Wireless Network Security and Privacy

Information Security Basics

Xiaoyu Ji

Department of Electrical Engineering
Zhejiang University

2024 Autumn

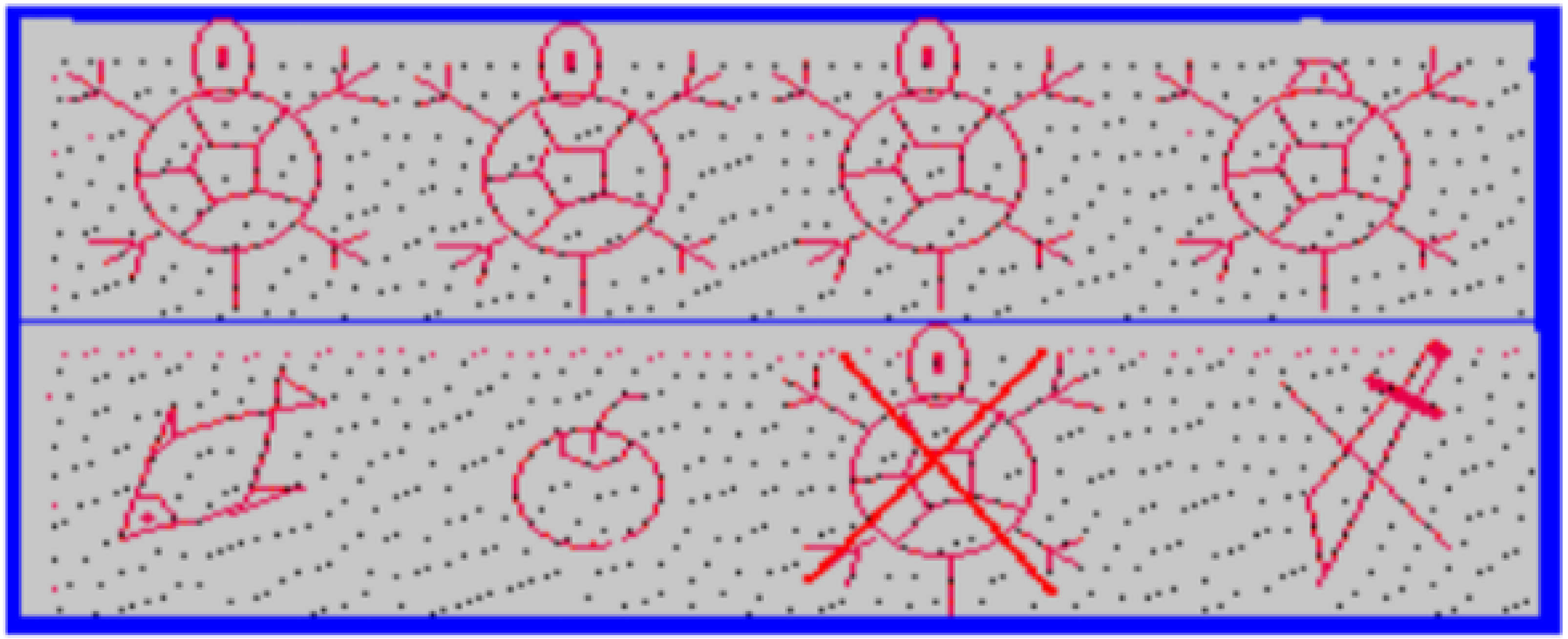
Course Outline

- Cryptography development and some concepts
- Classical Cryptography:
  - Caesar cipher
  - Affine cipher
  - Vigenère cipher
- Disposable Password Book
- modern cryptography

What is cryptography?

Ancient Chinese Cipher Art

- **古代留守在家的妻子给外出工作的丈夫的书信**



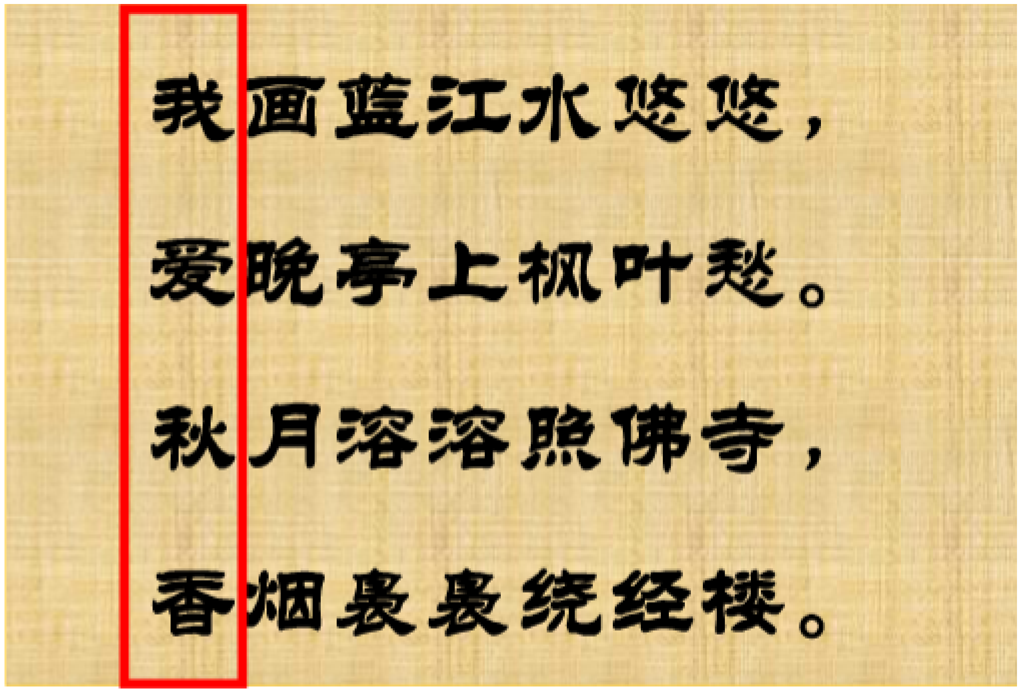
归，归，归！速归！如果（鱼果）不归，一刀两断

Ancient Chinese Cipher Art

- **会意诗**

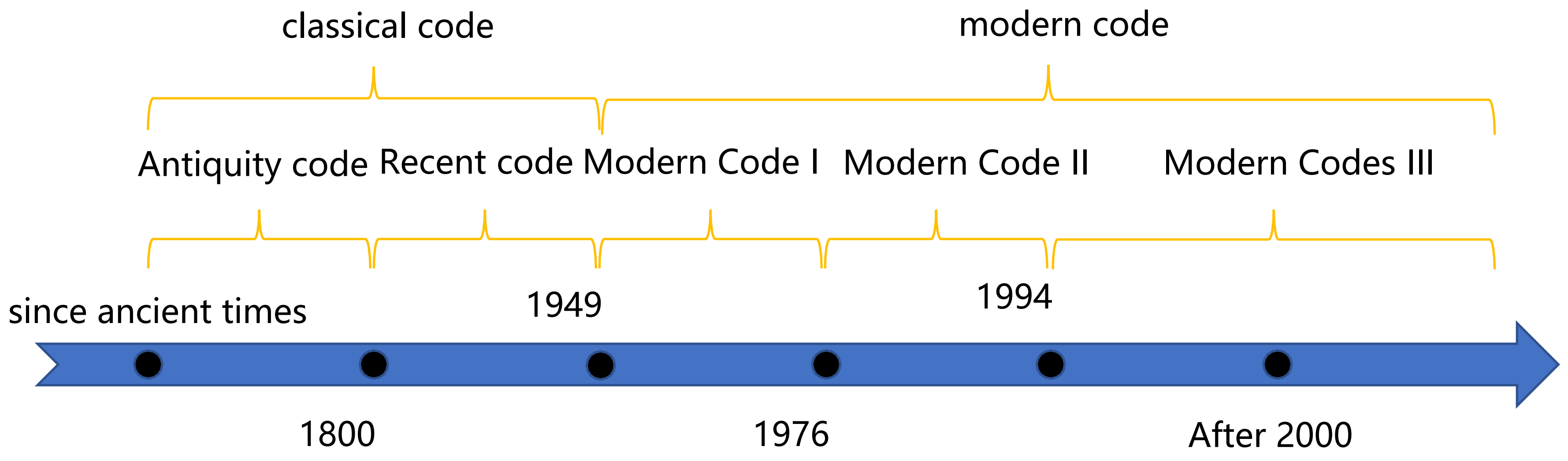
Ancient Chinese Cipher Art

- **藏头诗**



我画蓝江水悠悠，
爱晚亭上枫叶愁。
秋月溶溶照佛寺，
香烟袅袅绕经楼。

Cryptography Development Timeline



Classical code stage

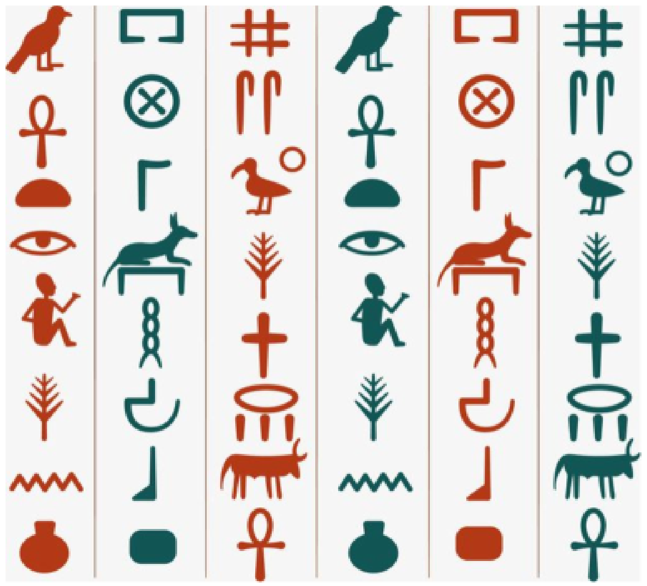
- **Time:**
  pre-1949
- **Features:**
  Cryptography is not yet a science, but the art emerges as the basic means of cryptographic algorithm design (**substitution & permutation**)
- **Confidentiality:**
  Data confidentiality **based on encryption algorithms**
- **Milestone Event:**
  The principle of cryptographic encoding was first made explicit by Kirchhoff in 1883

"Encryption algorithms should be based on the fact that the disclosure of the algorithm does not affect the security of the plaintext and the key, i.e., security depends on the secrecy of the key"

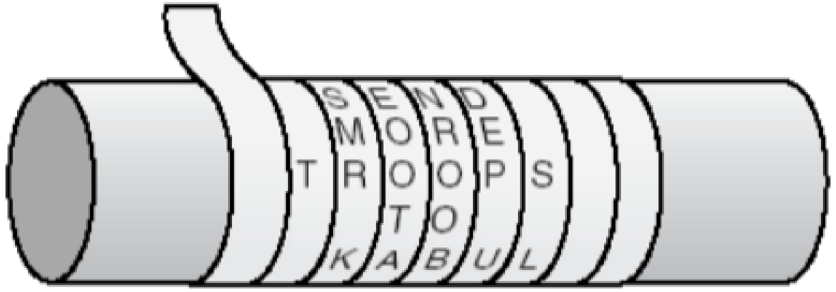
Substitution and Replacement in Classical Codes

- **Substitution: A character in the plaintext is replaced with another character in the ciphertext.**

- **Permutation: the letters of the plaintext remain unchanged, but the order is disrupted.**



ancient Egyptian hieroglyphics

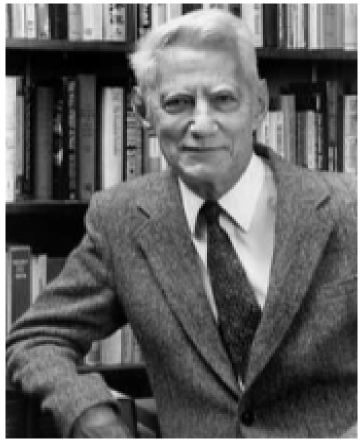


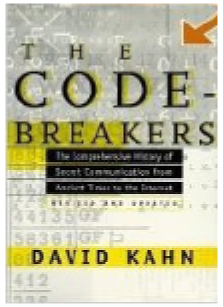
Ancient Spartans used the Book of Days

Wrap the parchment in the form of a long band around a round wooden stick and write on it; when the parchment is unwrapped, there is only a jumble of characters on it, and it can only be wrapped in the same way again around a stick of the same thickness

Modern Cipher Phase I

- **Time:**
  1949-1976
- **Features:**
  - ◻Cryptography goes from art to science
- **Milestone Event:**
  - ◻Shannon published "The Communication Theory of Secret Systems" in 1949.
  - ◻1967, David Kahn monograph The Code Breakers


Shannon



The Code Breakers

Modern Cipher Phase |

- ☐ Several technical reports published by Horst Feistel and others at IBM Waston Laboratory, 1971-1973
- ☐ In 1974, IBM submitted LUCIFER, which later became DES
- ☐ New feature: data security **based on key, not algorithm, secrecy**

- ■ **Shannon's contribution**
  - ■ Defines theoretical security and proposes the principles of diffusion and confusion.
  - ■ Laid the theoretical foundations of cryptography
- ■ **Diffusion:** spread each bit of plaintext into as many output ciphertexts as possible to hide the statistical properties of the plaintext digits.
- ■ **Obfuscation:** to make the relationship between the statistical properties of the ciphertext and the plaintext key as complex as possible

Modern Cryptography Phase II

- **Time:**

  1976-1994

- **Features:**
  - Public keys are starting to appear

- **Milestone Event:**
  - In 1976, Diffie & Hellman's "New Directions in Cryptography" introduced the concept of **public key cryptography**.
  - In 1977 Rivest, Shmir & Adleman proposed the RSA public key algorithm.

Diffie & Hellman win 2015 Turing Award

Modern Cipher Phase III

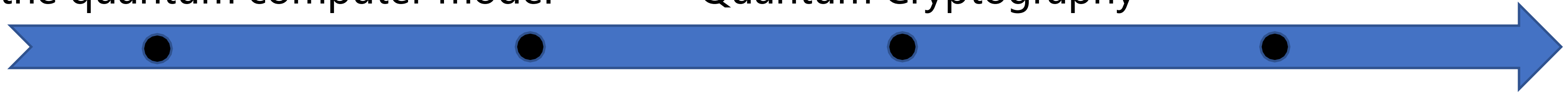
- Development Timeline:

1994
Shor proposes decomposition of large integer and polynomial time algorithms in the quantum computer model

2000
AES officially replaces DES as the new encryption standard

2006
First International Symposium on Post-Quantum Cryptography

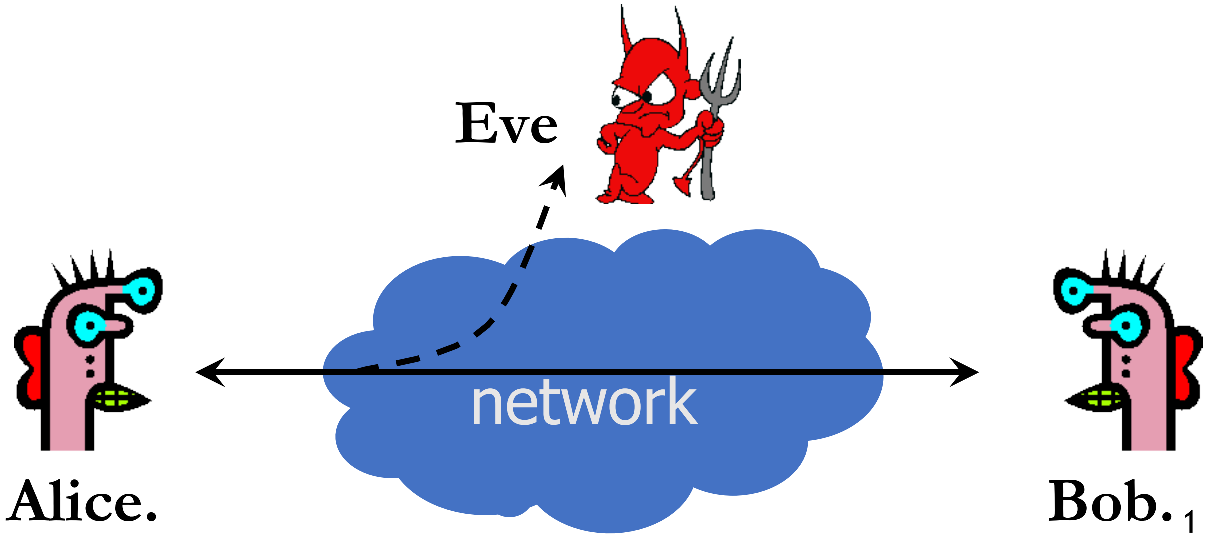
2017
NIST opens call for post-quantum cryptography standards



Information security basics

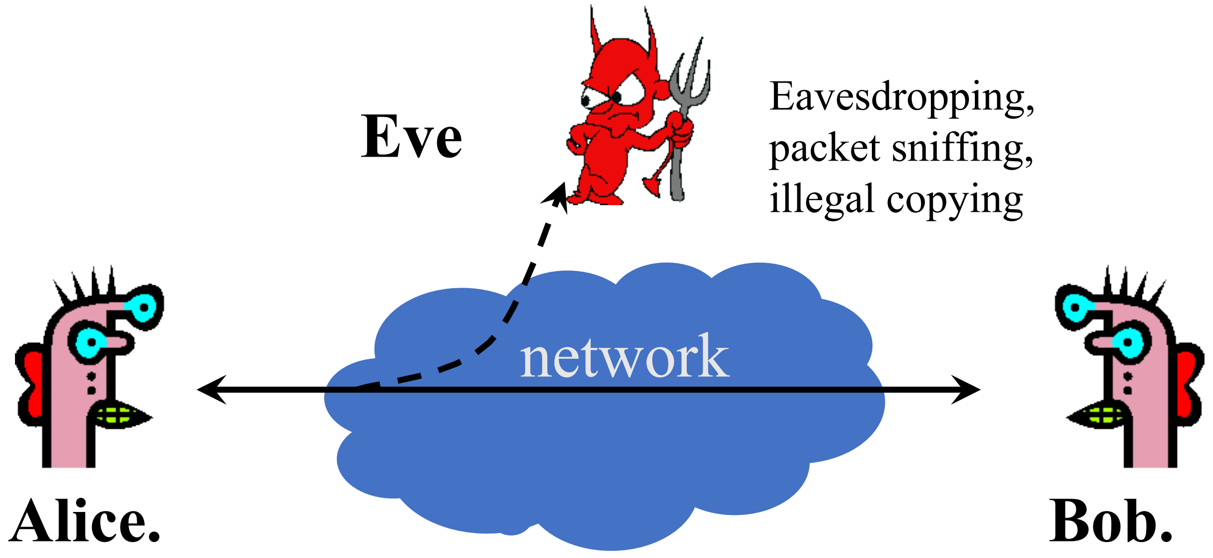
Objectives of information security

- Confidentiality: The assurance that information/data will not be disclosed to unauthorized individuals.
- Integrity: the assurance that information/data will not be modified by unauthorized individuals.
- Availability: Ensuring that information/data can be accessed by authorized individuals.
- Authenticity: the assurance that the information/data actually came from the source it purports to be.
- Non-repudiation: guarantees that all participants in a message/data interaction cannot deny that messages and data were ever sent.



Confidentiality

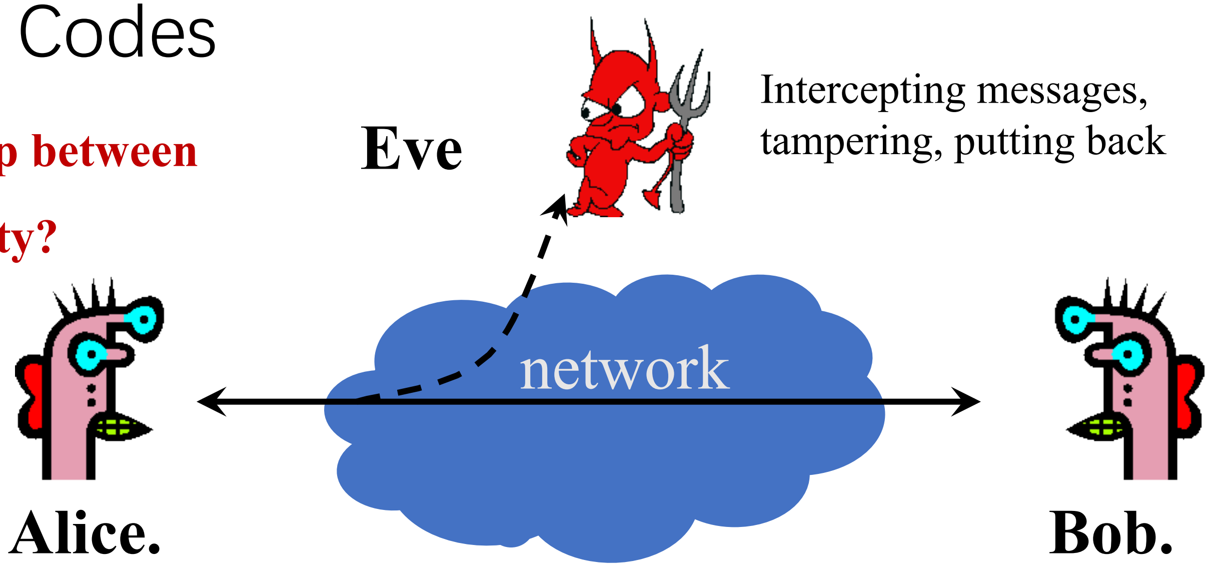
- The attacker can't parse out Alice's message to Bob.
- Defense method: encryption and decryption algorithms



Integrity

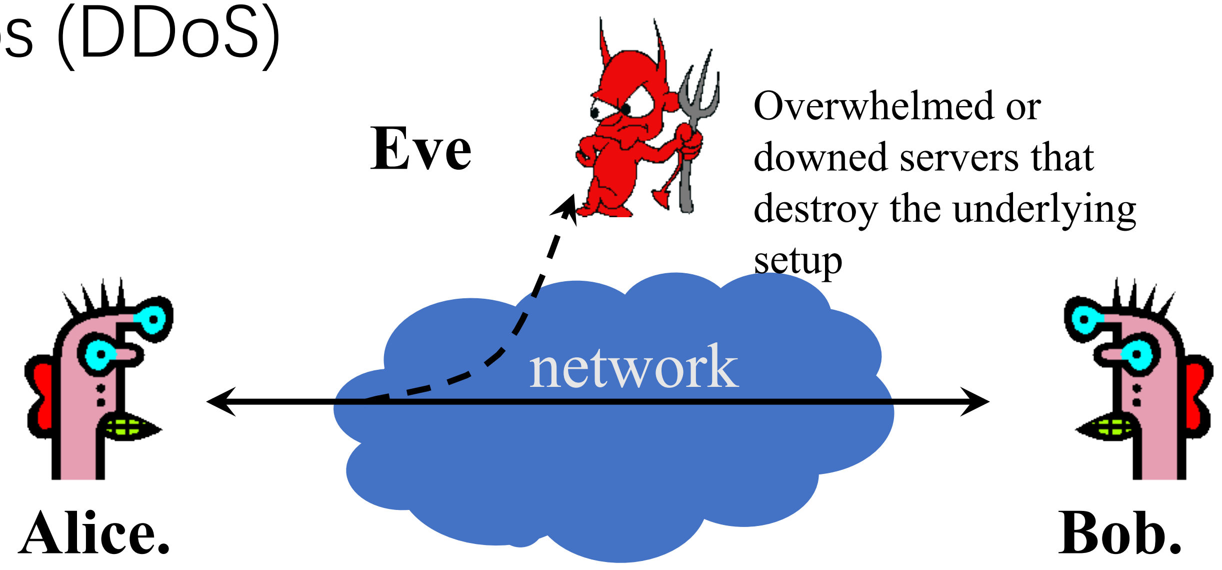
- Bob wants to make sure Alice's message hasn't been tampered with.
- Defenses: Hash Functions, Error Correcting Codes, Message Authentication Codes

Q: What is the relationship between confidentiality and integrity?



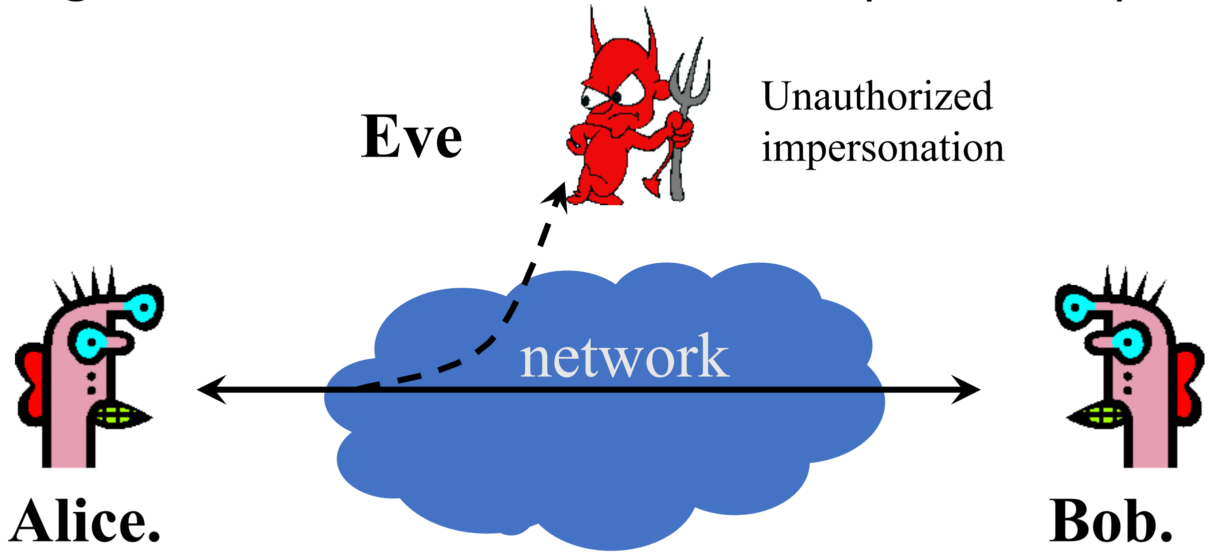
Availability

- Usability is the ability of users to use the information or resources they need
- Attacks that disrupt availability: Denial of Service (Dos), Distributed Dos (DDoS)


Eve
Overwhelmed or downed servers that destroy the underlying setup
network
Alice.
Bob.

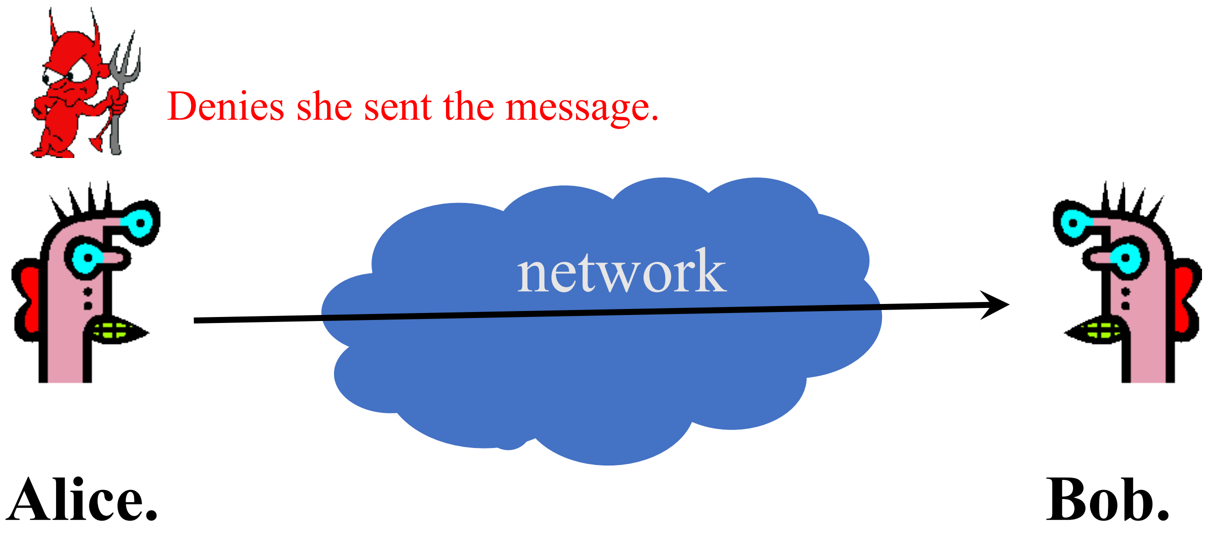
Authenticity

- Ability to identify/entity authenticate and determine data source
- Defense tools: cryptographic schemes, digital signatures, hash functions, message authentication codes, poll-response protocols



Non-repudiation (non-repudiation)

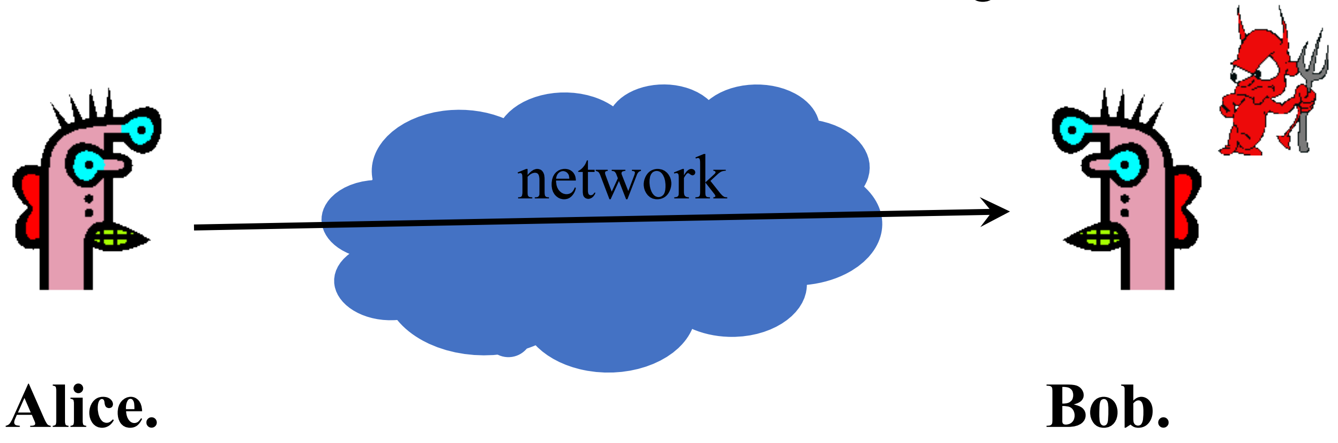
- Alice can't claim she didn't send the message
- Defense Method: Digital Signature



Anonymity

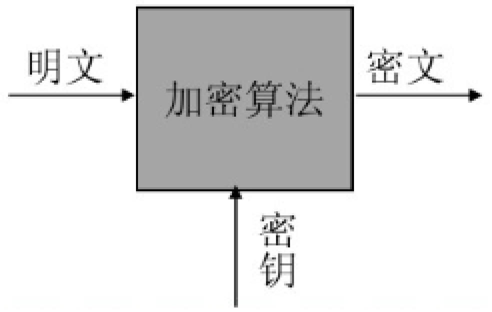
- The ability of the sender to hide his identity

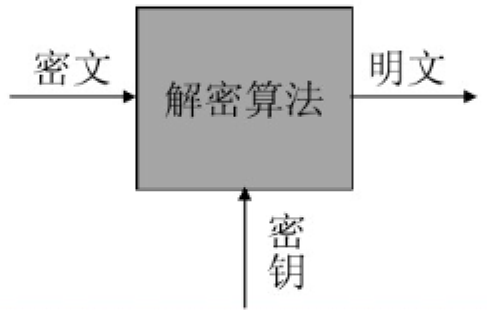
Bob identified Alice from the messages sent to him.



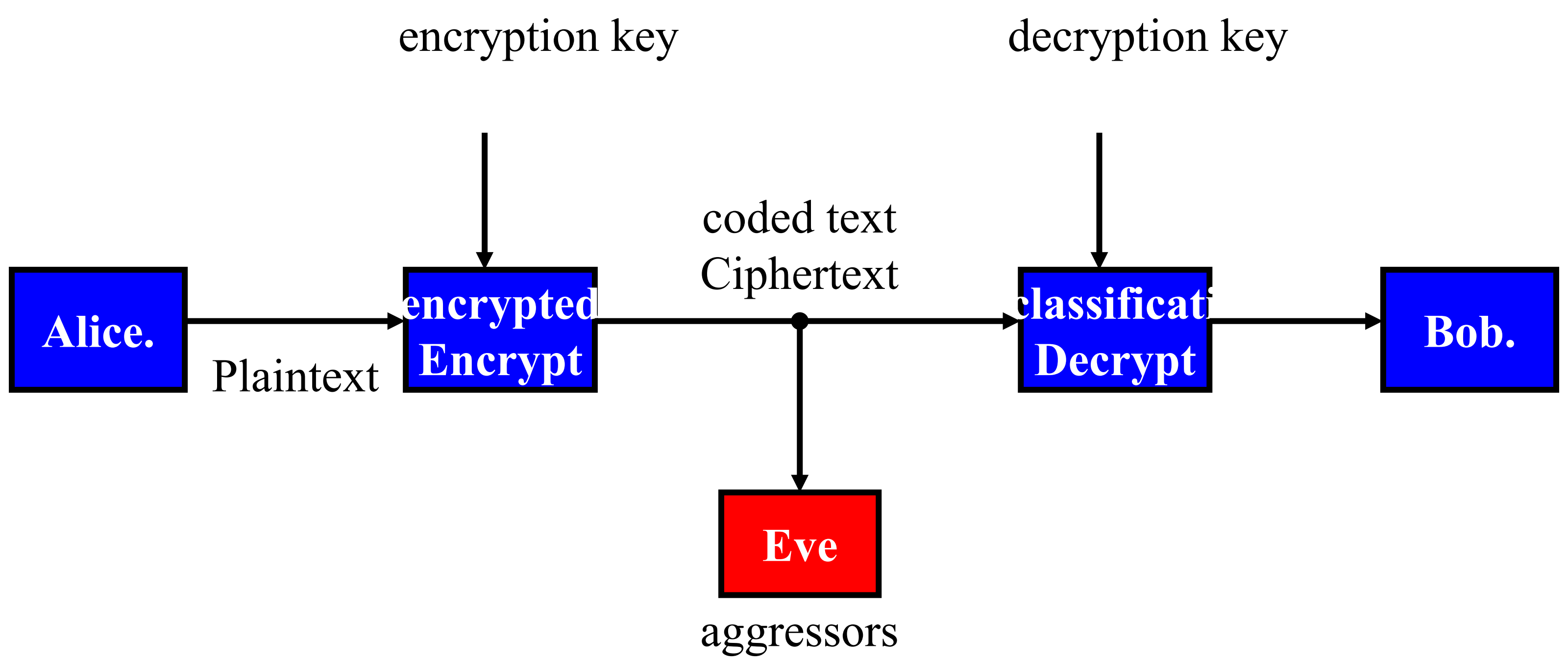
Basic concepts of cryptography

- Plaintext: information that needs to be transmitted secretly
- Ciphertext: the message in plaintext after ciphertext transformation
- Encryption (E(m)): the transformation from plaintext to ciphertext.
- Decryption (D(c)): the process of recovering plaintext from ciphertext.
- Deciphering: the process by which an illicit recipient attempts to analyze plaintext from ciphertext
- Key: a set of secret information used in encryption and decryption
- Encryption/decryption algorithms: rules for encrypting/decrypting plain/cipher text

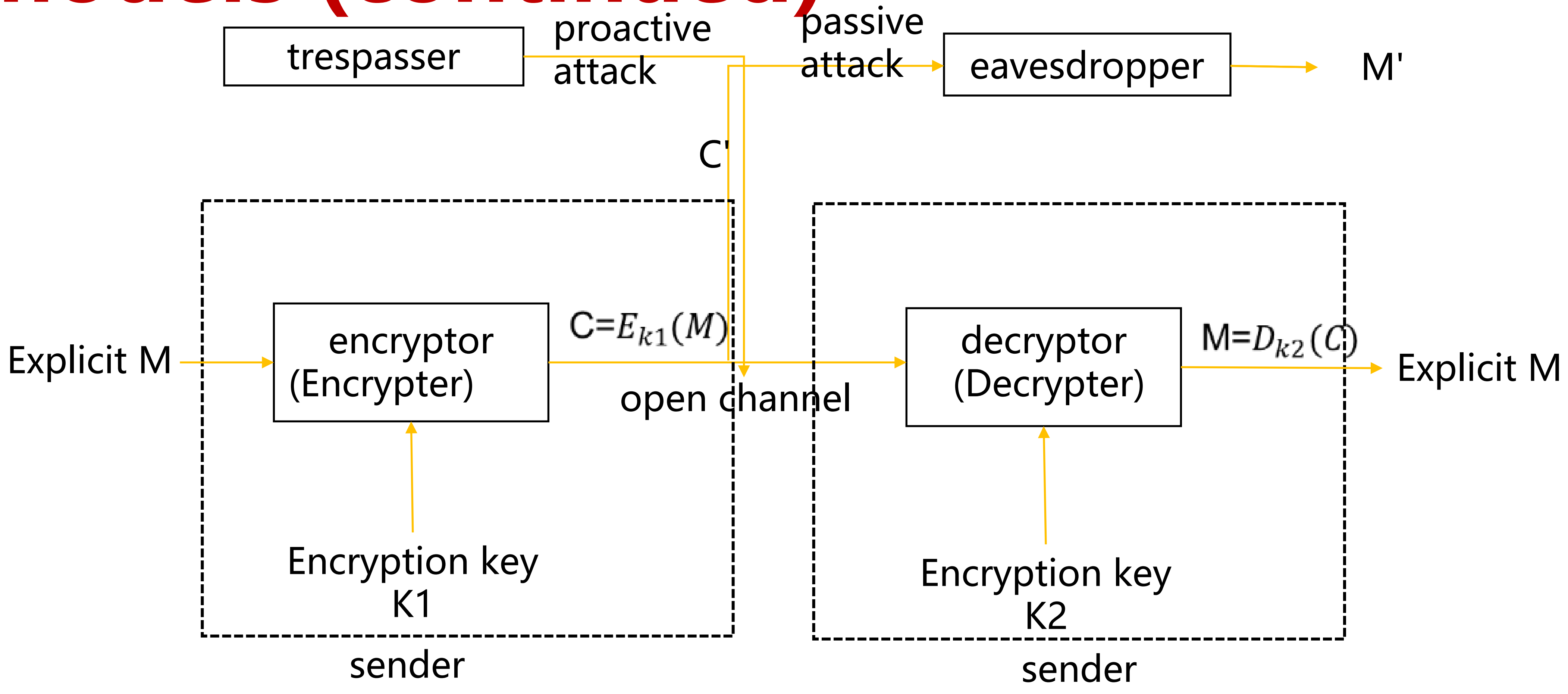




Cryptographic communication model



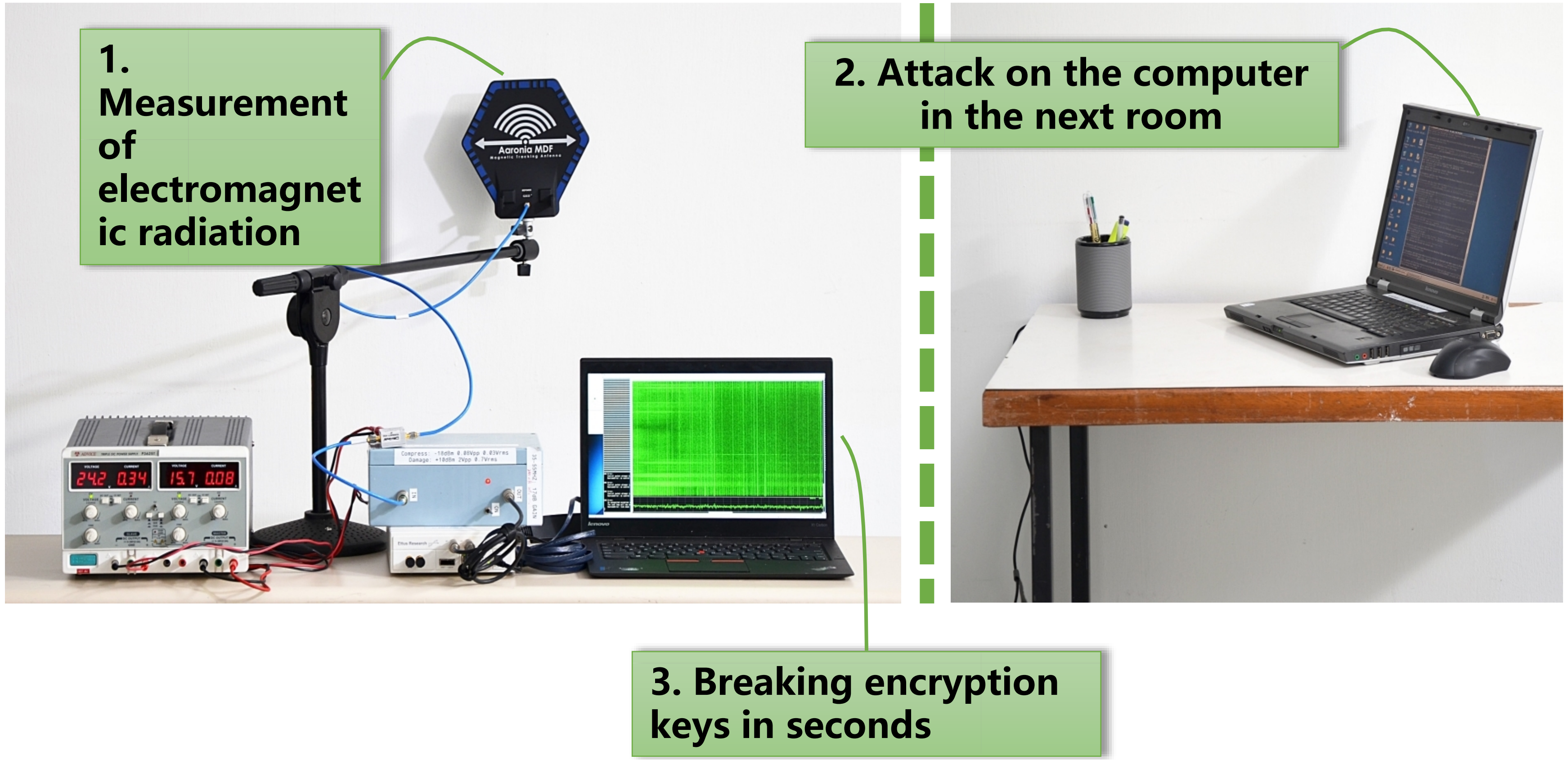
Cryptographic communication models (continued)



cryptanalysis

- cryptanalysis
  - Classical Code Analysis
    - brute force attack method
    - calculus
  - Implementing the attack: Side Channel analysis, e.g., measuring the power consumption of the processor handling the private key.
  - Social Engineering: Obtaining keys through bribery, blackmail, stalking, or detective work

Side channel / side channel attacks --Encryption key cracking


1. Measurement of electromagnetic radiation
2. Attack on the computer in the next room
3. Breaking encryption keys in seconds
Aaronia MDF

[1] Genkin D, Pachmanov L, Pipman I, et al. Stealing keys from PCs using a radio: cheap electromagnetic attacks on windowed exponentiation[C]// International workshop on cryptographic hardware and embedded systems. Springer, Berlin, Heidelberg, 2015: 207-228.
[2] Genkin D, Pachmanov L, Pipman I, et al. ECDH key-extraction via low-bandwidth electromagnetic attacks on PCs[C]//Cryptographers' Track at the RSA Conference. Springer, Cham, 2016: 219-235.

social engineering

- Social engineering: A way of communicating legitimately with others, exploiting the victim's psychological weaknesses, instinctive reactions, trust, curiosity, greed, etc. to psychologically influence him or her to perform certain actions or disclose some confidential information.

- Tactics: Usually in the form of conversation, false pretenses, flirting, online chatting, baiting, equivocation, sympathy, tailing, etc.

- Former number one hacker in the United States, Kevin Mitnick, is considered a master and pioneer of social engineering and is the author of the security book, The Art of Deception

Cryptographic Security Attack Classification

- **Classification of attacks (with the aim of obtaining keys and encryption/decryption algorithms)**
  - ■ **Cipher-only attack**: The attacker has one or more ciphers and the attack requires statistical analysis;
  - ■ **Known-plaintext attack**: The attacker has a copy of the ciphertext and the corresponding plaintext and performs algorithm and key derivation;
  - ■ **Chosen-plaintext attack**: The attacker has access to an encrypting machine, so he can choose any plaintext and generate the corresponding ciphertext with a higher probability of attack;
  - ■ **Chosen-cipher attack**: The attacker has access to a decryption machine, so he can choose some ciphertext and generate the corresponding plaintext.

Attacks are progressively more difficult from bottom to top

Cryptographic Algorithm Security

- Key Length

Bruce Schneier Public Key Length Suggested Value

year (e.g. school year, fiscal year)	For individuals	For companies	For the government
1995	768	1280	1536
2000	1024	1280	1536
2005	1280	1536	2048
2010	1280	1536	2048
2015	1536	2048	2048

The longer the key length, the less likely the encrypted data will be illegally decrypted

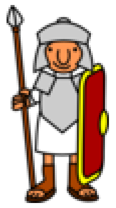
3.2 Classic Cryptograph

classical cryptography

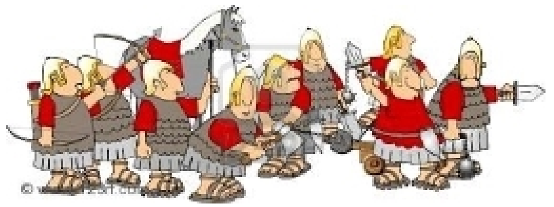
How did Julius Caesar command his army?

At 3:00 p.m., attacked the enemy's castle with 100,000 troops

Military orders could be stolen.



A messenger conveys orders to the army



How did Caesar get military orders safely delivered to the army?

3.2.1 Shift encryption (Caesar Cipher) Caesar Cipher

1. Since there are 26 letters in the English alphabet, it is possible to establish a correspondence between the English alphabet and the remainder of mode 26:

$$A = 0\ B = 1\ C = 2\ \ldots\ Y = 24\ Z = 25$$

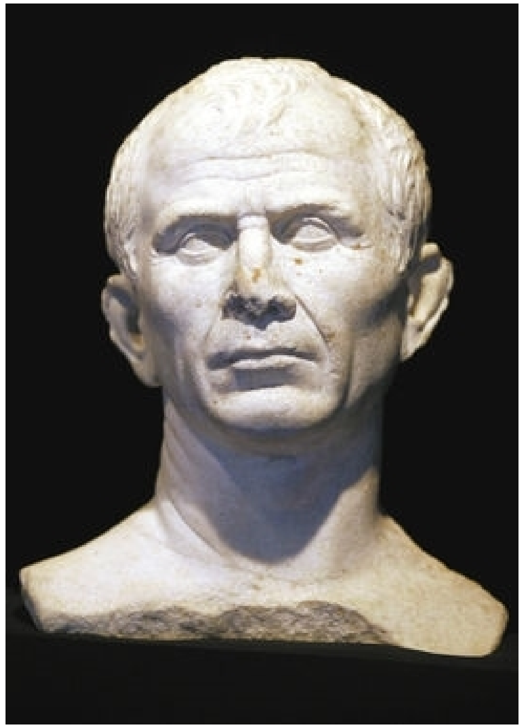
2. Encryption process.

$$y = x + k \pmod{26}$$

3. Decryption process

$$x = y - k \pmod{26}$$

4. where k is the encryption key. Caesar encrypts with k = 3.



Shift encryption security

- Caesar encryption, as follows

明文字母	ABCDEFGHIJKLMNOPQRSTUVWXYZ
密文字母	DEFGHIJKLMNOPQRSTUVWXYZABC

- If the plaintext is LOVE, then the ciphertext is ORYH.

- **Think about how to attack this encryption.**
  - Known plaintext attack: x,y are known, then

$$k = y - x \pmod{26}$$

  - Choice Explicit Attacks: the ability to choose your own choice of x. For example, choosing x='a'=0, then

$$y = k \pmod{26}$$

  - Choice of ciphertext attack: choose the ciphertext y yourself. e.g. choose y='a'=0, then

$$x = - k \pmod{26}$$

Shift encryption security (continued)

- **How do you attack if you only know the ciphertext?**
- Violent Exhaustive Enumeration: Try All 26 Possibilities

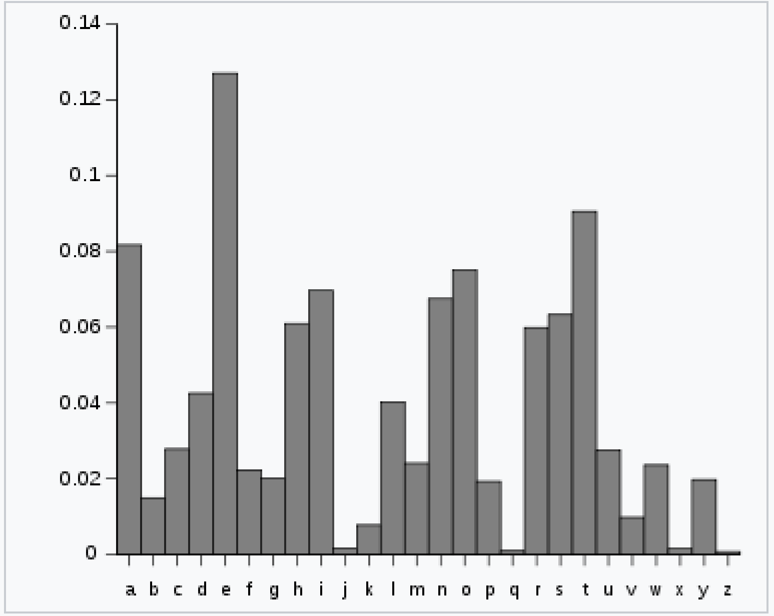
- **Word frequency statistics:**

1. Counting the frequency of letters in ciphertexts

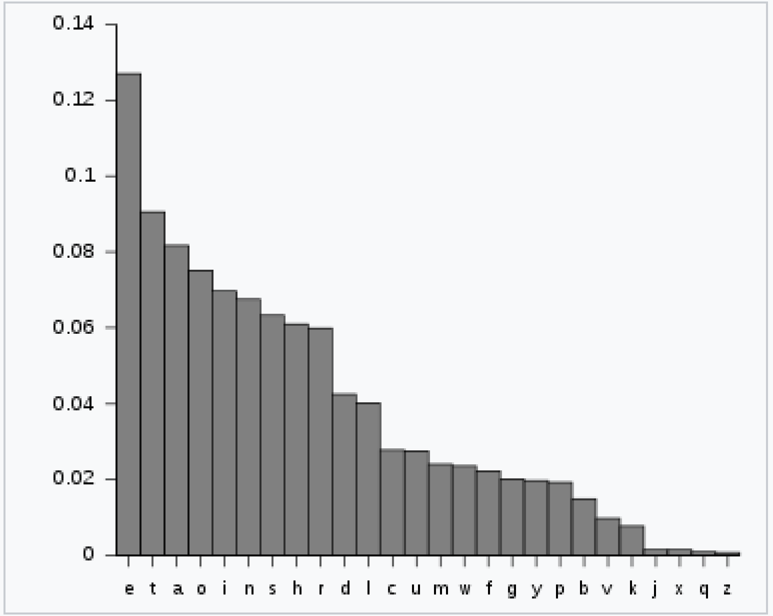
2. Comparison with the frequency of occurrence of standardized linguistic letters

3. Determination of the most probable value of key k

Word frequency statistical methods



英语语言材料中的字母频率



按大小排序后的英语字母频率

3.2.2 Affine cipher

- Encryption: given the key (α,β)

$$y = \alpha x + \beta \pmod{26}$$

- Decryption:

$$x = \frac{1}{\alpha}(y - \beta) \pmod{26}$$

Note that 1/α is not the inverse of α, but the inverse element.

- Simple definition of inverse: the inverse of a is the value of b that satisfies ab = 1 (mod 26)

For example, a=7,n=26 Then we find the inverse of a:

7*1 = 7 mod 26 7*4 = 28 = 2 mod 26

7*2 = 14 mod 26 7*5 = 9 mod 26

7*3 = 21 mod 26 7*6 = 16 mod 26 ...

yields 7*15=1mod26, i.e., 15 is the inverse of 7 mod 26.

Imitation encryption security

- Theorem: for α, if there exists an inverse element with respect to modulus n, then gcd(α,n)=1 needs to be satisfied.

- In order to obtain the inverse element, it is necessary to make gcd(α,26)=1, i.e., α and 26 must be mutually prime, and thus α takes on a range of values:

  {1, 3, 5, 7, 9, 11, 15, 17, 19, 21, 23, 25}

  There are only 12 values, and in addition β takes on a range of 26, so for affine encryption, there are only 12 * 26 = 312 possibilities for all!

It's very insecure with the current computing power of computers.

- Attacks on Imitation of Projective Encryption:
  - Violent exhaustion: exhaust all 312 possibilities
  - Word frequency statistics: think about what it takes?

Example of affine encryption - guess what?

Pu yfo of oin hvy ufa hrpkpyb, jlar ph hopkk py oin hvy oinan, svo jnjpkk klvbi rfan zfyupgnyo zlkr; pu ovayng of ufvyg iph fjy hilgfj, lmmafmaplon nhzlmn, oin hvy jpkk sn oiafvbi oin inlao,jlar nlzi mklzn snipyg oin zfayna; pu ly fvohoanozing mlkr zlyyfo ulkk svoonaukx, oiny zknyzing jlcpyb larh, bpcny mfjna; pu P zly'o ilcn sapbio hrpkn, po jpkk ulzn of oin hvyhipyn, lyg hvyhipyn hrpkn ofbnoina, py uvkk skffr.

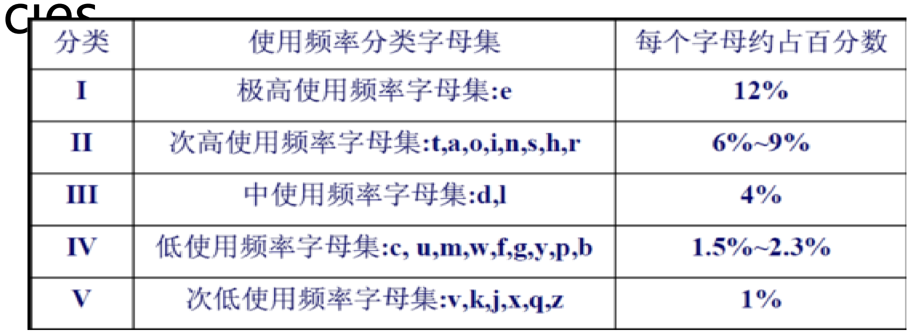
affine encryption crack

- Affine encryption is a linear mapping, so the plaintext and the corresponding ciphertext appear with the same frequency.

1. Counting the frequency of letters in ciphertexts

{'a': 18, 'c': 3, 'b': 7, 'g': 8, 'f': 19, 'i': 23, 'h': 17, 'k': 22, 'j': 10, 'm': 7, 'l': 21, 'o': 30, 'n': 37, 'p': 26, 's': 6, 'r': 10, 'u': 11, 'v': 13, 'y': 27, ' x': 1, 'z': 12}

2. Find two plaintext-ciphertext mappings by comparing them with standard letter frequencies



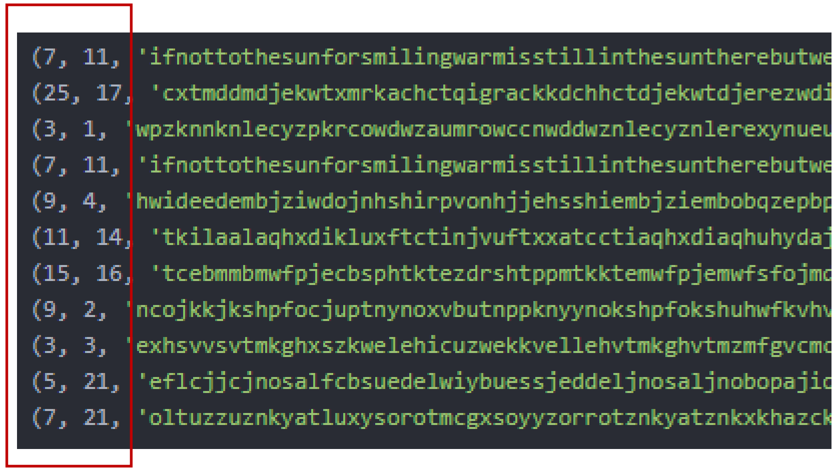
分类	使用频率分类字母集	每个字母约占百分数
I	极高使用频率字母集:e	12%
II	次高使用频率字母集:t,a,o,i,n,s,h,r	6%~9%
III	中使用频率字母集:d,l	4%
IV	低使用频率字母集:c, u,m,w,f,g,y,p,b	1.5%~2.3%
V	次低使用频率字母集:v,k,j,x,q,z	1%

3. Determining mapping relationships

e→n; t→o

affine encryption

4. Decrypt the ciphertext by solving the key (a,b) according to the mapping relation.



```
(7, 11, 'ifnottothesunforsmilingwarmisstillinthesuntherebutwe
(25, 17, 'cxtmddmdjekwtxmrkachctqigrackkdchhctdjekwtdjerezwdi
(3, 1, 'wpzknnknlecyzpkrcowdwzaumrowccnwddwznlecyznlerexynueu
(7, 11, 'ifnottothesunforsmilingwarmisstillinthesuntherebutwe
(9, 4, 'hwideedembjziwdojnhshirpvonhjjehsshiembjziembobqzepbp
(11, 14, 'tkilaalaqhxdikluxftctinjvuftxxatcctiaqhxdiaqhuhydaj
(15, 16, 'tcebmmbmwfpjecbsphtktezdrshtppmtkktemwfpjemwfsfojmc
(9, 2, 'ncojkkjkshpfocjuptnynoxvbutnppknyynokshpfokshuhwfkvhv
(3, 3, 'exhsvvsvtmkghxszkwelehicuzwekkvellehvtmkghvtmzmfgvcmc
(5, 21, 'eflcjjcjnosalfcbsuedelwiybuessjeddeljnosaljnobopajic
(7, 21, 'oltuzzuznkyatluxysorotmcgxsoyyzorrotznkyatznkxkhazck
```

In practice, there may be more than one set of keys (a,b) assumed for cracking, but for the current computational power of computers, affine encryption can be cracked very quickly, even using an exhaustive approach

If you don't ask the sun for a smile, the warmth is still in the sun, but we will smile more confidently and calmly; if you turn around and find your own shadow, appropriate hiding, the sun will be able to pass through the heart, warm every corner behind; if the spread palm can't point down the butterflies, then tightly clenched into a fist waving arms, giving strength; if I can't smile brilliantly, then throw your face into the bright sunshine, with the If I can't smile brightly, then I will throw my face to the bright sunshine and smile with the sunshine.

Example derivation process

- 1. Solve for the inverse element of a:

$$7*15=1\bmod 26$$

$$1/a=15$$

- 2. The plaintext X is obtained from x = 1/a * (y-b) mod 26, knowing that 1/a = 15:

coded text	p	u	y	f	o	o	f
y	15	20	24	5	14	14	5
x=1/a(y-b)	60	135	195	-90	45	45	-90
x mod 26	8	5	24	5	14	14	5
plaintext	i	f	n	o	t	t	o

3.2.3 Virginia cipher Vigenère cipher

- **History of the Virginia Code**
  - The Virginia Cipher was first created in 1553 by Giovan Battista Bellaso in his book, The Cipher of Mr. Giovan Battista Bellaso.
  - Mistakenly attributed to a Virginia invention in the 19th century.
  - The Virginia Code was invented as an automatic key cipher.

Blaise de Virginia.

The Virginia Code

- **Encryption method**:
  - List plaintexts and group them by key length
  - Shift encryption of **each grouped** letter with a key
  - Encryption formula: C=(P+K)mod26

- **Features**:
  - The Virginia cipher is actually an extension of the shift cipher.
  - Ability to **eliminate frequency characteristics of letters**

Q:Think about why?

- for example

H	e	r	e	i	s	h	o	w	i	t	
21	4	2	19	14	17	21	4	2	19	14	
c	i	t	x	w	j	c	s	y	b	h	

Security of the Virginia Code

- Cracking the Virginia Code
  - ◻1. Find the key length
  - ◻2. Find the key
- Cracking the key is easy after finding out the key length (Why?):
  - ◻Take the ciphertext according to the key length L and select the 1st, L+1st, and L+2nd ...... letters in the ciphertext for word frequency counting.
- A way to find the length of the key:
  - ◻Kasiski experiment
  - ◻Friedman test

Kasiski-Kasiski experiment

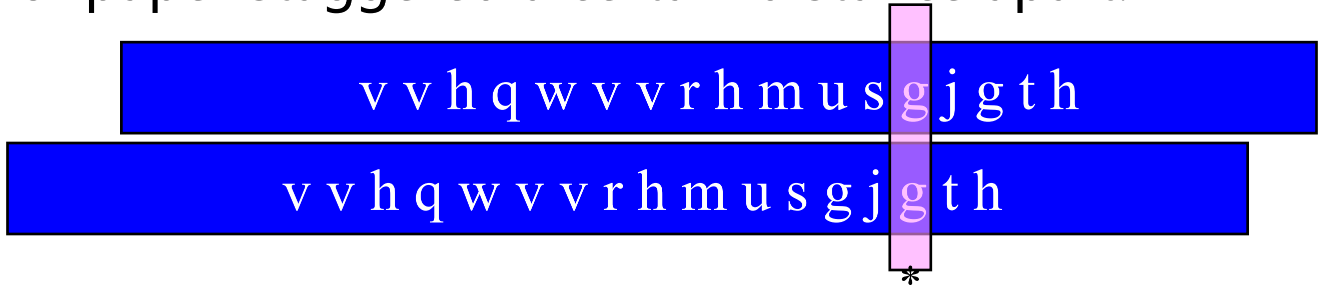
```
Key: FOREST FO RE STFO REST FO RES TFOR
Explicit: better to do well than to say well
Cipher: GSKXWK YC US OXQZ KLSG YC JEQ PJZC
```

12

- Use the patterns of English words to count repetition intervals.
- If "YC": interval 12, then the number of conventions: 1, 2, 3, 4, 6, 12 may be the key length.
- Other fields

Find the key length

1. Write the secret message on two strips of paper. Line them up top and bottom, with the two strips of paper staggered a certain distance apart.


v v h q w v v r h m u s g j g t h
v v h q w v v r h m u s g j g t h
*

2. Mark * in place of two identical letters.

3. Change the staggered position of the two strips of paper to record the number of identical letters.

4. The distance at which the most identical letters occur is most likely to be the length of the key.

Friedman test

- IC, index of coincidence
- IC = P(A)^2 + P(B)^2 + …… + P(Z)^2 = 0.065
- P(i) is obtained from letter frequency analysis
- The rationale for inferring the key length using the recombination index lies in the fact that for a sequence encrypted by a shift, the recombination index of the ciphertext should be equal to the recombination index of the original language, since all the alphabets are shifted by the same degree.
- Example: statistical overlap index in all Harry Potter books

Friedman test (continued)

- Explicit text: ABCDEABCDEABCDEABC

- If the key length is 2 and k=(1,2)

plaintext		coded text
ab		bd
cd	a	df
ea	(1,2)	fc
bc	$\longrightarrow$	ce
de		eg
ab		bd
cd		df
ea		fc
bc		ce

Explicit text group 1: ACEBDACEB

"k=1"

Cipher group 1: BDFCEBDFC

A B C D E F

Explicit: IC = (2/9)^2 + (1/9)^2 + (2/9)^2 + (1/9)^2 + (2/9)^2 0

Ciphertext: IC= 0 +(2/9)^2+(1/9)^2+(2/9)^2+(1/9)^2+(2/9)^2

Explicit group 2: BDACEBDAC

"k=2"

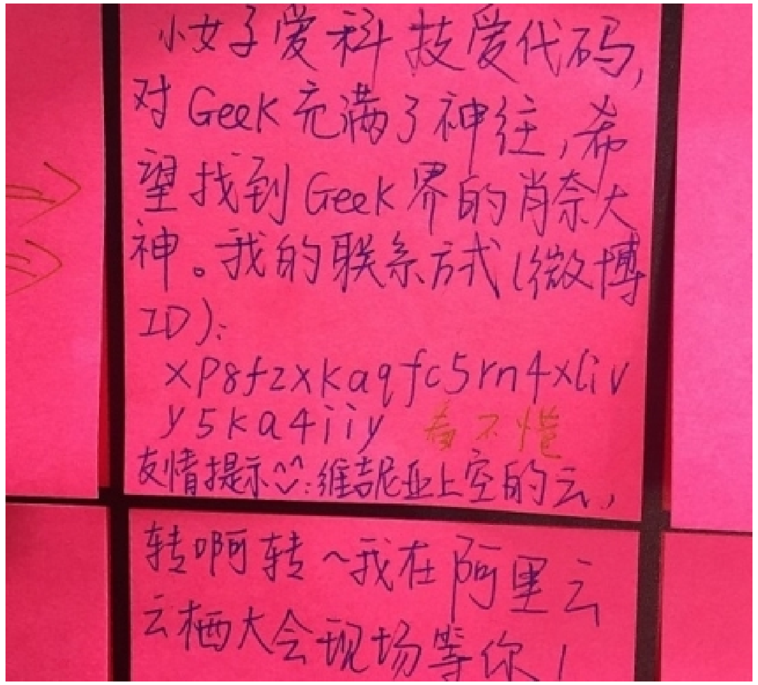
Cipher group 2: DFCEGDFCE

Since all letters have the same degree of displacement, the coincidence index of the ciphertext should be equal to the coincidence index of the original languge

Virginia Encryption Example

- Xe8xbdxafxe5xa4xaaxe5xa4xaa
- Key: aliyun
- Method: Virginia Code Decryption + Secret Key + UTF-8 Chinese Decoding

Explicit: Mrs. Soft (Twitter)



小女子爱科技爱代码，对Geek充满了神往，希望找到Geek界的肖奈大神。我的联系方式(微博ID)：
xp8fzxkaqfc5rn4xliv
y5ka4iiy 看不懂
友情提示😊：维吉尼亚上空的云，

转啊转～我在阿里云
云栖大会现场等你！

3.3 Disposable codebooks

- Both parties jointly maintain a book that is encrypted using a key equal in length to the plaintext. After each encryption, a new key is exchanged. The generation of the key is completely randomized.
- For example, the equivalent of a shift encryption in which each letter is shifted differently
- The one-time codebook is a theoretically absolutely secure encryption method
- For binary encryption, the process is as follows:

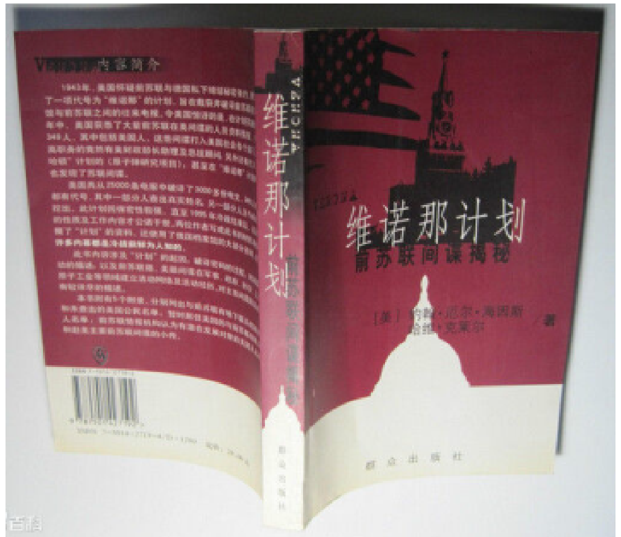
$$\begin{array}{lr} (\text{message}) & 00101001 \\ (\text{key}) & \oplus\ 10101100 \\ \hline (\text{ciphertxt}) & 10000101 \end{array}$$

- Practical application difficulty: to share a key that is as long as the plaintext, if there is a way to share this key absolutely securely, then why don't we just share the plaintext?

Disposable Password Book

- Password books can't be used multiple times: think about why.

  Real life example: 1941-1946, the Soviet Union used a one-time codebook to encrypt messages. The codebook was generated by manually throwing dice, but the Soviet Union felt that throwing dice was too wasteful of manpower, so it used the same codebook multiple times, which eventually led to it being cracked by the U.S. It was known in the U.S. as the Venona Project.



Summary of the chapter

- Knowledge of common classical cryptography such as Caesar ciphers (shift ciphers), affine ciphers, and the mechanism of the Virginia cipher
- Master the security of classical cryptography and the security attacks it faces such as brute force and word frequency statistics attacks
- Understand how classical cryptography influenced modern cryptography

Wireless Network Security and Privacy

Fundamentals of Cryptography - Modern Cryptography

Ji Xiaoyu (1929-), Chinese writer
Zhejiang University
Autumn, 2024

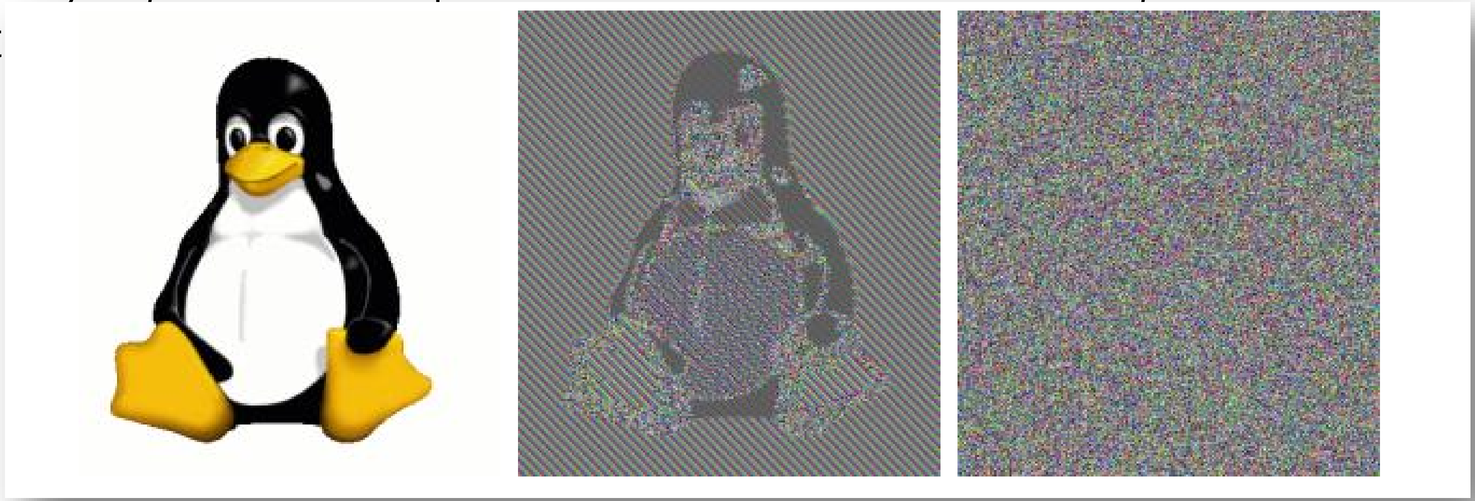
Course Outline

- **Basics**
- **Private key cipher (symmetric cipher)**
  - ◻ DES
  - ◻ 3DES
  - ◻ AES
- **Public key cryptography (asymmetric cipher)**
  - ◻ public key cryptosystem
  - ◻ DH key exchange protocol
  - ◻ RSA encryption
- **Cryptography Applications for the Internet of Things**
  - ◻ cryptographic application
  - ◻ Authentication Applications
  - ◻ Secret key management

Confusion and proliferation

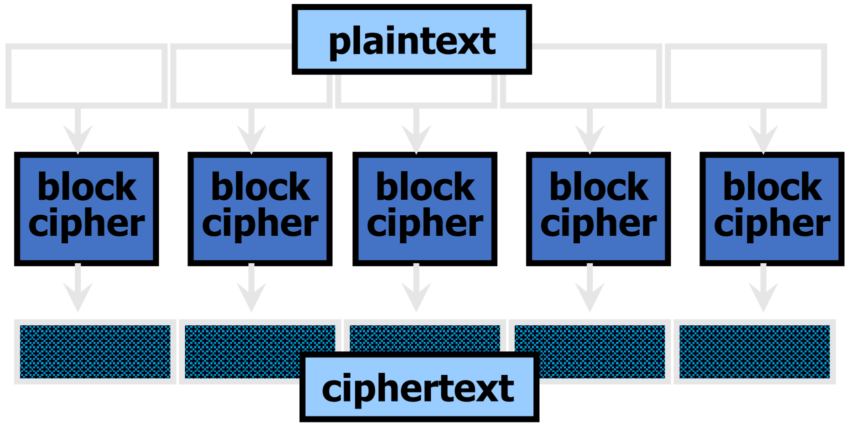
- **Confusion**: A cryptographic operation that makes the relationship between the key and the ciphertext as ambiguous as possible, used in both DES and AES.

- **Diffusion**: A cryptographic operation that spreads the effects of a plaintext symbol to multiple ciphertext symbols in order to hide the statistical properties of the plaintext, e.g., position swapping. Usually modifying 1 bit in the plaintext will result in an average of half of the out

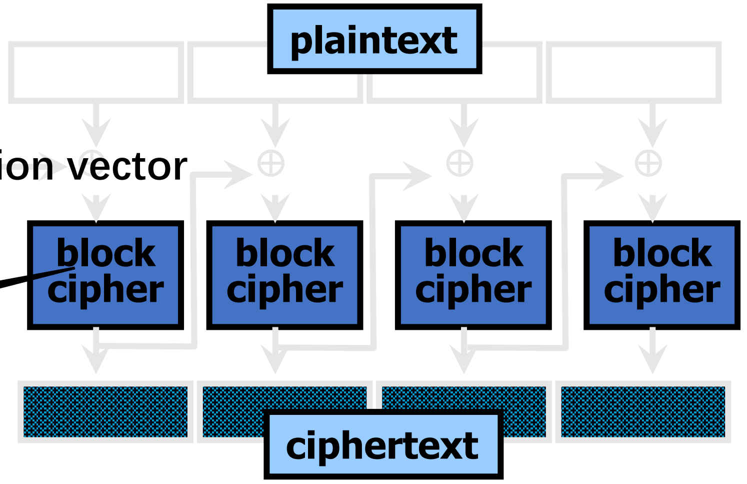


In the figure above, the original text (left), the poorly obfuscated encryption (center), and the well obfuscated encryption (right)

Confusion and proliferation (continued)



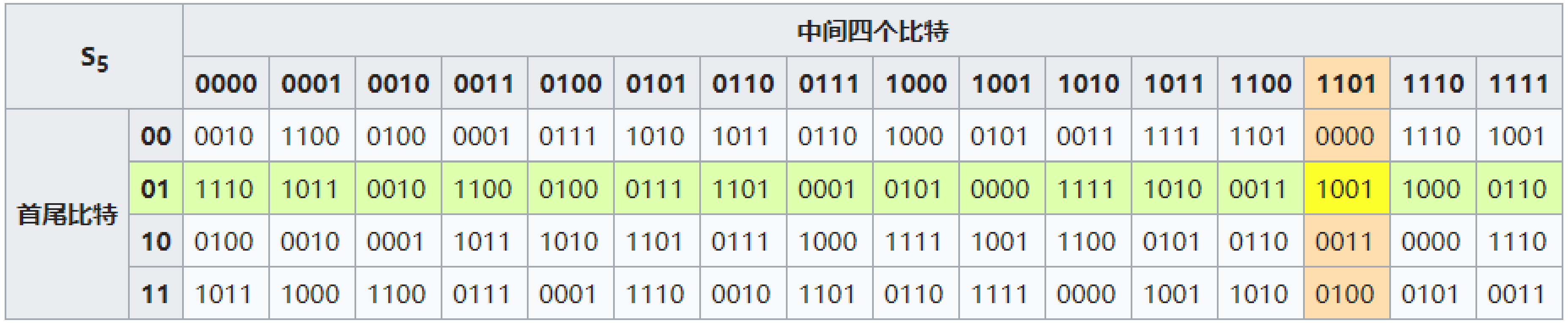
Diffusion-free encryption



cleartext-based grouping
Input and output diffusion encryption

Confusion and proliferation (continued)

- S-box (Substitution-box): in cryptography, the basic structure of a symmetric-key encryption algorithm performing substitution computations. In packet ciphers, they are often used to obscure the relationship between the key and the ciphertext - Shannon's theory of obfuscation.
- Typically, an S-Box accepts a specific number of input bits m and converts them to a specific number of output bits n, where n is not necessarily equal to m. An m × n S-Box can be realized by a lookup table containing 2^m entries with n bits per entry.
- The S-box is usually fixed (e.g., DES and AES encryption algorithms), and the implementation is kept secret, but there are some encryption algorithms for which the S-box



S_5		中间四个比特															
		0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
首尾比特	00	0010	1100	0100	0001	0111	1010	1011	0110	1000	0101	0011	1111	1101	0000	1110	1001
	01	1110	1011	0010	1100	0100	0111	1101	0001	0101	0000	1111	1010	0011	1001	1000	0110
	10	0100	0010	0001	1011	1010	1101	0111	1000	1111	1001	1100	0101	0110	0011	0000	1110
	11	1011	1000	1100	0111	0001	1110	0010	1101	0110	1111	0000	1001	1010	0100	0101	0011

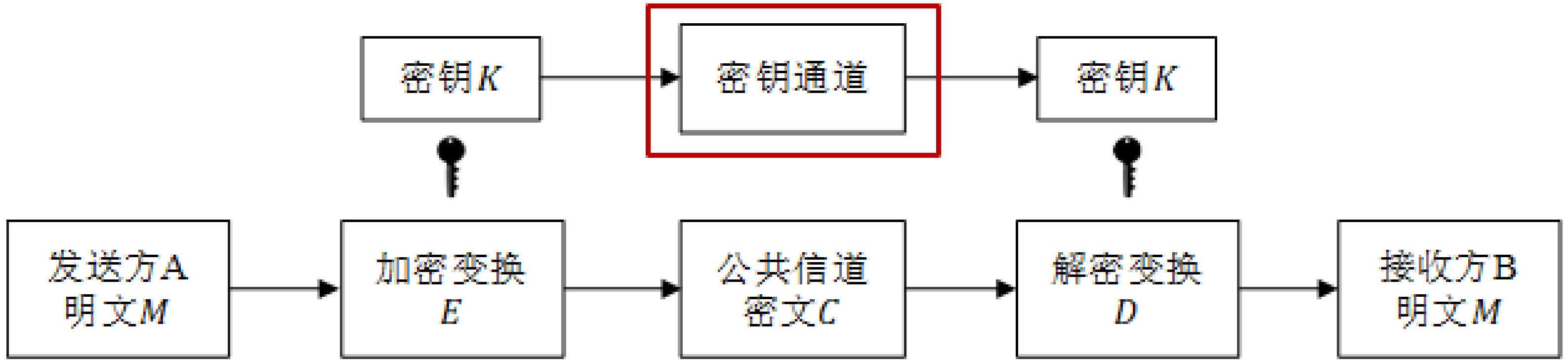
DES 6×4 box, "011011" → "1001"

Private key cipher (symmetric cipher)

3.1 Symmetric Encryption

symmetric encryption

- **Definition**: also known as private/single key encryption. The same key is used for both encryption and decryption. It is the only type of encryption before the birth of public key cryptography in the 1970s, and it is also the more widely used type of encryption.
- communications process



Encryption: $C = E(K, M)$

Decryption: $C = E(K, M)$

Q: What if there is no secure channel?

symmetric encryption

- **Symmetric encryption security is guaranteed by the following two conditions:**
  - ☐ The encryption algorithm is secure. This condition requires that even if an attacker steals a certain number of ciphertexts and corresponding plaintexts, they cannot be deciphered;
  - ☐ The communicating parties obtain each other's keys over a secure channel and store them securely. This condition requires that the key has confidentiality, otherwise all ciphertexts will be deciphered.

What is the key to confidentiality in symmetric encryption?

Confidentiality of keys

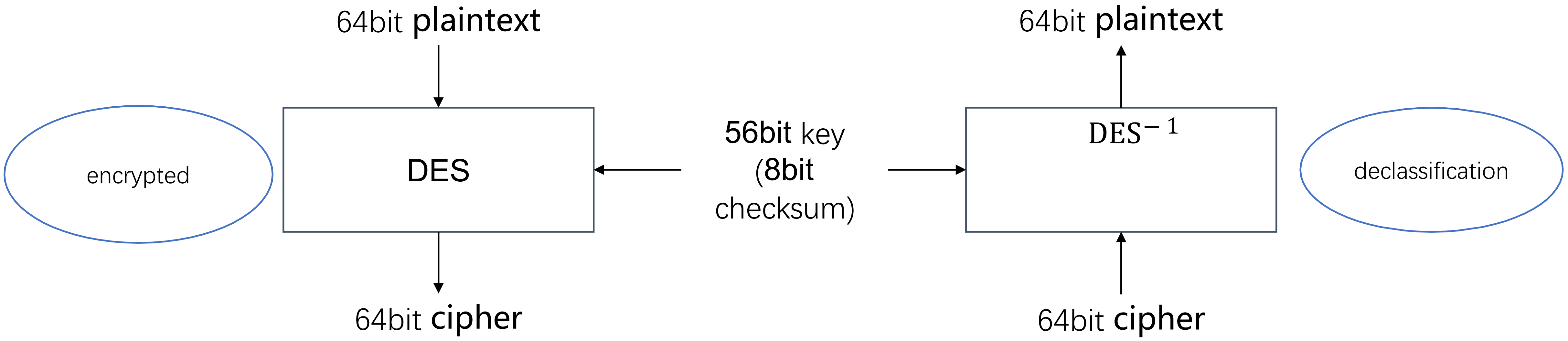
3.1.1 DES

- **Definition**: DES (Data Encryption Standard) is an iterative algorithm that encrypts 64-bit long packets using a 56-bit key.DES was the first publicly available packet encryption algorithm.

- **Characteristics**: The DES encryption process for each packet in the plaintext consists of 16 rounds, and each round is identical to the operation. Each round uses a different subkey, but the subkey is derived from the master key.

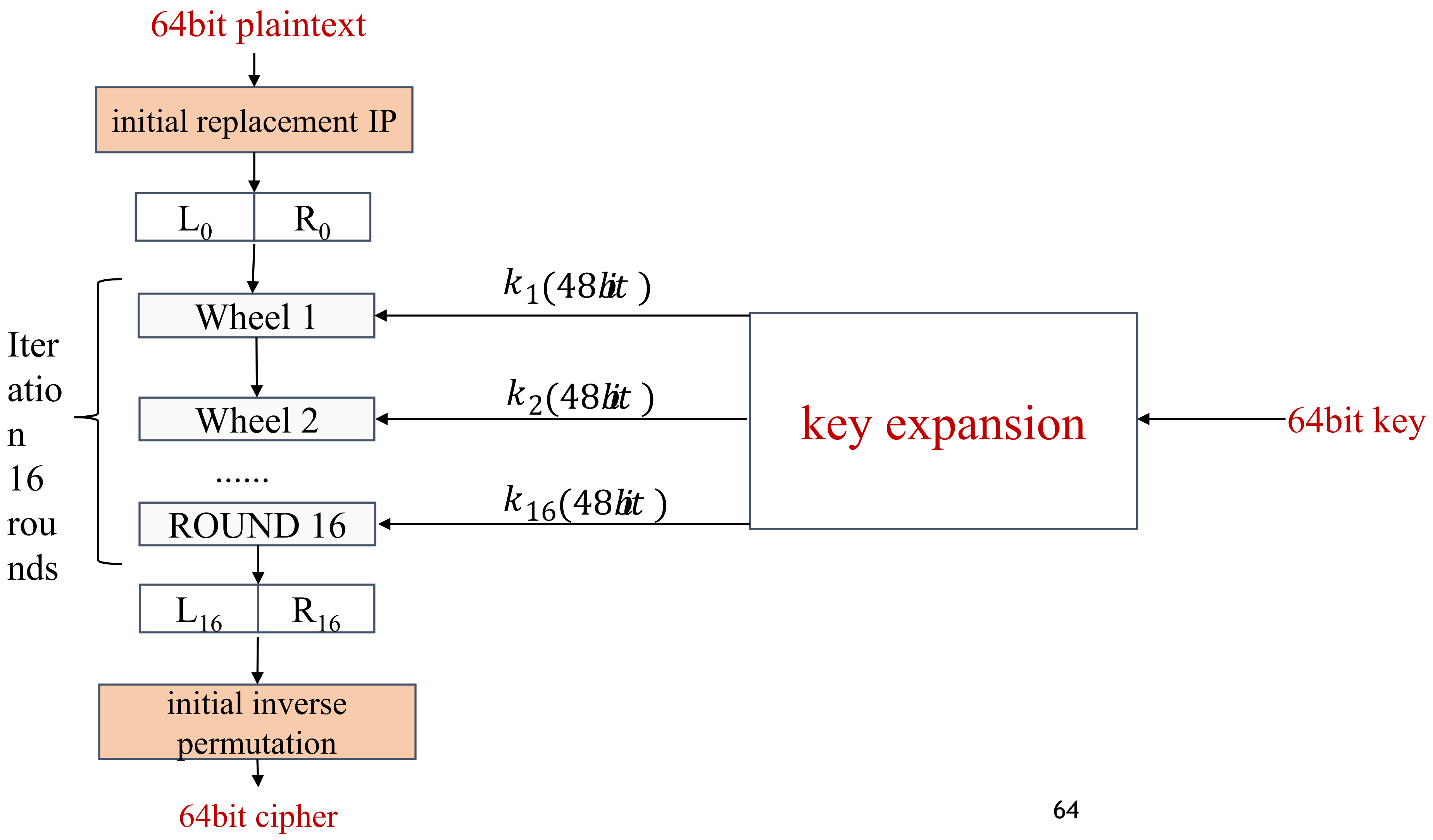
- Historical process: algorithmic recruitment and standardisation
  - ◻In 1972 the National Bureau of Standards (NBS) began a programme to develop computer data protection standards.
  - ◻1973, NBS open call for computer data encryption algorithms
  - ◻In 1977, NBS adopted a modified version of IBM's Lucifer algorithm as the DES

DES encryption diagram



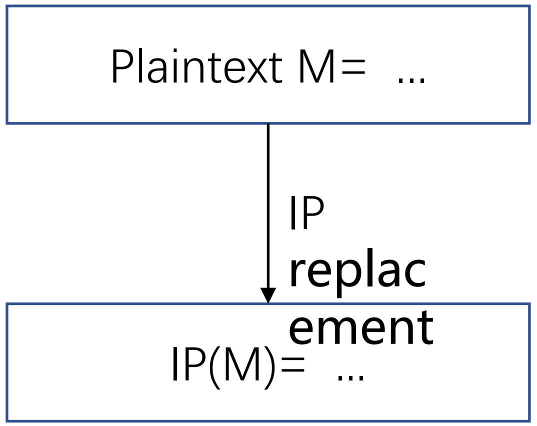
Packet length: 64 bits Key length: 64 bits Valid key length: 56 bits

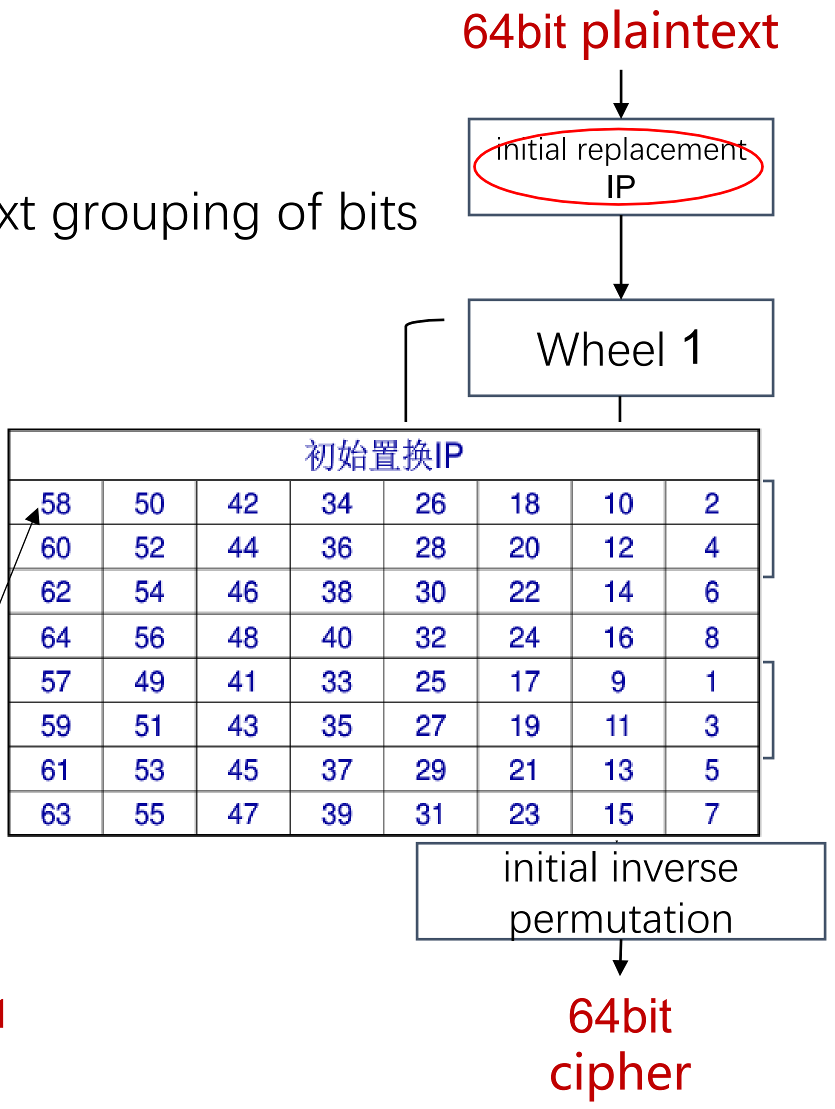
DES encryption process



DES encryption process 1-substitution

- **initial replacement IP**
  - 64-bit plaintext packet replacement
  - The effect is to get a scrambled plaintext grouping of bits





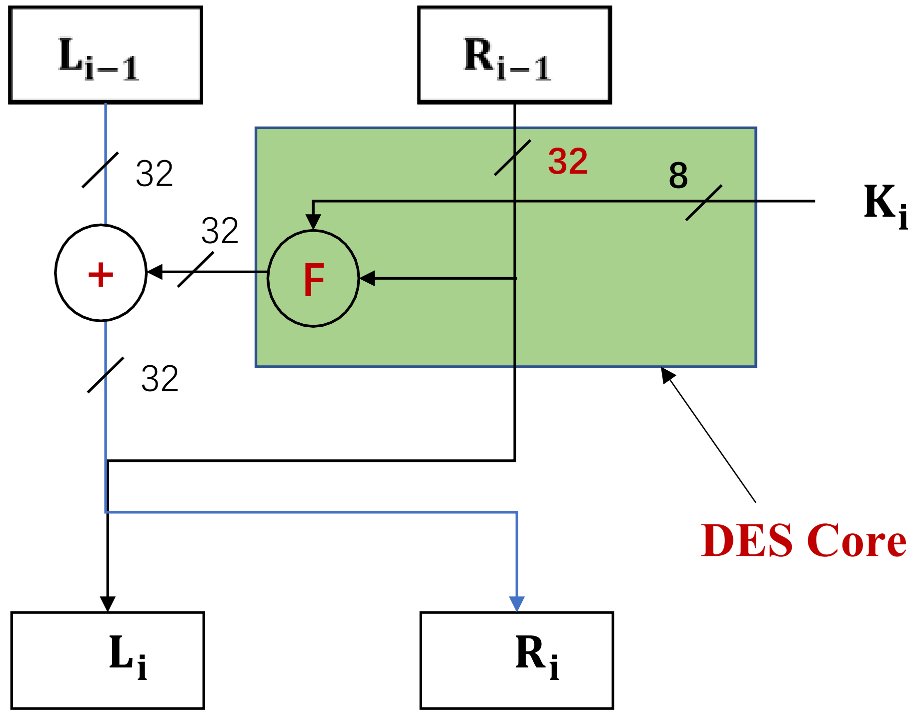
初始置换IP

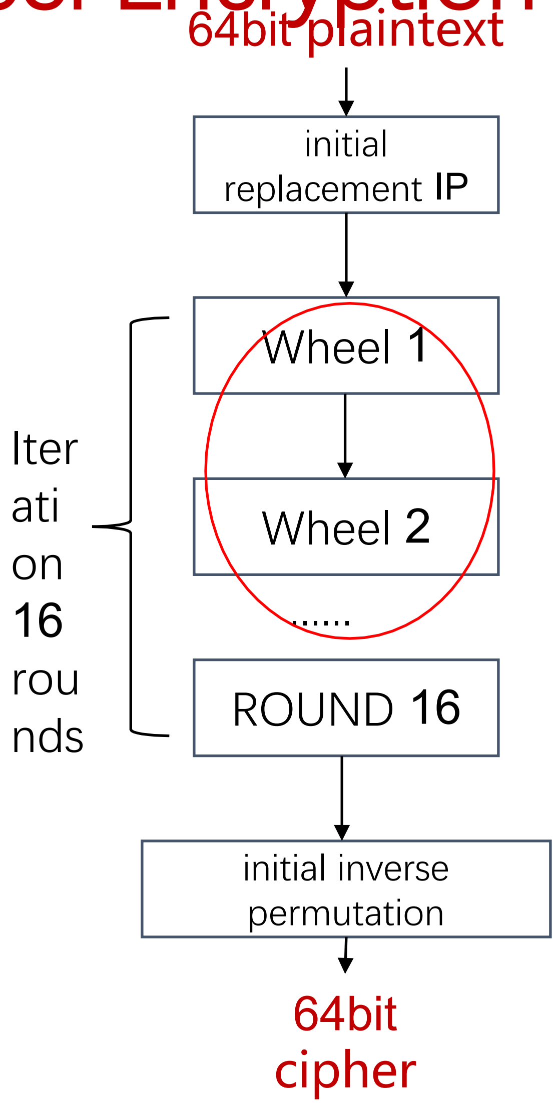
58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

Write the value
of bit 58 to bit 1

DES Encryption Process 2 - In-Wheel Encryption

- **Encryption method based on Feistel structure.**

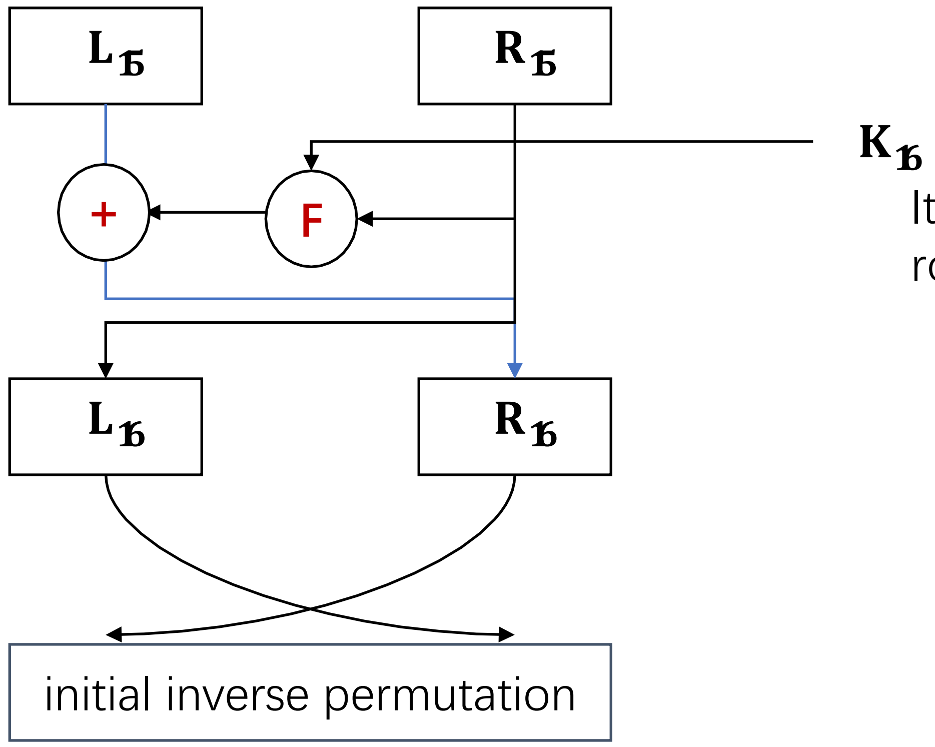


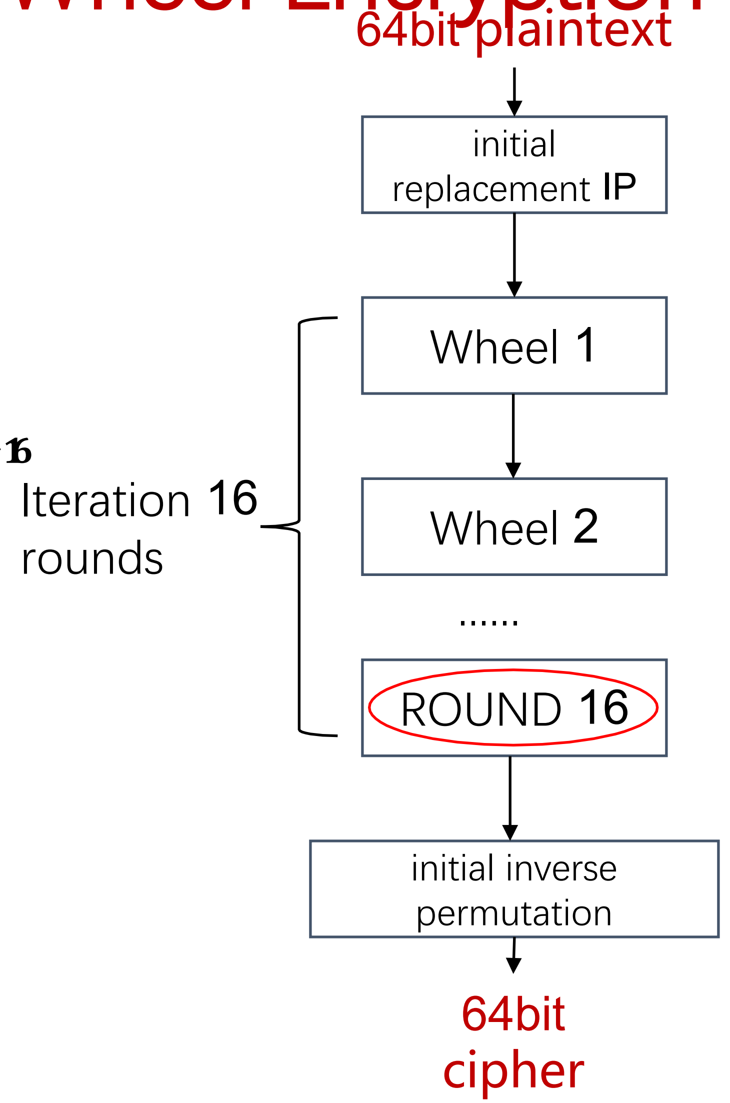


Q: How do 48-bit and 32-bit numbers operate?

DES Encryption Process 2 - In-Wheel Encryption

- **Iteration 16**

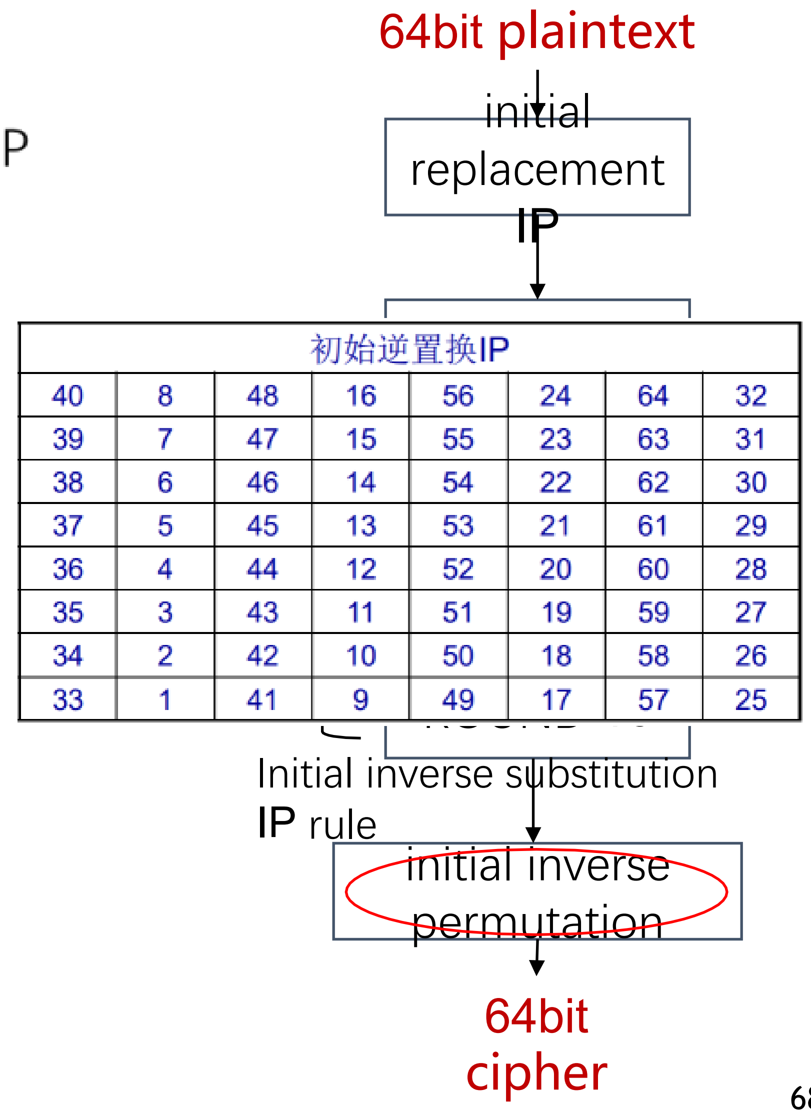




DES Encryption Process 3 - Inverse Substitution

- **initial permutation (math.)**
  - Inversion of initial replacement IP
  - $\mathrm{IP}^{-1}[\mathrm{IP}(\mathrm{M})] = \mathrm{M}$

$$\begin{tikzcd}
\boxed{\text{Plaintext M}=m_1\ m_2\cdots\ m_{64}} \arrow[d, "\text{IP replacement}"] \\
\boxed{\text{IP(M))}=m_{58}\ m_{50}\cdots\ m_7}
\end{tikzcd}$$

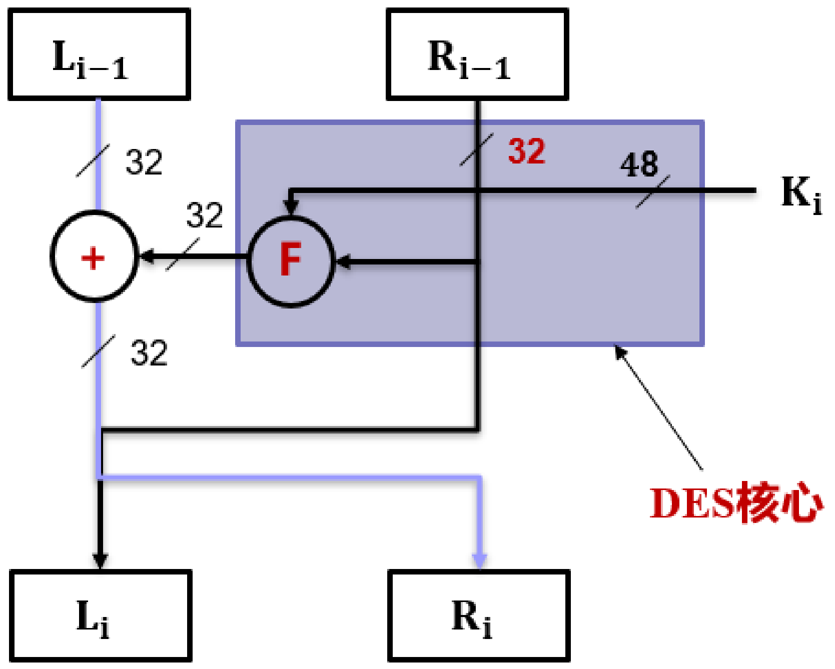


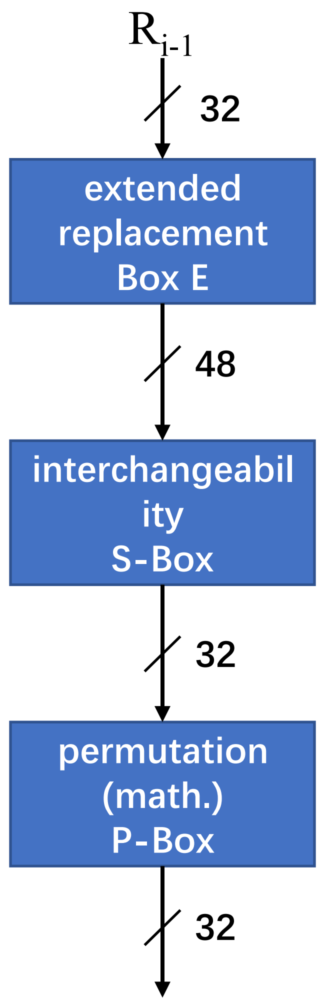
64bit plaintext
initial replacement IP
Initial inverse substitution IP rule
initial inverse permutation
64bit cipher

初始逆置换IP							
40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

The core of DES - F function

- The F function implements obfuscation and diffusion.





F-Function Calculation - Sample Demonstration

- Known:

Explicit text: 30 31 32 33 34 35 36 37$_{16}$

Key: 31 32 33 34 35 35 36 37 38$_{16}$

- request a ciphertext

DES Sample Demo

- **1. Initial IP replacement**

30 31 32 33 34 35 36 37

00110000 00110001 00110010 00110011 00110100 00110101 00110110 00110111

⬇

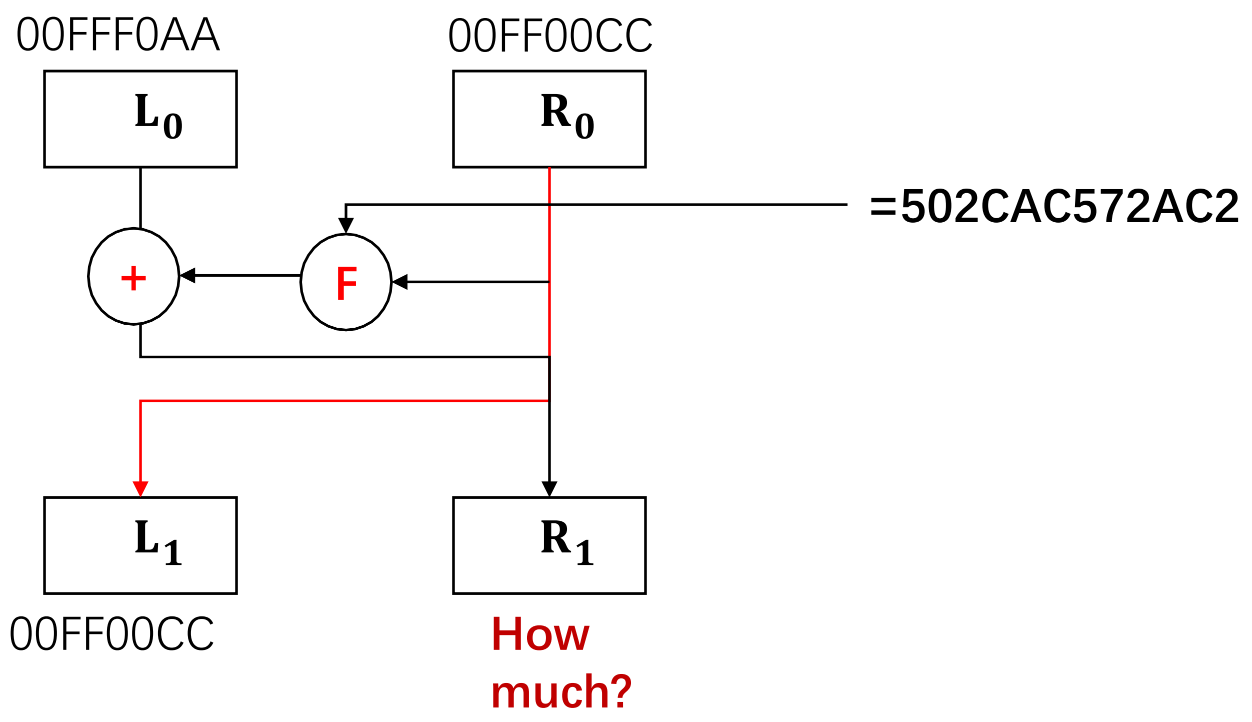
初始置换IP							
58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

00000000 11111111 11110000 10101010 00000000 11111111 00000000 11001100

00 ff f0 aa 00 ff 00 cc

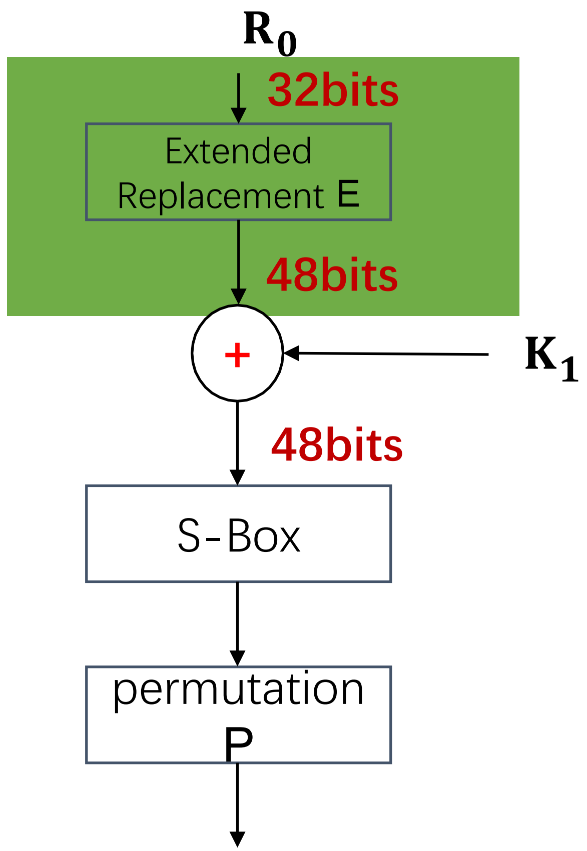
DES Sample Demo

- 2. First round of encryption
  - ☐Assume subkey K_1 =502CAC572AC2$_{16}$



DES Sample Demo

- 3. F-function-extended substitution E



R0=00FF00CC

Input: 0000 0000 1111 1111 0000 0000 0000 1100 1100

Row 1 Row 2 ……

① Assuming an extension of "**0000**", i.e., **5**, **6**, **7**, **8** bits.

Extended Replacement E					
32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	[illegible]	14	15	16	17
[illegible]	[illegible]	[illegible]	19	20	21
[illegible]	[illegible]	[illegible]	23	24	25
[illegible]	[illegible]	[illegible]	27	28	29
28	29	30	31	32	1

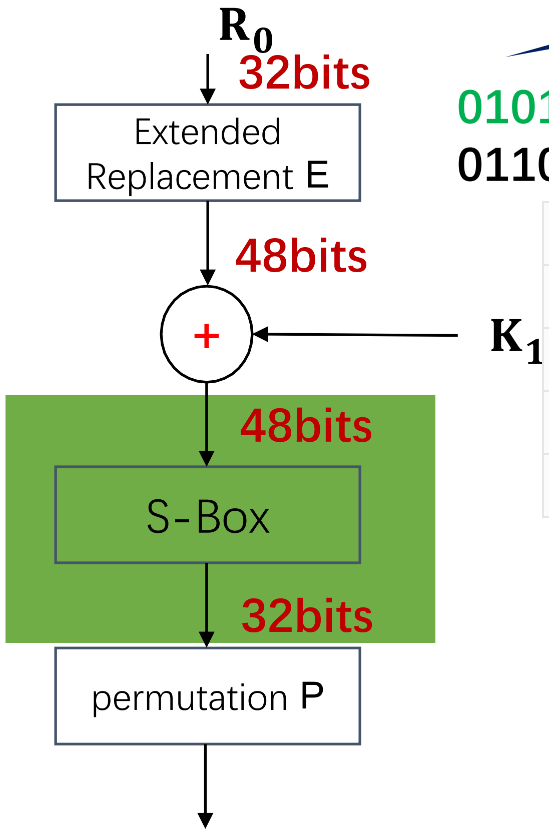
② **5**, **6**, **7** and **8** supplement the value of the **4th** and **9th** digits at the beginning and end, respectively.

Output: 000000 000001 011111 111110 100000 000001 011001 011000

DES Sample Demo

- 4. F-function-S box

Assuming that "010100" is shortened, and "1010" (decimal "10") and 00 (decimal "0") are used as indexes to look up the table, the resulting value is the scaling result. ") as index look up the table, the value obtained is the scaling result.



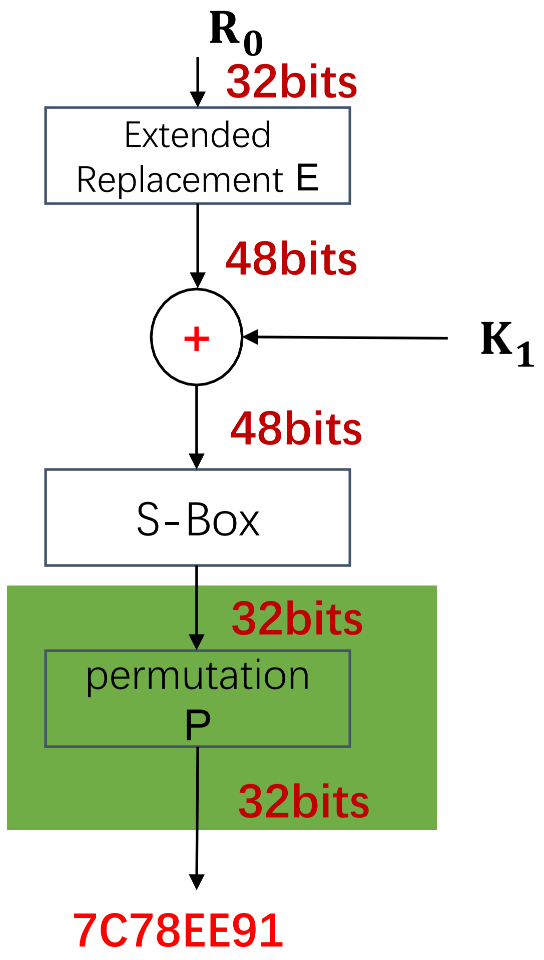
010100 000011 101101 010010 110101 110011 110010 011010

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

0110 1111 0001 1010 0011 1011 1100 1001

DES Sample Demo

- 5. F-function-replacement P



0110 1111 0001 1010 0011 1011 1100 1001

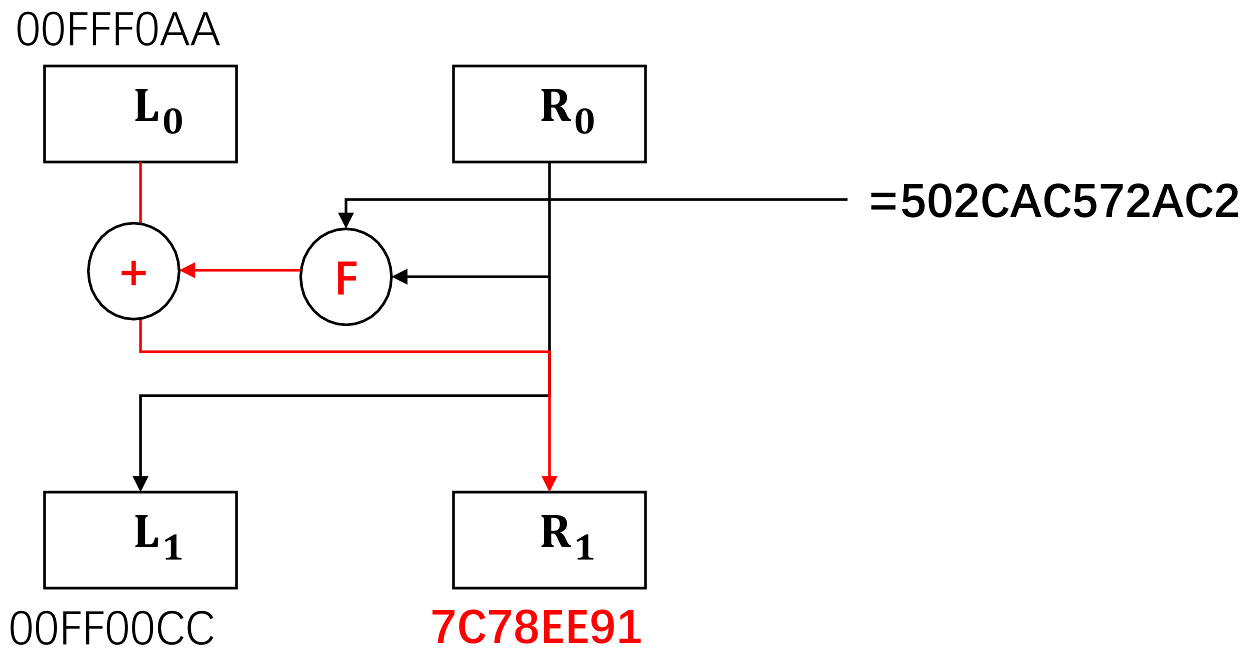
permutation P			
16	7	20	21
29	12	28	17
1	15	23	26
5	18	31	10
2	8	24	14
32	27	3	9
19	13	30	6
22	11	4	25

0011 1100 0111 1000 1110 1110 1001 0001

7C78EE91

DES Sample Demo

- First round of encryption
  - Assume subkey K_1 =502CAC572AC2$_{16}$



DES Sample Demo

- **6. Initial inverse substitution**

d4 16 8a a1 33 f6 ad 45

初始逆置换IP							
40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

8b b4 7a 0c f0 a9 62 6d

reflections

- Feistel encrypts (decrypts) only half of the input bits in each round, and the F function encrypts only the left half, not the right half.

- One of the advantages of DES is that the decryption process is exactly the same as the encryption process, except that the ith round of the decryption process requires the use of the 16th-i round to the key

How secure is DES?

- Analytical attack: exploits correlation between keys, virtually nothing in practice
- Exhaustive attack: $2^{56} = 7.2*10^{16}$
  - In 1997 Diffie and Hellman proposed building a chip that would test 106 keys per second and search the entire key space in a day for about $20 million
  - Deep Crack, $250,000, average 15 days
  - COPACOBANA: $10,000 for an average of 7 days

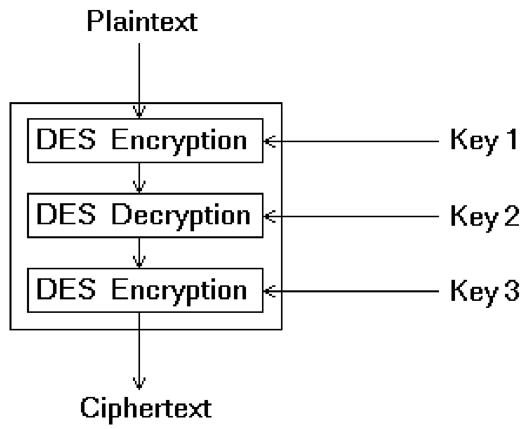
DES security (continued)

- **Key Search and Supercomputing**
  - DES's 56-bit short key faces a harsh reality: the supercomputing power of the Internet of Nations Internet
  - On 28 January 1997, the RSA Data Security Corporation of the United States launched the "**Key Challenge**" competition on the Internet. A programmer named Rocke Verser designed an exhaustive key search program that ran in segments over the Internet, and thousands of volunteers joined in, successfully finding the key on 17 June 1997, when the key was found. On 17 June 1997, the key was successfully found.

- **How is it reinforced?**

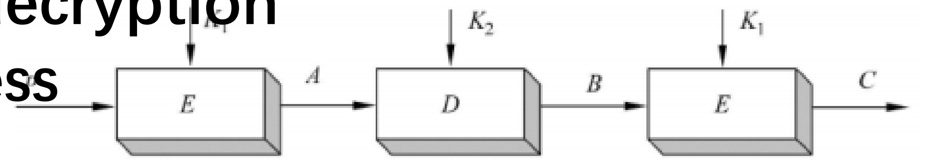
3.1.2 Improvements to DES - 3DES

- 3DES: Consists of three consecutive DES encryptions, also called Triple DES.

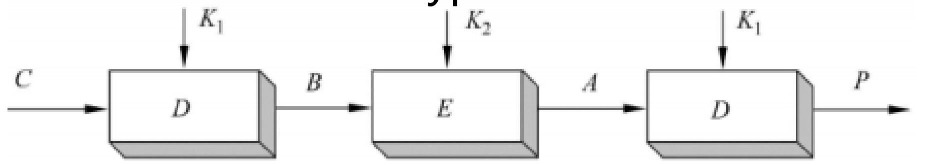


3DES

- **3DES encryption and decryption process**



encrypted



declassification

Q: What if $k_1 = k_2 = k_3$? Is it necessary?

- **3DES Security**
  - If are not equal to each other, the key length 56*3=168 brute force decryption takes $5.8 * 10^{29}$ years!
  - 3DES is still sufficiently secure for the time being

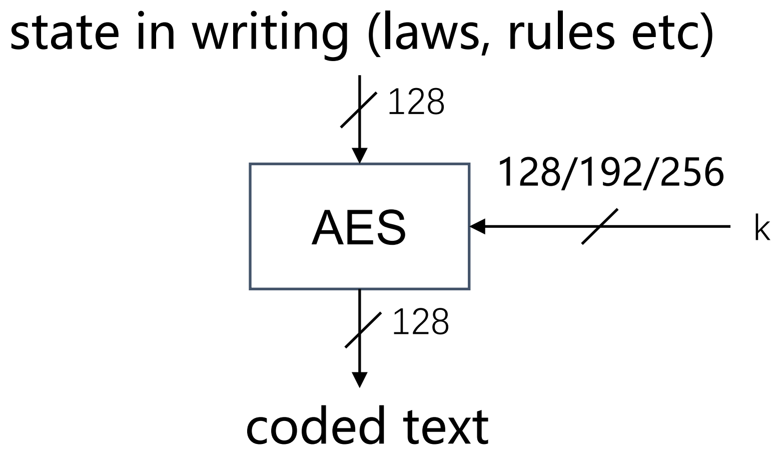
Problems with 3DES

- DES for legacy systems, 3DES for new systems

- Problems faced:
  - Implementation efficiency: 3DES implementation time is 3 times that of DES;
  - Security: Smaller grouping, only 64 bits.

3.1.3 AES

- **Advanced** Encryption Standard (Advanced Encryption Standard): 128-bit grouping, support for three key lengths (128, 192, 256) and efficient hardware and software implementation. It is the most widely used symmetric algorithm.



- In 1997, the National Institute of Standards and Technology (NIST) issued a call for new high-level encryption standards, and in 2001, Rijndael, a packet cipher submitted by two young Belgian scientists, Daemen and Rijmen, became the new high-level encryption standard. (Rijndael is pronounced Rain Doll.)

Daemen and Rijmen

http://www.esat.kuleuven.ac.be/~rijmen/rijndael/

Table 5.1 NIST Evaluation Criteria for AES (September 12, 1997) (page 1 of 2)

SECURITY

- **Actual security:** compared to other submitted algorithms (at the same key and block size).
- **Randomness:** The extent to which the algorithm output is indistinguishable from a random permutation on the input block.
- **Soundness:** of the mathematical basis for the algorithm's security.
- **Other security factors:** raised by the public during the evaluation process, including any attacks which demonstrate that the actual security of the algorithm is less than the strength claimed by the submitter.

COST

- **Licensing requirements:** NIST intends that when the AES is issued, the algorithm(s) specified in the AES shall be available on a worldwide, non-exclusive, royalty-free basis.
- **Computational efficiency:** The evaluation of computational efficiency will be applicable to both hardware and software implementations. Round 1 analysis by NIST will focus primarily on software implementations and specifically on one key-block size combination (128-128); more attention will be paid to hardware implementations and other supported key-block size combinations during Round 2 analysis. Computational efficiency essentially refers to the speed of the algorithm. Public comments on each algorithm's efficiency (particularly for various platforms and applications) will also be taken into consideration by NIST.
- **Memory requirements:** The memory required to implement a candidate algorithm--for both hardware and software implementations of the algorithm--will also be considered during the evaluation process. Round 1 analysis by NIST will focus primarily on software implementations; more attention will be paid to hardware implementations during Round 2. Memory requirements will include such factors as gate counts for hardware implementations, and code size and RAM requirements for software implementations.

Table 5.1 NIST Evaluation Criteria for AES (September 12, 1997) (page 2 of 2)

ALGORITHM AND IMPLEMENTATION CHARACTERISTICS

•**Flexibility:** Candidate algorithms with greater flexibility will meet the needs of more users than less flexible ones, and therefore, inter alia, are preferable. However, some extremes of functionality are of little practical application (e.g., extremely short key lengths); for those cases, preference will not be given. Some examples of flexibility may include (but are not limited to) the following:

- **a.** The algorithm can accommodate additional key- and block-sizes (e.g., 64-bit block sizes, key sizes other than those specified in the Minimum Acceptability Requirements section, [e.g., keys between 128 and 256 that are multiples of 32 bits, etc.])
- **b.** The algorithm can be implemented securely and efficiently in a wide variety of platforms and applications (e.g., 8-bit processors, ATM networks, voice & satellite communications, HDTV, B-ISDN, etc.).
- **c.** The algorithm can be implemented as a stream cipher, message authentication code (MAC) generator, pseudorandom number generator, hashing algorithm, etc.

•**Hardware and software suitability:** A candidate algorithm shall not be restrictive in the sense that it can only be implemented in hardware. If one can also implement the algorithm efficiently in firmware, then this will be an advantage in the area of flexibility.

•**Simplicity:** A candidate algorithm shall be judged according to relative simplicity of design.

AES assessment guidelines

- General Security
  - Public security analyses dependent on the cryptographic community
- software implementation
  - Software execution speed, cross-platform execution capability and speed change when key length is changed.
- Constrained space environment
  - Applications in e.g. smart cards
- hardware implementation
  - Hardware implementation can increase execution speed or reduce code length
- Defending against cryptanalysis attacks
- Key Flexibility
  - Ability to quickly change key lengths
- Additional versatility and flexibility
- Potential for Instruction-Level Parallel Execution

Table 5.2 Final NIST Evaluation of Rijndael (October 2, 2000) (page 1 of 2)

General Security

Rijndael has no known security attacks. Rijndael uses S-boxes as nonlinear components. Rijndael appears to have an adequate security margin, but has received some criticism suggesting that its mathematical structure may lead to attacks. On the other hand, the simple structure may have facilitated its security analysis during the timeframe of the AES development process.

Software Implementations

Rijndael performs encryption and decryption very well across a variety of platforms, including 8-bit and 64-bit platforms, and DSPs. However, there is a decrease in performance with the higher key sizes because of the increased number of rounds that are performed. Rijndael's high inherent parallelism facilitates the efficient use of processor resources, resulting in very good software performance even when implemented in a mode not capable of interleaving. Rijndael's key setup time is fast.

Restricted-Space Environments

In general, Rijndael is very well suited for restricted-space environments where either encryption or decryption is implemented (but not both). It has very low RAM and ROM requirements. A drawback is that ROM requirements will increase if both encryption and decryption are implemented simultaneously, although it appears to remain suitable for these environments. The key schedule for decryption is separate from encryption.

Hardware Implementations

Rijndael has the highest throughput of any of the finalists for feedback modes and second highest for non-feedback modes. For the 192 and 256-bit key sizes, throughput falls in standard and unrolled implementations because of the additional number of rounds. For fully pipelined implementations, the area requirement increases, but the throughput is unaffected.

Table 5.2 Final NIST Evaluation of Rijndael (October 2, 2000) (page 2 of 2)

Attacks on Implementations

The operations used by Rijndael are among the easiest to defend against power and timing attacks. The use of masking techniques to provide Rijndael with some defense against these attacks does not cause significant performance degradation relative to the other finalists, and its RAM requirement remains reasonable. Rijndael appears to gain a major speed advantage over its competitors when such protections are considered.

Encryption vs. Decryption

The encryption and decryption functions in Rijndael differ. One FPGA study reports that the implementation of both encryption and decryption takes about 60% more space than the implementation of encryption alone. Rijndael's speed does not vary significantly between encryption and decryption, although the key setup performance is slower for decryption than for encryption.

Key Agility

Rijndael supports on-the-fly subkey computation for encryption. Rijndael requires a one-time execution of the key schedule to generate all subkeys prior to the first decryption with a specific key. This places a slight resource burden on the key agility of Rijndael.

Other Versatility and Flexibility

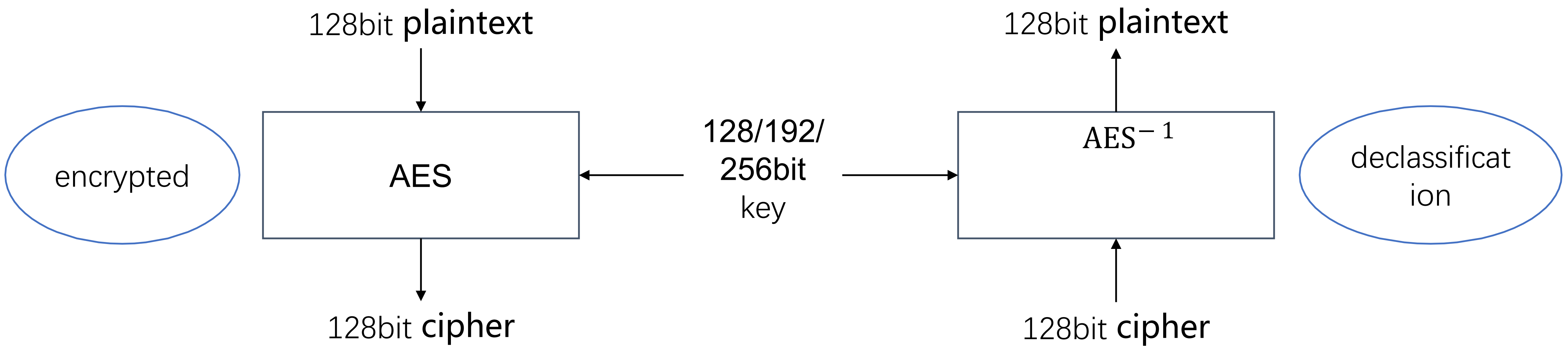
Rijndael fully supports block sizes and key sizes of 128 bits, 192 bits and 256 bits, in any combination. In principle, the Rijndael structure can accommodate any block sizes and key sizes that are multiples of 32, as well as changes in the number of rounds that are specified.

Potential for Instruction-Level Parallelism

Rijndael has an excellent potential for parallelism for a single block encryption.

3.1.3 AES

- **AES parameters**

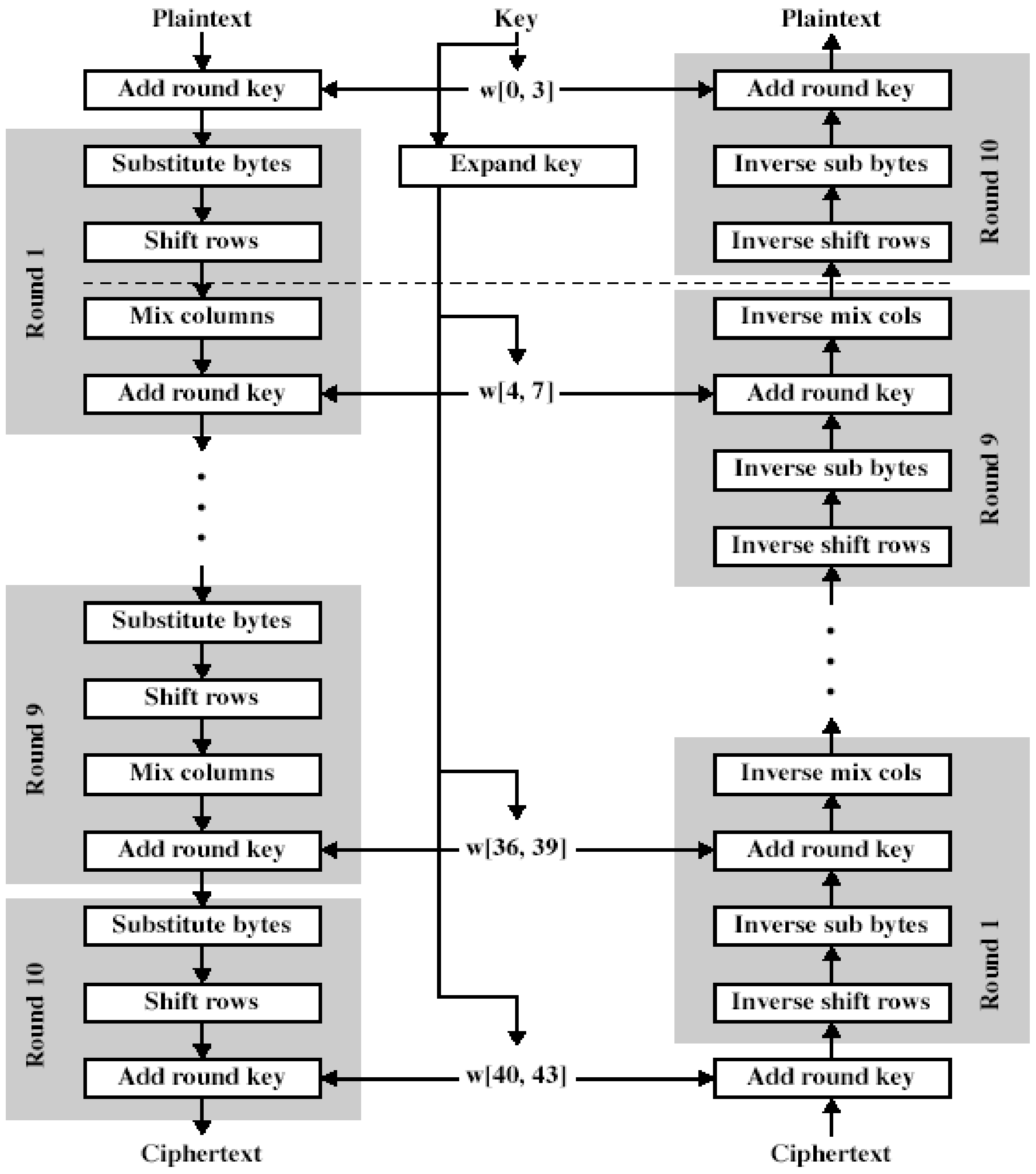


	AES-128	AES-192	AES-256
Key Length	128	192	256
Number of iteration rounds	10	12	14

0

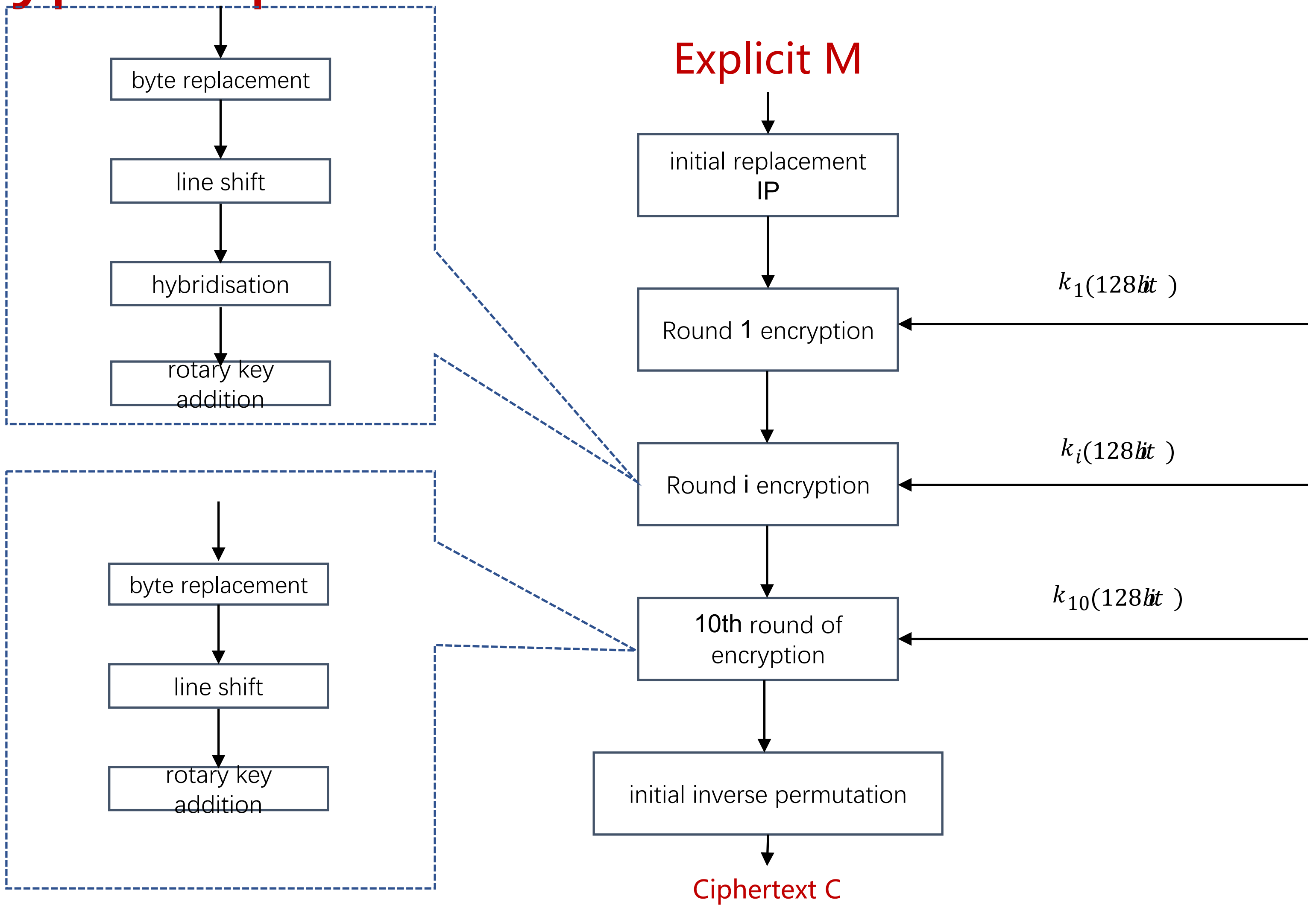
Structure of AES

- Non-Feistel structures
- The key is expanded into an array of 44 32-bit words, four of which are used per round.
- One confusion and three substitutions
  - Byte substitution: byte-by-byte substitution in groups is done with an S-box.
  - Row shifting: a simple substitution
  - Column confusion: substitution using arithmetic properties on Galois domains
  - Round key addition: bitwise XOR using the current packet and part of the extended key
- The algorithm has a simple structure and uses the key only in the round encryption phase
- Wheel key plus with the other three obfuscated alternates provide security
- Each stage is reversible and decryption uses the extended key in reverse order, but the algorithms are different
- Once the four stages are inverted, it can be shown that the decryption function can recover the plaintext
- The final round of both the encryption and decryption process consists of only three stages



(a) Encryption (b) Decryption

AES encryption process



AES round of encryption process

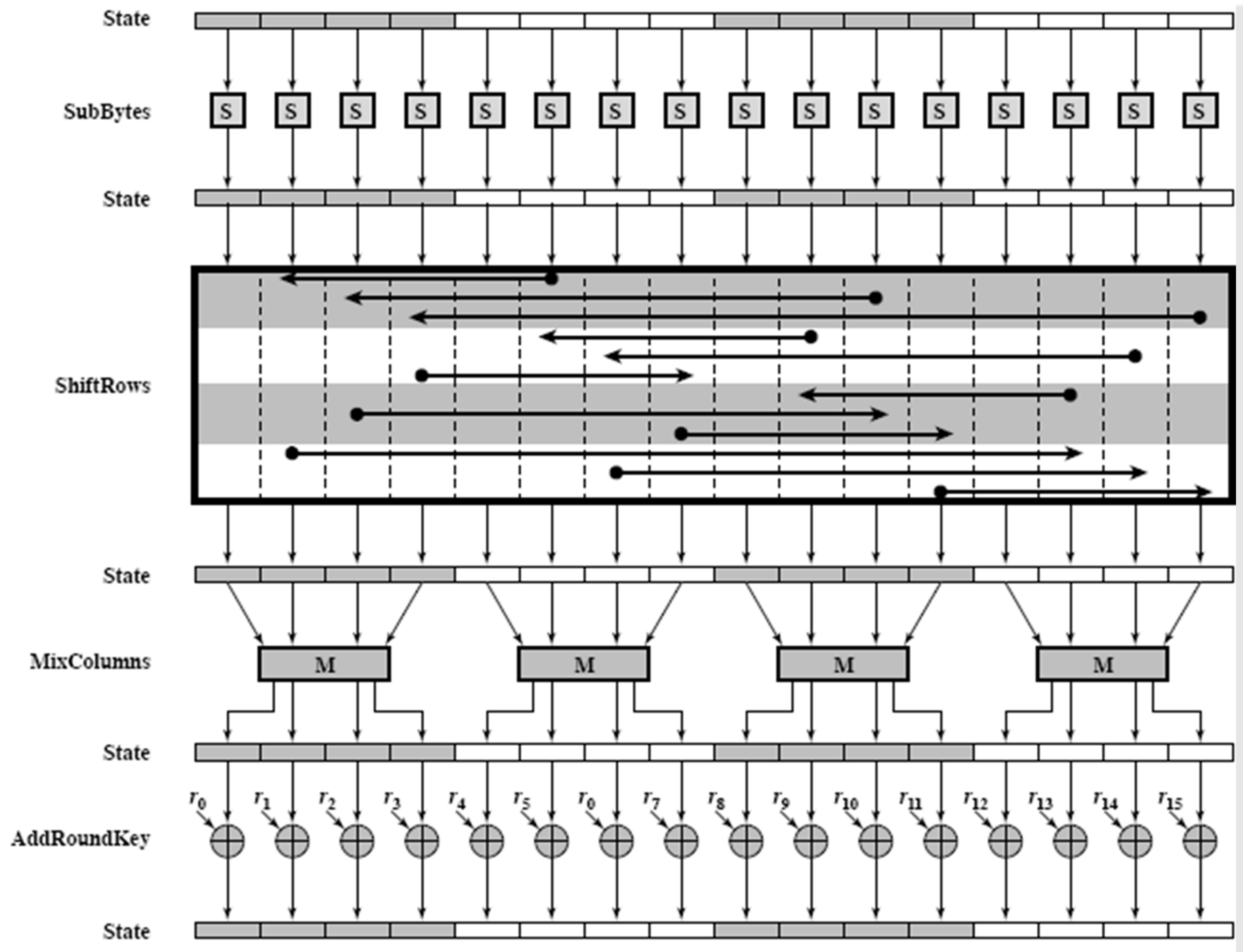
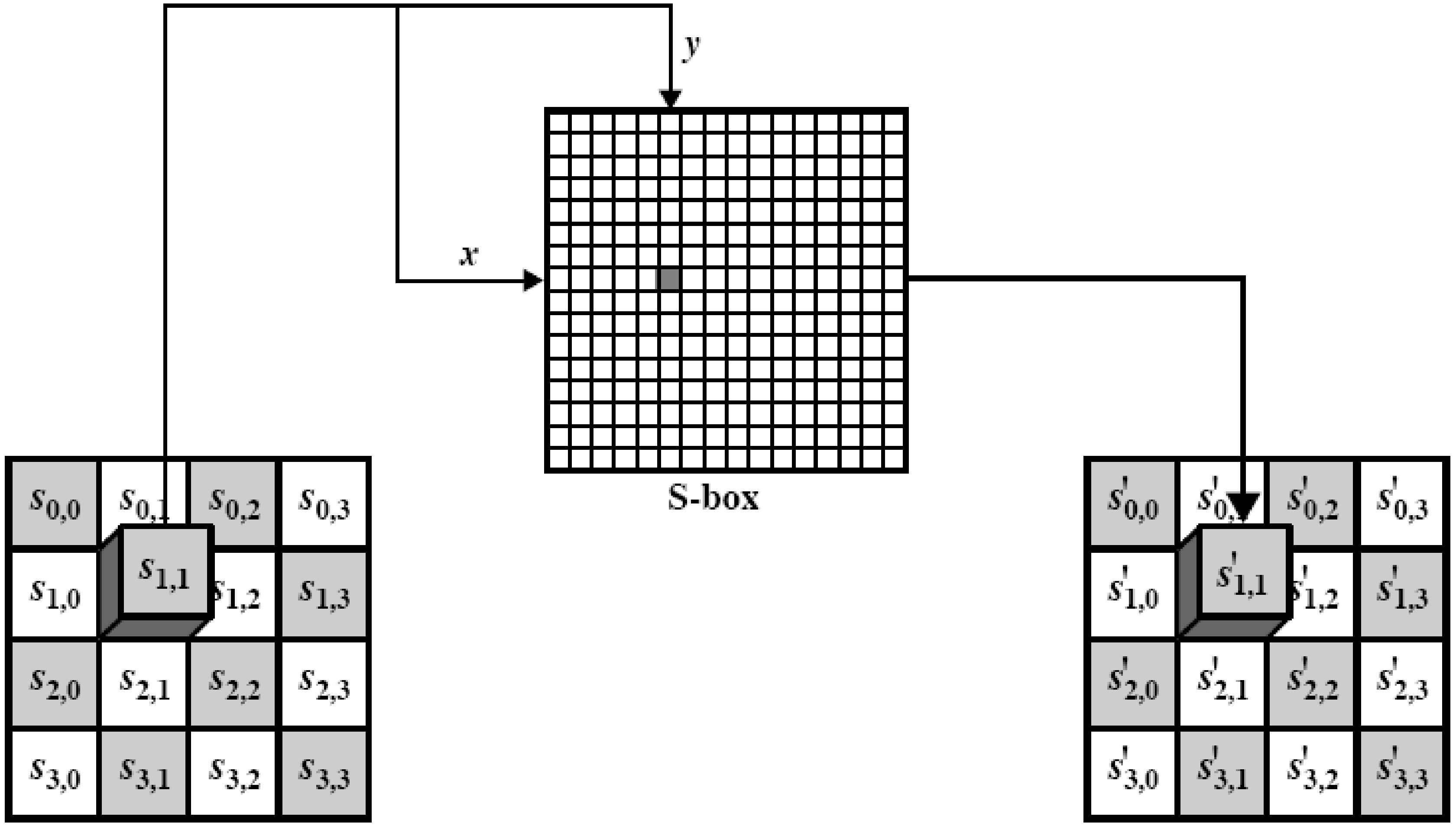

State
SubBytes
S
State
ShiftRows
State
MixColumns
M
State
AddRoundKey
r_0 r_1 r_2 r_3 r_4 r_5 r_6 r_7 r_8 r_9 r_{10} r_{11} r_{12} r_{13} r_{14} r_{15}
State

Figure 5.3 AES Encryption Round

The four phases of AES

- Substitute bytes
- ShiftRows
- Column MixColumns
- AddRoundKey

byte substitution



(a) Substitute byte transformation

S-box for AES

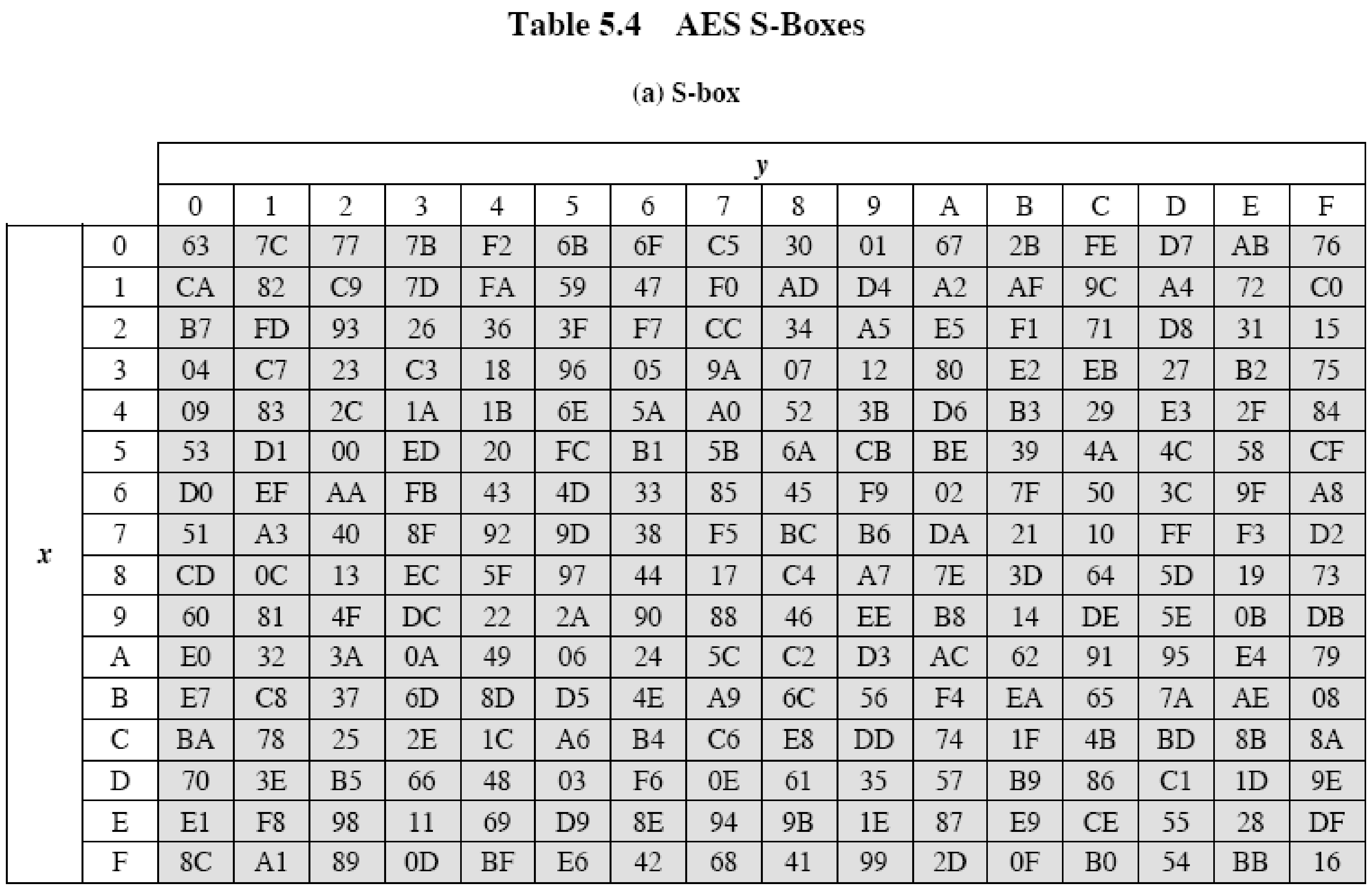


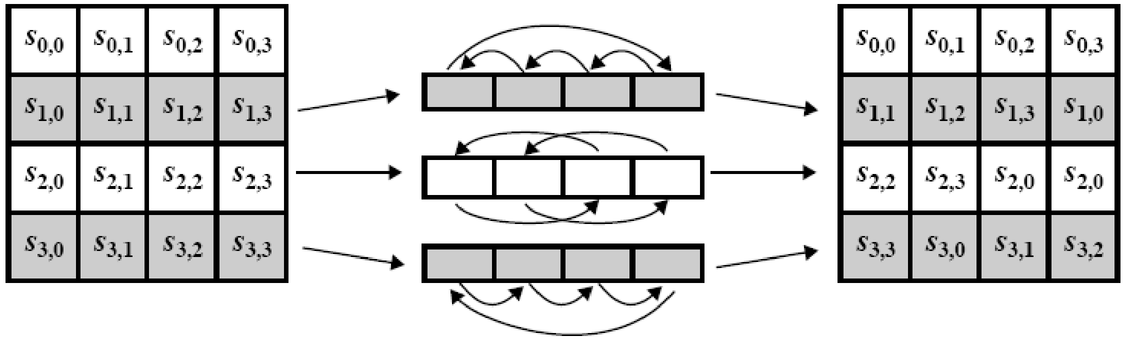
Table 5.4 AES S-Boxes

(a) S-box

		y															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
	0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
	1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
	2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
	3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
	4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
	5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
	6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
x	7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
	8	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
	9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
	A	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
	B	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
	C	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
	D	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
	E	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
	F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

line shift

- Movement rules:
  - First line unchanged
  - Second row shifted left by 1 byte
  - Third row shifted 2 bytes to the left
  - Move 3 bytes to the left in the fourth row
- Substitution between columns is implemented



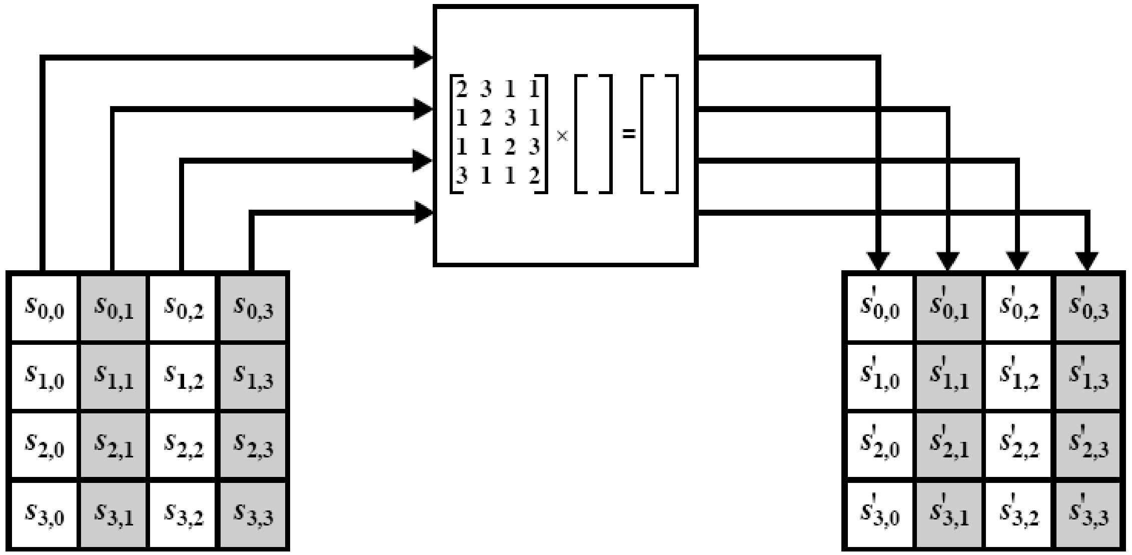
$s_{0,0}$ $s_{0,1}$ $s_{0,2}$ $s_{0,3}$
$s_{1,0}$ $s_{1,1}$ $s_{1,2}$ $s_{1,3}$
$s_{2,0}$ $s_{2,1}$ $s_{2,2}$ $s_{2,3}$
$s_{3,0}$ $s_{3,1}$ $s_{3,2}$ $s_{3,3}$
$s_{0,0}$ $s_{0,1}$ $s_{0,2}$ $s_{0,3}$
$s_{1,1}$ $s_{1,2}$ $s_{1,3}$ $s_{1,0}$
$s_{2,2}$ $s_{2,3}$ $s_{2,0}$ $s_{2,0}$
$s_{3,3}$ $s_{3,0}$ $s_{3,1}$ $s_{3,2}$

(a) Shift row transformation

confuse one thing with another

- Each column is treated separately
- Each byte in each column is replaced with the result that all 4 bytes in this column are related.
- Based on Galois domain operations.

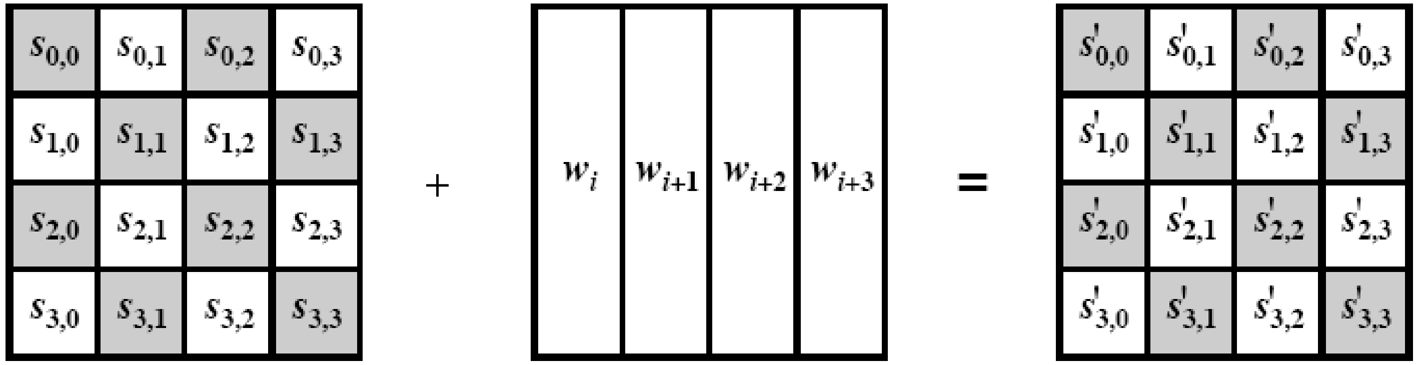
$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{bmatrix} = \begin{bmatrix} s'_{0,0} & s'_{0,1} & s'_{0,2} & s'_{0,3} \\ s'_{1,0} & s'_{1,1} & s'_{1,2} & s'_{1,3} \\ s'_{2,0} & s'_{2,1} & s'_{2,2} & s'_{2,3} \\ s'_{3,0} & s'_{3,1} & s'_{3,2} & s'_{3,3} \end{bmatrix}$$


2 3 1 1
1 2 3 1
1 1 2 3
3 1 1 2
×
=
$s_{0,0}$ $s_{0,1}$ $s_{0,2}$ $s_{0,3}$
$s_{1,0}$ $s_{1,1}$ $s_{1,2}$ $s_{1,3}$
$s_{2,0}$ $s_{2,1}$ $s_{2,2}$ $s_{2,3}$
$s_{3,0}$ $s_{3,1}$ $s_{3,2}$ $s_{3,3}$
$s'_{0,0}$ $s'_{0,1}$ $s'_{0,2}$ $s'_{0,3}$
$s'_{1,0}$ $s'_{1,1}$ $s'_{1,2}$ $s'_{1,3}$
$s'_{2,0}$ $s'_{2,1}$ $s'_{2,2}$ $s'_{2,3}$
$s'_{3,0}$ $s'_{3,1}$ $s'_{3,2}$ $s'_{3,3}$

(b) Mix column transformation

(math.) wheel key transformation

- Dissimilarise the 16-byte current state matrix with a subkey of length 16 bytes.



$s_{0,0}$ $s_{0,1}$ $s_{0,2}$ $s_{0,3}$
$s_{1,0}$ $s_{1,1}$ $s_{1,2}$ $s_{1,3}$
$s_{2,0}$ $s_{2,1}$ $s_{2,2}$ $s_{2,3}$
$s_{3,0}$ $s_{3,1}$ $s_{3,2}$ $s_{3,3}$
+
w_i w_{i+1} w_{i+2} w_{i+3}
=
$s'_{0,0}$ $s'_{0,1}$ $s'_{0,2}$ $s'_{0,3}$
$s'_{1,0}$ $s'_{1,1}$ $s'_{1,2}$ $s'_{1,3}$
$s'_{2,0}$ $s'_{2,1}$ $s'_{2,2}$ $s'_{2,3}$
$s'_{3,0}$ $s'_{3,1}$ $s'_{3,2}$ $s'_{3,3}$

(b) Add Round Key Transformation

AES encryption example

- Known:

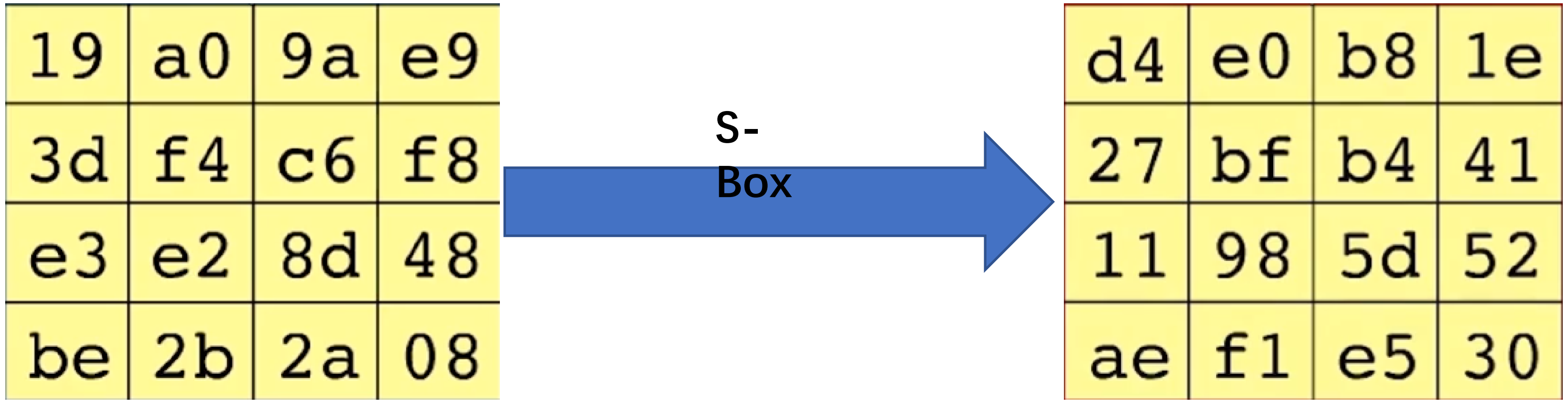
Plaintext: 32 43 f6 a8 88 5a 30 8d 31 31 98 a2 e0 37 07 34

Key: 2b 7e 15 16 28 ae d2 a6 ab f7 15 88 09 cf 4f 3c

- computed ciphertext

AES encryption example (for reference only)

- 1. Byte substitution

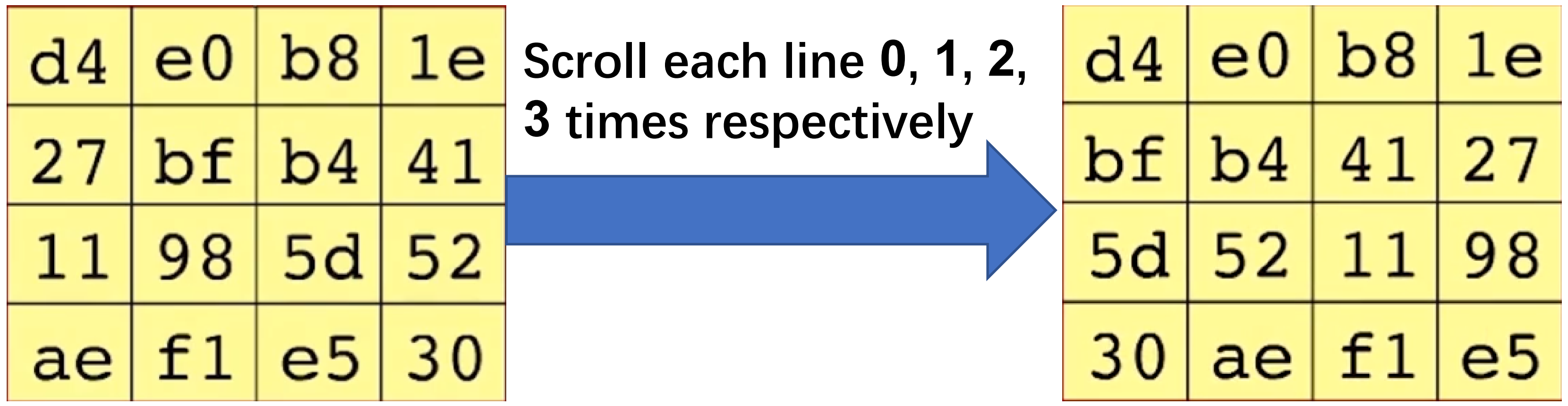


19	a0	9a	e9
3d	f4	c6	f8
e3	e2	8d	48
be	2b	2a	08

d4	e0	b8	1e
27	bf	b4	41
11	98	5d	52
ae	f1	e5	30

AES encryption example

- 2. Row shifting



d4	e0	b8	1e
27	bf	b4	41
11	98	5d	52
ae	f1	e5	30

Scroll each line **0**, **1**, **2**, **3** times respectively

d4	e0	b8	1e
bf	b4	41	27
5d	52	11	98
30	ae	f1	e5

AES encryption example

- **3. Column mixing**

d4	e0	b8	1e
bf	b4	41	27
5d	52	11	98
30	ae	f1	e5

$$\rightarrow \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \bullet \begin{bmatrix} d4 \\ bf \\ 5d \\ 30 \end{bmatrix} = \begin{bmatrix} 04 \\ 66 \\ 81 \\ e5 \end{bmatrix}$$

04	e0	48	28
66	cb	f8	06
81	19	d3	26
e5	9a	7a	4c

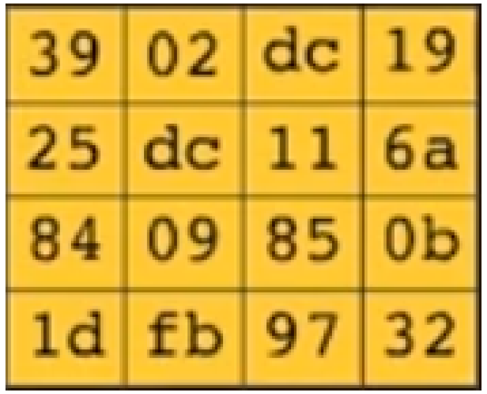
AES encryption example

- 4. Wheel key overlay

$$\begin{array}{|c|c|c|c|}\hline 04 & e0 & 48 & 28 \\ \hline 66 & cb & f8 & 06 \\ \hline 81 & 19 & d3 & 26 \\ \hline e5 & 9a & 7a & 4c \\ \hline \end{array} \oplus \begin{array}{|c|c|c|c|}\hline a0 & 88 & 23 & 2a \\ \hline fa & 54 & a3 & 6c \\ \hline fe & 2c & 39 & 76 \\ \hline 17 & b1 & 39 & 05 \\ \hline \end{array} = \begin{array}{|c|c|c|c|}\hline a4 & 68 & 6b & 02 \\ \hline 9c & 9f & 5b & 6a \\ \hline 7f & 35 & ea & 50 \\ \hline f2 & 2b & 43 & 49 \\ \hline \end{array}$$

AES encryption example

- 5. Repeat 10 times
  - encrypted ciphertext



39	02	dc	19
25	dc	11	6a
84	09	85	0b
1d	fb	97	32

Public key cryptography concepts (asymmetric keys)

3.3 Asymmetric Cryptograph

Symmetric Key System Flaws

- Symmetric cryptosystems (e.g. DES, AES) allow two users to establish a "secure channel" using a secret shared in advance, but it is not easy to share a secret between two communicating parties.

- Consider a group of N users who need to communicate securely with each other, using a symmetric cryptosystem to protect the communication between them: each user needs to share the private key with the remaining N - 1 users, and **the whole system needs to manage N(N - 1)/2 keys**.

- There is no guarantee of traceability (accountability), Why?

Difficulty of key distribution

No support for "open systems"

High key management costs

History of public key cryptosystems

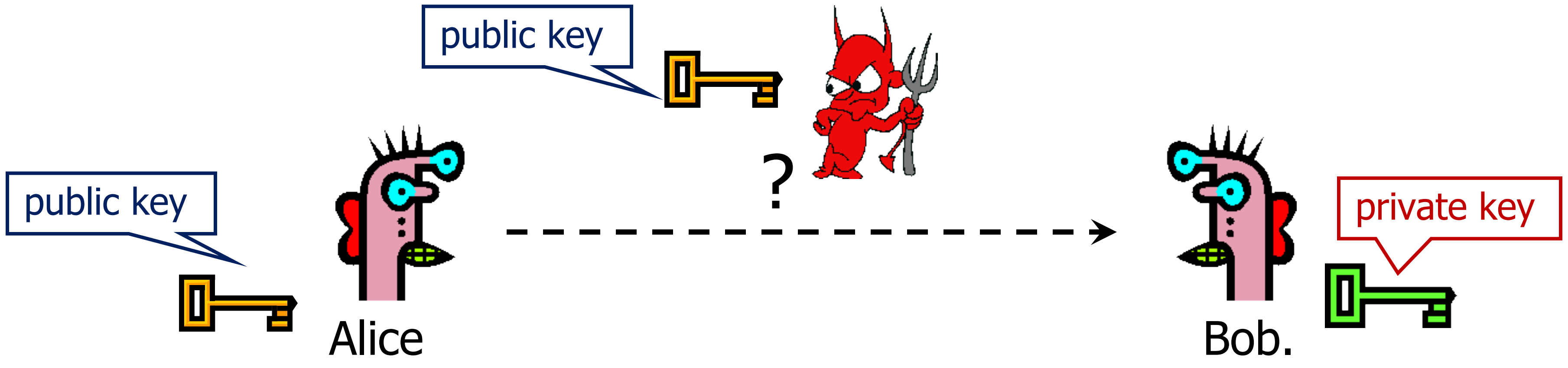
- 1976 Diffie and Hellman introduce the concept of "public-key cryptography", winning the ACM Turing Award in2015.
- 1977 Ronald Rivest, Adi Shamir and Leonard Adleman propose the RSA algorithm.

Introduction to public key cryptosystems

- **Main Idea**:

Some problems exhibit "asymmetry" - it is very easy to compute in one direction and difficult to compute in the other. That is, **encryption is easy and decryption is difficult!**

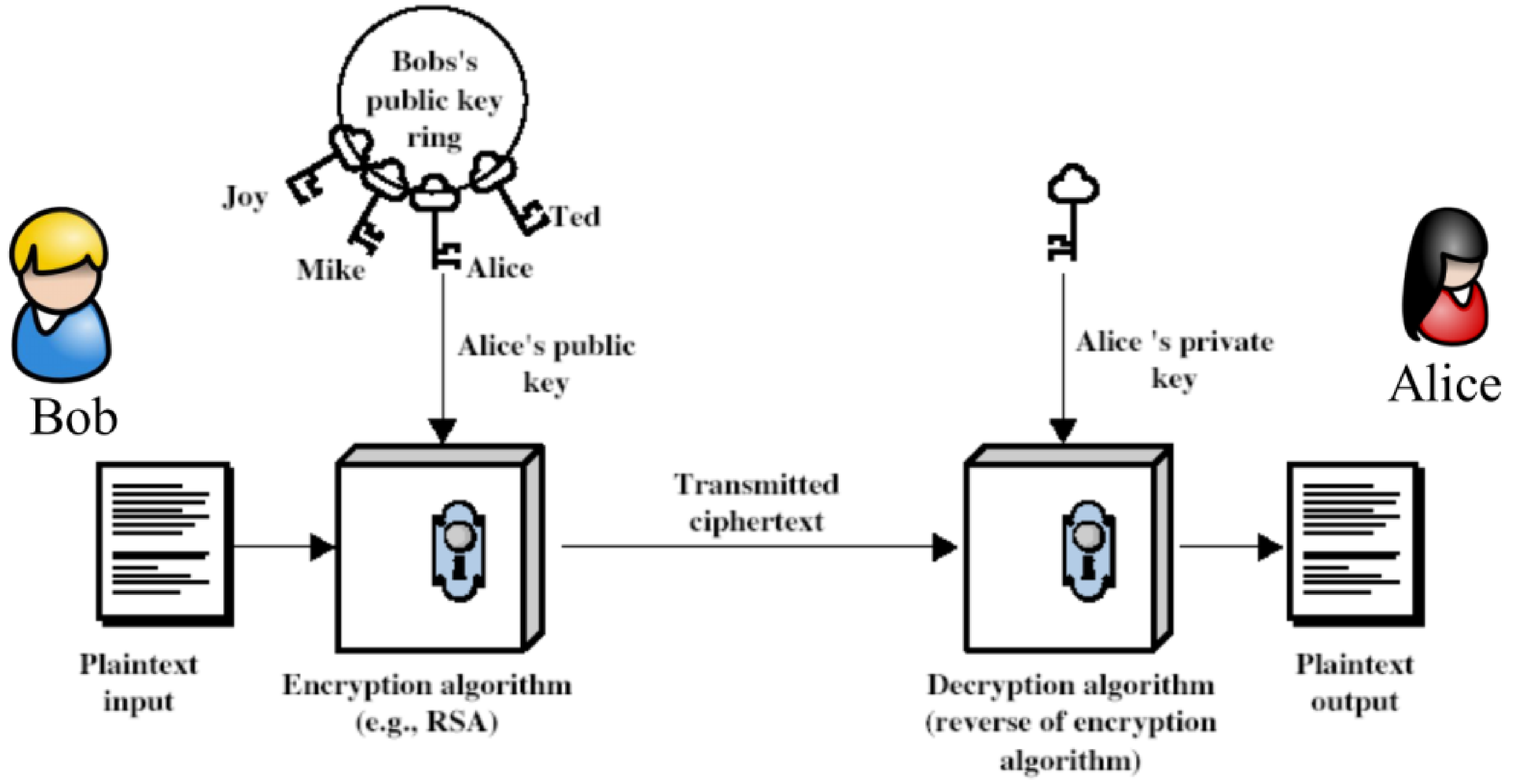
E.g., it is easy to compute the product of any given integer, but very difficult to compute the factor of a given large integer.



Public key PK: public within the system, used by other users

Private key sk: to be used by the user

public key cryptosystem (PKCS) basic idea



Introduction to public key cryptosystems

Key Generation: generates a pair of public key PK and private key SK through a relatively easy computational process.

- The operation of getting the private key SK is computationally infeasible if only the public key PK is obtained.

Encryption: Given a plaintext M and a public key PK, it is easy to compute the ciphertext C = E_{PK} (M).

Decryption: Given the ciphertext C=E_{PK} (M) and the private key SK, it is easy to compute the plaintext M

- If the private key SK is missing, it is **not possible to** compute the plaintext M from the ciphertext C.

No: under current computing power conditions, computation time and message length grow exponentially.

key distribution

Public keys can be transmitted using public (authenticated) channels

Key Management

In an N-person system, the entire system simply needs to maintain N public keys

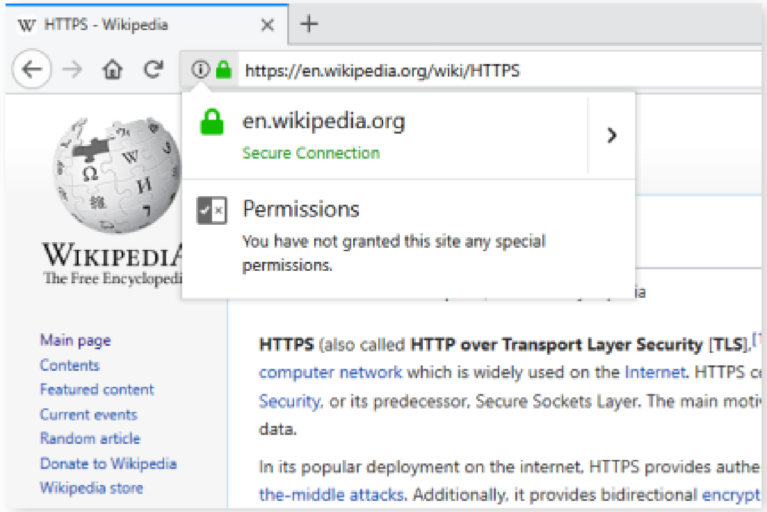
open system

Users without a pre-established relationship can also establish secure communication via each other's public keys

Public key cryptosystem applications

- **HTTPs**: Hyper Text Transfer Protocol over Secure Socket Layer
  Adding SSL to the HTTP protocol improves security and can be used to communicate sensitive information.
- **PGP**: Pretty Good Privacy: secure E-mail

- **Q: How about public key encryption as a key transfer method for private key encryption?**



HTTPS - Wikipedia

https://en.wikipedia.org/wiki/HTTPS

en.wikipedia.org
Secure Connection

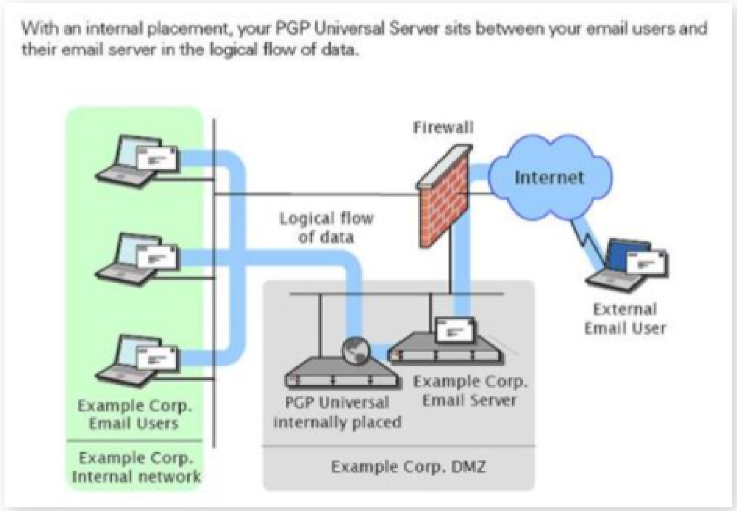
Permissions
You have not granted this site any special permissions.

WIKIPEDIA
The Free Encyclopedi

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store

HTTPS (also called **HTTP over Transport Layer Security [TLS]**, computer network which is widely used on the Internet. HTTPS c Security, or its predecessor, Secure Sockets Layer. The main moti data.

In its popular deployment on the internet, HTTPS provides authe the-middle attacks. Additionally, it provides bidirectional encrypt



With an internal placement, your PGP Universal Server sits between your email users and their email server in the logical flow of data.

Firewall
Internet
Logical flow of data
External Email User
Example Corp. Email Users
PGP Universal internally placed
Example Corp. Email Server
Example Corp. Internal network
Example Corp. DMZ

3.2 RSA Encryption Algorithm

RSA encryption algorithm

History of the RSA public key cryptosystem

- In 1978, Ron Rivest, Adi Shamir and Leonard Adleman of MIT proposed the RSA encryption algorithm.
- In 2002, he received the Turing Award.

Principles of RSA public key cryptography

Key generation:

1. Choose two large prime numbers p,q (e.g. 1024 bits each)
2. Choose $n = pq$ and $z = \varphi(n) = (p-1)(q-1)$
3. Randomly select e (where $e < n$), e and z have no convention. (e, z are "mutually prime")
4. Pick d such that ed-1 is completely divisible by z (i.e. $ed \bmod z = 1$)
5. The public key is (n, e) and the private key is (n, d).

RSA example

1. Select primes: p=61 & q=53
2. Compute $n = pq = 3233$
3. Compute $\varphi(n)=(p-1)(q-1) = 3120$
4. Select e : $\gcd(e, \varphi(n))=1$; choose $e = 17$
5. Determine d: ***de* = 1 mod φ(n)** and ***d* < φ(n)**.

   17*d - 3120*k = 1

   Private key: **d = 2753, (k=15)**

   Public key: n = 3233, e=17

   (PS: without p and q, calculating d is computationally infeasible.)

   Encryption and decryption example:

6. Suppose m=30120
7. Ciphertext $c=m^e \bmod n = (30120)^{17}$
8. Reconstruct plaintext: $m=c^d \bmod n = 30120$

Principles of RSA public key cryptography

Encryption/decryption algorithm:

Give (n,e) and (n,d) as above.

Encryption: The transformation of plaintext m into ciphertext c by $c = m^e \bmod \mathrm{n}$

(i.e., the remainder obtained when is divided by n).

Note: m<n (chunked if needed)

Decryption: $m = c^d \bmod \mathrm{n}$ (i.e., the remainder of divided by n).

Core Ideas: $m = (\underbrace{m^e \bmod n}_{c})^d \bmod n$

Principles of RSA public key cryptography

There are in **RSA**:

1. $n = p \cdot q$

2. $\varphi(n) = (p - 1) \cdot (q - 1)$

3. Choose integers e and d, with d being the inverse/module inverse element of e with respect to the modulus $\varphi(n)$

4. $e \cdot d = 1 + k \cdot \varphi(n)$ $(k > 0, k \epsilon Z)$

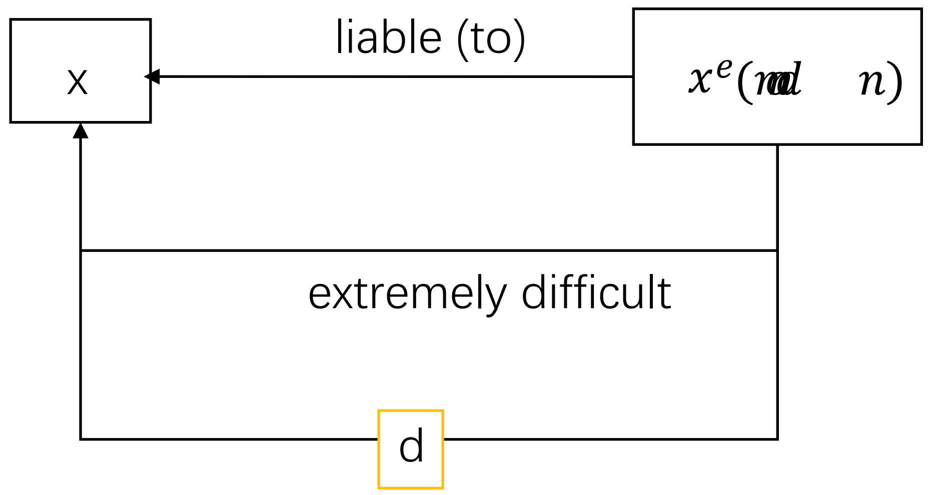
Q: Why?

Thus we have $C^d = (M^e)^d = M^{1+k\,\varphi(n)} = M*(M^{\varphi(n)})^k = M*(1)^k = M \bmod n$

by **Euler's theorem**:

When (a, n are mutually prime), $a^{\varphi(n)} \bmod n = 1$.

RSA Security Essentials



It's not hard if **d** is known, **d** is the trapdoor of the function

$d = e^{-1}\, mod\, (\varphi(n))$ i.e., if e is known and $\varphi(n)$ is unknown, find d

If $\varphi(n)$ is known, it is very easy to find d i.e. if n is known, find $\varphi(n)$

If we use $n = pq$, $\varphi(n) = (p-1)(q-1)$, the solution is very easy to find

Problem essence: n is known, find $n = pq$ i.e., the problem of prime decomposition of numbers. The time complexity of this problem is exp (sqrt (ln n lnln n))

In general, breaking RSA keys becomes a computationally unsolvable problem.

RSA Security Analysis

exhaustive attack

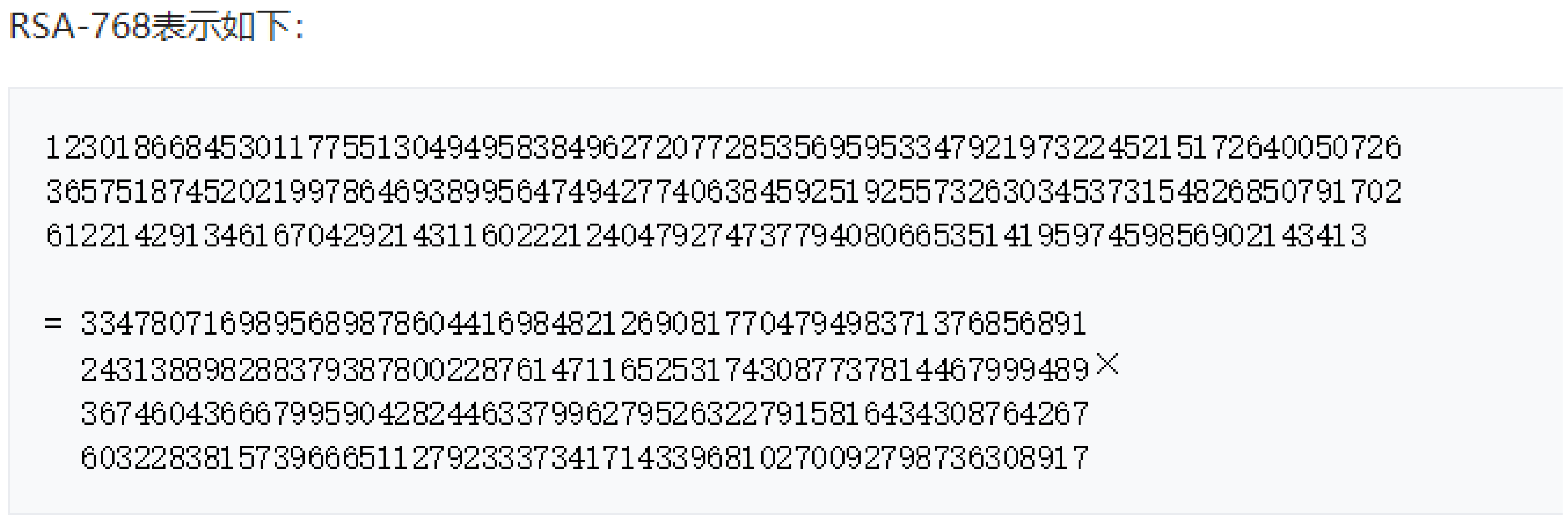
1. The attacker designs an M, C = M^e (mod n)
2. There are at most n-1 number of d's, try to use each d to crack, if M'=$C^{d'}$ (mod n) = M, d' is the solution
3. Let p, q be 100 bits (decimal) each, then n-1 is about 200 bits (decimal) n=10^{200}
4. Assuming you can do 100 million searches per second (10^8), you can search 10^8 per year *60*60*24*365=3*10^{15}

The time to search 10^{200} keys is 10^{200} / (3*10^{15}) = **3*10^{185} years!**

Not computationally feasible.

RSA Security

- On **12** December 2009, the number **RSA-768** (**768 bits, 232 digits**) was also successfully decomposed. This incident threatens the security of the **1024-bit** keys that are currently in use, and it is widely believed that users should upgrade to **2048-bit** or above as soon as possible.



RSA-768表示如下:

```
1230186684530117755130494958384962720772853569595334792197322452151726400507263
6575187452021997864693899564749427740638459251925573263034537315482685079170
2612214291346167042921431160222124047927473779408066535141959745985690214341
3

= 3347807169895689878604416984821269081770479498371376856891
  2431388982883793878002287614711652531743087737814467999489 ×
  3674604366679959042824463379962795263227915816434308764267
  6032283815739666511279233373417143396810270092798736308917
```

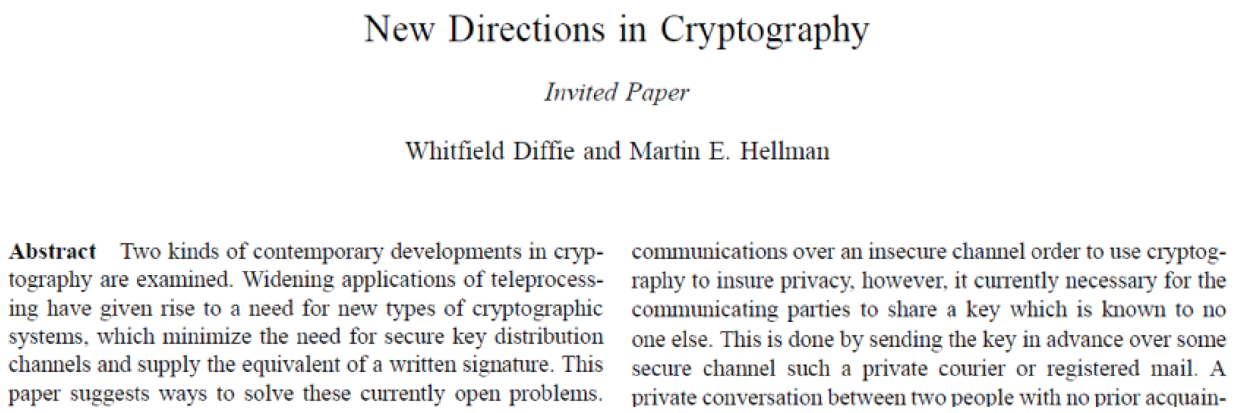
Problems with Public Key Encryption - How to Key Establishment

- Data needs to be encrypted before sending
- Keys need to be shared between the parties, and the problem of encrypting data is equivalent to how to create a security key
- Question: How do I create a shared key if the two parties are not physically together?
- Key establishment is the first step in public key encryption
- There are two methods of key establishment:
  - Key Agreement
  - Key Distribution

DH Key Exchange Protocol - Key Agreement

- **DH implements how two parties can get a common key when communicating under an insecure communication channel.**
  - The commercial encryption algorithms AES and DES require key sharing between the communicating parties prior to secure communication

The Diffie-Hellman algorithm was the first public key algorithm, proposed in 1976. Its security stems from the fact that computing discrete logarithms over a finite field is more difficult than computing exponents. The algorithm allows a key to be securely exchanged between two users, but cannot be used to encrypt or decrypt messages.



New Directions in Cryptography

Invited Paper

Whitfield Diffie and Martin E. Hellman

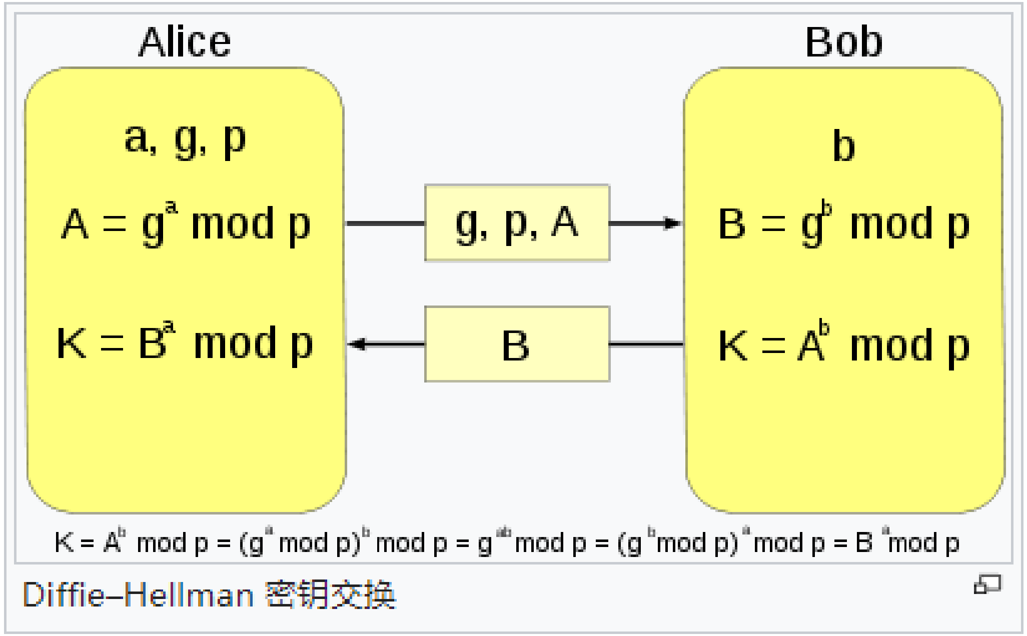
Abstract Two kinds of contemporary developments in cryptography are examined. Widening applications of teleprocessing have given rise to a need for new types of cryptographic systems, which minimize the need for secure key distribution channels and supply the equivalent of a written signature. This paper suggests ways to solve these currently open problems. communications over an insecure channel order to use cryptography to insure privacy, however, it currently necessary for the communicating parties to share a key which is known to no one else. This is done by sending the key in advance over some secure channel such a private courier or registered mail. A private conversation between two people with no prior acquain-

DH protocol fundamentals

- Given two parties A,B, A,B establishes a shared key by the following process:

$$A \rightarrow B : g^{a} \bmod p \qquad B \rightarrow A : g^{b} \bmod p$$
$$A : \left(g^{b}\right)^{a} \bmod p = g^{ab} \bmod p \qquad B : \left(g^{a}\right)^{b} \bmod p = g^{ab} \bmod p$$

g^{ab} is the shared key



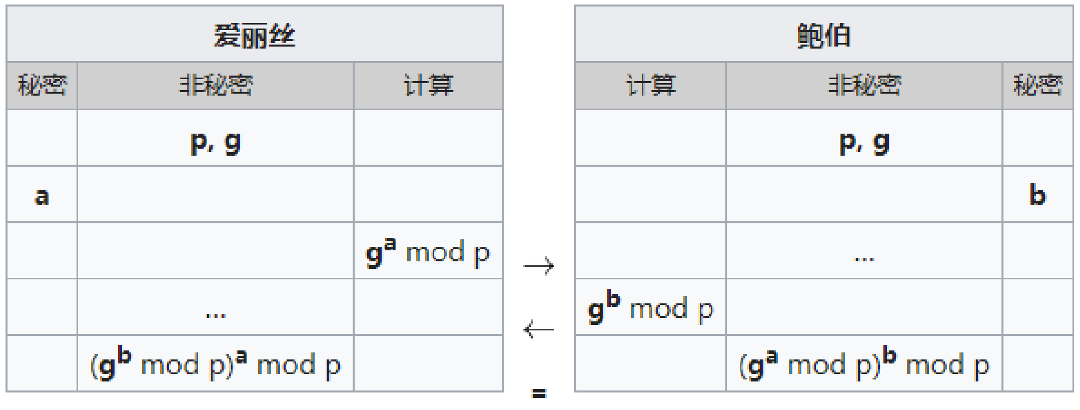
$K = A^b \bmod p = (g^a \bmod p)^b \bmod p = g^{ab} \bmod p = (g^b \bmod p)^a \bmod p = B^a \bmod p$

Diffie–Hellman 密钥交换

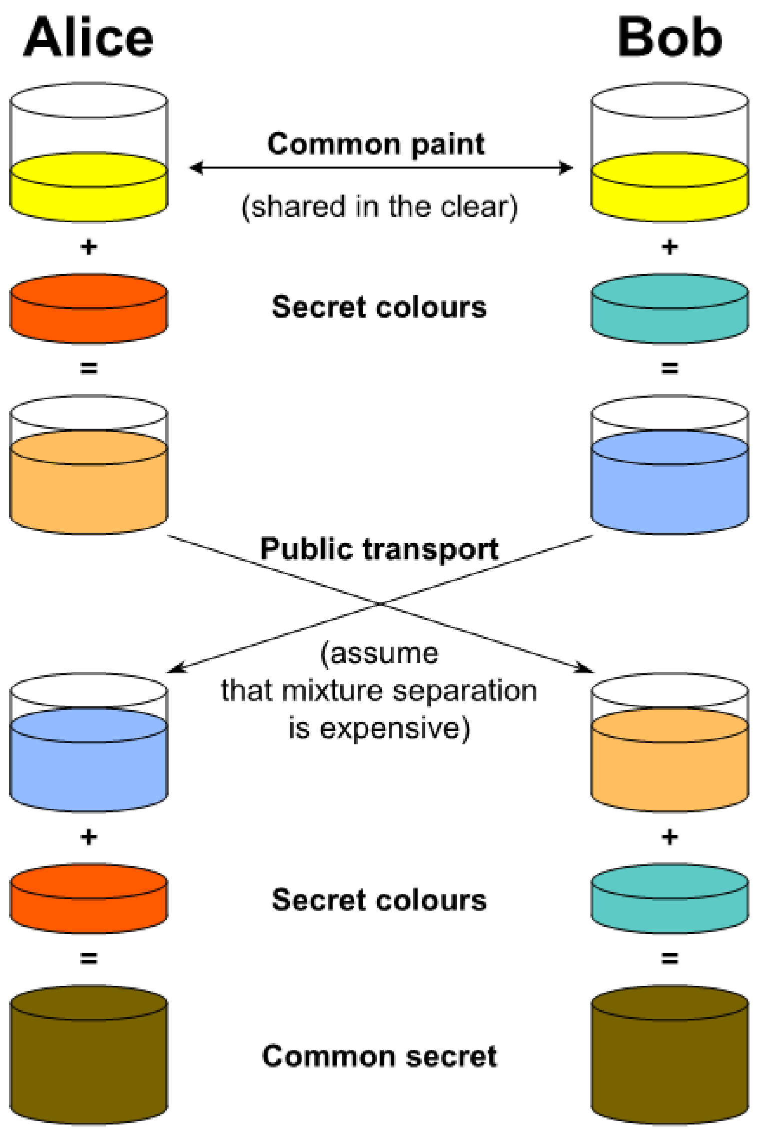
Q: What is confidential and what is public about the process on the left?

DH Key Exchange Protocol Example

1. Alice and Bob agree to use the prime numbers $p = 23$, base $g = 5$
2. Alice chooses the secret integer to give: $a = 6$; calculates $A = g^a \, mod \, p$ to send to Bob.
   - $A = 5^6 \, mod \, 23 = 8$
3. Bob chooses the secret integer to give: $b = 15$; calculates $B = g^b \, mod \, p$ and sends it to Alice.
   - $B = 5^{15} \, mod \, 23 = 19$
4. Alice Computing $s = B^a \, mod \, p$
   - $19^6 \, mod \, 23 = \mathbf{2}$.
5. Bob Calculations.
   - $8^{15} \, mod \, 23 = \mathbf{2}$.



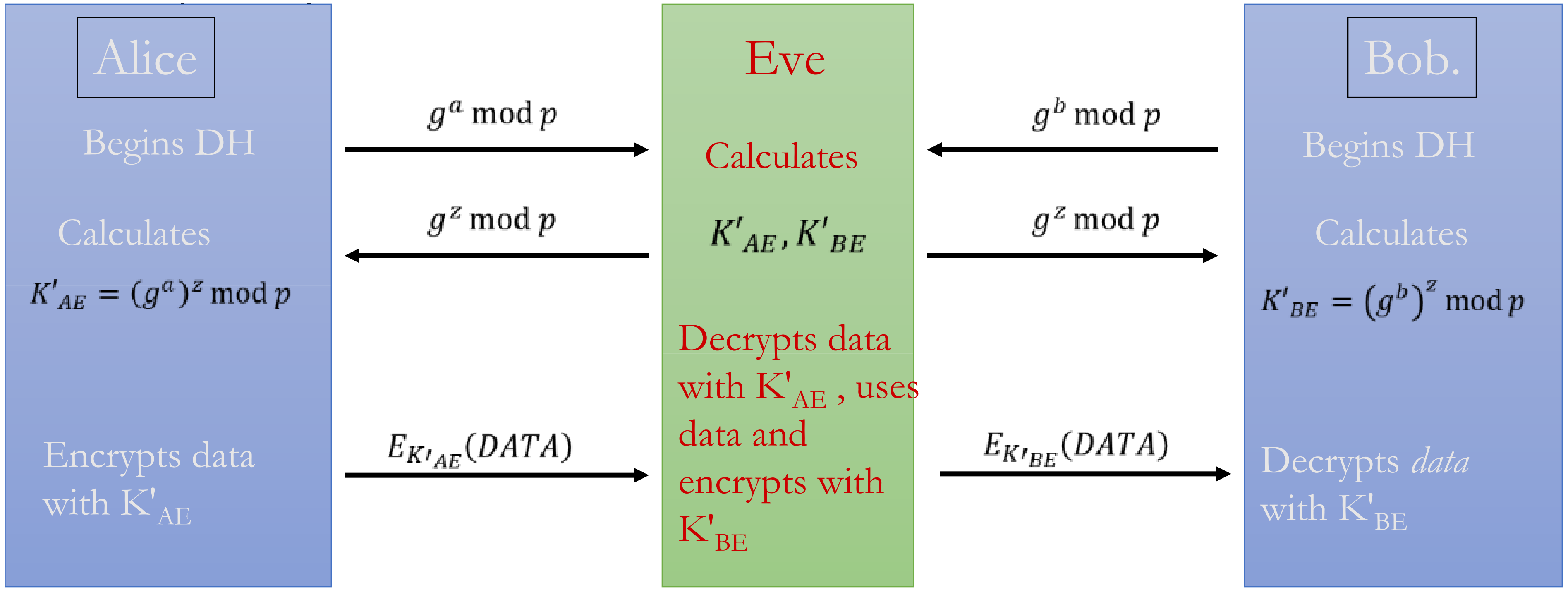
爱丽丝				鲍伯		
秘密	非秘密	计算		计算	非秘密	秘密
	p, g				p, g	
a						b
		g^a mod p	→		...	
	...		←	g^b mod p		
	$(g^b$ mod p$)^a$ mod p		=		$(g^a$ mod p$)^b$ mod p	


Alice
Bob
Common paint
(shared in the clear)
+
+
Secret colours
=
=
Public transport
(assume
that mixture separation
is expensive)
+
+
Secret colours
=
=
Common secret

Security Issues in DH Key Exchange

- Vulnerable to **blocking attacks**: since power operations are computationally intensive, when an adversary initiates a large number of key requests, the attacked party will spend larger computational resources to do the power operations;

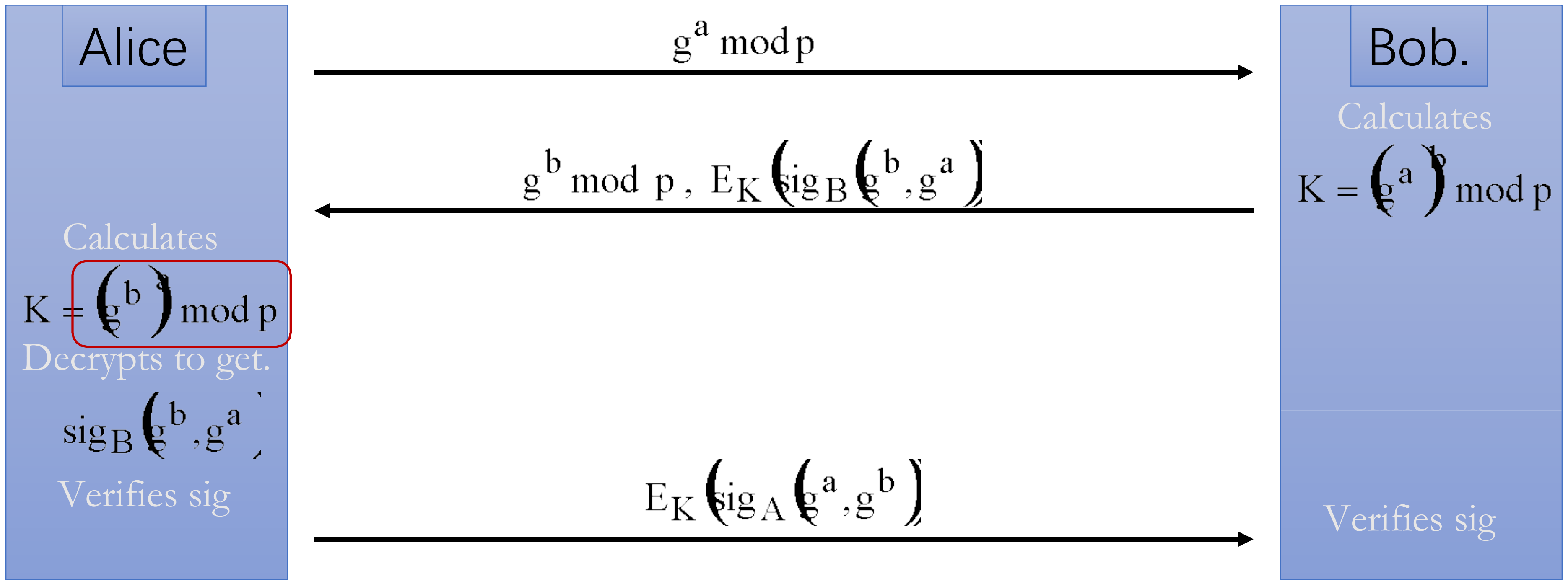
- Vulnerable to **man-in-the-middle attacks**: an adversary can impersonate one of users A and B, respectively, and exchange keys with the other (which can listen in on and pass on secret messages from A and B without being


Alice
Begins DH
Calculates
$K'_{AE} = (g^a)^z \bmod p$
Encrypts data with K'_{AE}
$g^a \bmod p$
$g^z \bmod p$
$E_{K'_{AE}}(DATA)$
Eve
Calculates
K'_{AE}, K'_{BE}
Decrypts data with K'_{AE}, uses data and encrypts with K'_{BE}
$g^b \bmod p$
$g^z \bmod p$
$E_{K'_{BE}}(DATA)$
Bob.
Begins DH
Calculates
$K'_{BE} = (g^b)^z \bmod p$
Decrypts *data* with K'_{BE}

Defence against DH man-in-the-middle attacks

- Digital signatures can be used to stop DH man-in-the-middle attacks
- Private sig( ) function: digital signature
- Public ver( ) function: authentication

Q: Why is it effective?



Alice
Bob.
$g^a \bmod p$
$g^b \bmod p,\ E_K\left(sig_B\left(g^b, g^a\right)\right)$
Calculates
$K = \left(g^a\right)^b \bmod p$
Calculates
$\boxed{K = \left(g^b\right)^a \bmod p}$
Decrypts to get.
$sig_B\left(g^b, g^a\right)$
Verifies sig
$E_K\left(sig_A\left(g^a, g^b\right)\right)$
Verifies sig

Diffie joins ZJU

3.4 Cryptography Encryption in IoT

Cryptographic Applications for the Internet of Things

Cryptographic Encryption Applications for the Internet of Things

- **Limitations of IoT devices:**
  - small memory
  - low energy supply
  - arithmetic limitations
  - Encryption protocols are embedded in the firmware and are difficult to change
  - Interaction of information requires encryption and decryption speed

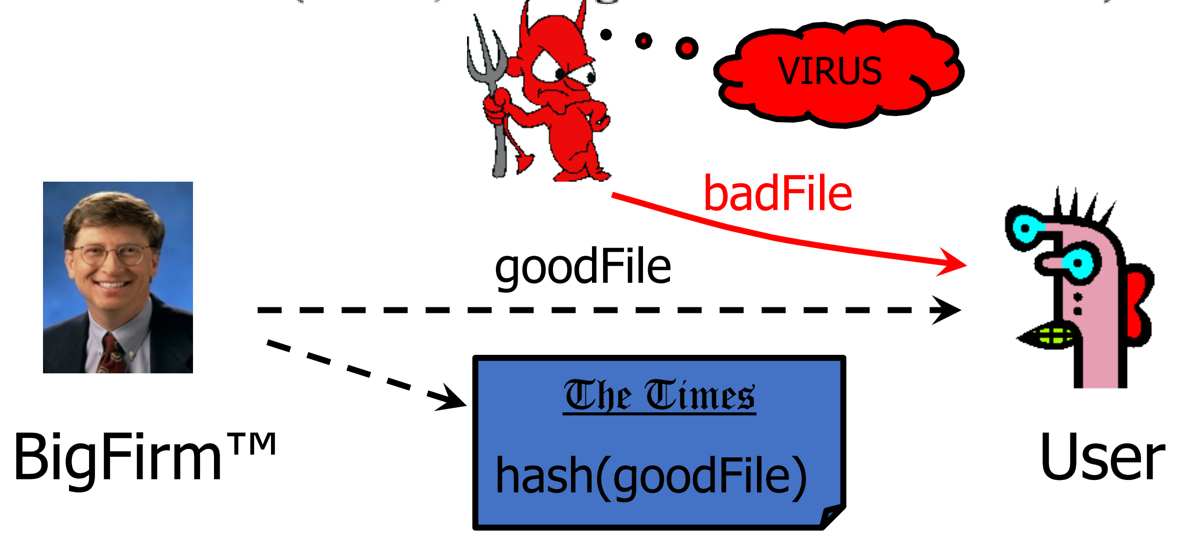
- **Lightweight encryption algorithm**
  - Saurabh Singh et al. summarised the existing lightweight encryption algorithms based on key length, number of encryption rounds

arithmetic	Key Length	encrypted rounds
AES	128/192/256	10/12/14
HEIGHT	128	32
PRESENT	80/128	31
RC5	0-2040	1-255
TEA	128	64
XTEA	128	64
LEA	128/192/256	24/28/32
DES	54	16
Seed	128	16
Twine	80/128	32
DESL	54	16
3DES	56/112/168	48
Hummingbird	256	4
Hummingbird	256	4
Iceberg	128	16
Pride	128	20

Lightweight encryption algorithms

IoT Authentication - How to Ensure Integrity?

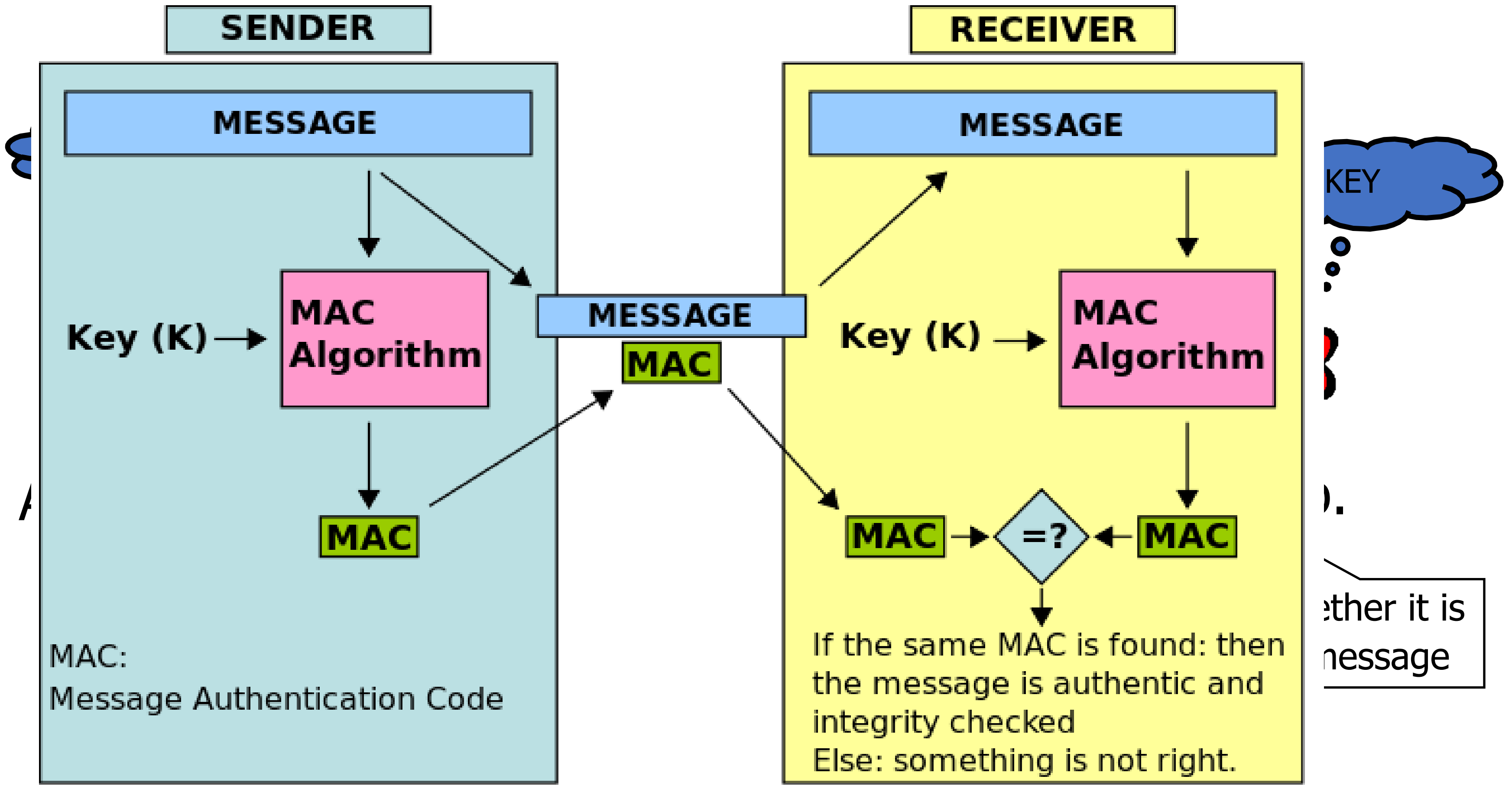
- **Message authentication code (MAC, message authentication code)**



$$\mathrm{MAC} = H(\mathrm{message} + \mathrm{key})$$

H is a hash function and '+' stands for concatenation operation

Non-encrypted authentication methods

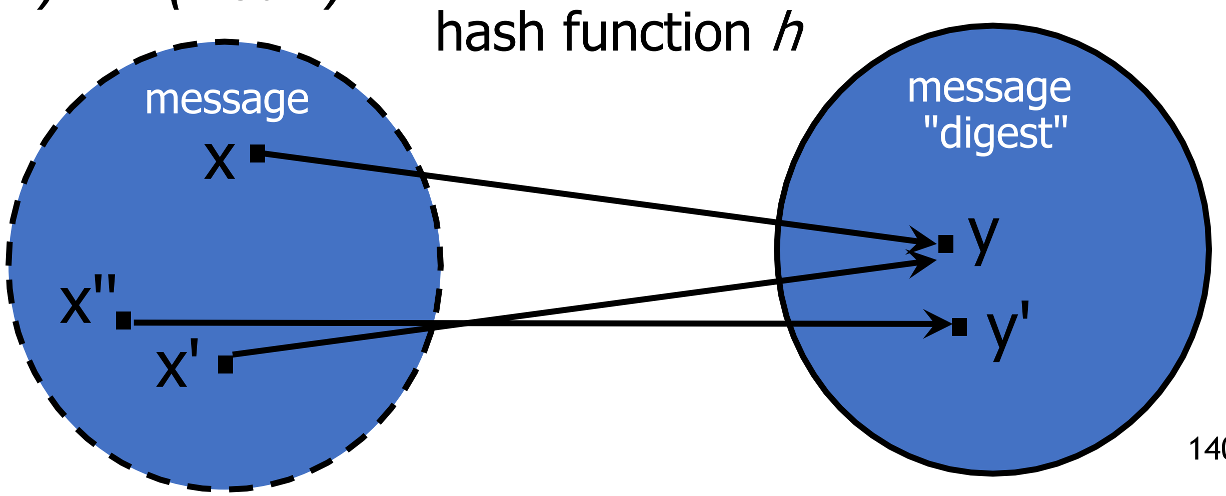

SENDER
MESSAGE
Key (K)
MAC Algorithm
MAC
MAC:
Message Authentication Code
MESSAGE
MAC
RECEIVER
MESSAGE
Key (K)
MAC Algorithm
MAC
=?
MAC
If the same MAC is found: then the message is authentic and integrity checked
Else: something is not right.
KEY
ether it is
nessage

Integrity Authentication: only someone who knows the key KEY can compute the MAC for a given message.

IoT Authentication Methods - Hash/Hash Functions

- **The hash/hash function y=h(x) :**
  - For any length of input x
  - For any length of input x, the output z is of fixed length
  - Low computational complexity
  - For a given output y, finding the corresponding input x is impossible
  - For a given x_1, finding x_2 that satisfies 〚h(x〛 _1)= 〚h(x〛 _2) is not possible

Example: ***h(m) = m (mod n)***



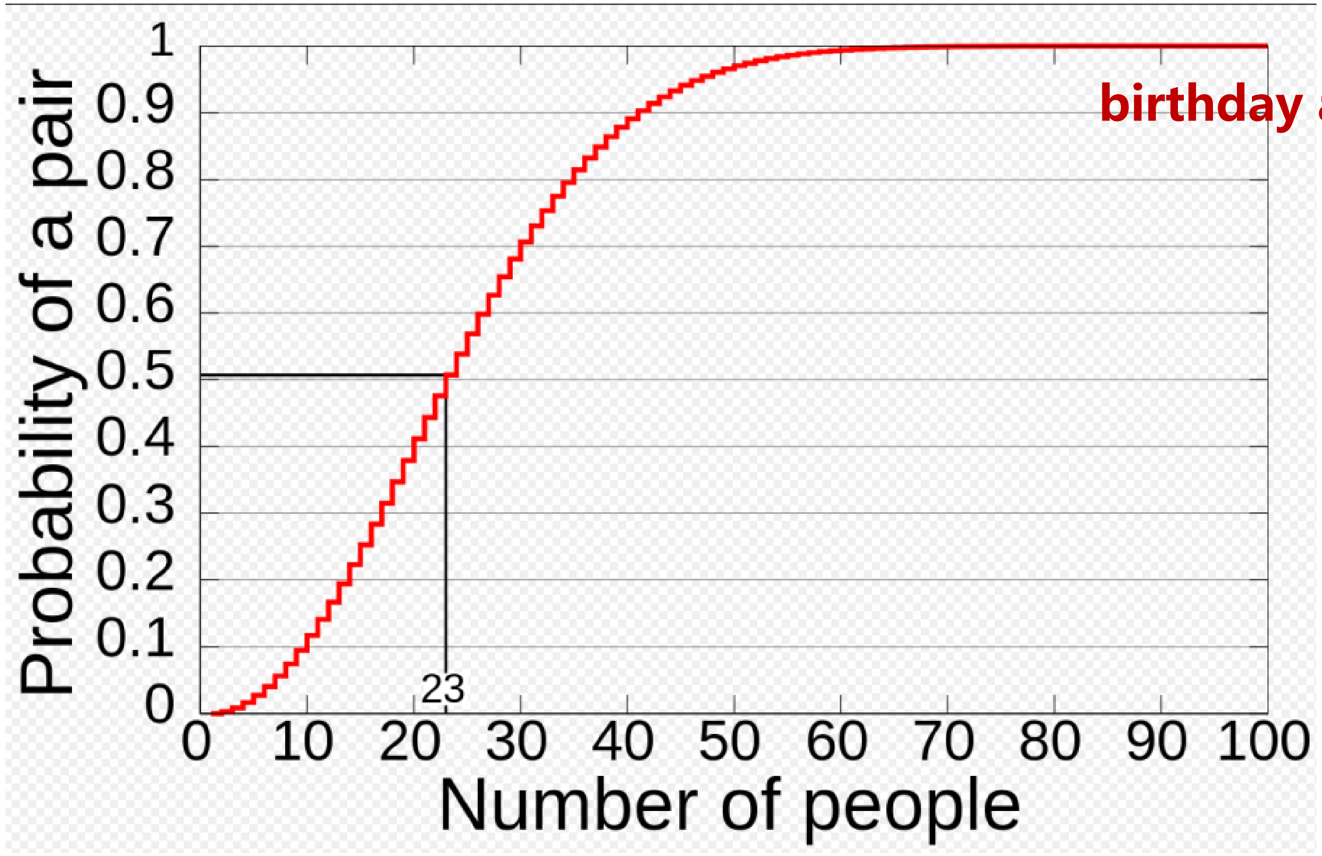
Characteristic requirements

- Unidirectional/Primary image resistance (Preimage resistance)
  - Let h(x')=y$\in\{0,1\}^n$ for a random x'
  - Given y, it should be hard to find any x such that h(x)=y

- crashworthiness
  - Should be hard to find any pair (x, x') such that h(x)=h(x')
  - Vulnerable to Birthday paradox

birthday attack (esp. on children's birthday)

- **Collisions in Hashing Algorithms**
  - will appear $h(x_1)=h(x_2)$

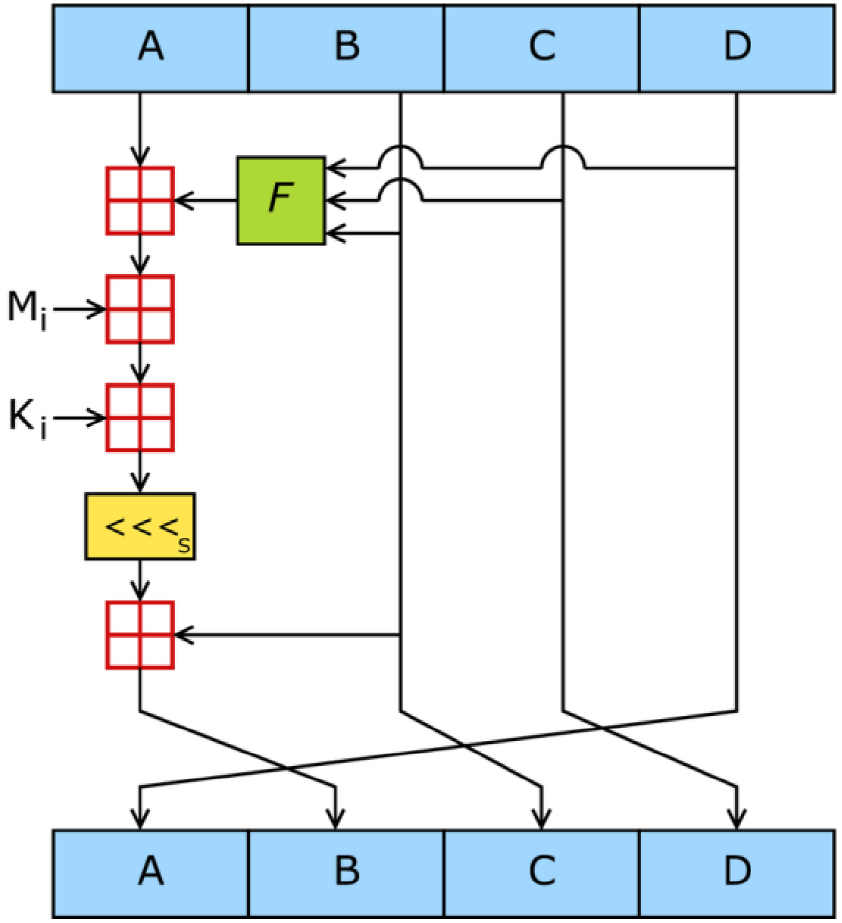


birthday attack (esp. on children's birthday)

Cryptographic Authentication Applications for the Internet of Things

- **Common Hash Functions**
  - **MD5** (Message Digest) algorithm
  - MD5 is a one-way hash function published by cryptographer Rivest in 1992.
  - Discovery collision in the summer of 2004

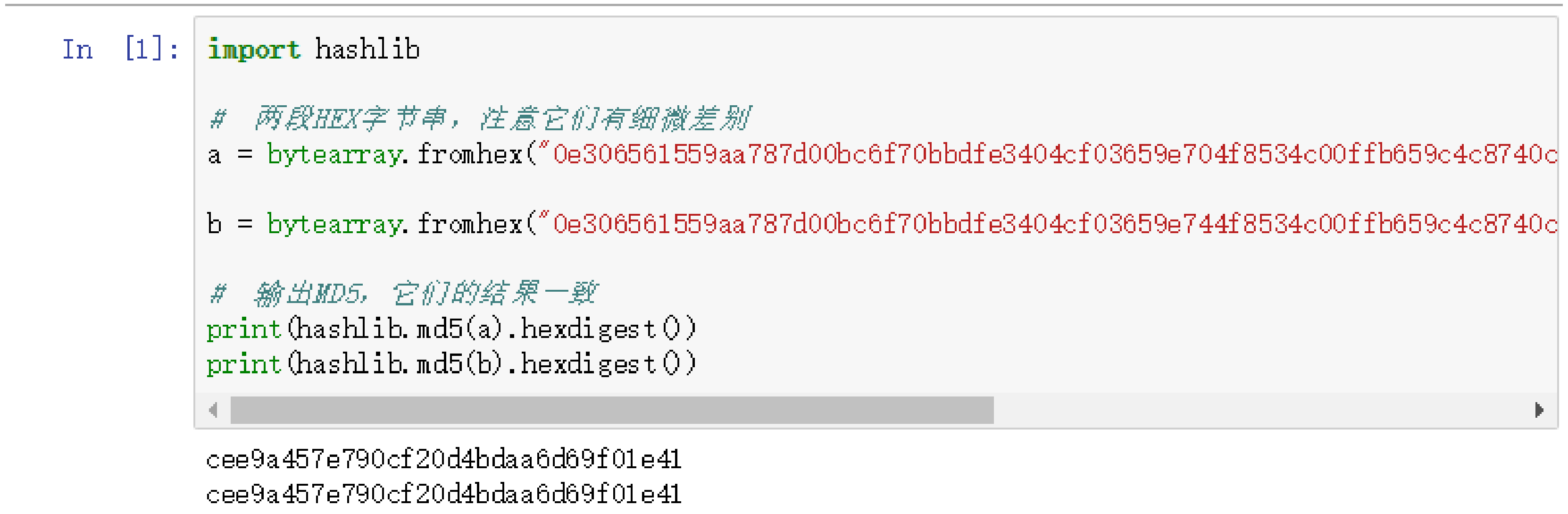
- **MD5 Features:**
  - The input message can be any length
  - Processes 512bits of message packets per iteration
  - The final output hash value is **128bits**



Example: BitTorrent download clip integrity

Cryptographic Authentication Applications for the Internet of Things

- **Crash case of MD5**



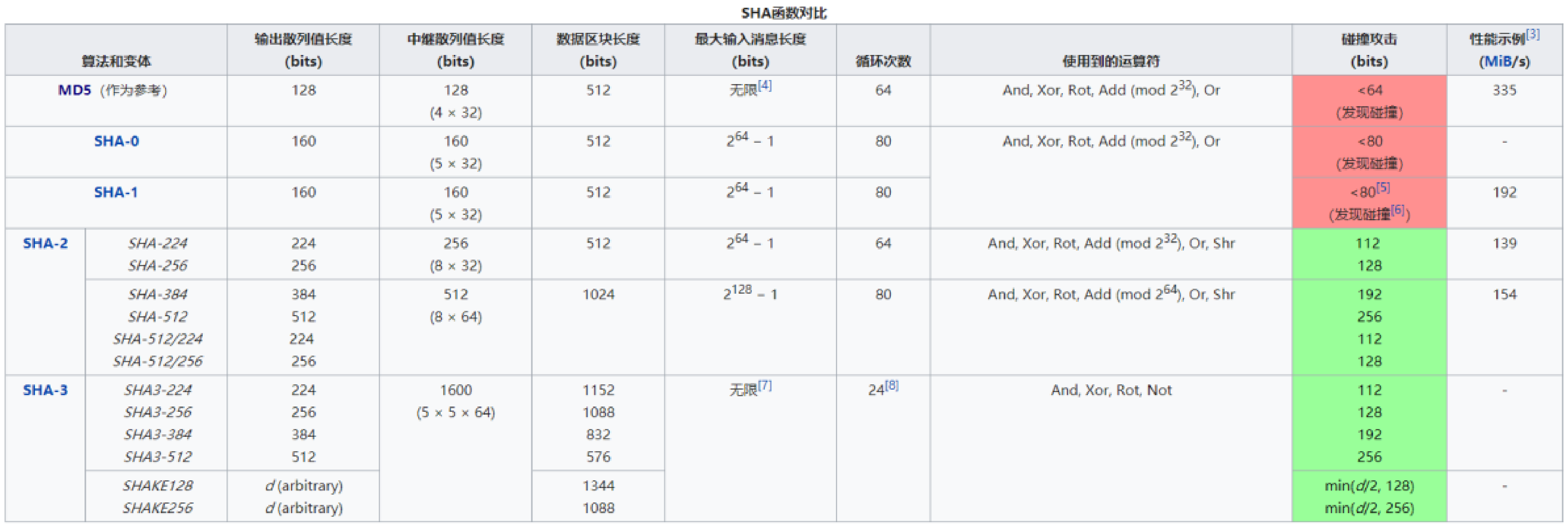
```
In  [1]: import hashlib

         #  两段HEX字节串，注意它们有细微差别
         a = bytearray.fromhex("0e306561559aa787d00bc6f70bbdfe3404cf03659e704f8534c00ffb659c4c8740c

         b = bytearray.fromhex("0e306561559aa787d00bc6f70bbdfe3404cf03659e744f8534c00ffb659c4c8740c

         #  输出MD5, 它们的结果一致
         print(hashlib.md5(a).hexdigest())
         print(hashlib.md5(b).hexdigest())

         cee9a457e790cf20d4bdaa6d69f01e41
         cee9a457e790cf20d4bdaa6d69f01e41
```

□In fact we can find a case of MD5 collision in our smartphones every few seconds, so a few years ago MD5 was not recommended as an algorithmic solution in applications, and it was replaced by the SHA (Secure Hash Algorithm) family of algorithms.

Cryptographic Authentication Applications for the Internet of Things

- **SHA (Security Hash Algorithm)**
  - ◻Secure Hash Algorithm
  - ◻Maximum message length < 2^{64}
  - ◻Message packet length 512bits
  - ◻Output hash length **160bits**
  - ◻**The collision was discovered in the summer of 2005.**

SHA family



SHA函数对比

算法和变体		输出散列值长度 (bits)	中继散列值长度 (bits)	数据区块长度 (bits)	最大输入消息长度 (bits)	循环次数	使用到的运算符	碰撞攻击 (bits)	性能示例[3] (MiB/s)
MD5（作为参考）		128	128 (4 × 32)	512	无限[4]	64	And, Xor, Rot, Add (mod 2^{32}), Or	<64 (发现碰撞)	335
SHA-0		160	160 (5 × 32)	512	$2^{64}-1$	80	And, Xor, Rot, Add (mod 2^{32}), Or	<80 (发现碰撞)	-
SHA-1		160	160 (5 × 32)	512	$2^{64}-1$	80		<80[5] (发现碰撞[6])	192
SHA-2	*SHA-224* *SHA-256*	224 256	256 (8 × 32)	512	$2^{64}-1$	64	And, Xor, Rot, Add (mod 2^{32}), Or, Shr	112 128	139
	SHA-384 *SHA-512* *SHA-512/224* *SHA-512/256*	384 512 224 256	512 (8 × 64)	1024	$2^{128}-1$	80	And, Xor, Rot, Add (mod 2^{64}), Or, Shr	192 256 112 128	154
SHA-3	*SHA3-224* *SHA3-256* *SHA3-384* *SHA3-512*	224 256 384 512	1600 (5 × 5 × 64)	1152 1088 832 576	无限[7]	24[8]	And, Xor, Rot, Not	112 128 192 256	-

	SHAKE128
	SHAKE256
		d (arbitrary)
		d (arbitrary)
				1344
				1088
								min(*d*/2, 128)
								min(*d*/2, 256)
									-

Cryptographic Authentication Applications for the Internet of Things

- **Comparison of MD5 and SHA**

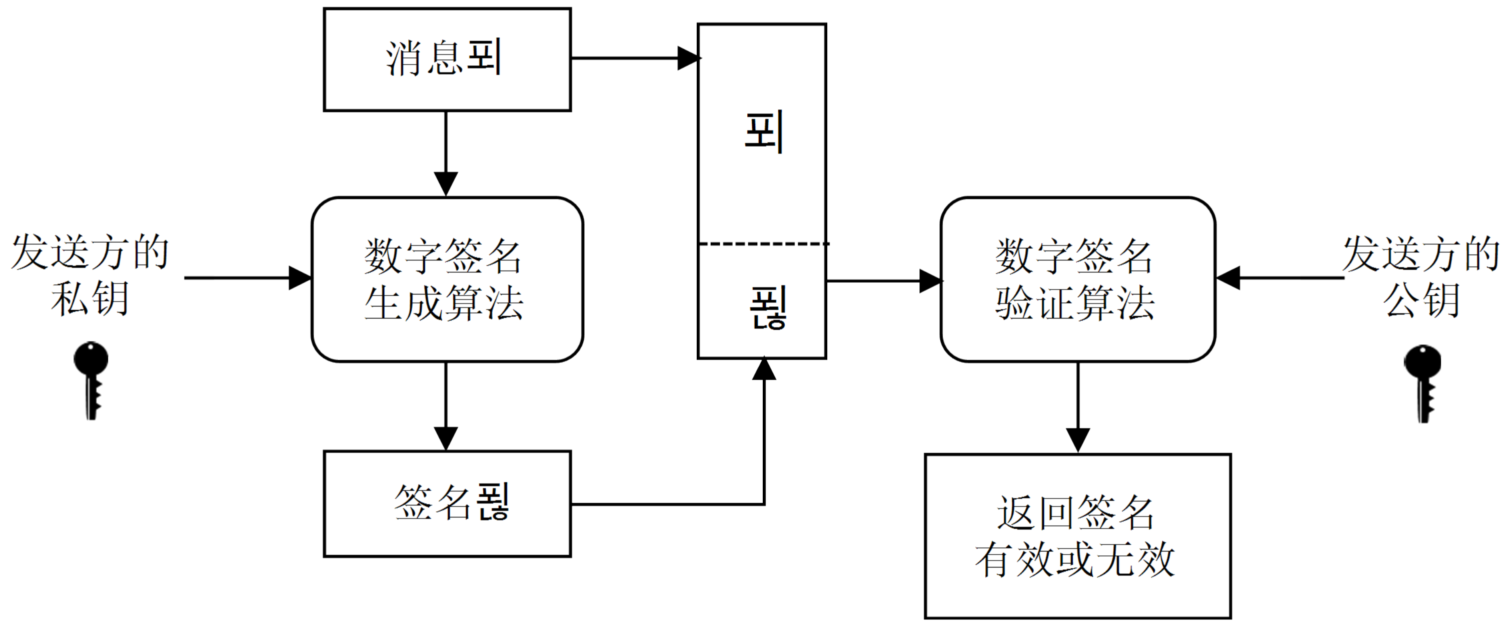
	SHA-1	MD5
Maximum Message Length	$<2^{64}$	not limited to
Message Packet Length	512	512
framework	Merkle Iteration	Merkle Iteration
Link variable length	160	128
Hash length	160	128

IoT Authentication - Digital Signature

- **Digital signatures: preventing repudiation by both parties to the communication**
  - System initialisation, parameters required to generate digital signatures
  - The sender signs the message with its own private key
  - The sender transmits both the original message and the digital signature as an attachment to the message receiver
  - The receiver decrypts the signature using its own public key
  - The receiver compares the decrypted message with the original message, and if it matches, the message has not been tampered with or corrupted during transmission

digital signature

- The process of digital signatures:



Digital signatures based on the RSA algorithm

- **RSA public key algorithm digital signature:**

  1. Parameter selection and key generation
  2. Signature process: User A signs the message M and computes the
  3. S=Sig(M)=$M^d mod\ n$
  4. M+S
  5. Verification process: user B verifies user A's digital signature on message M, calculates the

$$M' = \text{Ver(S)} = S^e \text{mod n}$$

$$M' = \text{M?}$$

- In addition to the RSA digital signature algorithm, there are also DSA algorithms, Elgamal, etc. Compared to other digital signature algorithms, RSA digital signatures have a low security level and slow computation speed, so they have gradually been replaced by the

IoT Key Management

- **Keys are also data**, which are used to encrypt other data, and when designing a cryptosystem, **the following issues** must be considered for keys**:**
  - Where the key is needed, how to set it up and install it in that place
  - What is the expected lifetime of the key and how often should the key be replaced?
  - How to keep your keys strictly protected

- **Basic principles of key protection:**
  - Keys can never be in plaintext outside of a cryptographic device, which can be hardware or software

IoT Key Management

- **Key distribution method:**
  - ☐A selects the key and manually passes it to B
  - ☐Third party C chooses the key to be passed manually to A,B respectively
  - ☐Transmit the new key with the original shared key of A and B.
  - ☐C, a third party that has a shared key with A and B respectively, transmits a new key to A and/or B.
- If symmetric encryption, N user sets require $\frac{N(N-1)}{2}$ shared keys

Summary of the chapter

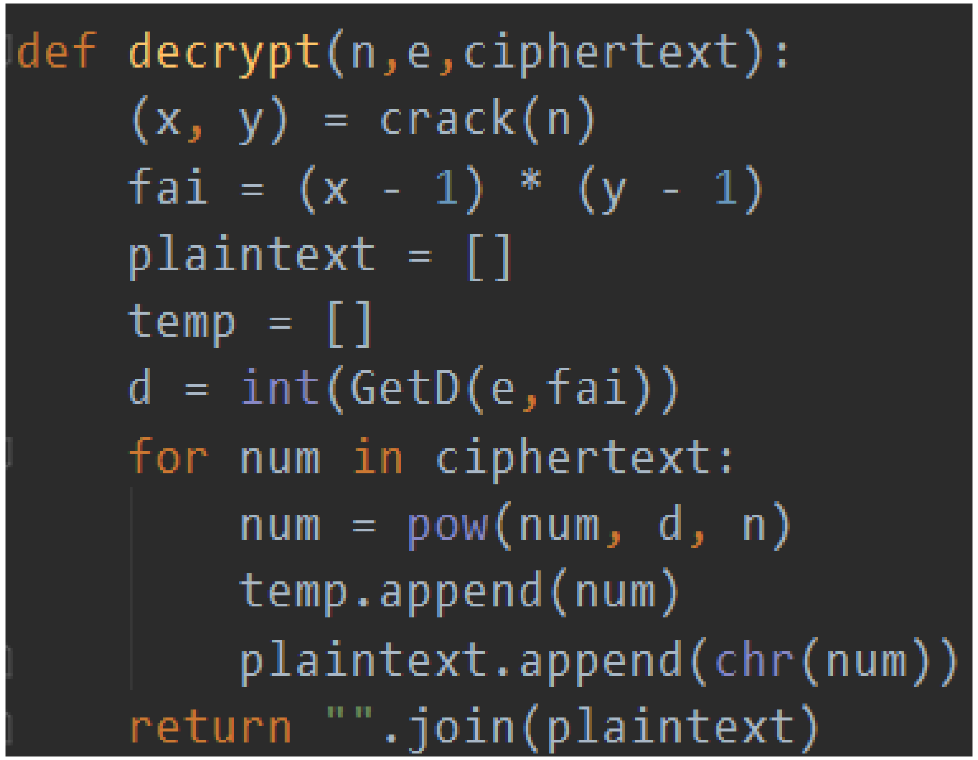
- Private key encryption principles and common algorithms such as DES;
- Public key cryptography ideas and common algorithms such as RSA;
- The differences between private and public key encryption, and the reasons in them;
- The process of key exchange protocol DH and its security issues;
- Other encryption methods that will be used in IoT, such as digital signatures, authenticated encryption algorithms (hash, SHA, MD, etc.)

exercise question

- Decrypt [167, 1915, 130, 591, 1988, 1235, 1032, 2088, 825] with RSA (each number corresponds to a letter)
- Public key (n,e)=(2449,7)
- Seek to crack the private key d and decode the plaintext.

Tips

- Procedural framework:



```
def decrypt(n,e,ciphertext):
    (x, y) = crack(n)
    fai = (x - 1) * (y - 1)
    plaintext = []
    temp = []
    d = int(GetD(e,fai))
    for num in ciphertext:
        num = pow(num, d, n)
        temp.append(num)
        plaintext.append(chr(num))
    return "".join(plaintext)
```

crack(n) performs a prime factorisation of **n**

GetD(e,fai) calculates the value of **d**

End of chapter