常用密码算法

XU, Hui

xuh@fudan.edu.cn

October 26, 2020

何时需要加密？

机密信息传输

身份认证

消息认证

密钥交换

摘要运算

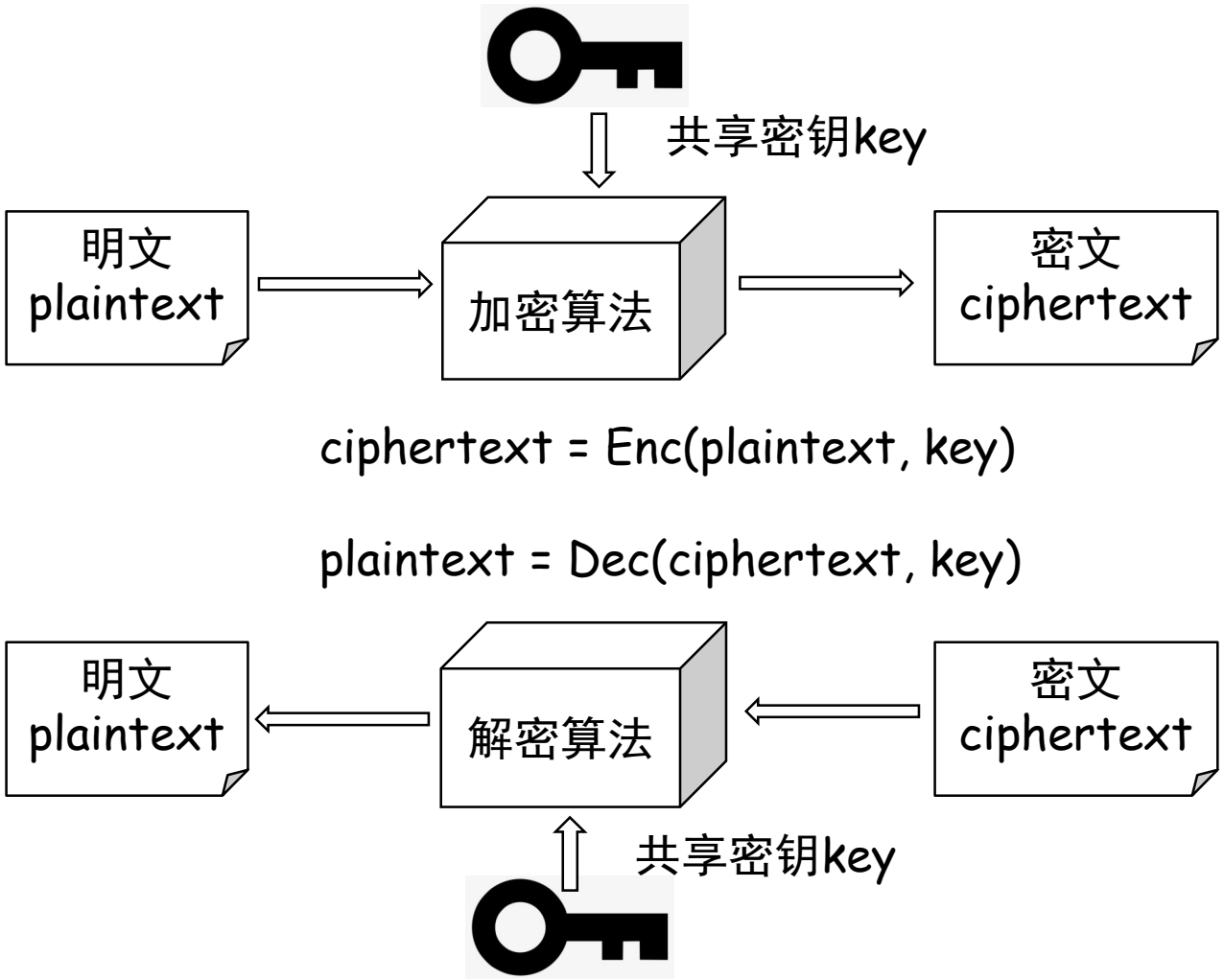
金融交易

主要内容

- 第一部分：对称加密
- 第二部分：哈希算法
- 第三部分：非对称加密

第一部分：对称加密

对称加密原理


共享密钥key
明文
plaintext
加密算法
密文
ciphertext
ciphertext = Enc(plaintext, key)
plaintext = Dec(ciphertext, key)
明文
plaintext
解密算法
密文
ciphertext
共享密钥key

练习

假设有对称密码算法， $C = (P \oplus K_0) \boxplus K_1$

其中P是8-bit明文，K=[K_0,K_1]是密钥16-bit的密钥。

假设P = 'A'，K="0000111111110000"，计算C

$C = (01000001 \oplus 00001111) \boxplus 11110000$

$C = 01001110 \boxplus 11110000$

$C = 00111110$

写出解密方程和过程： $P = (C \boxplus -K_1) \oplus K_0$

$P = (00111110 \boxplus 00010000) \oplus 00001111$

$P = 01001110 \oplus 00001111$

$P = 01000001$

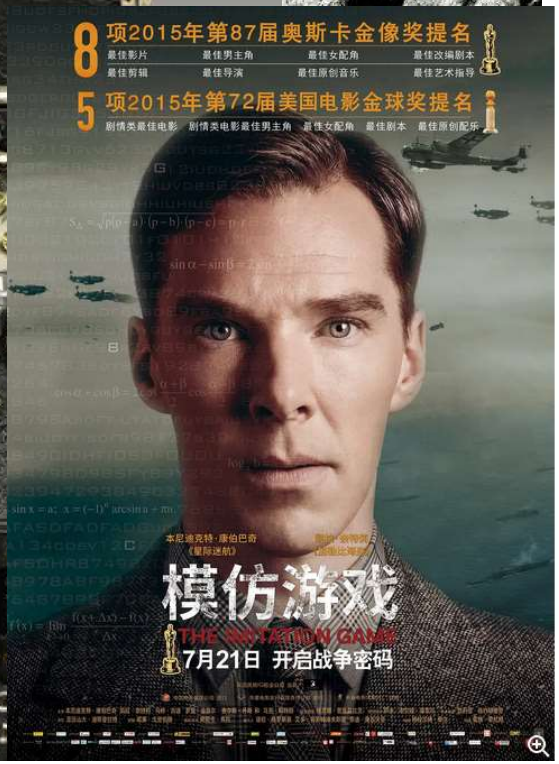
Kerckhoff原则

A cryptosystem should be secure even if everything about the system, except the key, is public knowledge.

一个密码系统在除了密钥以外的所有信息都公开的情况下也应当是安全的。

恩尼格码

通过破解密码影响了第二次世界大战的进程。



练习

$$C = (P \oplus K_0) \boxplus K_1$$

已知 P_1 = 'A', C_1 =00111110,

P_2 = 'B', C_2 =00111101

是否可以计算出K?

$(01000001 \oplus K_0) \boxplus K_1 = (01000010 \oplus K_0) \boxplus K_1 \boxplus 1$

$(01000001 \oplus K_0) = (01000010 \oplus K_0) \boxplus 1$

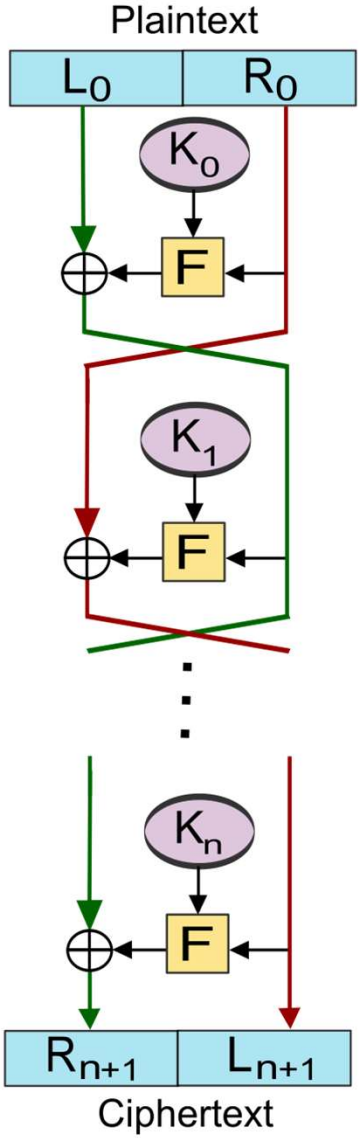
$K_0 = ****** 11$

主要内容

- 对称分组加密算法
- 分组密码工作模式
- 流密码

1. 对称分组加密算法

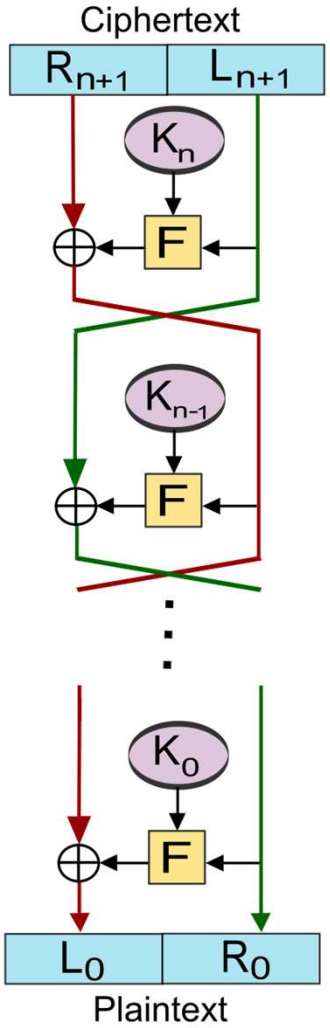
Feistel密码结构（Feistel network）



加密算法：

- 输入：固定长度（block size ）的明文
- 将明文分割为等长的两部分$\{L_0, R_0\}$
- 每一轮（共n+1轮）计算：
  - $L_{i+1}=R_i$, $R_{i+1}=L_i \oplus F(R_i, K_i)$
- 输出：$\{R_{n+1}, L_{n+1}\}$

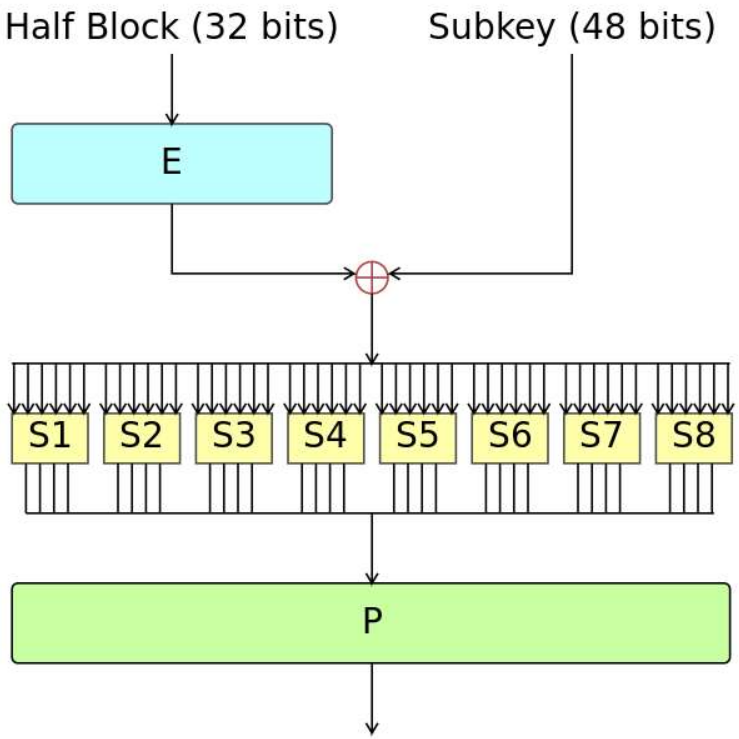
Feistel密码结构（Feistel network）



解密过程:

- 输入：密文(R_{n+1}, L_{n+1})
- 将密文分割为等长的两部分(R_{n+1}, L_{n+1})
- 每一轮（共n+1轮）计算：
  - $R_{i-1}=L_i$, $L_{i-1}=R_i \oplus F(L_i, K_i)$
- 输出：(L_0, R_0)

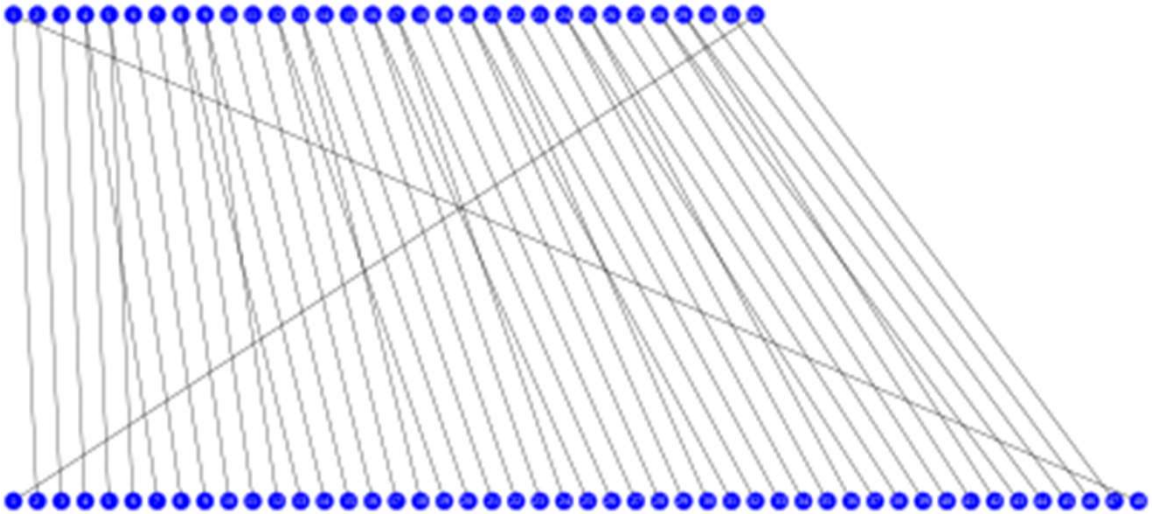
DES算法的Feistel Function



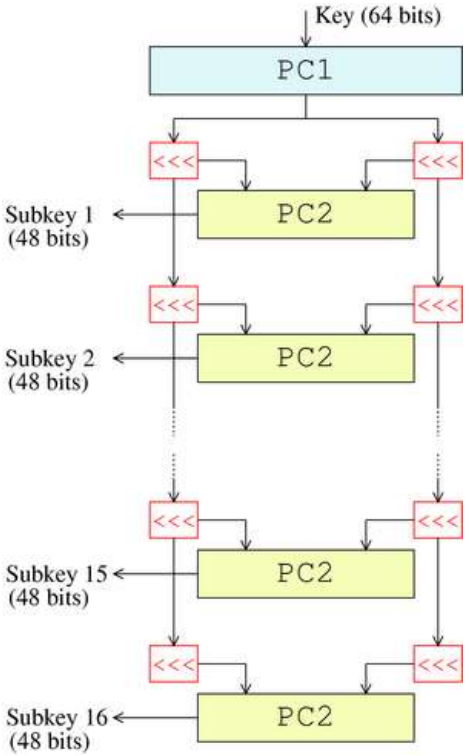
- 扩充（Expansion）：32 bits => 48bits
  - ✓ 4 bits * 8 => 6 bits * 8
  - ✓ 通过复制邻接片段的方式
- 密钥混合（Key Mixing）：异或运算
- 替换（Substitution）：48 bits => 32bits
  - ✓ 6 bits * 8 => 4 bits * 8
  - ✓ 通过lookup table (S-box)查询
- 重排列（Permutation）：将每一片的4 bits 分散到4个不同的片段中（P-box）

扩充

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



DES的密钥派生方法



- Permuted Choice 1 (PC-1)
  - 64 bits => 56 bits;
  - 分割为两部分，每部分28 bits;
- 每一轮（共n+1轮）计算：
  - 左移1 bit或2 bits;
  - Permuted Choice 2 (PC-2): 48 bits

57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	52	44	36

PC-1 (左侧)

63	55	47	39	31	23	15
7	62	54	46	38	30	22
14	6	61	53	45	37	29
21	13	5	28	20	12	4

PC-1 (右侧)

14	17	11	24	1	5
3	28	15	6	21	10
23	19	12	4	26	8
16	7	27	20	13	2
41	52	31	37	47	55
30	40	51	45	33	48
44	49	39	56	34	53
46	42	50	36	29	32

PC-2

S-Box 和 P-Box

S_1	-0000-	-0001-	-0010-	-0011-	-0100-	-0101-	-0110-	-0111-	-1000-	-1001-	-1010-	-1011-	-1100-	-1101-	-1110-	-1111-
0----0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
0----1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
1----0	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
1----1	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13
S_2																

P-Box

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

Confusion和Diffusion

- 香农提出的安全密码算法的两个重要属性
- 攻击者的目标是破解密钥，假设其可以自由使用
  - 加密服务：chosen paintext
  - 解密服务：chosen ciphertext
- Diffusion：隐藏明文和密文之间的关系
  - 更改1比特明文，至少一半的密文信息会被改变
  - Permutaion
- Confusion：隐藏密文和密钥之间的关系
  - 密文的每一比特应该与密钥的多个部分相关，从而混淆密文和密钥之间的关系
  - Substituion：与密钥有关

对称分组加密算法

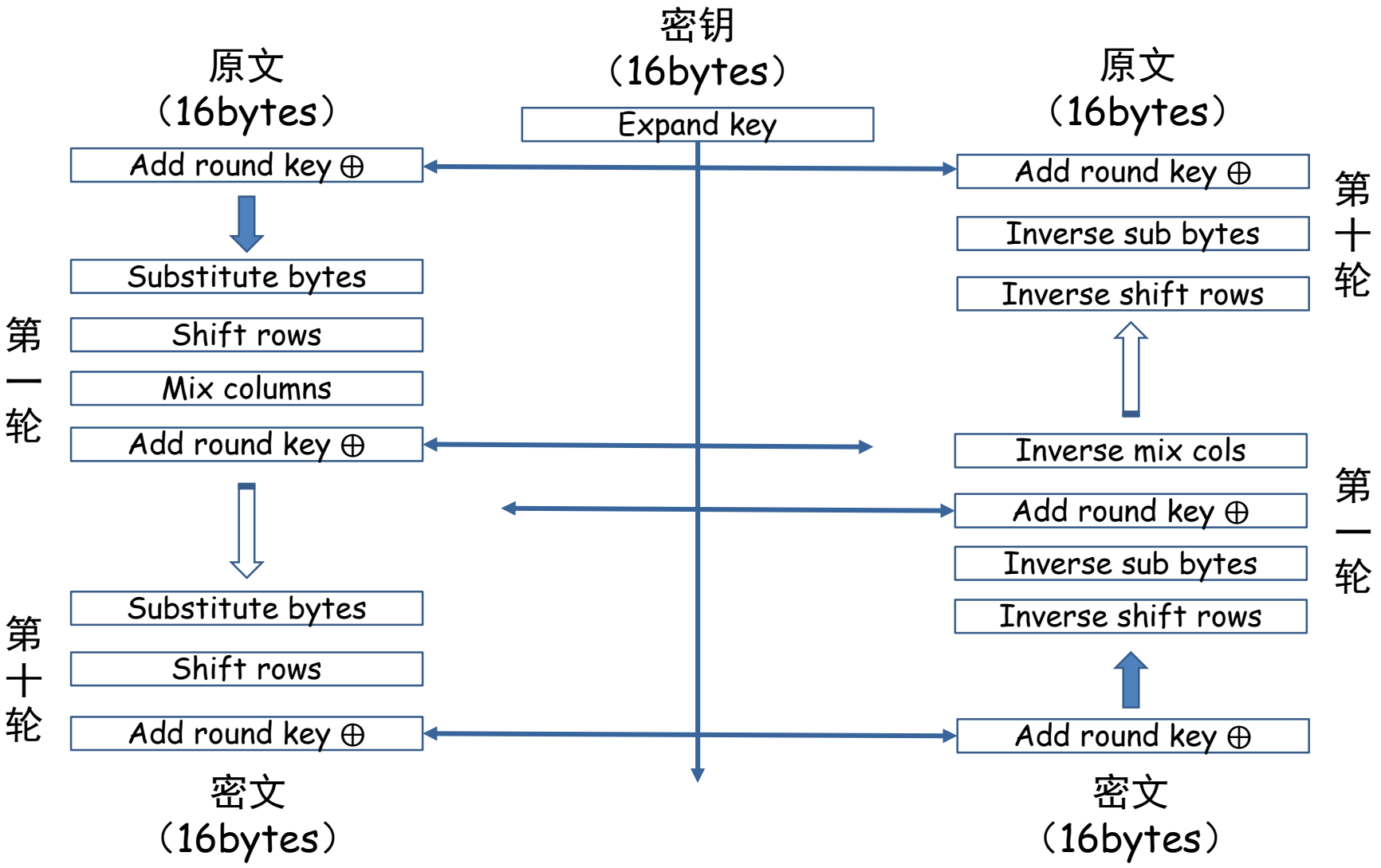
- DES
  - 1977年发布，FIPS PUB 46，密钥空间2^{56}
  - 1998年，DES在3天以内被250，000美元制造的设备破解
- 3DES
  - NIST SP 800-67 (1998), FIPS PUB 46-3 (1999)
  - 1999年，NIST将3-DES指定为过渡的加密标准
  - 3DES算法可以在美国敏感的政府信息系统中继续使用到2030年
- AES
  - 2001年发布， FIPS PUB 197

3DES

- 加密：Ciphertext = Enc(K_3, Dec(K_2, Enc(K_1, plaintext)))
- 解密：Plaintext = Dec(K_1, Enc(K_2, Dec(K_3, plaintext)))
- 主要优势：密钥空间增大
  - ✓ $K_1 \neq K_3$：2^{168}
  - ✓ $K_1 = K_3$：2^{112}
- 主要缺点：速度慢，约3倍DES耗时

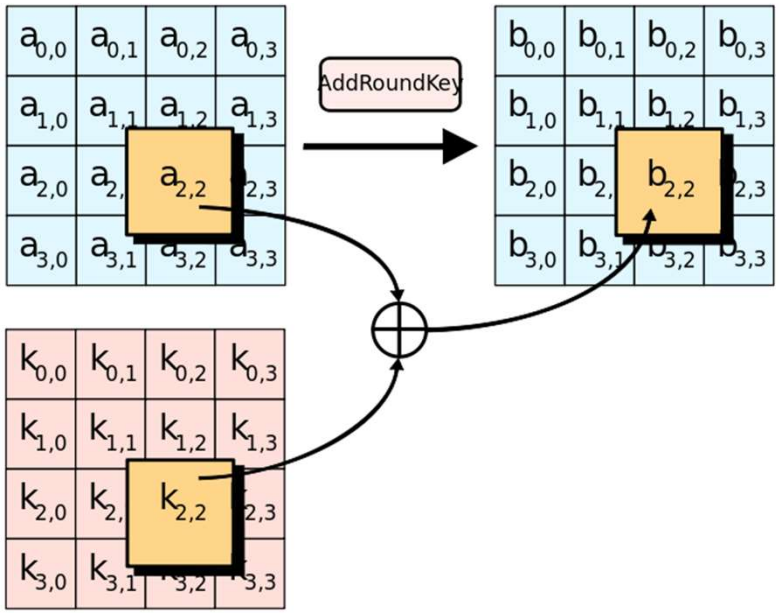
AES算法（Rijndael）

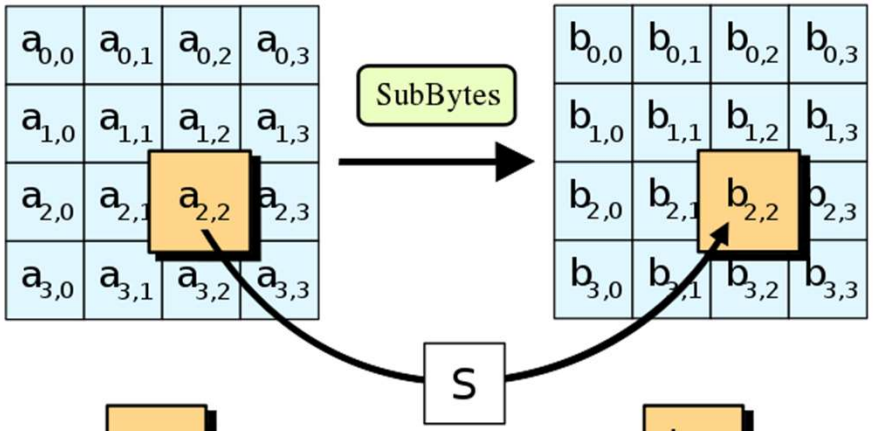
非Feistel密码结构

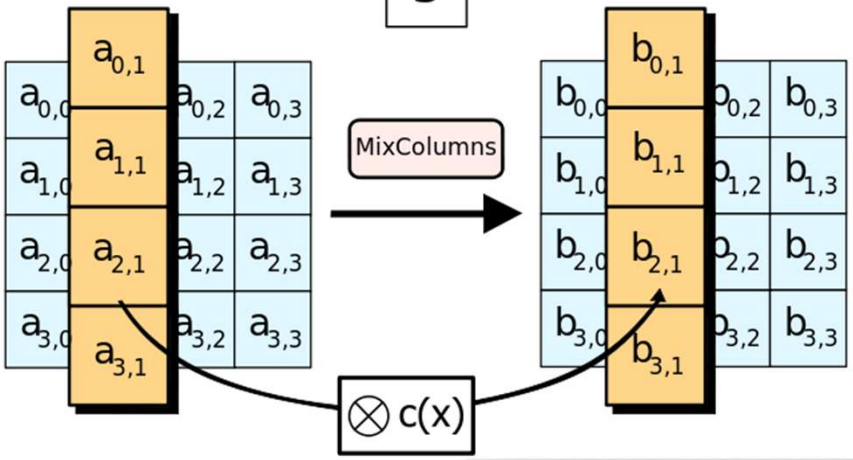


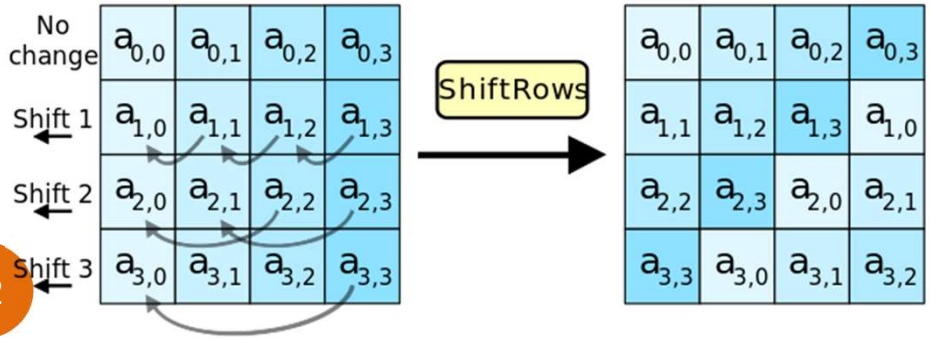
More on AES

Representing 16 bytes as 4 & 4 matrix









$$\begin{bmatrix} b_{0,j} \\ b_{1,j} \\ b_{2,j} \\ b_{3,j} \end{bmatrix} = \begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix} \begin{bmatrix} a_{0,j} \\ a_{1,j} \\ a_{2,j} \\ a_{3,j} \end{bmatrix}$$

AES的安全性

- 选项：
  - 10 rounds for 128-bit keys.
  - 12 rounds for 192-bit keys.
  - 14 rounds for 256-bit keys.
- 主要攻击：
  - 暴力破解：降低复杂度
  - 旁路（Side-channel）攻击

练习

假设TEA算法用128bit的密钥加密64bit的明文密码快，该密码块首先被分为两个32bit的块（L_0, R_0），然后用4个32bit的密钥块进行加密，假设第i轮和第i+1轮的计算方法如下：

$$L_i = R_{i-1}$$
$$R_i = L_{i-1} \boxplus F(R_{i-1}, K_0, K_1, \delta_i)$$
$$L_{i+1} = R_i$$
$$R_{i+1} = L_i \boxplus F(R_i, K_2, K_3, \delta_{i+1})$$

其中δ_i是预先定义好的常数，F的定义如下

$$F(R_i, K_i, K_j, \delta_i) = ((R_i \ll 4) \boxplus K_i) \oplus (R_i \gg 5) \boxplus K_j) \oplus (R_i \boxplus \delta_i)$$

画出TEA的密码结构图

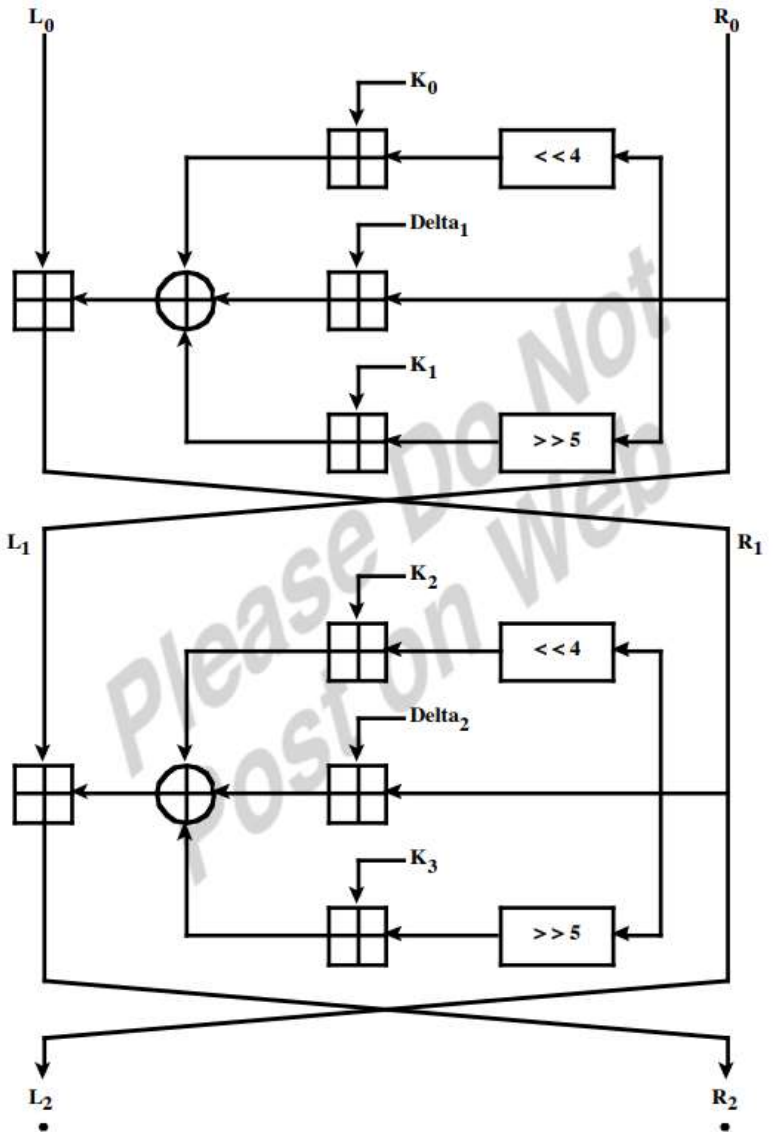
$$F(R_i, K_i, K_j, \delta_i) = ((R_i \ll 4) \boxplus K_i) \oplus (R_i \gg 5) \boxplus K_j) \oplus (R_i \boxplus \delta_i)$$

$$L_i = R_{i-1}$$

$$R_i = L_{i-1} \boxplus F(R_{i-1}, K_0, K_1, \delta_i)$$

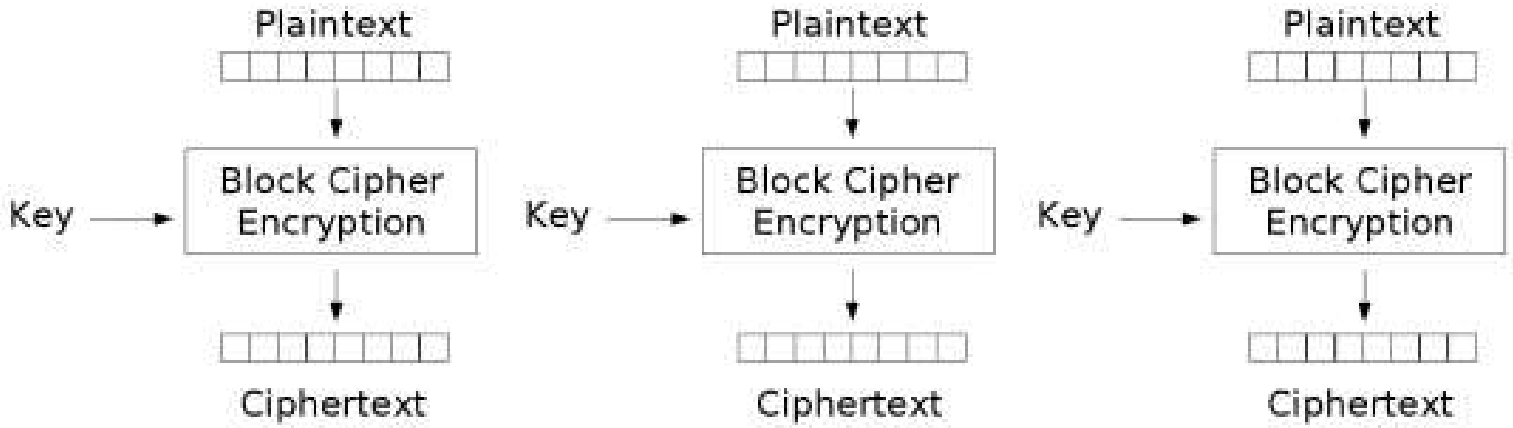
$$L_{i+1} = R_i$$

$$R_{i+1} = L_i \boxplus F(R_i, K_2, K_3, \delta_{i+1})$$

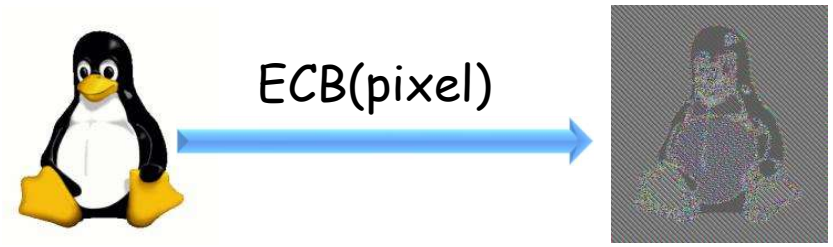


2. 分组密码工作模式

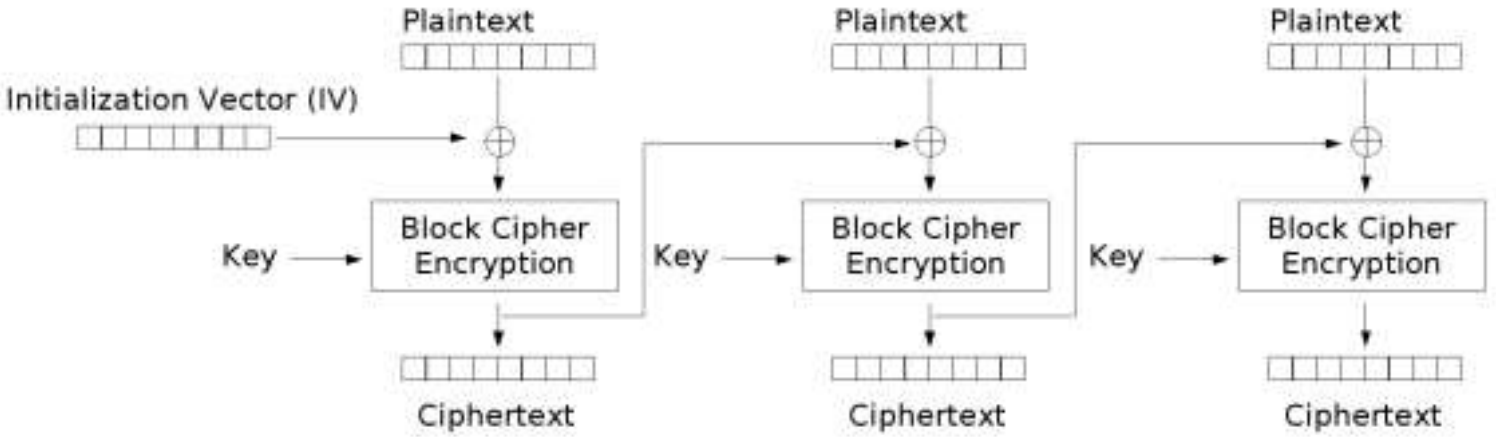
Electronic Codebook (ECB)



- 优点：可以并行计算
- 缺点：同样的原文块会被加密成相同的密文块

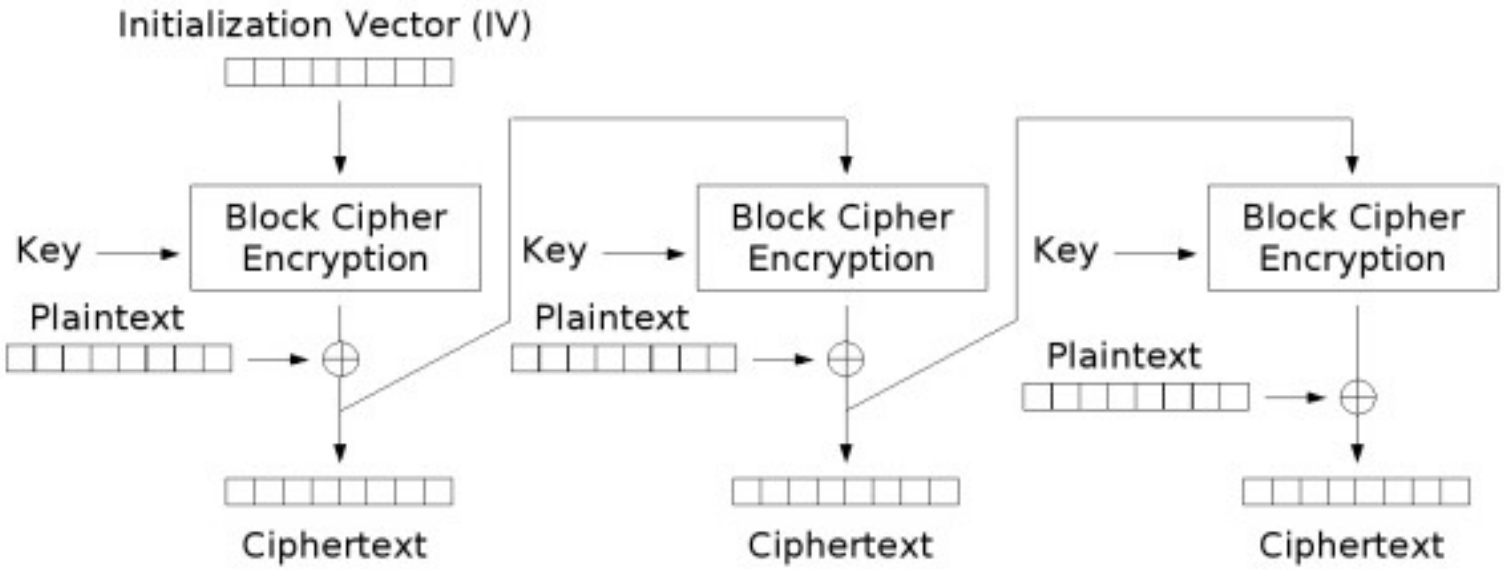


Cipher-Block Chaining（CBC）



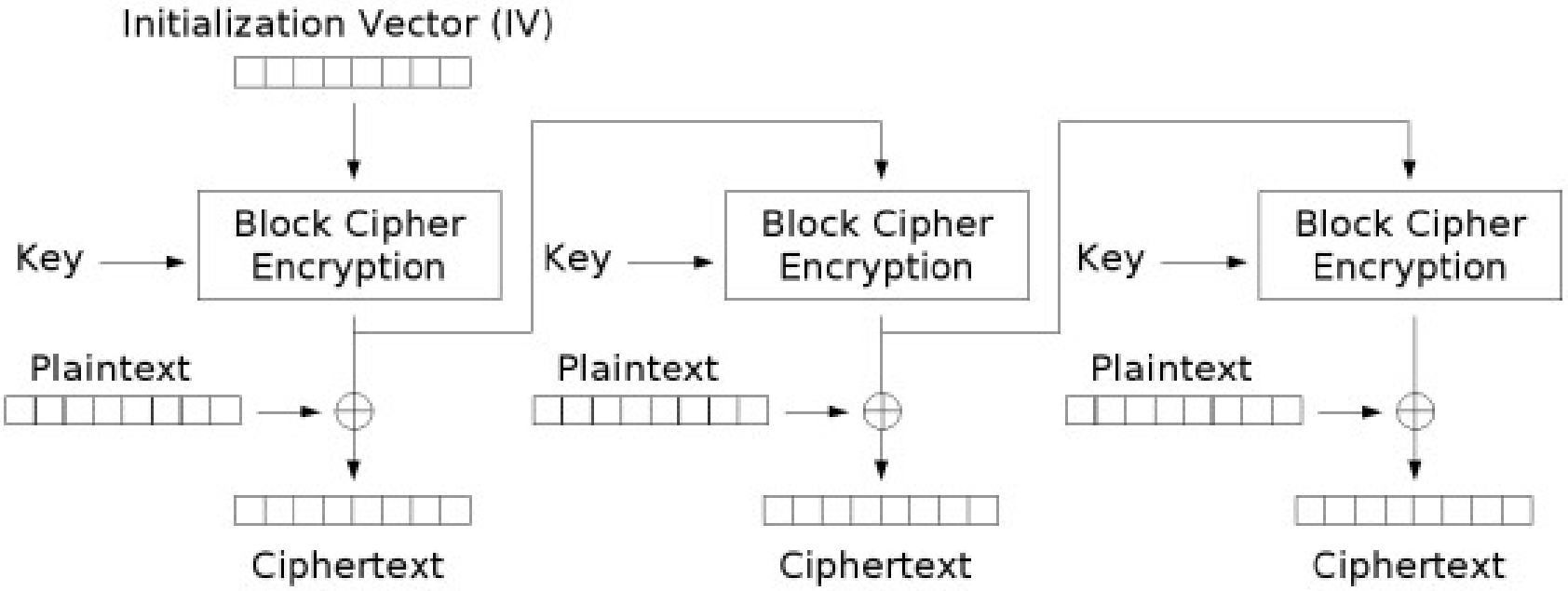
- 加密不能并行
- 解密可以并行

Cipher Feedback Mode (CFB)



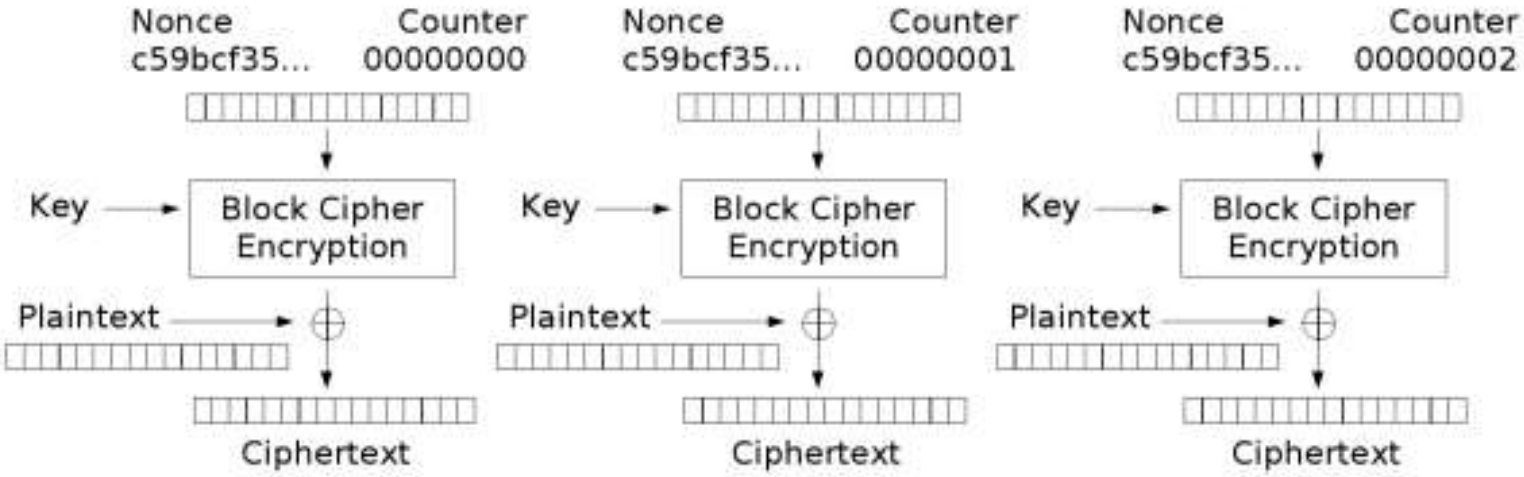
- 加密不能并行
- 解密可以并行
- 对流式数据支持较好（最后一个数据块无需定长）

Output Feedback Mode (OFB)



- 解密可以部分并行（需初始化密钥）
- 对流式数据支持较好（最后一个数据块无需定长）

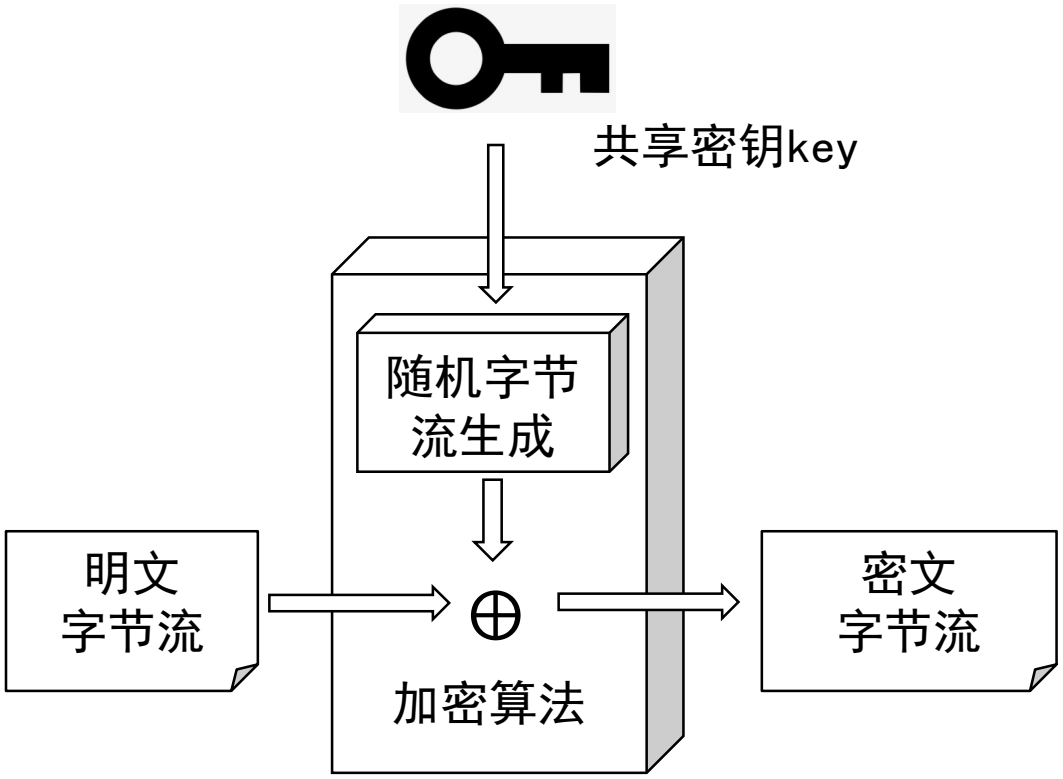
Counter Mode (CTR)



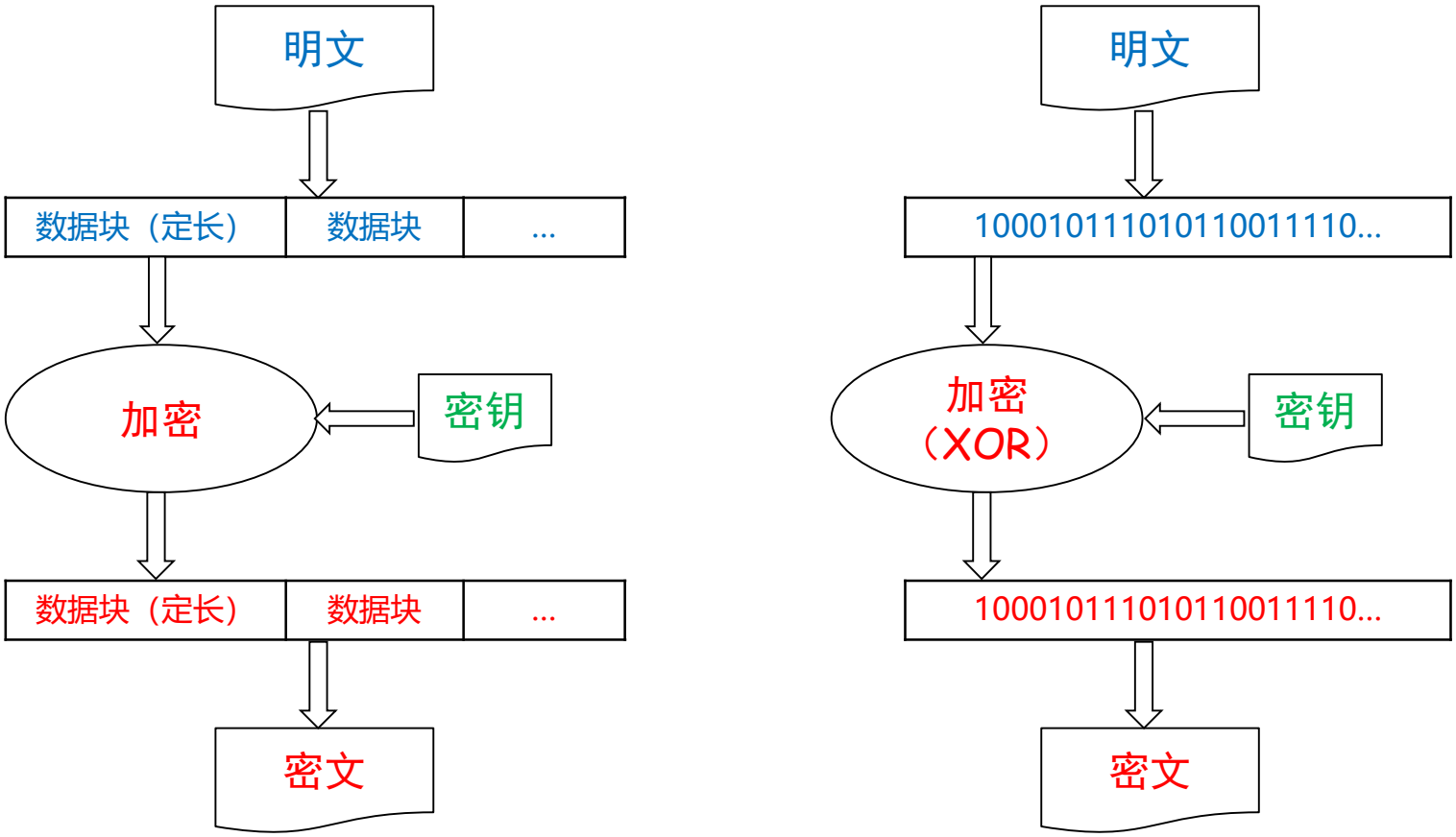
- 加密、解密都可以并行

3. 流密码

流密码算法



块密码VS流密码


明文
数据块（定长）
数据块
...
加密
密钥
数据块（定长）
数据块
...
密文
明文
10001011101011001111O...
加密
（XOR）
密钥
10001011101011001111O...
密文

RC4算法：初始化

Key-scheduling algorithm (KSA)　　1987年由Ron Rivest发明

```
for i from 0 to 255
    S[i] := i
endfor
j := 0
for i from 0 to 255
    j := (j + S[i] + key[i mod keylength]) mod 256
    swap values of S[i] and S[j]
endfor
```

练习：假设key长度为8 bit， key = 01000001，数组S长度为8，应用KSA算法

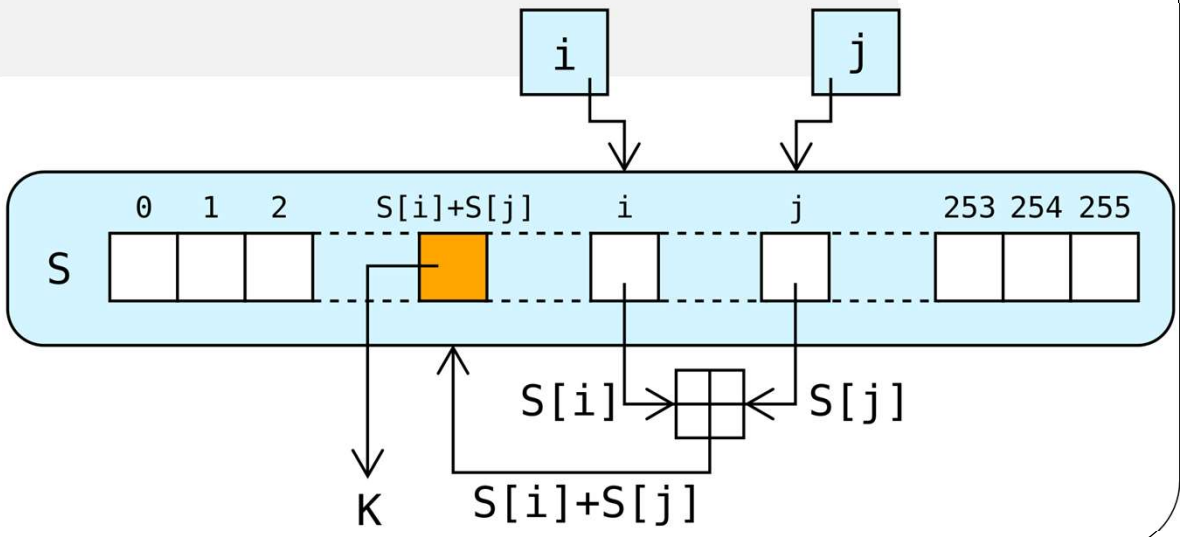
S[8] = {0,…7}

i=0, j=0, S[8] = {0,…7}
i=1, j=2, S[8] = {0,2,1,3,4,5,6,7}
…
i=7, S[8] = {0,1,3,4,5,6,7,2}

RC4算法： 随机字节生成

Pseudo-random generation algorithm (PRGA)

```
i := 0 j := 0
while GeneratingOutput:
    i := (i + 1) mod 256
    j := (j + S[i]) mod 256
    swap values of S[i] and S[j]
    K := S[(S[i] + S[j]) mod 256]
    output K
endwhile
```



3,0,3,3,2,35,1,1,0,0,1,7,2,1,6,0,
2,4,7,7,7,1,7,4,4,7,7,1,5,5,1,...

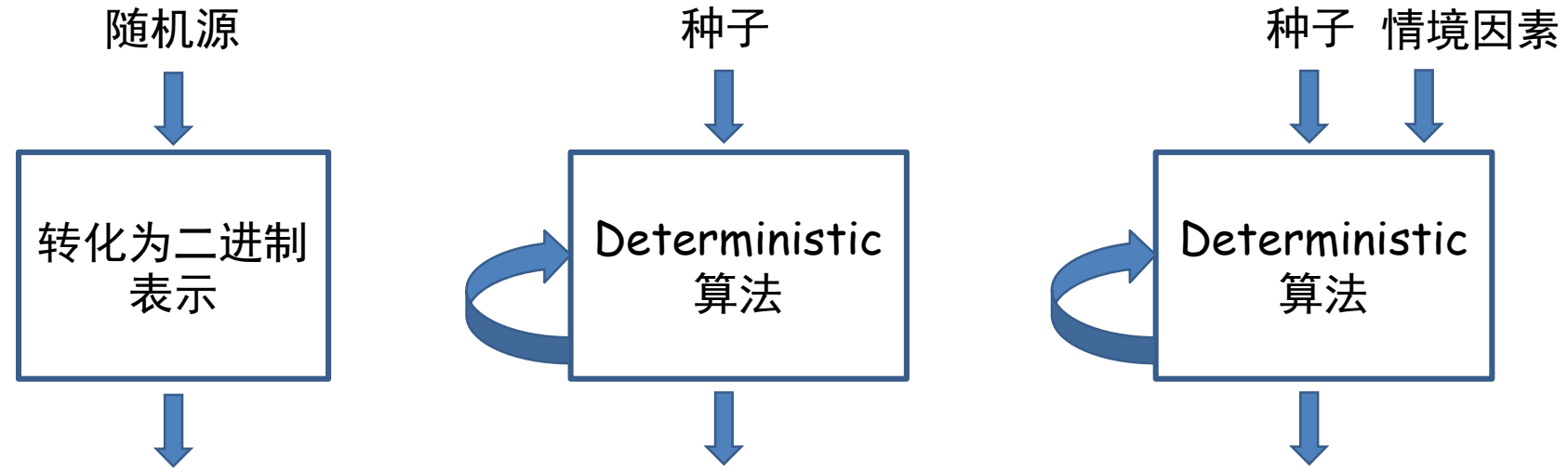
随机数算法的重要特性

- 随机性
  - 均匀分布（Uniform Distribution）： 0和1出现的频次应当相同。
  - 独立性（Independence）：基于任意子序列不能推出来其它部分。
- 不可预测性
  - 在随机数序列中，每一个随机数与前序随机数无关。

伪随机数和真随机数

- 随机数生成算法是deterministic的，因此生成的随机数之间不是绝对随机的。
- 如果可以通过随机性测试，则称为伪随机数

TRNG、PRNG、和PRF



如何构造PRNG？

- 专用随机数生成，如RC4中的随机数
- 基于已有密码算法：
  - 对称加密
  - 哈希算法

练习

假设有一个真随机数{0|1}*生成器，生成1的概率是$0.5+\delta$，生成0的概率是$0.5-\delta$，$0<\delta<0.5$；假设有一个纠偏算法对随机序列进行（非重叠）检查，如果检测到00或11序列直接丢弃，如果是01序列则替换为0，10序列则替换为1，请问：

a) 原始随机序列中00、11、01、10的概率分别是多少？
b) 纠偏后的序列中0和1的概率分别是多少？
c) 如果要输出长度为x的随机序列，需要长的原始输入？

Pair	Probability	
00	$(0.5-\partial)^2$	$= 0.25-\partial+\partial^2$
01	$(0.5-\partial)\times(0.5+\partial)$	$= 0.25-\partial^2$
10	$(0.5+\partial)\times(0.5-\partial)$	$= 0.25-\partial^2$
11	$(0.5+\partial)^2$	$= 0.25+\partial+\partial^2$

第二部分：哈希算法

HASH函数的功能要求

- 实用性：
  - H(x)=y可应用于任意大小的数据x；
  - 对于任意输入x，H(x)=y的结果y是固定长度；
  - H(x)=y的计算复杂度不高，如线性复杂度；
- 单向函数（one-way function）：给定结果y，不能计算出x，使得H(x)=y；
- 冲突对抗（collision resistant）
  - 弱冲突对抗：给定任意x_1，不能计算出x_2，使得$H(x_1)=H(x_2)$
  - 强冲突对抗：不能计算出任意(x_1, x_2)，使得$H(x_1)=H(x_2)$

Birthday Attack

- 暴力破解复杂度取决于哈希的函数的结果长度：2^n？
  - 碰撞成功的概率接近：$2^{-n/2} > 2^{-n}$
- 假设一个班级有n名学生，至少有两个学生同一天生日的概率是？

$$1 - \frac{365!}{(365-n)!*365^n} = 70.63\%$$

- ✓ n = 2, p = 0.2%
- ✓ n = 10, p =11.7%
- ✓ n = 20, p =41.1%
- ✓ n = 40, p =89.1%
- ✓ n = 50, p =97%

练习

假设针对消息$M = (a_1, a_2, \ldots a_n)$，有如下哈希函数定义

$$Hash(M) = \left(\sum_{i=1}^{t} a_i \right) \bmod n$$

如果$M = (189,632,900,722,349)$， $n = 989$，计算哈希结果 814

该哈希函数是否满足哈希函数的要求？

- H(x)=y可应用于任意大小的数据x；
- 对于任意输入x，H(x)=y的结果y是固定长度；
- H(x)=y的计算复杂度不高，如线性复杂度；
- 给定结果y，不能计算出x，使得H(x)=y；
- 弱冲突对抗：给定任意x_1，不能计算出x_2，使得$H(x_1)=H(x_2)$
- 强冲突对抗：不能计算出任意(x_1，x_2)，使得$H(x_1)=H(x_2)$

练习

如果哈希函数改为如下形式呢？

$$Hash(M) = \left(\sum_{i=1}^{t} (a_i)^2 \right) \bmod n$$

- 给定结果y，不能计算出x，使得H(x)=y；
- 弱冲突对抗：给定任意x_1，不能计算出x_2，使得$H(x_1)=H(x_2)$

常见哈希算法

- MD5
  - 1991, Ron Rivest设计
  - 128-bit (16-byte) 哈希值
  - 64轮运算
  - 不安全：可以在数秒找到冲突
- SHA1
  - FIPs 180, 1993, 基于MD4设计
  - 160bits 散列码长度, 80 轮
  - 增强版SHA2（FIPS180-3, 2002）和SHA3（2007）
  - 安全性
    - 2005，王小云发现找到2个散列值相同的输入数据复杂度是$2^{69}<2^{80}$
    - NIST 发布消息应当在2010年前将生产系统中的SHA1算法替换为SHA2

SHA1算法

初始化变量

```
h0 = 0x67452301
h1 = 0xEFCDAB89
h2 = 0x98BADCFE
h3 = 0x10325476
h4 = 0xC3D2E1F0
```

预处理：消息补全，使其长度为512的倍数

append the bit '1' to the message

append 0 ≤ k < 512 bits ’0’, 使得消息的长度为： −64 ≡ 448 (mod 512)

将于原消息的长度ml作为64bit的大序整数。

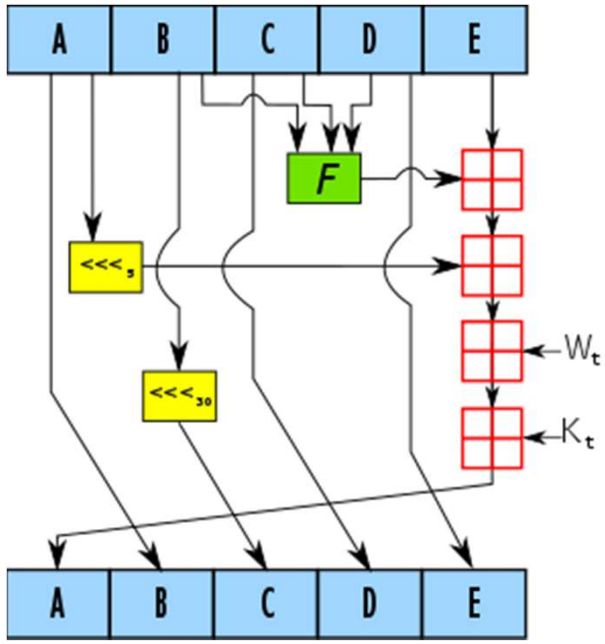
对于每一块长度512bit的数据,将其表示为长度为16的32bit数组w[16]，依次进行运算更新h0-h4。

最终结果

```
hh = (h0 leftshift 128) or (h1 leftshift 96) or (h2 leftshift 64)
or (h3 leftshift 32) or h4
```

主要循环体：

```
for i from 16 to 79
        w[i] = (w[i-3] xor w[i-8] xor w[i-14] xor w[i-16]) leftrotate 1
a = h0, b = h1, c = h2, d = h3, e = h4
for i from 0 to 79
        if 0 ≤ i ≤ 19 then
                f = (b and c) or ((not b) and d)
                k = 0x5A827999
        else if 20 ≤ i ≤ 39
                f = b xor c xor d
                k = 0x6ED9EBA1
        else if 40 ≤ i ≤ 59
                f = (b and c) or (b and d) or (c and d)
                k = 0x8F1BBCDC
        else if 60 ≤ i ≤ 79
                f = b xor c xor d
                k = 0xCA62C1D6
        temp = (a leftrotate 5) + f + e + k + w[i]
        e = d
        d = c
        c = b leftrotate 30
        b = a
        a = temp
```



更新h0-h4：

```
h0 = h0 + a h1 = h1 + b h2 = h2 + c h3 = h3 + d h4 = h4 + e
```

算法强度对比

Algorithm		Year	Output size (bits)	Collisions found	Security
MD5		1991	128	≤18	No
SHA-0		1993	160	<34	No
SHA-1		1995	160	<63	No
SHA-2	SHA-256/224	2001	256/224	≥112	Yes
	SHA-512/384	2001	512/384	≥192	Yes
SHA-3		2015	224/256/384/512	≥112	Yes

如何进行消息认证？

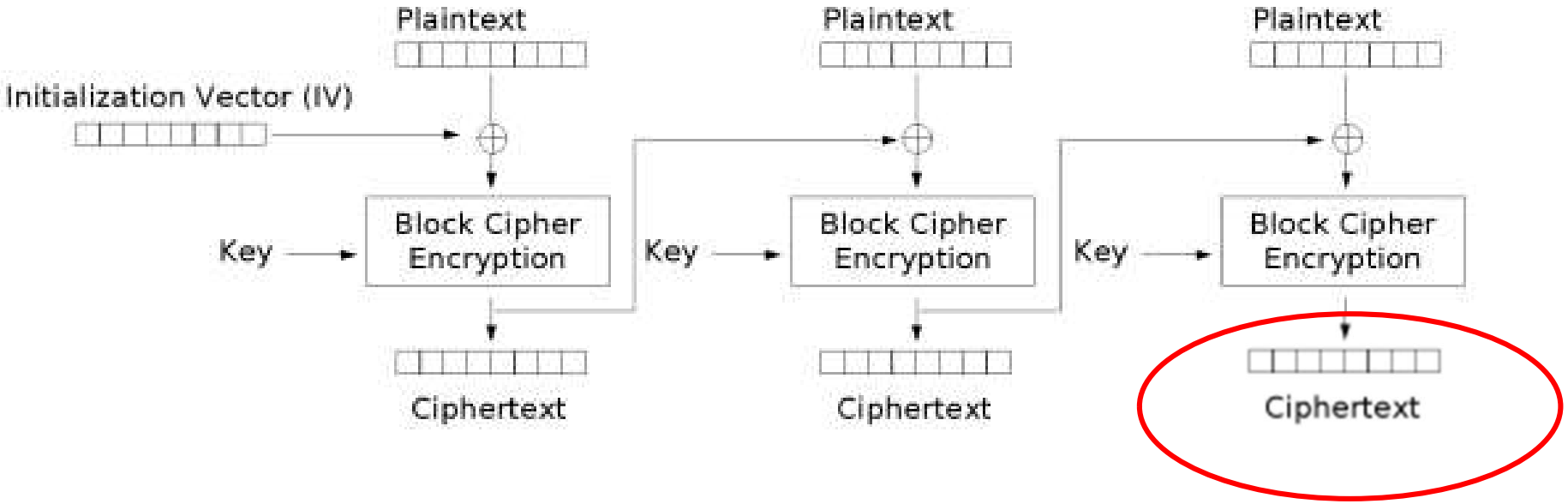
基于哈希算法？ $MAC_M=HASH(M)$

基于对称加密： $MAC_M=Enc(K_{AB},M)$

基于非对称加密： $MAC_M=Enc(PriKey_A,M)$

基于HMAC算法： $MAC_M=Hash(S||M)$

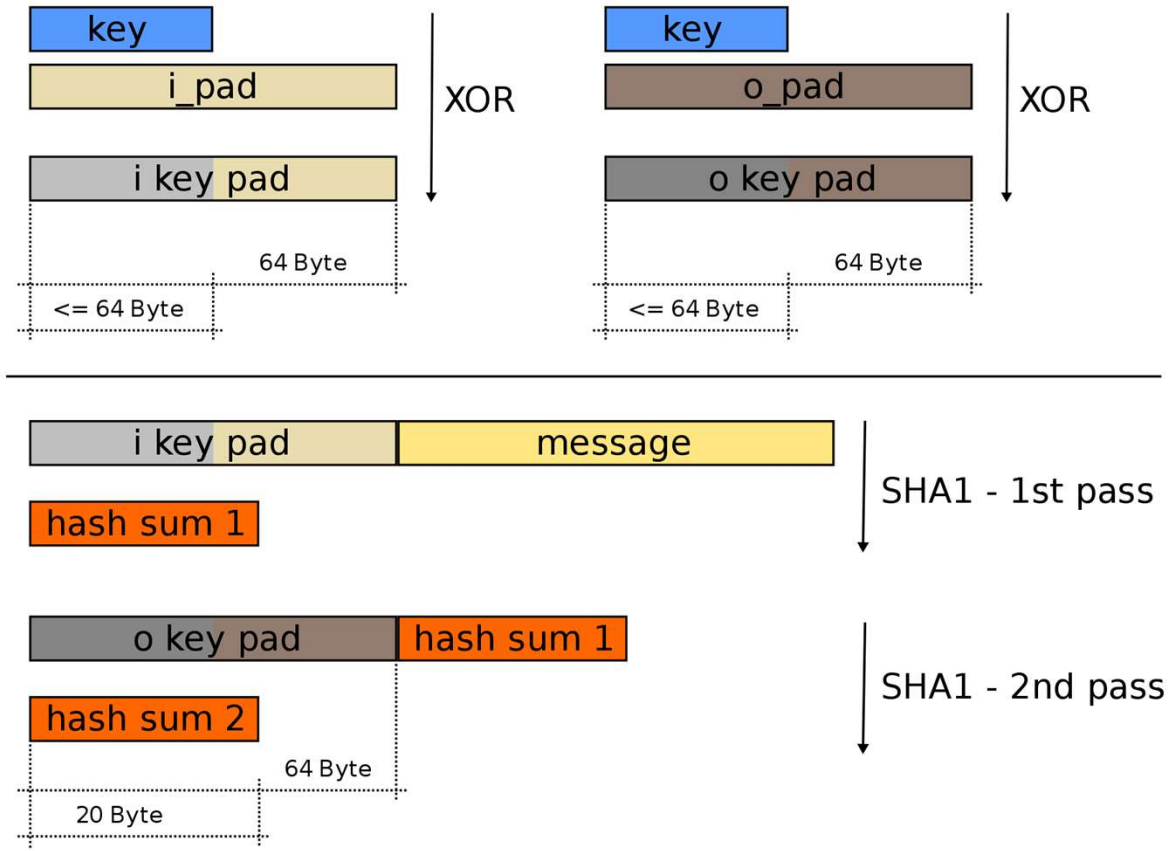
基于块加密的MAC



取最左侧任意长度的数字

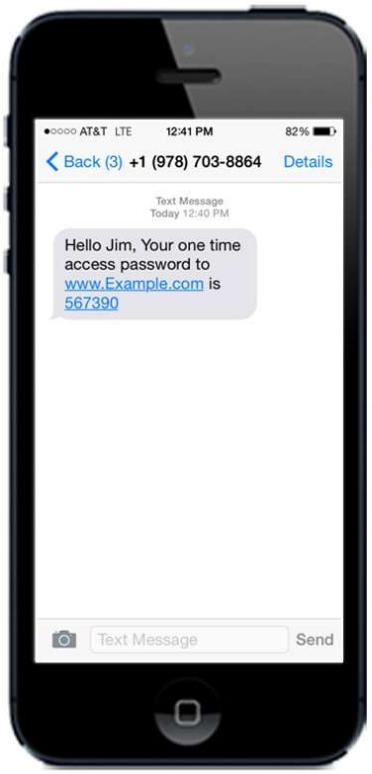
HMAC

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



ipad:00110110...
opad:01011100...

应用场景：动态口令

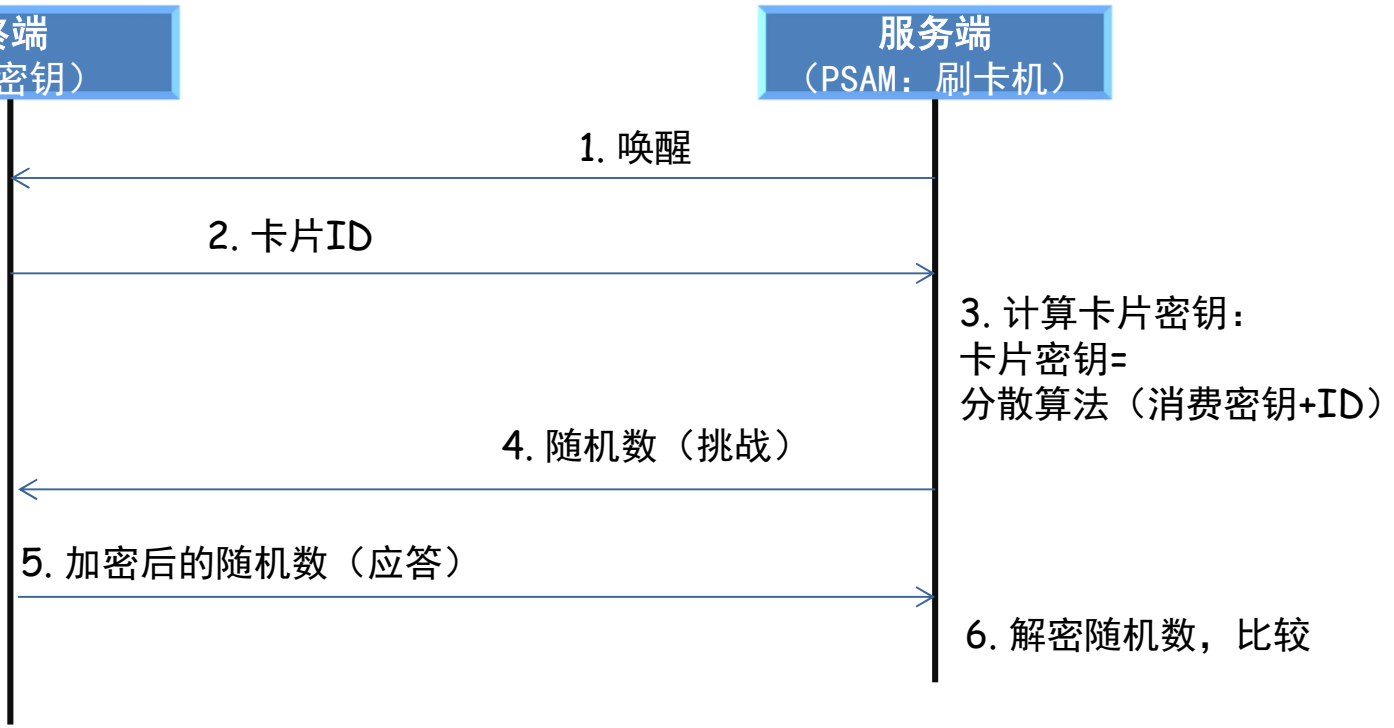


- 客户端内置密码种子Key，服务端保存Key的拷贝
- 密码：HMAC(Key, Time)

应用场景：公交卡消费

密钥分散：

- 母密钥（服务端）：Key
- 子密钥： $Key_i = HMAC(Key, ID)$



第三部分：非对称加密

对称加密的主要问题

- 密钥的保密性和共享需求：
  - 保密性：密钥不能泄漏，否则影响密文的安全性
  - 共享性：由于加密解密密钥相同，A给B发消息需要知道B的密钥
- 挑战：如何安全地共享密钥？

1. Diffie-Hellman密钥交换

Diffie-Hellman密钥交换

基于离散对数问题：
$g^x = y \bmod n$
=> x=?

Alice

Bob

共享p(质数)和g
$p = 23, g = 5$

生成私钥：$prk_A = 4$

计算公钥：
$puk_A = g^{prkA} \bmod p$
$= 5^4 \bmod 23 = 4$

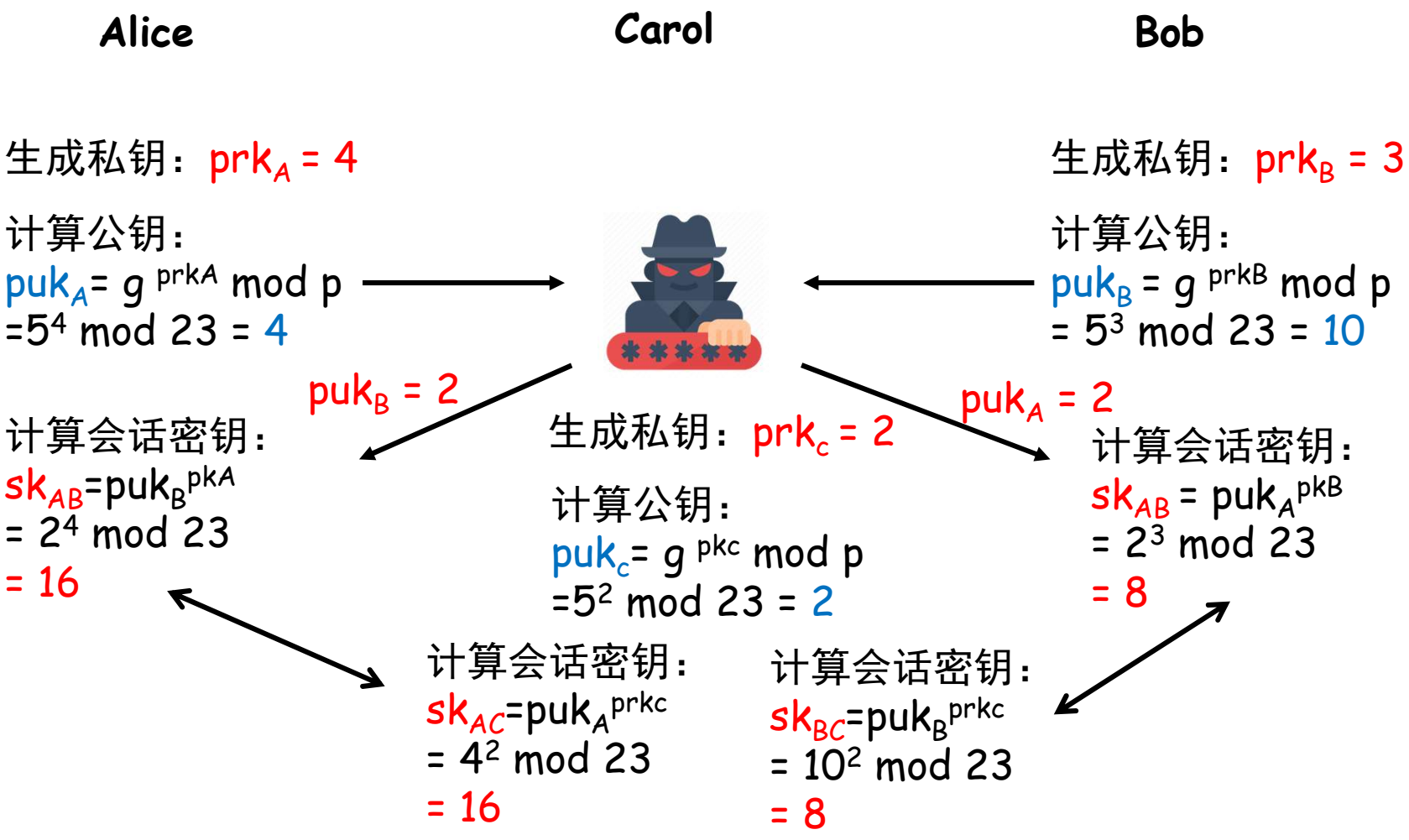
计算会话密钥：
$sk_{AB} = puk_B^{prkA}$
$= 10^4 \bmod 23$
$= 18$

生成私钥：$prk_B = 3$

计算公钥：
$puk_B = g^{prkB} \bmod p$
$= 5^3 \bmod 23 = 10$

计算会话密钥：
$sk_{AB} = puk_A^{prkB}$
$= 4^3 \bmod 23$
$= 18$

MITM攻击


Alice
Carol
Bob
生成私钥：$prk_A = 4$
计算公钥：
$puk_A = g^{prkA} \bmod p$
$=5^4 \bmod 23 = 4$
生成私钥：$prk_B = 3$
计算公钥：
$puk_B = g^{prkB} \bmod p$
$= 5^3 \bmod 23 = 10$
$puk_B = 2$
$puk_A = 2$
计算会话密钥：
$sk_{AB}=puk_B^{pkA}$
$= 2^4 \bmod 23$
$= 16$
生成私钥：$prk_c = 2$
计算公钥：
$puk_c = g^{pkc} \bmod p$
$=5^2 \bmod 23 = 2$
计算会话密钥：
$sk_{AB} = puk_A^{pkB}$
$= 2^3 \bmod 23$
$= 8$
计算会话密钥：
$sk_{AC}=puk_A^{prkc}$
$= 4^2 \bmod 23$
$= 16$
计算会话密钥：
$sk_{BC}=puk_B^{prkc}$
$= 10^2 \bmod 23$
$= 8$

2. RSA算法

RSA算法

- 1977年，Rivest Shamirh Adleman发明
- 安全性：基于质因数分解问题
  - 只要其钥匙的长度足够长，用RSA加密的信息实际上是不能被解破的。
  - 2009年12月12日，编号为 RSA-768 （768 bits, 232 digits）数也被成功分解。
  - 美国NIST和中国国家密码局分别于2009年和2011年发布了RSA1024算法的升级要求。即通过升级RSA1024到RSA2048或ECC算法来保证密码算法的安全性。

RSA密钥生成

- 初始化：
  - 选取两个不同的大质数：p,q
  - 计算n = p*q
  - 计算φ(n) = (p – 1)(q – 1)
- 生成公钥
  - 选取整数e, 使得e满足：1 < e < φ(n)，并且gcd(e,φ(n)) = 1
  - 公钥为：(e,n)
- 计算d = e^{-1} mod φ(n);
  使用扩展欧几里得算法。
  - 私钥为：(d,n)

举例：
p=13, q=7
n=13*7=91
φ(n)=(13-1)*(7-1)=72
e=11
公钥：(11, 91)
d=11^{-1} mod 72=59,
私钥：(59, 91)

应用扩展欧几里得算法计算私钥

72 = 11*6 + 6
11 = 6*1 + 5
6 = 5*1 + 1

1 = 6 -5
1 = 6 - (11-6) = 2*6 - 11
1 = 2*(72-11*6) - 11= 2*72 - 11*13

d = -13 mod 72 = 59

计算：https://www.programiz.com/python-programming/online-compiler/

RSA加密/解密运算

加密运算

$ciphertext=plaintext^e \bmod n$

解密运算

$plaintext=ciphertext^d \bmod n$

举例：
公钥：(11, 91)
私钥：(59, 91)
明文：65
密文：$65^{11} \bmod 91=39$
明文：$39^{59} \bmod 91=65$

证明$m^{ed} \equiv m \bmod n$

m^{ed}

$= m^{1+k\varphi(n)}$

$= m*(m^{\varphi(n)})^k$

根据欧拉定理：如果m和n互质，则$m^{\varphi(n)} \bmod n = 1$

m^{ed}

$\equiv m*(1)^k \bmod n$

$\equiv m \bmod n$

练习

p=3，q=11，e=7，M=5，计算私钥并用私钥进行加密、私钥解密：

$\varphi(n)=(11-1)*(3-1)=20$
$d=7^{-1} \bmod 20 = 3,$

$20 = 7*2 + 6$
$7 = 6*1 + 1$

$1 = 7 - 6$
$1 = 7 - (20-7*2) = 7*3 - 20$

加密
$5^3 \bmod 33 = 26,$
解密
$26^7 \bmod 33 = 8$

练习

p=5，q=11，e=3，M=9，计算私钥并用私钥进行加密、私钥解密：

$\varphi(n)=(11-1)*(5-1)=40$
$d=3^{-1} \bmod 20 = 27$,

$40 = 3*13 + 1$

$1 = -13 \bmod 40 = 27$

加密
$9^{27} \bmod 55 = 4$,
解密
$4^3 \bmod 55 = 9$

3. ECC算法

离散对数问题

群上的数学计算难题

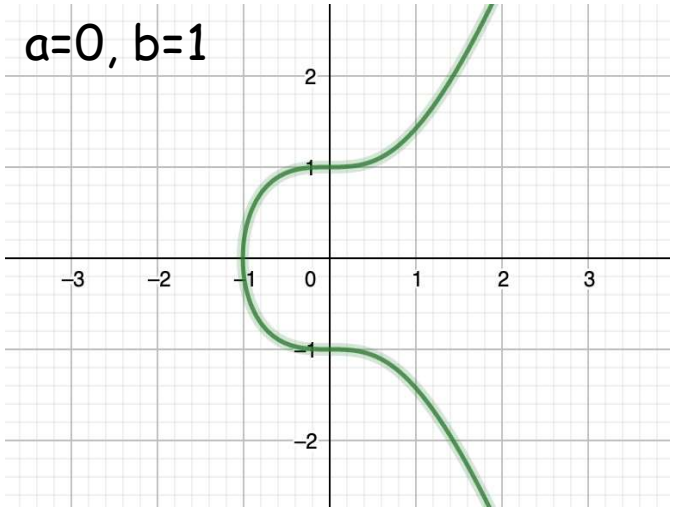
$g^x = y \bmod n$

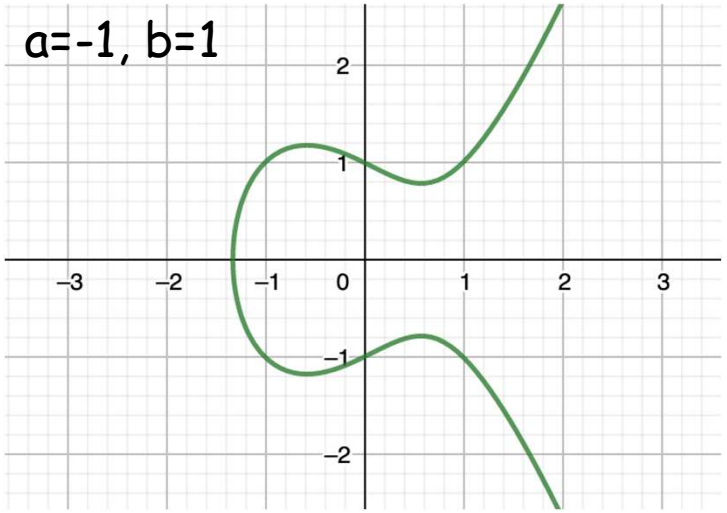
=> x=?

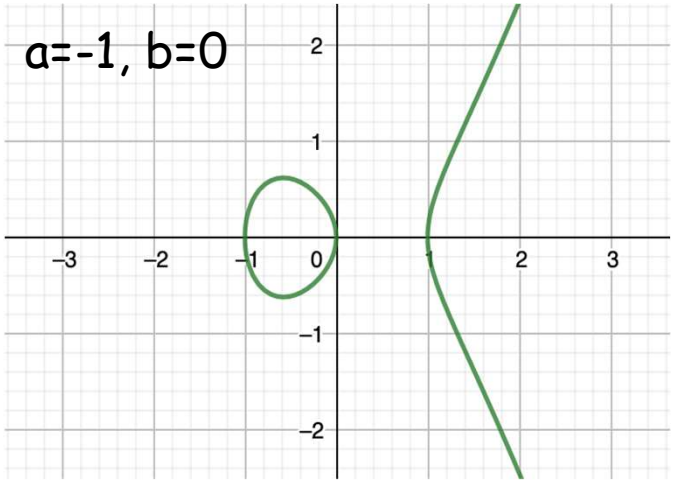
ECC基于椭圆曲线上的离散对数问题
1985由 Neal Koblitz 和 Victor S. Miller发明

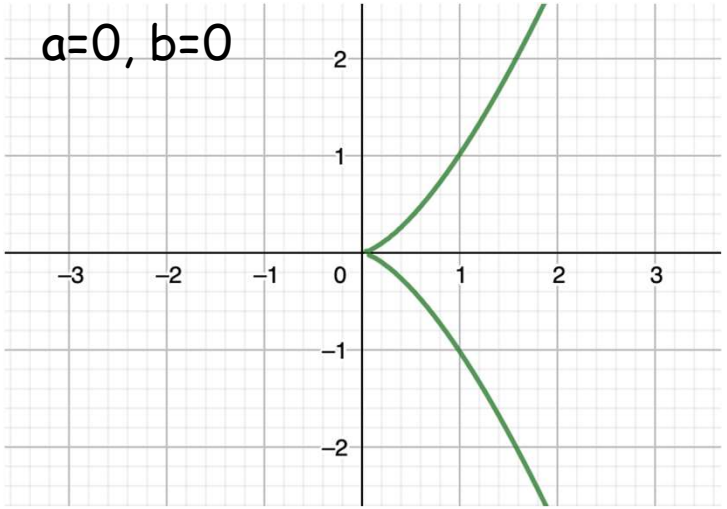
椭圆曲线

$y^2=x^3+ax+b$





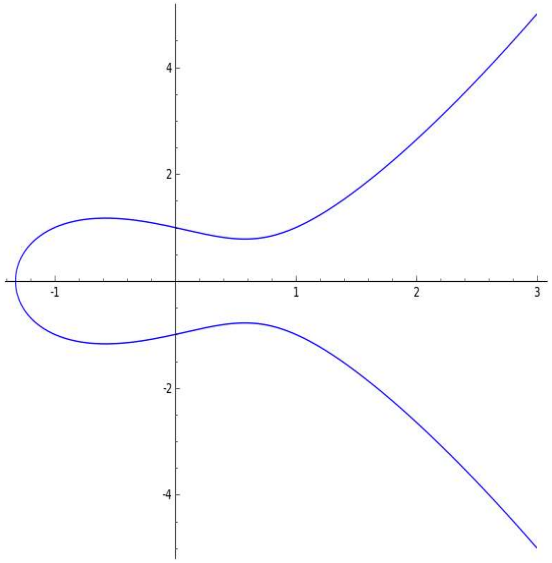


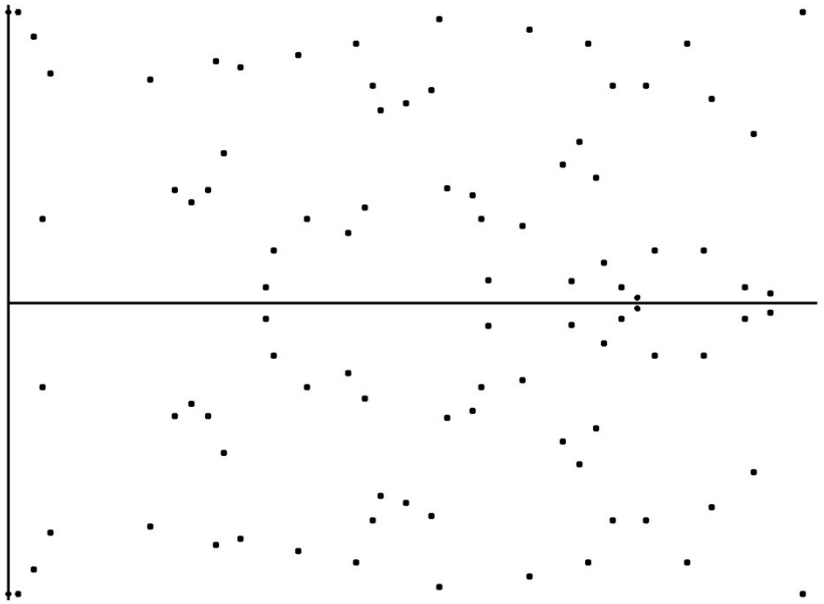


$4a^3 + 27b^2 \neq 0$

https://www.geogebra.org/graphing?lang=en

有限域上的椭圆曲线



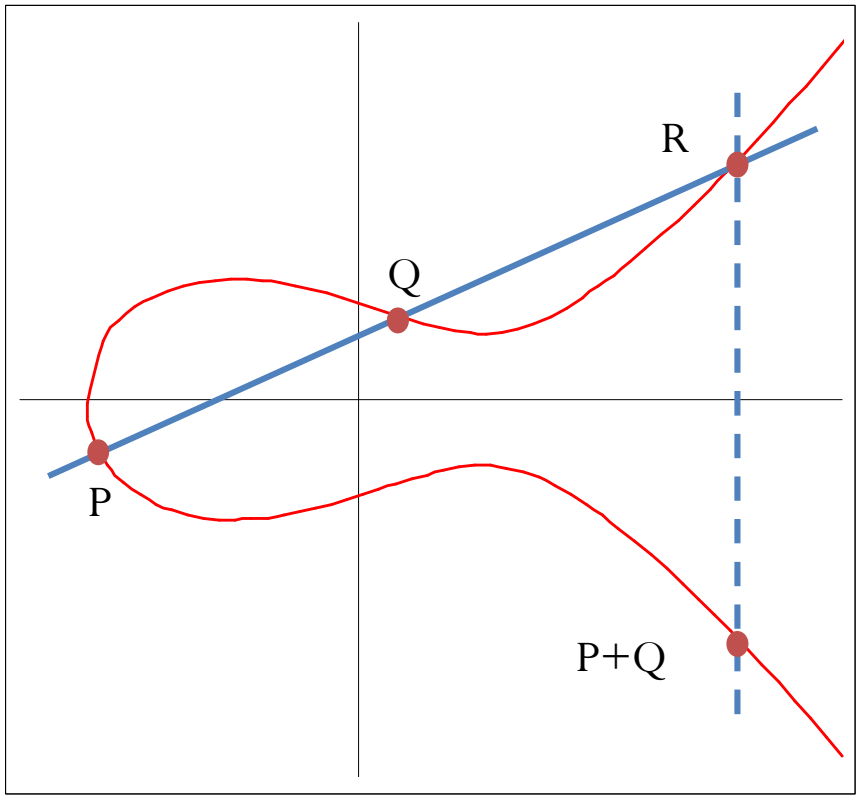


$y^2=x^3+ax+b$

$y^2 \equiv x^3+ax+b \pmod{p}$

根据庞加莱(Poincaré)定理：(E(𝔽q)，+)是一个阿贝尔群（或加群）

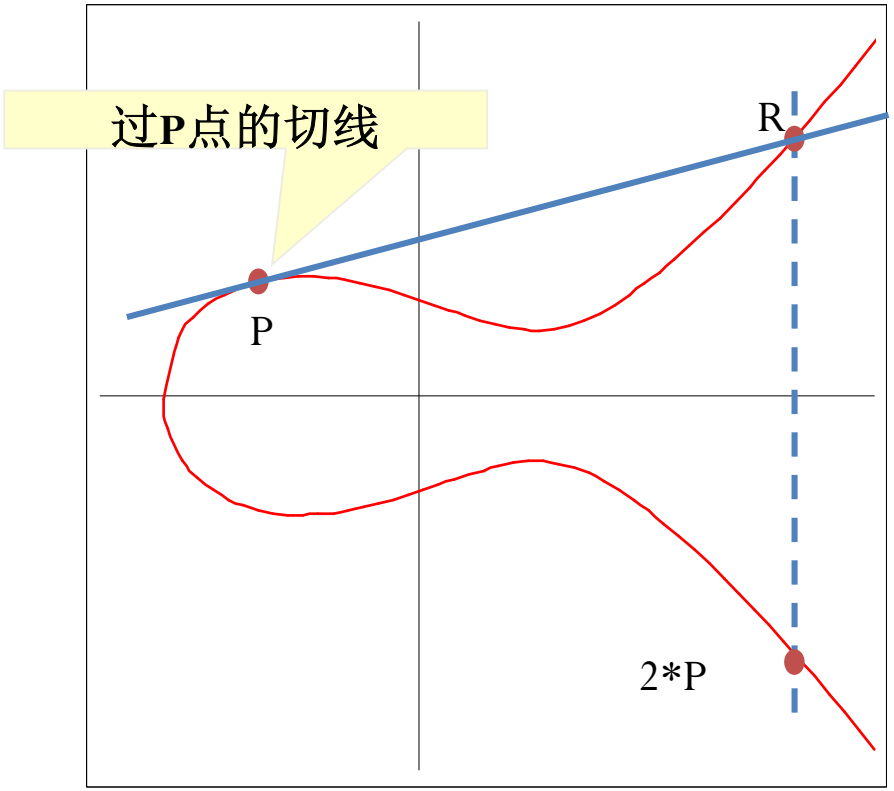
点的加法



- $P + Q = -R$
- $(x_p, y_p) \oplus (x_q, y_q) = (x_{p+q}, y_{p+q})$
  - $s = (y_q - y_p)/(x_q - x_p)$
  - $x_{p+q} = s^2 - 2x_p$
  - $y_{p+q} = s(x_p - x_{p+q}) - y_p$

暂时不考虑特殊情况

倍点运算



- $2P = -R$
- $2(x_p, y_p) = (x_{2p}, y_{2p})$
  - $s = (3x_p^2 + a)/(2y_p)$
  - $x_{2p} = s^2 - 2x_p$,
  - $y_{2p} = s(x_p - x_{2p}) - y_p$

公钥和私钥

- 私钥：随机数
- 公钥：点，私钥 * G（倍点运算）
- 根据公钥和G计算私钥
- 不同国家ECC密码体系环境变量不同
  - 选择ECC算法在有限域上的椭圆曲线E(a,b)和大质数p;
  - 选择ECC算法的椭圆曲线上的一点G，存在nG=O，且n非常大。
  - 中国ECC-256算法（SM2）的环境变量如下

椭圆曲线$y^2=x^3+a*x+b$

p=FFFFFFFE FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 00000000 FFFFFFFF FFFFFFFF

a=FFFFFFFE FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 00000000 FFFFFFFF FFFFFFFC

b=28E9FA9E 9D9F5E34 4D5A9E4B CF6509A7 F39789F5 15AB8F92 DDBCBD41 4D940E93

n=FFFFFFFE FFFFFFFF FFFFFFFF FFFFFFFF 7203DF6B 21C6052B 53BBF409 39D54123

Gx=32C4AE2C 1F198119 5F990446 6A39C994 8FE30BBF F2660BE1 715A4589 334C74C7

Gy=BC3736A2 F4F6779C 59BDCEE3 6B692153 D0A9877C C62A4740 02DF32E5 2139F0A0

中华人民共和国密码法

- 2020年1月开始实施。
- 密码分类：
  - 第六条　国家对密码实行分类管理。
    - 密码分为核心密码、普通密码和商用密码。
  - 第七条　核心密码、普通密码用于保护国家秘密信息，核心密码保护信息的最高密级为绝密级，普通密码保护信息的最高密级为机密级。
    - 核心密码、普通密码属于国家秘密。密码管理部门依照本法和有关法律、行政法规、国家有关规定对核心密码、普通密码实行严格统一管理。
  - 第八条　商用密码用于保护不属于国家秘密的信息。
    - 公民、法人和其他组织可以依法使用商用密码保护网络与信息安全。

商用密码管理条例

- 商用密码技术属于国家秘密。国家对商用密码产品的科研、生产、销售和使用实行专控管理。
- 国家密码管理委员会及其办公室主管全国的商用密码管理工作。
- 商用密码的科研任务由国家密码管理机构指定的单位承担。
- 商用密码的科研成果，由国家密码管理机构组织专家按照商用密码技术标准和技术规范审查、鉴定。
- 商用密码产品由国家密码管理机构指定的单位生产。
- 商用密码产品由国家密码管理机构许可的单位销售。
- …

主要国密算法

- 对称：SM1, SM4
- 非对称：SM2
- 哈希算法：SM3

国家密码管理局：https://www.sca.gov.cn

课后阅读

- 《网络安全基础-应用与标准（第6版）》
  - 第2章：对称加密和消息机密性
  - 第3章：公钥密码和消息认证