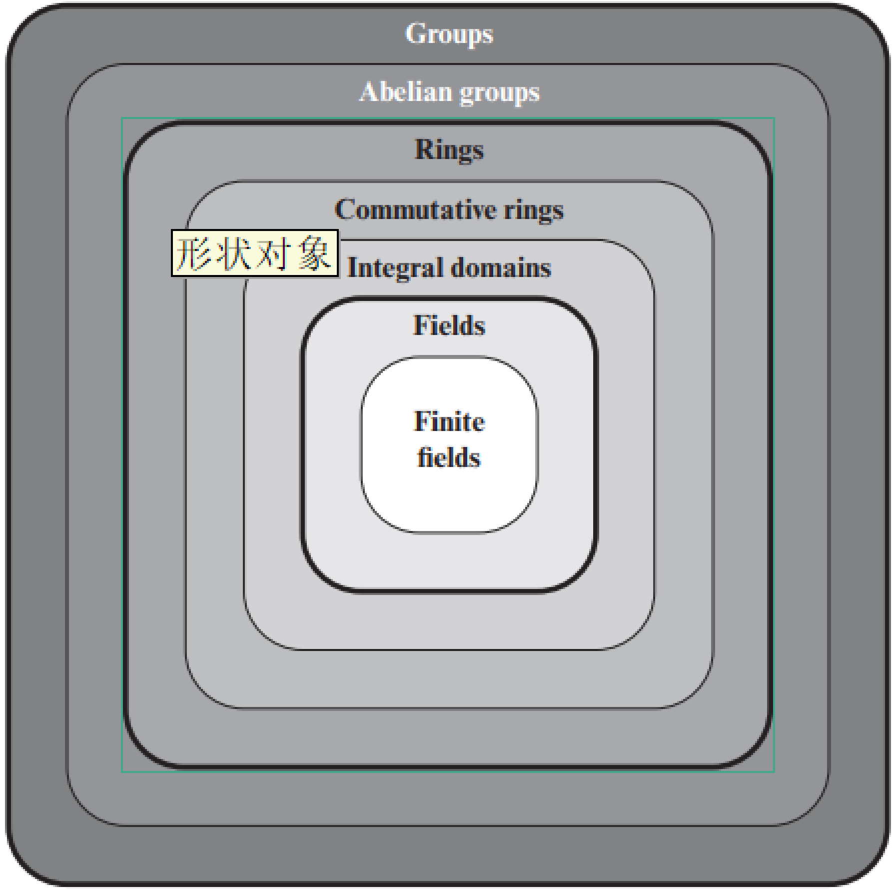
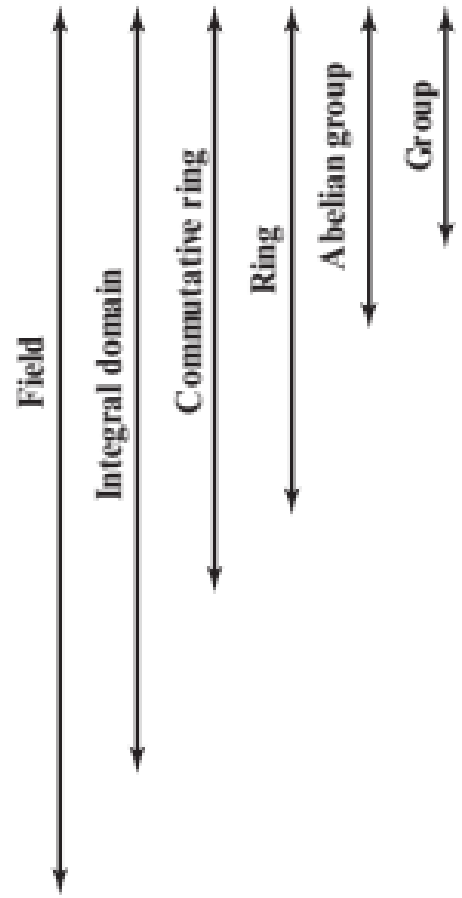


Figure 5.1 Groups, Rings, and Fields

Groups, Rings, and Fields



(A1) Closure under addition: If a and b belong to S, then $a + b$ is also in S
(A2) Associativity of addition: $a + (b + c) = (a + b) + c$ for all a, b, c in S
(A3) Additive identity: There is an element 0 in R such that $a + 0 = 0 + a = a$ for all a in S
(A4) Additive inverse: For each a in S there is an element $-a$ in S such that $a + (-a) = (-a) + a = 0$

(A5) Commutativity of addition: $a + b = b + a$ for all a, b in S

(M1) Closure under multiplication: If a and b belong to S, then ab is also in S
(M2) Associativity of multiplication: $a(bc) = (ab)c$ for all a, b, c in S
(M3) Distributive laws: $a(b + c) = ab + ac$ for all a, b, c in S; $(a + b)c = ac + bc$ for all a, b, c in S

(M4) Commutativity of multiplication: $ab = ba$ for all a, b in S

(M5) Multiplicative identity: There is an element 1 in S such that $a1 = 1a = a$ for all a in S
(M6) No zero divisors: If a, b in S and $ab = 0$, then either $a = 0$ or $b = 0$

(M7) Multiplicative inverse: If a belongs to S and $a \neq 0$, there is an element a^{-1} in S such that $aa^{-1} = a^{-1}a = 1$

Figure 5.2 Properties of Groups, Rings, and Fields

Galois Fields

- finite fields play a key role in cryptography
- in particular often use the fields:
  - GF(p):
    - p is a prime
    - number of elements in a finite field is P
  - GF(2^n)
    - number of elements in a finite field must be a power of a prime, p^n
    - known as Galois fields, denoted GF(p^n)

Finite Fields of the form GF(p)

- **GF(7):** using modular arithmetic modulo 7

+	0	1	2	3	4	5	6
0	0	1	2	3	4	5	6
1	1	2	3	4	5	6	0
2	2	3	4	5	6	0	1
3	3	4	5	6	0	1	2
4	4	5	6	0	1	2	3
5	5	6	0	1	2	3	4
6	6	0	1	2	3	4	5

(d) Addition modulo 7

×	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6
2	0	2	4	6	1	3	5
3	0	3	6	2	5	1	4
4	0	4	1	5	2	6	3
5	0	5	3	1	6	4	2
6	0	6	5	4	3	2	1

(e) Multiplication modulo 7

w	0	1	2	3	4	5	6
$-w$	0	6	5	4	3	2	1
w^{-1}	—	1	4	5	2	3	6

(f) Additive and multiplicative inverses modulo 7

Example GF(2^3)

Table 4.6 Polynomial Arithmetic Modulo ($x^3 + x + 1$)

	+	000 0	001 1	010 x	011 $x+1$	100 x^2	101 x^2+1	110 x^2+x	111 x^2+x+1
000	0	0	1	x	$x+1$	x^2	x^2+1	x^2+x	x^2+x+1
001	1	1	0	$x+1$	x	x^2+1	x^2	x^2+x+1	x^2+x
010	x	x	$x+1$	0	1	x^2+x	x^2+x+1	x^2	x^2+1
011	$x+1$	$x+1$	x	1	0	x^2+x+1	x^2+x	x^2+1	x^2
100	x^2	x^2	x^2+1	x^2+x	x^2+x+1	0	1	x	$x+1$
101	x^2+1	x^2+1	x^2	x^2+x+1	x^2+x	1	0	$x+1$	x
110	x^2+x	x^2+x	x^2+x+1	x^2	x^2+1	x	$x+1$	0	1
111	x^2+x+1	x^2+x+1	x^2+x	x^2+1	x^2	$x+1$	x	1	0

(a) Addition

	×	000 0	001 1	010 x	011 $x+1$	100 x^2	101 x^2+1	110 x^2+x	111 x^2+x+1
000	0	0	0	0	0	0	0	0	0
001	1	0	1	x	$x+1$	x^2	x^2+1	x^2+x	x^2+x+1
010	x	0	x	x^2	x^2+x	$x+1$	1	x^2+x+1	x^2+1
011	$x+1$	0	$x+1$	x^2+x	x^2+1	x^2+x+1	x^2	1	x
100	x^2	0	x^2	$x+1$	x^2+x+1	x^2+x	x	x^2+1	1
101	x^2+1	0	x^2+1	1	x^2	x	x^2+x+1	$x+1$	x^2+x
110	x^2+x	0	x^2+x	x^2+x+1	1	x^2+1	$x+1$	x	x^2
111	x^2+x+1	0	x^2+x+1	x^2+1	x	1	x^2+x	x^2	$x+1$

(b) Multiplication

choose an irreducible polynomial of degree 3 : ($x^3 + x + 1$)

Table 5.1. Arithmetic Modulo 8

+	0	1	2	3	4	5	6	7
1	1	2	3	4	5	6	7	0
2	2	3	4	5	6	7	0	1
3	3	4	5	6	7	0	1	2
4	4	5	6	7	0	1	2	3
5	5	6	7	0	1	2	3	4
6	6	7	0	1	2	3	4	5
7	7	0	1	2	3	4	5	6

(a) Addition modulo 8

×	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7
2	0	2	4	6	0	2	4	6
3	0	3	6	1	4	7	2	5
4	0	4	0	4	0	4	0	4
5	0	5	2	7	4	1	6	3
6	0	6	4	2	0	6	4	2
7	0	7	6	5	4	3	2	1

(b) Multiplication modulo 8

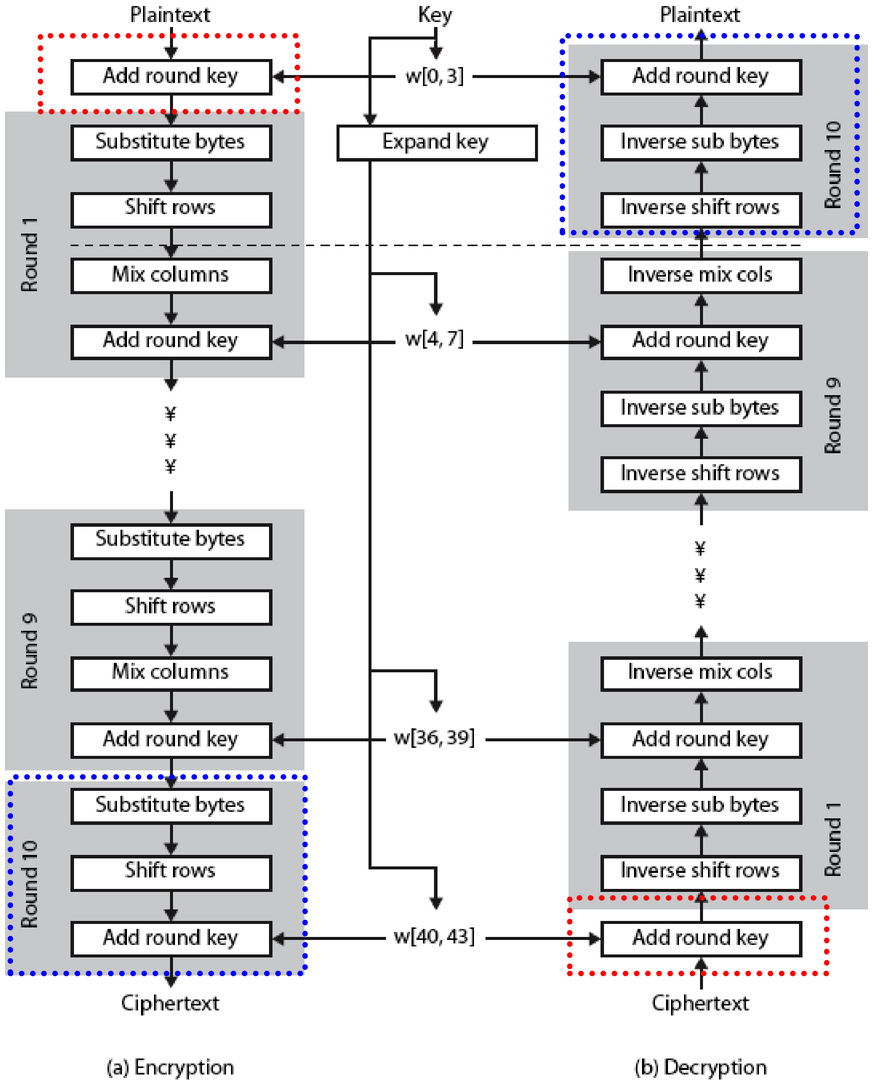
w	$-w$	w^{-1}
0	0	—
1	7	1
2	6	—
3	5	3
4	4	—
5	3	5
6	2	—
7	1	7

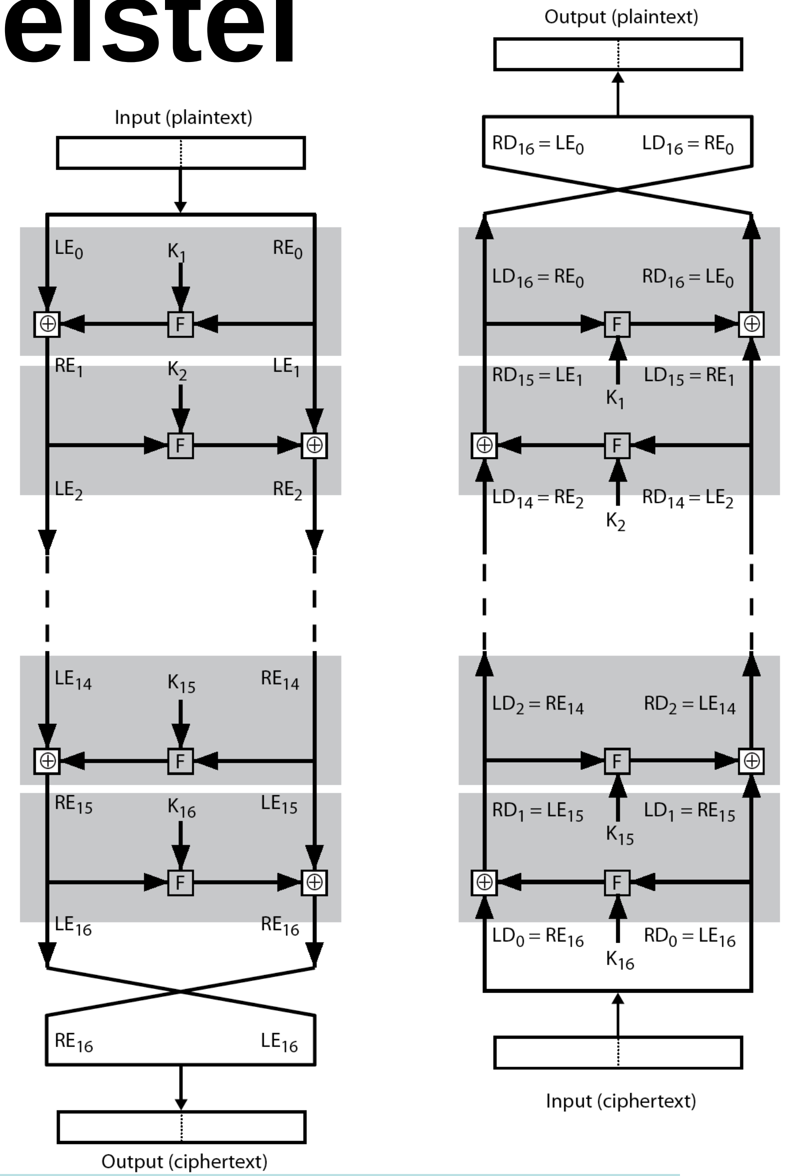
(c) Additive and multiplicative inverses modulo 8

Summary: Operations of GF(2^n)

- Any element of GF(2^n) is represented as a polynomial
  - E.g. 1100: $x^3 + x^2$
- +(addition)
  - polynomial notation: adding corresponding coefficients based on modulo 2
  - binary notation: a bitwise XOR operation.
- ·(multiplication)
  - polynomial notation: perform the ordinary rules of polynomial arithmetic and Arithmetic on the coefficients is performed modulo 2, then modulo some irreducible polynomial m(x) of degree
  - binary notation: multiplication is shift & XOR

Structure of AES vs. Feistel

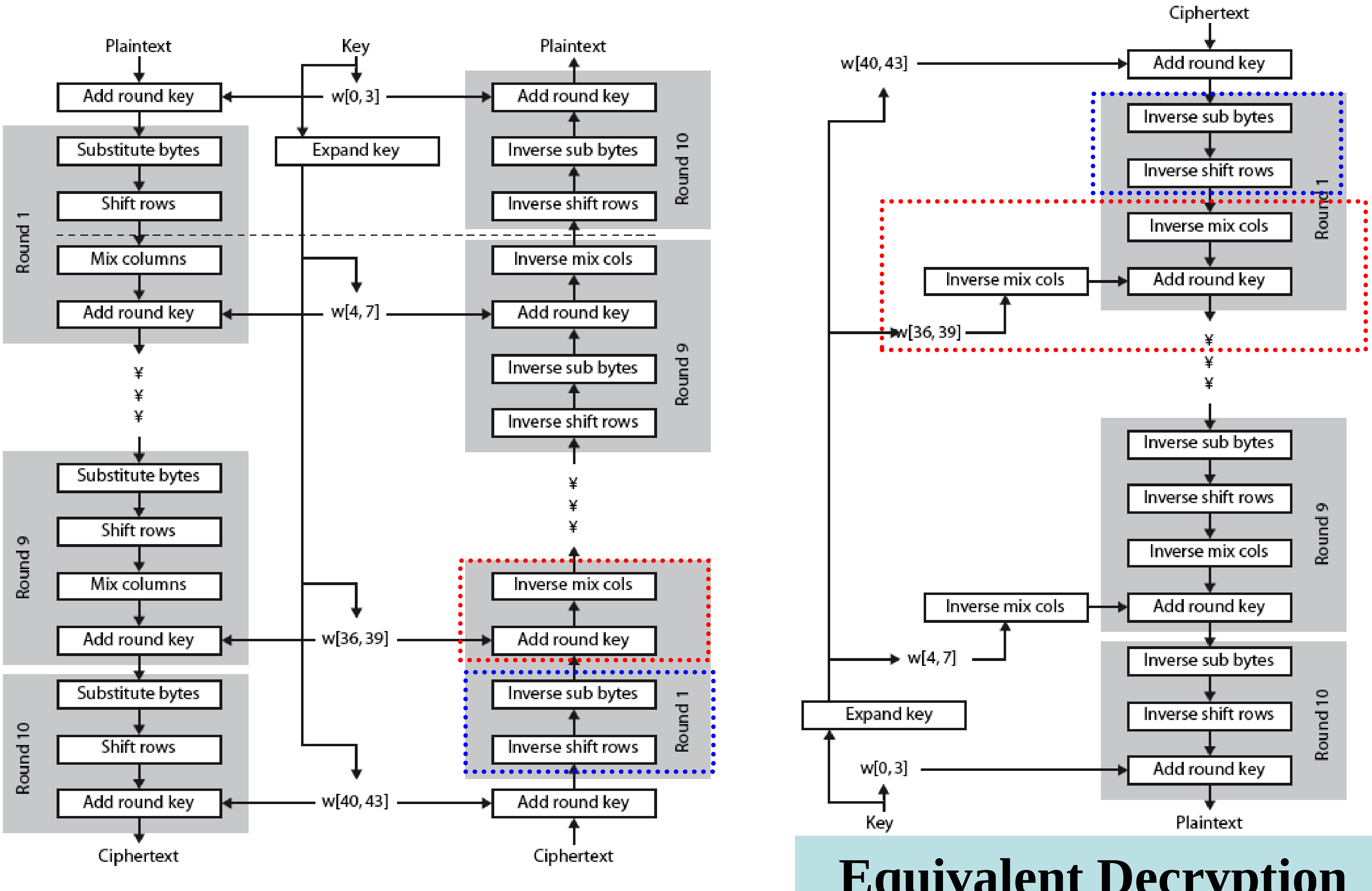




AES is an iterative rather than feistel cipher

AE

$$\text{InvMixColumns}\ (S_i \oplus w_j) = [\text{InvMixColumns}\ (S_i)] \oplus [\text{InvMixColumns}\ (w_j)]$$



(a) Encryption

(b) Decryption

Equivalent Decryption

5 Modes of Operation and RC4

- modes of operation: some way to en/decrypt arbitrary amounts of data in practise
  - Block modes: may need padding the last block
    - Electronic Codebook Book (ECB)
    - Cipher Block Chaining (CBC)
  - Stream modes: Encryption function also used for decryption
    - Cipher Feedback (CFB)
    - Output Feedback (OFB)
    - Counter (CTR)

- **Compare five modes of operation**
  - **How to work in encryption and decryption**
  - **Properties(优?劣?)**
    - **Same plaintext blocks (under the same key) result in same ciphertext blocks?**
    - **Error propagation：**会影响到后续多少个分组的解密？
    - **Self-synchronizing：**能否？
    - **High speed?**
  - **Traditional application**

Mode	Description	Typical Application
Electronic Codebook (ECB)	Each block of 64 plaintext bits is encoded independently using the same key.	• Secure transmission of single values (e.g., an encryption key)
Cipher Block Chaining (CBC)	The input to the encryption algorithm is the XOR of the next 64 bits of plaintext and the preceding 64 bits of ciphertext.	• General-purpose block-oriented transmission • Authentication
Cipher Feedback (CFB)	Input is processed *j* bits at a time. Preceding ciphertext is used as input to the encryption algorithm to produce pseudorandom output, which is XORed with plaintext to produce next unit of ciphertext.	• General-purpose stream-oriented transmission • Authentication
Output Feedback (OFB)	Similar to CFB, except that the input to the encryption algorithm is the preceding DES output.	• Stream-oriented transmission over noisy channel (e.g., satellite communication)
Counter (CTR)	Each block of plaintext is XORed with an encrypted counter. The counter is incremented for each subsequent block.	• General-purpose block-oriented transmission • Useful for high-speed requirements

A simple rule of thumb(常用规则) is, unless you have to use CBC mode, choose CTR instead.

6 Secure Communication
——Confidentiality Using Symmetric Encryption

- Key Hierarchy(层次结构)
  - session(会话) key
    - temporary key
    - used for encryption of data between users
    - for one logical session then discarded
  - master(主) key
    - used to encrypt session keys
    - shared by user & key distribution center for long time

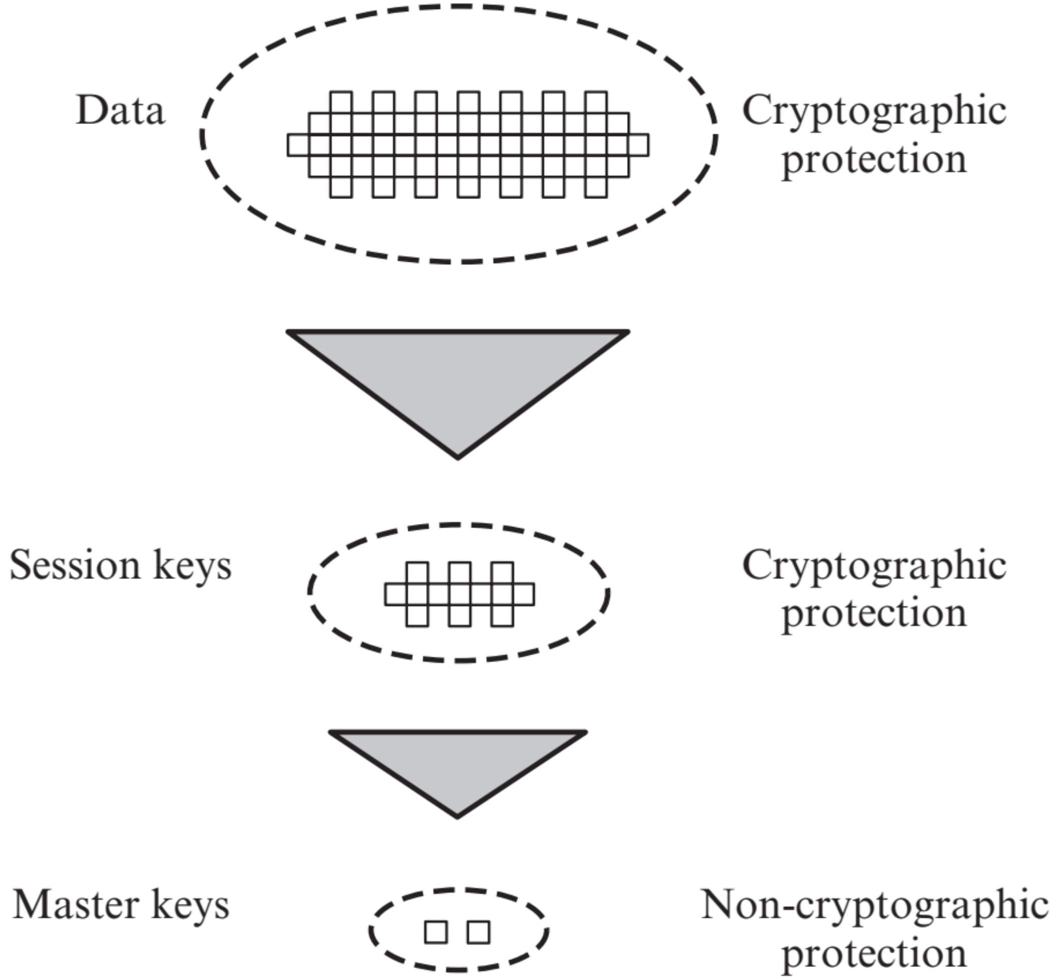


Figure 14.2 The Use of a Key Hierarchy

- **Given parties A and B have various** key distribution **alternatives:**
  1. **A can select key and physically deliver to B**
  2. **third party can select & deliver key to A & B**
  3. **Decentralized Key Control:** if A & B have communicated previously, they can use previous key to encrypt a new key
  4. **Centralized Key Control:** if A & B have secure communications with a third party C, C can relay(传达) key between A & B
     - ✓ **Needham-Schroeder Shared-key Protocol** & improvements
     - ✓ Kerberos
  5. **Use public encryption to protect secret key shared by both**
     - ✓ **Simple Secret Key Distribution: Man-in-the-middle Attack**
     - ✓ **Secret Key Distribution with Confidentiality and Authentication**
  6. **Key Pre-distribution Schemes(no key required)**
     - ✓ **Anonymous Diffie-Hellman: (base Diffie-Hellman algorithm) -> STS**
     - ✓ **Shamir's no-key protocol**

Use master key (items 3–4)

- session key
  - temporary key
  - used for encryption of data between users
  - for one logical session then discarded
  - By symmetric or public encryption
- master key
  - used to encrypt session keys
  - shared by user & key distribution center for long time
  - Generated by Key Pre-distribution, public encryption, password

7 Public-Key Cryptography

- **Evolution of Cryptography**
  - **Before 1976, all cryptographic systems have been based on the elementary tools of substitution and permutation**
    - **by hand -> rotor encryption/decryption machine -> With the availability of computers: DES, AES……**
  - **1976, the concept of public-key cryptography is developed by Diffie and Hellman.**
    - **public-key algorithms are based on mathematical functions rather than on substitution and permutation**
      - **RSA: factoring large numbers**
      - **ElGamal: discrete logarithms problem**
    - **developed to address two key issues:**
      - **key distribution – how to have secure communications in general without having to trust a KDC with your key**
      - **digital signatures – how to verify a message comes intact(完整的) from the claimed sender**

- **public-key/two-key/asymmetric cryptography involves the use of two keys:**
  - **a <u>public-key</u>, which may be known by anybody, and can be used to encrypt messages, and verify signatures**
  - **a <u>private-key</u>, known only to the recipient, used to decrypt messages, and sign (create) signatures**
- **Public-Key Applications:**
  - **encryption/decryption (provide secrecy)**
  - **digital signatures (provide authentication)**
  - **key exchange (of session keys)**

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

- **Requirements for Public-Key Cryptography**

①computationally easy to generate a pair (Pu_b and PR_b).

②computationally easy to compute cipher-text for a sender A knowing the public key and the plain-text M : $C = E(PU_b, M)$

③computationally easy to recover the original message for the receiver B knowing cipher-text and private key : $M = D(PR_b, C)$

④computationally infeasible for an adversary, knowing the public key PU_b, to determine <u>private key PR_b</u>.

⑤computationally infeasible for an adversary, knowing the public key Pu_b and a cipher-text C, to recover <u>recover M</u>.

⑥$M = D[PU_b, E(PR_b, M)] = D[PR_b, E(PU_b, M)]$ (not necessary)

- **trap-door one-way function**

$$Y = \mathrm{f}_k(X) \qquad \text{easy, if } k \text{ and } X \text{ are known}$$
$$X = \mathrm{f}_k^{-1}(Y) \qquad \text{easy, if } k \text{ and } Y \text{ are known}$$
$$X = \mathrm{f}_k^{-1}(Y) \qquad \text{infeasible, if } Y \text{ is known but } k \text{ is not known}$$

Concepts from number theory

- <u>Prime number:</u> (c.f. Section 2.4 in textbook)
  - is an integer that can only be divided without remainder by positive and negative values of itself and 1.
- <u>Greatest Common Divisor</u>:(c.f. Section 2.2 in textbook)
  - gcd[a(x), b(x)] is the polynomial of maximum degree that divides both a(x) and b(x).
- <u>Euler's totient function(欧拉函数)</u> ø(n) (c.f. Section 2.5 in textbook)
  - ø(n)<u>defined as</u> the number of positive integers less than n and relatively prime to n
- <u>Euler's Theorem(欧拉定理）</u>(c.f. Section 2.5 in textbook)
  - $a^{ø(n)}$mod n = 1 where gcd(a,n)=1.
- <u>Fermat's Theorem(费马定理)</u>(c.f. Section 2.5 in textbook)
  - a^{p-1} mod p=1, where p is prime and a is a positive integer not divisible by p.
- <u>Chinese remainder theorem(CRT)(中国剩余定理)</u>(c.f. Section 2.7)
  - provides a way to manipulate (potentially very large) numbers mod M in terms of tuples of smaller numbers.
- <u>extended Euclid's algorithm</u>
  - Euclidean algorithm: gcd(a, b)=gcd(b, a mod b) (if a>b)

- **Modular arithmetic exhibits the following properties: (c.f. Section 2.3 in textbook)**
  - **[(a mod n) + (b mod n)] mod n = (a + b) mod n**
  - **[(a mod n)-(b mod n)] mod n = (a - b) mod n**
  - **[(a mod n) x (b mod n)] mod n = (a x b) mod n**
- **E.g.**

11 mod 8 = 3; 15 mod 8 = 7

[(11 mod 8) + (15 mod 8)] mod 8 = 10 mod 8 = 2
(11 + 15) mod 8 = 26 mod 8 = 2

[(11 *mod* 8) (15 *mod* 8)] *mod* 8 = 4 *mod* 8 = 4
(11 15) *mod* 8 = 4 *mod* 8 = 4

[(11 mod 8) x (15 mod 8)] mod 8 = 21 mod 8 = 5
(11 x 15) mod 8 = 165 mod 8 = 5

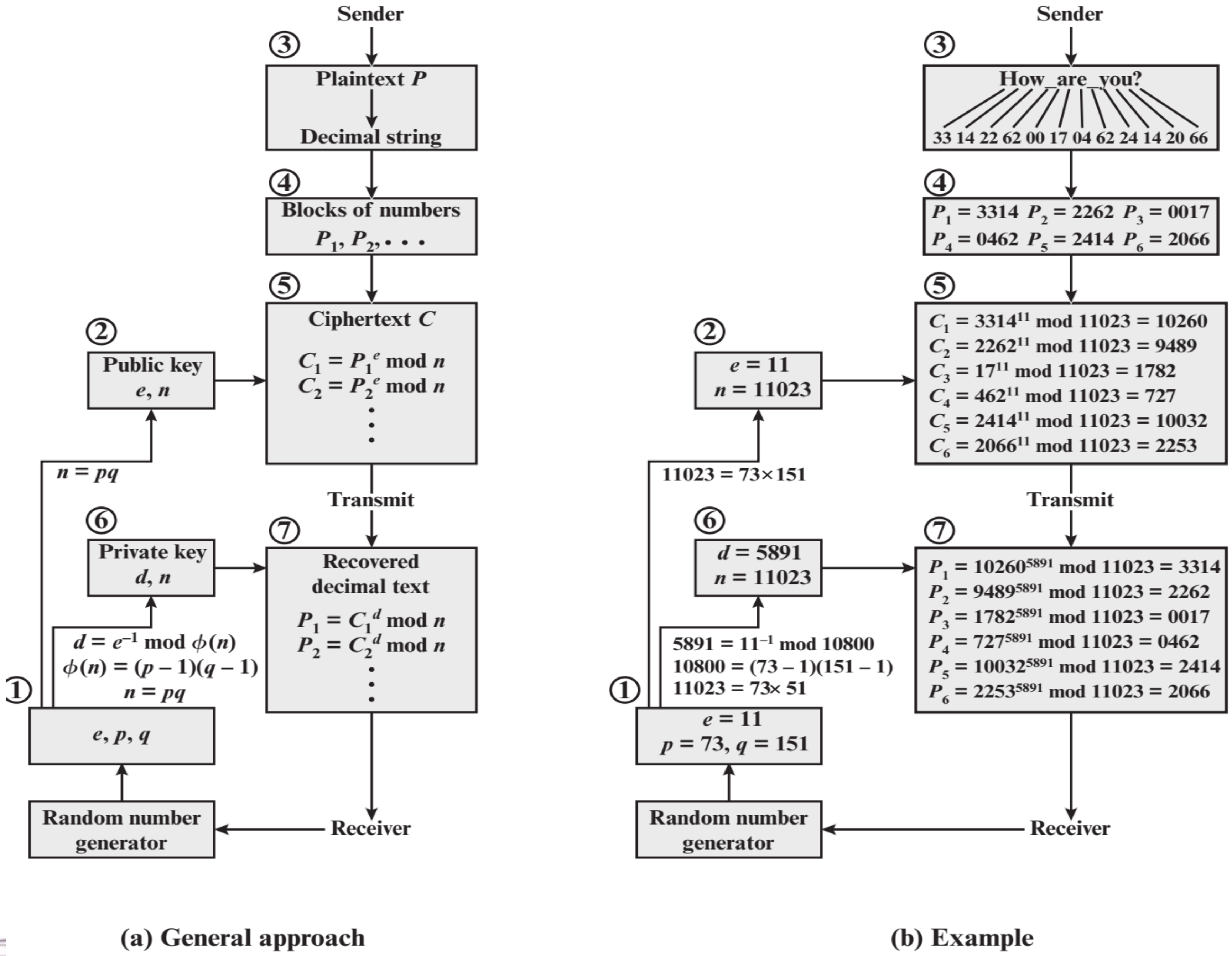


Figure 9.7 RSA Processing of Multiple Blocks

■Computational Aspects of RSA

- Exponentiation in Modular Arithmetic
  - can use the Square and Multiply Algorithm
- Efficient Operation Using Public Key e
  - ensure gcd(e,ø(n))=1
  - If e has small number of 1 bits, encryption will be faster
    - e=3, e=0x10001
  - If e=3, can attack using Chinese remainder theorem
    - use the Chinese Remainder Theorem (CRT)
- Efficient Operation Using Private Key d
  - d may have many 1 bits
  - use the CRT
- Key Generation： use the extended Euclid's algorithm

- $M = C^d \bmod n$
- Using CRT, $M = (V_p X_p + V_q X_q) \bmod n$
  - $V_p = C^d \bmod p$
  - $V_q = C^d \bmod q$
  - $X_p = q \times (q^{-1} \bmod p)$
  - $X_q = p \times (p^{-1} \bmod q)$
- X_p and X_q can be precalculated

- **If gcd(C,p)==1**
  - **Using Fermat's Theorem,$C^{(p-1)}$ mod p=1. For some k1 and K2, d=k1*(p-1)+k2 where d>(p-1), hence**
  - **$Vp = C^d \bmod p = C^{k1*(p-1)+k2} \bmod p = (C^{(p-1)})^{k1} * C^{k2} \bmod p$**

    $= 1*C^{k2} \bmod p = C^{d \bmod (p-1)} \bmod p$
  - **Compute Vq in two case**
    - **Case Gcd(C,q)≠1, Vq=0.**
    - **Case Gcd(C,q)==1, $Vq = C^d \bmod q = C^{d \bmod (q-1)} \bmod q$.**
- **Else //Gcd(C,p)≠1**
  - **//Assume C=k*p, then must have gcd(C,q)==1 because p, q are prime and C<n==p*q.**
  - **$Vp = C^d \bmod p = (k*p)^d \bmod p = 0$**
  - **$Vq = C^d \bmod q = C^{d \bmod (q-1)} \bmod q$**

- RSA is vulnerable to a Chosen Ciphertext Attack (CCA)
  - E(PU,M1)*E(PU,M2)=E(PU,M1*M2)
- Countermeasures:
  - counter with random pad of plaintext

    E(PU,P(M1))*E(PU,P(M2))=E(PU,P(M1)*P(M2))
    ≠E(PU,P(M1*M2))
  - Pad M to EM by using Optimal Asymmetric Encryption Padding (OAEP)
  - Then Encrypt EM by RSA algorithm.

- **Distribution of Public Keys:**
  - **public announcement(发布)**
  - **publicly available directory(目录)**
  - **public-key authority(授权)**
  - **public-key certificates(证书)**

Requirements on Public-Key Certificates

- Any participant can read a certificate to determine the name and public key of the certificate's owner.
- Any participant can verify that the certificate originated from the certificate authority and is not counterfeit.
- Only the certificate authority can create and update certificates.

8 Hash Functions

- Defition
- Hash value vs. MAC

Requirements for Hash Functions *P349*

1. can be applied to any sized message `M`
2. produces fixed-length output `h`
3. is easy to compute `h=H(M)` for any message `M`
4. <u>One-way property</u>: given `h` is infeasible to find `x` s.t.(满足) `H(x)=h`
5. <u>Weak collision resistance</u>: given `x` is infeasible to find `y` s.t. `H(y)=H(x)`
6. <u>Strong collision resistance</u>: is infeasible to find any `x,y` s.t. `H(y)=H(x)`

Basic Requirements

Table 11.1 Requirements for a Cryptographic Hash Function H

Requirement	Description
Variable input size	H can be applied to a block of data of any size.
Fixed output size	H produces a fixed-length output.
Efficiency	$H(x)$ is relatively easy to compute for any given x, making both hardware and software implementations practical.
Preimage resistant (one-way property)	For any given hash value h, it is computationally infeasible to find y such that $H(y) = h$.
Second preimage resistant (weak collision resistant)	For any given block x, it is computationally infeasible to find $y \neq x$ with $H(y) = H(x)$.
Collision resistant (strong collision resistant)	It is computationally infeasible to find any pair (x, y) with $x \neq y$, such that $H(x) = H(y)$.
Pseudorandomness	Output of H meets standard tests for pseudorandomness.

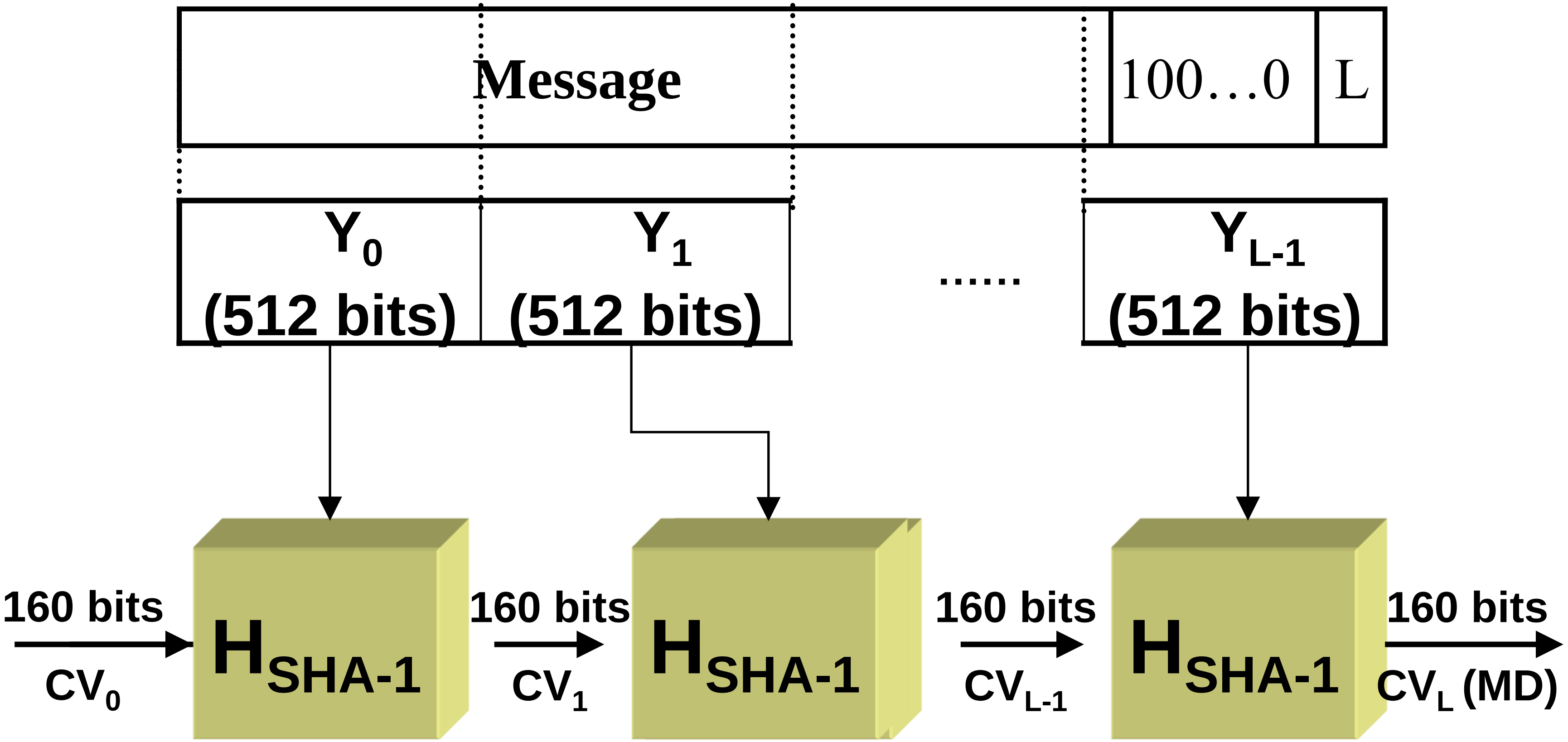
Birthday Attack *Appendix 11A P250*

Q: How many students must be in a class so that there is a greater than 50% chance that

1. one of the students shares the teacher's birthday (up to the day and month)?
2. any two of the students share the same birthday (up to the day and month)? ——Birthday paradox(悖论)

366/2=183

$$1.18*(366)^{1/2}\approx 23$$



$Y=Y_0Y_1\ldots Y_{L-1}$ (Y_i has 16 words=512bits)

CV_0 (Initial Value of Buffer(ABCDE)) (5 word)

$CV_{q+1}=H_{SHA-1}(CV_q,Y_q)$

$MD=CV_{q+1}$（Output）

9 Message Authentication

- **Requirements for MACs** *P392*

1. **knowing a message and MAC, is infeasible to find another message with same MAC**
   - **infeasible to find collision**
2. **MACs should be uniformly distributed**
   - **Resistance against brute-force attack based on chosen plaintext**
3. **MAC should depend equally on all bits of the message**
   - **"weak spots" of message don't exit**

- MACs Based On Hash Functions: HMAC
  - original proposal:
    - `KeyedHash= Hash(Key|Message)` forge?
    - `KeyedHash= Hash(Message|Key)` collision?
- MACs Based On Block Ciphers: DAA and CMAC

10 Digital Signatures

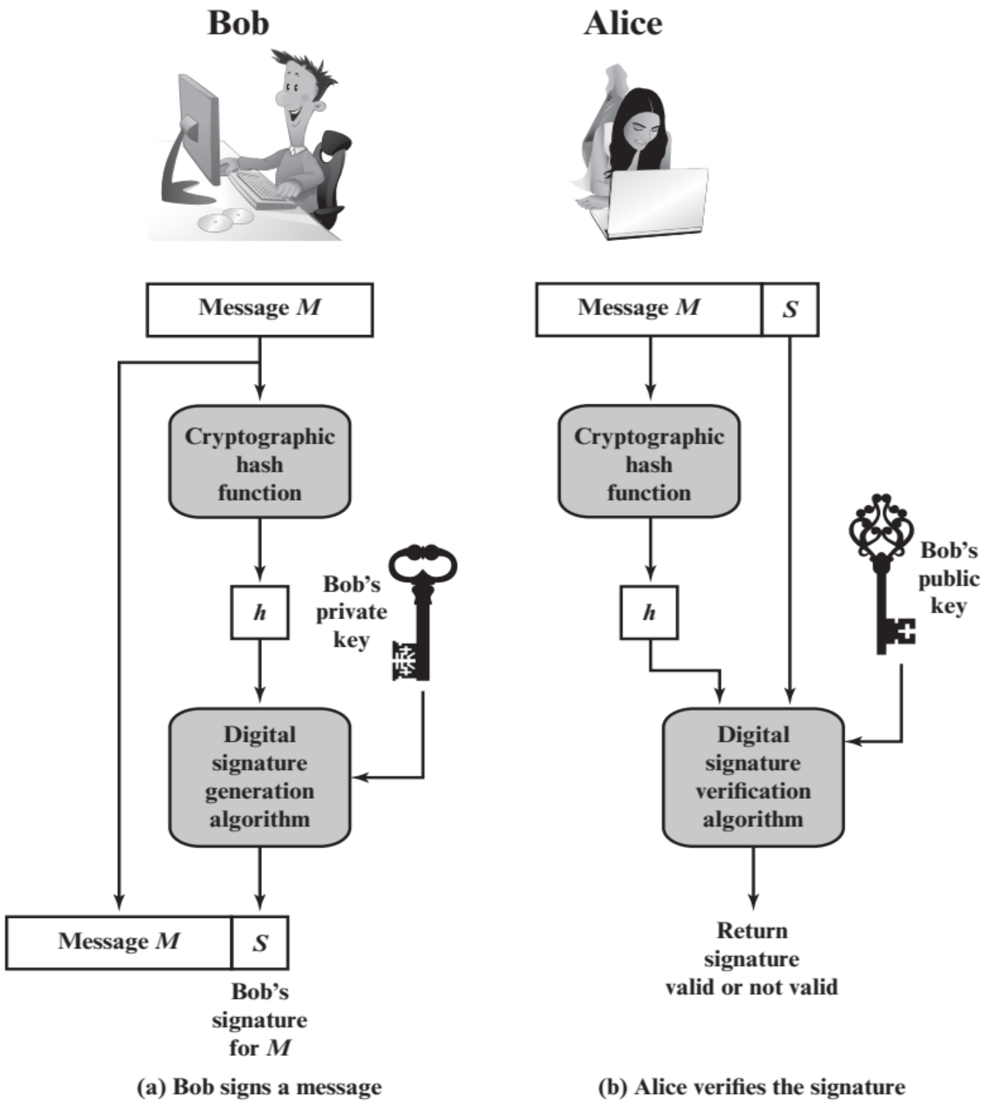


Figure 13.1 Simplified Depiction of Essential Elements of Digital Signature Process

15 User Authentication

- **Entity authentication: is the process whereby one party(*verifier*) is assured of the identity of a second party(*claimant*) involved in a protocol, and that the second has actually participated.**
- **How?**
  - **Passwords: "Encrypted" password**
  - **Challenge-response Identification: Time-variant parameters**
    - **Challenge: *Prover/Claimant* ← *Verifier* *(not necessary)***
    - **Response: *Prover/Claimant* → *Verifier***
  - **Customized and Zero-knowledge Identification Protocols**
    - **A → B : (public) witness computed from a random element**
    - **A ← B : challenge by selecting one question.**
    - **A → B : response by answering question (and further B judges by checking for its correctness).**

Objectives of identification protocols

- For honest parties A and B, A is able to successfully authenticate(证明) itself to B, i.e., B will complete the protocol having accepted A’s identity
- (No transferability) B cannot reuse an identification exchange with A so as to successfully impersonate A to a third party C.
- (No impersonation) The probability of successful impersonation is negligible.

Basis of Identification

- something known.
  - E.g., passwords, Personal Identification Numbers (PINs), and the secret or private keys.
- something possessed: typically a physical accessory.
  - E.g., magnetic-striped cards, *chipcards*, and hand-held customized calculators (*password generators*) which provide time-variant passwords.
- something inherent (to a human individual): use of human physical characteristics and involuntary actions (*biometrics*)
  - E.g., handwritten signatures, fingerprints, voice, retinal patterns, hand geometries, and dynamic keyboarding characteristics. (not discussed further here)