

# Table of Contents

概述	1.1
<h2>加密与解密</h2>	
第一章：密码分类与历史	2.1
密码分类	2.1.1
密码常识	2.1.2
密码历史	2.1.3
第二章：对称密码	2.2
异或	2.2.1
DES	2.2.2
3DES	2.2.3
AES	2.2.4
第三章：分组密码	2.3
ECB	2.3.1
CBC	2.3.2
CFB	2.3.3
OFB	2.3.4
CTR	2.3.5
第四章：公钥密码	2.4
密钥配送	2.4.1
流程	2.4.2
时钟运算	2.4.3
加解密过程	2.4.4
攻击	2.4.5
证明	2.4.6
第五章：混合密码	2.5
概述	2.5.1
加密	2.5.2
解密	2.5.3
<h2>认证</h2>	
第六章：单项散列函数	3.1
哈希应用	3.1.1
分类	3.1.2
keccak	3.1.3
攻击	3.1.4
第七章：消息认证码	3.2
流程	3.2.1
HMAC	3.2.2
攻击	3.2.3
第八章：数字签名	3.3
流程	3.3.1
攻击	3.3.2
对比	3.3.3

第九章：证书	3.4
流程	3.4.1
PKI	3.4.2
攻击	3.4.3
第十章：SSL/TLS	3.5
概述	3.5.1
流程	3.5.2
攻击	3.5.3

## 应用与现实

第十一章：应用与现实	4.1
离线数字签名	4.1.1
比特币	4.1.2
完美密码	4.1.3
现实	4.1.4

# 概述

Written By

Hensh / xu.huang@100credit.com

---

## 什么是密码学？

- 密码学（在西欧语文中之源于希腊语κρυπτός，“隐藏的”，和γράφειν，“书写”）是研究如何隐密地传递信息的学科。
- 在现代特别指对信息以及其传输的数学性研究，常被认为是数学和计算机科学的分支，和信息论也密切相关。

## 什么是密码技术？

- 密码技术是保障信息安全的核心技术。
- 密码技术在古代就已经得到应用。历史久远，公元前2000年，古埃及就有使用密码的证据。
- 它是集数学、计算机科学、电子与通信等诸多学科于一身的交叉学科。
- 密码技术分为古典密码技术，近代密码技术，现代密码技术。

## 密码的重要性

- 隐私
- 企业
- 生命
- 国家
- 世界格局

# 第一章：密码分类与历史

## 密码学家的工具箱

- 对称密码
- 公钥密码
- 单向散列函数
- 消息验证码
- 数字签名
- 伪随机数生成器

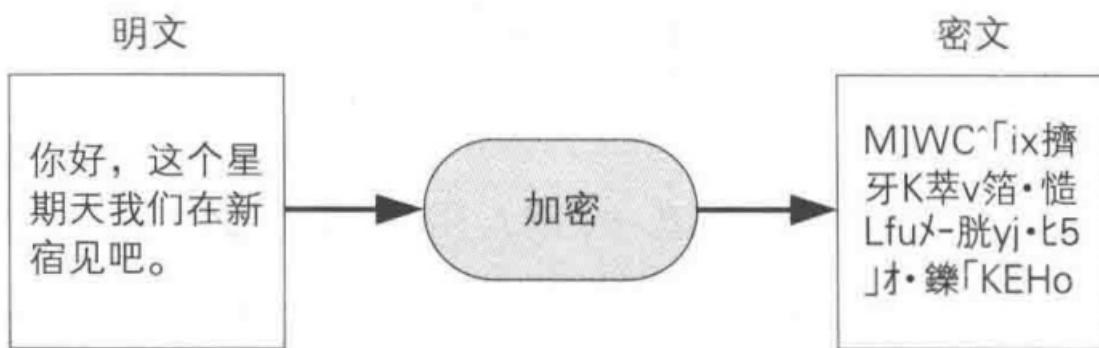
## 密码学三大历史

- 古典密码技术
- 近代密码技术
- 现代密码技术

## 密码分类

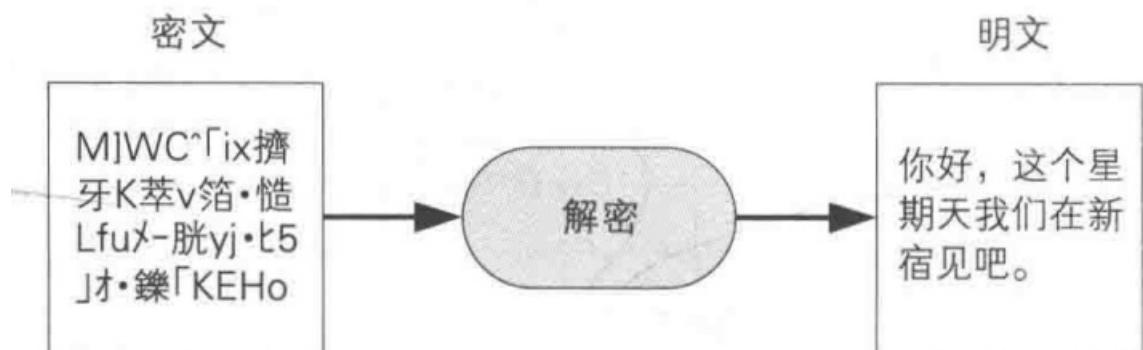
密码三大过程

加密



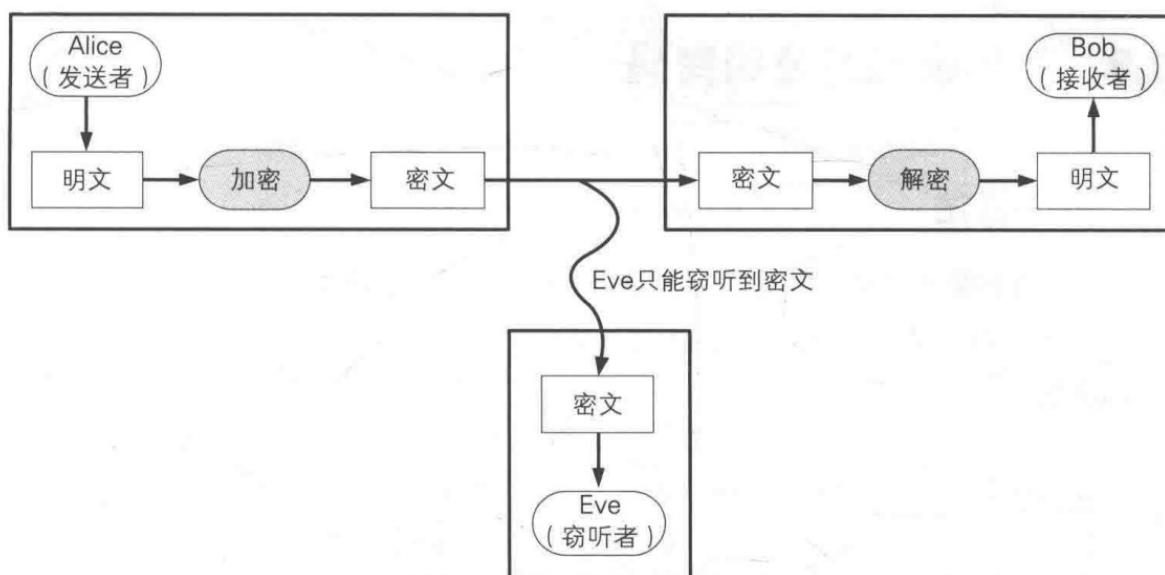
明文加密，变成看不懂的密文

解密



密文解密，还原为明文

窃听



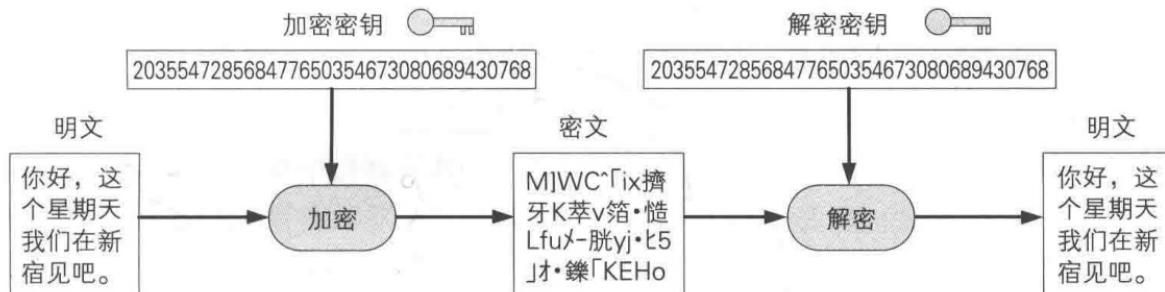
加密后发送，窃听者只能获取密文

## 加密解密

对称密码

加解密时，使用同一密钥的方式

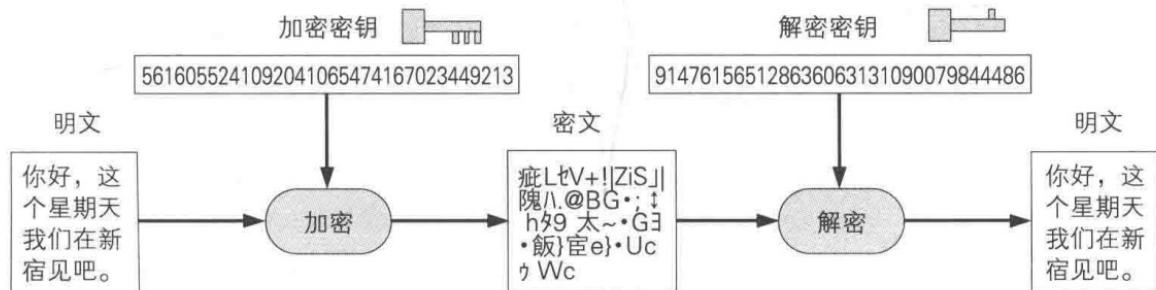
对称密码中，加密用的密钥和  
解密用的密钥是相同的



公钥密码

加解密时，使用不同密钥的方式

公钥密码中，加密用的密钥和  
解密用的密钥是不同的



## 其他密码

单向散列函数

一种保证完整性的技术，最常见的就是文件比较是否完全一致。

消息认证码

确认消息是否来自希望的对象，能识别篡改和做认证。

数字签名

能够防止伪装、篡改、否认等威胁的技术。

伪随机数生成器

承担密钥生成重要责任。

密码应用大多使用算法来生成随机数。这些算法是确定的，所以产生的序列并非统计随机的。

不过，要是算法好的话，产生的序列可以经受随机性检测，这样的数一般称为伪随机数。



## 密码常识

### 不要使用保密的密码算法

- 密码算法的秘密早晚会公诸于世。
- 开发高强度的密码非常困难。

### 使用低强度的密码比不使用什么密码更加危险

16世纪，苏格兰女王写了一封她认为只有自己能破解的密信，信中内容为刺杀伊丽莎白女王，结果把她送上了断头台。

### 任何密码总有一天总会被破解

- 暴力破解

### 密码只是信息安全的一部分

- 因为社会工程学的存在

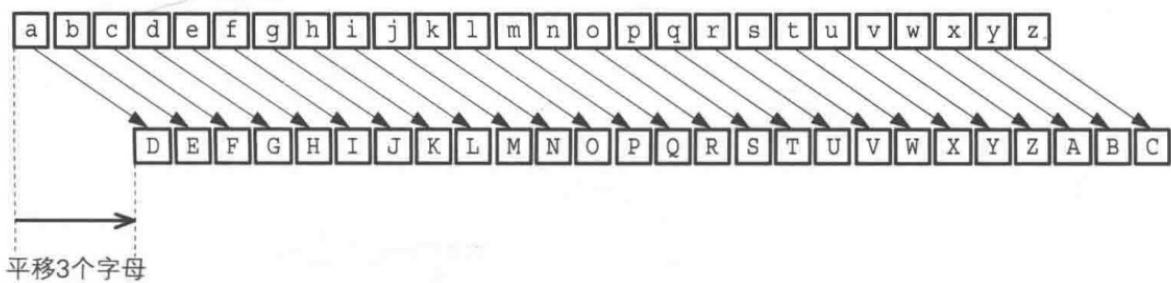
# 密码历史

## 古典密码

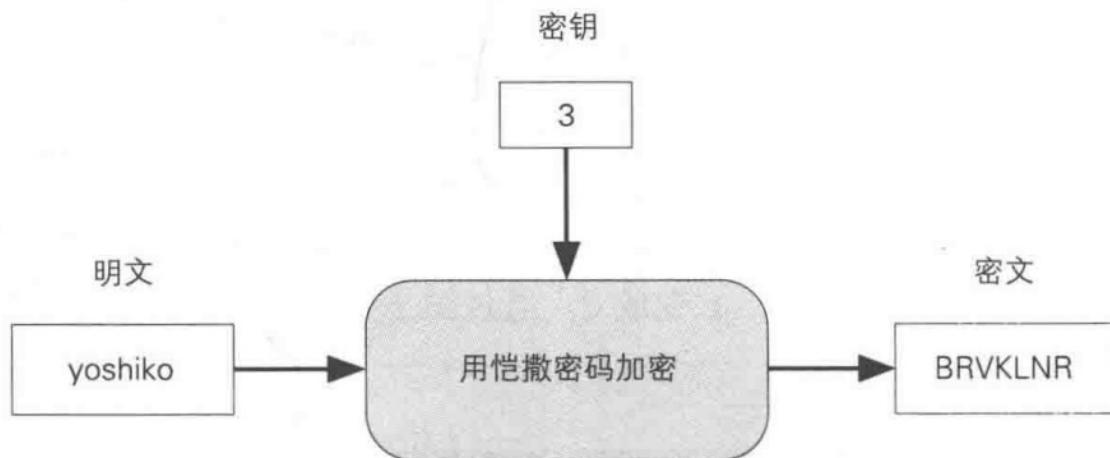
### 凯撒密码

- 诞生于公元前100年左右
- 古罗马
- 平移

### 算法



### 加密



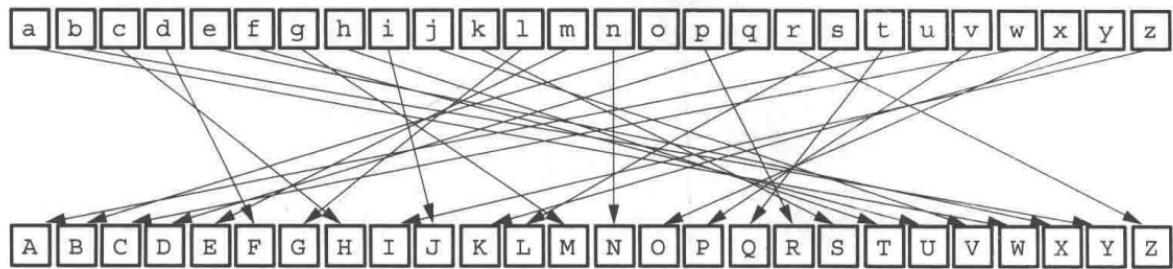
### 破解

BRVKLNR → 用密钥 0 解密 → brvklnr  
BRVKLNR → 用密钥 1 解密 → aqujkmq  
BRVKLNR → 用密钥 2 解密 → zptijlp  
**BRVKLNR → 用密钥 3 解密 → yoshiko**  
BRVKLNR → 用密钥 4 解密 → xnrg hjn  
BRVKLNR → 用密钥 5 解密 → wmqfgim  
BRVKLNR → 用密钥 6 解密 → vlpefhl  
BRVKLNR → 用密钥 7 解密 → ukodegk  
BRVKLNR → 用密钥 8 解密 → t jncdfj  
BRVKLNR → 用密钥 9 解密 → simbcei  
BRVKLNR → 用密钥 10 解密 → rhlabdh  
BRVKLNR → 用密钥 11 解密 → qgkzacg  
BRVKLNR → 用密钥 12 解密 → pfjyzbf

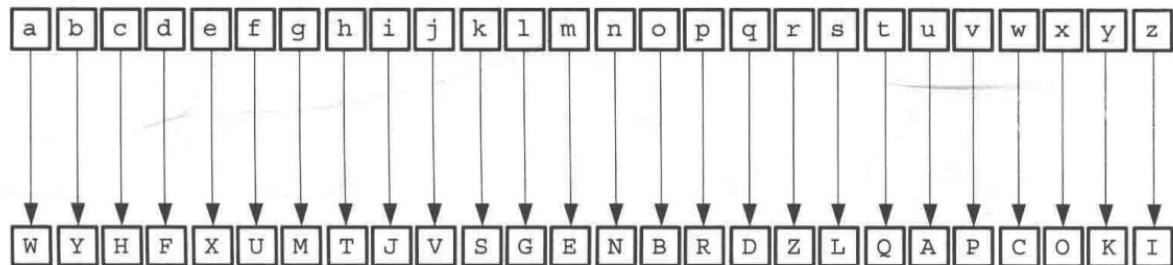
简单替换密码

算法

凯撒密码的升级版，将平移替换为无序对应。



将对应关系表现为更易读的形式（内容相同）



### 破解

- 暴力破解：每秒计算10亿次，需要计算60亿年。 $\$ \$ 26! \approx 4.03 * 10^{26} \$ \$$
- 频率分析：统计每个字母出现的频率，和英文中词语使用频率对应，就能破解。

### 弱点

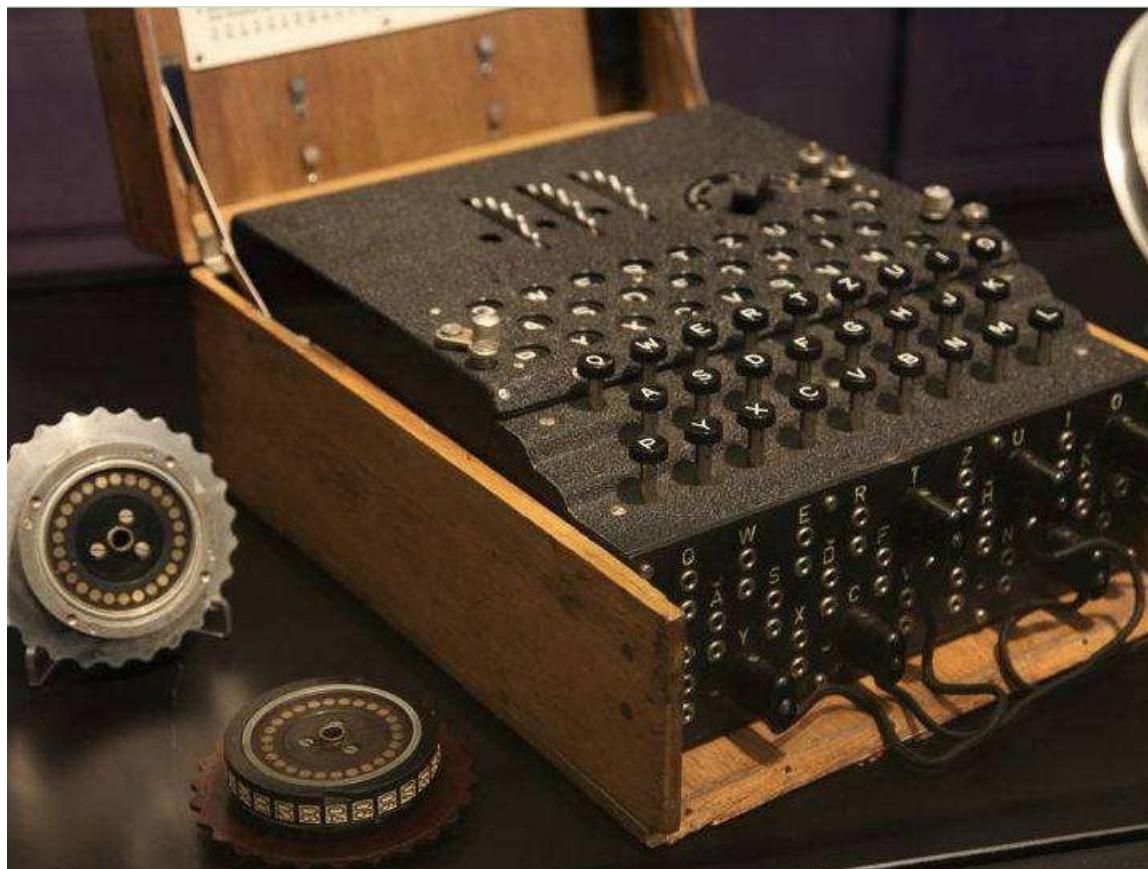
- 相同明文加密后为相同密文

## 近现代密码

### 时间

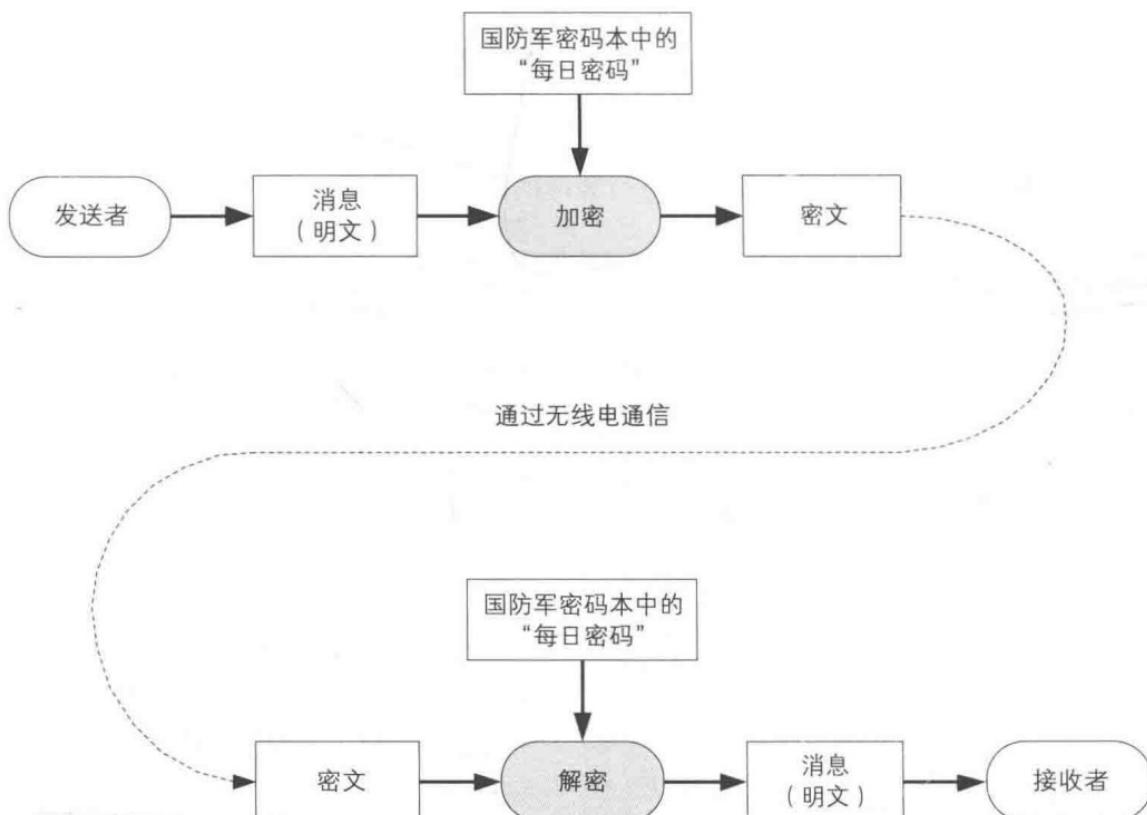
- 第一次世界大战、第二次世界大战到1976

### Enigma



二战中，德国用来发送消息的密码装置。

加解密



优点

同一个字母，加密后可以是不同字母。

缺点

- 密钥需要加密两次：因为信号不好，但同时也提供了破解线索
- 密码本的配送：密钥配送问题
- 同一字母加密后不会是该字母本身

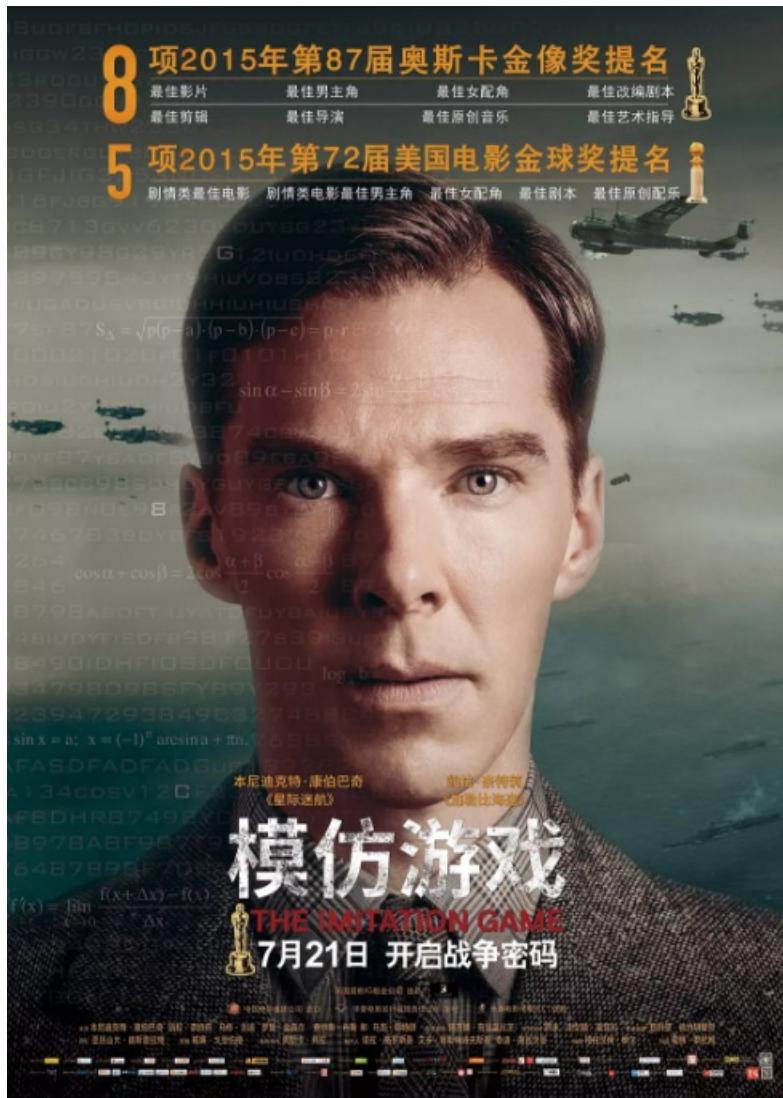
历史人物

希特勒（1889.4.20-1945.4.30）：有统一世界的强大野心





图灵（1912.6.23—1954.6.7）：enigma破译者(1940年)



丘吉尔（1874.11.30-1965.1.24）：英国命运的主宰者



现代密码

现在我们所使用的密码。

## 第二章：对称密码

用相同的密钥进行加解密。

## 异或

### 编码

任何文本在计算机上都会被编码为计算机能够识别的code，即0和1。

什么是异或？

相同为0，不同为1

0 XOR 0 = 0

(0与0的XOR结果为0)

0 XOR 1 = 1

(0与1的XOR结果为1)

1 XOR 0 = 1

(1与0的XOR结果为1)

1 XOR 1 = 0

(1与1的XOR结果为0)

### 性质

还原

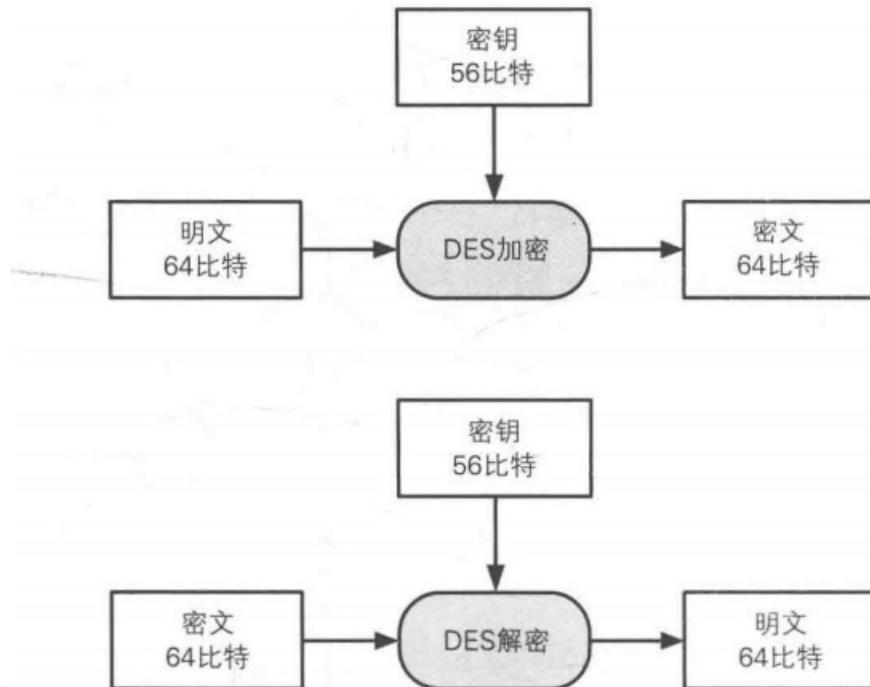
```
// 第一次 XOR  
1010 ^ 1111 // 0101  
  
// 第二次 XOR  
0101 ^ 1111 // 1010
```

# DES

什么是**DES**?

- (Data Encryption Standard) 1977年被美国采用的一种密码。
- 已经不安全。
- 1999年的公开破解中用了22小时。

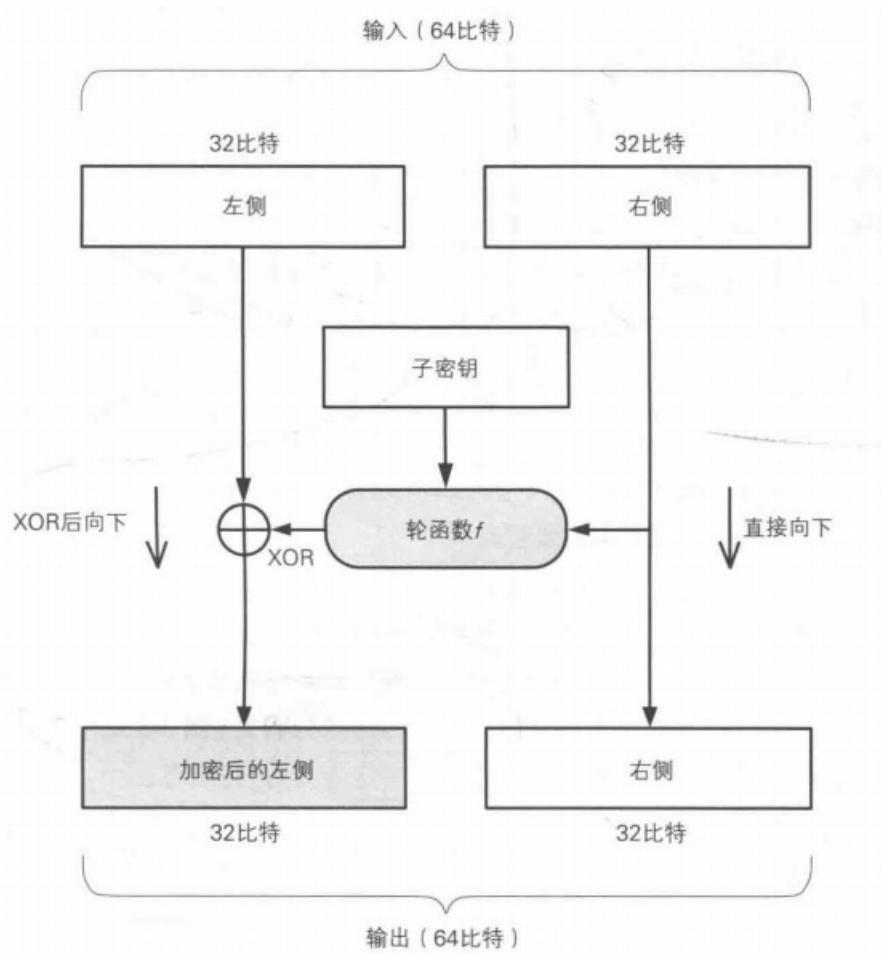
加解密



- 每次只对64比特进行加解密。
- 原文需要分组。
- 密钥每7比特有一个校验位，所以实际有56比特。

**Feistel**网络

一层加解密



疑问？

1、为什么使用轮函数 $f$ ? 而不直接XOR!

- 不使用轮函数就没有机密性可言，就能直接还原了！
- 通过原文和密文即可算出密钥。

2、为什么使用两部分处理?

- 解密需要依靠右半部分，增加了安全性。

攻击

差分分析

- 改变一部分明文，分析密文怎么改变。

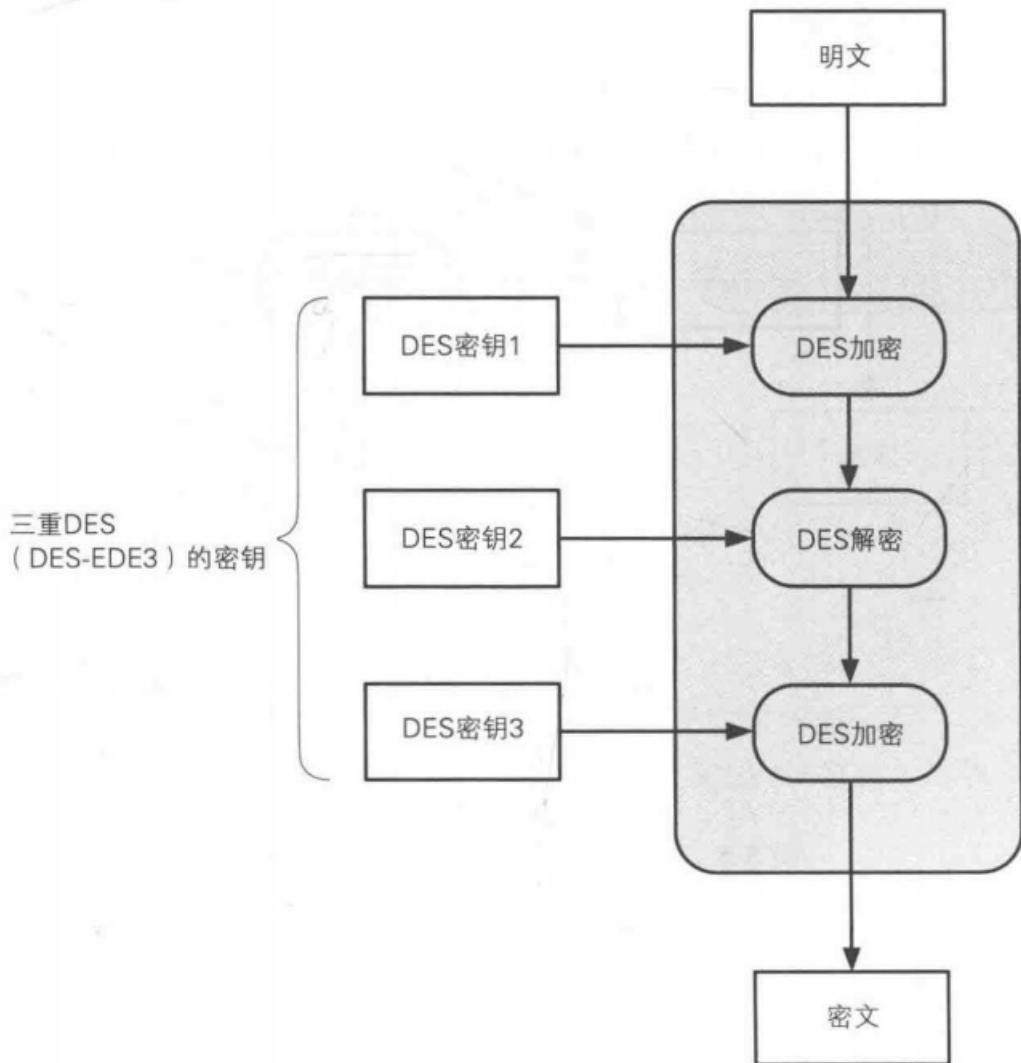
线性分析

- 找到明文和密文XOR后为0的概率，如果偏离 $1/2$ ，则会提供一些和密钥相关的线索。

## 3DES

- (Triple-DES) 执行3次DES过程的密码算法
- 由IBM设计

算法



- 密钥长度为DES的三倍（168比特）
- EDE: Encryption (加密) —> Decryption (解密) —> Encryption (加密)
- EDE为了兼容DES（使用相同密钥，执行EDE）

# AES

什么是AES?

- (Advanced Encryption Standard) 高级加密标准
- 2000年诞生
- 三年公开选拔

## Rijndael

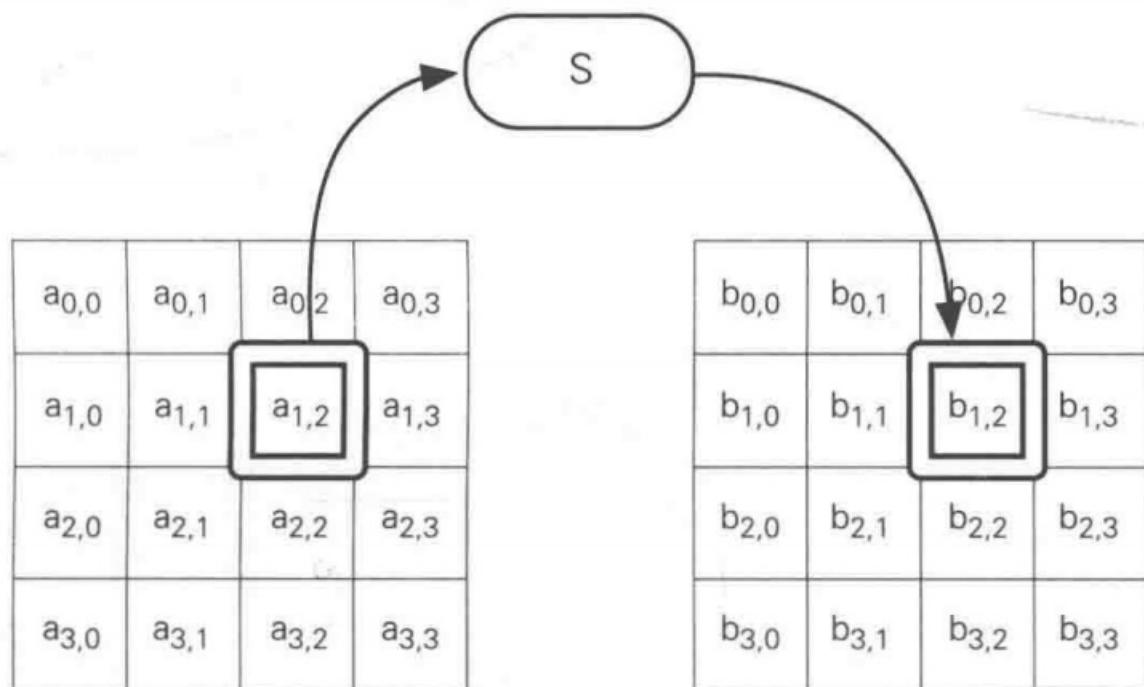
AES标准密码算法

算法过程

SubBytes

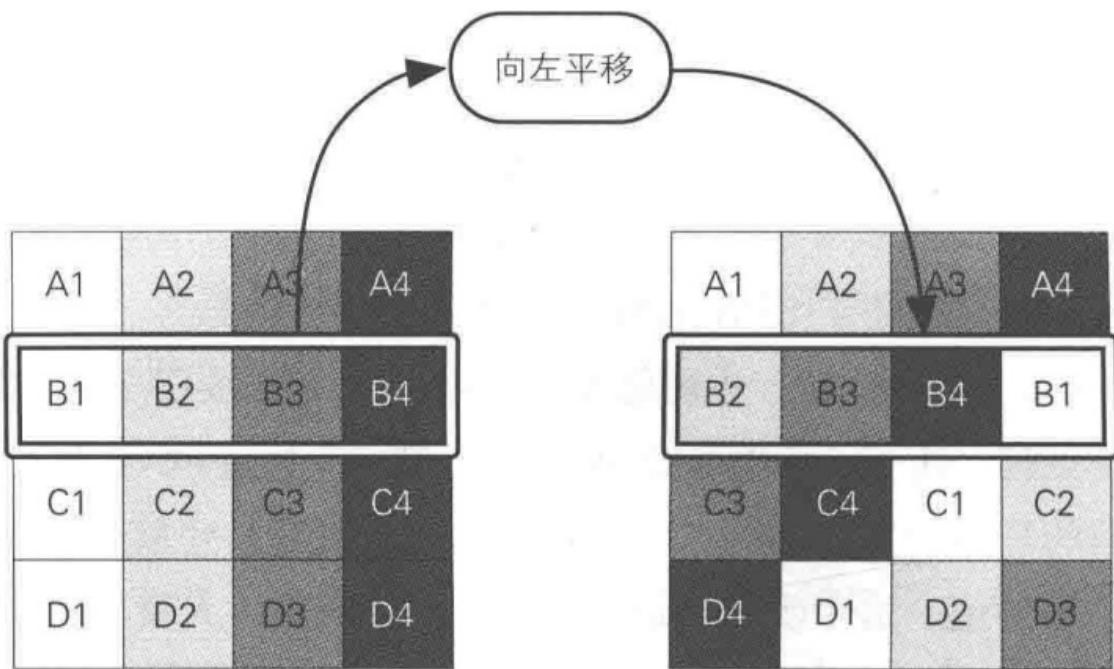
128位, 16字节, 0~255, Substitute-Box

简单密码替换

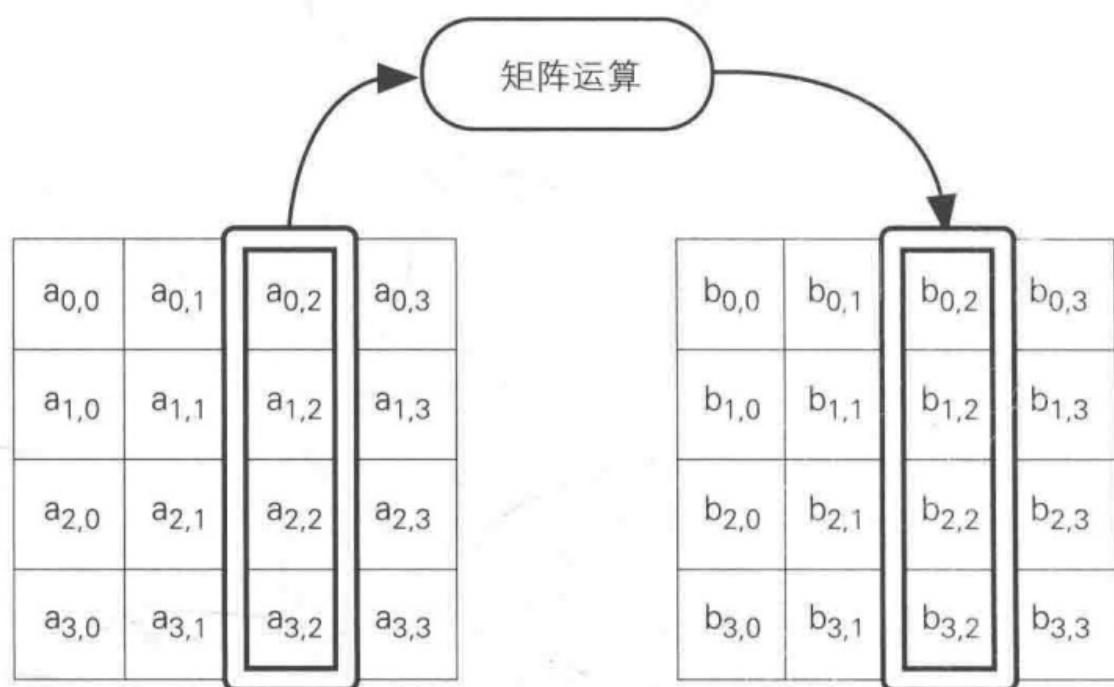


ShiftRows

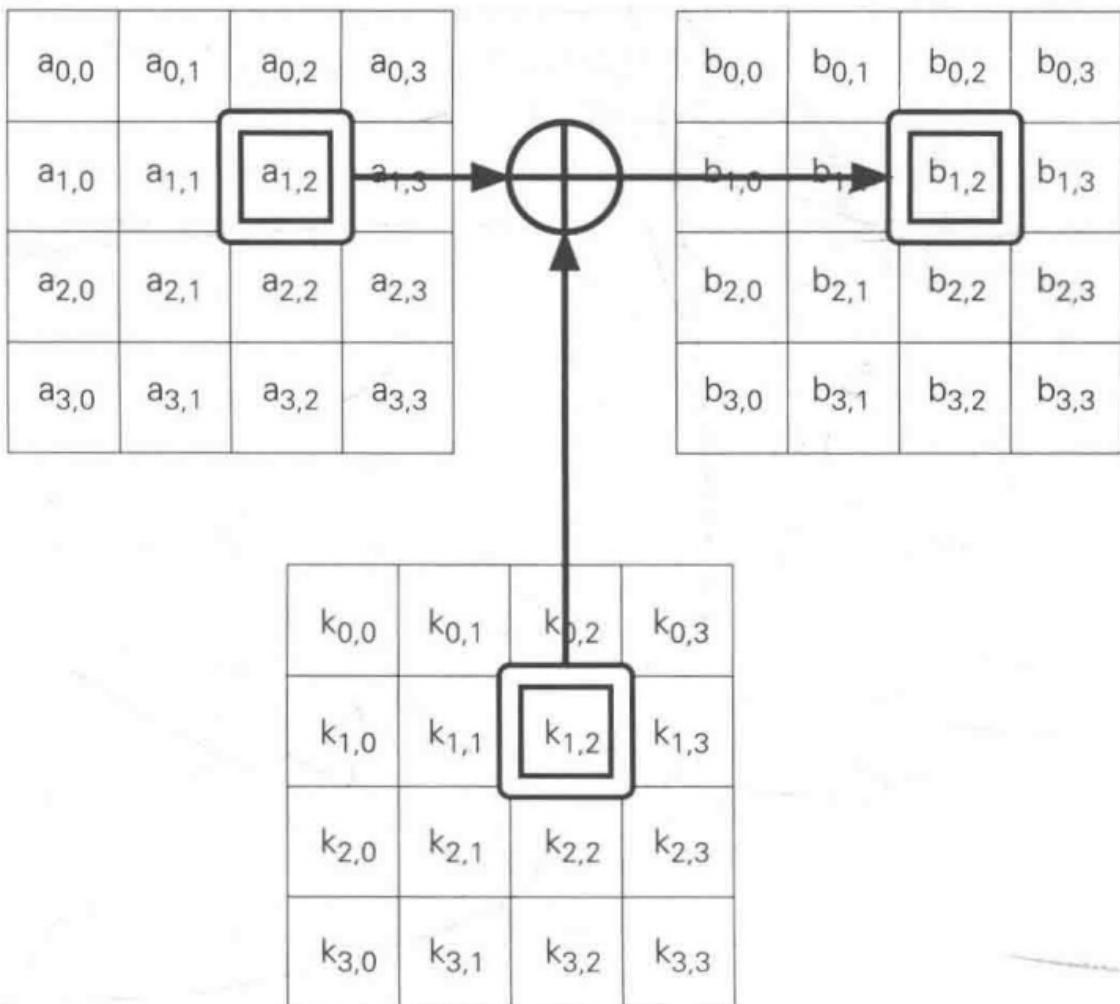
根据规则平移



MixColumns



AddRoundmKey



算法过程动态展示

[http://www.formaestudio.com/rijndaelinspector/archivos/Rijndael\\_Animation\\_v4\\_eng.swf](http://www.formaestudio.com/rijndaelinspector/archivos/Rijndael_Animation_v4_eng.swf)

优点

- 速度快
- 暂无有效攻击

疑问

能推导出密钥吗？

提示：不能，密钥进过扩展，并存在多重异或。

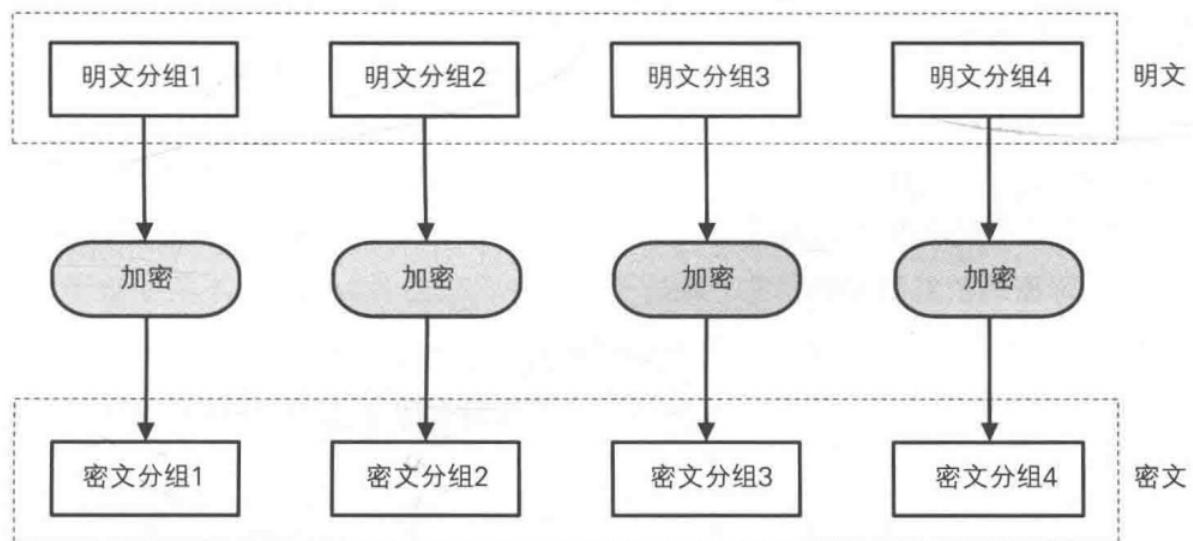
### 第三章：分组密码

- 对任意长度的密码进行加密的方式，就是分组加密
- 分组的方法，就是分组模式

## ECB

- (Electronic Codebook) 电子密码本
- 把明文分组后直接加密成密码的方式

加解密



弱点

相同分组加密后，密文一样。

攻击

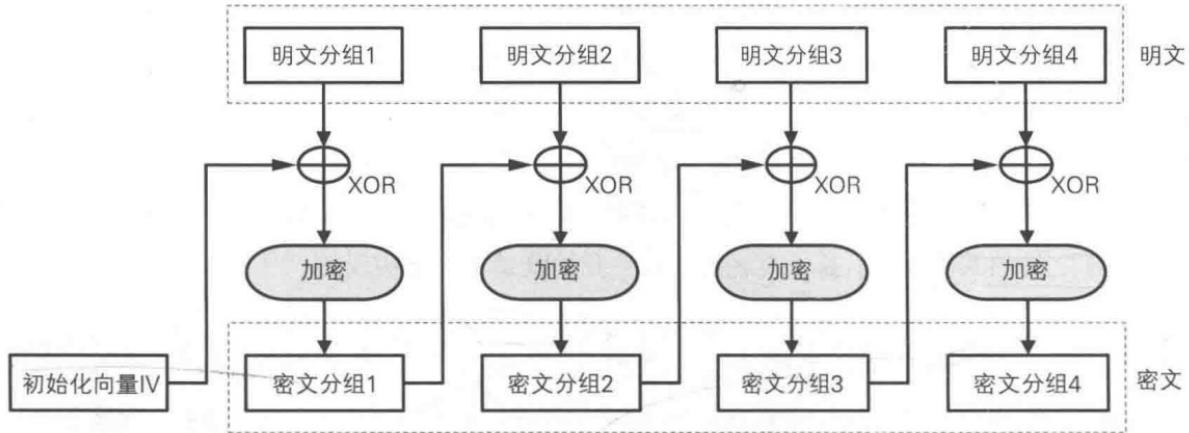
- 交换分组：转账
- 口令覆盖：登录

## CBC

- (Cipher Block Chaining) 密文分组链接模式
- 明文分组与前一个密文分组XOR后再加密

算法

CBC模式的加密



## IV

- 随机生成的比特序列

特点

- 一个密文分组损坏，则只影响两个明文分组

攻击

比特攻击

- 对初始化向量一个比特的修改，会造成明文分组1的一个比特的改变
- 密文分组1的一个比特修改，会造成明文分组1的多个比特发生改变

填充提示攻击

方法

- 对最后一个分组的填充，返回错误信息，获取与明文相关信息

例子

- 2014年 SSL3.0 POODLE攻击

预防

- 确保明文由合法发送者，知道明文的情况下发送

## IV攻击

例子

- TLS1.0 IV使用上一次CBC加密时的最后一个分组

预防

- TLS1.1 为显示传送

## 疑问

为什么**XOR**不在加密之后？

提示： 和**ECB**等价

## CFB

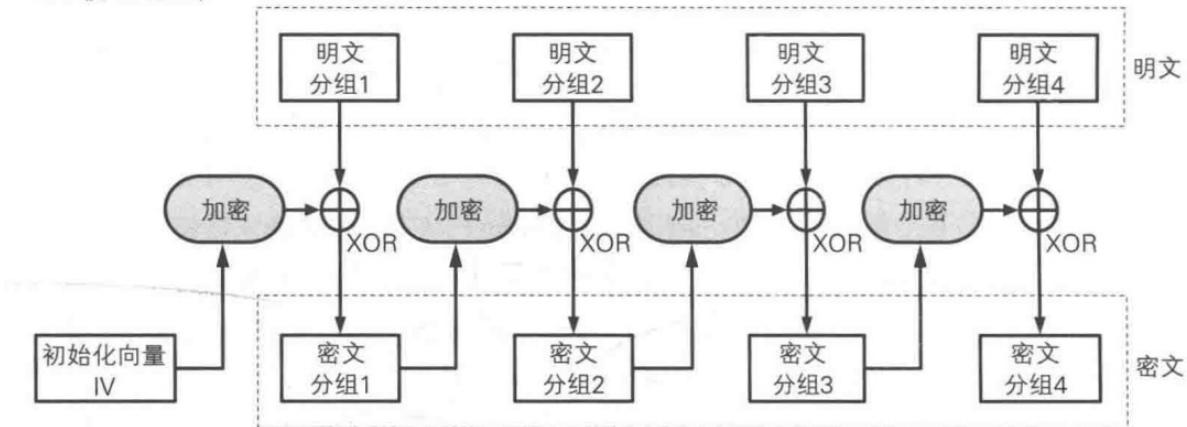
- (Cipher Feedback) 密文反馈模式
- 前一个密文作为加密输入

算法

加密

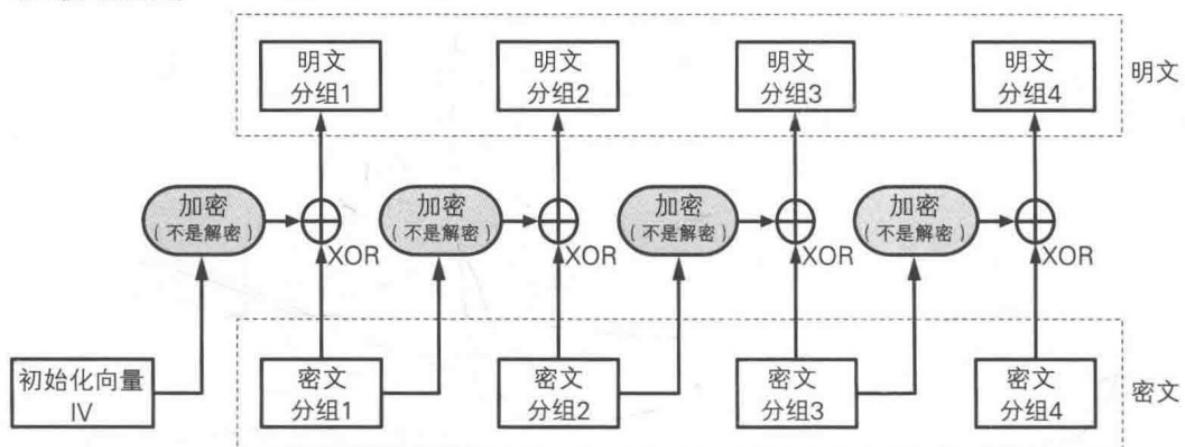
明文和上一次输出密文进行加密后，再XOR

CFB模式的加密



解密

CFB模式的解密



疑问

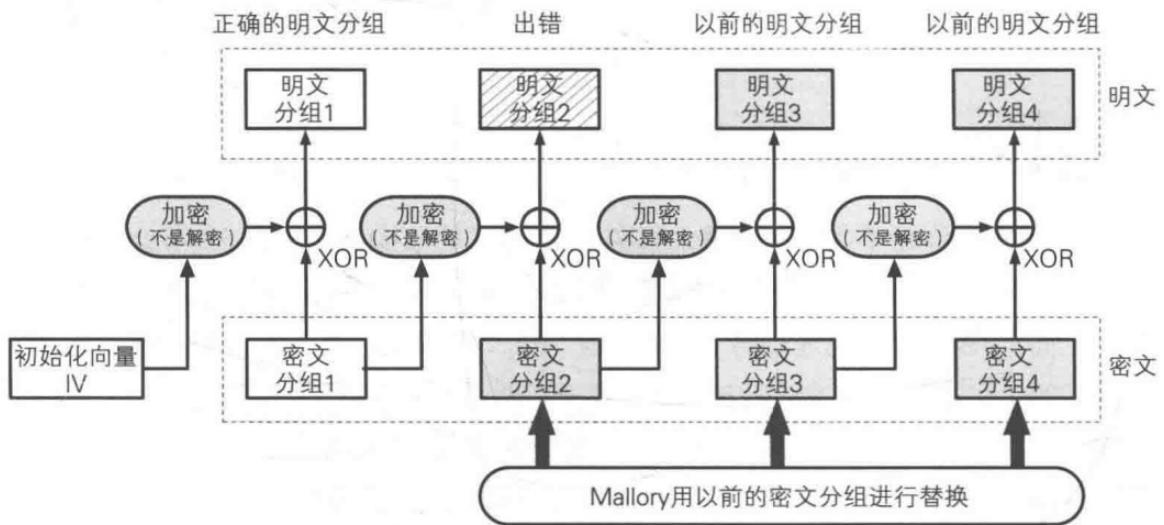
为什么解密使用的是加密？

- 保证能算出明文
- 只使用加密，增强了密钥的安全，可以使用不可逆函数

攻击

重放攻击

- 用以前的数据再次发送，还能生效



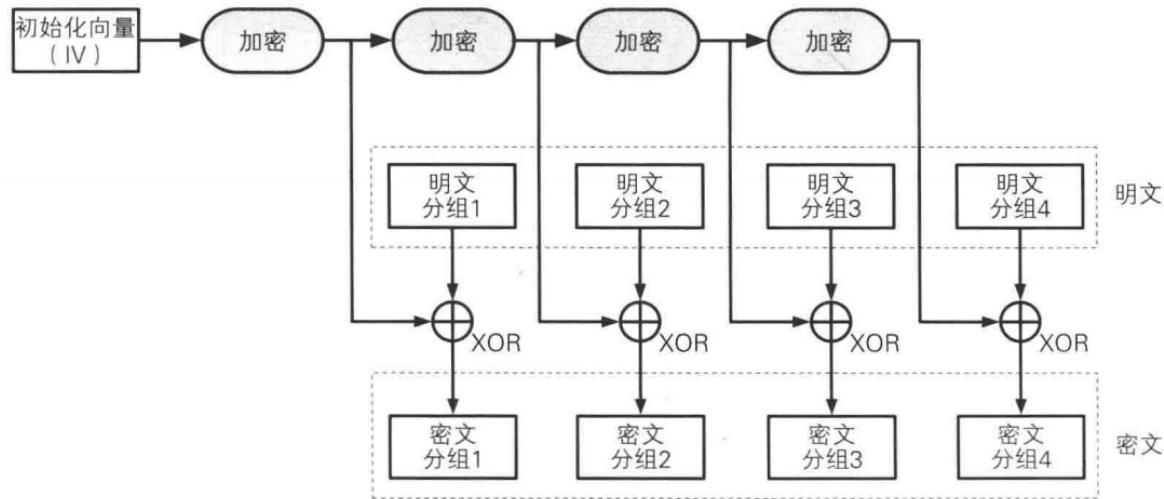
## OFB

- (Output Feedback) 输出反馈模式
- 明文和上一次加密密码进行XOR

算法

加解密

加密解密使用相同加密算法



特点

- IV作用到每一个分组，所以无法实施重放攻击

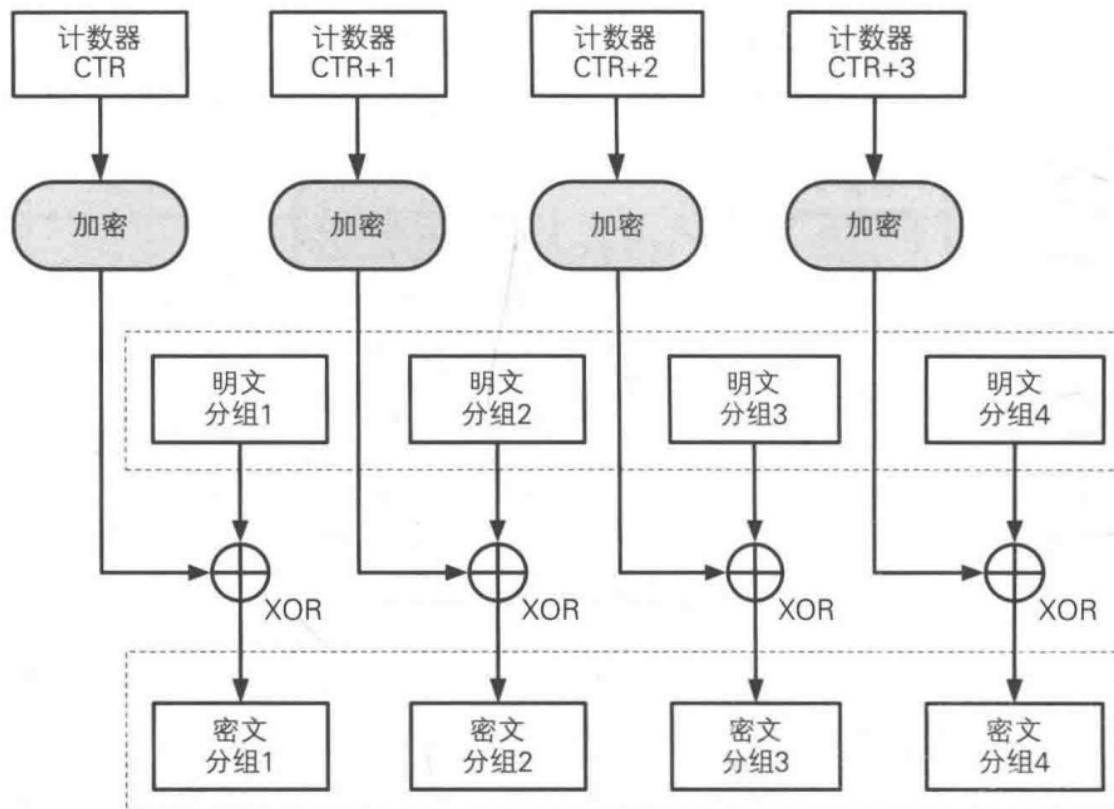
## CTR

- (CounTeR) 计数器模式
- 初次累加的计数器进行加密后，和对应明文分组XOR

算法

加解密

加密和解密都是用加密算法



CTR生成规则

66 1F 98 CD 37 A3 8B 4B 00 00 00 00 00 00 00 01  
\_\_\_\_\_ nonce \_\_\_\_\_ 分组序号 \_\_\_\_\_

特点

- 可并行计算
- 速度快

攻击

比特攻击

- 和OFB一样，密文一个比特的反转，就会造成明文一个比特的变化。



## 第四章：公钥密码

- 1978年诞生
- 非对称密码：使用不同密钥进行加解密
- 用户拥有加密密钥和解密密钥
- 加密密钥是公开的

# 密钥配送

## 密钥配送问题

- 对称密码中，加解密使用的是同一密钥，因此，加密者必须向解密者发送密钥
- 可能会被窃听
- 怎么向使用同一密钥的用户安全发送密钥，就是密钥配送问题

## 怎么解决

事先共享

方法

- 在加密前，先把密钥给对方

安全性

- 但是怎么安全的给？同桌直接亲手交付

局限

- 每个人都需要一个，密钥数量会极度膨胀，不实用

密钥分配中心

方法

- 由机器来维护

安全性

- 所有密钥集中，容易导致攻击

局限

- 增加机器负载
- 容易瘫痪

迪菲－赫尔曼（**Diffie-Hellman**）密钥交换

方法

- 发送者和接受者之间先共享一些信息

这些信息被窃听者获取也没事

安全性

- 密码学史上最伟大的发明
- 安全性很高

公钥密码

方法

- 事先生成加密和解密密钥，加密密钥发给对方

安全性

- 加密密钥是公开的，可以给任何人
- 只有解密密钥才能解密

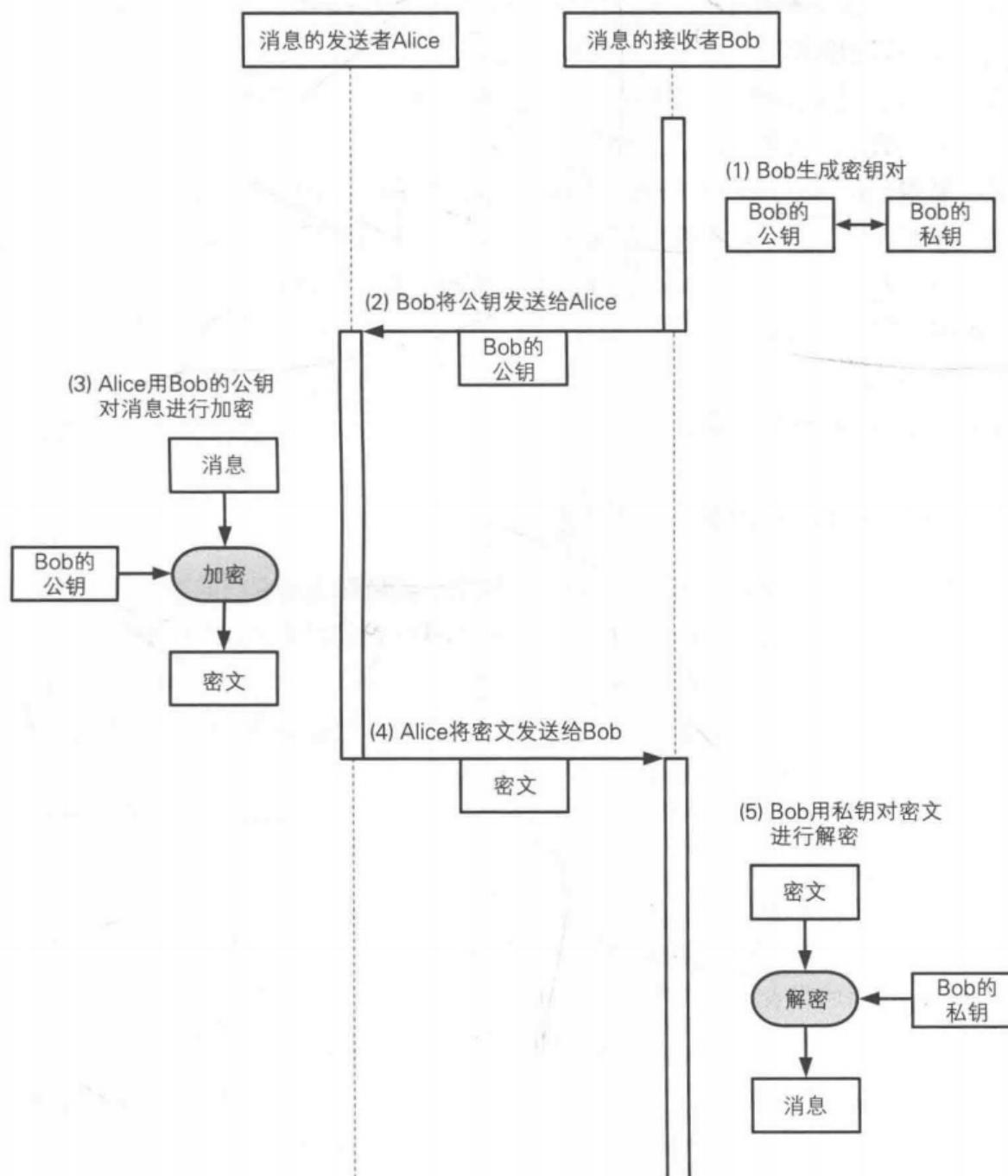


# 流程

## 特点

- 发送者只需要加密密钥
- 接受者只需要解密密钥
- 解密密钥不可以被窃听
- 加密密钥被窃听也没事

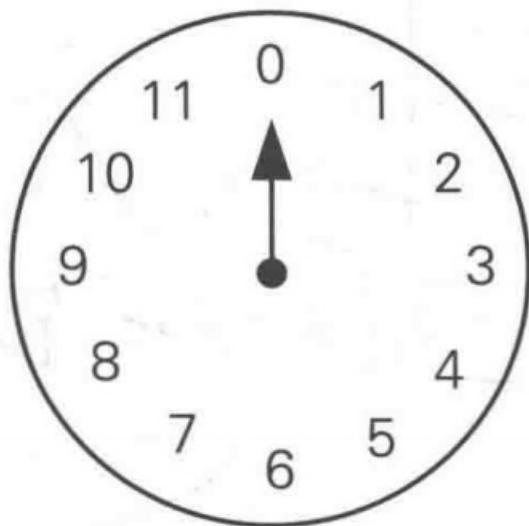
## 加解密流程



## 疑问

获取的公钥是否合法？

## 时钟运算



- 在只有一个指针的时钟上运行的运算

### 基本运算

加法

- $12 + 3 = ?$
- $12 + 3 = 3$

其实是取模 (**mod**) 运算

- $12 + 3 \equiv 3 \pmod{12}$
- $(12 + 3) \bmod 12 = 3$

减法

与加法类似

- $18 - 3 \equiv 3 \pmod{12}$

乘法

作累加运算

- $7 + 7 + 7 + 7 \equiv 4 \pmod{12}$

除法

乘法的逆运算

- $7 * 7 \equiv 1 \pmod{12}$
- $1 \pmod{12} \div ? = 7$

乘方

累乘运算

$$\$ \$ 7^{4} \equiv 1 \pmod{12}$$

\$\$

引申出来的理论

- 什么情况下，两个数的乘积与1在模n时同余？

$\$ \$ a * b \equiv 1 \pmod{n}$

$\$ \$$

例子

- 1  $\times$  ■ mod 12 = 1  $\rightarrow$  ■ = 1 … 1 和 12 的最大公约数是 1
- 2  $\times$  ■ mod 12 = 1  $\rightarrow$  ■不存在 … 2 和 12 的最大公约数不是 1
- 3  $\times$  ■ mod 12 = 1  $\rightarrow$  ■不存在 … 3 和 12 的最大公约数不是 1
- 4  $\times$  ■ mod 12 = 1  $\rightarrow$  ■不存在 … 4 和 12 的最大公约数不是 1
- 5  $\times$  ■ mod 12 = 1  $\rightarrow$  ■ = 5 … 5 和 12 的最大公约数是 1
- 6  $\times$  ■ mod 12 = 1  $\rightarrow$  ■不存在 … 6 和 12 的最大公约数不是 1
- 7  $\times$  ■ mod 12 = 1  $\rightarrow$  ■ = 7 … 7 和 12 的最大公约数是 1
- 8  $\times$  ■ mod 12 = 1  $\rightarrow$  ■不存在 … 8 和 12 的最大公约数不是 1
- 9  $\times$  ■ mod 12 = 1  $\rightarrow$  ■不存在 … 9 和 12 的最大公约数不是 1
- 10  $\times$  ■ mod 12 = 1  $\rightarrow$  ■不存在 … 10 和 12 的最大公约数不是 1
- 11  $\times$  ■ mod 12 = 1  $\rightarrow$  ■ = 11 … 11 和 12 的最大公约数是 1

最大公约数

- 指两个或多个整数共有约数中最大的一个

质数

- 在大于1的自然数中，除了1和它本身以外不再有其他因数

互质

- 公约数只有1的两个整数

离散对数

乘方的逆运算

$\$ \$ 7 ^{?} \equiv 1 \pmod{12}$

$\$ \$$

性质

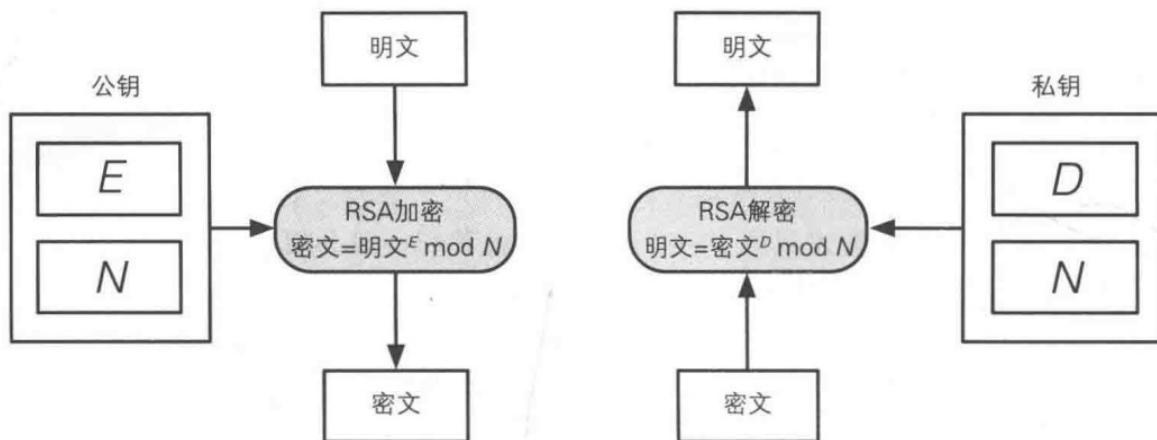
- 求解对数容易，但是求解离散对数非常困难，快速求解算法目前还没有发现

## 加解密过程

- 乘方

过程

加解密



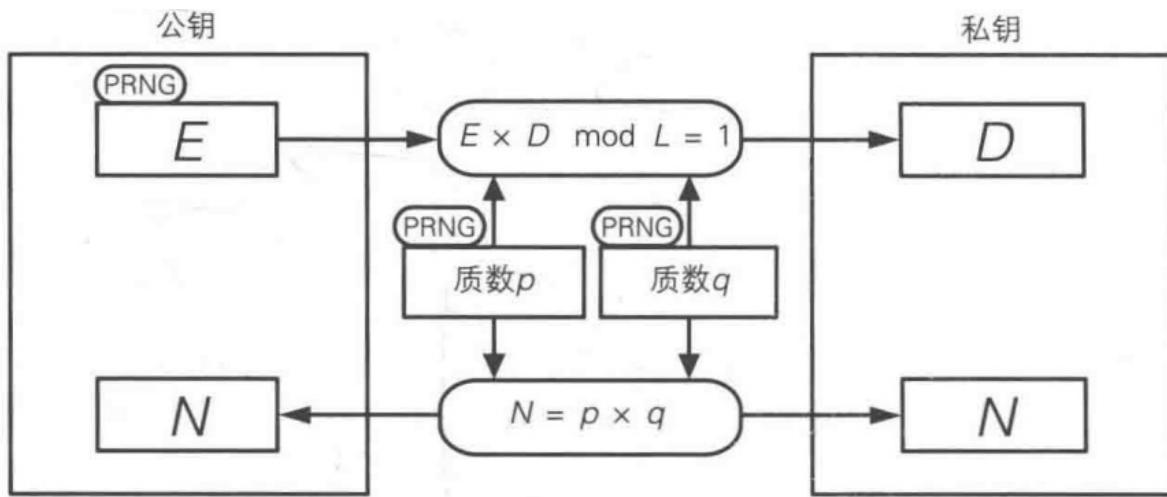
密钥对	公钥	数 $E$ 和数 $N$
	私钥	数 $D$ 和数 $N$
加密		密文 = 明文 $^E \text{ mod } N$ ( 明文的 $E$ 次方除以 $N$ 的余数 )
解密		明文 = 密文 $^D \text{ mod } N$ ( 密文的 $D$ 次方除以 $N$ 的余数 )

求解过程

- 求  $N$
- 求  $L$  (过程中使用的数)
- 求  $E$
- 求  $D$

(1) 求 $N$	(3) 求 $E$
用伪随机数生成器求 $p$ 和 $q$ , $p$ 和 $q$ 都是质数 $N = p \times q$	$1 < E < L$ $\gcd(E, L) = 1$ ; $E$ 和 $L$ 的最大公约数为 1 ( $E$ 和 $L$ 互质)
(2) 求 $L$	(4) 求 $D$
$L = \text{lcm}(p - 1, q - 1)$ ; $L$ 是 $p - 1$ 和 $q - 1$ 的最小公倍数	$1 < D < L$ $E \times D \text{ mod } L = 1$

对应关系



(PRNG) = 伪随机数生成器

$$L = \text{lcm}(p-1, q-1)$$

$$\gcd(E, L) = 1$$

$$1 < E < L$$

$$1 < D < L$$

## 实例

求N

$$p = 17, q = 19 \text{ (两个质数)}$$

$$N = p \times q \text{ (求积)}$$

$$= 17 \times 19$$

$$= 323$$

求L

$$L = \text{lcm}(p-1, q-1) \text{ (求} p-1, q-1 \text{的最小公倍数)}$$

$$= \text{lcm}(16, 18)$$

$$= 144$$

求E

1 ~ 144 的质数较多, 有:

5、7、11、13、17、19、23、25、29、31 ...

取E = 5

求D

$$E \times D \equiv 1 \pmod{L}$$

$$5 \times D \equiv 1 \pmod{144}$$

$$D = 29$$

结果

公钥

$$E = 5, N = 323$$

私钥

D = 29, N = 323

加密

\$\$ 明文^{\{E\}} \bmod N = 123^{\{5\}} \bmod 323 = 225

\$\$

解密

\$\$ 密文^{\{D\}} \bmod N = 225^{\{29\}} \bmod 323 = 123 \$\$

# 攻击

通过密文找明文

已知密文、**E**、**N**, 求明文:

\$\$ 明文^{\{E\}} \bmod N = ?^{\{5\}} \bmod 323 = 225

\$\$

- 目前没有找到高效算法

暴力破解私钥**D**

\$\$ 密文^{\{D\}} \bmod N = 225^{\{? \}} \bmod 323 = 123

\$\$

- **D**的长度约等于**N**的长度, 当**D**为2048位时, 暴力破解几乎不现实

通过**E**、**N**求解**D**

由公式:

\$\$ E \* D \equiv 1 \pmod{L}

\$\$ 可知:

要求解**D**, 必须知道**L**, 而**L**由公式:

\$\$ L = \text{lcm}(p-1, q-1)

\$\$ 可知:

要知道**L**, 必须求解**p**、**q**

而又由公式:

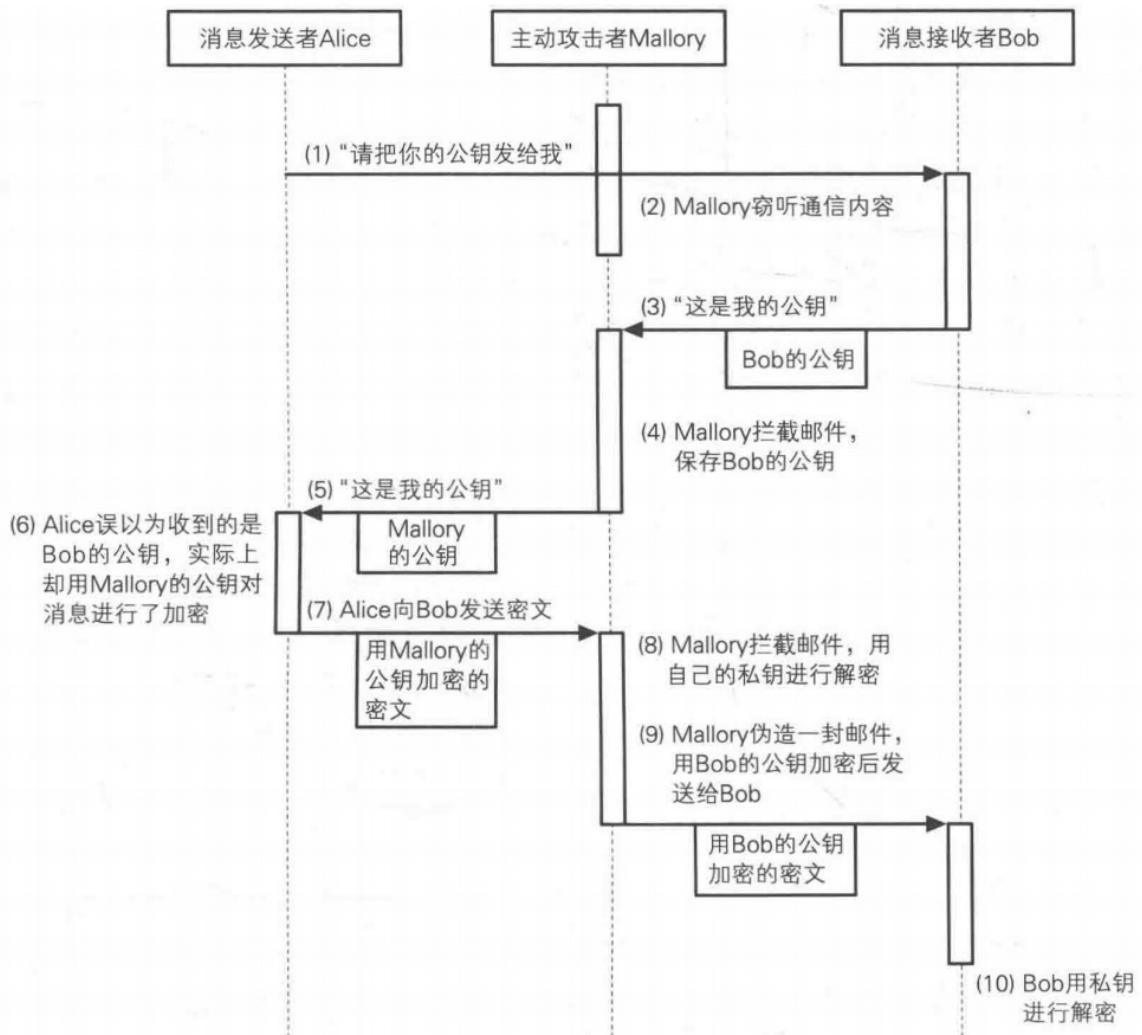
\$\$ p \* q = N

\$\$ 可知:

**p**、**q**可以**N**质数分解得出, 但是:

- 目前还没有高效的质因数分解算法

中间人攻击



### 选择明文攻击

- 通过解密信息，获取明文信息

# 证明

## 证明过程

### step1:

根据加密规则公式：

$$\$ \$ m^{\{e\}} \equiv c \pmod{n}$$

得出：

$$\$ \$ \text{Large } c = m^{e-kn}$$

\$\$

### step2:

将上述的c代入解密公式：

$$\$ \$ c^d \equiv m \pmod{n}$$

得出：

$$\$ \$ \text{Large } (m^{e-kn})^d \equiv m \pmod{n}$$

\$\$

### step3:

将上述方程式的左侧用二项式定理展开：

$$\$ \$ (m^{e-kn})^d = C_0 m^{\{ed\}} (-kn)^0 + C_1 m^{\{d-1\}} (-kn)^1 + C_2 m^{\{d-2\}} (-kn)^2 + \dots + C_d m^{\{0\}} (-kn)^d$$

\$\$

可以发现，由于第二项开始都是n的km倍数，所以step2的证明可以简化为

$$\$ \$ \text{Large } m^{\{ed\}} \equiv m \pmod{n}$$

\$\$

### step4:

由于：

$$\$ \$ ed \equiv 1 \pmod{\phi(n)}$$

所以：

$$\$ \$ \text{Large } ed = h\phi(n) + 1$$

\$\$

### step5:

上述结论代入step3的公式，得到

$$\$ \$ \text{Large } m^{\{h\phi(n)+1\}} \equiv m \pmod{n}$$

\$\$

### step6:

上述公式的证明需要分为两种情况。

(1)  $m$ 与 $n$ 互质

满足欧拉定理：

$$\$ \$ m^{\{\phi(n)\}} \equiv 1 \pmod{n}$$

\$\$

可得：

\$\$ \varphi(n) = kn + 1

\$\$

由二项式展开定理，可以得出下述结论：

\$\$ \text{Large}\color{blue}{(m^{\varphi(n)})^h \times m \equiv m \pmod{n})}

\$\$ step5 的公式得到证明。

(2)  $m$ 与 $n$ 不是互质关系

由大家证明吧！

## 疑问

为什么需要互质的 $p$ 、 $q$ ？

提示：互质的 $p$ 、 $q$ 是欧拉函数（与 $N$ 互质，并且小于 $N$ 的正整数的数量）成立的必要条件

\$\$ L = \text{lcm}(p-1, q-1) = \varphi(n)

\$\$

- $L$ 即是 $N$ 的欧拉值

为什么要找一个数是和 $p$ 、 $q$ 减一后的乘积（ $L$ ）成互质的数为公钥（ $e$ ）？

提示：互质满足欧拉定理

\$\$ e^{\varphi(L)} \equiv 1 \pmod{L}

\$\$ 则：

\$\$ e^{\varphi(L)-1} \cdot e \equiv 1 \pmod{L}

\$\$ 得出：

\$\$ d = \varphi(L) - 1

\$\$

- 一定能找到用于解密的密钥 $d$

## 结论

- 互质的 $p$ 、 $q$ 是整个过程能顺利执行的保证

## 第五章：混合密码

- 把各种密码技术组合起来使用的方式叫做混合密码

## 概述

对称密码

优点

- 速度快

缺点

- 存在密钥配送问题

公钥密码

优点

- 解决密钥配送问题

缺点

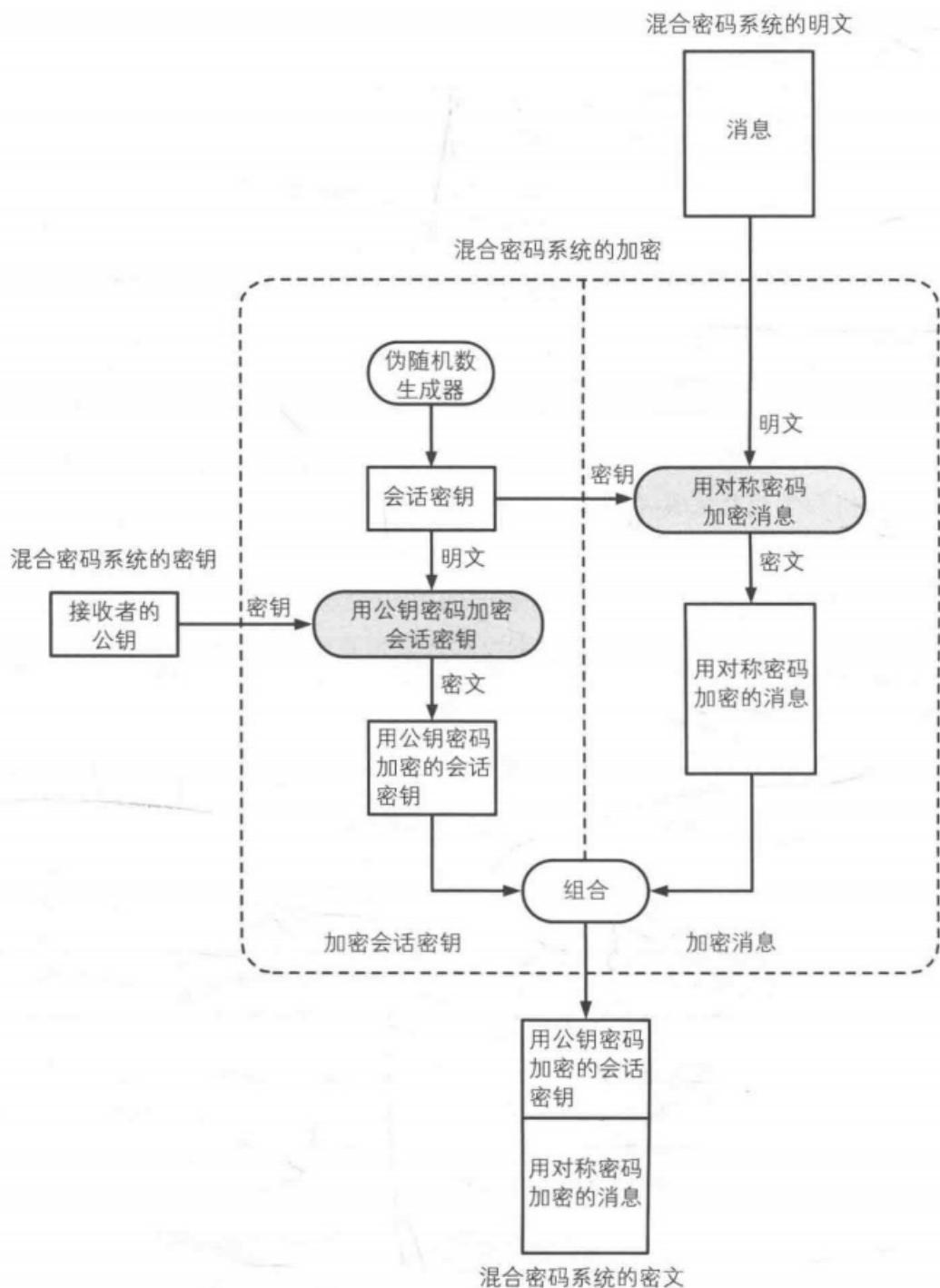
- 速度慢

# 加密

## 重点

- PRNG生成对称密钥
- 公钥加密对称密钥
- 用对称密钥加密消息

## 流程



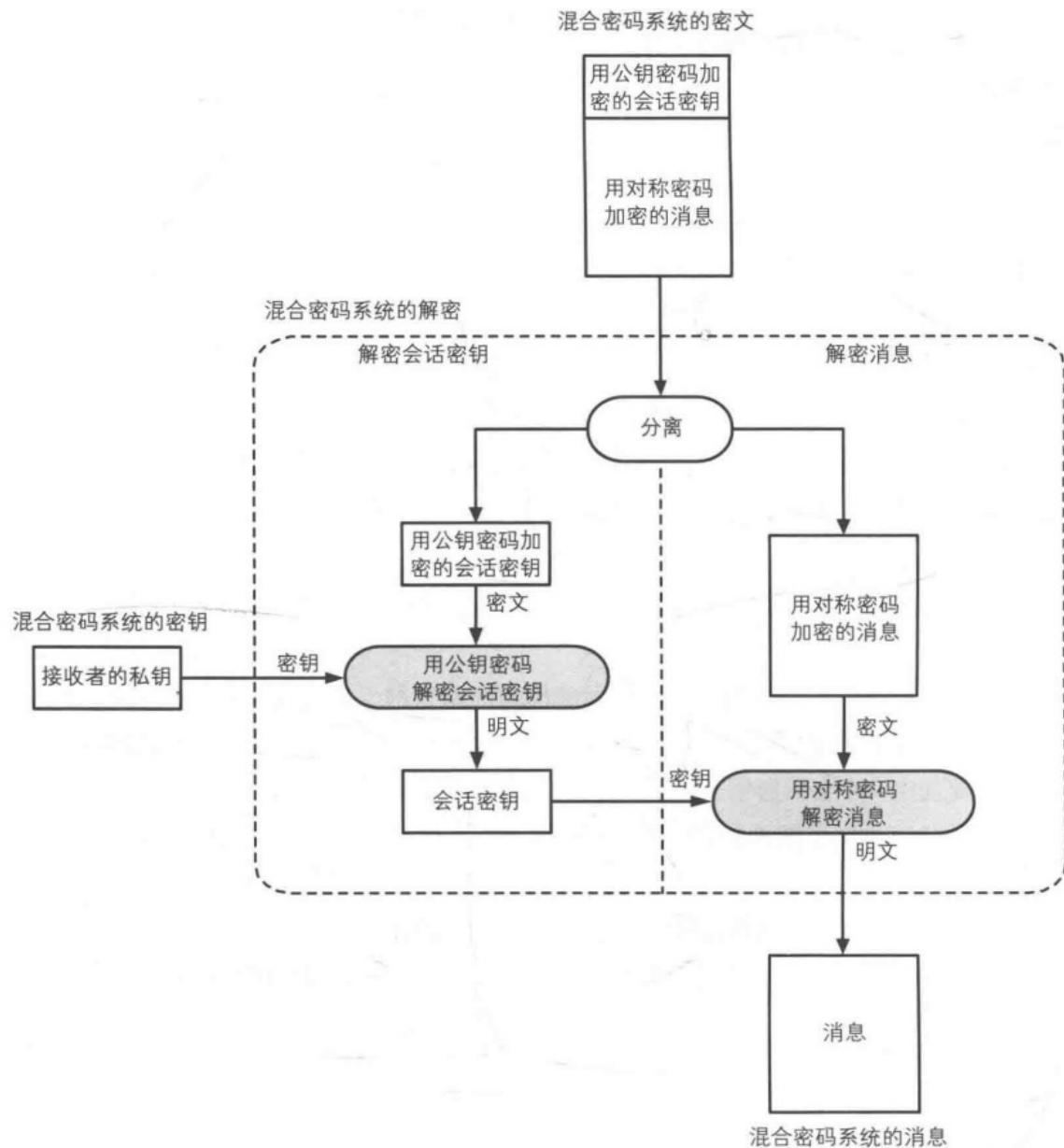


# 解密

## 重点

- 获取加密的对称密钥
- 私钥解密出对称密钥
- 用对称密钥解密出消息

## 流程



## 第六章：单项散列函数

- 给文件计算出一个唯一值
- 校验文件完整性的算法

## 哈希应用

存放不重复**key**数据

- `HashMap`
- `HashTable`
- `ConcurrentHashMap`
- `HashSet`

## Hbase

- 布隆过滤器：快速定位一条数据是否在一个文件中

## MemCache、Redis

- 一致性哈希环：增删节点时，减轻对已分布数据的影响

## ES

- 路由：让写入数据均匀分布

## 局部敏感哈希

- 大数据中，文本相似性判断

## 文件是否完全一致

- 迅雷（下载完整性校验）
- marmot（文件是否需要同步）
- 软件（是否篡改）

# 分类

## Message Digest

### MD4

- 1990年设计
- 128位
- 不安全

### MD5

- 1991年设计
- 128位
- 不安全（2004年攻破，王小川）

## Secure Hash Algorithm

### SHA-1

- 1993年诞生
- 160位
- 不安全（2005年攻破，王小川）

### SHA-2

- 2002年诞生
- SHA-256
- SHA-384
- SHA-512
- 安全

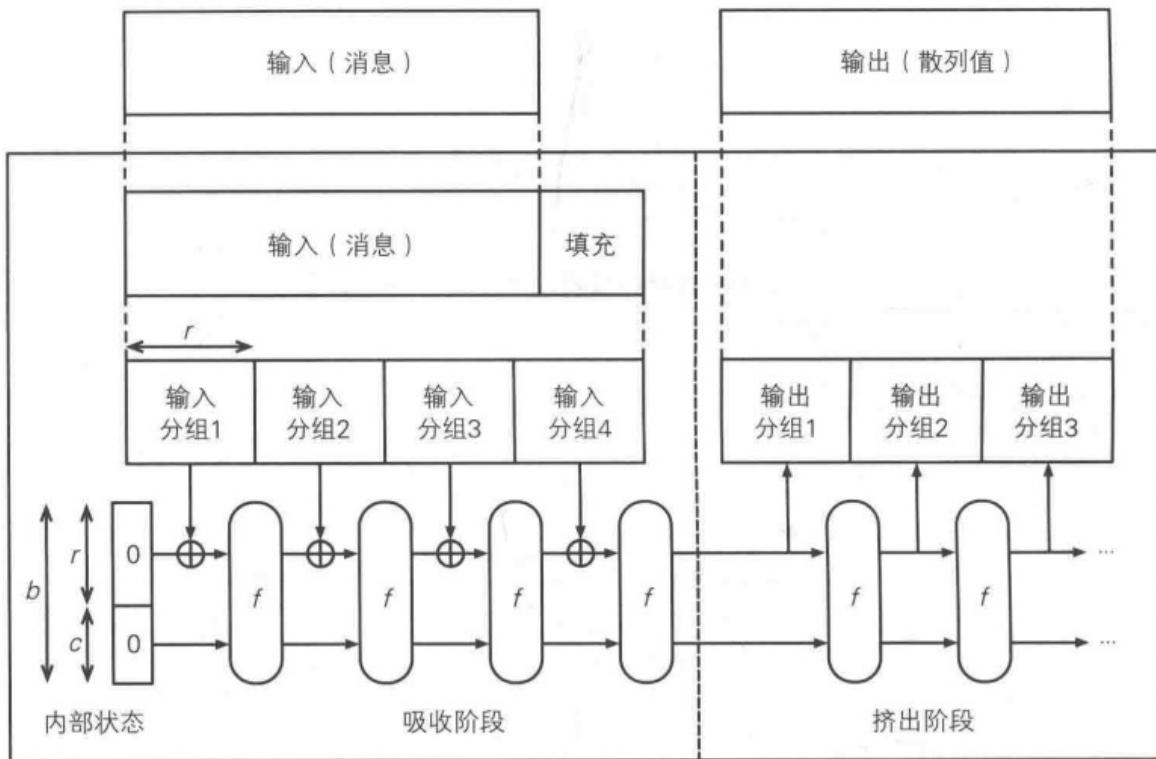
### SHA-3

- 2007年开始公开征集
- 2012年诞生
- 经过5年选拔
- Keccak脱颖而出
- 其中Joan Daemen也是AES的设计者

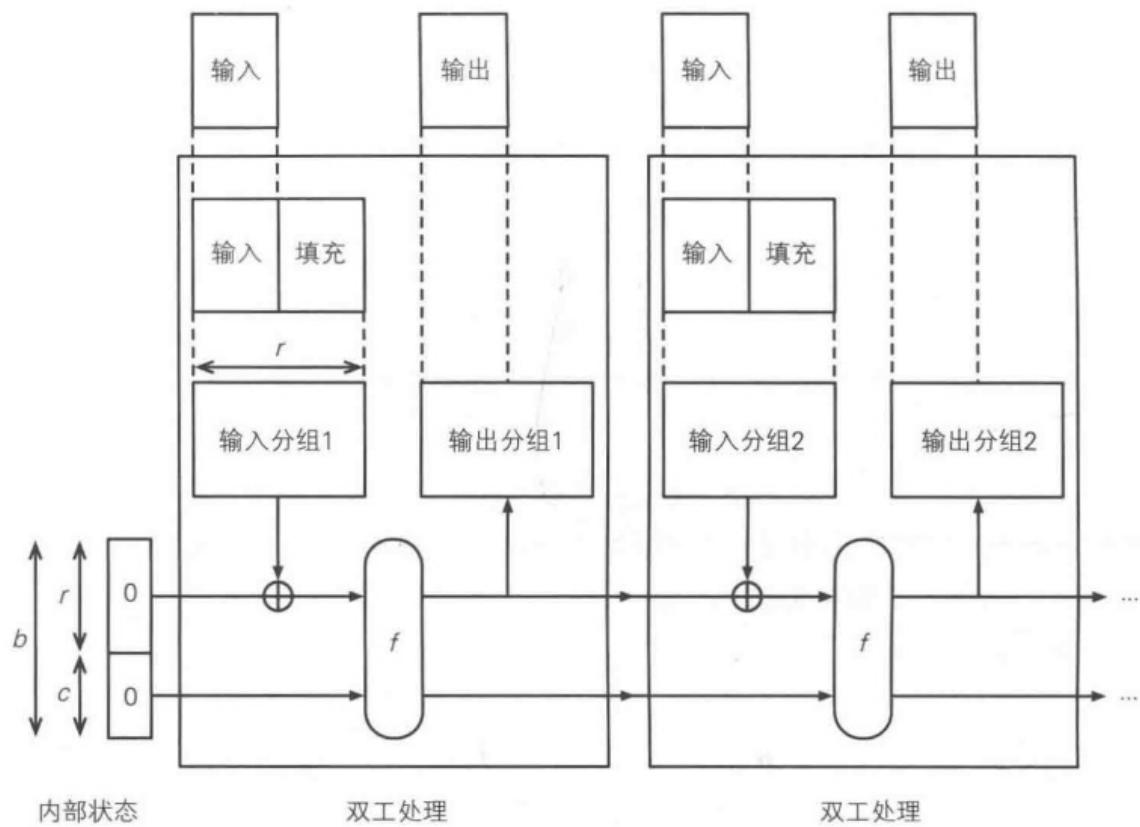
## keccak

- SHA3标准算法
- SHA3-224
- SHA3-256
- SHA3-384
- SHA3-512

海绵结构



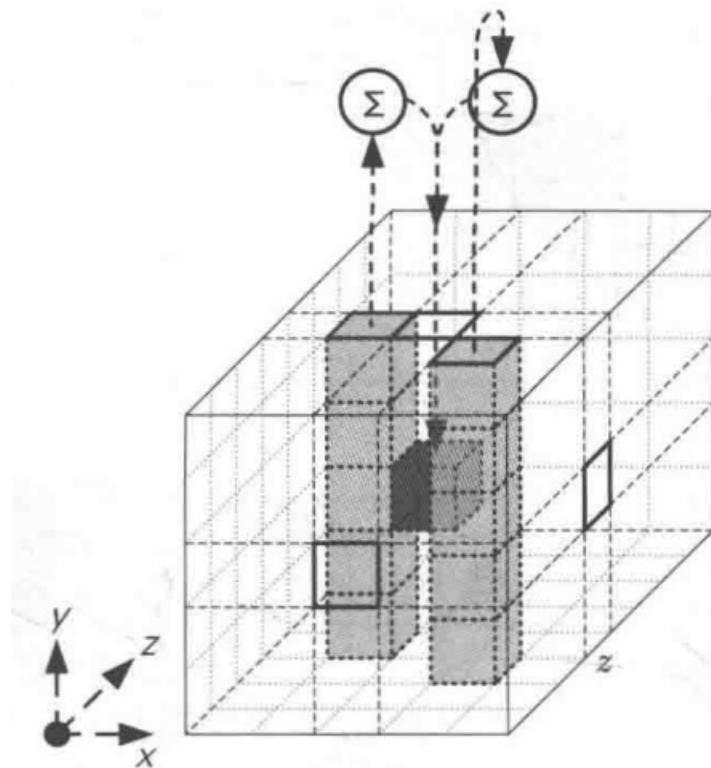
双工结构



### f函数内部构造

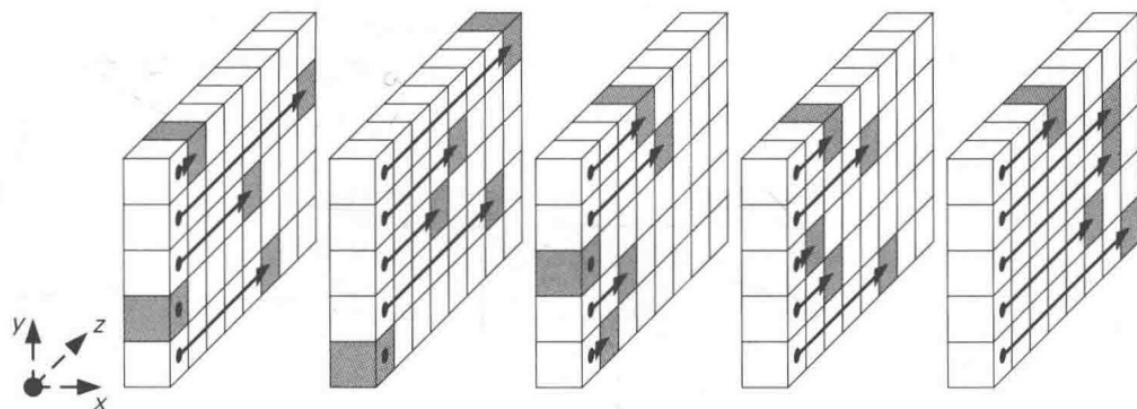
$\theta$  (西塔)

- 异或



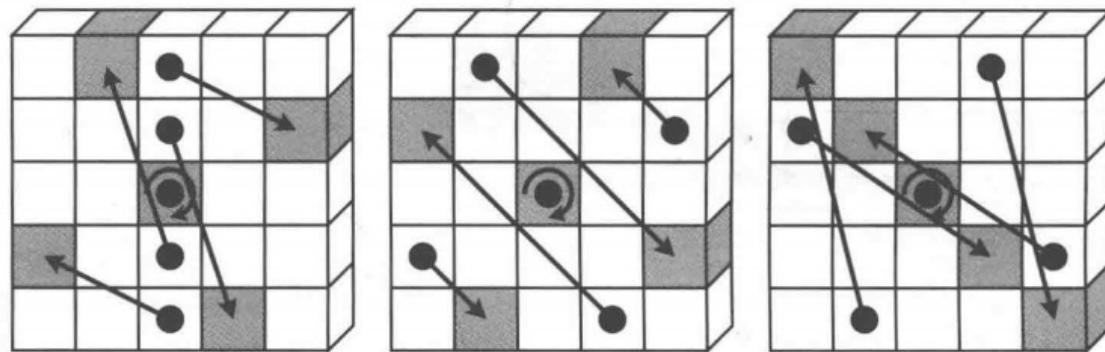
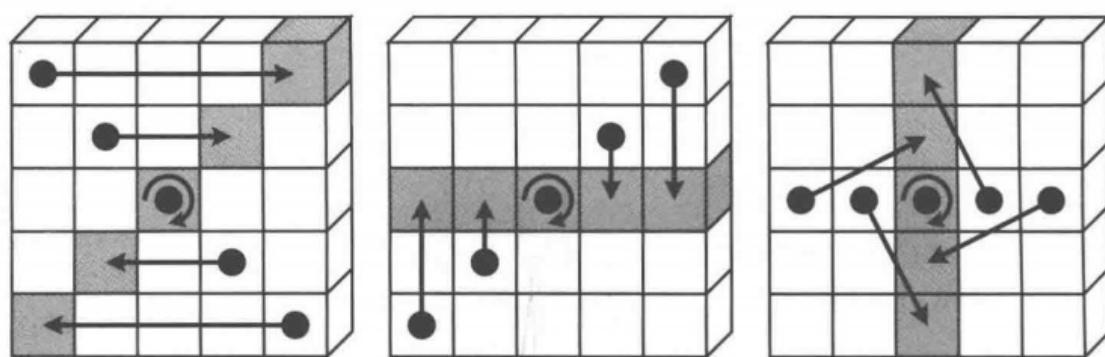
**p** (柔)

- 平移



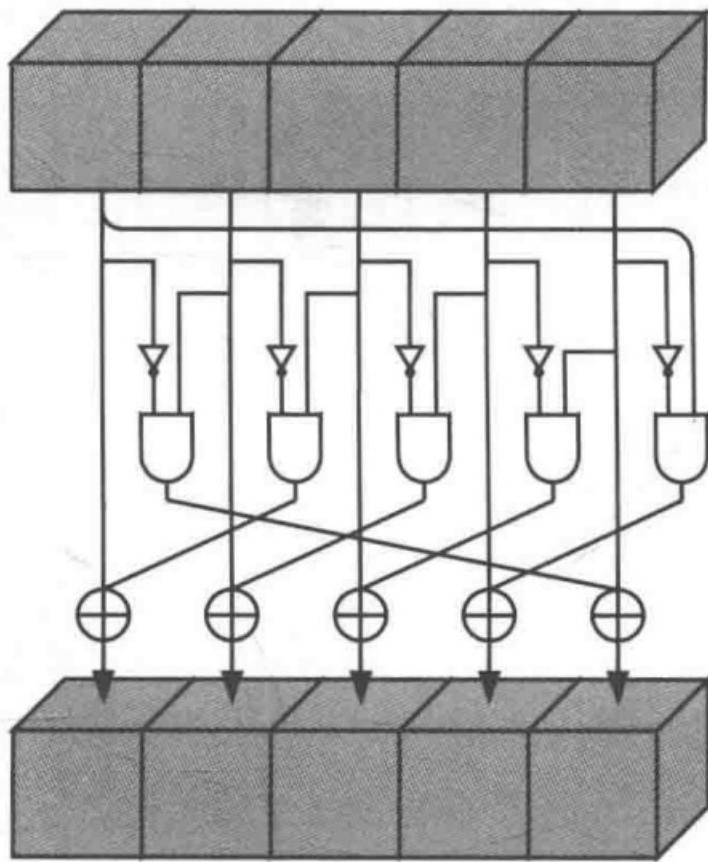
**π (派)**

- 移动



**x (凯)**

- AND
- XOR
- NOT



**T** (伊欧塔)

- 和常数进行XOR

### 疑问

**C**的存在意义

- 防止输入数据的敏感性：防止每轮输入F的操控

能够逆向还原吗

- 多个异或
- AND
- 不存在反函数

# 攻击

## 暴力破解

原像攻击

- 给定一个散列值，找到与该散列值一样的任意消息
- 碰撞

第二原像攻击

- 给定一条消息，找到与该消息散列值相同的另外一条消息
- 冗余

## 生日攻击

- $N$ 个人在一年 $Y$ 天内，至少有两人生日相同

$\$ \$ N \approx \sqrt{Y}$

$\$ \$$

假设512位长度的散列值，那么只需要

$\$ \$ N \approx \sqrt{2^{512}} = 2^{256}$

$\$ \$$

- 实际比想象要小得多

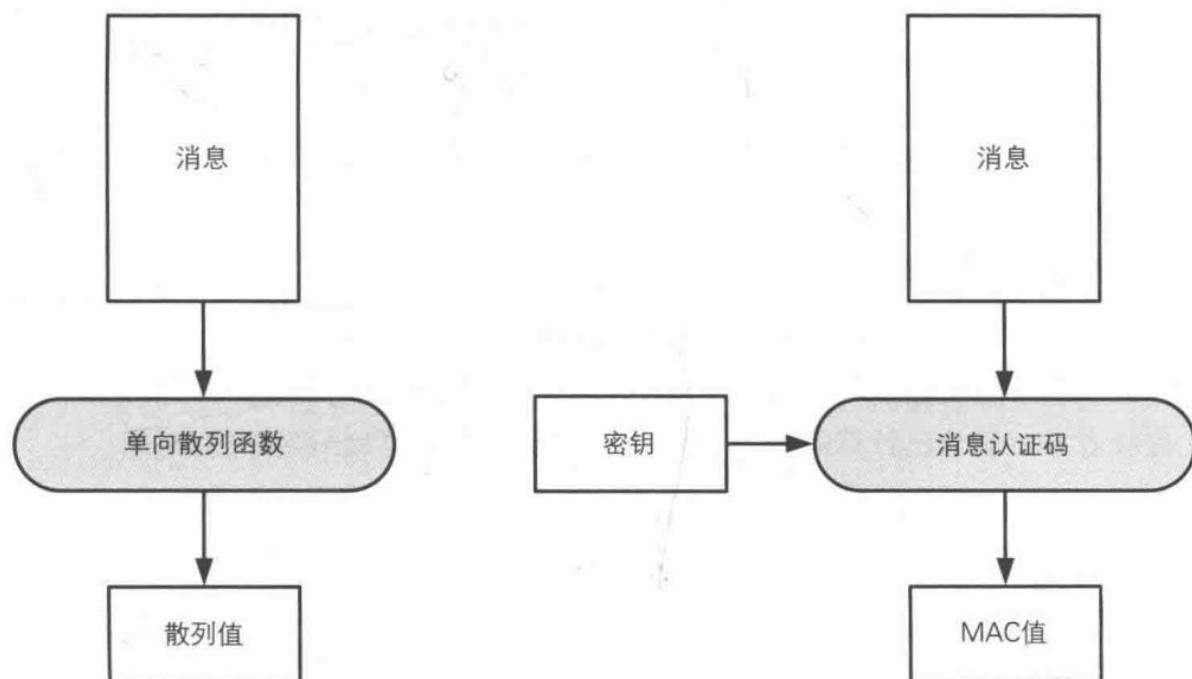
## 第七章：消息认证码

- 识别篡改、伪装
- 通过密钥进行控制

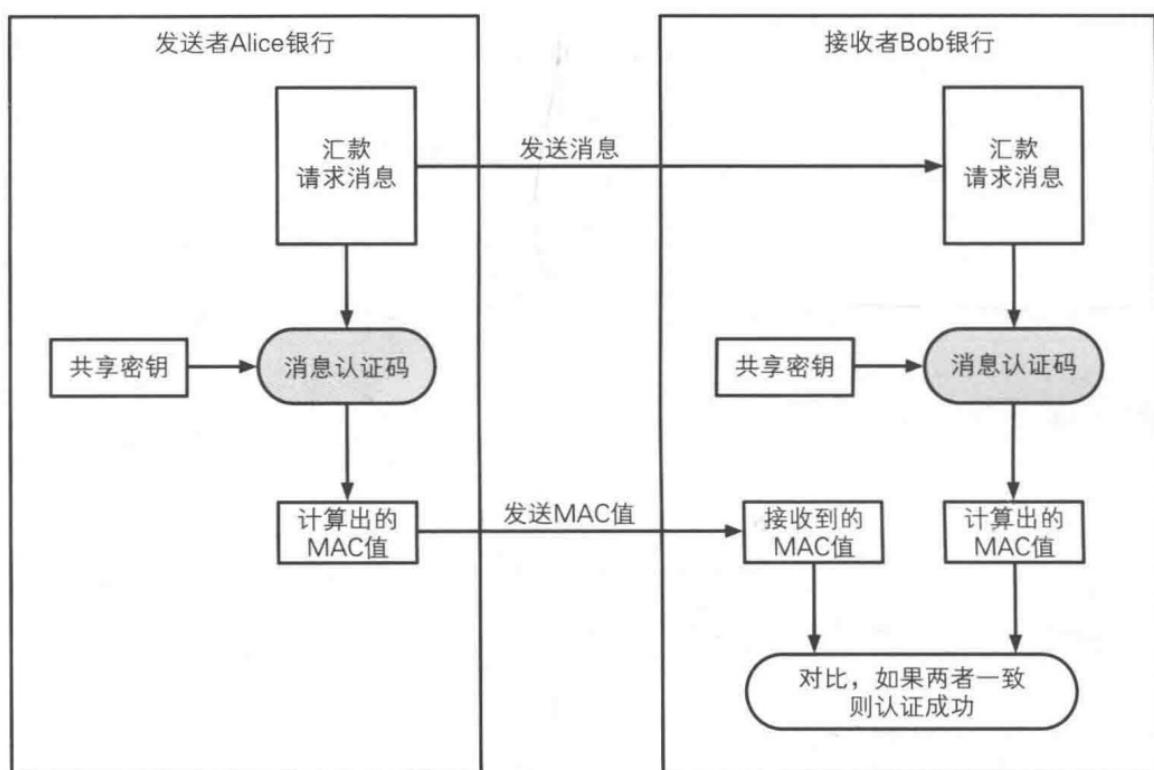
## 流程

对比

- 中间使用密钥进行验证



## 流程



## 应用

### **SWIFT**

- 环球银行金融电信协会，为国际银行交易保驾护航
- 银行与银行交易之前，密钥通过人来配送

### **IPsec**

- 互联网通信协议（IP协议）安全性的一方面，通信内容的认证和完整性校验

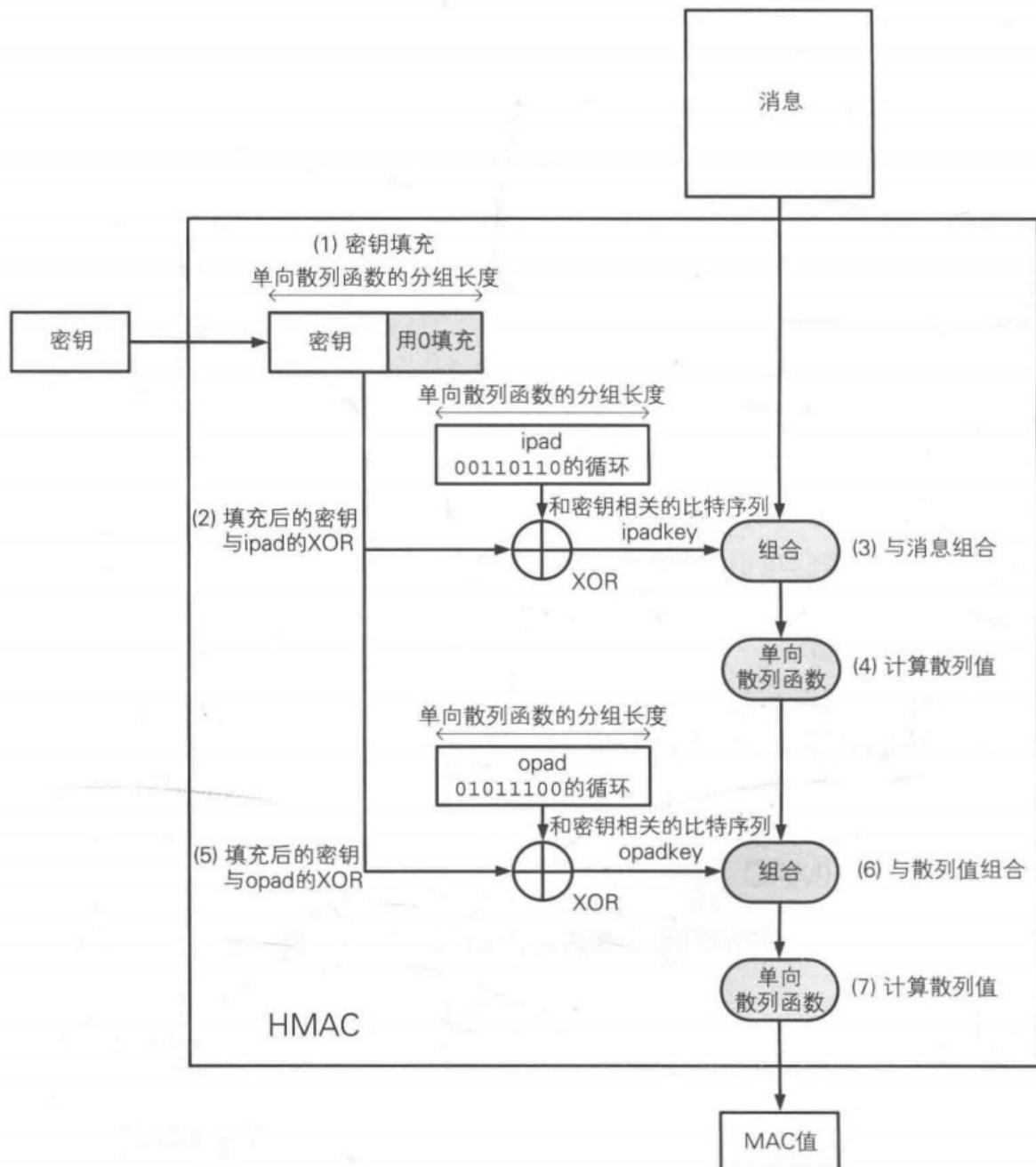
### **SSL/TLS**

- 通信内容的认证和完整性校验

# HMAC

- 散列消息身份验证码: Hashed Message Authentication Code
- 使用单向散列函数构造消息认证码的一种方法

算法过程



算法流程

HMAC 可以用下列伪代码来描述。

```
hash(opadkey || hash(ipadkey || message))
```

其中：

ipadkey 为  $\text{key} \oplus \text{ipad}$

opadkey 为  $\text{key} \oplus \text{opad}$

密钥记作 key

消息记作 message

x 的散列值记作 hash(x)

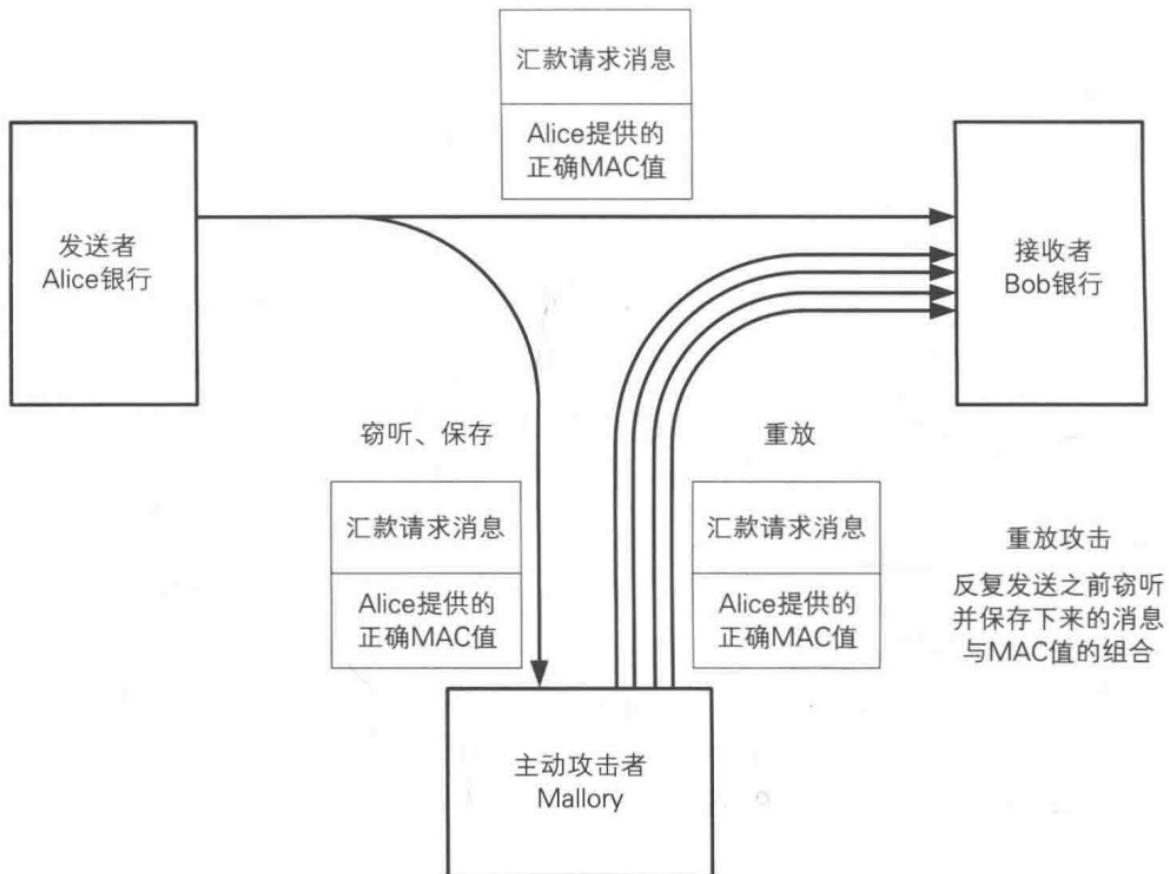
A 与 B 的组合记作 A || B

用这样的形式来描述的话，大家应该就能够理解为什么 ipadkey 是“内部”，而 opadkey 是“外部”了吧。因为 ipadkey 是在两层 hash 中的里面一层中出现的。

# 攻击

## 重放攻击

例子

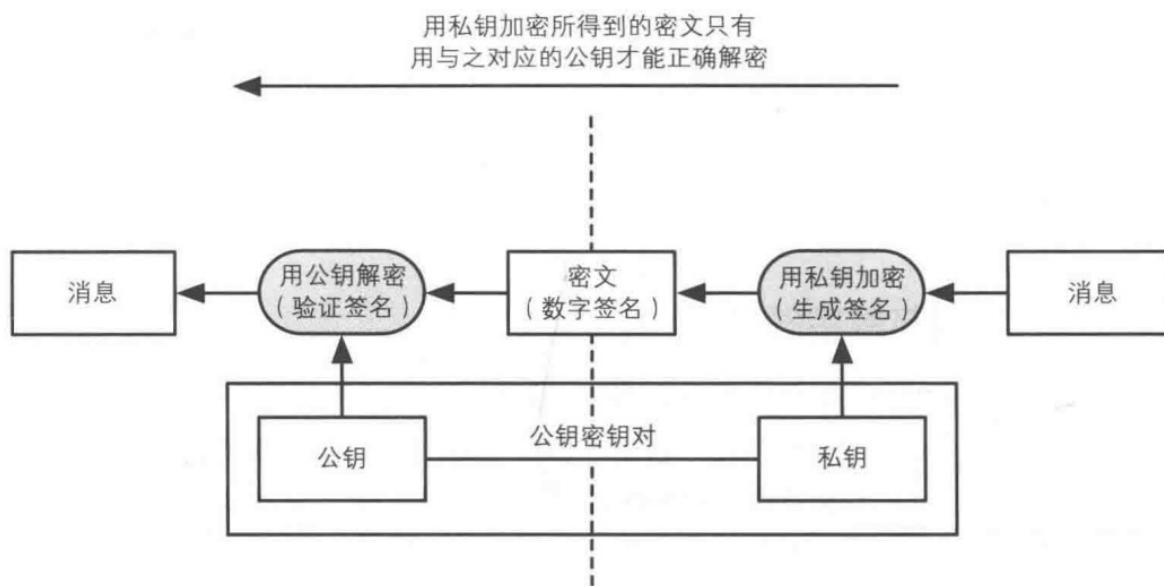


避免

- 传送递增序号
- 当前时间戳
- 传送随机数

## 第八章：数字签名

- 防止否认、篡改、伪装
- 利用公钥密钥
- 利用私钥只有个人持有的特性来防止否认

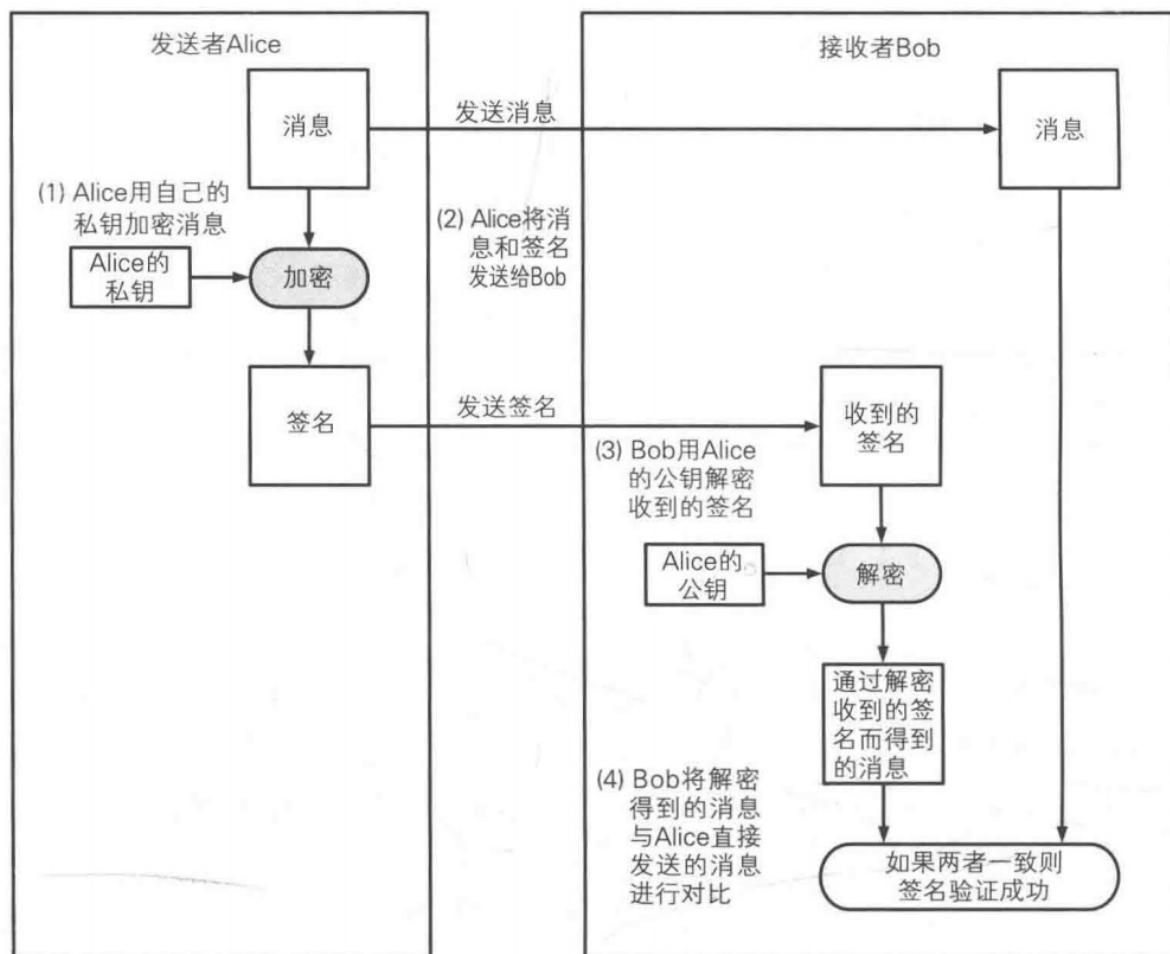


# 流程

对消息签名

- 一般不使用
- 太慢

对消息的散列值签名



# 攻击

## 中间人攻击

- 窃听者在中间交换密钥

## 单向散列函数攻击

- MD5
- SHA-1

## 数字签名攻击公钥密码

- 公钥加密内容，让对方来签个名（解密）

## 对比

- 密钥是机密性的精华
- 散列值是完整性的精华

	对称密码	公钥密码
发送者	用共享密钥加密	用公钥加密
接收者	用共享密钥解密	用私钥解密
密钥配送问题	存在	不存在，但公钥需要另外认证
机密性	○	○

	消息认证码	数字签名
发送者	用共享密钥计算 MAC 值	用私钥生成签名
接收者	用共享密钥计算 MAC 值	用公钥验证签名
密钥配送问题	存在	不存在，但公钥需要另外认证
完整性	○	○
认证	○ ( 仅限通信对象双方 )	○ ( 可适用于任何第三方 )
防止否认	×	○

## 第九章：证书

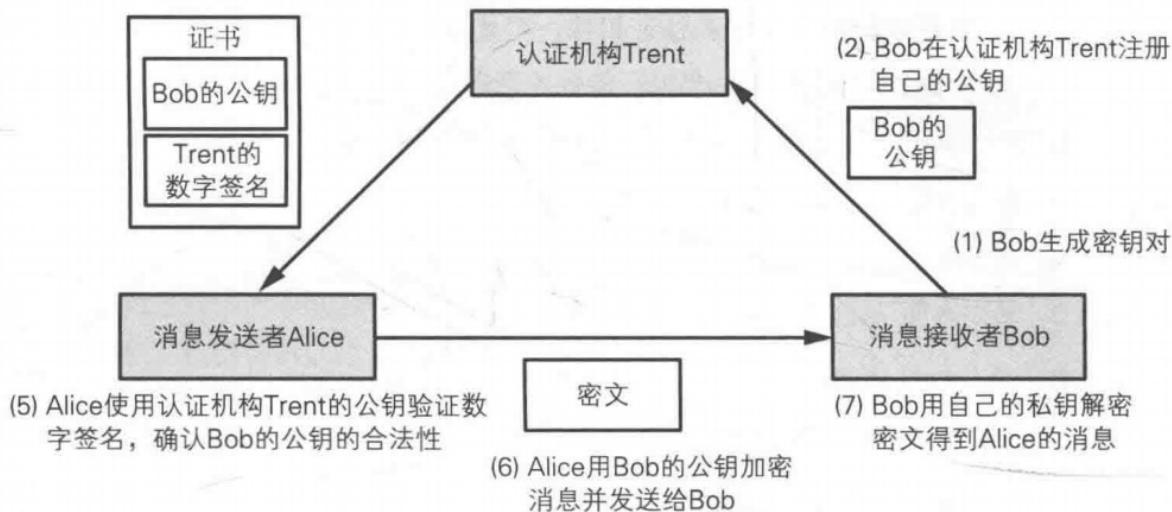
- 数字签名中的公钥怎么保证合法性
- 防止公钥的中间人攻击

## 流程

- 类比百融

(4) Alice得到带有认证机构Trent的数字签名的Bob的公钥（证书）

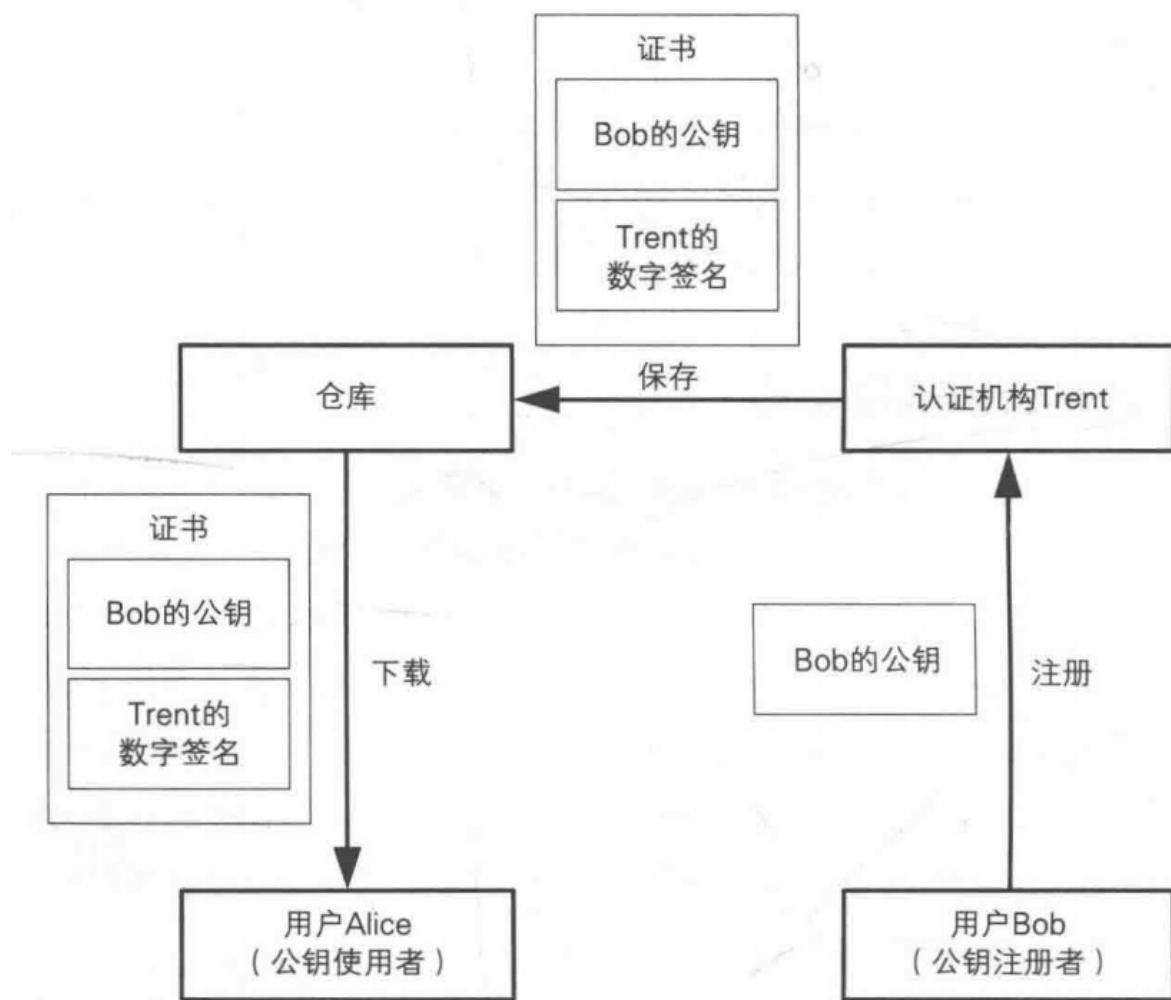
(3) 认证机构Trent用自己的私钥对Bob的公钥施加数字签名并生成证书



## PKI

(Public-Key Infrastructure) 公钥基础设施，为了有效运用公钥制定的一系列规范和规格总称。

流程



## CRL

(Certificate Revocation List) 证书作废清单

## CA

(Root Certificate Authority) 根证书认证机构

# 攻击

## 公钥注册前窃取

防止

- 注册前使用机构公钥加密
- 发送指纹

## 相似人名攻击

- tencent
- Tencent

防止

- 机构进行详细规则验证

## 窃取认证机构私钥

防止

- 及时更新CRL
- 及时检查序列号

## 伪装认证机构

防止

- 注意自己获取的证书的机构是否可信

## CRL攻击

- 时差：私钥被盗
- 否认：自导自演

## 第十章：SSL/TLS

- 网络上的通信是否安全？

# 概述

## 什么是SSL/TLS

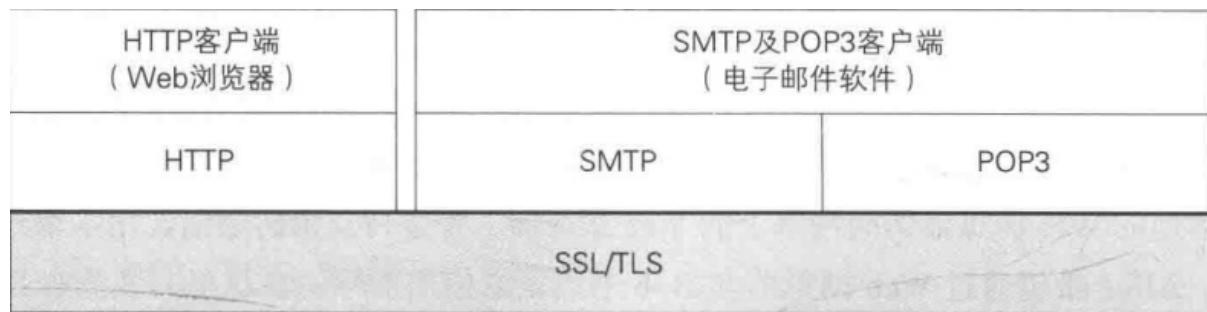
### SSL

- SSL (Secure Socket Layer) 安全套接字层
- 1994年发布
- 1995年发布SSL3.0
- 2014年SSL3.0发现POODLE
- 已经不安全

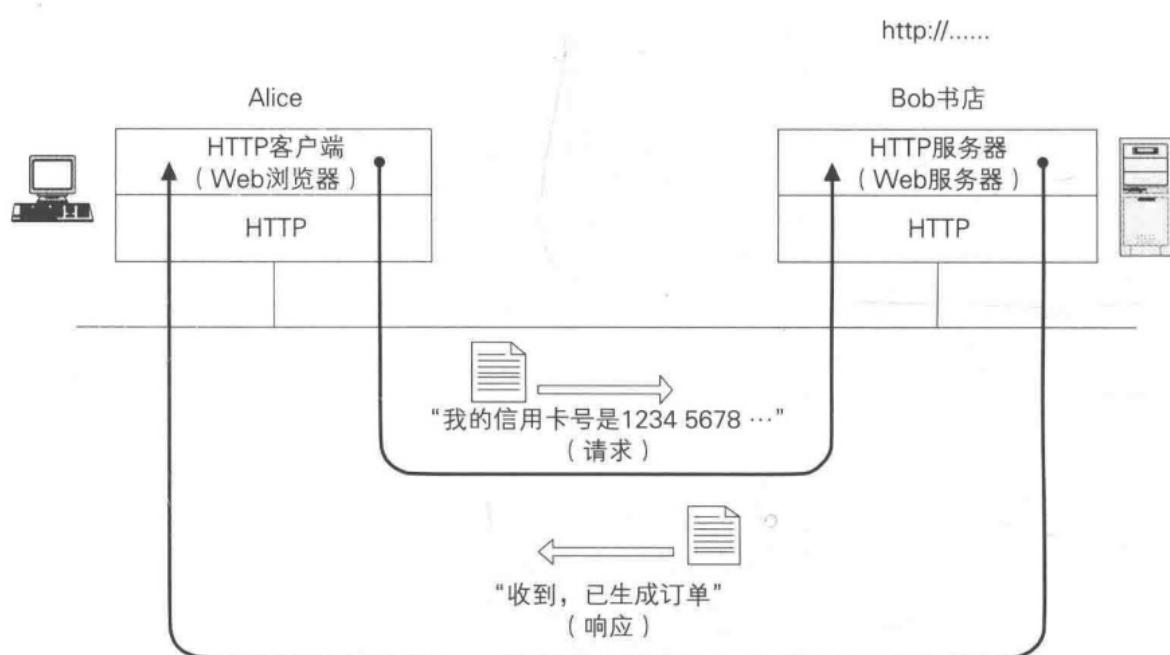
### TLS

- TLS (Transport Layer Security) 传输层安全
- TLS是SSL的升级版
- 1999年TLS1.0, 等于SSH3.1
- 2006年TLS1.1
- 2008年TLS1.2
- 2018年TLS1.3

TLS用于在两个通信应用程序之间提供保密性和数据完整性。该协议由两层组成：TLS记录协议（TLS Record）和TLS握手协议（TLS Handshake）

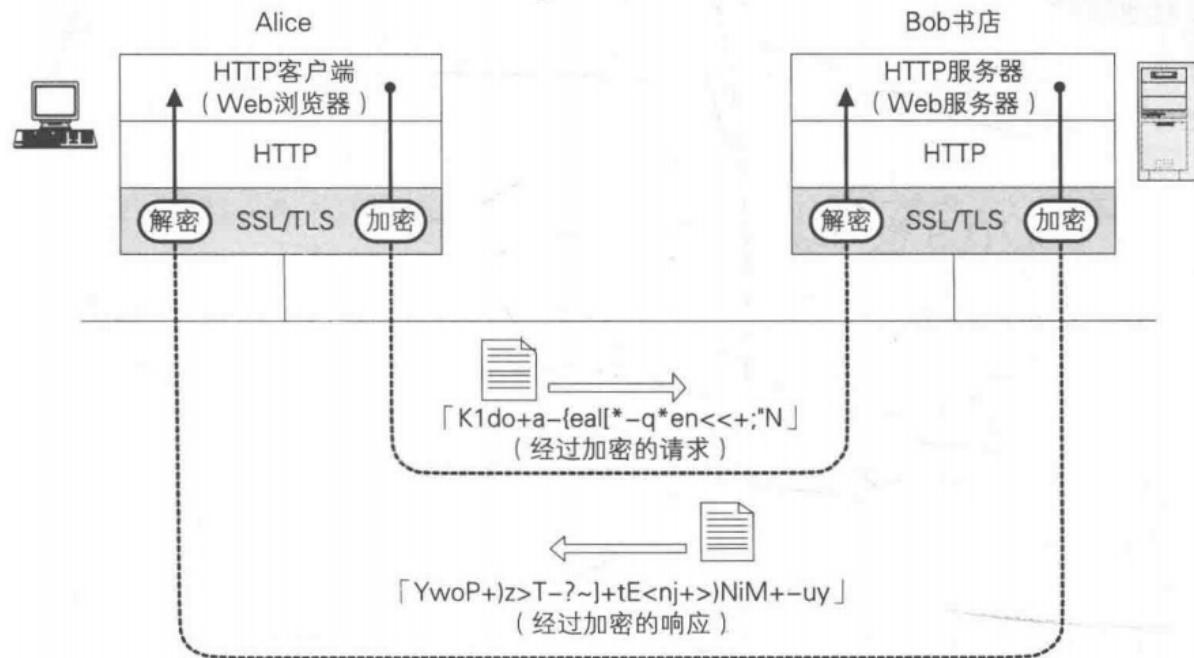


## HTTP



## HTTPS

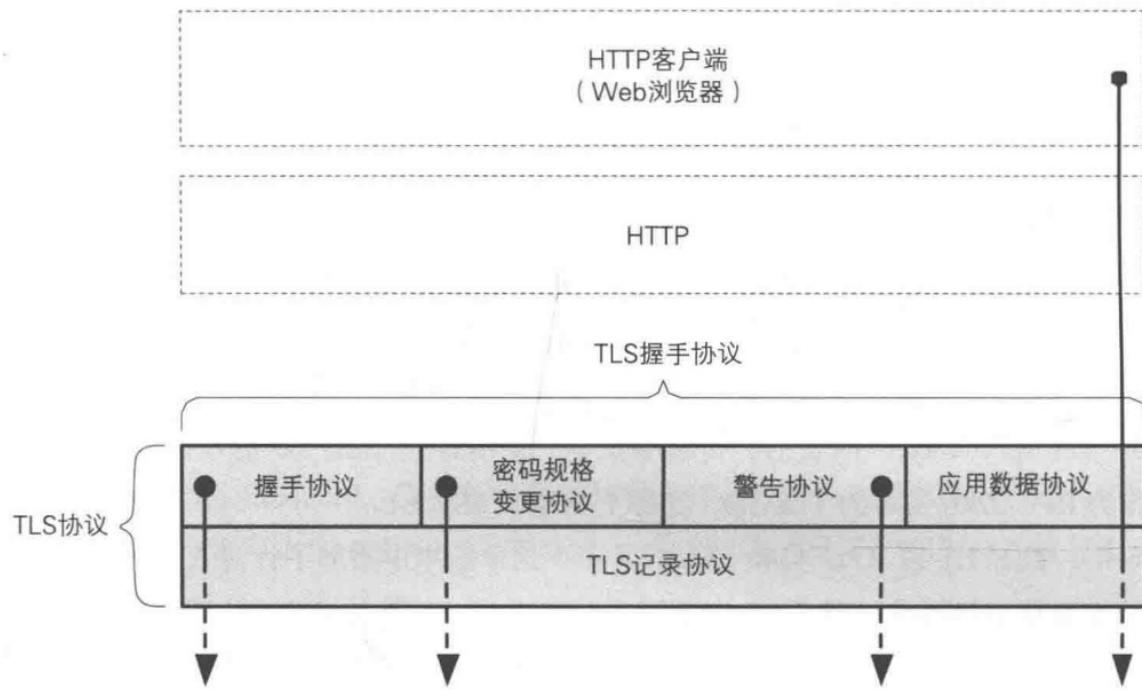
https://.....



# 流程

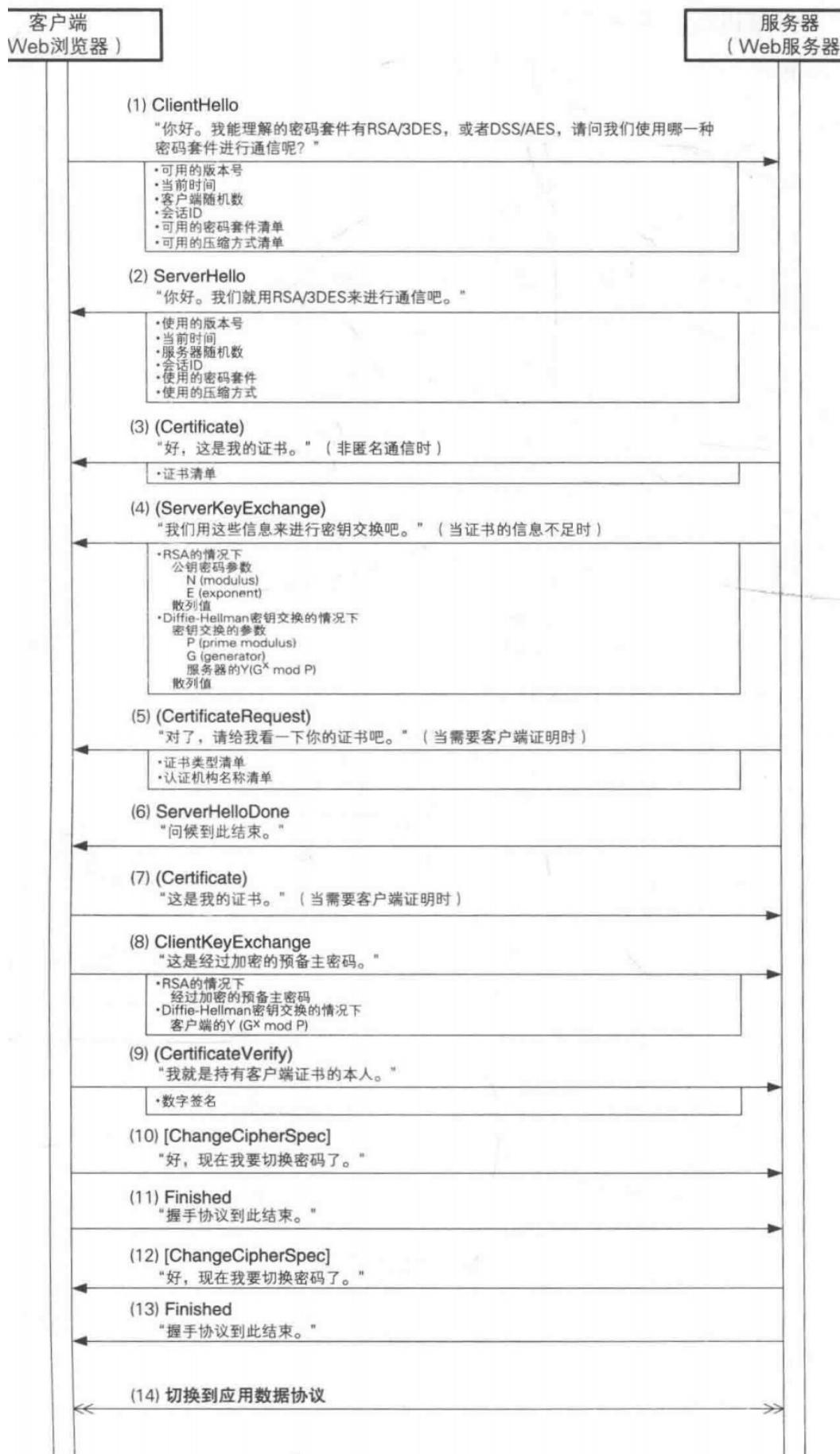
## 协议层次

- 位于传输层

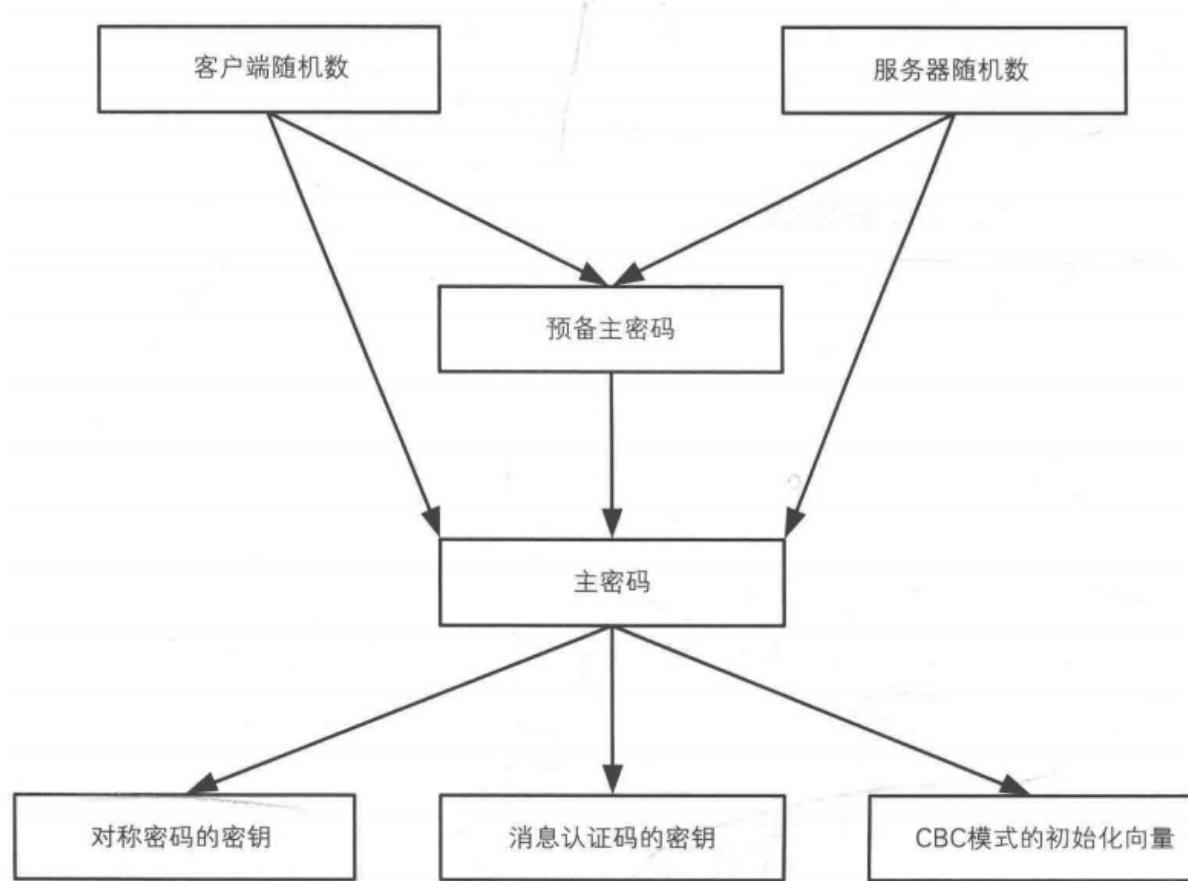


## 通信流程

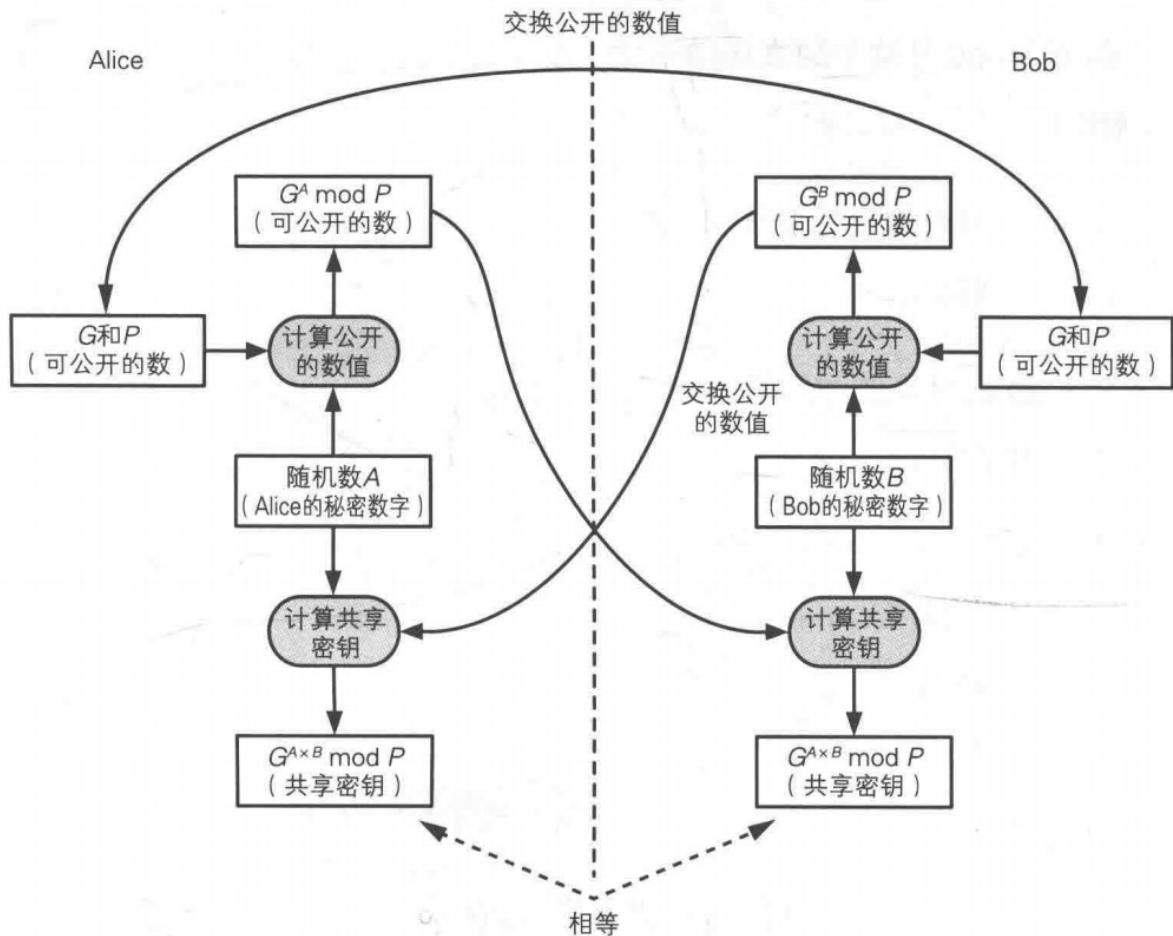
- **TLS记录协议**: 信息压缩、加密、认证
- **握手协议**: 协商加密信息用的加密算法和密钥
- **密码规则变更协议**: 传达变更密码消息
- **警告协议**: 协议发生错误时传到给对方（比如：无法解密）
- **应用数据协议**: 和通信对象之间传输数据



预备主密码



**Diffie-Hellman**密钥交换



# 攻击

## OpenSSL心脏出血漏洞

- 2014被Google发现
- 心跳扩展功能没有对请求数据大小做检查，导致将内存中与该请求无关的信息返回给请求者

OpenSSL的心跳处理逻辑没有检测心跳包中的长度字段是否和后续的数据字段相符合，攻击者可以利用这一点，构造异常的数据包，来获取心跳数据所在的内存区域的后续数据。这些数据中可能包含了证书私钥，用户名，用户密码，用户邮箱等敏感信息。该漏洞允许攻击者从内存中读取多达64KB的数据。

## SSL3.0漏洞

- 2014年被Google发现
- POODLE攻击（CBC填充提示攻击）
- 某些情况，TLS可以强制降级为SSL3.0

必须禁用SSL3.0

## FREAK（怪异）攻击

- 2015年发现
- 强制SSL/TLS使用一种RSA Export Suites的低强度密码套件
- 上世纪，美国禁止出口高强度密码软件导致

证书作废时间差攻击

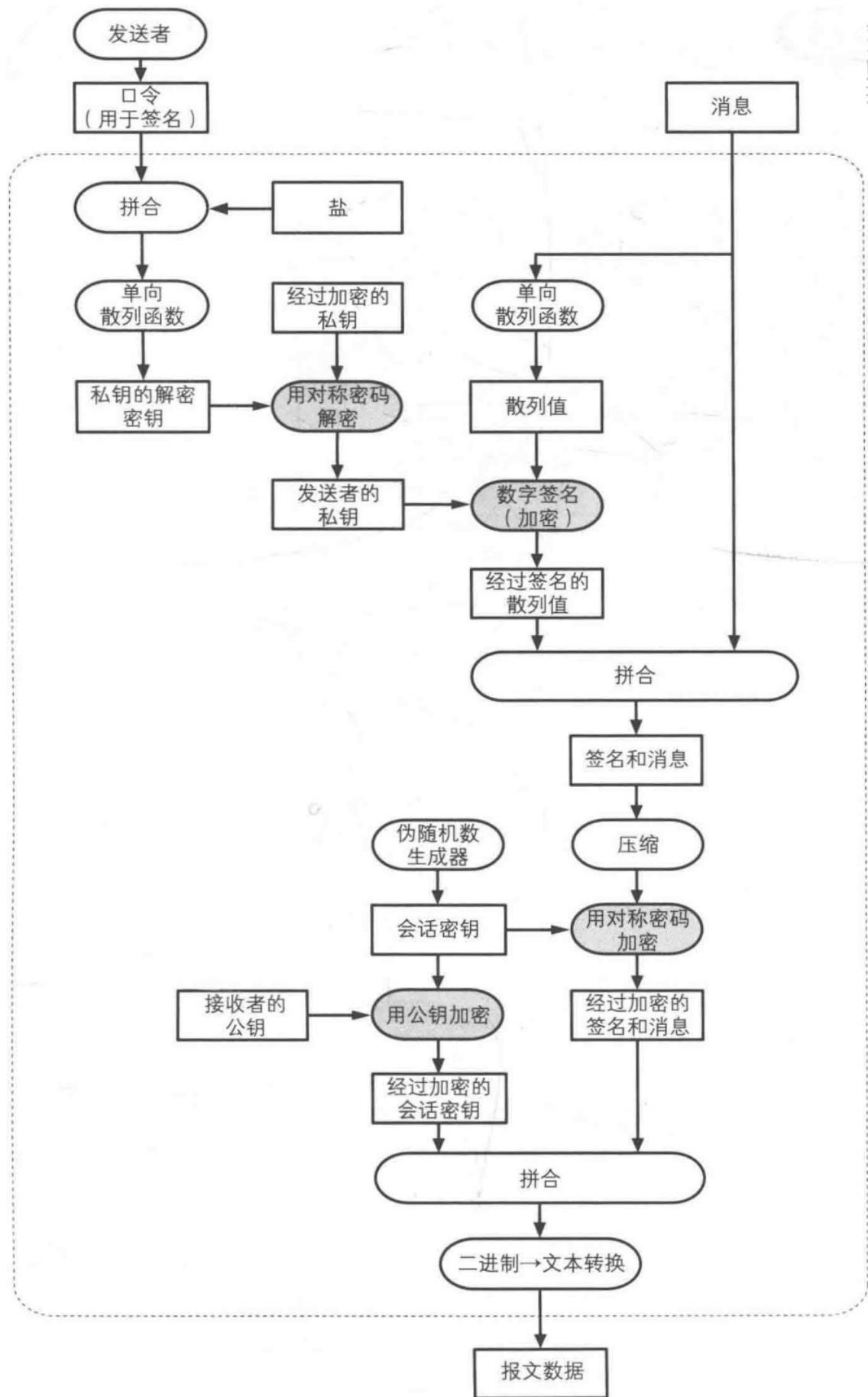
伪随机数攻击

## 第十一章：应用与现实

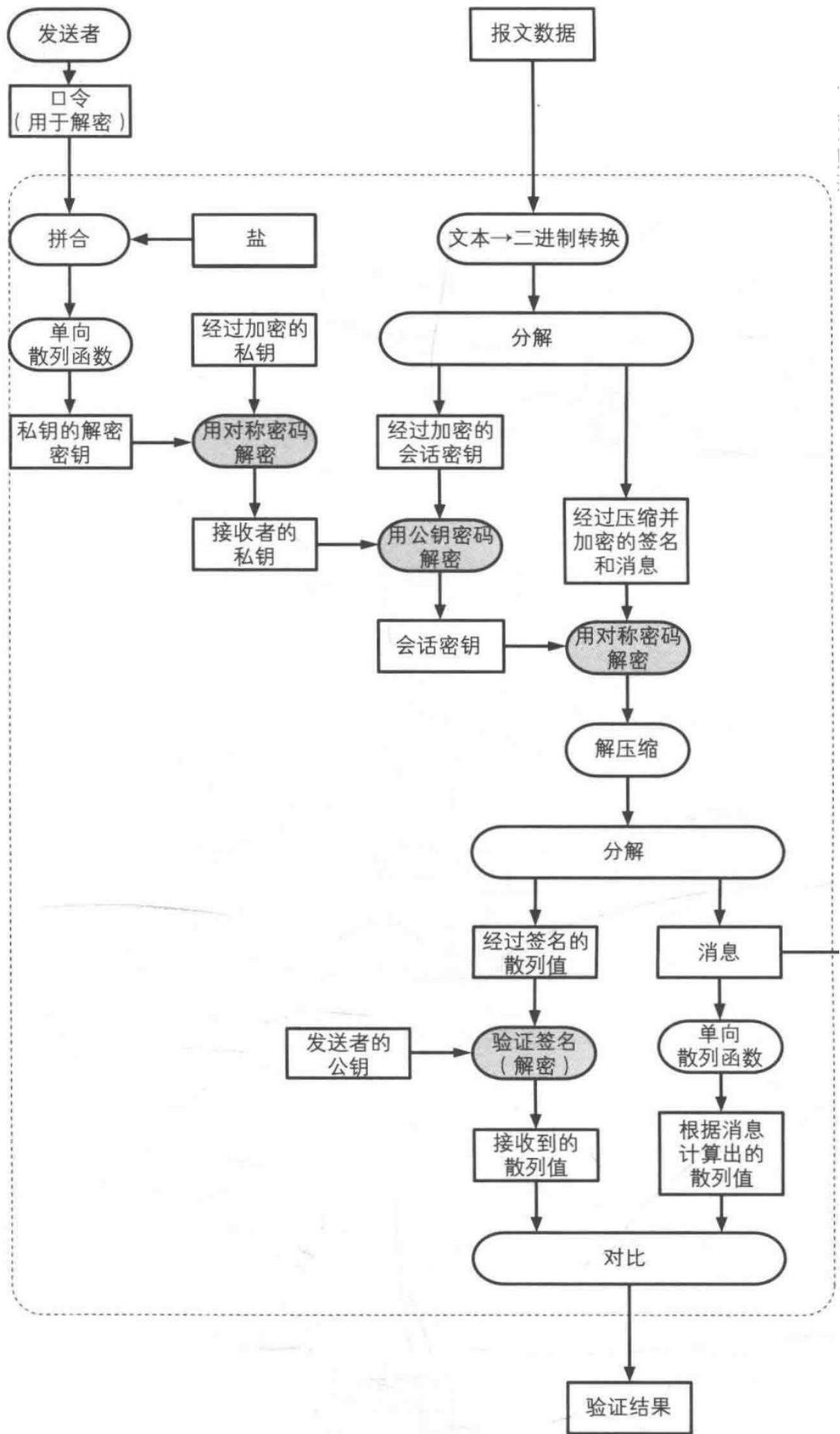
- 密码学在现实中的应用有什么？
- 未来发展怎么样？
- 现实状况是什么样的？

离线混合数字签名

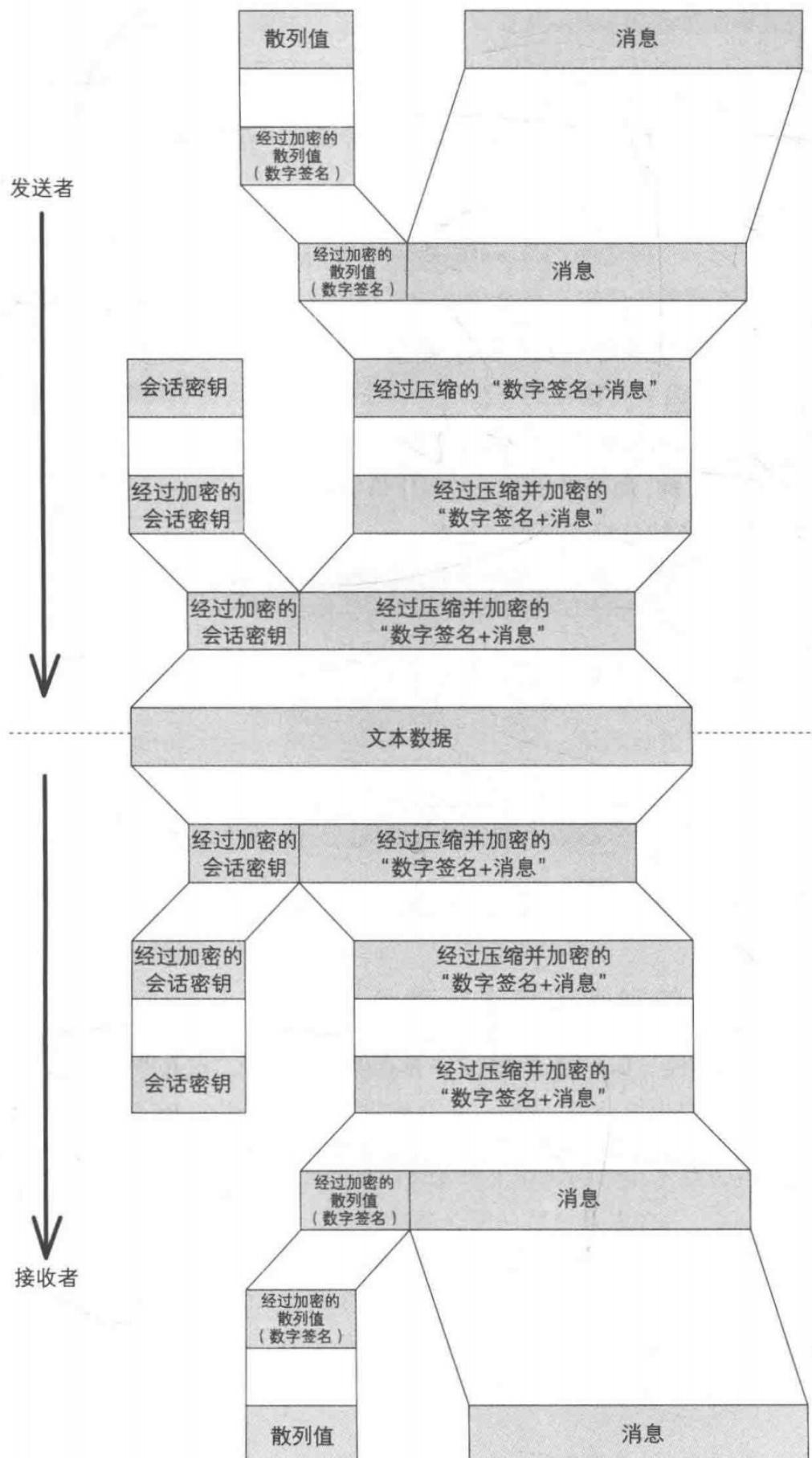
生成数字签名



验证数字签名



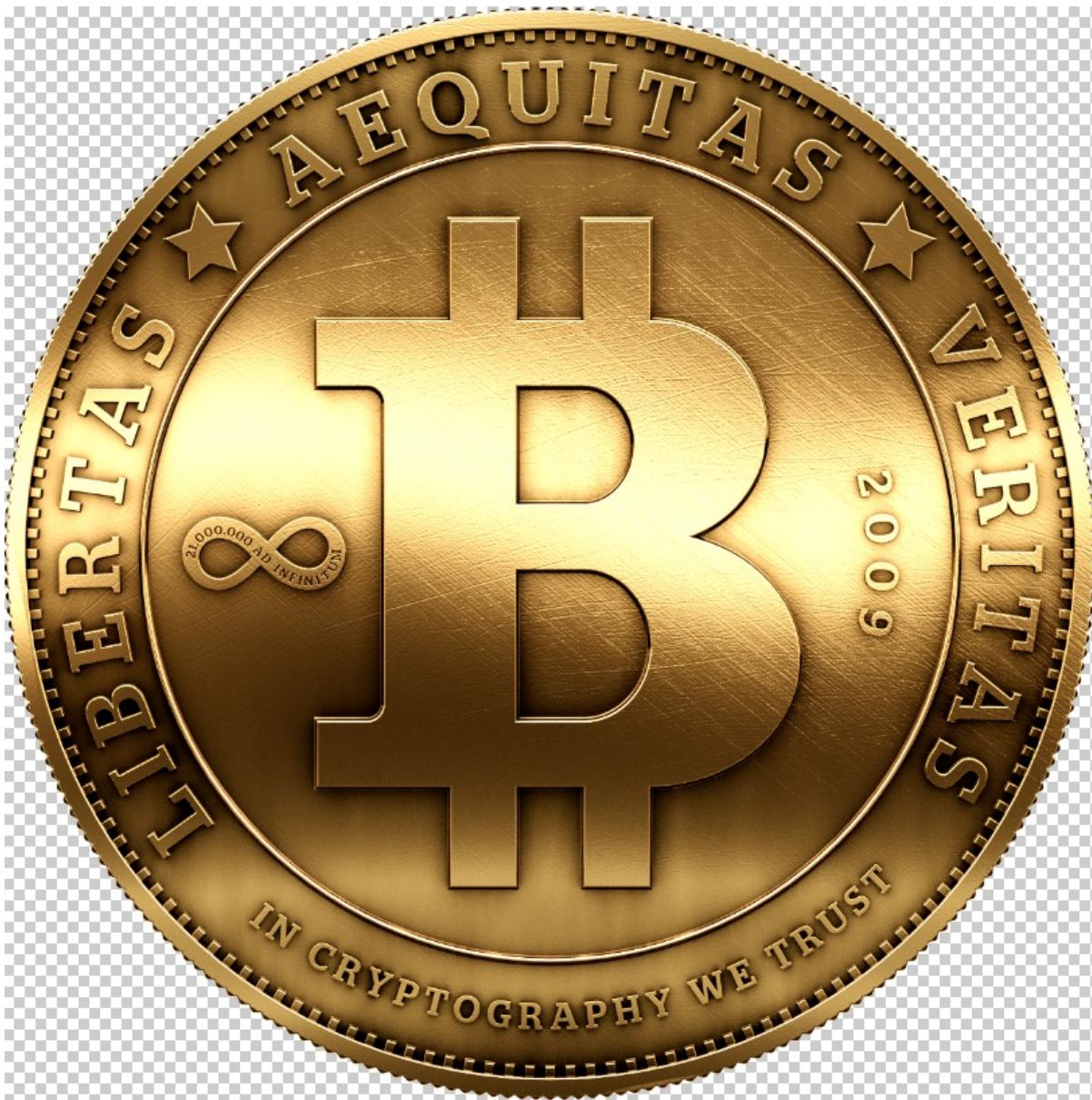
## 整体数据流程





## 比特币

- 虚拟货币
- 2009年由中本聪发布
- 区块链之父



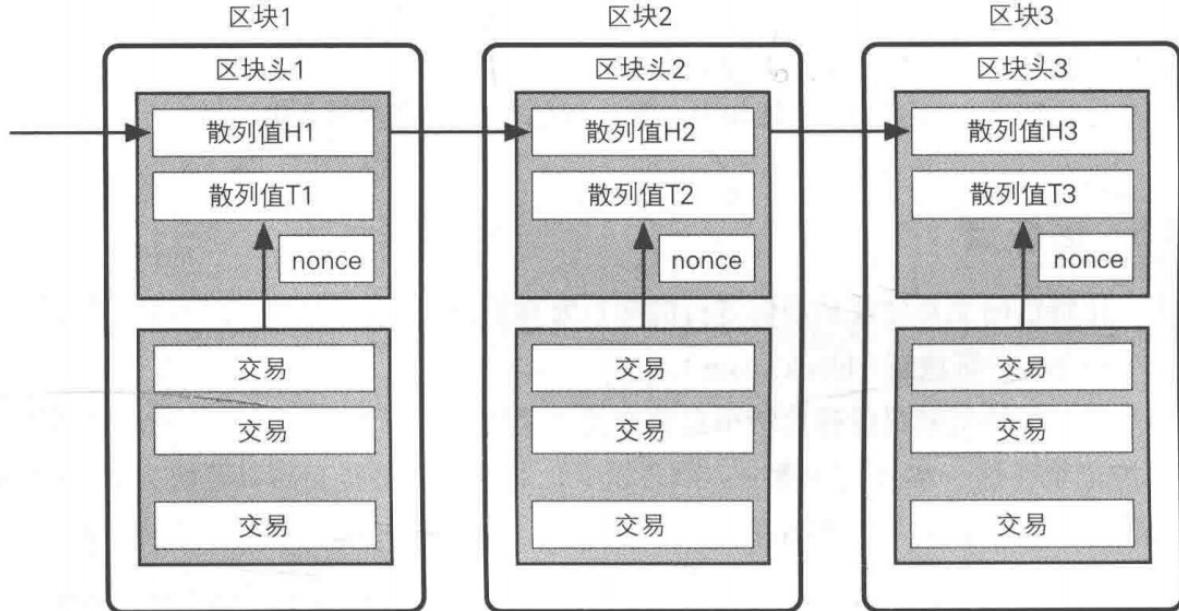
诞生

去中心化

- 通货膨胀，银行增钞（不靠谱）
- 不能由一个中心决定价值
- 可由每个人监督（分布式账本）
- 每一笔交易都可追溯
- 每一笔交易都公开
- 每一笔交易都不可篡改
- 解决信任问题

## 账本（区块链）

- 篡改非常困难



```
当前区块哈希 = Hash(交易内容 + 前一区块哈希 + 当前区块高度 + 时间戳 + Nonce);  
挖矿难度 = 哈希的前4位数为0;
```

```
if ( 当前区块哈希 == 挖矿难度 ){  
   Nonce 有效;  
    当前区块哈希 有效;  
}
```

## 加密算法

- RIPEMD-160: 单向散列函数
- DSA: 椭圆曲线

## 完美密码

### 量子密码

- 80年代提出
- 1989年美国实现32KM传输

无法准确测出光子的偏振方向

- 让窃听者得到的内容变得不正确

测量行为会导致光子的状态发生变化

- 接收者可以判断出通信是否被窃听

优点

- 无法破解

缺点

- 暂时无法远距离传输

### 量子计算机

- 量子密码是密码学家的终极工具
- 量子计算机是密码破译者的终极工具

一个粒子可以同时拥有多种状态，如果一个粒子同时计算0和1，那么128个粒子，则可以同时计算：

$\$ \$ 2^{128}$

$\$ \$$  种状态，现在的密码基本可以瞬间解密。

优点

- 并行计算能力无比强大

缺点

- 暂时没有进入实用领域

# 现实

## 人

- 就算有完美的加密算法，也没有完美的人
- 即使有了量子密码也一样
- 生物识别最终也是序列流，和密钥等价

## 防御与攻击

- 防御需要天衣无缝
- 攻击只需攻破一点

## 社会工程学

- 因为社会工程学的存在（伪装员工、**Https Dos**攻击）