Wireless Network Security and Privacy

Information Security Basics

Xiaoyu Ji

Department of Electrical Engineering

Zhejiang University

2026 Autumn

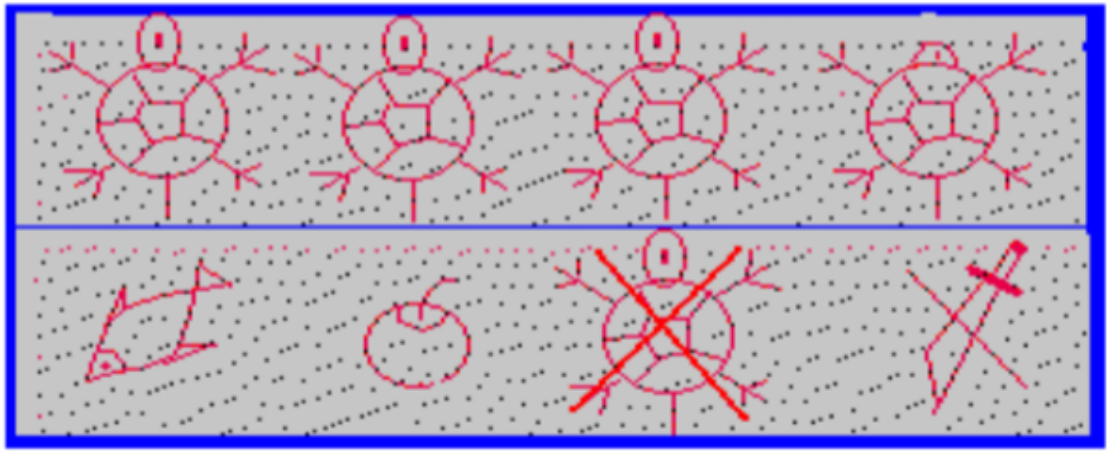
Outline

- Cryptography development
- Classical cryptography:
  - Caesar cipher
  - Affine cipher
  - Vigenère cipher
- Modern cryptography
  - DES
  - AES
  - RSA
  - Hash, MAC, etc.

What is cryptography?

The Art of Ancient Chinese Poetry

- A letter from a wife staying at home to her husband working away in ancient China
- What does this man? “Turtle, turtle, turtle, turtle hiding its head, fish, apple, knife…”
- It means: “Back, back, back, back quickly, if not, we will cut ties”



归，归，归！速归！如果（鱼果）不归，一刀两断

The Art of Ancient Chinese Poetry

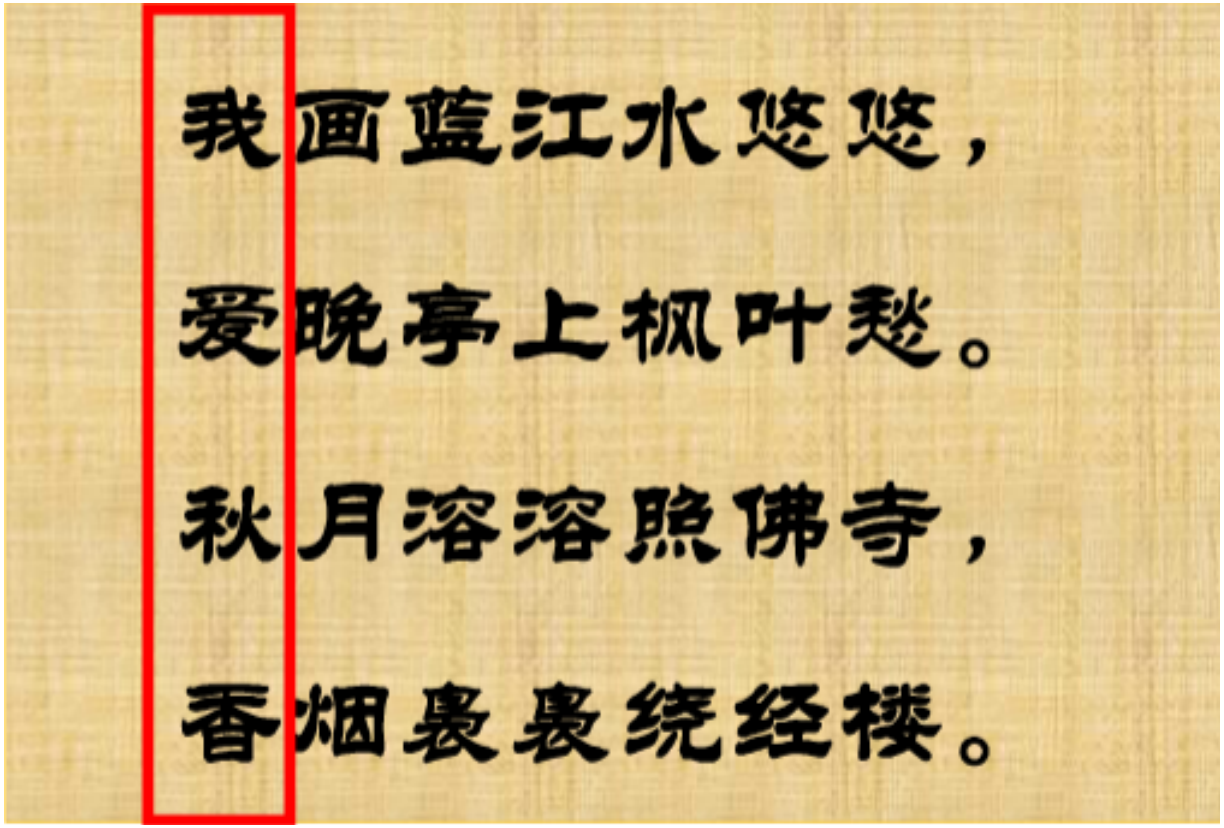
- **Associative poetry 会意诗**



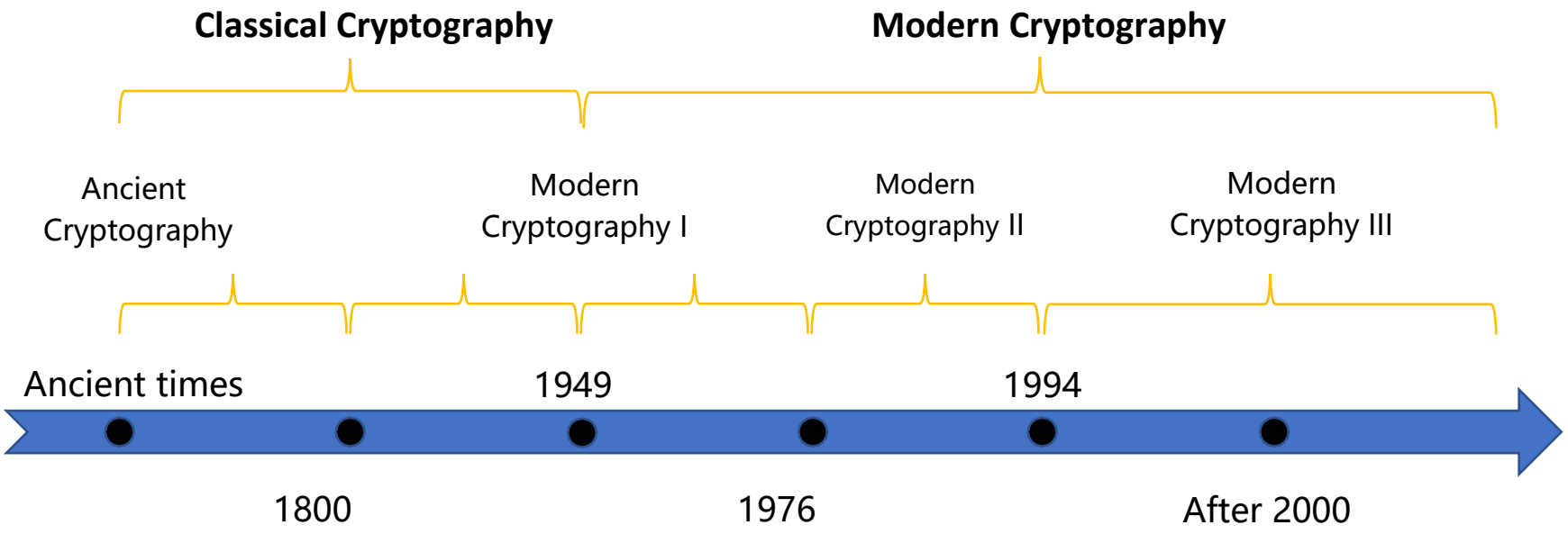


The Art of Ancient Chinese Poetry

- **Acrostic Poem 藏头诗**



Cryptography Timeline



Classical Cryptography

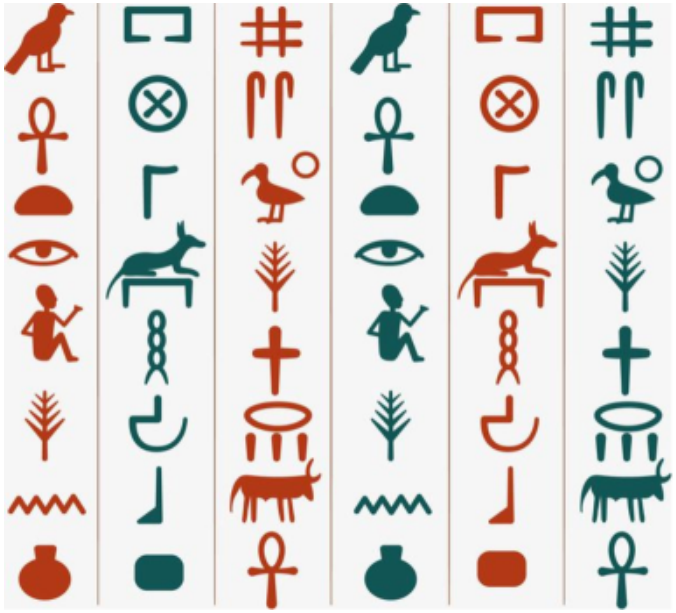
- **Time:** Ancient times - 1949
- **Features:**
  - Cryptography is not yet a science, but the art emerges as the basic means of cryptographic algorithm design (**substitution & permutation**)
- **Confidentiality:**
  - Data confidentiality **based on encryption algorithms**
- **Milestone Event:**
  - **Kerckhoffs's principle**: 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 **confidentiality** of the plaintext and the key…"

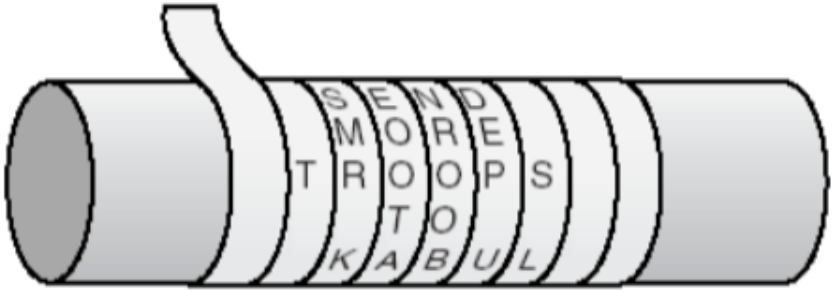
Substitution and Permutation

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



Ancient Egyptian hieroglyphics

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

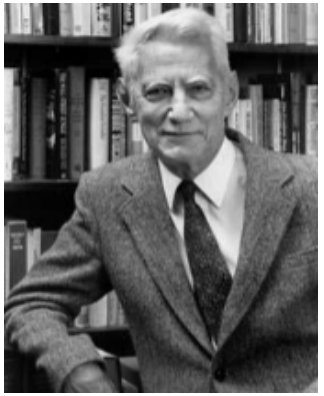


Ancient Spartans used the Book of Days

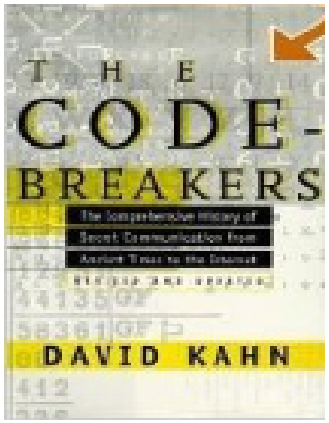
Wrap the parchment in the form of a long band around a round wooden stick and write on it; when the parchment is unwrapped, there is only a jumble of characters on it, and it can only be wrapped in the same way again around a stick of the same thickness

Modern Cryptography Phase I

- **Time:** 1949-1976
- **Features:**
  - Cryptography goes from art to science
- **Confidentiality:**
  - Data confidentiality based on secret keys instead encryption algorithms
- **Milestone Event:**
  - 1949, Shannon published "The Communication Theory of Secret Systems"
  - 1967, David Kahn published “The Code Breakers”



Shannon



The Code Breakers

Modern Cryptography Phase I

- **Shannon's contribution**
  - Defines theoretical security and proposes the principles of diffusion and **Obfuscation**.
  - 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.

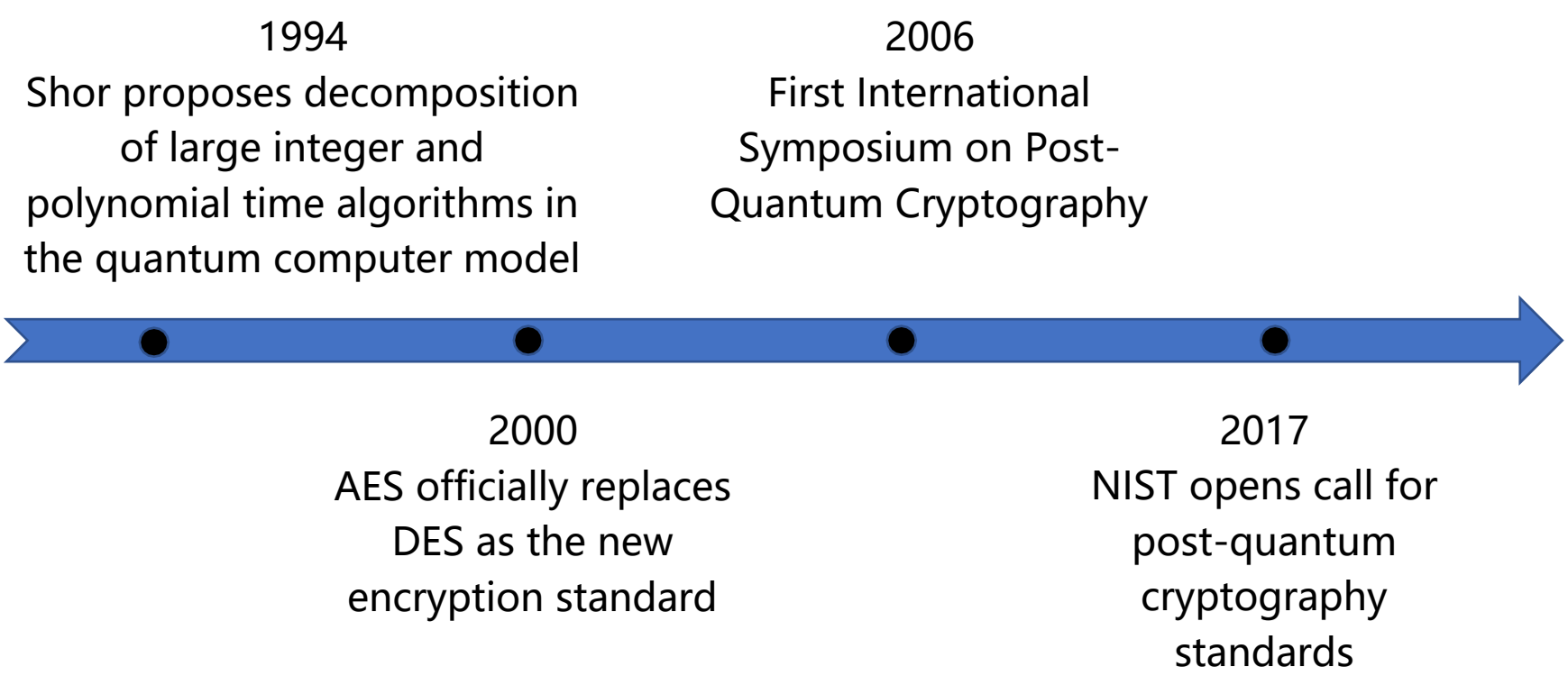




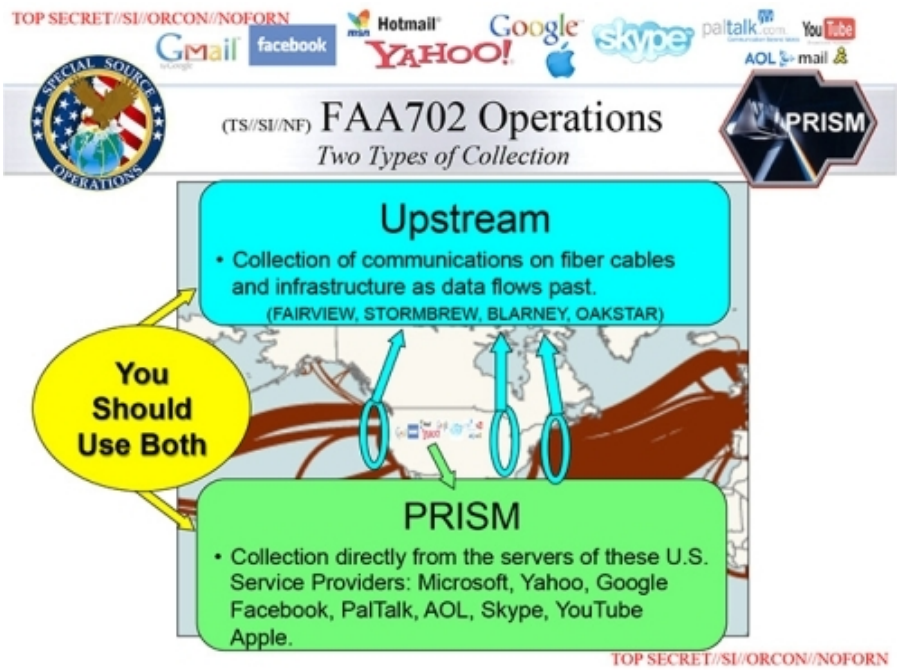
Diffie & Hellman won **the** Turing Award **in 2015**

Modern Cryptography Phase III

- The Era of Quantum Cryptography is coming



What will happen in The Era of Quantum Cryptography



RISM & UPSTREAM

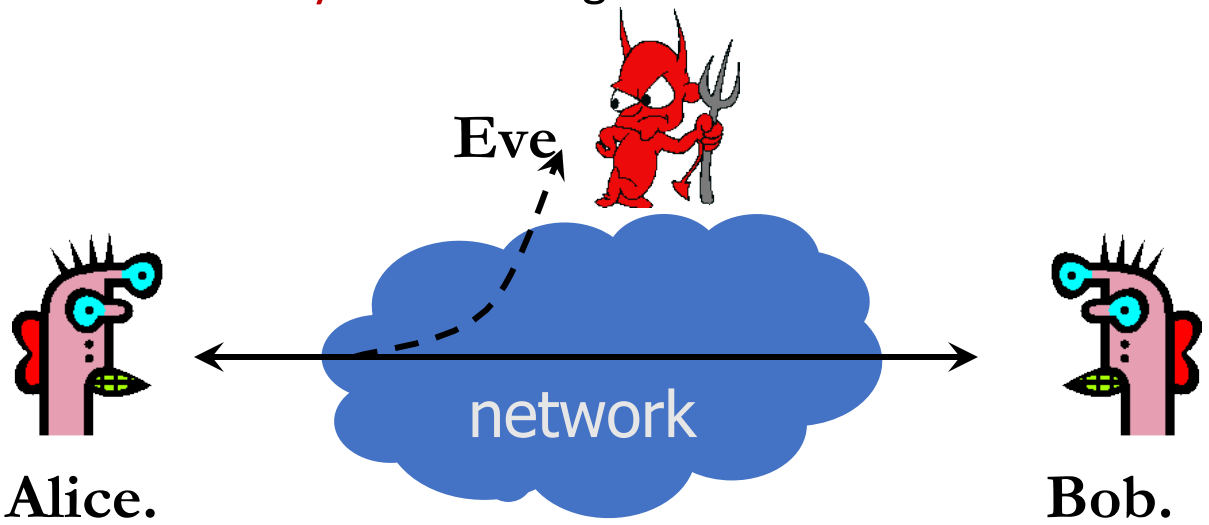


Edward Snowden

Information security basics

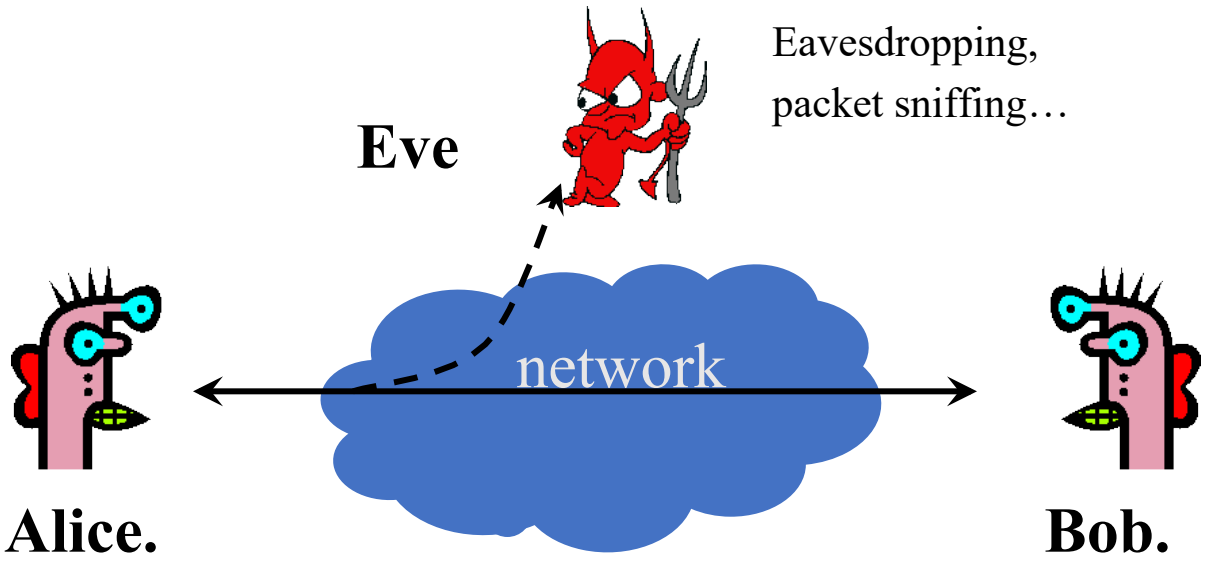
Information security objectives

- **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 information/data actually came from the source it claims to be.
- **Non-repudiation**: Ensuring that all participants in a message/data interaction cannot deny that messages and data were ever sent.



Confidentiality

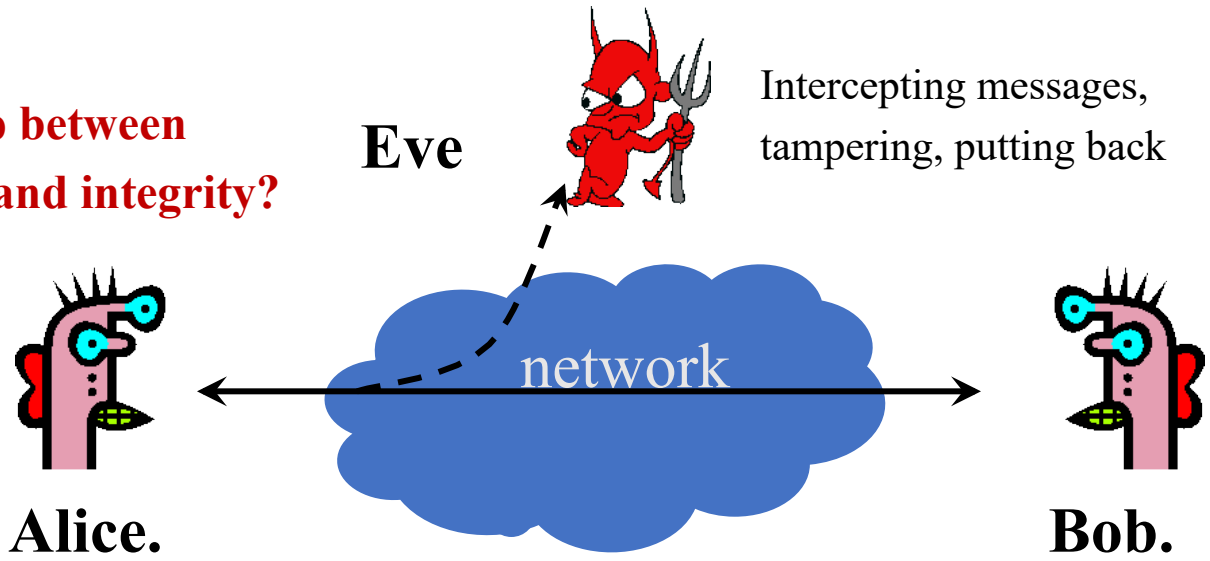
- Keeping sensitive information private and secure, ensuring that it is only accessible to authorized individuals or entities.
- **Solutions**: encryption and decryption algorithms.



Integrity

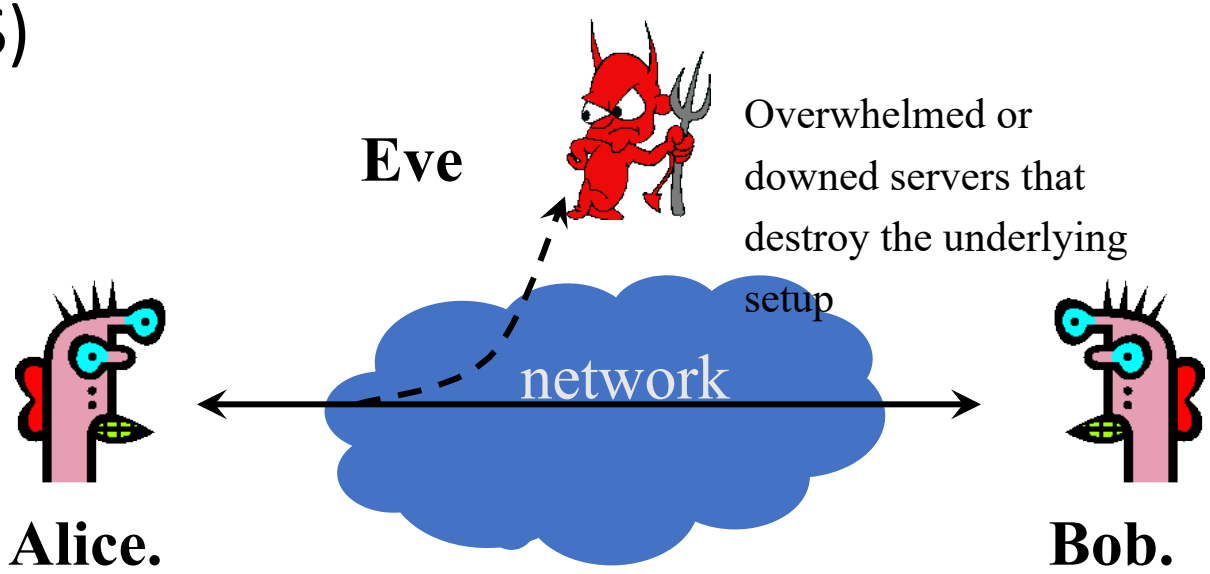
- refers to the accuracy, consistency, and reliability of data throughout its lifecycle. It ensures that data remains unaltered and trustworthy, protecting it from unauthorized modification, corruption, or loss.
- **Solutions**: Hash, Message Authentication Codes, etc.

Q: Relationship between confidentiality and integrity?



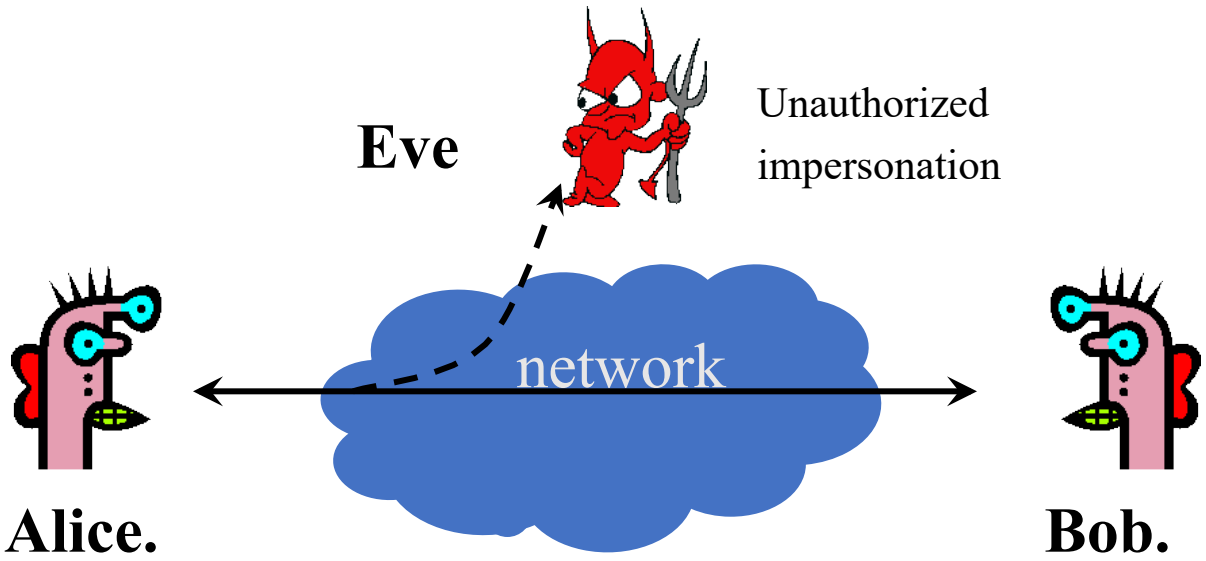
Availability

- refers to the accessibility of data when it is needed by authorized users. It ensures that information is readily accessible and usable, minimizing downtime and disruptions in access
- **Attacks**: Denial of Service (Dos), Distributed Dos (DDoS)



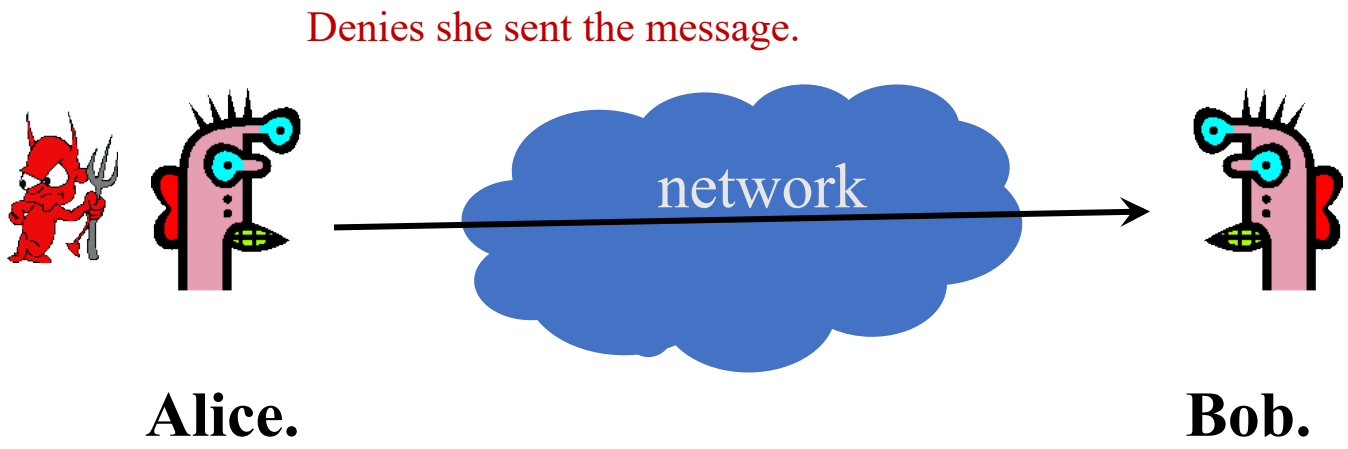
Authenticity

- refers to the assurance that the information originates from a legitimate source and can be verified as accurate and reliable
- **Solutions**: cryptographic schemes, digital signatures, hash functions, etc.



Non-repudiation

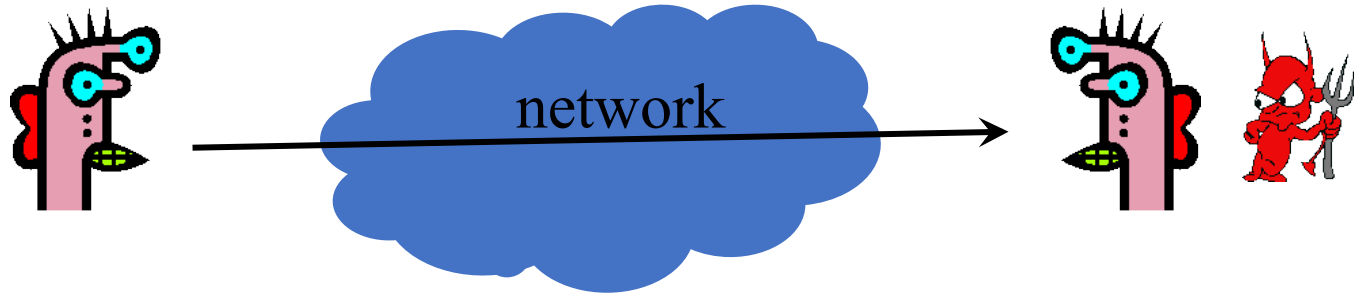
- the assurance that a party in a communication or transaction cannot deny the authenticity of their signature or the sending of a message.
- **Solutions**: Digital signature



Anonymity

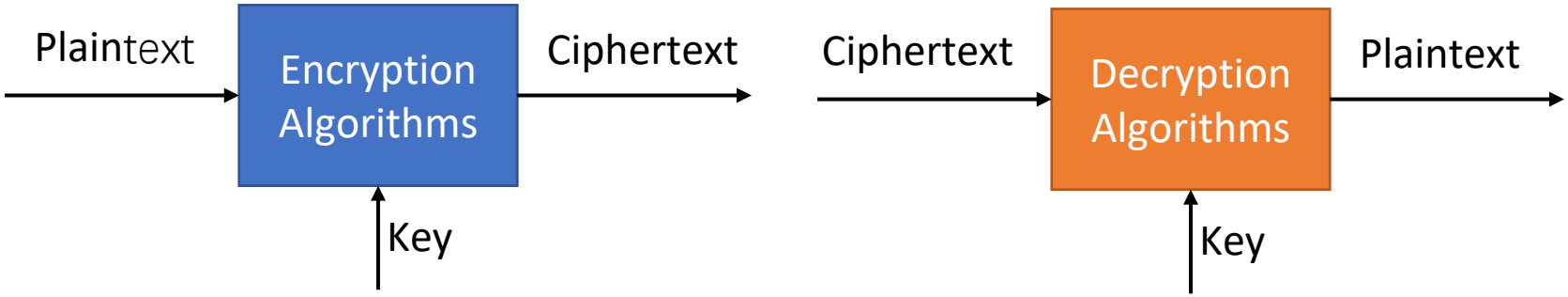
- refers to the practice that the information cannot be traced back to a specific individual, thereby safeguarding their privacy.

Bob identified Alice from the messages sent to him.



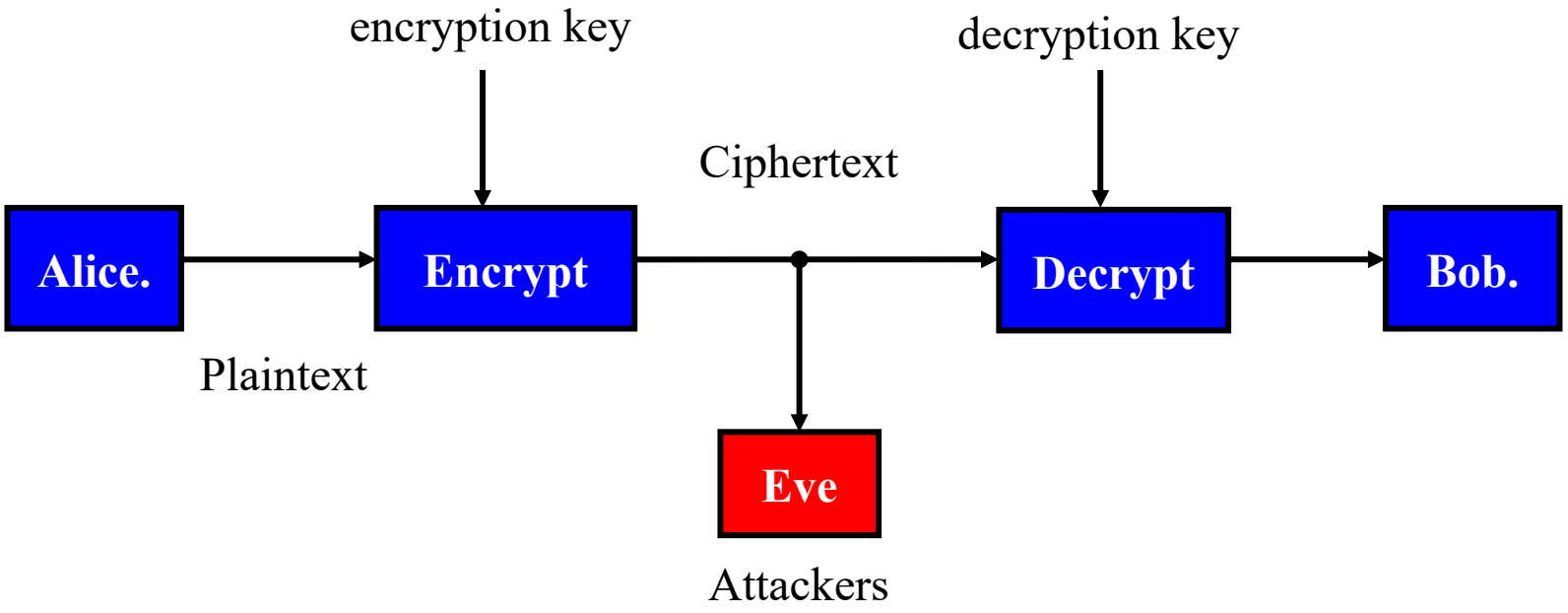
Basic concepts

- **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.
- **Key**: a set of secret information used in encryption and decryption
- **Encryption/decryption** algorithms: rules for encrypting/decrypting plain/cipher text

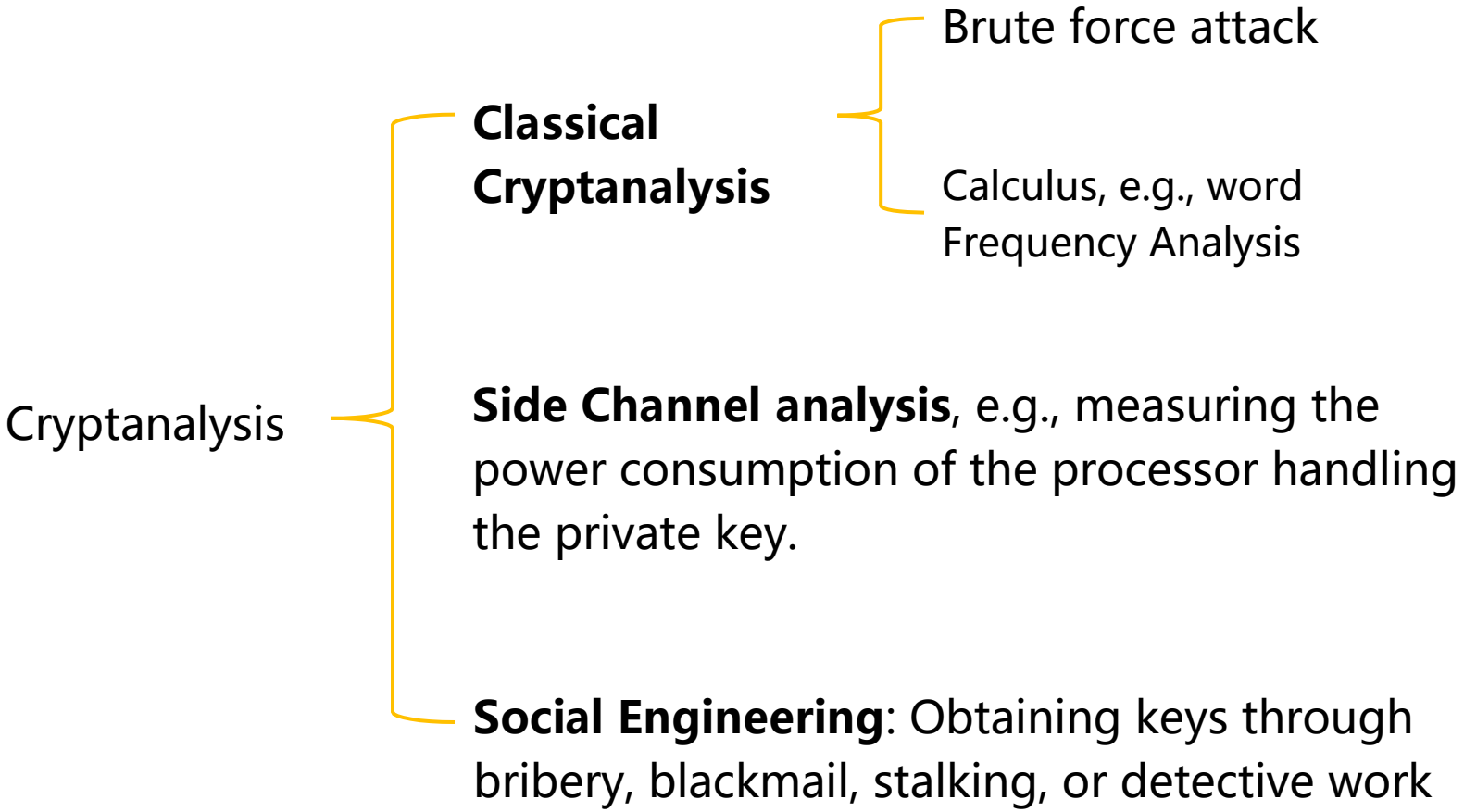


Cryptographic communication model

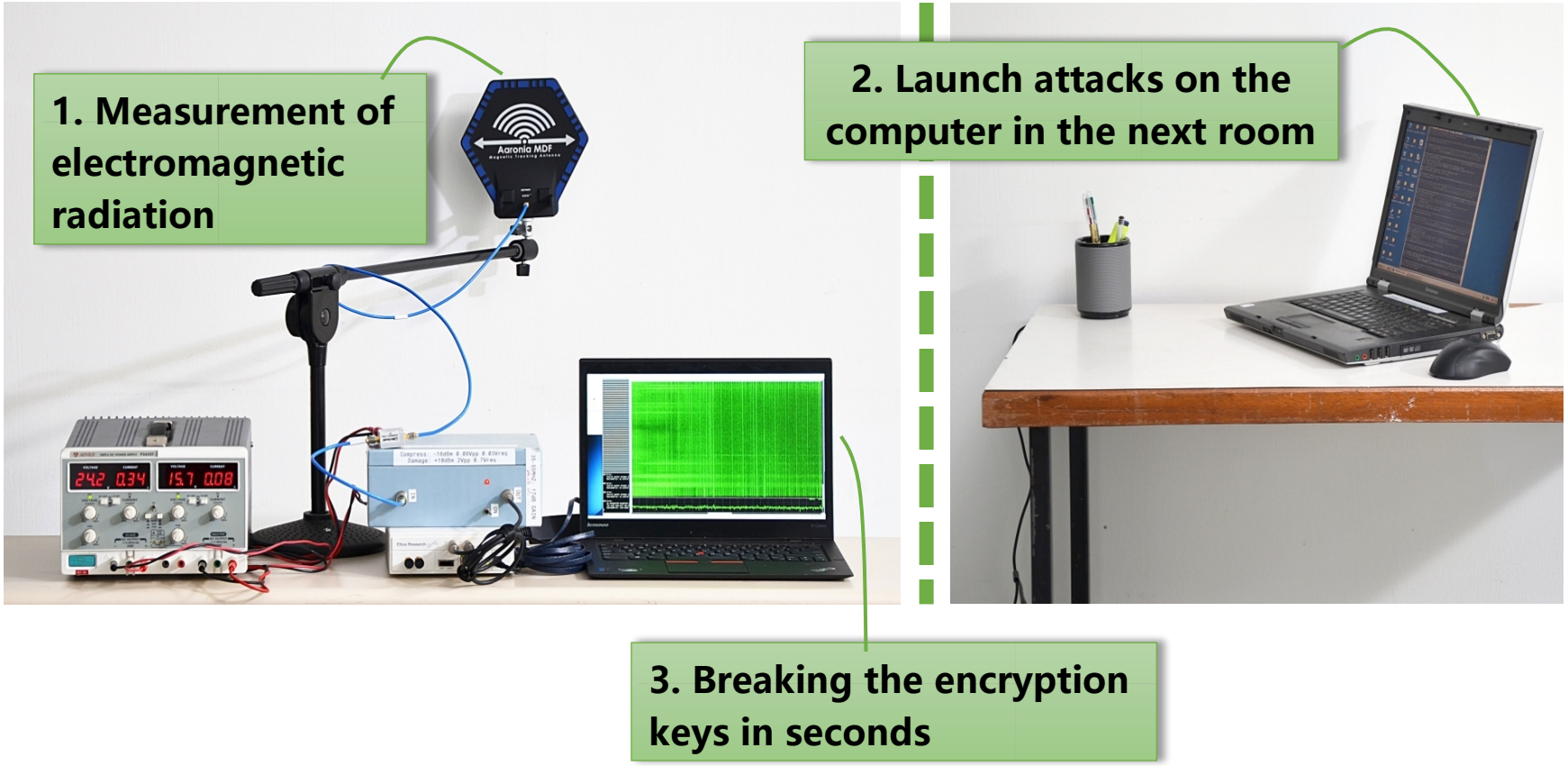
- Attacker can break the communication channel between Alice and Bob



Cryptanalysis



Side channel analysis



[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 manipulation technique that exploits human psychology to gain confidential information, access, or valuables. It often involves tricking individuals into revealing sensitive data, such as passwords, credit card numbers, or personal identification information.
- Common Techniques:
  - **Phishing**: Sending fraudulent emails or messages that appear to be from legitimate sources to trick individuals into providing sensitive information.
  - **Pretexting**: Creating a fabricated scenario to obtain information from a target, often by impersonating someone in a position of authority.
  - **Baiting**: Offering something enticing (like free software or a prize) to lure individuals into providing personal information or downloading malicious software.
  - **Tailgating**: Gaining physical access to restricted areas by following someone who has legitimate access.
- A famous hacker, Kevin Mitnick, is considered the master and pioneer of social engineering, who published "The Art of Deception"

Cryptographic Attacks 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

Classical Cryptography

How Caesar commands his army?



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

Military commands could be stolen.



A messenger conveys command to the army

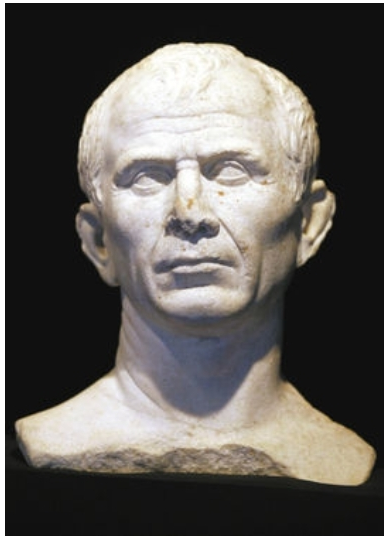


How did Caesar get military commands secretly delivered to the army?

Shift 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 Cipher 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

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

  - **Chosen-plaintext attacks**: the ability to choose your own choice of x. For example, choosing x='a'=0, then

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

  - **Chosen-cipher attacks**: choose the ciphertext y yourself. e.g. choose y='a'=0, then

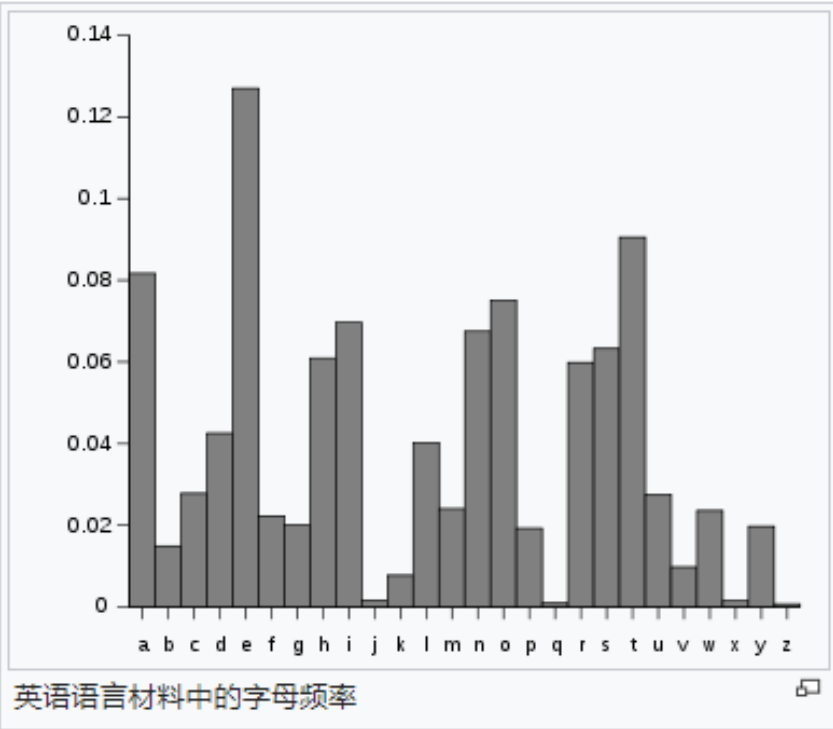
$$x = -k \pmod{26}$$

Shift Cipher Security

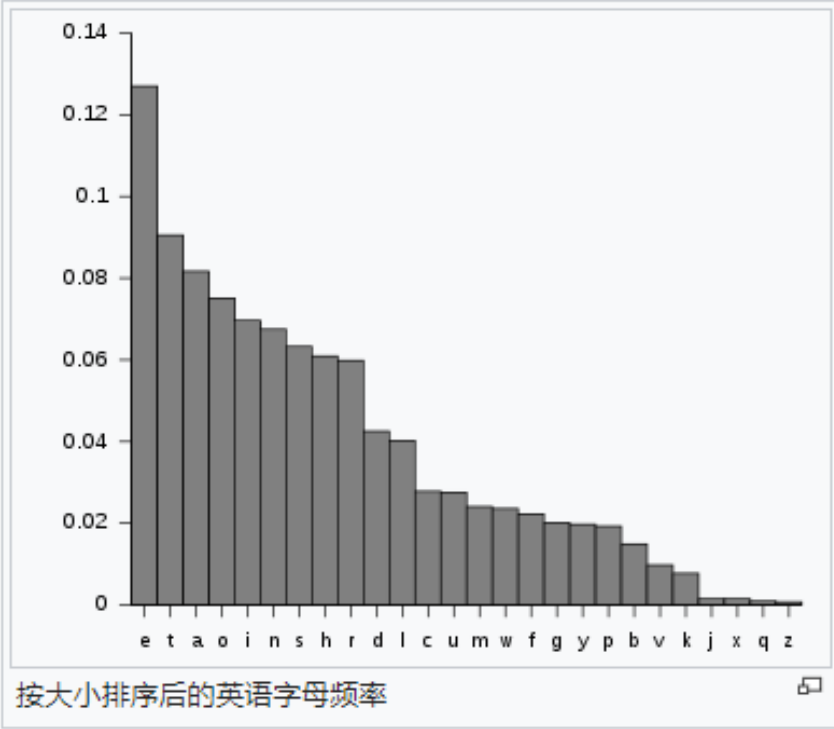
- How to attack if you only know the ciphertext?
- Try all 26 possibilities?

- We have a smarter way: **Word frequency statistics**
  - Calculating the frequency of letters in ciphertexts
  - Comparison the calculated frequency of a character with the frequency of occurrence of standardized linguistic letters
  - Determination of the most probable value of key k

Word frequency statistical



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

Affine Cipher

- Encryption: given the key (α,β)

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

- Decryption:

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

- when referring to numbers in modular arithmetic: Inverse element of a is the value of b that satisfies ab = 1 (mod 26), for example, when a=7,n=26, 15 is the inverse element of 7 mod 26.

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 ...
7*15=1 mod26	……

Affine Cipher Security

- for α, if there exists an inverse element with respect to modulus n, then $\gcd(\alpha,n)=1$
- In order to obtain the inverse element, it is necessary to make $\gcd(\alpha,26)=1$, 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 choices, and in addition β takes on a range of 26, so for affine encryption, there are only 12 * 26 = 312 possibilities for all!

- Attacks on Affine Cipher :
  - Brute force attack: try all 312 possibilities
  - Word frequency statistics: think about what it takes?

Example of affine encryption

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 Cipher attack

- Affine cipher is a linear mapping, so the plaintext and the corresponding ciphertext appear with the same frequency.
  - **Step 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}

  - **Step 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%

  - **Step 3**. Determining mapping relationships

e→n; t→o

Affine Cipher attack

- **Step 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
```

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.

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

Virginia Cipher

Encryption method:

- List plaintexts and group them by key length
- Apply shift encryption of each grouped letters with a key
- Encryption formula:

  C=(P+K)mod26

Features:

- Virginia cipher is actually an extension of the shift cipher.
- Able to eliminate frequency characteristics of letters

Q:Think about why?

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

Wireless Network Security and Privacy

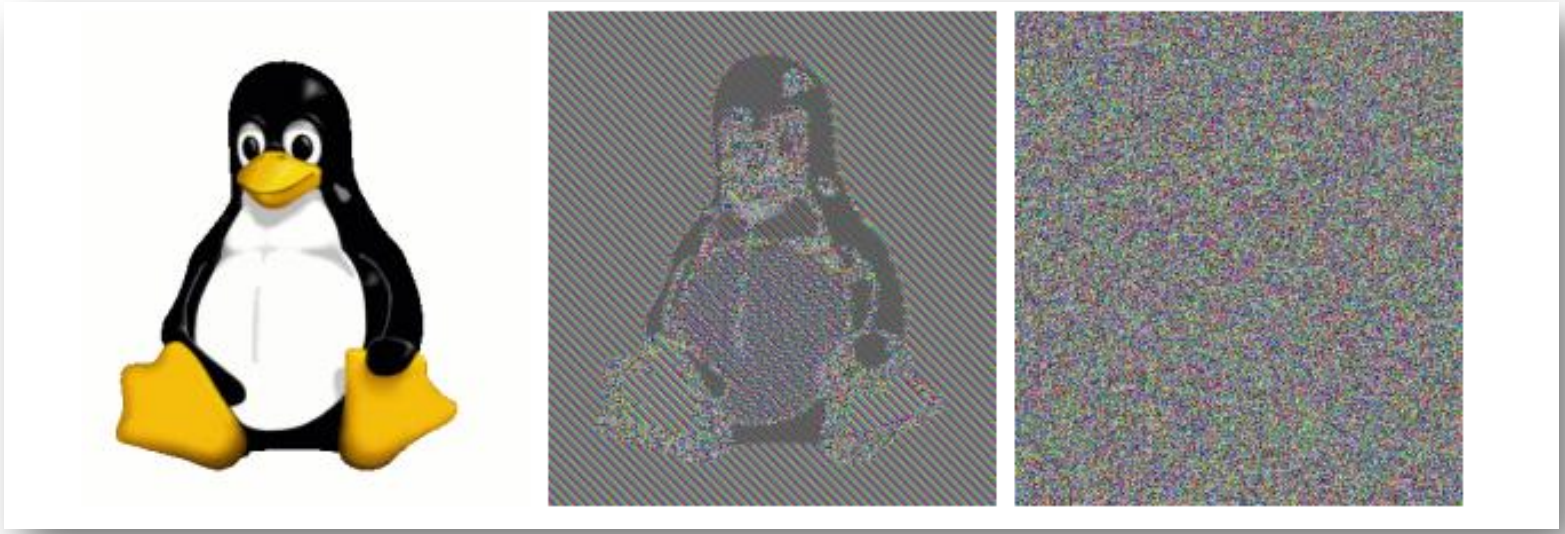
Modern Cryptography

Outline

- **Basics**
- **Private key cryptography (symmetric cipher)**
  - DES
  - 3DES
  - AES
- **Public key cryptography (asymmetric cipher)**
  - public key cryptosystem
  - DH key exchange protocol
  - RSA encryption
- **Cryptography Applications in IoT**
  - cryptographic application
  - Authentication Applications
  - Secret key management

Obfuscation and Diffusion

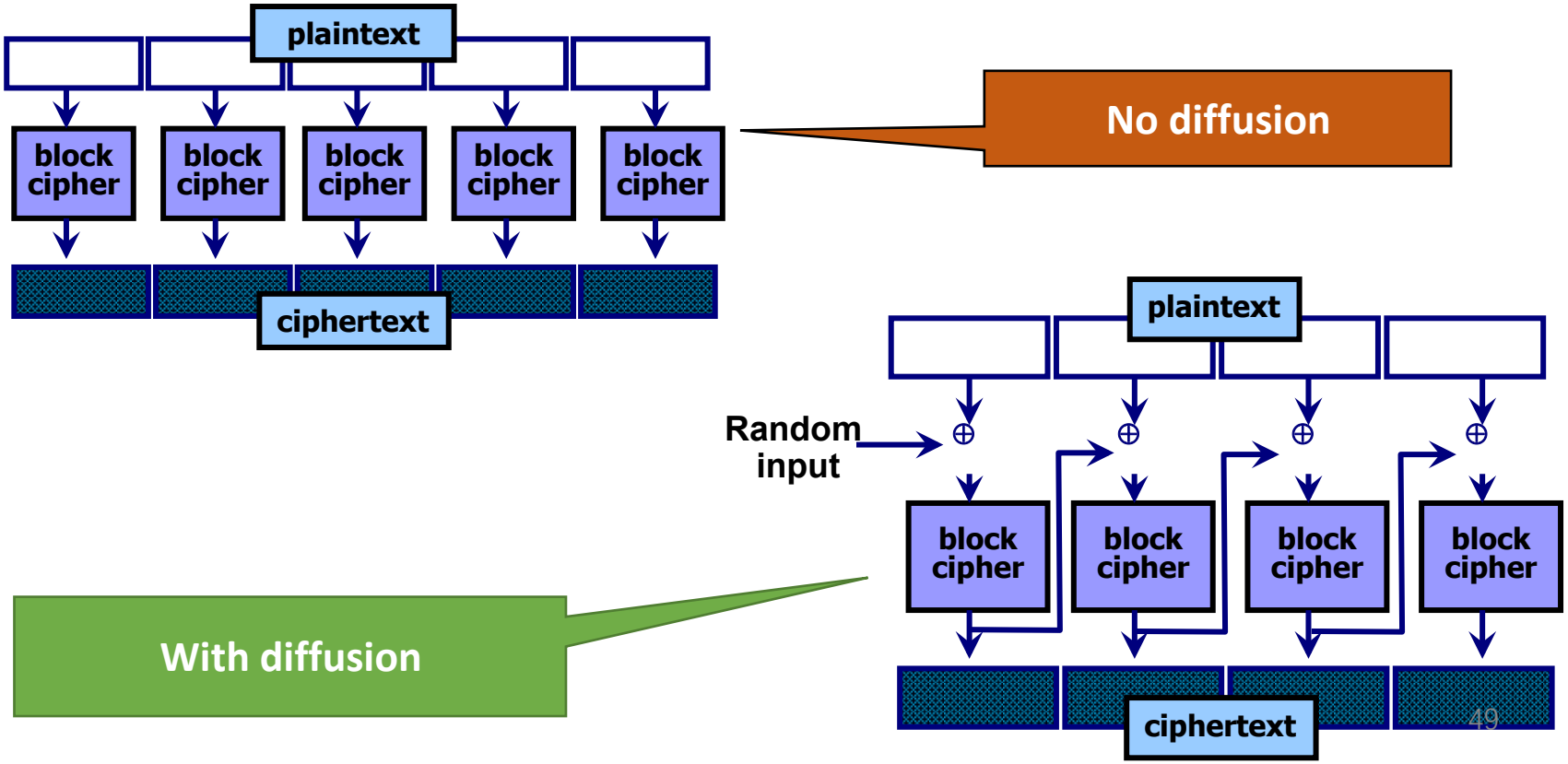
- **Obfuscation**: A cryptographic operation that makes the relationship between the key and the ciphertext as ambiguous as possible, used in both DES and AES.



The original text (left), the poorly obfuscated encryption (center), and the well obfuscated encryption (right)

Obfuscation and diffusion

- **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. Usually modifying 1 bit in the plaintext will result in an average of half of the output bits being changed.



Obfuscation and diffusion

- S-box (Substitution-box): is the basic structure of a symmetric-key encryption algorithm performing substitution computations.
- 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.

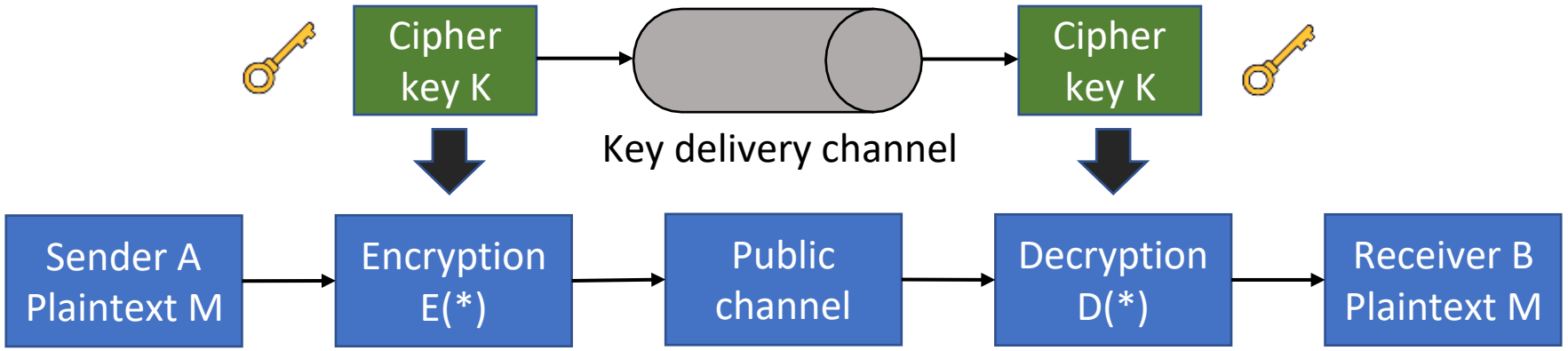
S_5		中间四个比特															
		0000	**0001**	**0010**	**0011**	**0100**	**0101**	**0110**	**0111**	**1000**	**1001**	**1010**	**1011**	**1100**	**1101**	**1110**	**1111**
首尾比特	**00**	0010	1100	0100	0001	0111	1010	1011	0110	1000	0101	0011	1111	1101	0000	1110	1001
	01	1110	1011	0010	1100	0100	0111	1101	0001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

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.



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 encryption algorithm is secure. An attacker can steal a certain number of ciphertexts and the corresponding plaintexts, they cannot be deciphered; also the encryption algorithm should be public.
  - The communicating parties can 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 most important thing to symmetric encryption?

Confidentiality of the keys

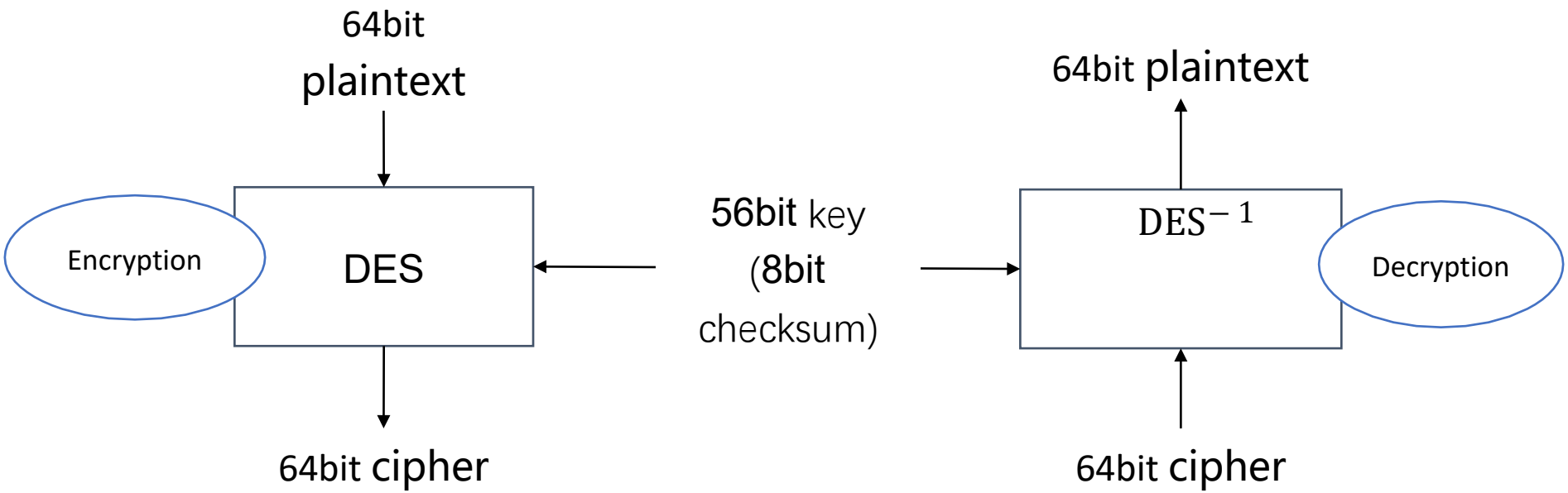
DES

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

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

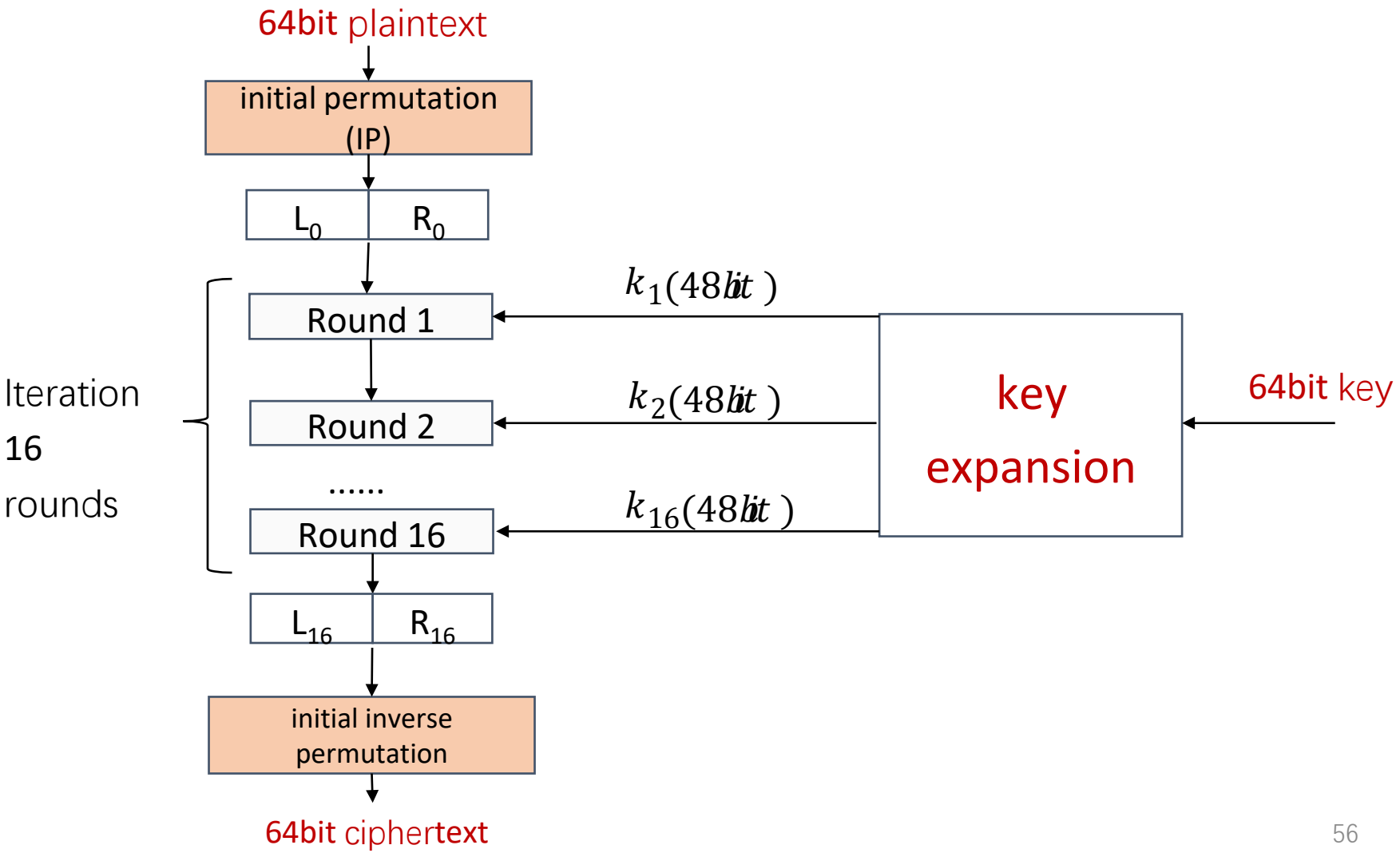
- History: recruitment and standardisation
  - In 1972 the National Bureau of Standards (NBS) began a programme to develop computer data protection standards.
  - In 1973, NBS open call for computer data encryption algorithms
  - In 1977, NBS adopted a modified version of IBM's Lucifer algorithm as the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

DES encryption diagram



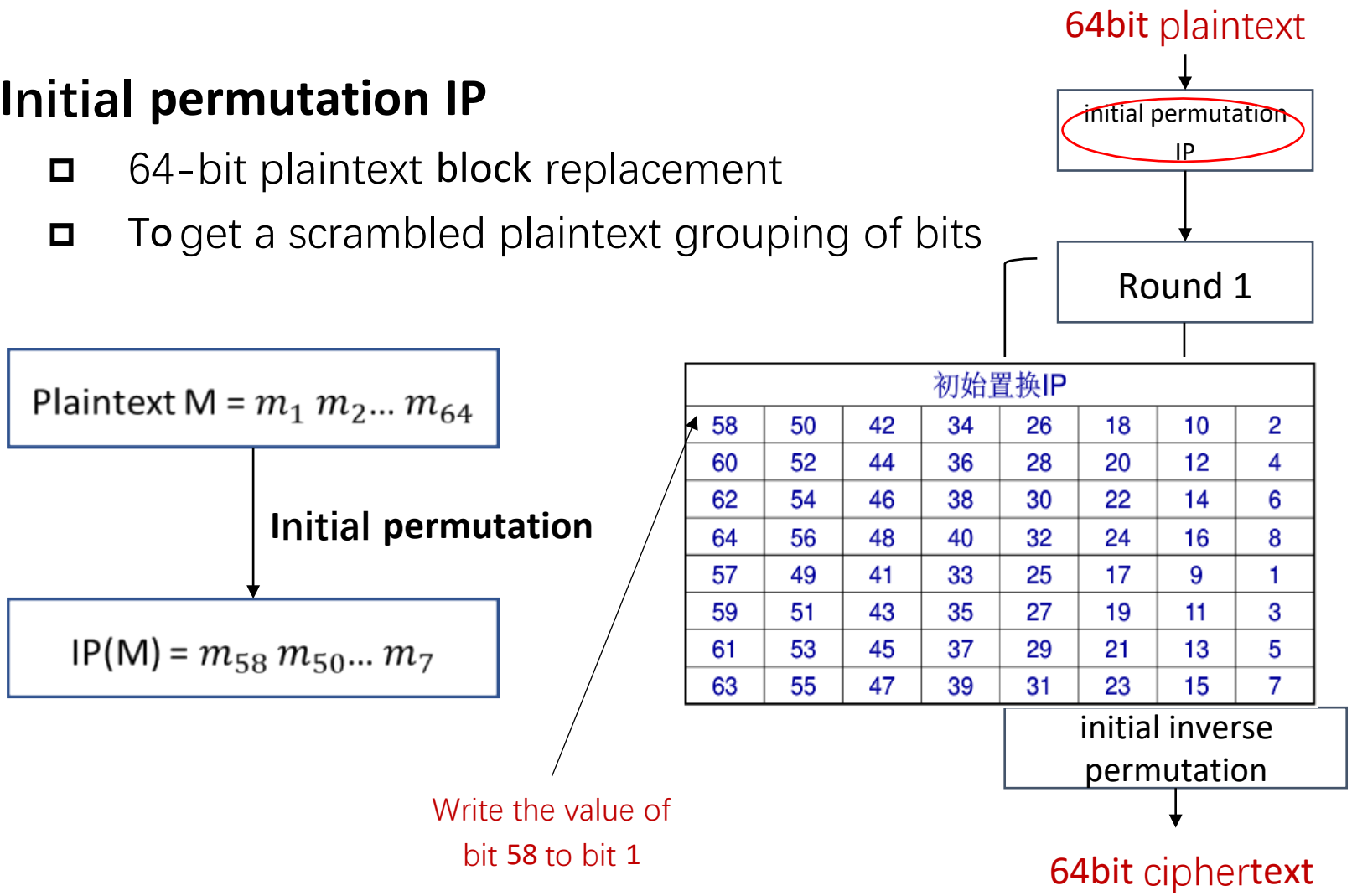
Block length: 64 bits
Key length: 64 bits
Valid key length: 56 bits

DES encryption process

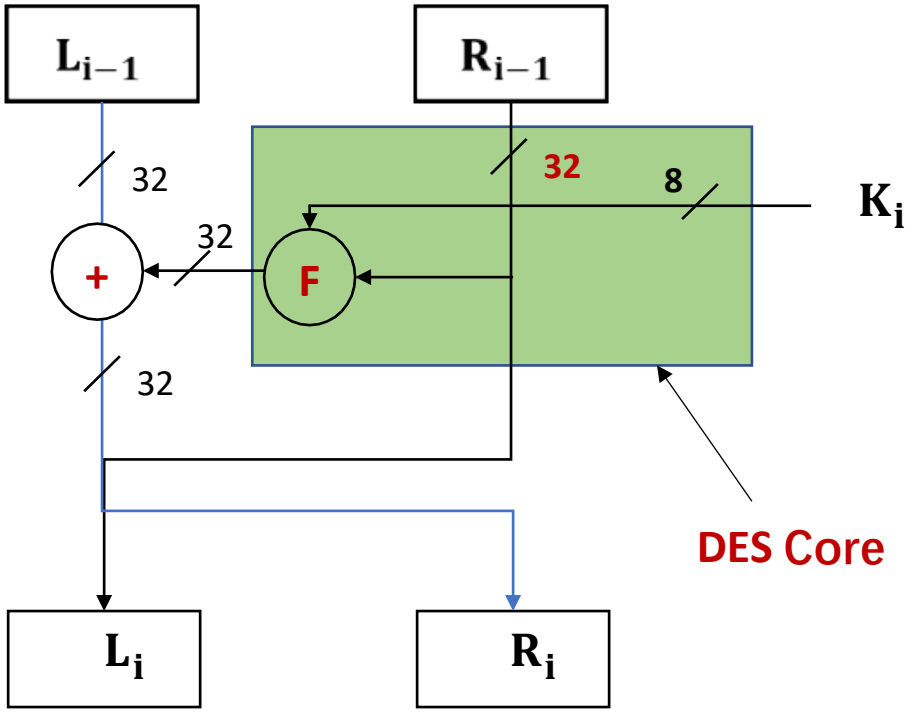


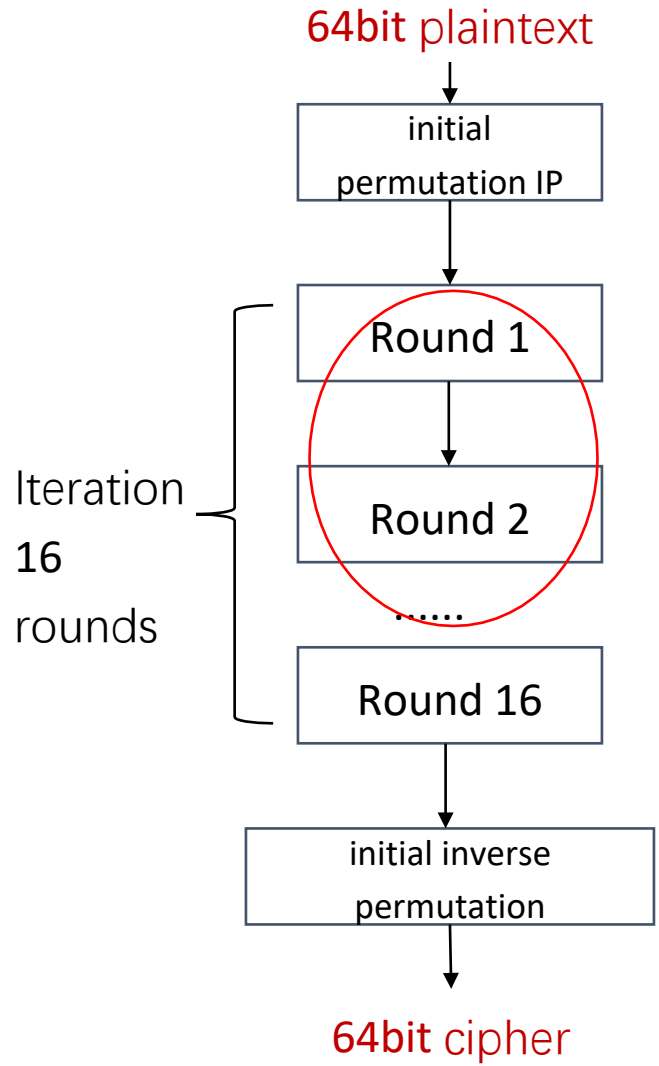
DES encryption - Permutation

- **Initial permutation IP**
  - 64-bit plaintext **block** replacement
  - To get a scrambled plaintext grouping of bits



DES Encryption - In-round Encryption

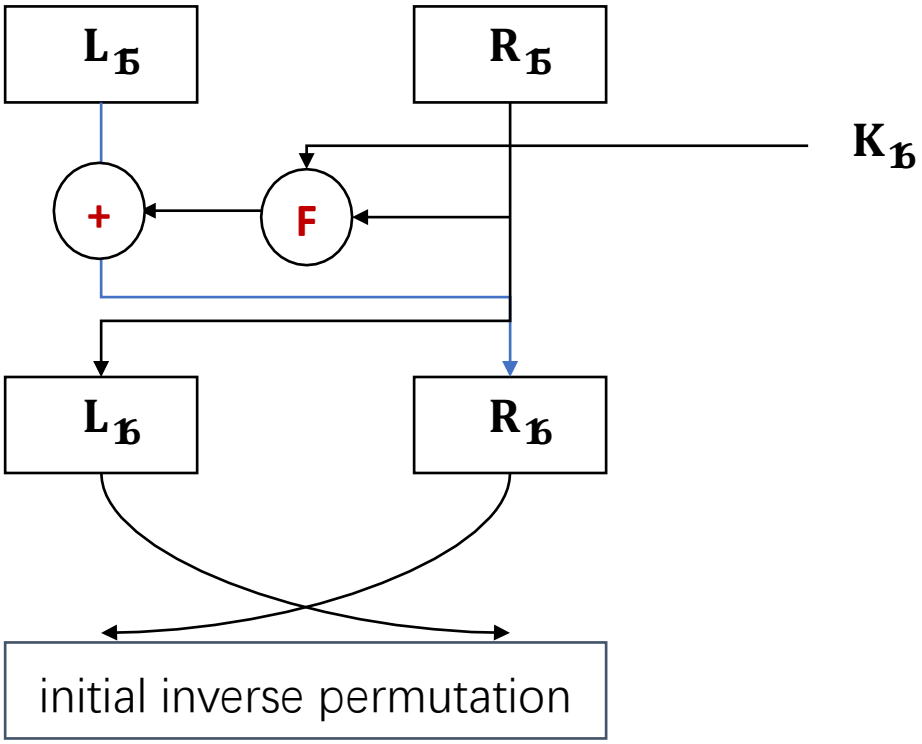


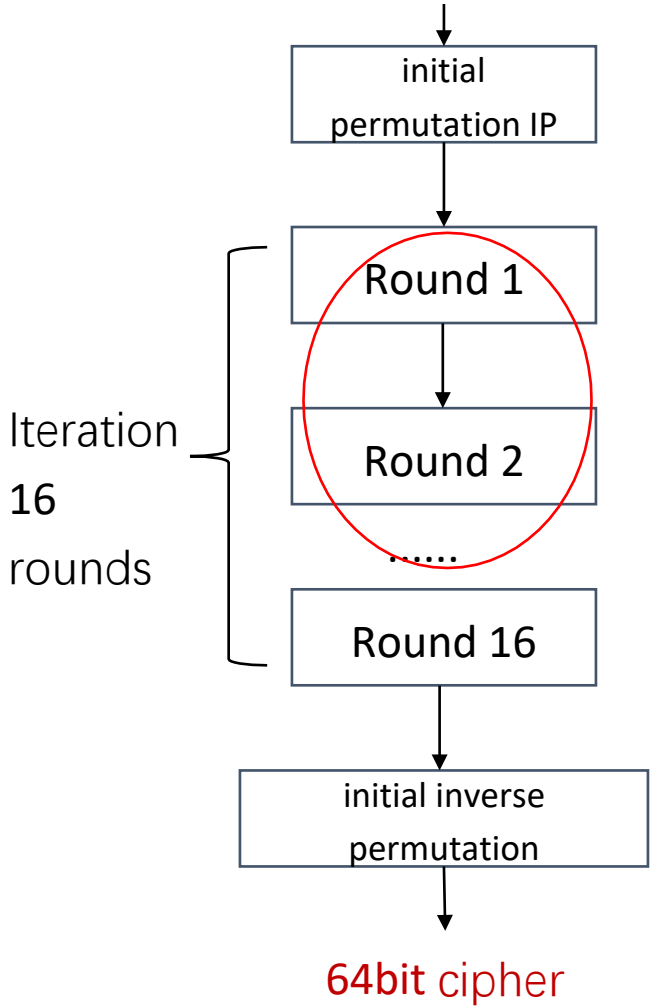


Q: How do 48-bit and 32-bit numbers operate?

DES Encryption - In-round Encryption

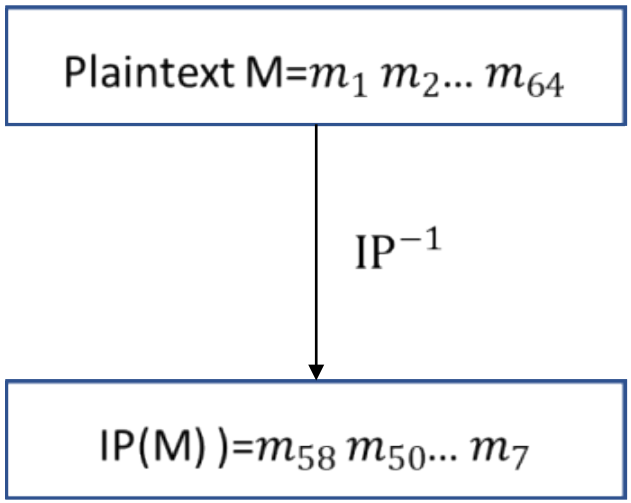
- **Iteration 16**

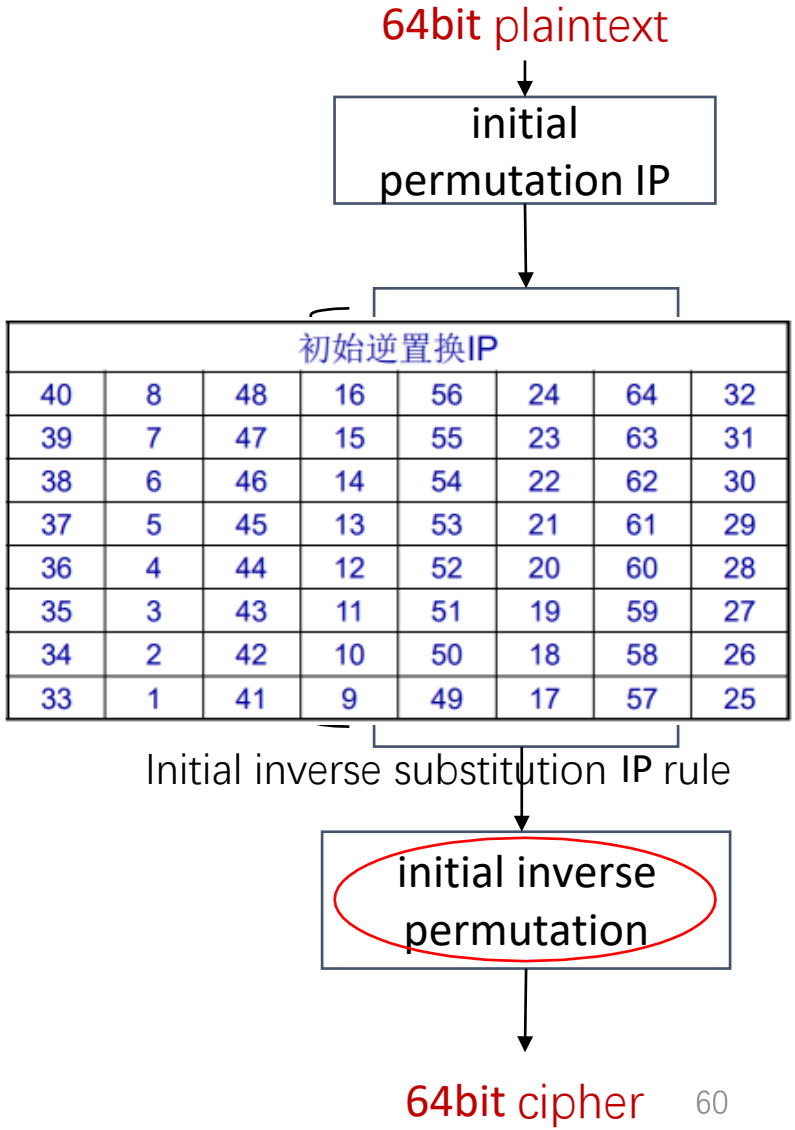




DES Encryption - Initial Inverse Permutation

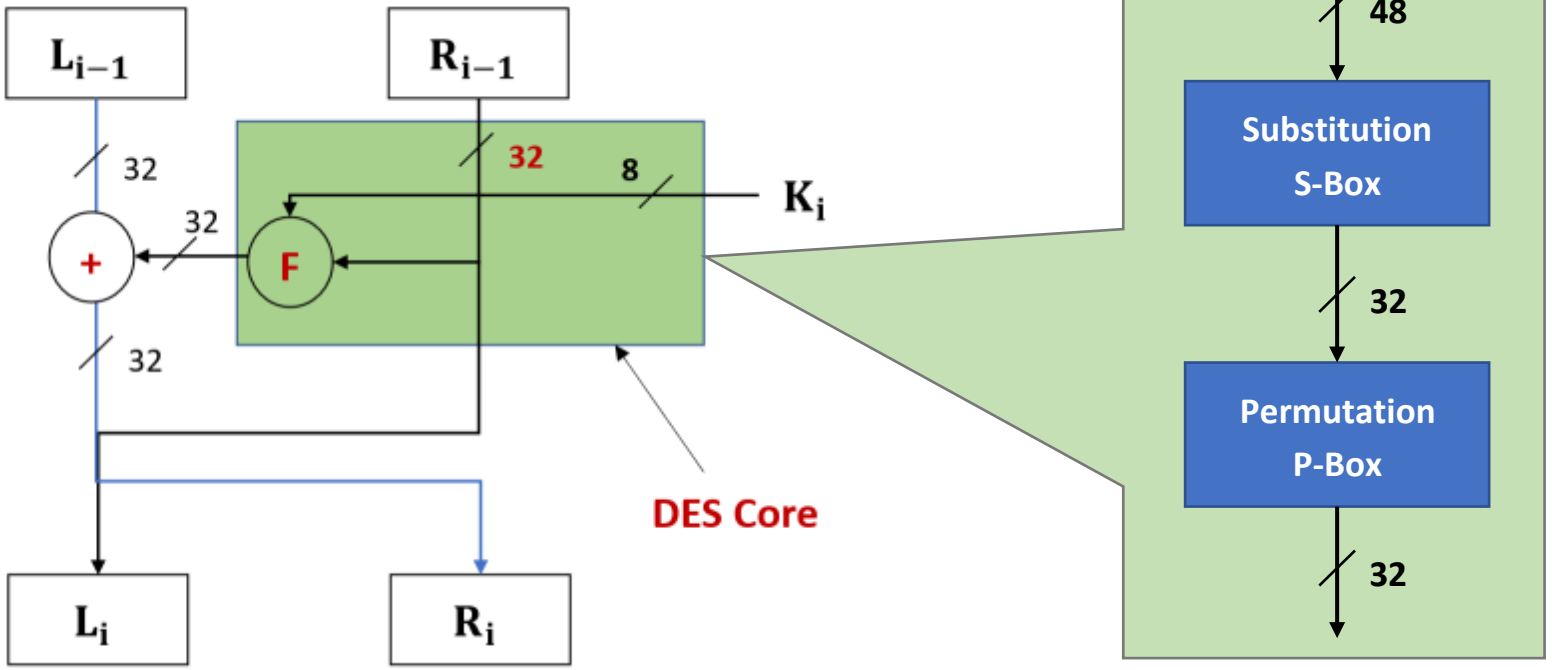
- **Initial inverse permutation (IP^{-1})**
  - Inversion of initial permutation IP
  - $IP^{-1}[IP(M)] = M$





The core of DES - F function

- The F function achieves:
  - Obfuscation
  - Diffusion



DES Example

- **Plaintext**: 30 31 32 33 34 35 36 3716
- **Key**: 31 32 33 34 35 35 36 37 3816
- Calculate its ciphertext

DES Example

- **1. Initial IP replacement**

30 31 32 33 34 35 36 37

00110000 00110001 00110010 00110011 00110100 00110101 00110110 00110111



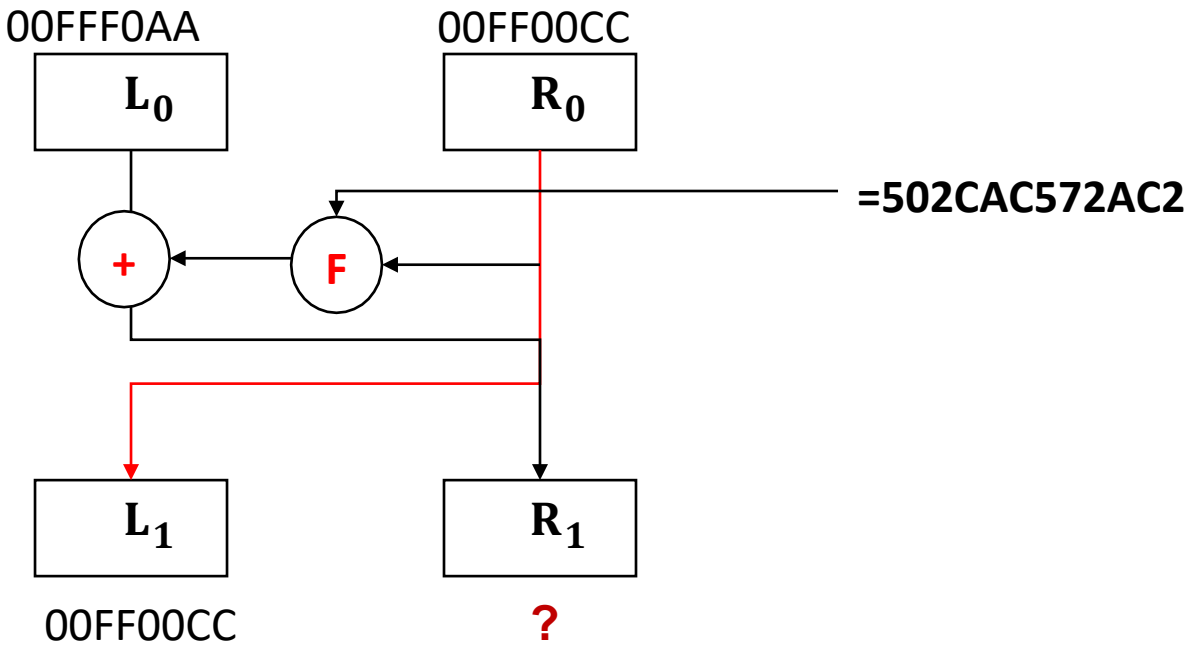
初始置换IP							
58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

00000000 11111111 11110000 10101010 00000000 11111111 00000000 11001100

00 ff f0 aa 00 ff 00 cc

DES Example

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



DES Example

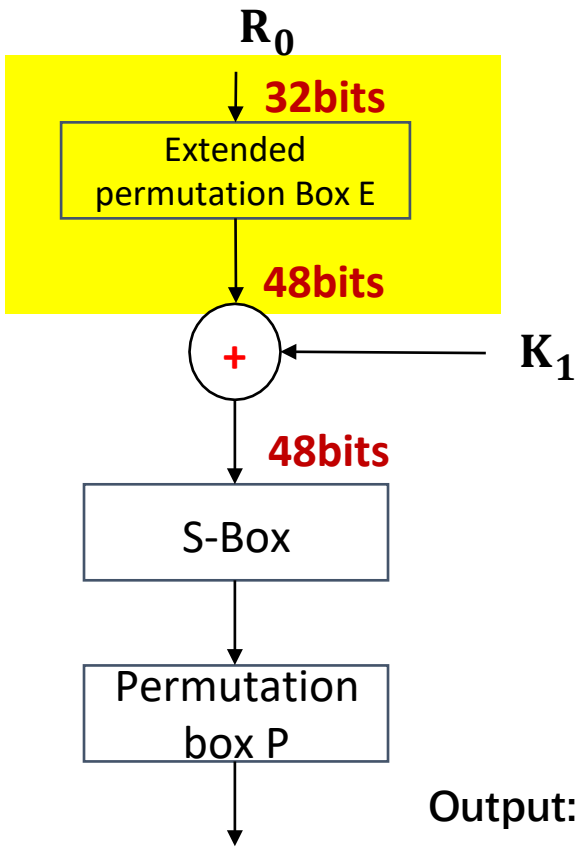
R0=00FF00CC

Input: **0000 0000 1111 1111 0000 0000 0000 1100 1100**

Row 1 Row 2 .....

■ 3. F-function:

- **Extended permutation box E**



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

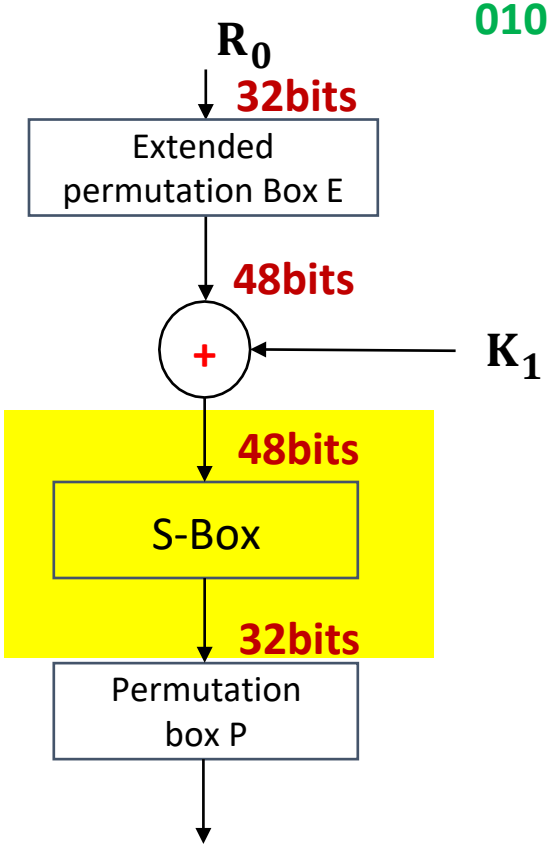
Extended Replacement E					
32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12		14	15	16	17
		8	19	20	21
		2	23	24	25
		6	27	28	29
28	29	30	31	32	1

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

Output: **000000 000001 011111 111110 100000 000001 011001 011000**

DES Example

- **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, and the tuple is the resulting value

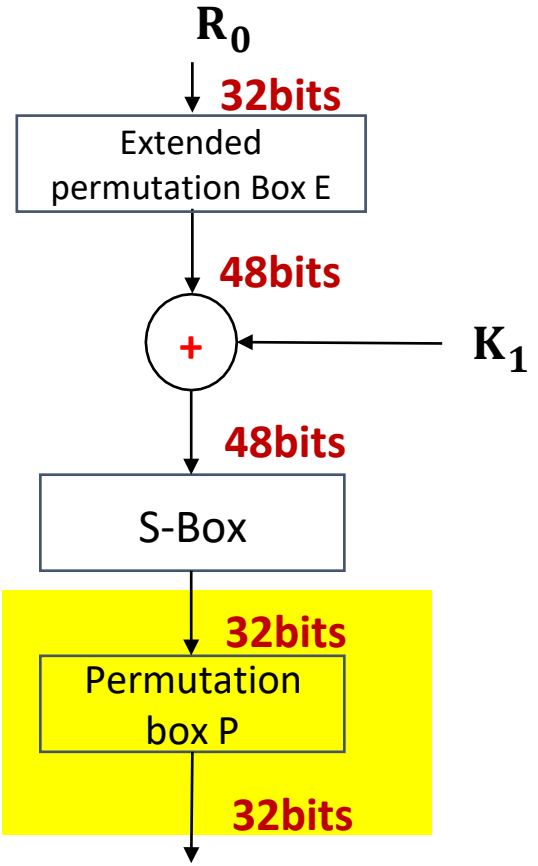
010100 000011 101101 010010 110101 110011 110010 011010

	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 Example

- 5. F-function:
  - Permutation box P



7C78EE91

0110 1111 0001 1010 0011 1011 1100 1001

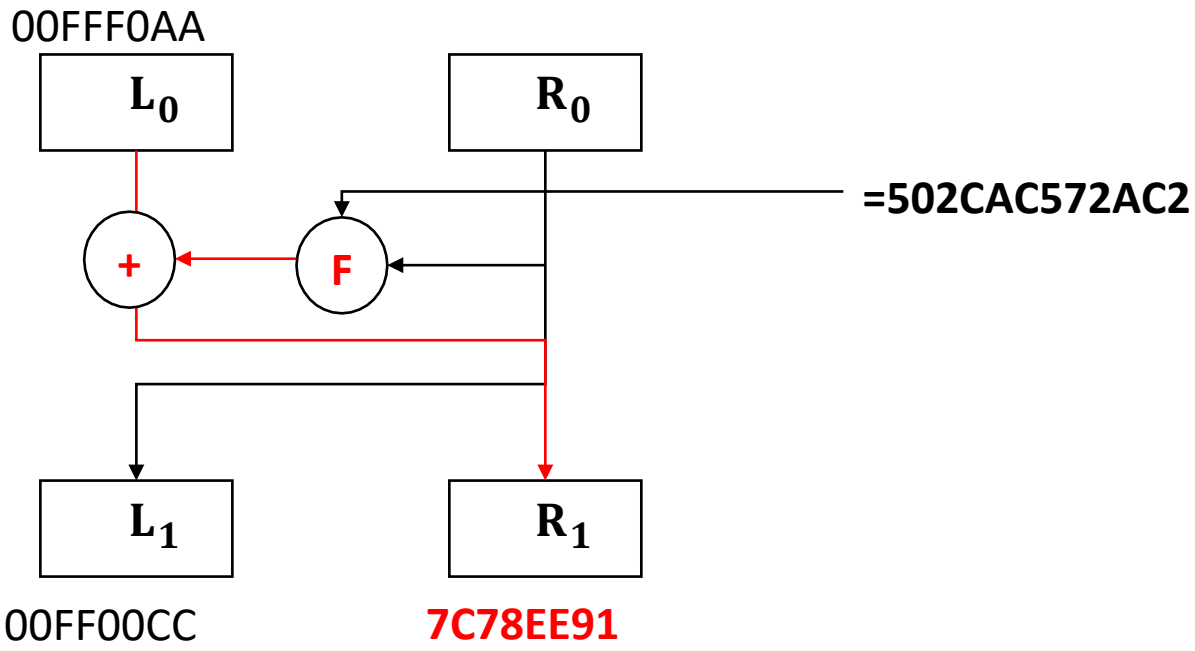
permutation P			
16	7	20	21
29	12	28	17
1	15	23	26
5	18	31	10
2	8	24	14
32	27	3	9
19	13	30	6
22	11	4	25

0011 1100 0111 1000 1110 1110 1001 0001

7C78EE91

DES Example

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



DES Example

- **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

Some thoughts

- **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.

- **In** DES, the decryption process is exactly the same as the encryption process

How secure is DES?

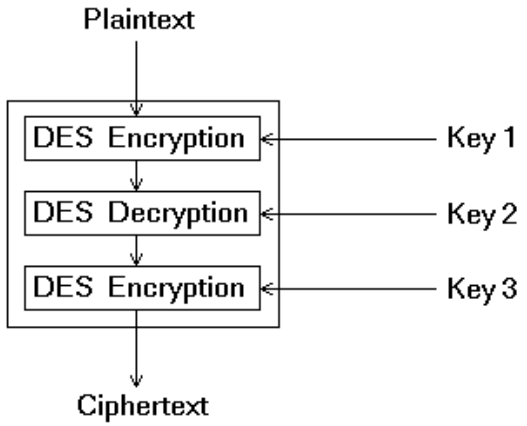
- **Analytical attack**: exploits correlation between keys, virtually nothing in practice

- **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

From DES to 3DES

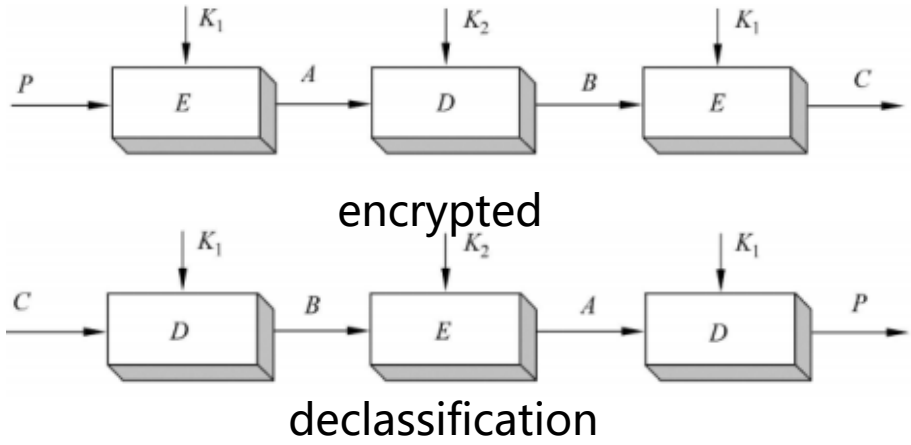
- **3DES**: Three consecutive DES encryptions, also called Triple DES.
- The three cipher components can be both an encryption and a decryption function.

- When $k_1 = k_3$ it becomes a double key 3DES
- Encryption: $y = E_{k_1}[D_{k_2}[E_{k_1}[x]]]$
- Decryption: $x = D_{k_1}[E_{k_2}[D_{k_1}[y]]]$



3DES

- 3DES encryption and decryption process



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

- 3DES Security
  - If **k_1 k_2 k_3** are not equal to each other, brute force attacks against the key length 56*3=168 take 5.8*10^29 years!
  - 3DES is still sufficiently secure currently

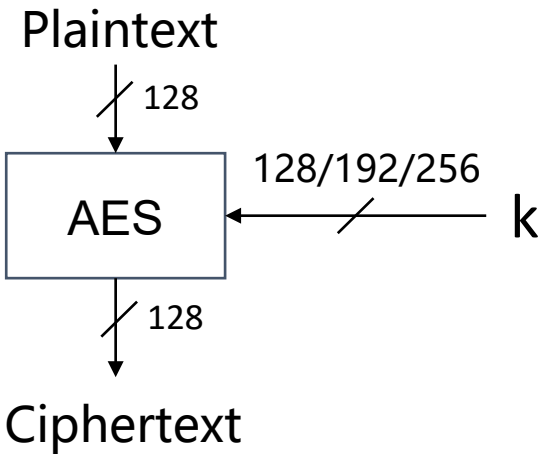
Problems with 3DES

- DES for legacy systems, 3DES for new systems

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

AES

- **Advanced Encryption Standard**: 128-bit **block**, 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-resource 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

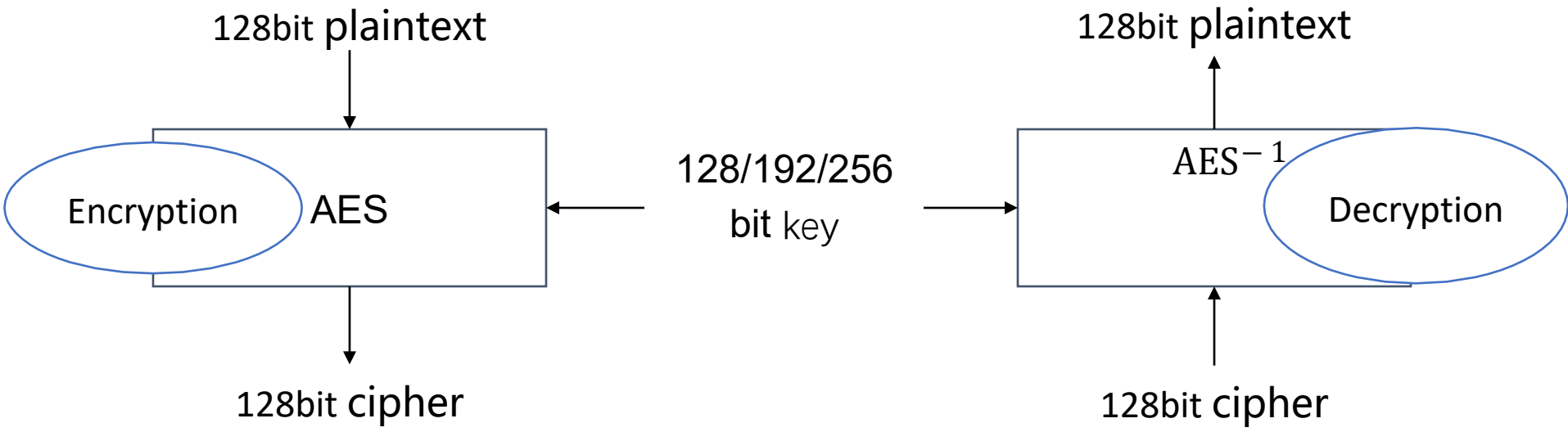
Rijndael fully supports block sizes and key sizes of 128 bits, 192 bits and 256 bits, in any combination. In principle, the Rijndael structure can accommodate any block sizes and key sizes that are multiples of 32, as well as changes in the number of rounds that are specified.

Potential for Instruction-Level Parallelism

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

3.1.3 AES

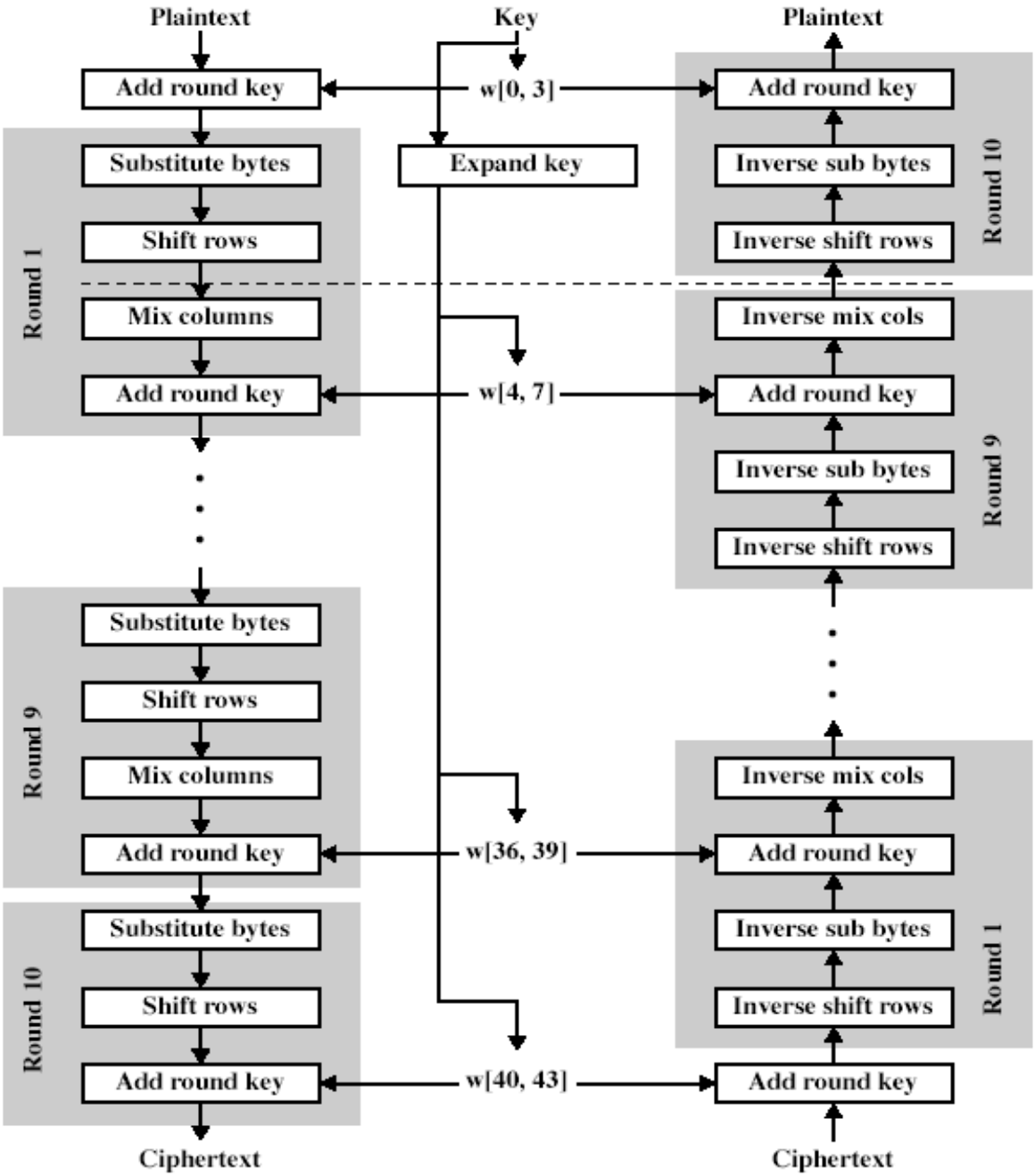
■ AES framework



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

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 **obfuscations** and three substitutions
  - **Byte substitution（字节代替）**: byte-by-byte substitution in groups is done with an S-box.
  - **Row shifting（行移位）**: a simple substitution
  - **Column obfuscation（列混淆）:** substitution using arithmetic properties on Galois domains
  - **Round key addition（轮密钥加）**: 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
- Each stage is reversible and decryption uses the extended key in reverse order, but the algorithms are different



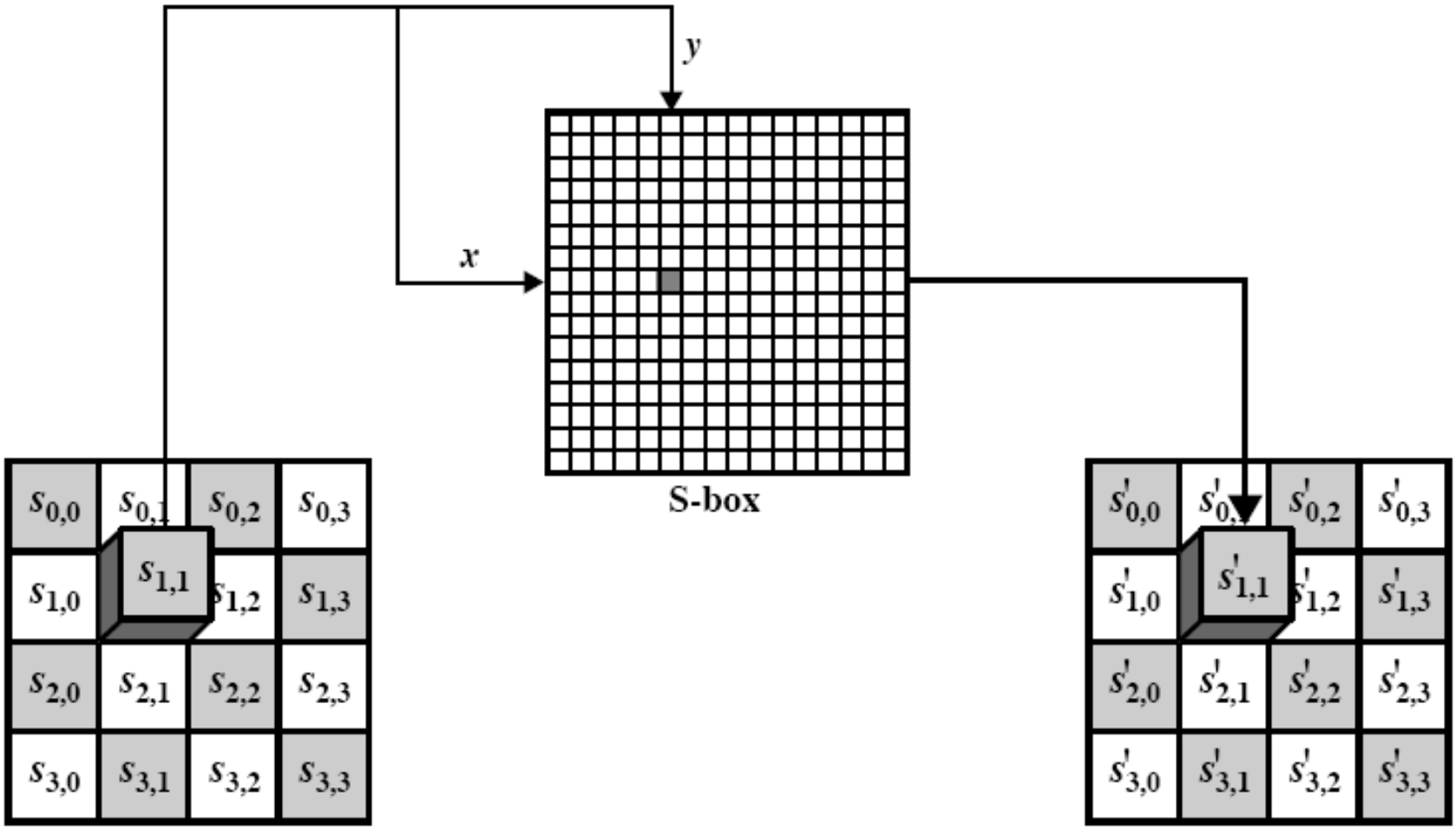
(a) Encryption

(b) Decryption

The four phases of AES

- Byte substitution (substitute byte transformation)
- Row shifting (shift row transformation)
- Column obfuscation (mix column transformation)
- Round key addition (add round key transformation)

Byte substitution



(a) Substitute byte transformation

S-box for AES

Table 5.4 AES S-Boxes

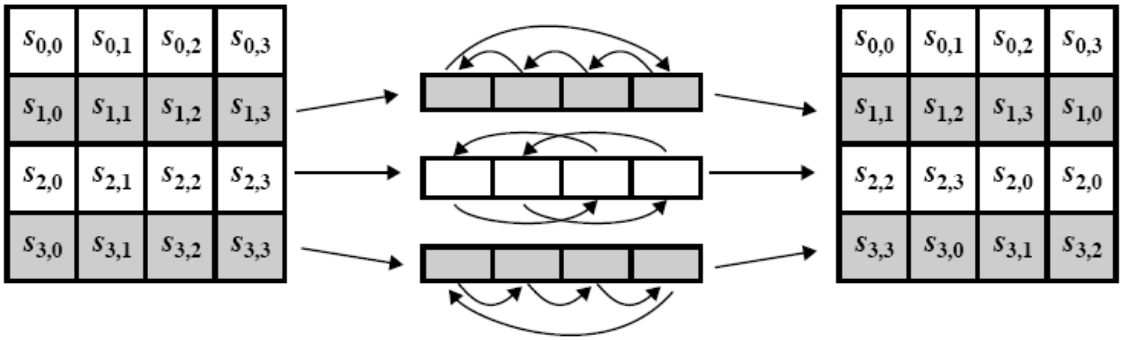
(a) S-box

		y															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
x	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
	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

Row shifting

- Shifting rules:
  - First row 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

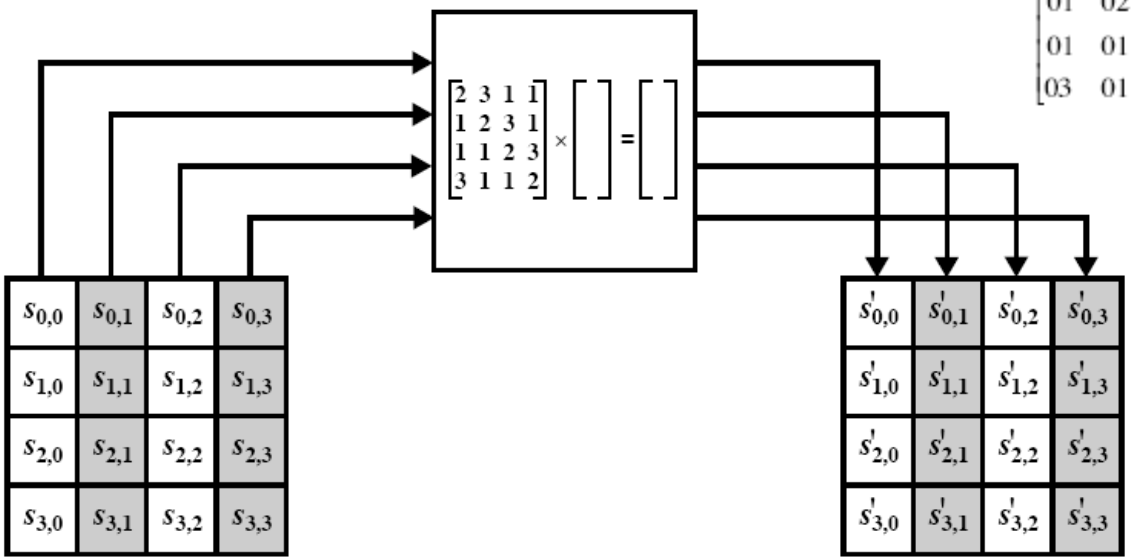


(a) Shift row transformation

Column obfuscation

- Each column is treated separately
- Each byte in each column is replaced with a result associated with all the 4 bytes in this column

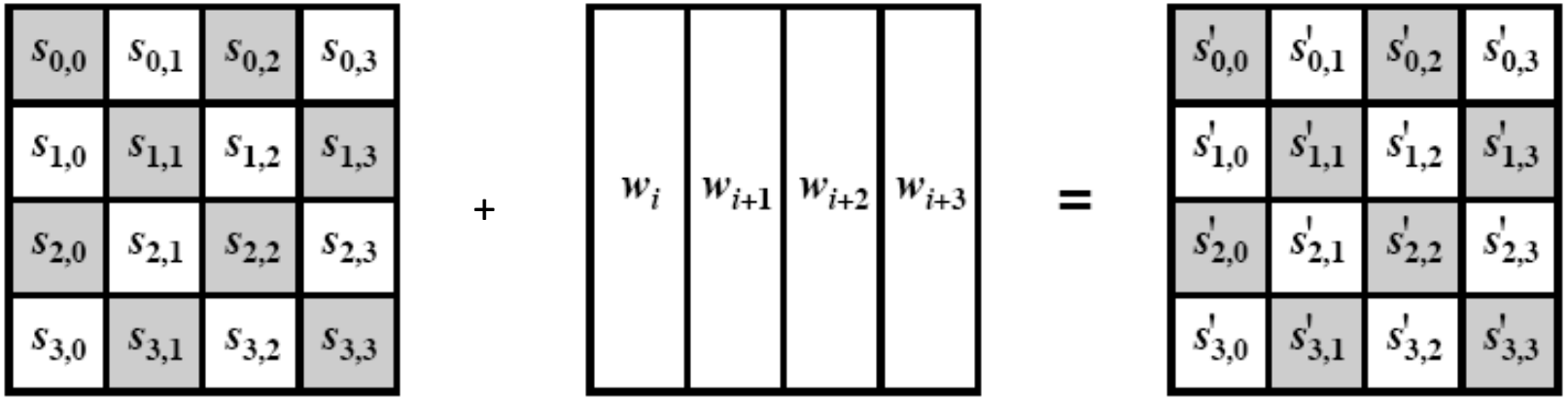
$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{bmatrix}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



(b) Mix column transformation

Round key addition transformation

- XOR the 16-byte matrix with a subkey of length 16 bytes.



(b) Add Round Key Transformation

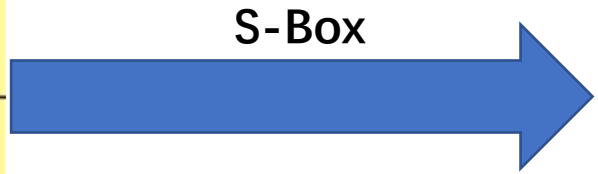
AES encryption example

- **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
- Compute the ciphertext

AES encryption example

- **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**

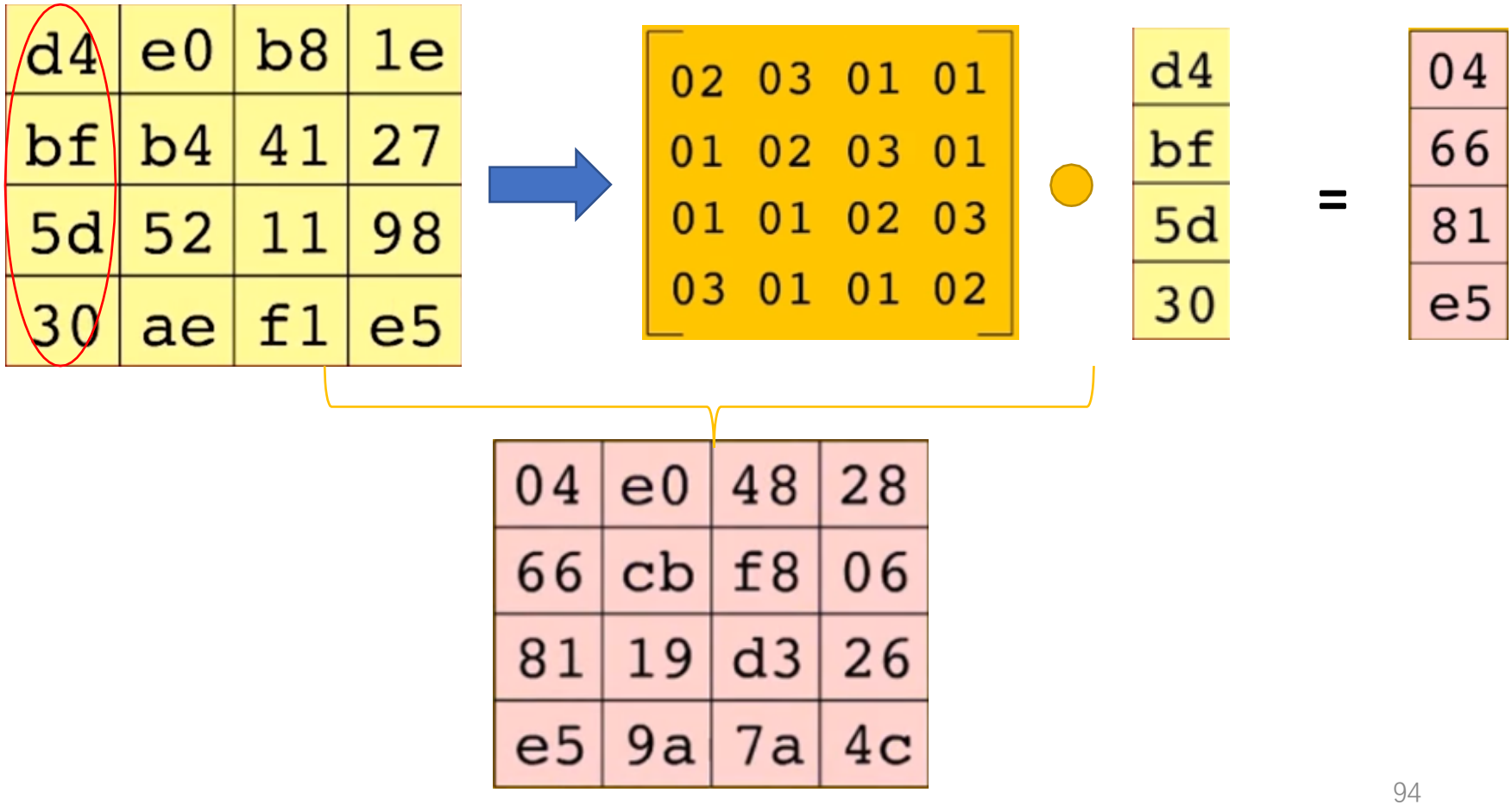
d4	e0	b8	1e
27	bf	b4	41
11	98	5d	52
ae	f1	e5	30



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

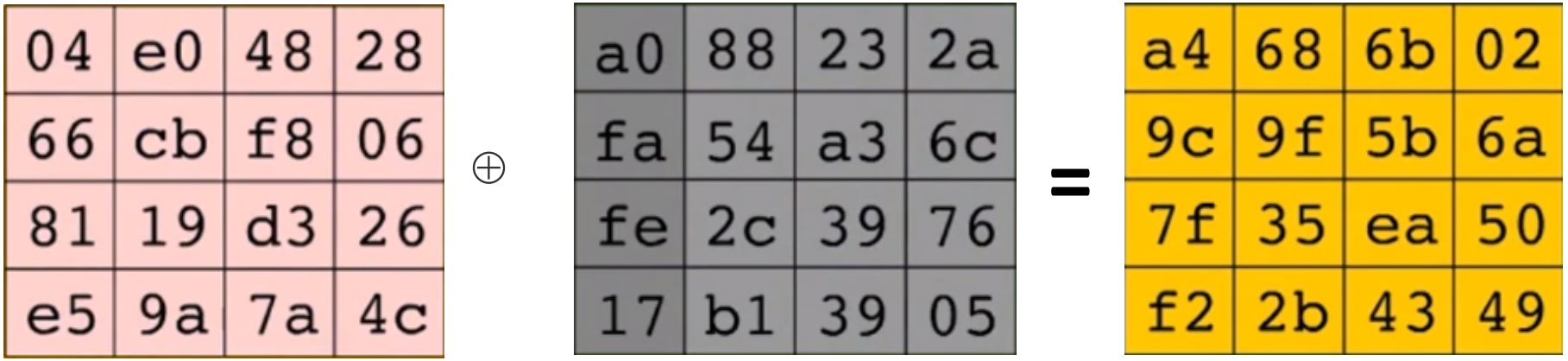
AES encryption example

■ 3. Column obfuscation



AES encryption example

- **4. Round key addition**



AES encryption example

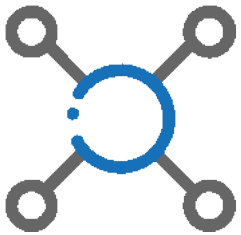
- **5. Repeat the above round 10 times**
  - **Ciphertext is:**

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

Public key cryptography (asymmetric Cryptograph)

What is the problem with symmetric key?

- Symmetric cryptosystems (e.g. DES, AES) need two users to establish a "secure channel"in advance, which is not easy to achieve.
- Consider a group of N users who need to communicate securely with each other, for a symmetric cryptosystem, each user needs to share the private key with the remaining N - 1 users, and total number of keys is **N(N - 1)/2 keys**.
- Also, there is no guarantee of accountability, why?



Difficulty of key distribution



No support for "open systems"



High key management costs

History of public key cryptography

- 1976, **Diffie** and **Hellman** introduce the concept of **"public-key cryptography"**, **which won** the **ACM** Turing Award in **2015**.

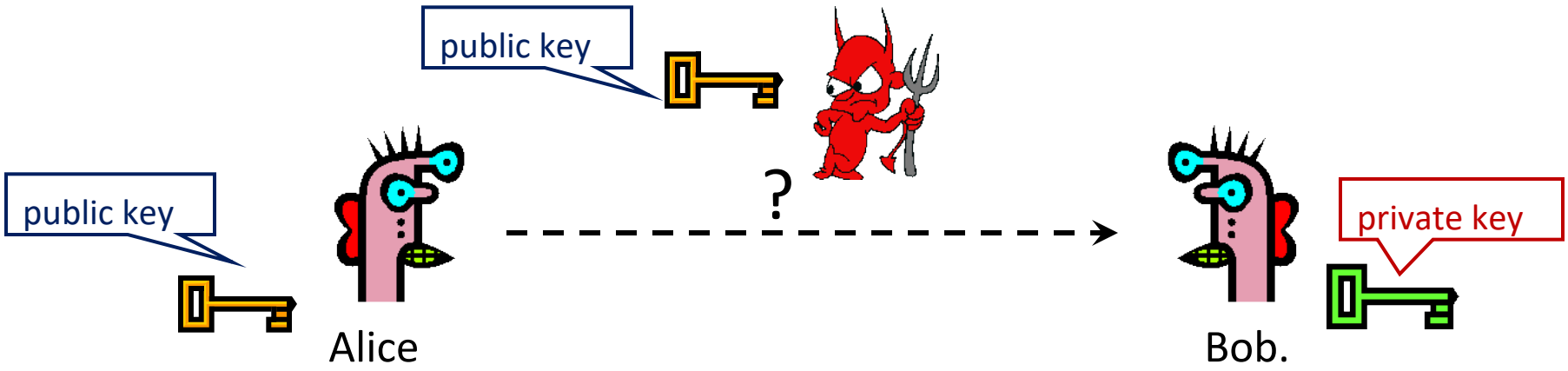
- **1977, Ronald Rivest, Adi Shamir and Leonard Adleman** propose the **RSA** algorithm.



The public key cryptography

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

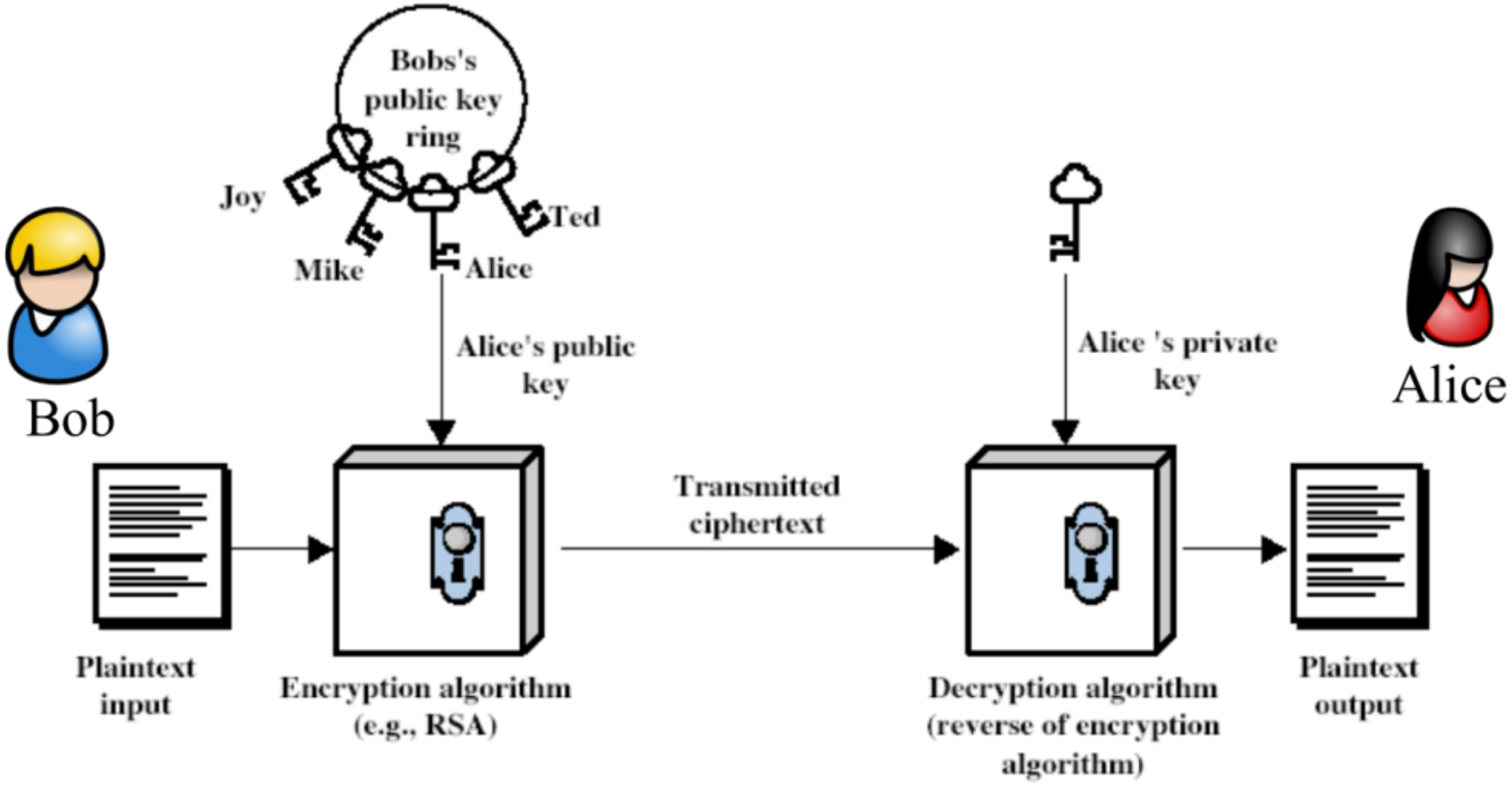
E.g., it is easy to compute the product of any given integer, but very difficult to compute the factor of a given large integer.



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

Private key sk: to be used by the user

Public key cryptography



Introduction to public key cryptography

- **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



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

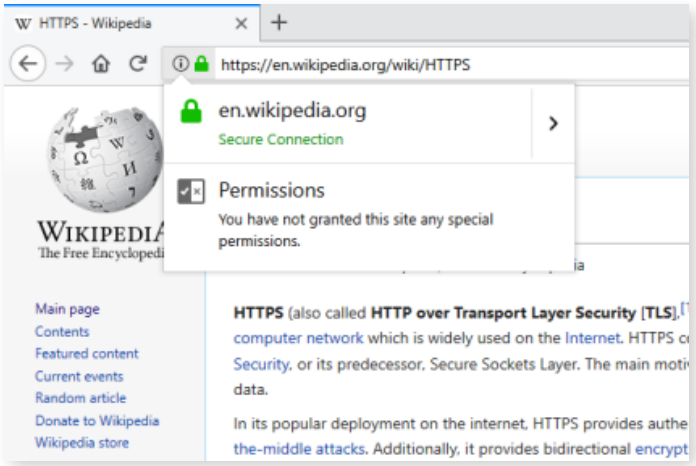


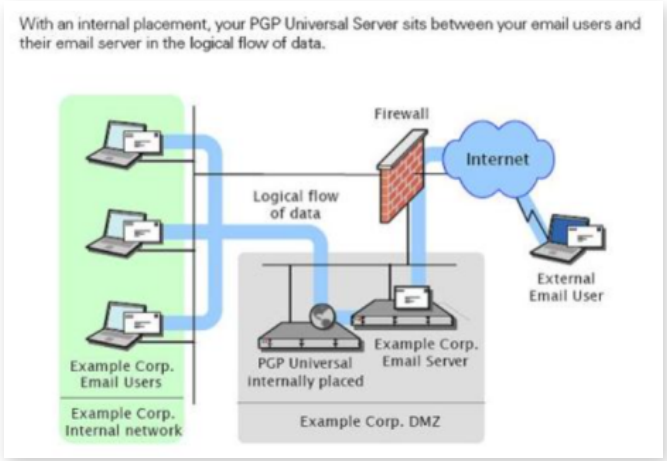
open system

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

Public key cryptography 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?**





RSA encryption algorithm

History of the RSA

- In 1978, Ron Rivest, Adi Shamir and Leonard Adleman of MIT proposed the RSA encryption algorithm.
- In 2002, they 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 = φ(n) = (p-1)(q-1)
3. Randomly select e (where e < n), e and z are "mutually prime"
4. Pick d such that ed mod z = 1
5. The public key is (n, e) and the private key is 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: $de = 1 \bmod \varphi(n)$ and $d < \varphi(n)$.

   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:

Suppose m=30120

Ciphertext $c=m^e \bmod n = (30120)^{17}$

Reconstruct plaintext: $m=c^d \bmod n = 30120$

RSA Security Analysis

Exhaustive attack

1. The attacker chooses an M, $C = M^e \pmod n$
2. The choices of d is at most n-1, so attackers try each choice of d to crack, if $M'=C^{d'} \pmod n = M$, d' is the key
3. Suppose p, q is 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 $10^{200} / (3*10^{15}) = \mathbf{3*10^{185}}$ **years!**

Computationally infeasible.

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：

```
123018668453011775513049495838496272077285356959533479219732245215172640050726
365751874520219978646938995647494277406384592519255732630345373154826850791702
612214291346167042921431160222124047927473779408066535141959745985690214341 3

= 33478071698956898786044169848212690817704794983713768568 91
  24313889828837938780022876147116525317430877378144679994 89×
  36746043666799590428244633799627952632279158164343087642 67
  60322838157396665112792333734171433968102700927987363089 17
```

How to achieve 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 **establish** a security key

- **Question**: How **to establish** 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**

Key Agreement - DH Key Exchange Protocol

- DH implements how two parties can **establish** a common key when communicating under an insecure communication channel.

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-

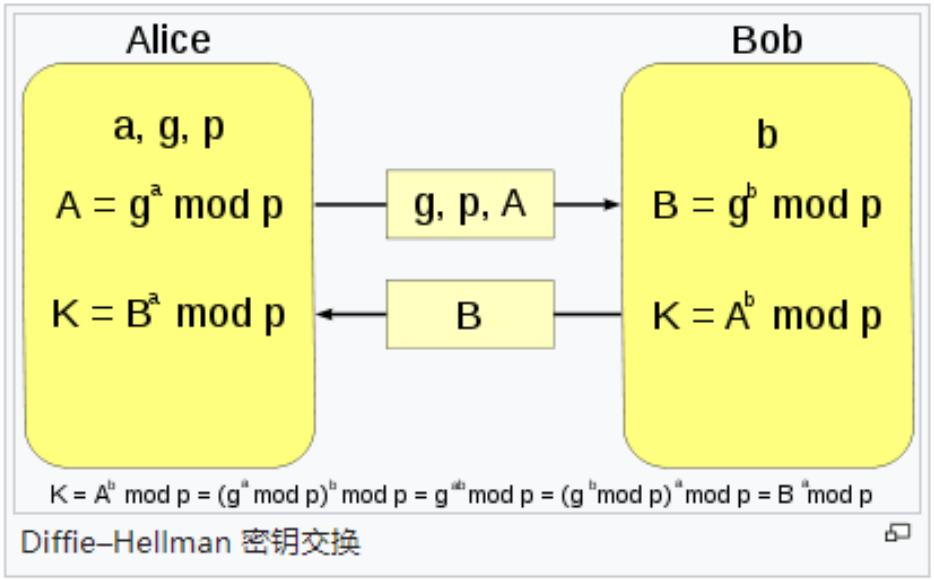
DH protocol fundamentals

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

$$A \to B : g^a \bmod p \qquad B \to A : g^b \bmod p$$

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

g^{ab} is the shared key



Diffie–Hellman 密钥交换

Q: What is confidential and what is public?

DH Key Exchange Protocol Example

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

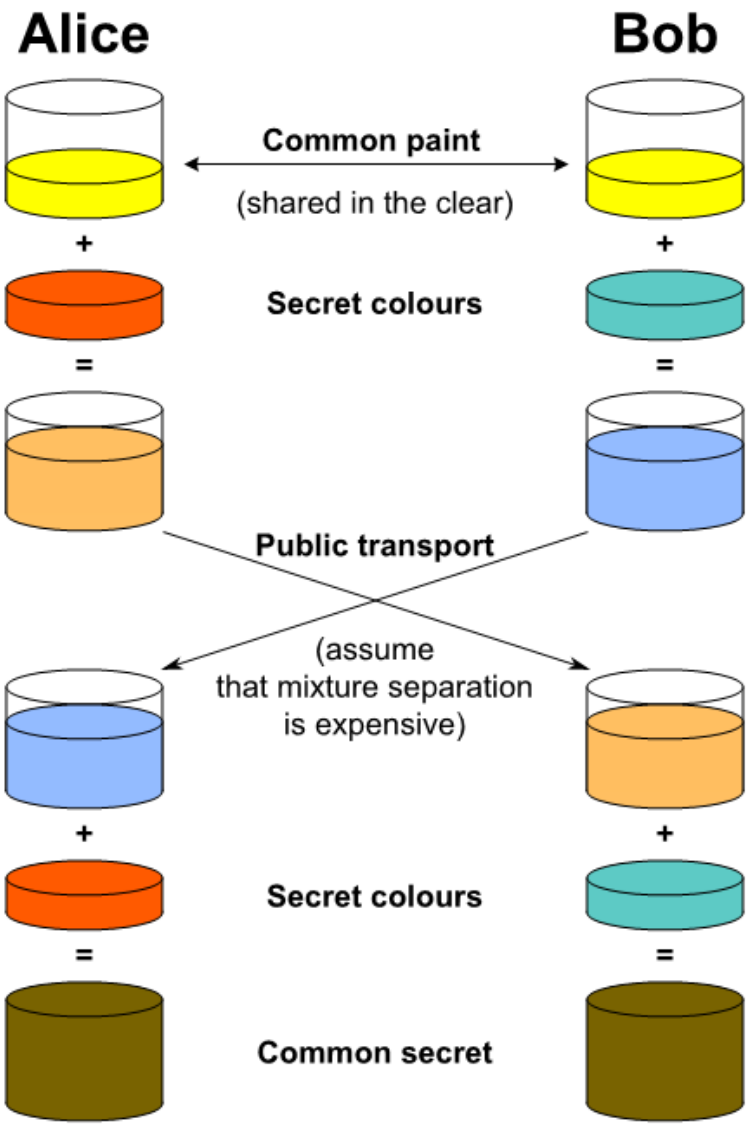
爱丽丝		
秘密	非秘密	计算
	p, g	
a		
		$\mathbf{g}^{\mathbf{a}}$ mod p
	...	
	$(\mathbf{g}^{\mathbf{b}}$ mod p$)^{\mathbf{a}}$ mod p	

→

←

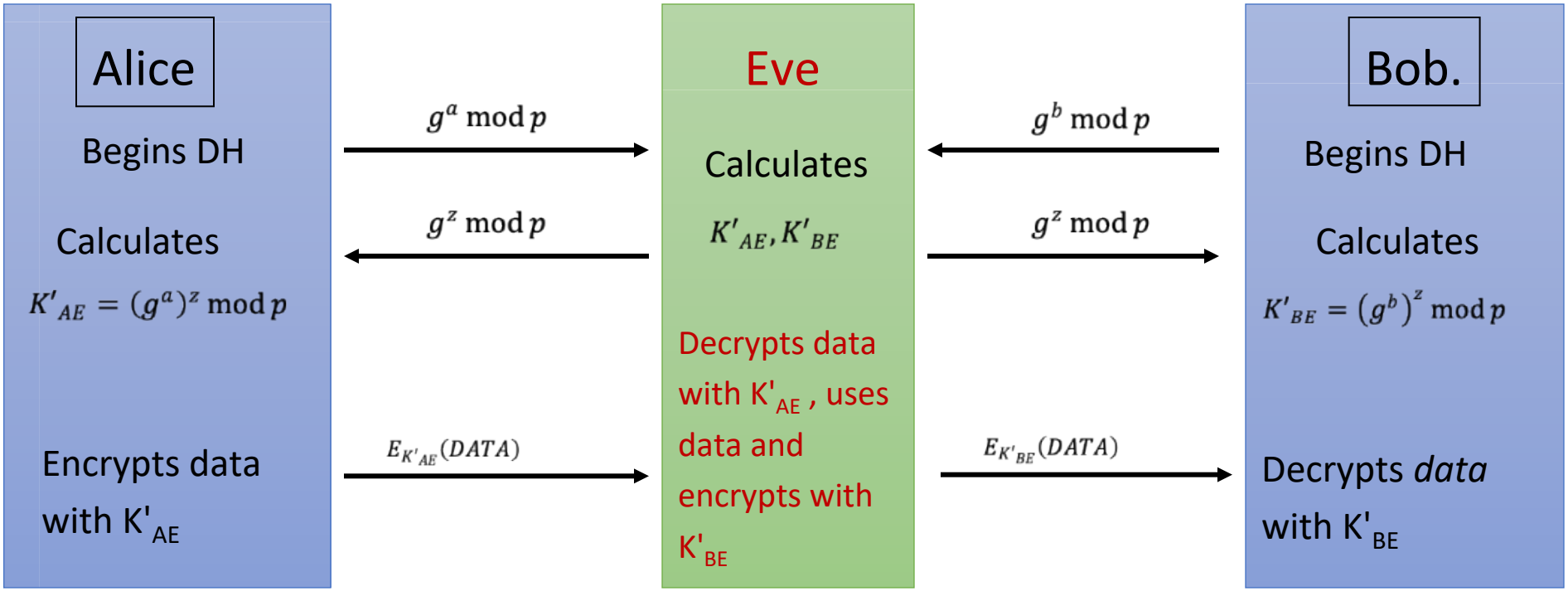
=

鲍伯		
计算	非秘密	秘密
	p, g	
		b
	...	
$\mathbf{g}^{\mathbf{b}}$ mod p		
	$(\mathbf{g}^{\mathbf{a}}$ mod p$)^{\mathbf{b}}$ mod p	


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

Attacks against DH

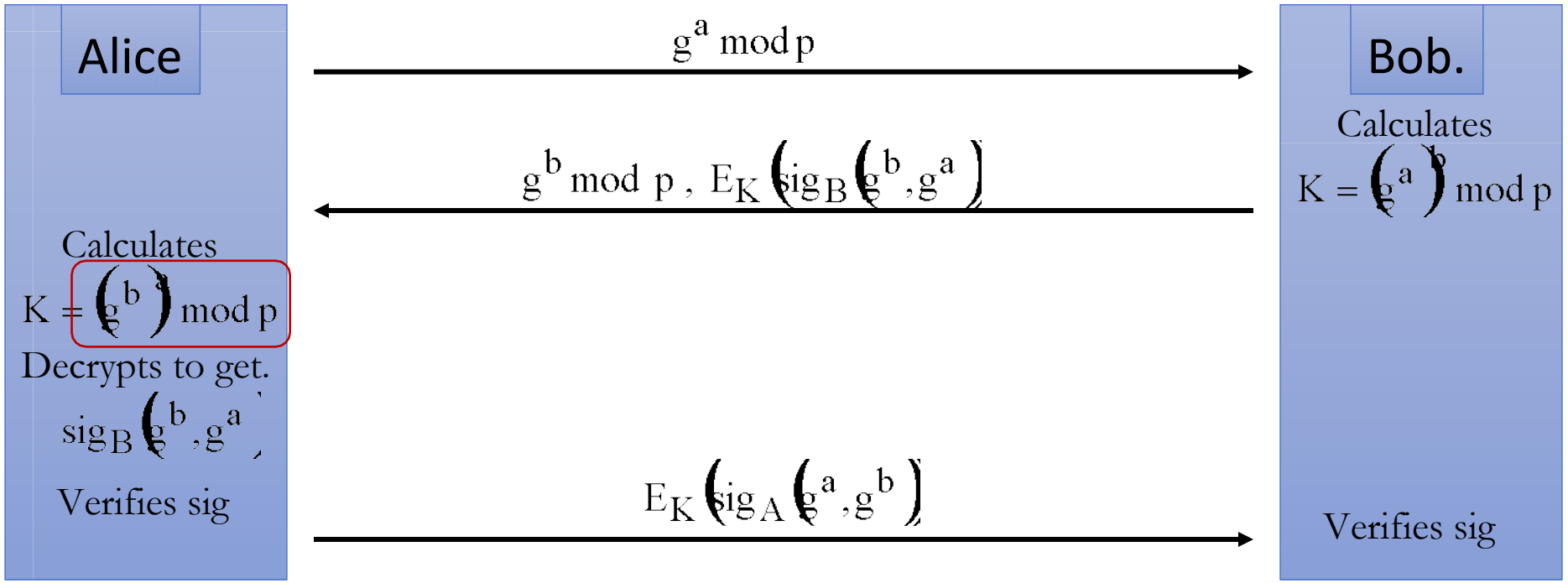
- **DoS attack**: an adversary initiates a large number of key requests, which will cause misuse and denial of computational resources
- **Man-in-the-middle attacks**: an adversary can impersonate one of users A and B, respectively, and exchange keys with the other (which can listen in on and pass on secret messages from A and B without being detected).



Defence against the man-in-the-middle attacks

- Digital signatures to **prevent the** man-in-the-middle attacks
- **Private sig( )** function: digital signature
- **Public ver( )** function: authentication

Q: Why is it effective?



Diffie joins ZJU



Cryptography in IoT

Cryptographic Applications for IoT

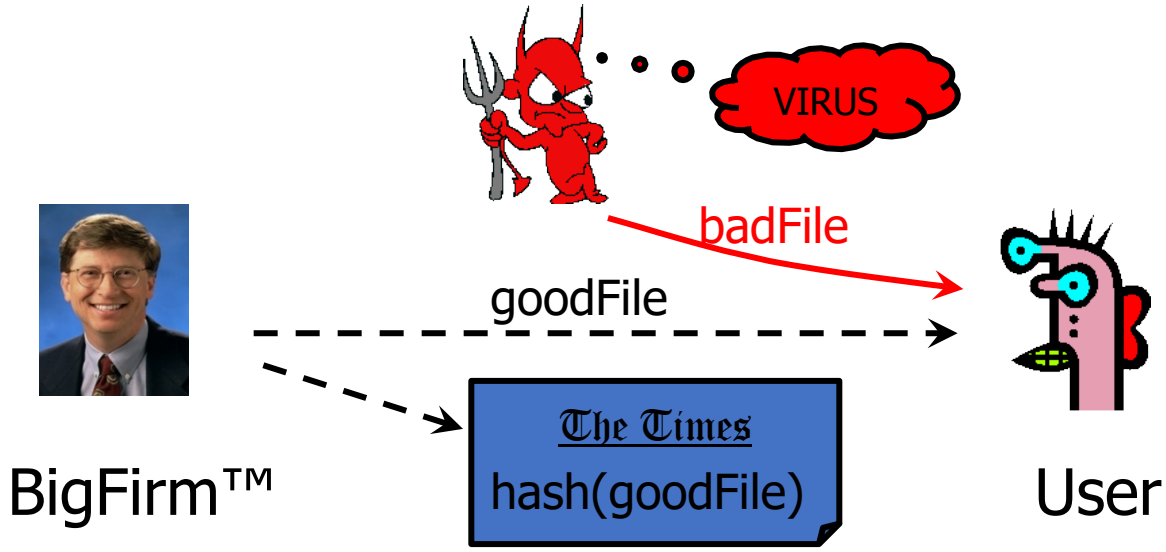
- **Limitations of IoT devices:**
  - Constrained computation, memory
  - Constrained power supply
  - Embedded firmware and hardware
  - Interaction with the physical world
- **Lightweight encryption algorithm**
  - lightweight encryption algorithms optimizing the key length, number of encryption rounds

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 - How to ensure integrity?

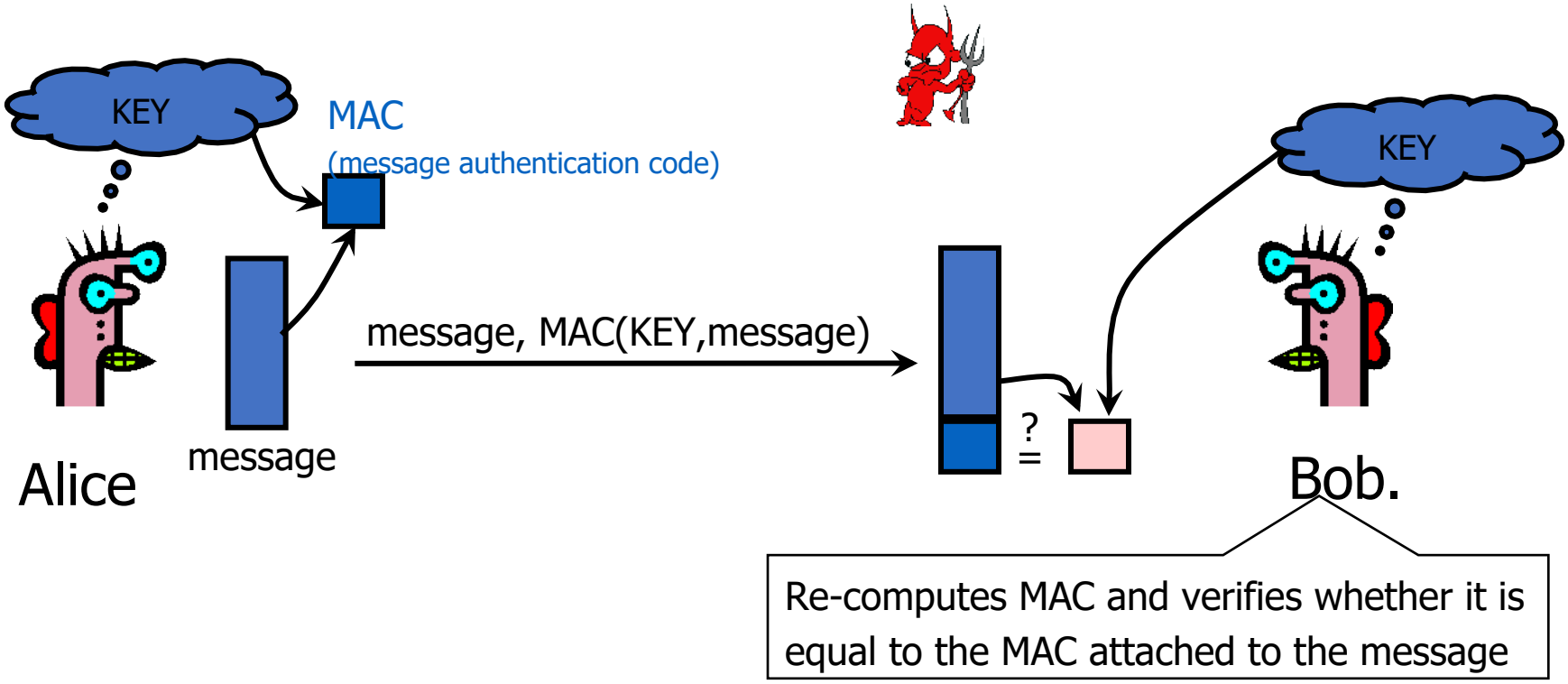
- **Message authentication code (MAC, message authentication code)**



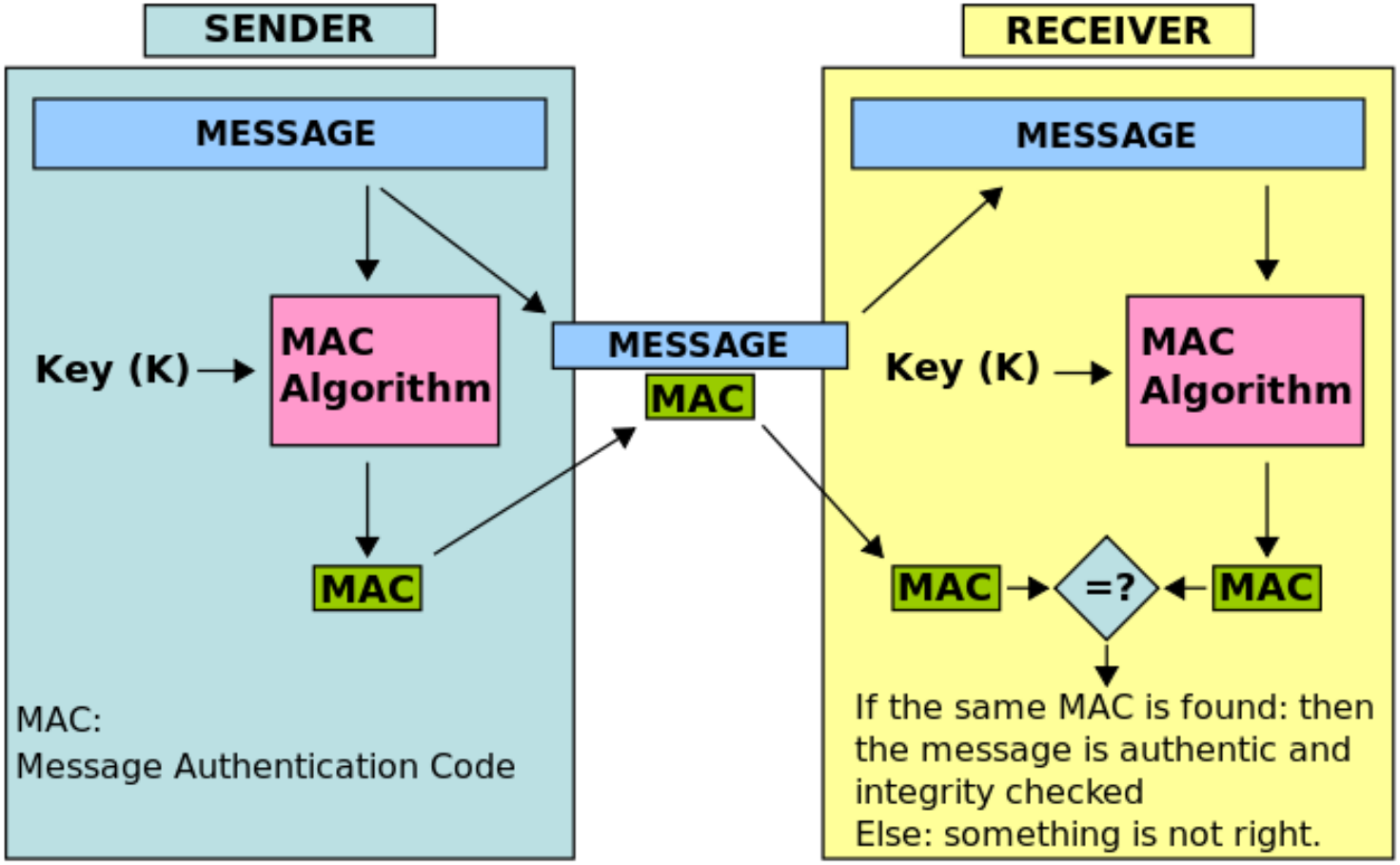
$$MAC = H(message + key)$$

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

Message authentication code



Message authentication code

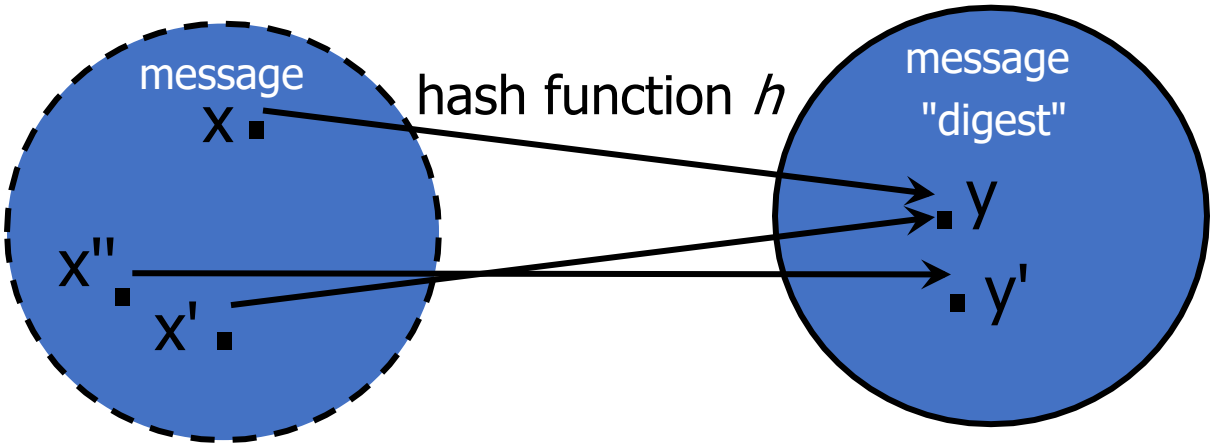


Integrity Authentication: only someone who knows the key KEY can compute the MAC for a given message.

Integrity - Hash/Hash Functions

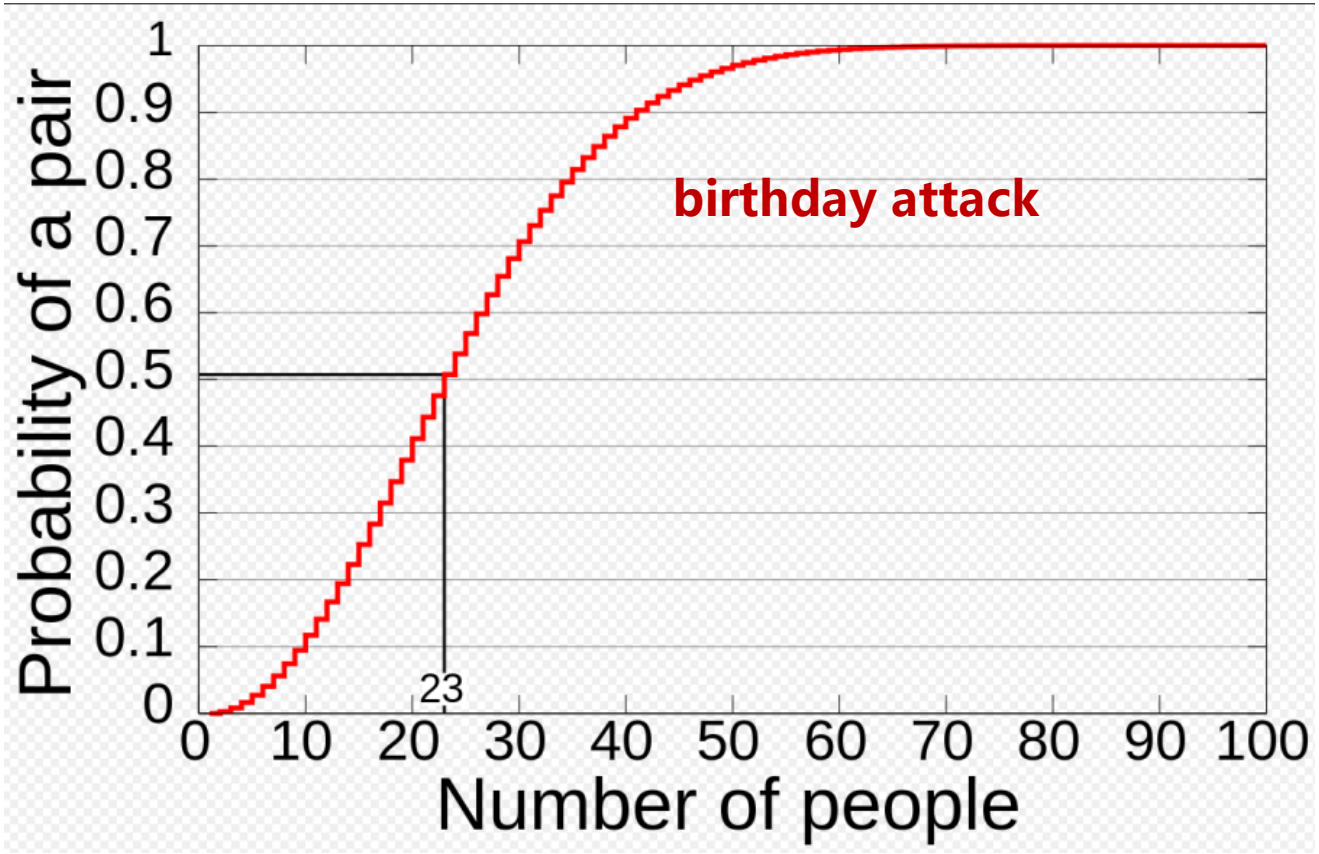
- **The hash/hash function y=h(x) :**
  - Support any length of input x
  - For any length of input x, the output z is of fixed length
  - Low computational complexity
  - For a given output y, finding the input x is impossible
  - For a given x_1, finding x_2 that satisfies h(x_1)= h(x_2) is not possible

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



Birthday attack

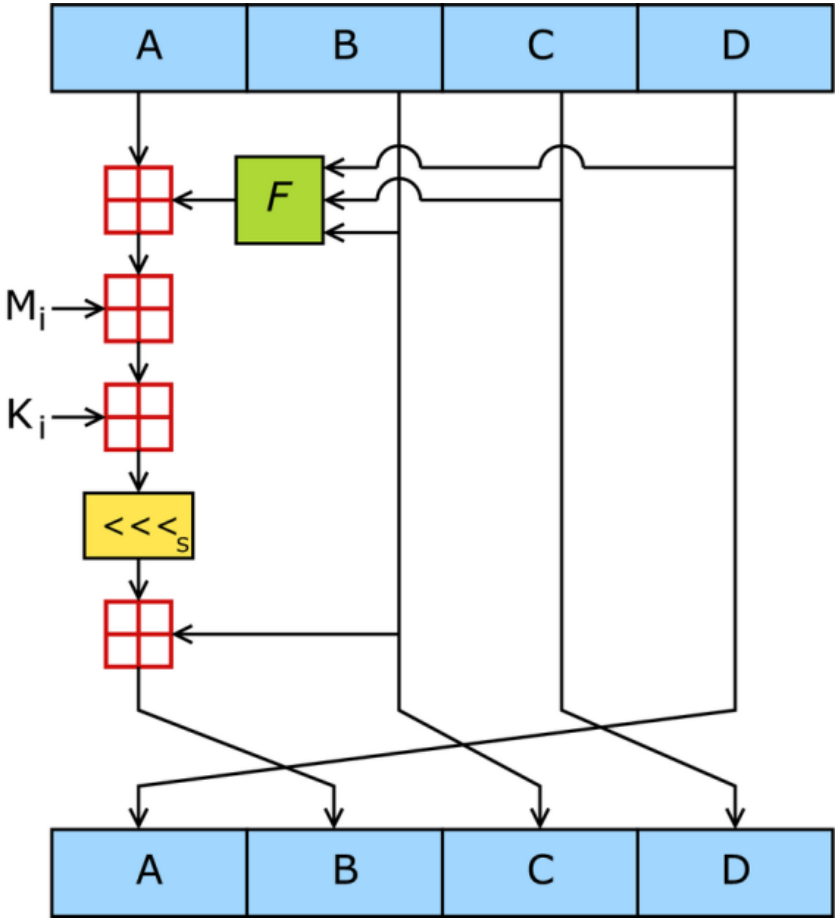
- **Collisions in Hashing**
  - Attack happens when $h(x_1)=h(x_2)$



Integrity – hash function MD5

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

Integrity – hash function MD5

■ Collision 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, MD5 collision in our smartphones happens 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.

Integrity – hash function SHA

- **SHA (Security Hash Algorithm)**
  - Maximum message length < 2^{64}, message group length 512bits
  - Output length **160bits**
  - **The first collision was discovered in the summer of 2005.**

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)	-

Integrity – hash function MD5 vs. SHA

- Comparison of **MD5** and **SHA**

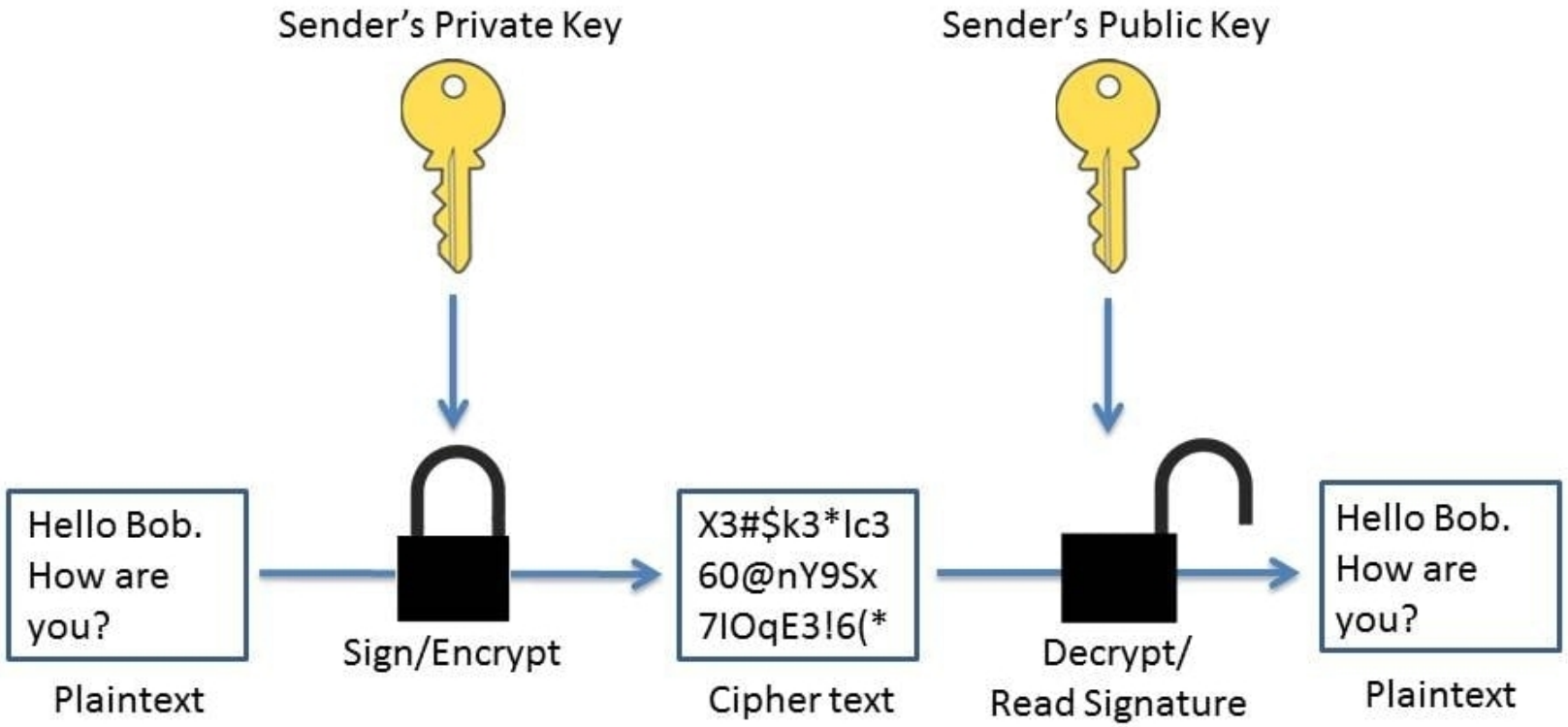
	SHA-1	MD5
Maximum Message Length	$<2^{64}$	not limited
Message Group Length	512	512
Framework	Merkle Iteration	Merkle Iteration
Link variable length	160	128
Hash length	160	128

IoT Authenticity/non repudiation - Digital Signature

- **Digital signatures:** preventing repudiation by both parties to the communication
  - System initialisation
  - 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 with public cryptography

- The process of digital signatures using public/private keys



Digital signatures with RSA algorithm

- **RSA public key algorithm digital signature:**
  - Parameter selection and key generation
  - **Signaturging process**: User A signs the message M and computes the

$$S=Sig(M)=M^{d} mod\ n$$

$$M+S$$

  - **Verification process**: user B verifies user A's digital signature on message M, calculates

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

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

IoT Key Management

- **Key management**: involves the processes of key generation, distribution, verification, updating, storage, backup, and destruction.

- **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

- **Key distribution:**
  - **Solution 1**: A selects the key and manually passes it to B
  - **Solution 2:** Third party C chooses the key to be passed manually to A,B respectively
  - **Solution 3:** Transmit the the new key with the original private key between A and B.
  - **Solution 4:** The third party C who has a shared key with A and B respectively, transmits the new key to A and B.

- For symmetric encryption, N user sets require $\frac{\mathrm{N}(N-1)}{2}$ shared keys

Discussion

- How LLM can enhance and help break cryptography?

Reference video

