现代密码学

复习（2023）

信息与软件工程学院

目的

毕业要求指标点：

- GR3.1 能够根据用户需求，分析和确定设计目标
- GR3.2 能够在社会、健康、安全、法律、文化以及环境等约束条件下，通过技术经济评价对设计方案的可行性进行研究
- GR6.1掌握至少一个应用领域中软件工程技术的应用方法和工程实践
- GR10.1能够撰写报告和设计文稿，清晰阐述复杂工程问题

课程目标

- CO1：掌握分析保密通信系统中安全需求的方法
- CO2：理解密码学的基本概念、基本原理和一些典型的密码算法的原理
- CO3：理解各类密码算法的应用场景和相关的安全需求
- CO4：掌握常用密码算法的实现

考核方式与成绩构成

- 考核方式： 笔试+课程报告
- 成绩构成：平时50% + 期末50%
- 平时成绩主要来源于作业（30%）和课程报告（20%），期末考试为闭卷考试。
- 平时作业来源于SPOC，其中单元测验20%，单元作业20%，期末考试50%，在线讨论10%。
- 课程报告的考核：学生分组选题，课外进行分析、设计和实现，提交课程研讨小论文。要求每一组做一次陈述，每小组要有明确分工，结题汇报时每人上台陈述自己所做工作（1-2分钟）。个人评分10%，课程研讨小论文60%，结题汇报30%。

期末考试题型

- 判断题，4题，共20分
- 简答题，4题，共20分
- 计算题，3题，共30分
- 综合题，2题，共30分

目 录

- 引言
- 流密码
- 分组密码
- 公钥密码
- 消息摘要和杂凑算法
- 数字签名
- 密码协议

第一章 引言

复习要点

- 密码学的基本概念
- 密码学的分类
- 密码体制攻击的四种类型
- 保密系统的安全性分析

什么是密码？

- 什么是密码？
  - 密码是指采用特定变换的方法对信息等进行加密保护、安全认证的技术、产品和服务。——《中华人民共和国密码法》
- 什么是密码学？
  - 密码学是研究编制密码和破译密码的技术科学。
    - 研究密码变化的客观规律，应用于编制密码以保护通信秘密的，称为密码编码学。
    - 应用于破译密码以获取通信情报的，称为密码分析学或破译学。

Cryptology（密码学）= Kryptós（隐藏）+ Lógos（信息）

密码学可解决的信息安全问题

1、机密性 Confidentiality

保证信息为授权者享用而不泄漏给未经授权者

2、完整性 Integrity

数据完整性，未被未授权篡改或者损坏

3、认证（Authentication）

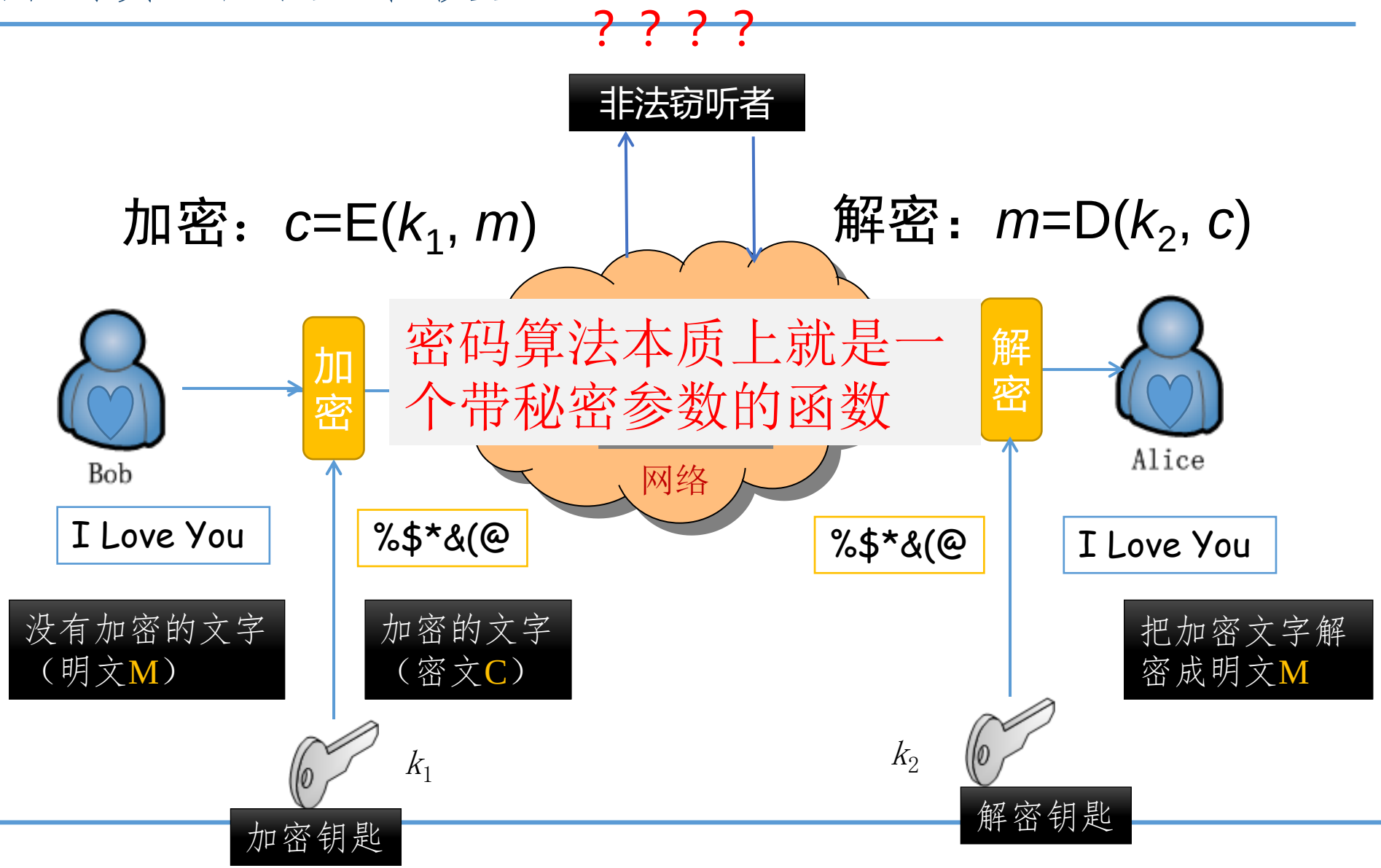
消息认证，保证消息来源的真实性

身份认证，确保通信实体的真实性

4、信息的不可否认性（Non-repudiation ）

要求无论发送方还是接收方都不能抵赖所进行的传输

密码算法的基本模型


？？？？
非法窃听者
加密：$c=E(k_1, m)$
解密：$m=D(k_2, c)$
密码算法本质上就是一个带秘密参数的函数
加密
解密
网络
Bob
Alice
I Love You
%$*&(@
%$*&(@
I Love You
没有加密的文字（明文M）
加密的文字（密文C）
把加密文字解密成明文M
k_1
k_2
加密钥匙
解密钥匙

基本概念

- 明文M ——要处理的数据——**Message (Plaintext)**
- 密文C ——处理后的数据——**Ciphertext**
- 密钥k ——秘密参数——**Key**
- 加密函数：$C = E(k, M)$或$C = E_k(M)$——**Encryption**
- 解密函数：$M = D(k, C)$或 $M = D_k(C)$——**Decryption**

密码算法（续）

- 密码算法需求：
  - 需求1：可逆——算法的使用者可以将密文恢复成明文
  - 需求2：不可逆——敌手无法将密文恢复成明文
  - 秘密参数——密钥
- 密码算法实际上是一个带有秘密参数的函数。
  - 知道秘密参数，求逆非常容易
  - 不知道秘密参数，求逆是不可行的

密码算法（续）

一个好的密码体制至少应满足的两个条件：

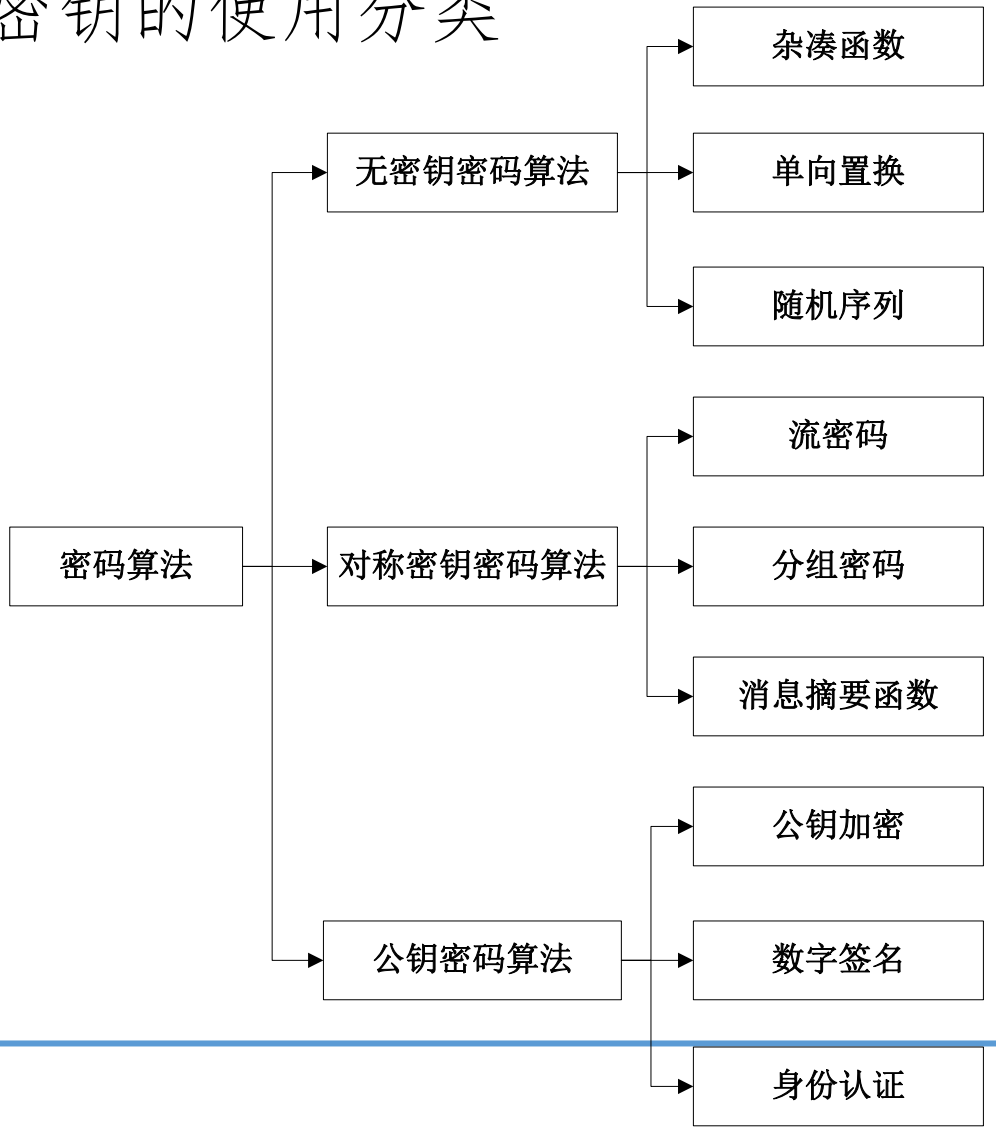
（1）在已知明文m和加密密钥k_1时，计算$c = E_{k_1}(m)$容易，

在已知密文c和解密密钥k_2时，计算$m = D_{k_2}(c)$容易；

（2）在不知解密密钥k_2时，不可能由密文c恢复出明文m。

密码算法分类

按照密钥的使用分类



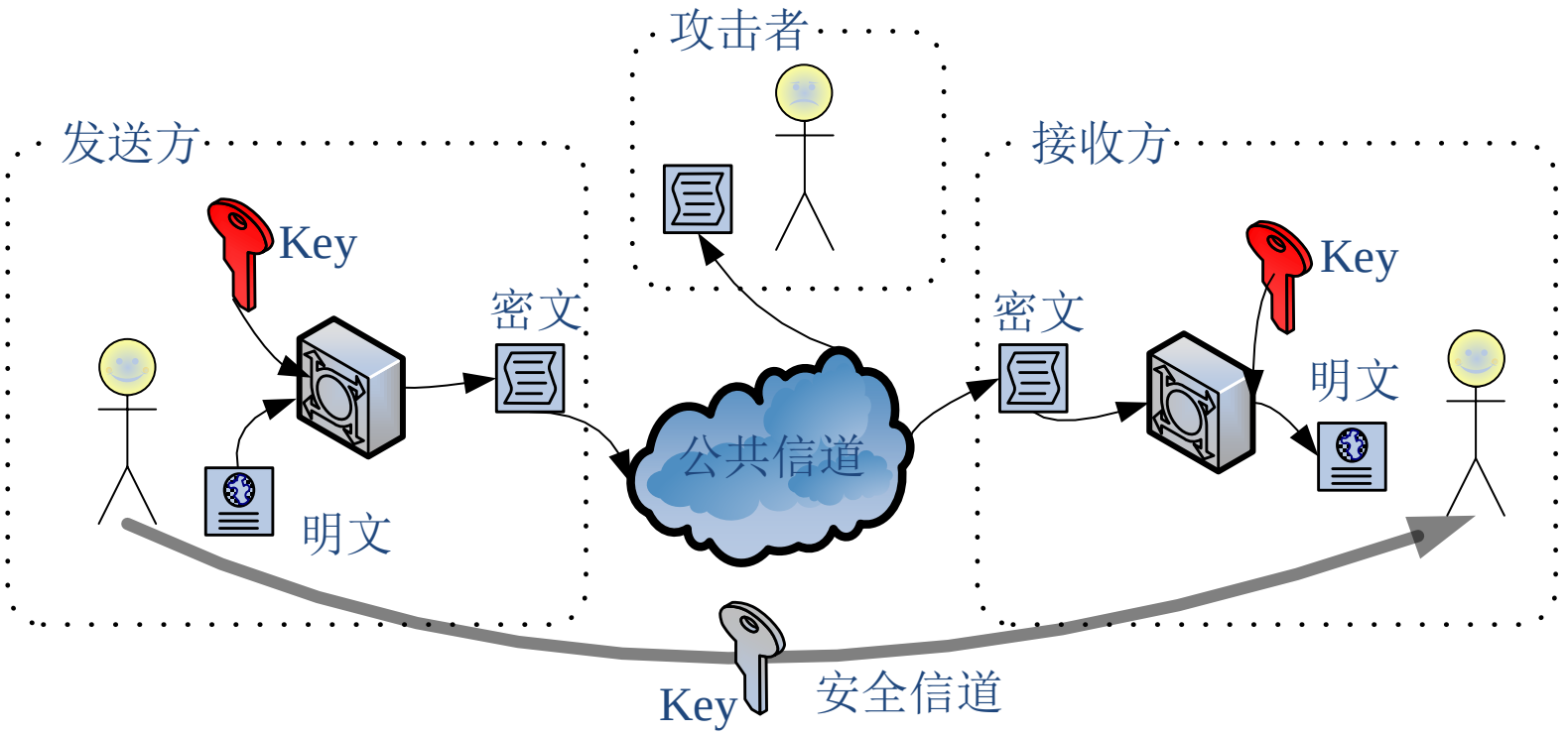
按照功能分类

加密算法：用于机密性解决方案

杂凑函数、消息摘要函数：用于完整性解决方案

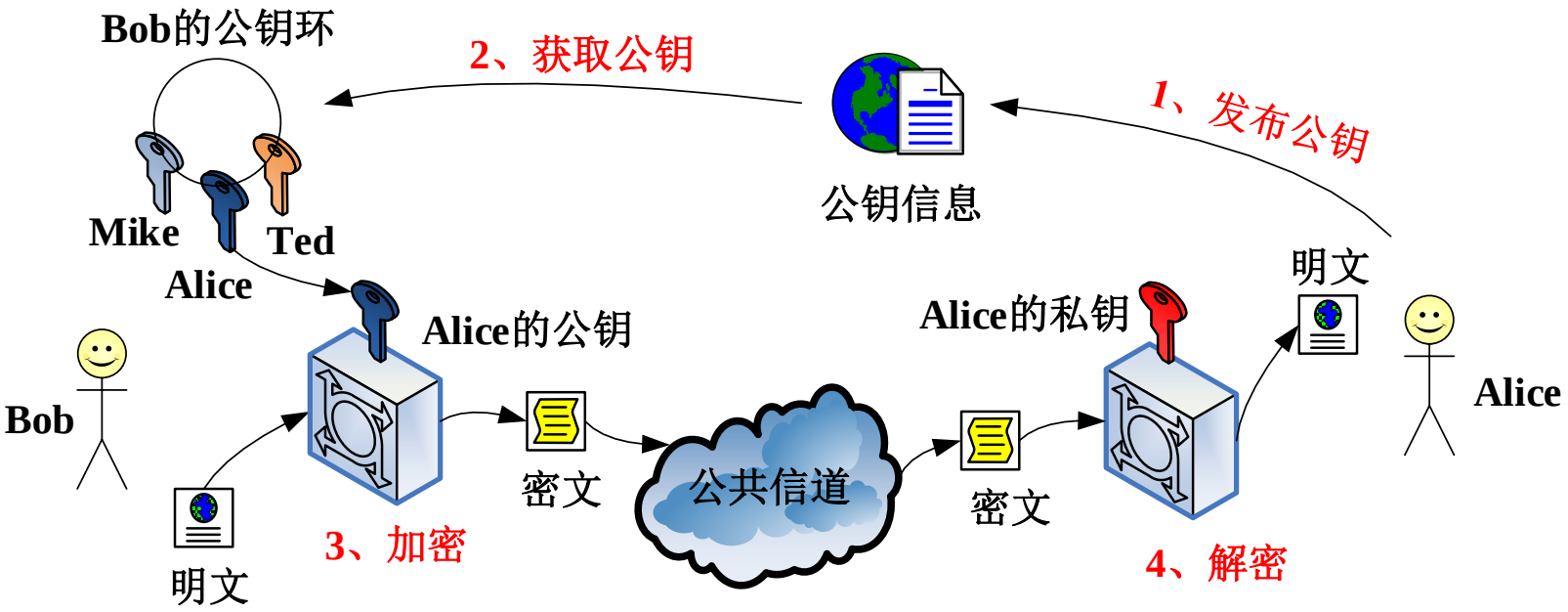
数字签名：用于认证和不可否认性

对称密钥加密算法



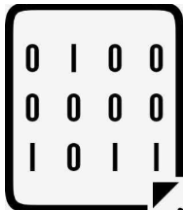
- 特点：
  - 加解密密钥相同
  - 加解密速度快
- 应用：
  - 大量数据加密
  - 消息认证码
- 常见算法：
  - ZUC，DES，AES，SM4……

公开密钥密码体制



- 特点：
  - 加解密密钥不同
  - 加解密速度慢
- 应用：
  - 短消息加密
  - 数字签名
  - 身份认证
- 常见算法：
  - RSA，ECC，SM2，ElGamal……

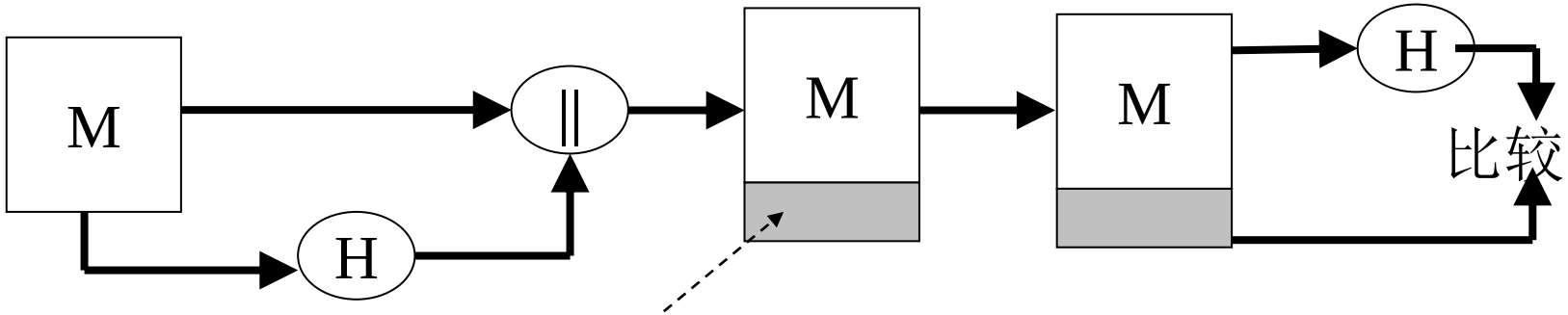
杂凑函数（Hash算法）



定长摘要值

任意长度数据

- 特点：
  - 任意长输入映射为定长输出；
  - 输入变化，输出发生不可预测的变化；
  - 输出无法推导出输入。
- 应用：
  - 完整性校验；
- 常见算法：
  - SHA-1，MD5，SM3……



Bob

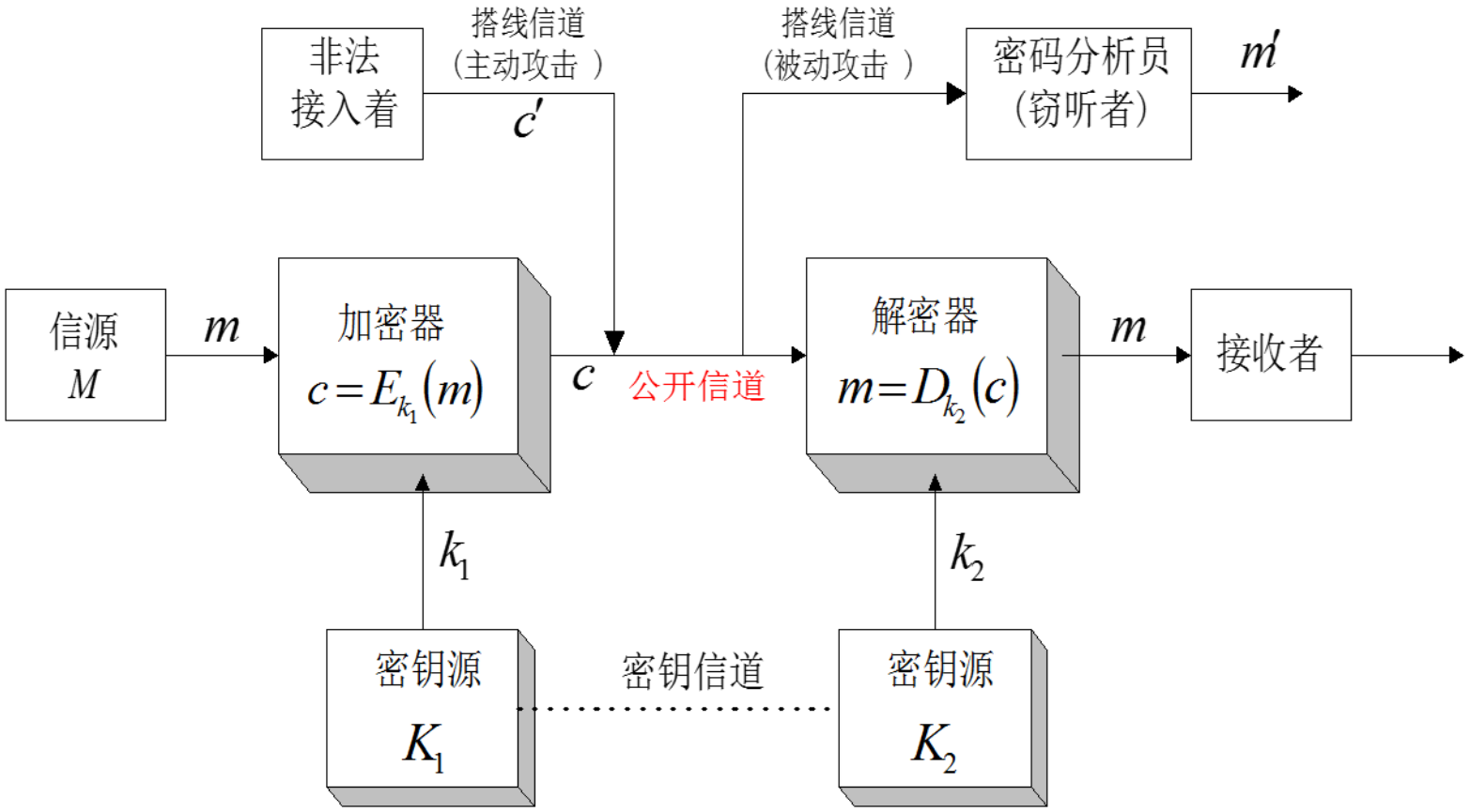
Alice

密码算法分类

密码算法分类		国产商用密码算法	国际商用密码算法
对称加密算法	分组加密	SM1, SM4, SM7	DES, IDEA, AES, RC5, RC6
	序列加密	ZUC(祖冲之算法)	RC4
非对称加密算法	大数分解		RSA, DSA, ECDSA
	离散对数	SM2, SM9	DH, DSA, ECC, ECDH
密码杂凑/散列算法		SM3	MD5, SHA-1, SHA-2

保密系统

- 一个保密系统由：明文空间、密文空间、密钥空间、加密算法、解密算法构成



保密系统应当满足的要求

为了保护信息的保密性，抗击密码分析，保密系统应当满足下述要求：

① 系统即使达不到理论上是不可破的，也应当为实际上不可破的。就是说，从截获的密文或某些已知明文密文对，要决定密钥或任意明文在计算上是不可行的。

② 系统的保密性不依赖于对加密体制或算法的保密，而依赖于密钥。这是著名的 Kerckhoff原则。

③ 加密和解密算法适用于密钥空间中的所有元素。

④ 系统便于实现和使用。

密码分析学的前提

- Kerckhoffs假设：假定密码分析者和敌手知道所使用的密码系统。即密码体制的安全性仅依赖于对密钥的保密，而不应依赖于算法的保密
  - 假设敌手知道：
  - （1）所使用的加密算法
  - （2）知道明文的概率分布规律；
  - （3）知道密钥的概率分布规律；
  - （4）知道所有可能的破译方法
  - （5）敌手能够拿到加密装置，可以对其进行能量消耗分析等等

一切秘密皆蕴含在密钥中！

密码分析学的目标

- 恢复密钥：针对加密和签名体制
- 恢复明文：针对加密体制
- 伪造签名：针对签名体制
- 找到碰撞：针对杂凑函数和消息认证码

密码体制的攻击方法

密码分析者攻击密码体制的方法：

（1）穷举攻击：通过试遍所有的密钥来进行破译。

对抗：可增大密钥的数量。

（2）统计分析攻击：通过分析密文和明文的统计规律来破译。

对抗：设法使明文和密文的统计规律不一样。

（3）解密变换攻击：针对加密变换的数学基础，通过数学求解设法找到解密变换。

对抗：选用具有坚实的数学基础和足够复杂的加密算法。

密码体制的攻击（密码破译）

攻击强度

- 唯密文攻击（Ciphertext Olny Attack）
- 已知明文攻击（Known Plaintext Attack）
- 选择明文攻击（Chosen Plaintext Attack）
- 选择密文攻击（Chosen Ciphertext Attack）

根据敌手获得的资源不同进行分类

无条件安全与计算上安全

- 无条件安全的(不可破译的)：
  - 无论截获多少密文，都没有足够信息来唯一确定明文，则该密码是无条件安全的，即对算法的破译不比猜测有优势
- 计算上安全的：
  - 使用有效资源对一个密码系统进行分析而未能破译，则该密码是强的或计算上安全的

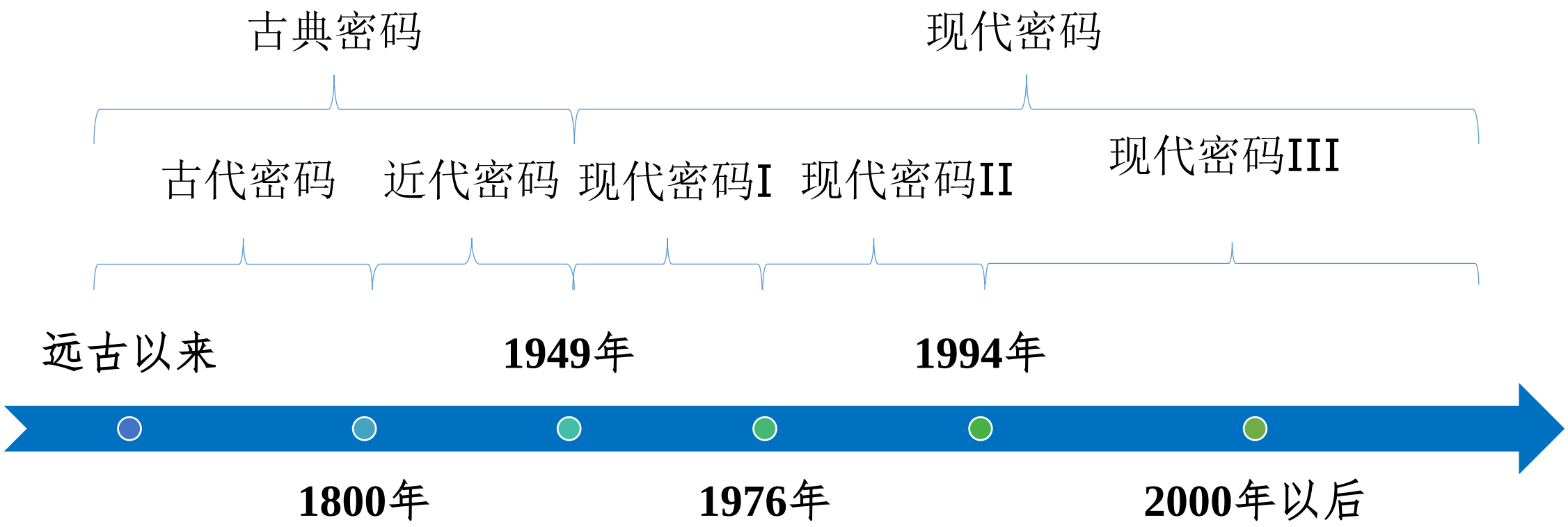
计算上安全的密码算法要满足的准则

（1） 破译密文的代价超过被加密信息的价值。

（2 ）破译密文所花的时间超过信息的有用期。

满足以上两个准则的密码算法称为计算上安全的。

密码学发展时间轴


古典密码
现代密码
古代密码
近代密码
现代密码I
现代密码II
现代密码III
远古以来
1949年
1994年
1800年
1976年
2000年以后

单表代替密码——仿射密码

- 加密函数： $y = ax + b(\mathrm{mod}\,26)$
- 密钥：a, b
- 解密函数:$x = a^{-1}(y - b)(\mathrm{mod}\,26)$
- 条件：$(a, 26) = 1$

关键在于计算 k^{-1} ：

方法：扩展的欧几里得算法

若 $(m, n) = 1$，则存在整数 k_1, k_2 使得$k_1 m + k_2 n = 1$

这里k_1就是$m^{-1} \bmod n$，

注意要将 k_1变为正数

$$-k_1 \bmod n = (n - k_1) \bmod n$$

多表替代

设 $n=2, N=26$，加密函数为

$$\begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \begin{pmatrix} 6 & 7 \\ 5 & 9 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} (\mathrm{mod} 26)$$

求解密函数。

第二章 流密码

复习要点

- 流密码的基本思想
- 伪随机序列要满足的条件
- 线性反馈移位寄存器序列的基本概念
  - m序列的定义
  - m序列的伪随机性

流密码概况

- 流密码(stream cipher)是一种重要的密码体制
  - 明文消息按字符或比特逐位加密
  - 流密码也称为序列密码(Sequence Cipher)
- 流密码在20世纪50年代得到飞跃式发展
  - 密钥流可以用移位寄存器电路来产生，也促进了线性和非线性移位寄存器发展
  - 流密码主要是基于硬件实现
- 常用流密码算法
  - ZUC（国产商用密码）、A5、SNOW、RC4

流密码的基本思想

- 流密码的基本思想
  - 利用密钥k产生一个密钥流$z=z_0z_1z_2\ldots$，并使用如下规则对明文串$x=x_0x_1x_2\ldots$加密：

$$y=y_0y_1y_2\ldots=E_{z_0}(x_0)E_{z_1}(x_1)\,E_{z_2}(x_2)\ldots,$$

- 密钥流
  - 由密钥流发生器 f 产生：$z_i=f(k,\sigma_i)$
  - σ_i是加密器中的记忆元件在时刻 i 的状态
  - f 是由k, σ_i 产生的函数

流密码的需求

- 一次一密密码是加法流密码的原型（也是豪密的基础）
  - 如果密钥用作滚动密钥流，则加法流密码就退化成一次一密密码。
- 密码设计者的最大愿望是设计出一个滚动密钥生成器，使得密钥经其扩展成的密钥流序列具有如下性质：
  - 极大的周期
  - 良好的统计特性
  - 抗线性分析

二元序列的伪随机性

- GF（2）上的一个无限序列

$$\underrightarrow{a} = (a_1, a_2, \ldots, a_n, \ldots)$$

称为二元序列，若 $a_i \in GF(2)$。

- 周期：对于二元序列 $\underrightarrow{a}$，如果存在正整数**l**，使得对于一切正整数**k**都有

$$a_k = a_{k+l}$$

则称 $\underrightarrow{a}$ 是周期的。

满足上述条件的最小正整数称为 $\underrightarrow{a}$ 的周期记为 $p(\underrightarrow{a})$

游程的定义

设 $\underset{\sim}{a}$ 是 GF（2）上周期为 $p(\underset{\sim}{a})$ 的周期序列。将 $\underset{\sim}{a}$ 的一个周期

$$(a_1, a_2, \ldots, a_{p(\underset{\sim}{a})})$$

依次排列在一个圆周上使 $a_{p(\underset{\sim}{a})}$ 与 a_1 相连，把这个圆周上形如

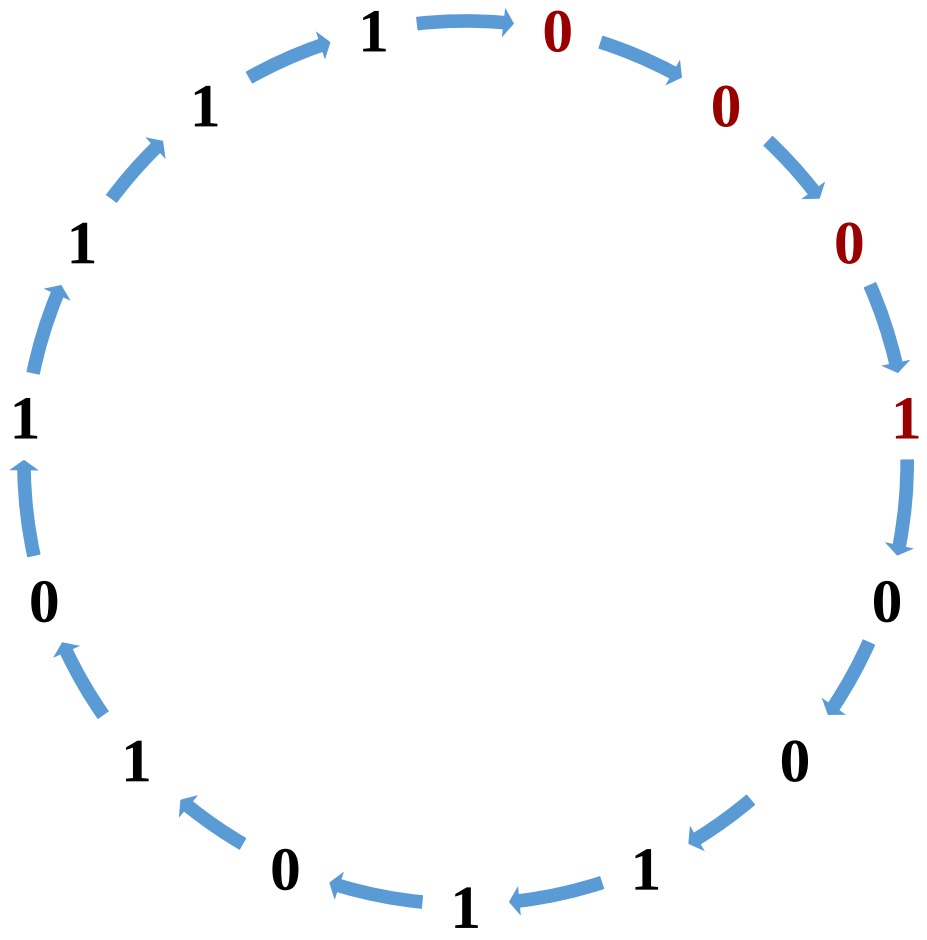
$$0\underbrace{11\cdots11}_{\text{都是1}}0 \text{ 或 } 1\underbrace{00\cdots00}_{\text{都是0}}1$$

的一连串两两相邻的项分别称为 $\underset{\sim}{a}$ 的一个周期中一个 1 游程或一个 0 游程。而 1 游程中 1 的个数或 0 游程中 0 的个数称为游程的长度。

游程的例子

周期为15的二元序列
100010011010111

011110为1的4游程
10001为0的3游程



自相关函数

GF(2)上周期为T的序列$\{a_i\}$的自相关函数定义为

$$R(\tau) = (1/T)\sum_{k=1}^{T}(-1)^{a_k}(-1)^{a_{k+\tau}} \quad 0 \le \tau \le T-1$$

当τ=0时，R(τ)=1；当τ≠0时，称R(τ)为异相自相关函数。

Golomb伪随机公设

3个随机性公设：

① 在序列的一个周期内，0与1的个数相差至多为1。

- 说明$\{a_i\}$中0与1出现的概率基本上相同

② 在序列的一个周期内，长为i的游程占游程总数的$1/2^i$ (i=1,2,…)，且在等长的游程中0的游程个数和1的游程个数相等。

- 说明0与1在序列中每一位置上出现的概率相同

③ 异相自相关函数是一个常数。

- 意味着通过对序列与其平移后的序列做比较，不能给出其他任何信息

伪随机序列还应满足的条件

C1．周期p要足够大，如大于10^{50}；

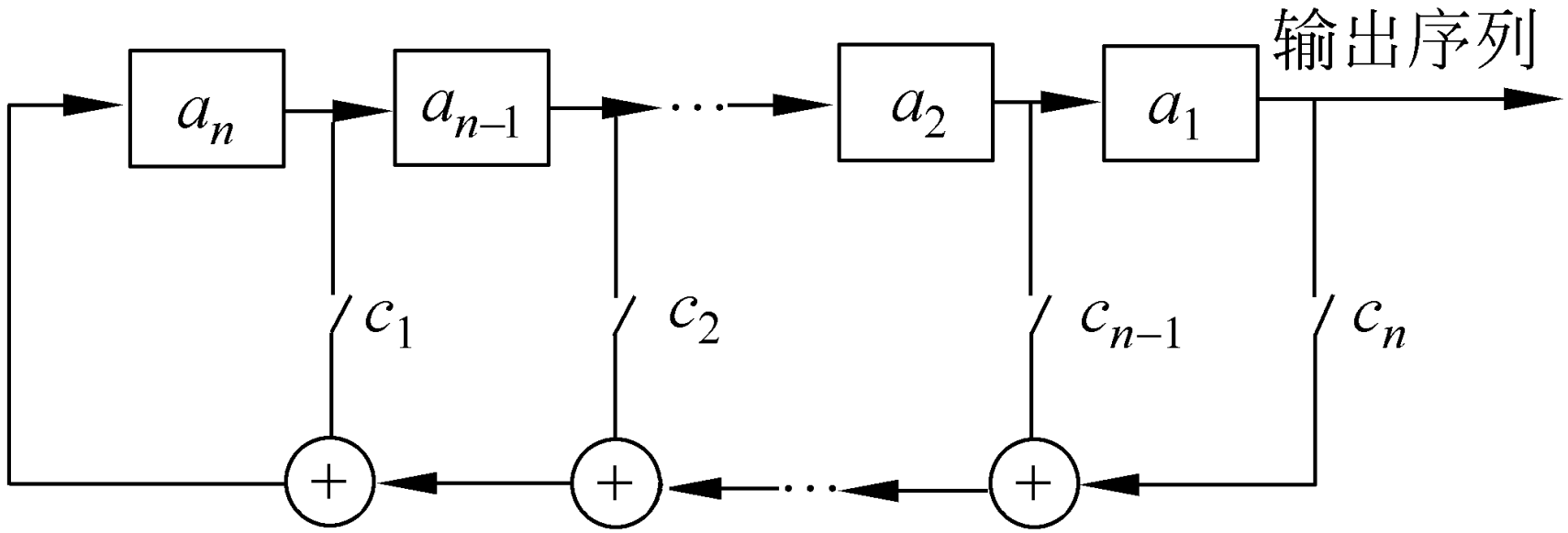
C2．序列$\{a_i\}_{i\geq 1}$产生易于高速生成；

C3．当序列$\{a_i\}_{i\geq 1}$的任何部分暴露时，要分析整个序列，提取产生它的电路结构信息，在计算上是不可行的，称此为不可预测性。

C3决定了密码的强度，是流密码理论的核心。它包含了流密码要研究的许多主要问题，如线性复杂度、相关免疫性、不可预测性等等。

线性反馈移位寄存器LFSR（linear feedback shift register）

GF(2)上的n级线性反馈移位寄存器



$$f(a_1, a_2, \cdots, a_n) = c_1 a_n \oplus c_2 a_{n-1} \oplus \cdots \oplus c_n a_1$$

LFSR的反馈函数

输出序列$\{a_t\}$满足：

$$f(a_a, a_2, \ldots, a_n) = c_1 a_n \oplus c_2 a_{n-1} \oplus \cdots \oplus c_n a_1$$

$$a_{n+1} = c_1 a_n \oplus c_2 a_{n-1} \oplus \cdots \oplus c_n a_1$$

$$a_{n+2} = c_1 a_{n+1} \oplus c_2 a_n \oplus \cdots \oplus c_n a_2$$

$$\cdots\cdots$$

$$a_{n+t} = c_1 a_{n+t-1} \oplus c_2 a_{n+t-2} \oplus \cdots \oplus c_n a_t,\ t = 1, 2, \ldots$$

线性反馈移位寄存器：实现简单、速度快、有较为成熟的理论，成为构造密钥流生成器的最重要的部件之一。

LFSR的性质

总是假定$c_1,c_2,\ldots,c_n$中至少有一个不为0，否则$f(a_1,a_2,\ldots,a_n)\equiv 0$。

总是假定$c_n=1$。

- **LFSR输出序列的性质：**完全由其反馈函数决定。
- **n级LFSR状态数：**最多有2^n个
- **n级LFSR的状态周期：**$\leq 2^n-1$
- 输出序列的周期=状态周期，$\leq 2^n-1$

- 选择合适的反馈函数可使序列的周期达到最大值2^n-1，周期达到最大值的序列称为m序列。

线性移位寄存器的一元多项式表示

定义2.1 设n级线性移位寄存器的输出序列满足递推关系

$$a_{n+k}=c_1a_{n+k-1}\oplus c_2a_{n+k-2}\oplus \dots \oplus c_na_k \qquad (*)$$

用延迟算子 $D(Da_k=a_{k-1})$ 作为未定元，给出的反馈多项式为：

$$p(D)=1+c_1D+\dots+c_{n-1}D^{n-1}+c_nD^n$$

这种递推关系可用一个一元高次多项式

$$p(x)=1+c_1x+\dots+c_{n-1}x^{n-1}+c_nx^n$$

表示，称这个多项式为**LFSR的特征多项式。**

m-序列产生的充要条件

定义3.4 若n次不可约多项式p(x)的阶为2^n-1，则称p(x)是n次本原多项式。

定理3.6 设$\{a_i\}\in G(p(x))$，$\{a_i\}$为m序列的充要条件是p(x)为本原多项式。

对于任意的正整数n，至少存在一个n次本原多项式。所以对于任意的n级LFSR，至少存在一种连接方式使其输出序列为m序列

m序列的随机性

m序列满足Golomb的3个随机性公设。

定理3.7 GF(2)上的n长m序列$\{a_i\}$具有如下性质：

① 在一个周期内，0、1出现的次数分别为$2^{n-1}-1$和2^{n-1}。

② 在一个周期内，总游程数为2^{n-1}；对$1\le i\le n-2$，长为i的游程有2^{n-i-1}个，且0、1游程各半；长为n-1的0游程一个，长为n的1游程一个。

③ $\{a_i\}$的自相关函数为

$$R(\tau)=\begin{cases}1, & \tau=0\\ -\frac{1}{2^n-1}, & 0<\tau\le 2^n-2\end{cases}$$

m-序列的安全性

- 寻找m序列的递推关系式。
  - 已知一段序列，如果知道其反馈多项式，就可以将其后的序列依次求出
  - 已知序列能否获得相应的反馈多项式(或线性递推式)呢？
    - 解方程方法——已知序列$\{a_i\}$是由**n 级**线性移存器产生的，并且知道$\{a_i\}$的**连续2n**位，可用解线性方程组的方法得到反馈多项式
    - 线性反馈移位寄存器综合解——**Berlekamp-Massey**算法

例1 设序列$\underline{a}=(01111000\ldots)$是由4级线性移存器所产生序列的连续8个信号，求该移存器的线性递推式。

解：设该4级移存器的线性递推式为：

$$a_n = c_1 a_{n-1} \oplus c_2 a_{n-2} \oplus c_3 a_{n-3} \oplus c_4 a_{n-4} \quad (n \geq 4)$$

由于知道周期序列的连续8个信号，不妨设为开头的8个信号，即

$$a_0 a_1 a_2 a_3 a_4 a_5 a_6 a_7 = 01111000$$

当n=4时，由递归式可得：$a_4 = c_1 a_3 \oplus c_2 a_2 \oplus c_3 a_1 \oplus c_4 a_0$

即：$c_1 \oplus c_2 \oplus c_3 = 1 \quad (1)$

同理可得：$c_1 \oplus c_2 \oplus c_3 \oplus c_4 = 0 \quad (2)$

$$c_2 \oplus c_3 \oplus c_4 = 0 \quad (3)$$

$$c_3 \oplus c_4 = 0 \quad (4)$$

解方程组得 $c_1 = 0, c_2 = 0, c_3 = 1, c_4 = 1$

故所求移存器递推式为：$a_n = a_{n-3} \oplus a_{n-4} \ (n \geq 4)$

密钥流生成器的设计原则

设计一个性能良好的序列密码是一项十分困难的任务。最基本的设计原则是“密钥流生成器的不可预测性”，它可分解为下述基本原则：

① 长周期。

② 高线性复杂度。

③ 统计性能良好。

④ 足够的“混乱”。

⑤ 足够的“扩散”。

⑥ 抵抗不同形式的攻击。

第三章 分组密码

复习要点

- 分组密码的设计准则
- 分组密码的分类
- 分组密码的四种工作模式
- DES算法的主要原理及参数
- AES算法的主要原理及参数
- SMS4算法参数设置

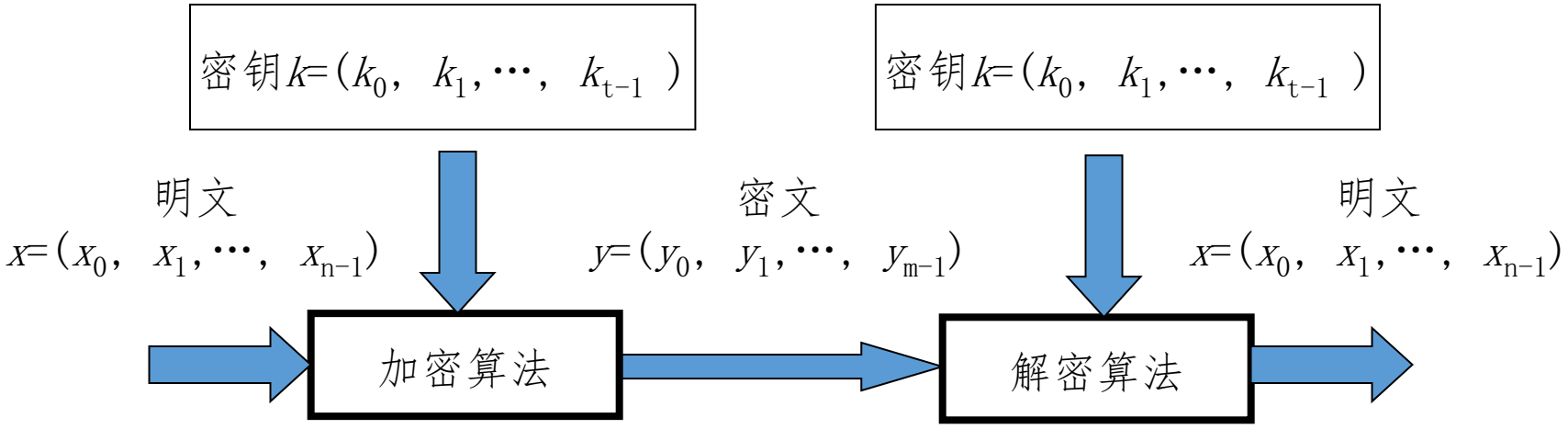
分组密码概述

明文序列 $x_0, x_1, \ldots, x_{n-1}, \ldots$， 分组长度为n

加密函数 E: $V_n \times K \rightarrow V_m$

密文分组长度为m

这种密码实质上是字长为n的数字序列的代换密码。



安全性设计原则

- **1. 混淆原则(Confusion)**
- 混淆原则就是将密文、明文、密钥三者之间的统计关系和代数关系变得尽可能复杂，使得敌手即使获得了密文和明文，也无法求出密钥的任何信息；即使获得了密文和明文的统计规律，也无法求出明文的新的信息。
- 可进一步理解为：
- （1）明文不能由已知的明文，密文及少许密钥比特代数地或统计地表示出来。
- （2）密钥不能由已知的明文，密文及少许密钥比特代数地或统计地表示出来。

安全性设计原则

- **2. 扩散原则(Diffusion)**
- 扩散原则就是应将明文的统计规律和结构规律散射到相当长的一段统计中去(Shannon的原话)。
- 也就是说让明文中的每一位影响密文中的尽可能多的位，或者说让密文中的每一位都受到明文中的尽可能多位的影响。
- 如果当明文变化一个比特时,密文有某些比特不可能发生变化,则这个明文就与那些密文无关,因而在攻击这个明文比比特时就可不利用那些密文比特.

分组密码算法应满足的要求

- 分组长度n要足够大：

  防止明文穷举攻击法奏效。

- 密钥量要足够大：

  尽可能消除弱密钥并使所有密钥同等地好，以防止密钥穷举攻击奏效。

- 由密钥确定置换的算法要足够复杂：

  充分实现明文与密钥的扩散和混淆，没有简单的关系可循，要能抗击各种已知的攻击。

分组密码算法应满足的要求

- 加密和解密运算简单：

  易于软件和硬件高速实现。

- 数据扩展：

  一般无数据扩展，在采用同态置换和随机化加密技术时可引入数据扩展。

- 差错传播尽可能地小。

  一个密文分组的错误尽可能少的影响其他密文分组的解密

SP型分组密码的加密思想

SP型分组密码的加密思想为：设x是待加密的明文，长度为$n$bit，令$X_0=x$，对于$1\le i\le r$，在子密钥K_i的控制下，对X_{i-1}做代换S，然后再做置换或可逆的线性变换P，密文为$y=X$，r是圈变换的迭代次数。

SP型分组密码的圈变换如图3.4所示。在SP型分组密码中，代换S一般被称为混淆层，主要起混淆的作用；置换或可逆的线性变换P一般被称为扩散层，主要起扩散的作用。

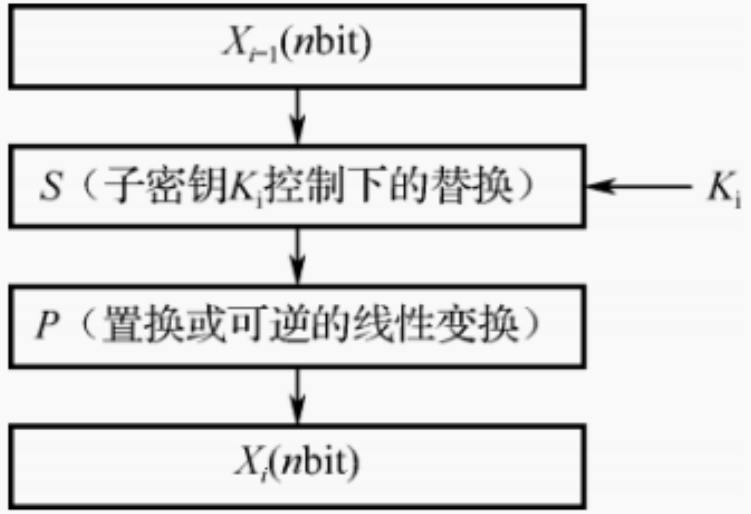


图 3.4　SP 型分组密码的圈变换

Feistel加密结构

- 输入是分组长为$2w$的明文和一个密钥K。将每组明文分成左右两半L_0和R_0，在进行完n轮迭代后，左右两半再合并到一起以产生密文分组。第i轮迭代的输入为前一轮输出的函数：

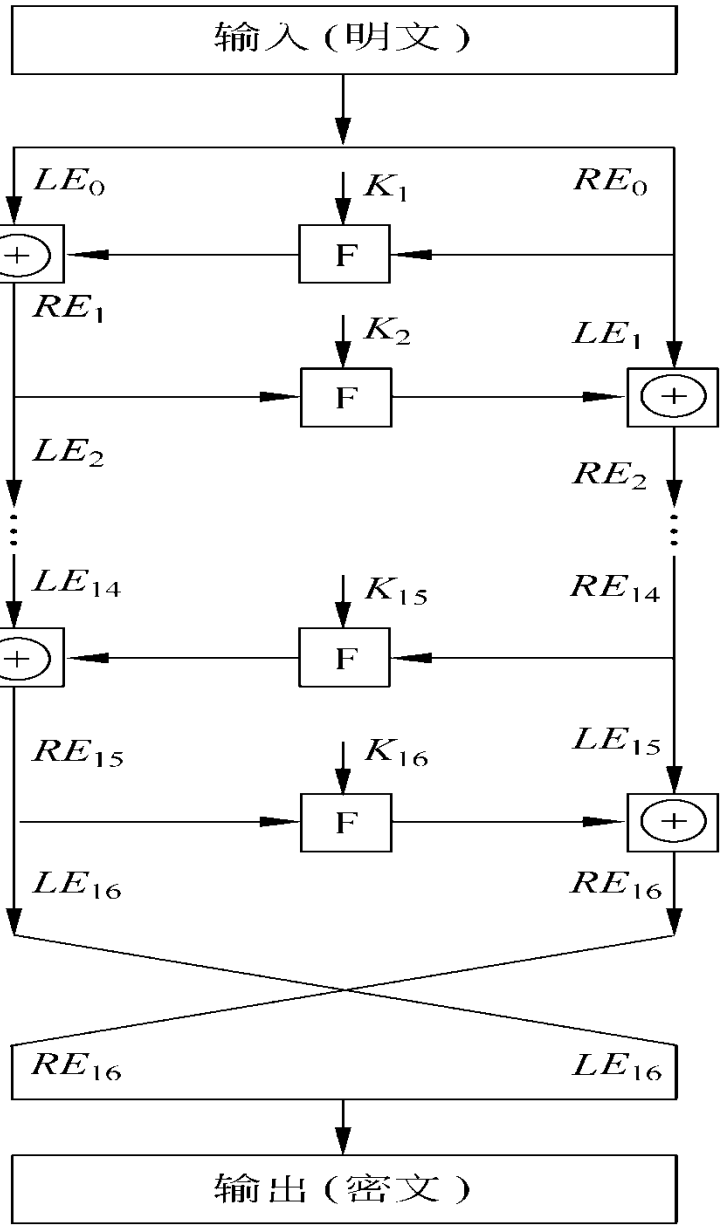
$$L_i = R_{i-1}$$

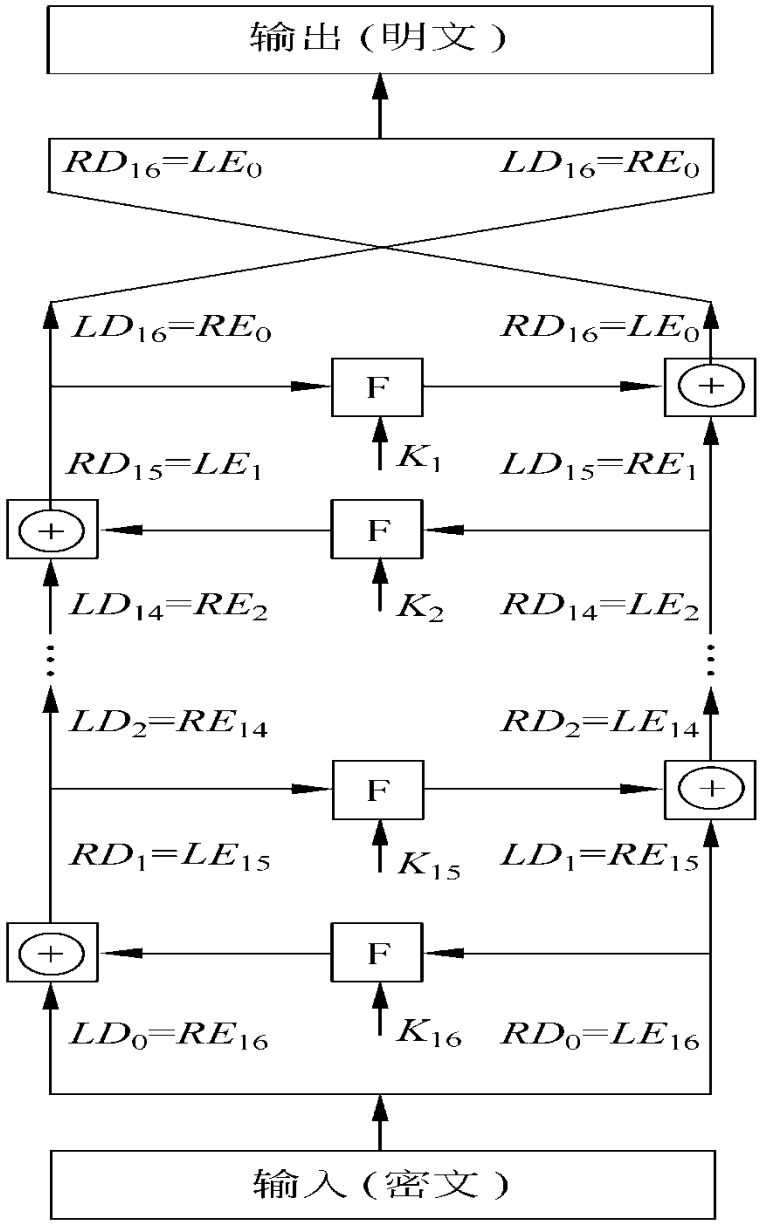
$$R_i = L_{i-1} \oplus F\left(R_{i-1},\ K_i\right)$$

- 其中K_i是第i轮用的子密钥，由加密密钥K得到。一般地，各轮子密钥彼此不同而且与K也不同。

Feistel解密结构

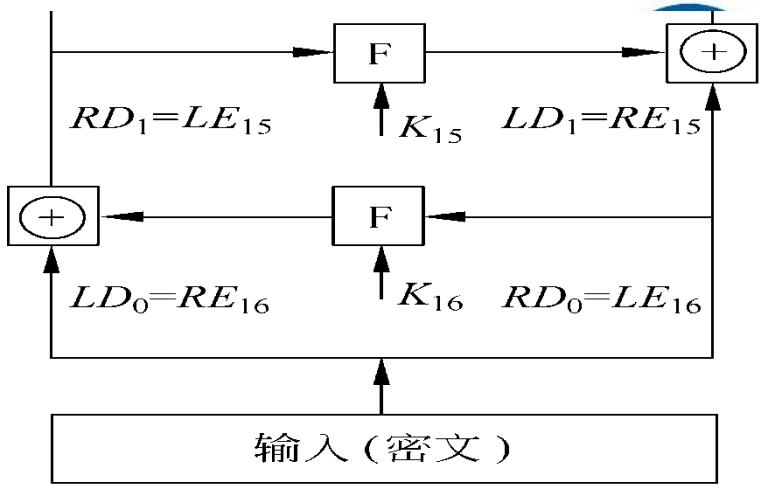
- Feistel解密过程本质上和加密过程是一样的，算法使用密文作为输入
- 但使用子密钥K_i的次序与加密过程相反，即第1轮使用K_n，第2轮使用K_{n-1}，……，最后一轮使用K_1。这一特性保证了解密和加密可采用同一算法。





Feistel加解密过程

Feistel密码解密的正确性



在加密过程中：

$$LE_{16} = RE_{15}$$

$$RE_{16} = LE_{15} \oplus F\left(RE_{15},\ K_{16}\right)$$

在解密过程中

$$LD_1 = RD_0 = LE_{16} = RE_{15}$$

$$RD_1 = LD_0 \oplus F\left(RD_0,\ K_{16}\right) =$$

$$= \left[LE_{15} \oplus F\left(RE_{15},\ K_{16}\right)\right] \oplus F\left(RE_{15},\ K_{16}\right)$$

$$= LE_{15}$$

所以解密过程第1轮的输出为$\mathrm{LE}_{15}\|\mathrm{RE}_{15}$，等于加密过程第16轮输入左右两半交换后的结果。

Feistel密码解密的正确性（续）

- 容易证明这种对应关系在16轮中每轮都成立。一般地，加密过程的第i轮有

$$LE_i = RE_{i-1}$$

$$RE_i = LE_{i-1} \oplus F\left(RE_{i-1},\ K_i\right)$$

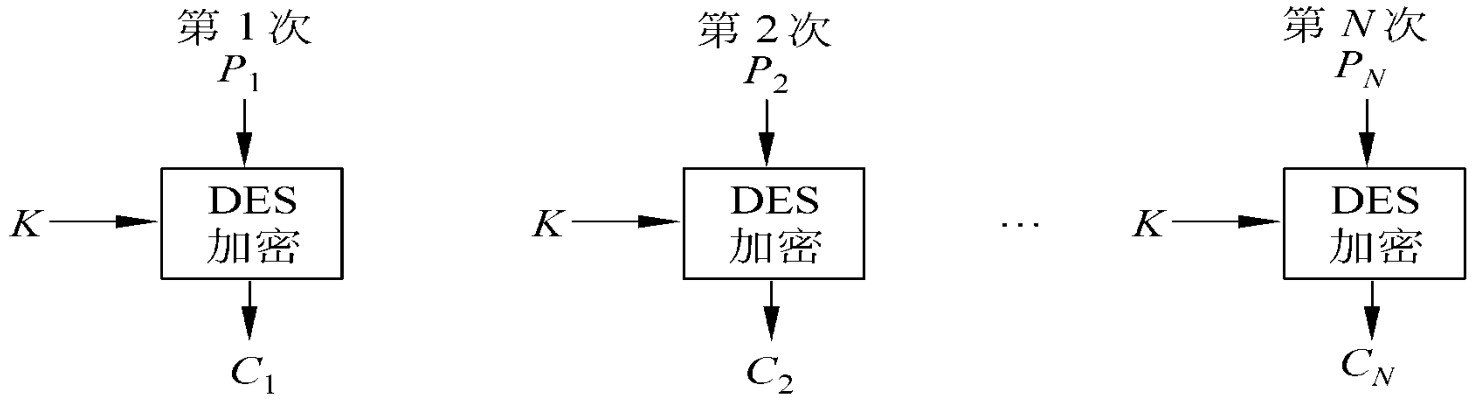
因此

$$\begin{aligned} LD_{16-i} &= RE_i \\ RD_{16-i} &= LE_i \end{aligned} \Rightarrow \begin{aligned} LD_{16-i+1} &= RD_{16-i} = LE_i = RE_{i-1} \\ RD_{16-i+1} &= LD_{16-i} \oplus F(RD_{16-i}, k_{16-(16-i)}) \\ &= RE_i \oplus F(LE_i, k_i) \\ &= LE_{i-1} \oplus F(RE_{i-1}, k_i) \oplus F(RE_{i-1}, k_i) \\ &= LE_{i-1} \end{aligned}$$

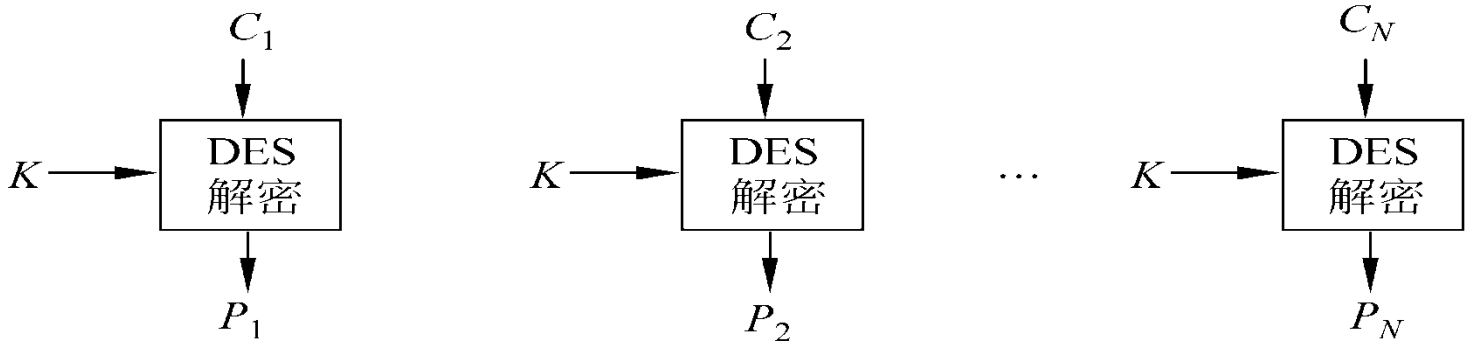
分组密码的主要工作模式

1. 电码本(ECB)模式
2. 密码分组链接(CBC)模式
3. 密码反馈(CFB)模式
4. 输出反馈(OFB)模式
5. 计数器模式

电码本ECB（Electronic Code Book）模式



(a) 加密



(b) 解密

ECB模式的优、缺点

- 优点：
  - （1）实现简单;
  - （2）不同明文分组的加密可并行实施，尤其是硬件实现时速度很快
- 缺点：
  - （1）相同明文分组对应相同密文分组
  - （2）不能隐蔽明文分组的统计规律和结构规律,不能抵抗替换攻击
- 应用：
  - （1）用于随机数的加密保护
  - （2）用于单分组明文的加密

密码分组链接CBC（Cipher Block Chaining）模式

- 这种模式先将明文分组与上一次的密文块进行按比特异或，然后再进行加密处理。这种模式必须选择一个初始向量c_0=IV，用于加密第一块明文。
- 加密过程为

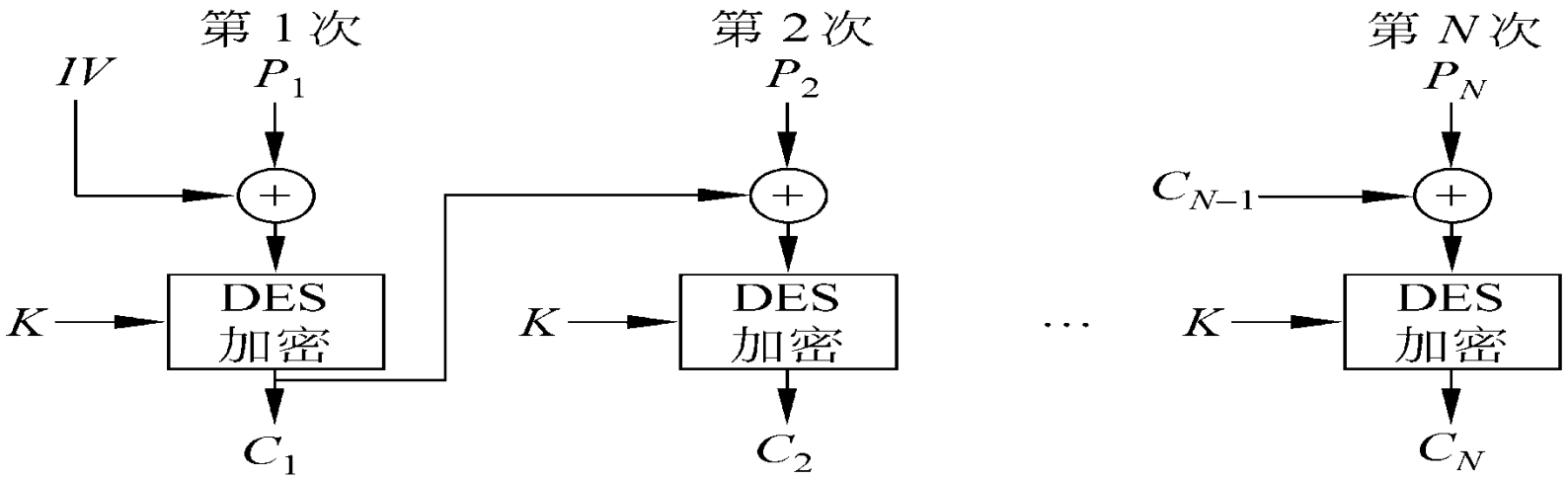
$$C_i = E_K(P_i \oplus C_{i-1})$$

- 解密过程为

$$P_i = D_K(C_i) \oplus C_{i-1}$$

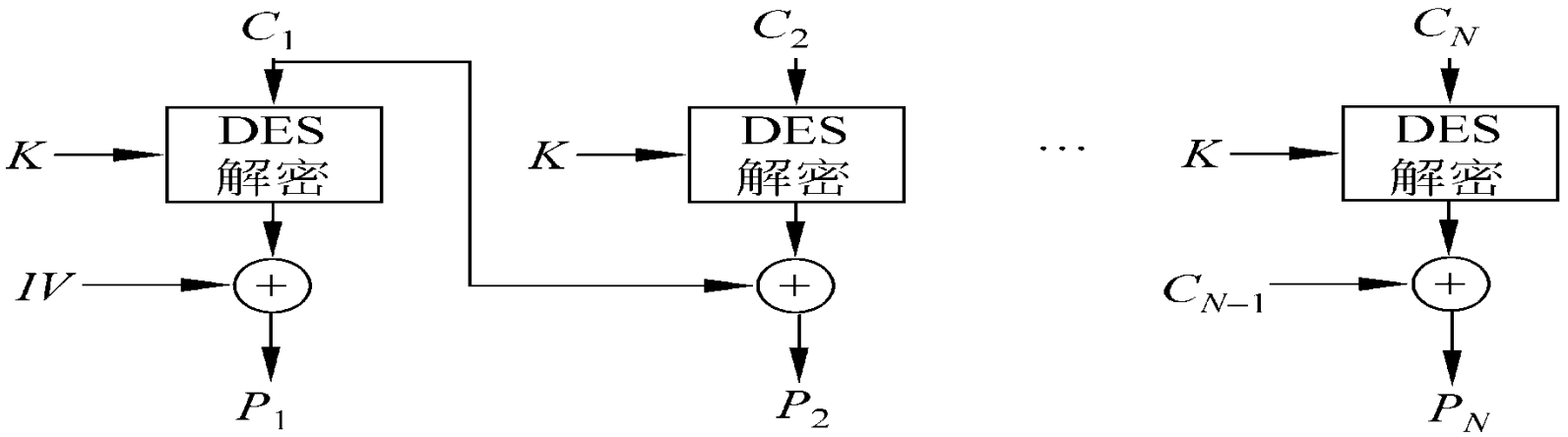
密码分组链接CBC（Cipher Block Chaining）模式

$$C_i = E_K(P_i \oplus C_{i-1})$$



(a) 加密

$$P_i = D_K(C_i) \oplus C_{i-1}$$



(b) 解密

CBC模式的特点

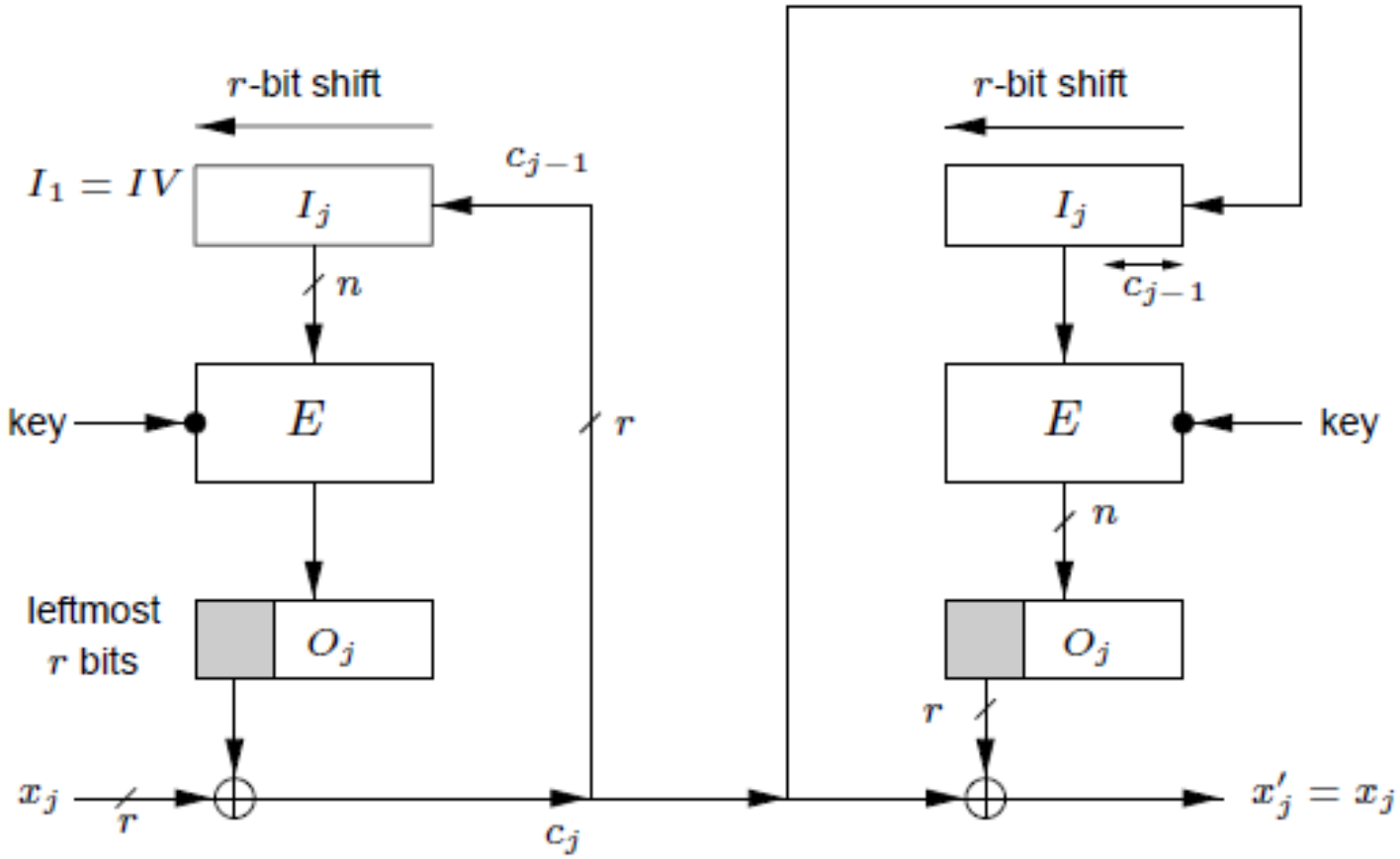
1. 明文块的统计特性得到了隐蔽
   - 由于在CBC模式中，各密文块不仅与当前明文块有关，而且还与以前的明文块及初始化向量有关，从而使明文的统计规律在密文中得到了较好的隐藏
2. 具有有限的错误传播特性
   - 一个密文块的错误将导致两个密文块不能正确解密
3. 具有自同步功能
   - 密文出现丢块和错块不影响后续密文块的解密.若从第t块起密文块正确，则第t+1个明文块就能正确求出

密码反馈CFB（Cipher Feedback）模式

- 若待加密消息需按字符、字节或比特处理时，可采用**CFB**模式。并称待加密消息按 ***r*** 比特处理的**CFB**模式为 ***r*** 比特CFB模式。
- 适用范围:
  - 适用于每次处理 ***r***比特明文块的特定需求的加密情形,能灵活适应数据各格式的需要
  - 例如,数据库加密要求加密时不能改变明文的字节长度,这时就要以明文字节为单位进行加密

CFB的加密解密

- 若记$IV=c_{-l+1}\ldots c_{-1}\,c_0$，$|c_i|=r$，则加密过程可表示为：$c_i = x_i \oplus left_r(E_k(c_{i-l}\cdots c_{i-2}c_{i-1}))$



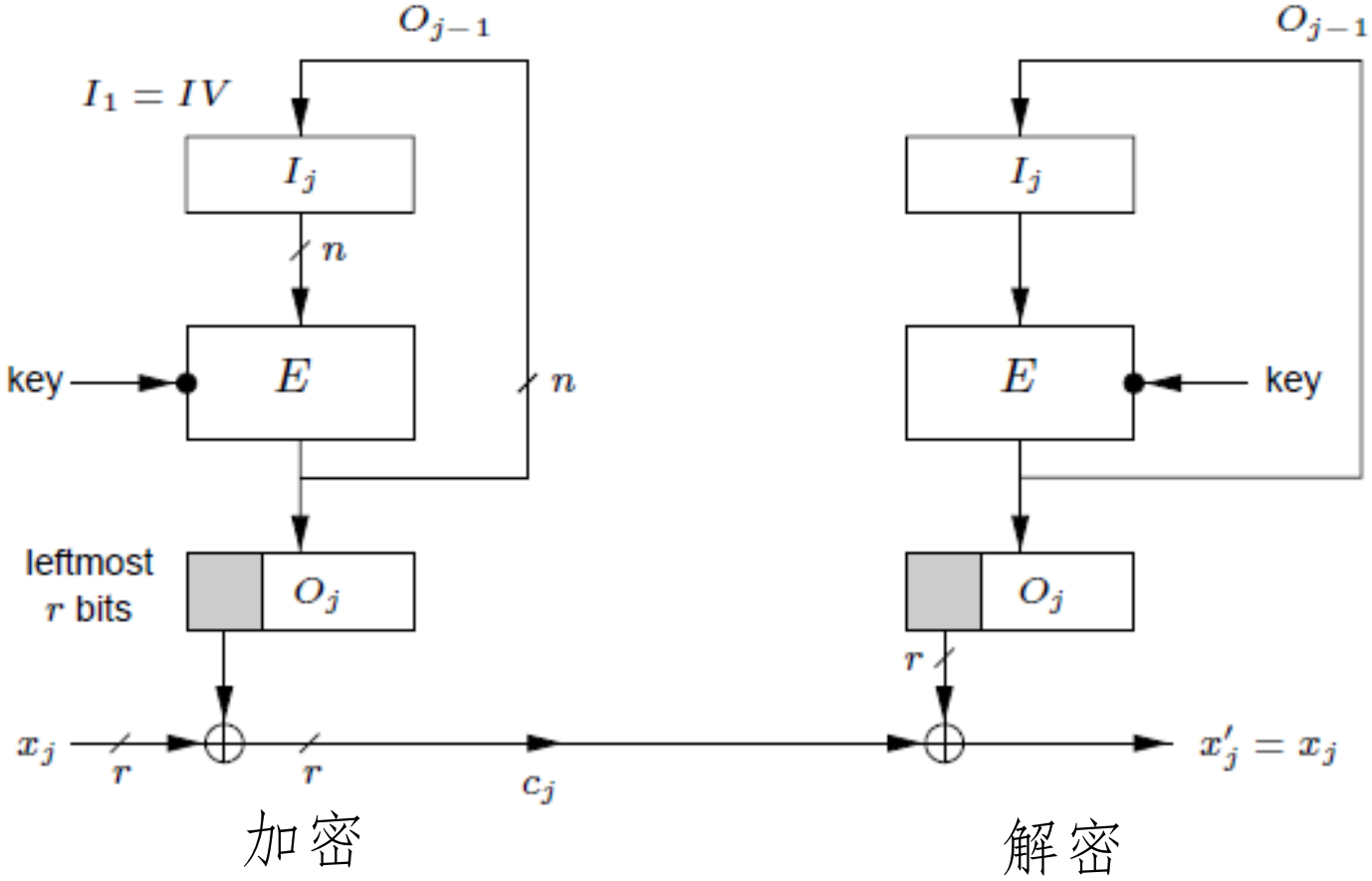
加密　　解密

CFB模式的特点

- 相同明文：
  - 和按**CBC**模式加密一样，改变***IV***同样会导致相同的明文输入得到不同的加密输出。***IV***无需保密（虽在某些应用中**IV**须是不可预测的）。
- 链接依赖性：
  - 类似**CBC**加密，链接机制致使密文组依赖于当前明文组和其前面的明文组；因此，重排密文组会影响解密。
- 错误的传播：
  - 一个或多个比特错误出现在任一个**r**比特的密文组中会影响这个组和后继 $\lceil n/r \rceil$ 个密文组的解密。
- 错误恢复：
  - **CFB**和**CBC**相似，也是自同步的，但它需有 $\lceil n/r \rceil$ 个密文组才能还原

输出反馈OFB(Output Feedback)模式

- **OFB模式在结构上类似于CFB模式，但反馈的内容是DES的输出而不是密文！**



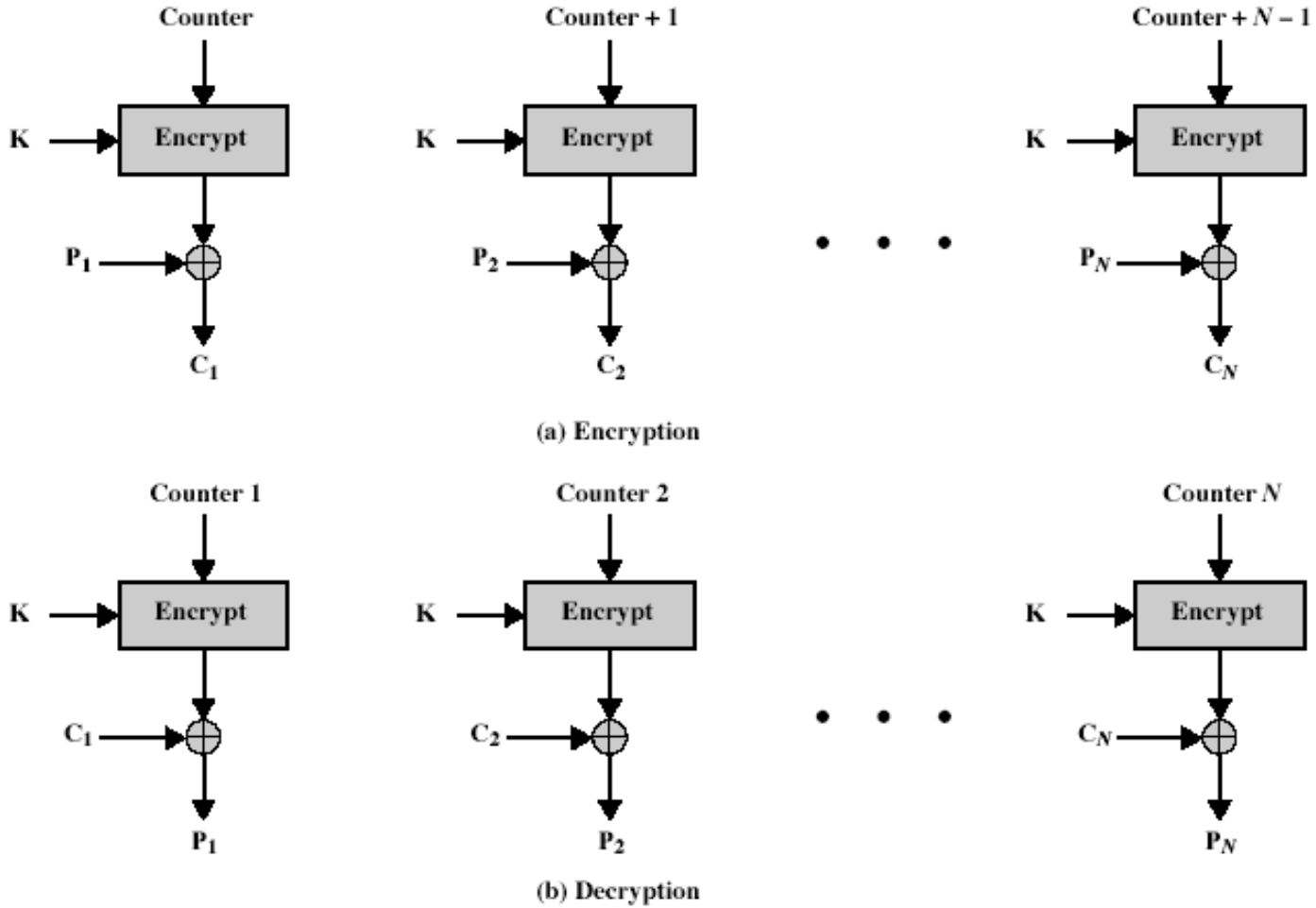
OFB工作模式的特点

- 相同明文：和**CBC**及**CFB**一样，改变***IV***同样会导致相同的明文输入得到不同的加密输出。
- 链接依赖性：密钥流是独立于明文的。
- 错误传播：有一个或多个比特错误的任一密文字符仅会影响该字符的解密，密文字符的某比特位置出错将致使还原明文的相应位置也出错。
- 错误恢复：**OFB**模式能从密文比特错误中得以恢复，但在丢失密文比特后就无法实现自同步了，这是因为丢失密文比特会破坏密钥流的编排。

计数器模式

- 利用固定密钥**k**对自然数序列**1,2,3,…,n, …**加密，将得到的密文分组序列看作密钥流序列，按加法密码的方式与明文分组逐位异或的一种方式
- 利用这种方式可以产生伪随机数序列,其伪随机特性远比计算机产生的随机数的性质好

计数器模式的结构


Counter
Counter + 1
Counter + N – 1
K
Encrypt
P1
P2
PN
C1
C2
CN
(a) Encryption
Counter 1
Counter 2
Counter N
(b) Decryption

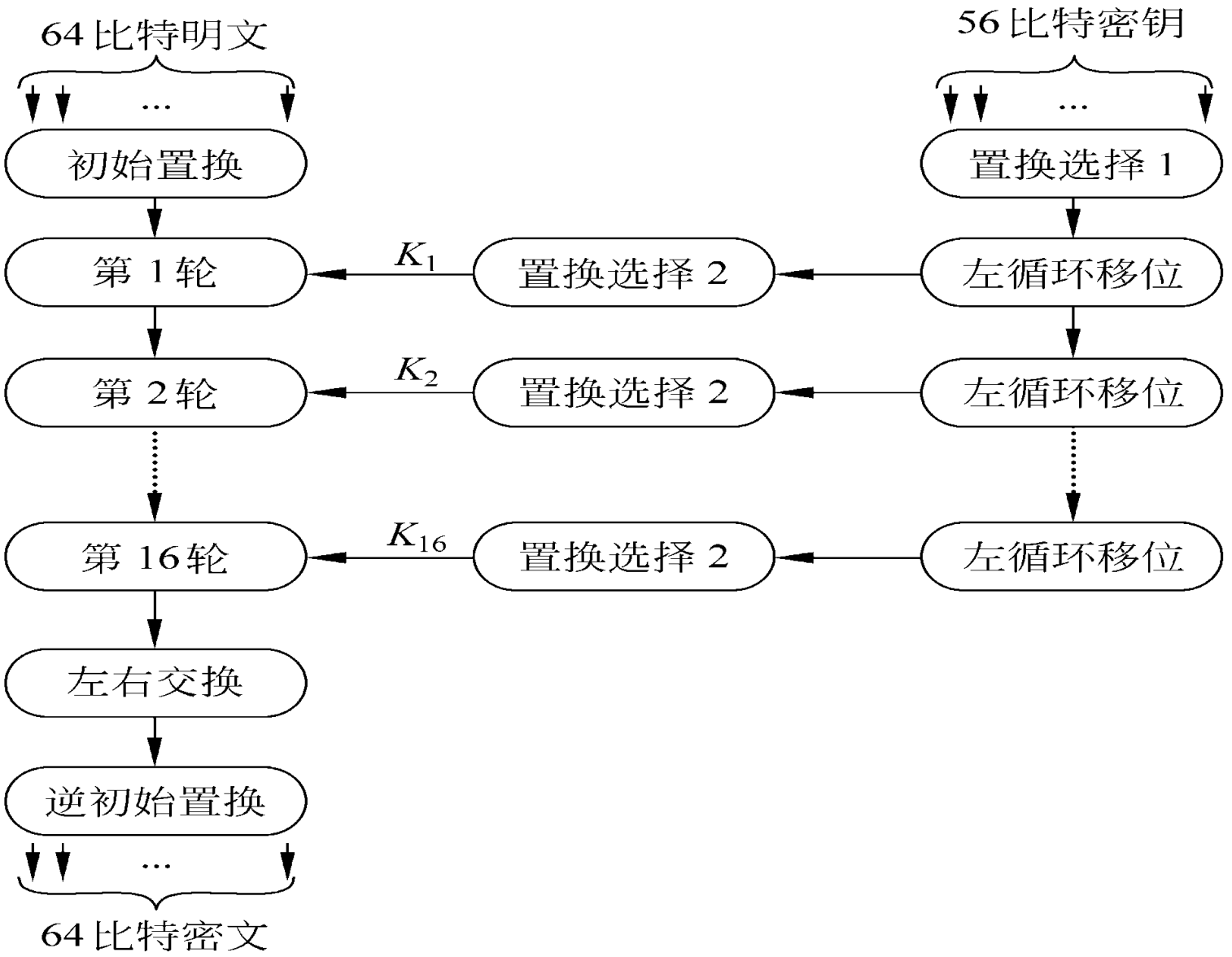
CTR的优点

- 效率
  - 可并行加密
  - 预处理
  - 吞吐量仅受可使用并行数量的限制
- 加密数据块的随机访问
- 可证明安全
- 简单性（只要求实现加密算法）

DES 算法

- 分组长度为64 bits (8 bytes)
- 密文分组长度也是64 bits。
- 密钥长度为64 bits，有8 bits奇偶校验，有效密钥长度为56 bits。
- 算法主要包括：初始置换*IP*、16轮迭代的乘积变换、逆初始置换IP^{-1}以及16个子密钥产生器。

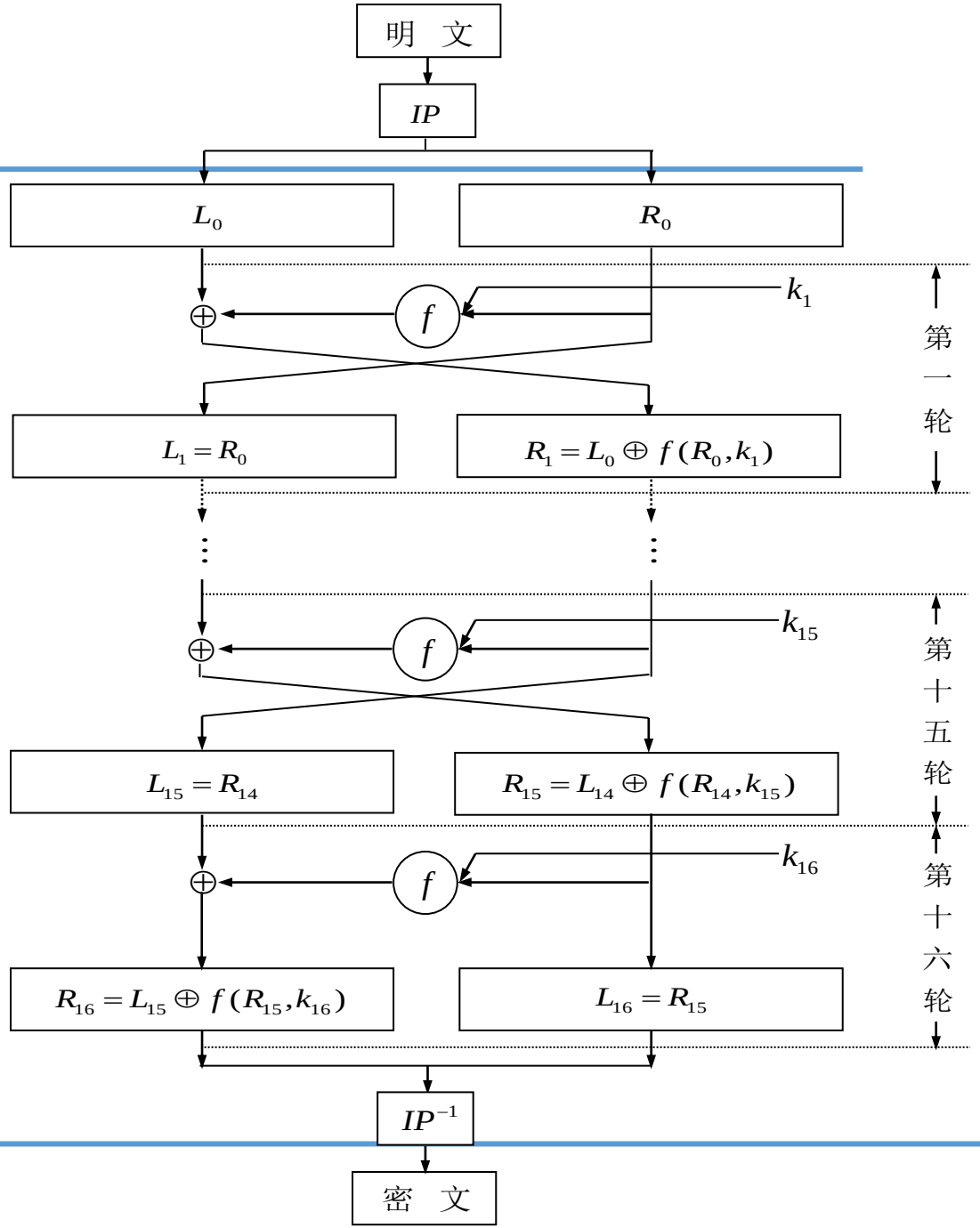
DES算法框图


64 比特明文
56 比特密钥
初始置换
置换选择 1
第 1 轮
K_1
置换选择 2
左循环移位
第 2 轮
K_2
置换选择 2
左循环移位
第 16 轮
K_{16}
置换选择 2
左循环移位
左右交换
逆初始置换
64 比特密文

DES算法流程

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

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



DES的S-盒的输入和输出关系

x_5 x_0

1 0

x_5 x_4 x_3 x_2 x_1 x_0

1 0 1 1 0 0

$(y_3, y_2, y_1, y_0)=(0,0,1,0)$

行号 \ 列号	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	2	14	10	0	6	13

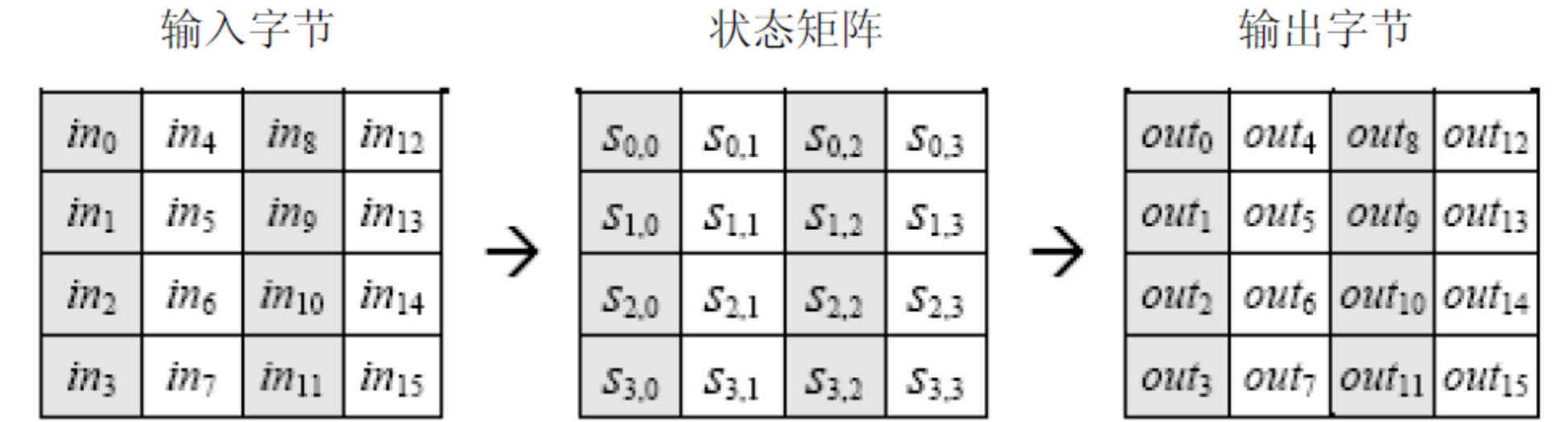
AES算法说明

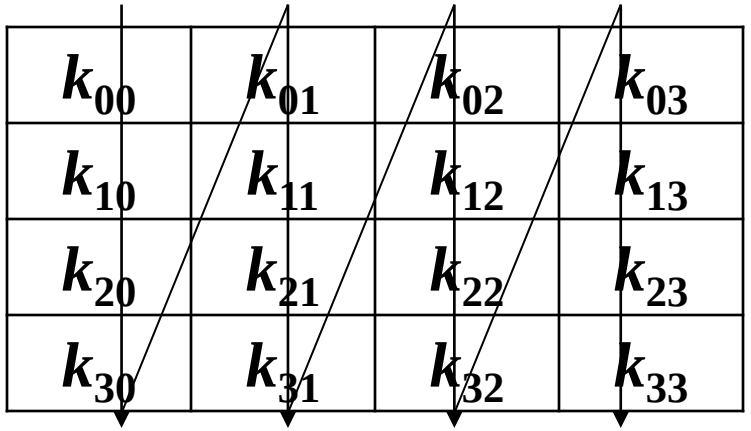
- 分组长度为128比特，密钥长度可变，为128、192、256比特。
- 状态
  - 算法中间的结果也需要分组，称之为状态，状态可以用以字节为元素的矩阵阵列表示，该阵列有4行，列数N_b为分组长度除32
- 种子密钥
  - 以字节为元素的矩阵阵列描述，阵列为4行，列数N_k为密钥长度除32

算法说明

- 算法的输入、输出和种子密钥可看成字节组成的一维数组。
- 下标范围
  - 输入输出：0—$4N_b$-1，
  - 种子密钥：0—$4N_k$-1

N_b=4和N_k=4的状态密钥阵列



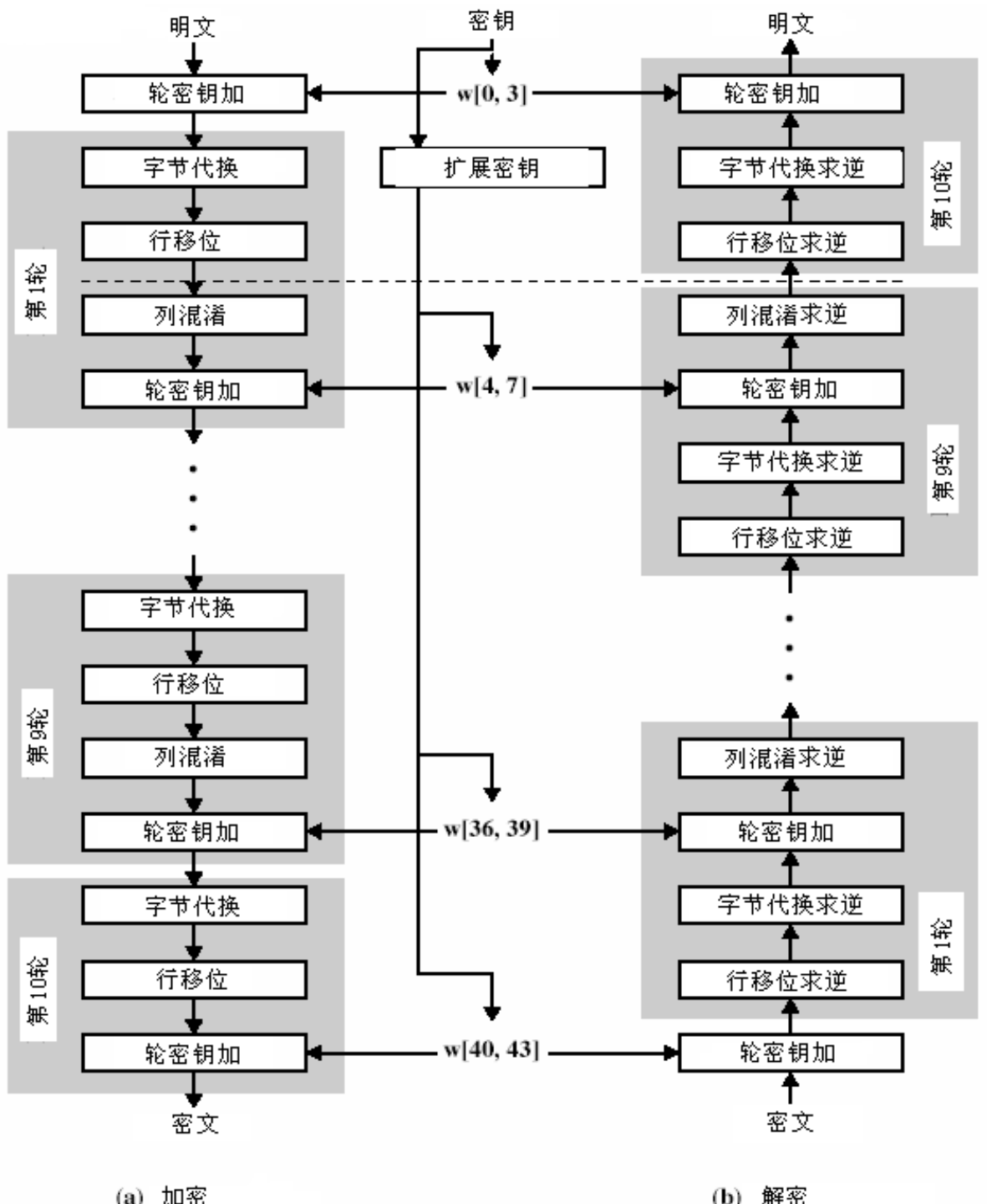


按此顺序放入

分组和阵列中元素对应关系

- 分组下标n
- 阵列位置(i,j)
  - i=n mod 4, j=[n/4];n=i+4j
- 轮数N_r与N_b和N_k对应关系

	N_b=4
N_k=4	10
N_k=6	12
N_k=8	14


明文
密钥
明文
轮密钥加
w[0, 3]
轮密钥加
扩展密钥
字节代换
行移位
列混淆
轮密钥加
第1轮
w[4, 7]
字节代换求逆
行移位求逆
列混淆求逆
轮密钥加
第10轮
第9轮
字节代换
行移位
列混淆
轮密钥加
w[36, 39]
字节代换求逆
行移位求逆
列混淆求逆
轮密钥加
第9轮
第1轮
字节代换
行移位
轮密钥加
w[40, 43]
轮密钥加
第10轮
密文
密文

(a) 加密　　(b) 解密

AES的S盒

		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

AES的S盒的使用：输入8a，输出7e，即7e=S（8a）

		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

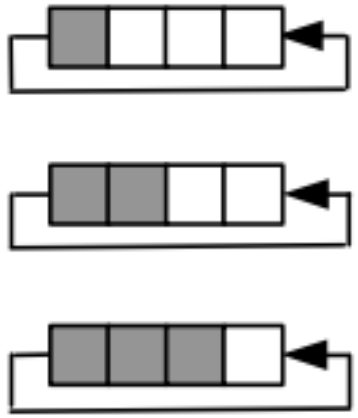
行移位

- 将状态阵列的各行进行循环移位，不同行的移位量不同
- 0行：不动
- 1行：循环左移C1字节
- 2行：循环左移C2字节
- 3行：循环左移C3字节
- 记为：ShiftRow(State)

N_b	C1	C2	C3
4	1	2	3
6	1	2	3
8	1	3	4

行移位示意图

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



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

列混合

- 将每列视为GF(2^8)上多项式，与固定的多项式c(x)进行模x^4+1乘法，记为⊗,要求c(x)模x^4+1可逆。
- 表示为MixColumn(State)

$$c(x) = '03'\,x^3 + '01'\,x^2 + '01'\,x + '02'$$

列混合的矩阵表示

列混合运算也可写为矩阵乘法。设 $b(x)=c(x)\otimes a(x)$，则

$$\begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{pmatrix} = \begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{pmatrix}$$

SMS4概况

- SMS4分组密码算法是国家密码管理局于2006年1月6日公布的无线局域网产品使用的密码算法，是国内官方公布的第一个商用密码算法。
- SMS4是一个分组密码算法，分组长度和密钥长度均为128比特。加密算法与密钥扩展算法都采用32轮非线性迭代结构。
  - 非线性变换中所使用的S盒是一个具有很好密码学特性的、由8比特输入产生8比特输出的置换，在设计原理上，SMS4比AES的S盒设计多了一个仿射变换。
  - SMS4有很高的灵活性，所采用的S盒可以灵活地被替换，以应对突发性的安全威胁。算法的32轮迭代采用串行处理，这与AES中每轮使用代换和混淆并行地处理整个分组有很大不同。
- 它的解密算法与加密算法的结构相同，只是轮密钥的使用顺序相反，解密轮密钥是加密轮密钥的逆序。

SM4的加密算法和解密算法

- 设明文输入为 $(X_0, X_1, X_2, X_3) \in (Z_2^{32})^4$，密文为 $(Y_0, Y_1, Y_2, Y_3) \in (Z_2^{32})^4$，轮密钥为 $rk_i \in Z_2^{32}$。加密变换为：

$$X_{i+4} = F(X_i, X_{i+1}, X_{i+2}, X_{i+3}, rk_i) = X_i \oplus T(X_{i+1} \oplus X_{i+2} \oplus X_{i+3} \oplus rk_i), \quad i = 0,1,\cdots,31$$

$$(Y_0, Y_1, Y_2, Y_3) = (X_{35}, X_{34}, X_{33}, X_{32})$$

- SM4算法的解密变换和加密变换结构相同，不同的仅是轮密钥的使用顺序。
  - 加密时轮密钥的使用顺序为 $(rk_0, rk_1, \cdots, rk_{31})$，
  - 解密时轮密钥的使用顺序为 $(rk_{31}, rk_{30}, \cdots, rk_0)$。

SM4加密算法整体结构

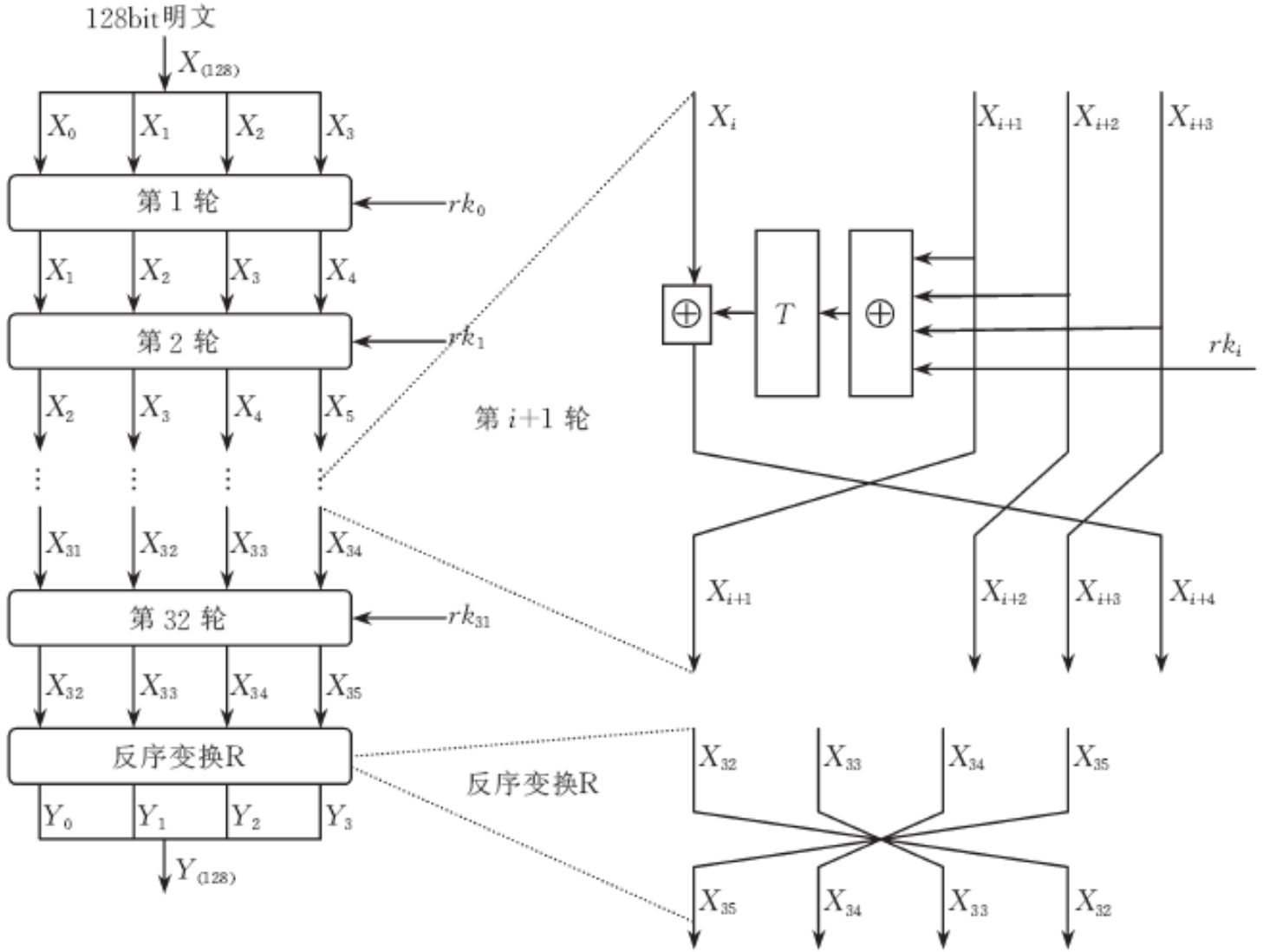
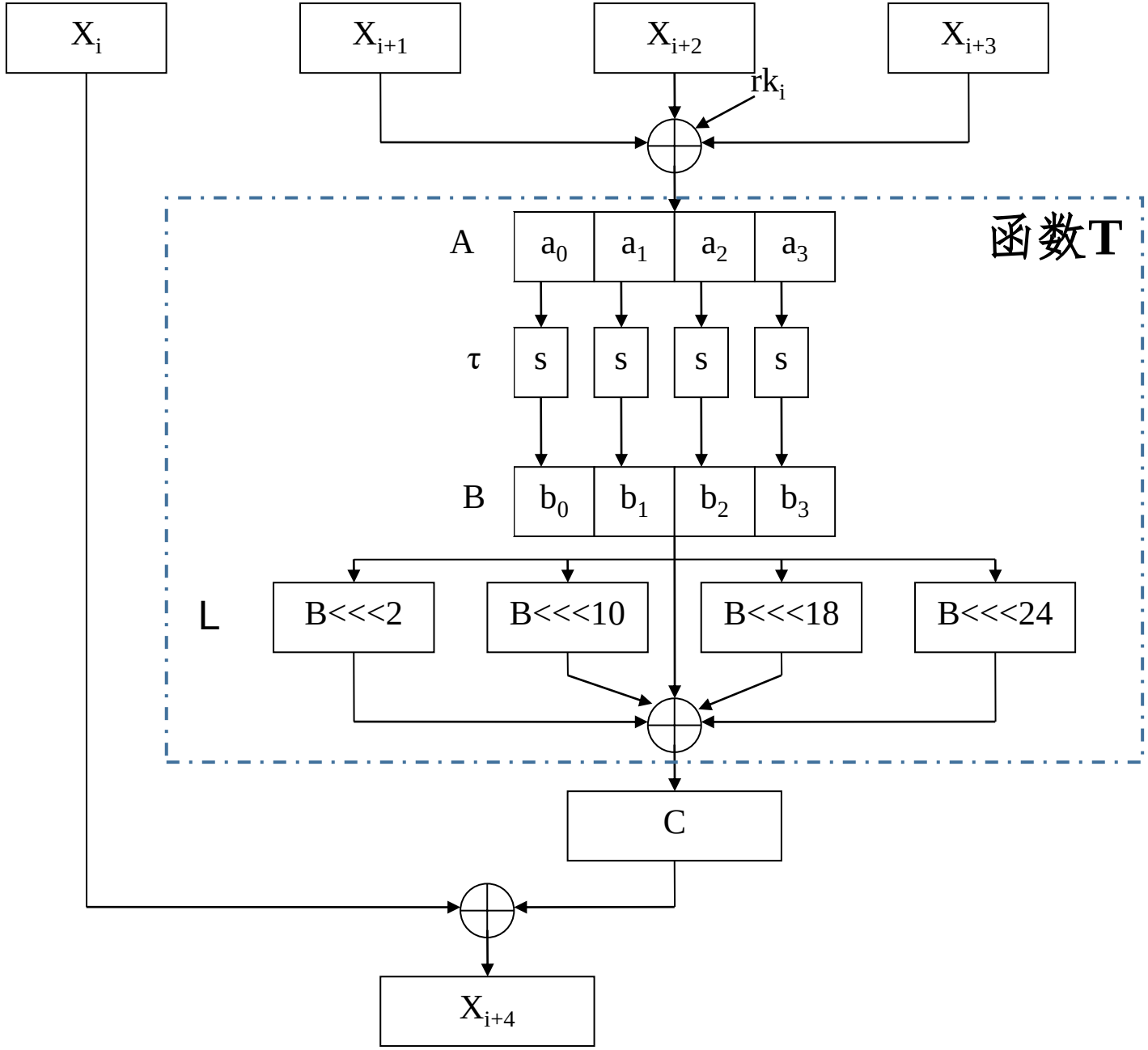


图 1　SMS4 加密算法整体结构


X_i
X_{i+1}
X_{i+2}
X_{i+3}
rk_i
函数T
A
a_0 a_1 a_2 a_3
τ
s s s s
B
b_0 b_1 b_2 b_3
L
B<<<2
B<<<10
B<<<18
B<<<24
C
X_{i+4}

SM4解密的合理性

$$X_{i+4} = F(X_i, X_{i+1}, X_{i+2}, X_{i+3}, rk_i) = X_i \oplus T(X_{i+1} \oplus X_{i+2} \oplus X_{i+3} \oplus rk_i), \quad i = 0,1,\cdots,31$$

$$(Y_0, Y_1, Y_2, Y_3) = (X_{35}, X_{34}, X_{33}, X_{32})$$

$$X_{35} = X_{31} \oplus T(X_{34} \oplus X_{33} \oplus X_{32} \oplus rk_{31})$$

$$\begin{aligned} Y_4 &= F(Y_0, Y_1, Y_2, Y_3, rk_{31}) \\ &= Y_0 \oplus T(Y_1 \oplus Y_2 \oplus Y_3 \oplus rk_{31}) \\ &= X_{35} \oplus T(X_{34} \oplus X_{33} \oplus X_{32} \oplus rk_{31}) \\ &= X_{31} \oplus T(X_{34} \oplus X_{33} \oplus X_{32} \oplus rk_{31}) \oplus T(X_{34} \oplus X_{33} \oplus X_{32} \oplus rk_{31}) \\ &= X_{31} \end{aligned}$$

第四章 公钥密码

复习要点

- 公钥密码的基本思想
- RSA算法原理及相关安全性分析
- ElGamal算法原理及相关安全性分析
- SM2算法原理
- 选择密文攻击

公钥密码体制的基本概念

- 对称密码算法的缺陷
  - 密钥分配问题： 通信双方加密通信前要通过秘密的安全信道协商加密密钥，这种安全信道可能很难实现；对这个信道安全性的要求比正常传送消息信道的安全性要高
  - 密钥管理问题： 在多用户网络中，任何两个用户之间都需要有共享的秘密钥，n个用户需要$C_n^2=n(n-1)/2$个密钥，n=5000时，$C_n^2=12,497,500$，系统开销非常大
  - 没有签名功能： 当主体A收到主体B的电子文挡时，无法向第三方证明此电子文档确实来源于B， 传统单钥加密算法无法实现抗抵赖的需求

公钥密码的理论基础：陷门单向函数

单向函数的例子？

单向函数：已知x，计算y使得y=f(x)容易；

已知y，计算x使得y=f(x)是难解的。

陷门单向函数：t是与f有关的一个参数

已知x，计算y使得y=f(x)容易；

如果不知道t，已知y，计算x使得y=f(x)是难的，

但知道t时，已知y，计算x使得y=f(x)是容易的。

参数t称为陷门。

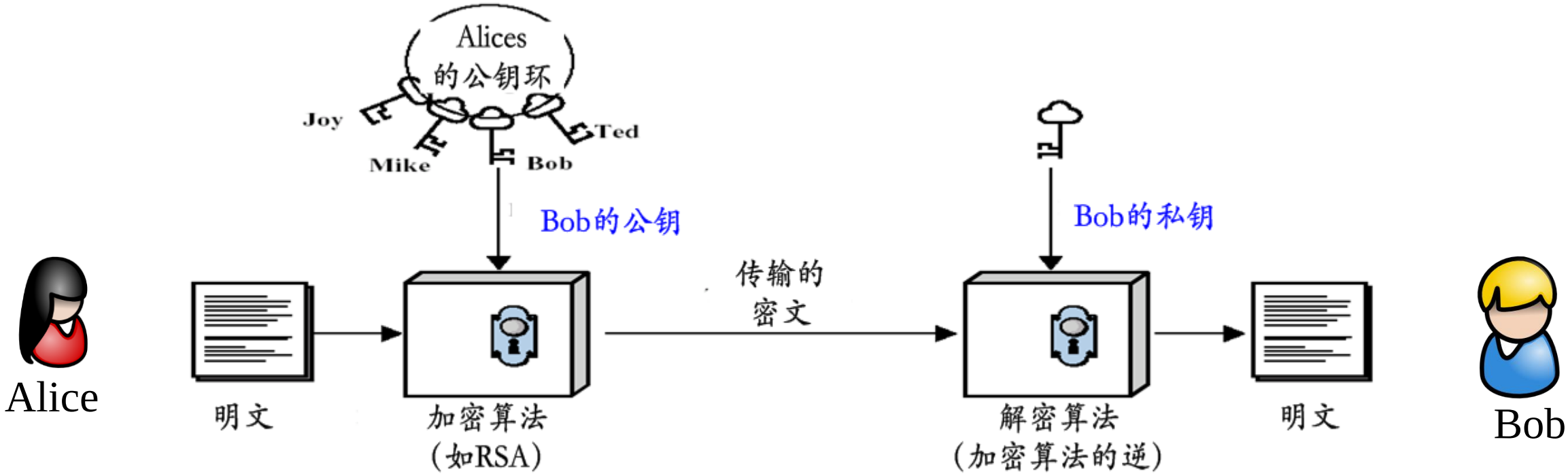
公钥密码体制

- 每个用户生成一个密钥对：一个公钥**pk**和一个对应的私钥 **sk**
  - 公钥将在系统内被公开
  - 私钥由用户本人安全保管

- 私钥由用户本人使用，而公钥则由系统中其他用户使用

- 公钥密码体制也被称为： 非对称密码体制

公钥密码体制的基本思想



公钥密码算法应满足的要求

•公钥密码算法应满足以下要求：

•① 接收方B产生密钥对（PK_B和SK_B）在计算上是容易的。

•② 发方A对消息m加密以产生密文c，即$c=E_{PKB}[m]$在计算上是容易的。

•③ 收方B用自己的秘密钥对c解密，即$m=D_{SKB}[c]$在计算上是容易的。

•④ 敌手由PK_B求SK_B在计算上是不可行的。

•⑤ 敌手由密文c和PK_B恢复明文m在计算上是不可行的。

•以上要求的本质之处在于要求一个陷门单向函数。

•研究公钥密码算法就是要找出合适的陷门单向函数。

困难问题

- （1）大整数分解问题（factorization problem）

若已知两个大素数p和q，求$n = pq$是容易的，只需一次乘法运算，而由n，求p和q则是困难的，这就是大整数分解问题。

- （2）离散对数问题（discrete logarithm problem）

给定一个大素数p，$p-1$含另一大素数因子q，则可构造一个乘法群，它是一个$p-1$阶循环群。设g是的一个生成元，$1<g<p-1$。已知x，求$y=g^x \bmod p$是容易的，而已知y、g、p，求x使得$y=g^x \bmod p$成立则是困难的，这就是离散对数问题。

RSA加密算法描述－密钥产生

- 独立地选取两大素数p和q(各100～200位十进制数字)
- 计算 $n=p\times q$，其欧拉函数值$\varphi(n)=(p-1)(q-1)$
- 随机选一整数e，$1\le e<\varphi(n)$，$\gcd(\varphi(n), e)=1$
- 在模$\varphi(n)$下，计算e的有逆元$d=e^{-1} \bmod \varphi(n)$
- 以n，e为公钥。秘密钥为d。(p, q不再需要，可以销毁。)

RSA算法描述－加密和解密

- 加密

将明文分组，各组对应的十进制数小于n

$$c=m^e \mod n$$

- 解密

$$m=c^d \mod n$$

n与φ(n)

是否有不通过分解大整数的其它攻击途径？下面我们证明由直接确定$\varphi(n)$等价于对n的分解。

设 $n=p\times q$中， $p>q$, 由 $\varphi(n)=(p-1)(q-1)$, 我们有

$$p+q=n-\varphi(n)+1$$

以及

$$p-q=\sqrt{(p+q)^2-4n}=\sqrt{[n-\varphi(n)+1]^2-4n}$$

由此可得，

$$p=\frac{1}{2}\left[(p+q)+(p-q)\right]$$

$$q=\frac{1}{2}\left[(p+q)-(p-q)\right]$$

所以，由p、q确定$\varphi(n)$和由$\varphi(n)$确定p、q是等价的。

RSA的安全性

为保证算法的安全性，还对p和q提出以下要求：

（1） |p-q|要大。

（2）p和q的长度应该相差不多。

（3） p-1和q-1都应有大素因子。

（4） gcd(p-1, q-1)应该很小。

对RSA的攻击

•1. 共模攻击

•设两个用户的公开钥分别为e_1和e_2，且e_1和e_2互素，明文消息是m，密文分别是

$$c_1 = m^{e_1} \bmod n$$

$$c_2 = m^{e_2} \bmod n$$

•敌手截获c_1和c_2后，可如下恢复m。用推广的Euclid算法求出满足$re_1+se_2=1$的两个整数r和s，其中一个为负，设为r。再次用推广的Euclid算法求出c_1^{-1}，由此得

$$c_1^r c_2^s = (c_1^{-1})^{-r} c_2^s \equiv m \bmod n$$

3. 选择密文攻击

RSA是确定性的加密算法，不能抵御选择密文攻击。

假设攻击者得到了RSA公钥(n,e)，截获到某个密文c_1，假设其明文为m_1,即

$$c_1 = m_1^e (\mathrm{mod}\ n)$$

然后，攻击者选取明文m，构造新密文c_2:

$$c_2 = c_1 m^e (\mathrm{mod}\ n)$$

攻击者让解密者对c_2解密，获得明文m_2，则计算：

$$m_1 = m_2 m^{-1} (\mathrm{mod}\ n)$$

ElGamal密码体制

1. 密钥产生过程： 选择一素数p以及小于p的随机数x， g是p的原根，计算 $y \equiv g^x \bmod p$。

   (y，g，p)作为公开密钥，x作为秘密密钥。

2. 加密过程： 明文消息M，随机选一整数 k<p-1，计算

   $C_1 \equiv g^k \ (\bmod\ p)$， $C_2 \equiv y^k M \ (\bmod\ p)$

密文为$C=(C_1, C_2)$。

3. 解密过程：

$$M = c_2 (c_1^x)^{-1} (\bmod\ n)$$

$$c_2 (c_1^x)^{-1} (\bmod\ p) = y^k M (g^{kx})^{-1} (\bmod\ p) = y^k M (y^k)^{-1} (\bmod\ p) = M (\bmod\ p)$$

ElGamal密码体制的特点

1. 安全性基于有限域上的离散对数的难解性
2. 加密算法是概率算法
3. 不能抵御选择密文攻击
4. 存在密文扩张

ElGamal密码的安全性

- 参数要求：p应为150位以上十进制数，500位以上的二进制数，p-1应有大素数因子。
- K必须保密而且必须是一次性的。
  - K泄露，则敌手可计算出y^k从而可以计算出M
  - 使用同一k加密不同的明文M，M’，如果敌手知道M就可由$C_2/C_2'=M/M'$求出M’

ElGamal的选择密文攻击

- 假设敌手要解密

$$C_1 \equiv g^k \bmod p, \quad C_2 \equiv y^k M \bmod p$$

在选择密文攻击下,攻击者可以根据目标密文$C=(C_1,C_2)$，自己选择明文m，计算 $C=(C_1,mC_2)$，通过解密算法得到对应的明文$M'=mM$,因此 计算$M=M'/m$得到目标密文$C=(C_1,C_2)$的明文。

椭圆曲线密码体制

为保证RSA算法的安全性，它的密钥长度需一再增大，使得运算负担越来越大。相比之下，椭圆曲线密码体制ECC（elliptic curve cryptography）、SM2可用短得多的密钥获得同样的安全性，因此具有广泛的应用前景。

ECC和RSA/DSA体制在保持同等安全的条件下各自所需的密钥的长度

RSA/DSA	512	768	1024	2048	21000
ECC	106	132	160	211	600

有限域上的椭圆曲线

密码中普遍采用的是有限域上的椭圆曲线，是指曲线方程定义式中，所有系数都是某一有限域GF(p)中的元素（p为一大素数）。其中最为常用的是由方程$E_p(a,b)$:

$$y^2 \equiv x^3+ax+b(\mathrm{mod}\ p) \quad (a,b \in GF(p), 4a^3+27b^2 \neq 0(\mathrm{mod}\ p)) \qquad (4.2)$$

定义的曲线。

椭圆曲线上点的加法运算

设$P=(x_1,y_1)$，$Q=(x_2,y_2)$，$P\neq -Q$，则$P+Q=(x_3,y_3)$由以下规则确定：

$$x_3=\lambda^2-x_1-x_2$$

$$y_3=\lambda(x_1-x_3)-y_1$$

其中

$$\lambda=\begin{cases}\dfrac{y_2-y_1}{x_2-x_1}, P\neq Q\\ \dfrac{3x_1^2+a}{2y_1}, P=Q\end{cases}$$

椭圆曲线上的密码

椭圆曲线上的数学困难问题

在椭圆曲线构成的Abel群$E_p(a,b)$：

$P \in E_p(a,b)$, P的阶是一个非常大的素数，P的阶是满足$nP=O$的最小正整数n。

$Q=kP$,

(1) 已知k和P易求Q;

(2) 已知P、Q求k则是困难的

这就是椭圆曲线上的离散对数问题，可应用于构造公钥密码体制。

Diffie-Hellman密钥交换和ElGamal密码体制可推广到椭圆曲线来实现。

SM2的基本参数

基于素数域 F_p 的SM2算法参数如下：

- F_p 的特征p，为 m 比特长的素数 p，要尽可能大，但太大会影响计算速度；通常选择160比特大小。
- 长度不小于192比特的比特串 $SEED$ ；
  - F_p 上的2个元素 a,b ，满足 $4a^3+27b^2 \neq 0$ ，定义

$$E(F_p): y^2 = x^3 + ax + b(modp)$$

- 基点 $G=(x_G, y_G) \in E(F_p), G \neq O$ ；
- G 的阶 n 为 m 比特长的素数，满足 $n > 2^{191}$ 且 $n > 4\sqrt{p}$
- $h=\frac{|E(F_p)|}{n}$ 称为余因子，其中 $|E(F_p)|$ 是曲线 $E(F_p)$ 的点数。

加密算法

设发送方是A，A要发送的消息表示成比特串M，M的长度为$klen$。

加密运算如下：

（1）选择随机数 $k \leftarrow_R \{1,2,\cdots,n-1\}$；

（2）计算椭圆曲线点$C_1 = kG = (x_1, y_1)$，将(x_1, y_1)表示为比特串；

（3）计算椭圆曲线点$S = hP_B$，若S是无穷远点，则报错并退出；

（4）计算椭圆曲线点$kP_B = (x_2, y_2)$，将(x_2, y_2)表示为比特串；

（5）计算 $t = KDF(x_2 \| y_2, klen)$，若$t$为全 0 的比特串，则返回（1）；

（6）计算 $C_2 = M \oplus t$；

（7）计算 $C_3 = Hash(x_2 \| M \| y_2)$；

（8）输出密文 $C = (C_1, C_2, C_3)$。

其中第（5）步$KDF(\cdot)$是密钥派生函数，其本质上就是一个伪随机数产生函数，用来产生密钥，取为密码哈希函数SM3。第（3）步$Hash$函数也取为SM3。

解密算法

B 收到密文后，执行以下解密运算：

（1）从C 中取出比特串C_1，将 C_1表示为椭圆曲线上的点，验证 C_1是否满足椭圆曲线方程，若不满足则报错并退出；

（2）计算椭圆曲线点 $S=hC_1$，若S是无穷远点，则报错并退出；

（3）计算 $d_BC_1=(x_2,y_2)$，将坐标 x_2, y_2 表示为比特串；

（4）计算 $t=KDF(x_2\|y_2,klen)$ ，若t 为全0比特串，则报错并退出；

（5）从C 中取出比特串 C_2 ，计算$M'=C_2\oplus t$ ；

（6）计算 $u=Hash(x_2\|M'\|y_2)$ ，从 C 中取出C_3，若$u\neq C_3$ ，则报错并退出；

（7）输出明文M'。

解密的正确性

解密的正确性：

因为 $P_B = d_B G$ ，$C_1 = kG = (x_1, y_1)$，由解密算法的第（3）步可得

$$d_B C_1 = d_B kG = k(d_B G) = kP_B = (x_2, y_2)$$

所以解密算法第（4）步得到的 t 与加密算法第（5）步得到的 t相等，由 $C_2 \oplus t$，便得到明文 。

第五章 MAC和Hash函数

复习要点

- 消息认证码的定义及构造
- 杂凑函数的定义及性质
- 杂凑函数设计的一般模式
- SHA1的分组长度、填充、摘要长度、迭代轮数
- SM3的分组长度、填充、摘要长度、迭代轮数

消息认证码

消息认证码的定义及使用方式

•消息认证码：指消息被一密钥控制的公开函数作用后产生的用作认证符的固定长度的数值，或称为密码校验和。

$$M||C_K(M)$$

•此时需要通信双方A和B共享一密钥K。

• 如果仅收发双方知道K，且B计算得到的MAC与接收到的MAC一致，则这一就实现了以下功能：

•① 接收方相信发送方发来的消息未被篡改。

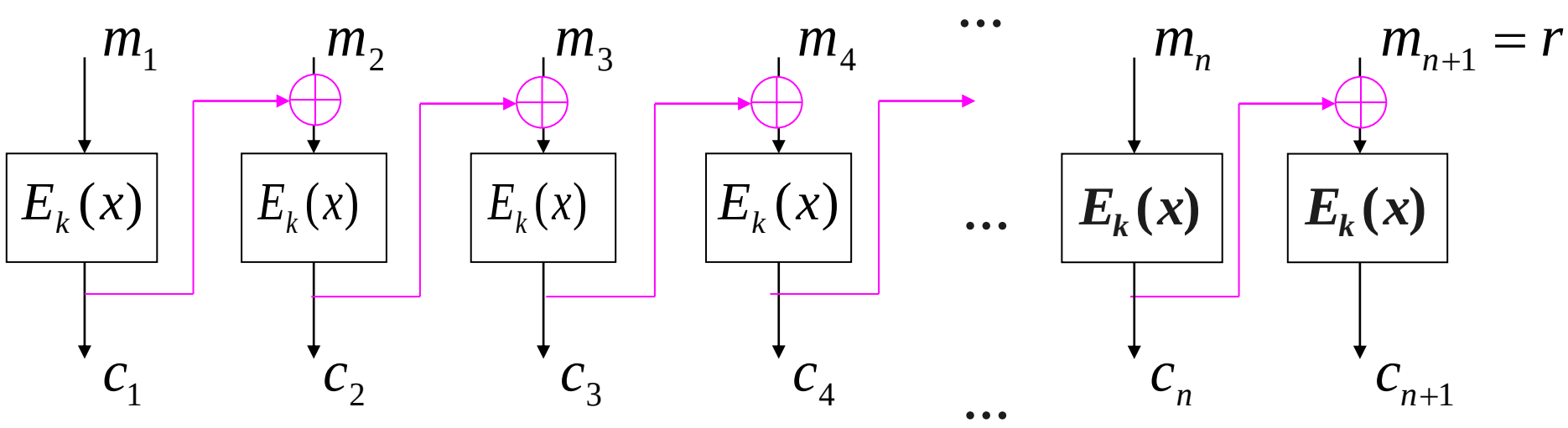
•② 接收方相信发送方不是冒充的。

认证码生成

n个分组明文 $m=(m_1,\cdots,m_n)$，校验码为

$$r=m_{n+1}=m_1\oplus\cdots\oplus m_n$$

C_{n+1}为认证码。

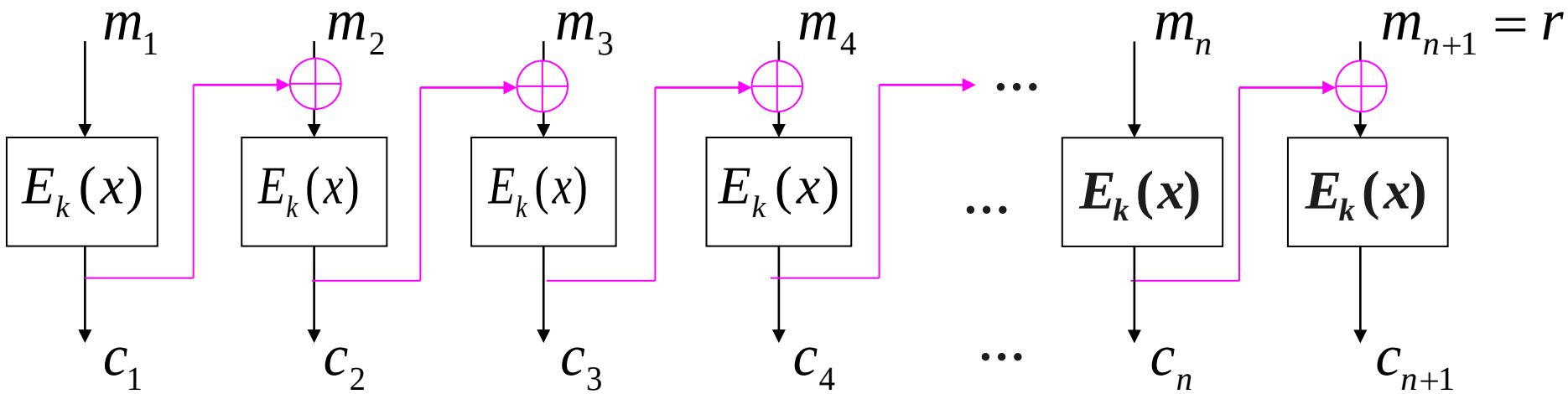


(1) 仅需对明文认证,而不需加密时,传送明文**m**和认证码C_{n+1}，此时也可仅保留C_{n+1}的 **t** 个比特作为认证码;

(2) 既需对明文认证,又需要加密时,传送密文**C**和认证码C_{n+1}

认证码检验

- （1）仅需对明文认证而不需加密时,此时验证者仅收到明文m和认证码C_{n+1}，他需要：
- Step1 产生明文m的校验码 $r = m_{n+1} = m_1 \oplus \cdots \oplus m_n$
- Step2 利用共享密钥使用CBC模式对(m,r)加密，将得到的最后一个密文分组与接收到的认证码C_{n+1}比较，二者一致时判定接收的明文无错；二者不一致时判定明文出错。



Hash函数满足条件

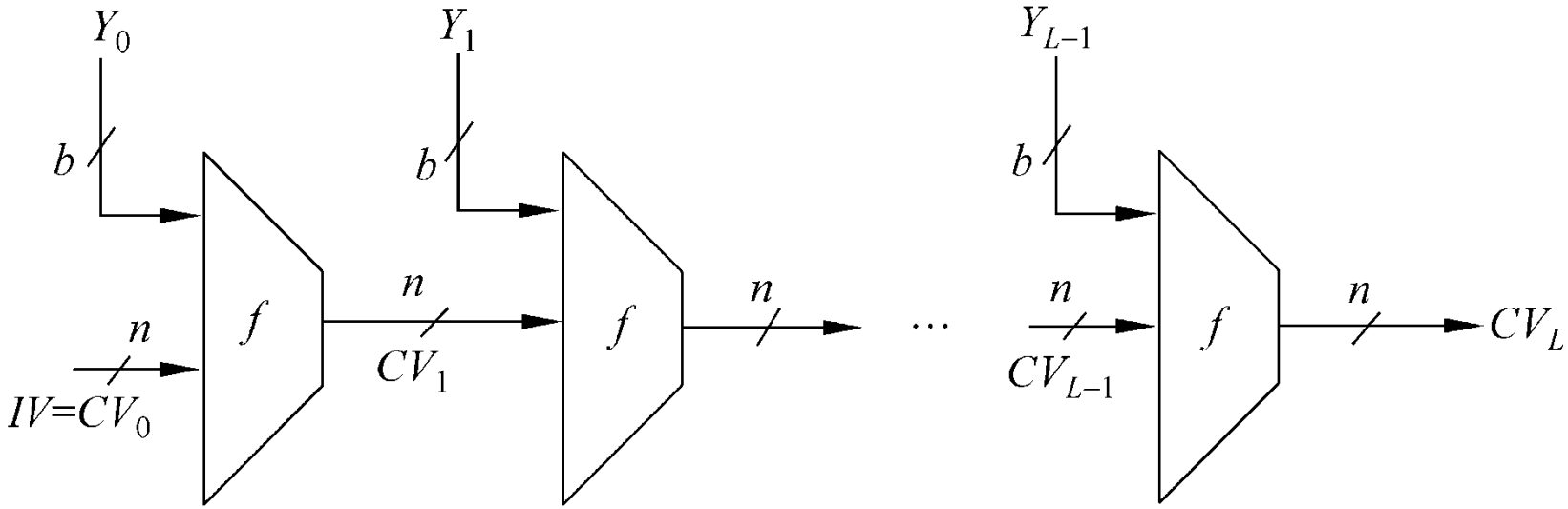
- **Hash函数的目的是为需认证的数据产生一个“指纹”，Hash函数应满足以下条件：**
  - **Hash函数函数的输入可以是任意长**
  - **Hash函数函数的输出是固定长**
  - 易于在软件和硬件实现

Hash函数满足的安全条件

- 同时，**Hash**函数为了实现安全认证，需要满足如下安全条件：
  - 单向性：已知**x**，求**H(x)**较为容易；但是，已知**h**，求使得**H(x)=h**的**x**在计算上是不可行的。
  - 抗弱碰撞性：已知**x**，找出**y(y≠x)**使得**H(y)=H(x)**在计算上是不可行的。
  - 抗强碰撞性：找出任意两个不同的输入**x**、**y**，使得**H(y)=H(x)**在计算上是不可行的。

迭代型哈希函数的一般结构

- 迭代



CV_0=IV=n比特长的初值；

$CV_i=f(CV_{i-1},Y_{i-1})1\leq i\leq L$；

$H(M)=CV_L$

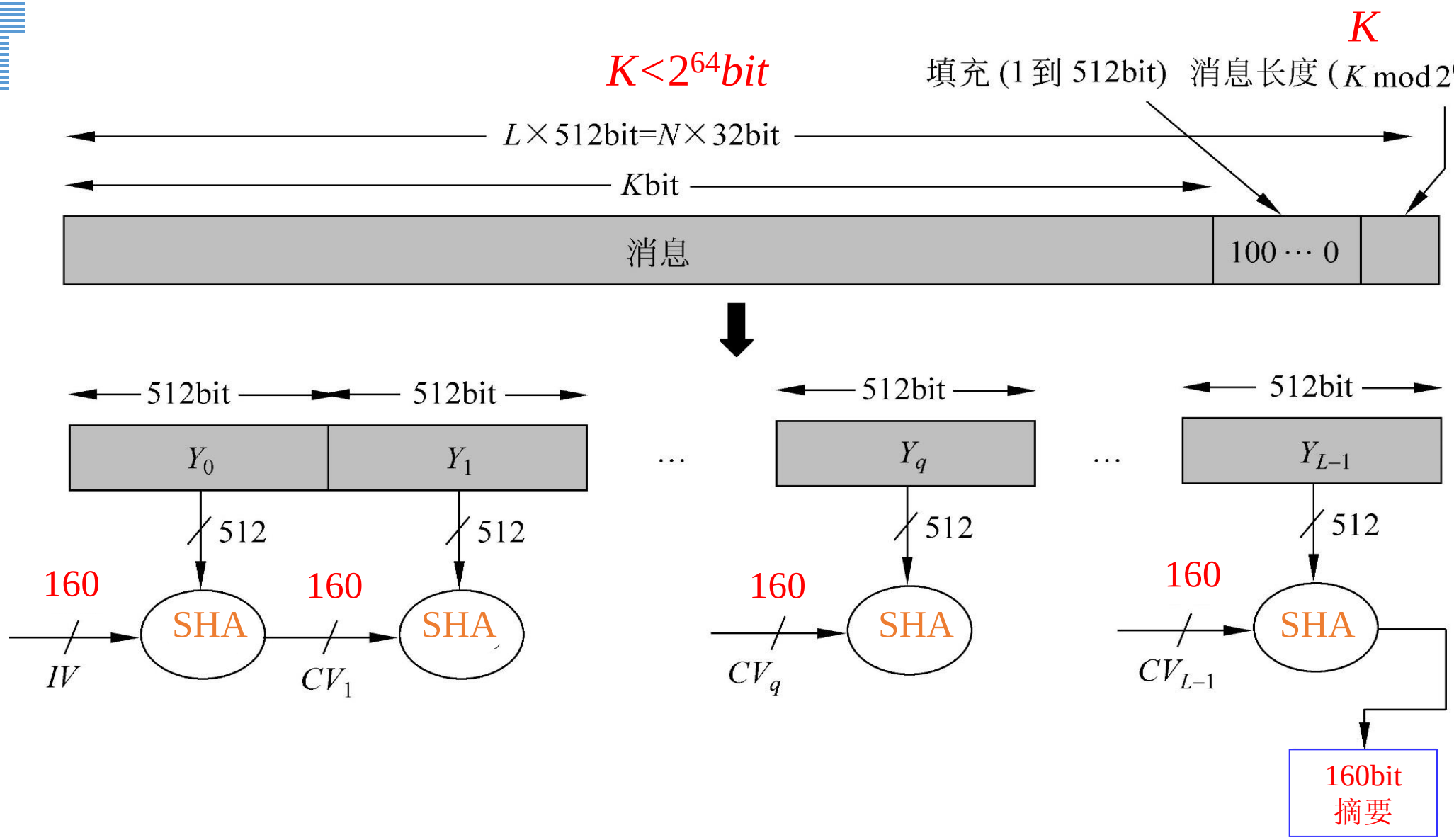
CV_{i-1}，称为链接变量

通常b>n，称函数f为压缩函数

分析时需要先找出f 的碰撞

安全杂凑算法SHA-1算法描述

- 算法的输入：小于2^{64}比特长的任意消息，分为512比特长的分组。
- 算法的输出：160比特长的消息摘要。
- 算法的框图与MD5一样，但杂凑值的长度和链接变量的长度为160比特


K
K<2^64bit
填充 (1 到 512bit)
消息长度 (K mod 2^64)
L×512bit=N×32bit
Kbit
消息
100 … 0
512bit
512bit
512bit
512bit
Y0
Y1
Yq
YL–1
512
512
512
512
160
160
160
SHA
SHA
SHA
SHA
IV
CV1
CVq
CVL–1
160bit
摘要

基本逻辑函数f_t

迭代的步数	函数名	定义
$0 \le t \le 19$	$f_1 = f_t(B,C,D)$	$(B \wedge C) \vee (\overline{B} \wedge D)$
$20 \le t \le 39$	$f_2 = f_t(B,C,D)$	$B \oplus C \oplus D$
$40 \le t \le 59$	$f_3 = f_t(B,C,D)$	$(B \wedge C) \vee (B \wedge D) \vee (C \wedge D)$
$60 \le t \le 79$	$f_4 = f_t(B,C,D)$	$B \oplus C \oplus D$

- 基本逻辑函数的输入为3个32比特的字，输出是一个32比特的字。表中$\wedge$，$\vee$，$-$，$\oplus$分别是与、或、非、异或4个逻辑运算。

SM3密码杂凑算法

- **SM3**是中国国家密码管理局颁布的中国商用公钥密码标准算法，它是一类密码杂凑函数，可用于数字签名及验证、消息认证码生成及验证、随机数生成。
- 标准起草人：王小云、李峥、于红波、张超、罗鹏、吕述望
- **2012年3月，成为中国商用密码标准（GM/T 0004-2012）**
- **2016年8月，成为中国国家密码标准（GB/T 32905-2016）**

SM3密码杂凑算法的描述

算法的输入数据长度为l比特，$l < 2^{64}$,输出哈希值长度为256比特。

1. 常数与函数

(1) 常数

初始值

$$IV = 7380166F4914B2B9$$
$$172442D7DA8A0600$$
$$A96F30BC163138AA$$
$$E38DEE4DB0FB0E4E$$

常量

$$T_j = \begin{cases} 79CC4519, & 0 \le j \le 15 \\ 7A879D8A, & 16 \le j \le 63 \end{cases}$$

SM3密码杂凑算法的描述

(2) 函数

布尔函数：

式中 X,Y,Z 为32位字，$\wedge,\vee,-,\oplus$ 分别是逻辑与、逻辑或、逻辑非和逐比特异或运算

$$FF_j(X,Y,Z)=\begin{cases}X\oplus Y\oplus Z, & 0\le j\le 15\\ (X\wedge Y)\vee(X\wedge Z)\vee(Y\wedge Z), & 16\le j\le 63\end{cases}$$

$$GG_j(X,Y,Z)=\begin{cases}X\oplus Y\oplus Z, & 0\le j\le 15\\ (X\wedge Y)\vee(\bar{X}\wedge Z), & 16\le j\le 63\end{cases}$$

混淆

置换函数：

$$P_0(X)=X\oplus(X<<<9)\oplus(X<<<17);$$
$$P_1(X)=X\oplus(X<<<15)\oplus(X<<<23).$$

扩散

式中 X 为32位字，符号 $a<<<n$ 表示把 a 循环左移 n 位。

SM3密码杂凑算法的描述

2. 算法描述

算法对数据首先进行填充，再进行迭代压缩后生成哈希值。

(1) 填充并附加消息的长度

- 对消息填充的目的是使填充后的数据长度为512的整数倍。
- 设消息m的长度为l比特。首先将比特“1”添加到m的末尾，再添加k个“0”，其中k满足

$$l+1+k=448 \bmod 512$$

- 然后再添加一个64位比特串，该比特串是长度l的二进制表示。

举例：消息为01100001011000100110001 1，其长度为l=24，填充后的比特串为

423个0　　64比特

$$01100001011000100110001\,1\,\overbrace{00\cdots00}^{423\text{个}0}\,\overbrace{00\cdots011000}^{64\text{比特}}$$

SM3密码杂

$ABCDEFGH = V^{(i)}$

FOR j=0 to 63

$SS_1 \leftarrow \left((A <<< 12) + E + \left(T_j <<< j\right)\right) <<< 7;$

$SS_2 = SS_1 \oplus (A <<< 12);$

$TT_1 = FF_j(A,B,C) + D + SS_2 + W'_j;$

$TT_2 = GG_j(E,F,G) + H + SS_1 + W_j;$

$D = C;$

$C = B <<< 9;$

$B = A;$

$A = TT_1;$

$H = G;$

$G = F <<< 19;$

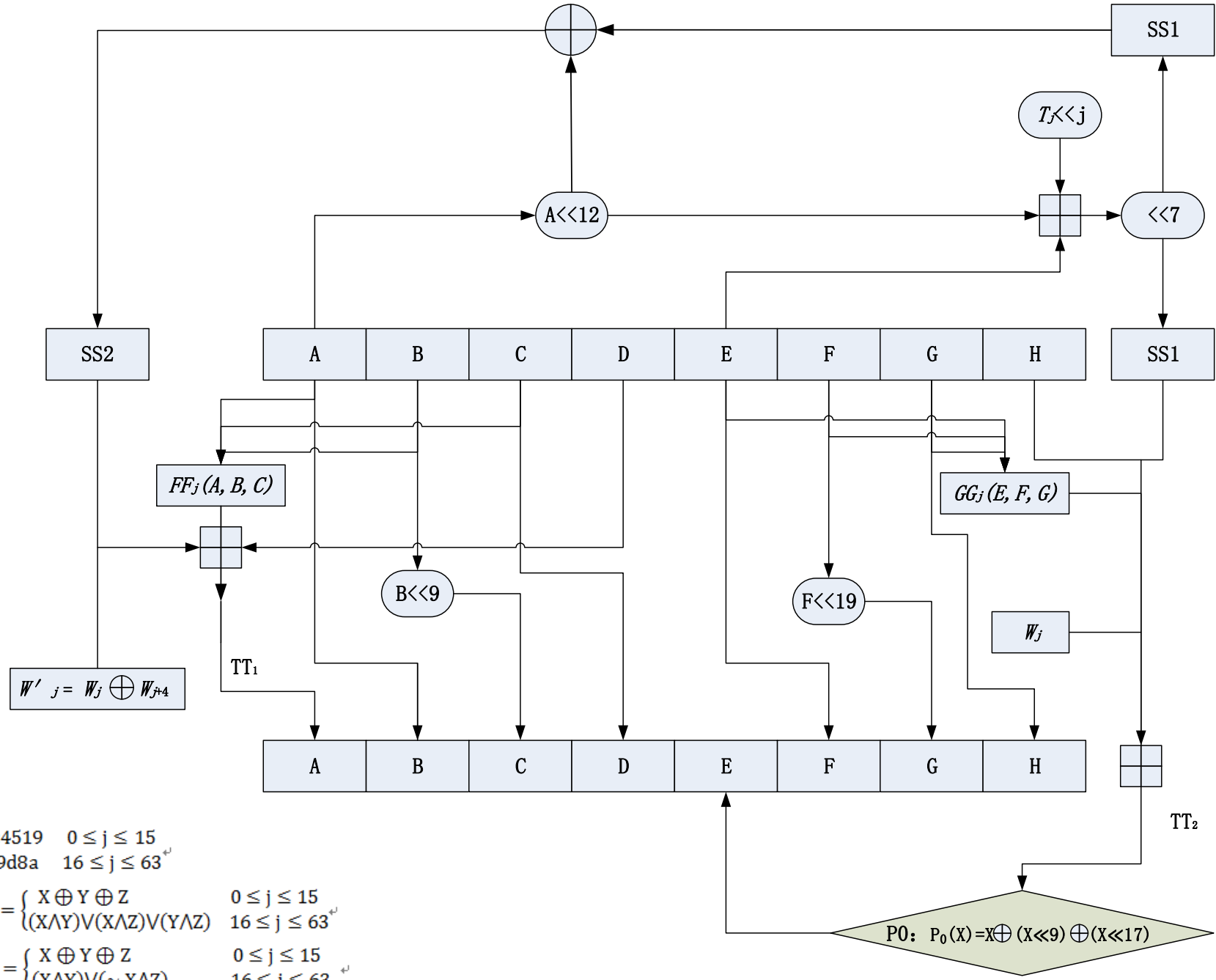
$F = E;$

$E = P_0(TT_2)$

$ENDFOR$

$V^{(i+1)} = ABCDEFGH \oplus V^{(i)}$

SM3算法第j轮迭代压缩流程图



其中：

$$Tj = \begin{cases} 79cc4519 & 0 \le j \le 15 \\ 7a879d8a & 16 \le j \le 63 \end{cases}$$

$$FF_j(X,Y,Z) = \begin{cases} X \oplus Y \oplus Z & 0 \le j \le 15 \\ (X \wedge Y) \vee (X \wedge Z) \vee (Y \wedge Z) & 16 \le j \le 63 \end{cases}$$

$$GG_j(X,Y,Z) = \begin{cases} X \oplus Y \oplus Z & 0 \le j \le 15 \\ (X \wedge Y) \vee (\sim X \wedge Z) & 16 \le j \le 63 \end{cases}$$

第六章 数字签名

复习要点

- 数字签名应具有的特性
- **RSA数字签名体制的原理、弱点及改进**
- **ElGamal数字签名体制**
- **DSS数字签名体制**
- 交互式证明

数字签名的基本概念

数字签名(**digital signature**)：用于对数字消息签名,以防消息的伪造或篡改,也可用于通信双方的身份鉴别。

数字签名应具有的特性：

（1）签名是可信的：任何人可验证签名的有效性。

（2）签名是不可伪造的：除合法签名者外，其他人伪造签名是困难的。

（3）签名是不可复制的：一消息的签名不能复制为另一消息的签名。

（4）签名的消息是不可改变的：经签名的消息不能被篡改。

（5）签名是不可抵赖的：签名者事后不能否认自己的签名。

数字签名：对身份认证，保持数据完整性、不可否认性。

RSA数字签名

RSA签名体制为例说明签名的产生过程。

体制参数：

选两个保密的大素数p和q，计算$n=p\times q$，$\varphi(n)=(p-1)(q-1)$；

选一整数e，满足$1<e<\varphi(n)$，且$\gcd(\varphi(n),e)=1$；

计算d，满足$d\cdot e\equiv 1 \bmod \varphi(n)$；

以{e,n}为公开钥，{d,p,q}为秘密钥。

签名：

设消息为$m \in Z_n$，对其签名为

$$s \equiv m^d \bmod n$$

验证：

接收方在收到消息m和签名s后，验证

$$m \overset{?}{\equiv} s^c \pmod{n}$$

是否成立，若成立，则发送方的签名有效。

安全性：大数分解的困难性。

RSA签名体制的安全性分析

缺点：

（1）对任意$y \in Z_n$，任何人可计算$x \equiv y^e \bmod n$，因此任何人可伪造对随机消息x的签名。

（2）如果消息x_1和x_2的签名分别为y_1和y_2，则知道x_1, y_1, x_2, y_2的人可伪造对消息$x_1 x_2$的签名$y_1 y_2$。

（3）在RSA签名方案中，需签名的消息$x \in Z_n$，所以每次只能对$\lfloor \log_2 n \rfloor$位长的消息进行签名。签名速度慢。

克服缺陷的方法：签名之前先求消息的Hash值。

数字签名的一般描述

明文消息：m　　私钥：x　　公钥：y

签名算法：$s=Sig_x(m)$

验证算法：$Ver_y(s, m)$

$$Ver_y(s,m)=\begin{cases} True & s=Sig_x(m) \\ False & s\neq Sig_x(m) \end{cases}$$

算法的安全性：从m和s难求密钥x，

或伪造消息m'，使m'和s可被验证为真。

DSS数字签名标准

- **1**．参数与密钥生成
- ① 选取大素数***p***，满足$2^{L-1}<p<2^{L}$，其中**512≤*L*≤1024**且***L***是**64**的倍数。显然，***p***是***L***位长的素数，***L***从**512**到**1024**且是**64**的倍数。
- ② 选取大素数***q***，***q***是***p*–1**的一个素因子且 $2^{159}<q<2^{160}$ ，即***q***是**160**位的素数且是***p*–1**的素因子。
- ③ 选取一个生成元 $g=h^{(p-1)/q} \bmod p$，其中***h***是一个整数，满足**1<*h*<*p*–1**并且 $h^{(p-1)/q} \bmod p>1$ 。
- ④ 随机选取整数***x***，**0<*x*<*q***，计算 $y=g^{x} \bmod p$ 。
- ⑤ ***p***、***q***和***g***是公开参数，***y***为公钥，***x***为私钥。

- **2．签名**

  对于消息***m***，首先随机选取一个整数***k*** ，**0＜*k*＜*q***，然后计算：

$$r=(g^k \bmod p) \bmod q \quad , \quad s=k^{-1}(h(m)+xr) \bmod q$$

  则***m***的签名为**(*r*, *s*)**，其中***h***为**Hash**函数，**DSS**规定**Hash**函数为**SHA-1**。

- **3．验证**

  对于消息签名对**(*m*, (*r*, *s*))**，首先计算：

$$w=s^{-1} \bmod q, \quad u_1=h\ (m)\ w \bmod q$$
$$u_2=rw \bmod q, \quad v=(\ g^{u_1} y^{u_2} \bmod p\ ) \bmod q$$

  然后验证：$v=r$

  如果等式成立，则**(*r*, *s*)**是***m***的有效签名；否则签名无效。

算法合理性证明

- 因为： $s=k^{-1}(h(m)+xr) \bmod q$

所以： $ks=(h(m)+xr) \bmod q$

- 于是我们有：

$$
\begin{aligned}
v &= (g^{u_1} y^{u_2} \bmod p) \bmod q \\
&= (g^{h(m)w} y^{rw} \bmod p) \bmod q \\
&= (g^{h(m)w} g^{xrw} \bmod p) \bmod q \\
&= (g^{(h(m)+xr)w} \bmod p) \bmod q \\
&= (g^{ksw} \bmod p) \bmod q \\
&= (g^{kss^{-1}} \bmod p) \bmod q \\
&= (g^{k} \bmod p) \bmod q \\
&= r
\end{aligned}
$$

2. ElGamal签名体制

体制参数:

p: 大素数;

g: Z^*_p的一个生成元;

x: 用户A的秘密钥,

$x \in_R Z^*_p$;

y: 用户A的公开钥,

$y \equiv g^x \pmod p$。

签名：

用户为待签消息m选取

秘密随机数$k \in Z^*_{p-1}$

定义$Sig(m, k) = (r, s)$

$r = g^k \bmod p$

$s = k^{-1}(H(m) - xr) \bmod (p-1)$

验证：

$ver(m, (r, s), y) = 真 \Leftrightarrow$

$y^r r^s \equiv g^{H(m)} \bmod p$

讨论两个问题：
（1）用ElGamal方案计算一个签名时，使用的随机数k能不能泄露？
（2）若Bob用相同的k值来签名不同的两份消息，Oscar能否攻破这个体制？

由于私钥x是保密的，所以，攻击者要得到这个密钥，必须求解离散对数问题$\log_g y$，这是一个困难问题。但是，秘密随机数k一旦暴露，则解：$x = (h(m) - sk)r^{-1} \bmod(p-1)$，即可求得密钥$x$，方案被攻破。

不可用同一个k作两次签名。若Bob利用相同k签名两次，即m_1, m_2的签名为(r, s_2)及(r, s_1)，则Oscar可以由联立方程式

$$\begin{cases} h(m_1) = xr + ks_1 \bmod(p-1) \\ h(m_2) = xr + ks_2 \bmod(p-1) \end{cases}$$

可得：$x = [h(m_1)s_2 - h(m_2)s_1](s_2 - s_1)^{-1} r^{-1} \bmod(p-1)$，同时可以求得k。因此，同一个k不可重复使用。

交互证明系统

交互证明系统由两方参与

证明者P（prover）

验证者V（verifier）

P知道某一秘密（如公钥密码体制的秘密钥或一平方剩余x的平方根），P希望使V相信自己的确掌握这一秘密。

交互证明比较典型的方式：每轮V都向P发出一询问，P向V做出一应答。

所有轮执行完后，V根据P是否在每一轮对自己发出的询问都能正确应答，以决定是否接受P的证明。

交互证明和数学证明的区别： 数学证明的证明者可自己独立地完成证明，而交互证明是由P产生证明、V验证证明的有效性来实现，因此双方之间通过某种信道的通信是必需的。

交互证明系统须满足以下要求：

① 完备性：若P知道某一秘密，V将接受P的证明。

② 可靠性：如果P能以一定的概率使V相信P的证明，则P知道相应的秘密。

Schnorr身份识别方案

Schnorr协议需要一个可信中心，记为TA.

TA将为协议选择下列一些参数：

（1）p及q是两个大素数，且q|(p-1).q至少140位，而p至少为512位。

（2）$g \in Z^*_p$为q阶元

（3）h是一输出为t位的单向函数，t为一个安全参数；

（4）公开密钥y和秘密密钥x，用作签名。

p,q,h及其公开密钥都公布。每位用户自己选定个人秘密密钥$x \in [1,q-1]$,且计算公开密钥$y=g^x \bmod p$。

Schnorr identification scheme

G, q, g, h

Prover

x

$(G, q, g) \leftarrow \mathcal{G}(1^n)$

$x \leftarrow \mathbb{Z}_q$

$y = g^x$

$r = g^k$

e

$s = [ex + k \bmod q]$

Verifier

$e \leftarrow \mathbb{Z}_q$

$g^s y^{-e} \stackrel{?}{=} r$

$$g^s y^{-e} = g^{ex+k} y^{-e} = (g^x)^e g^k y^{-e} = g^k = r$$

3. Schnorr签名体制

体制参数:

p：大素数，$p \geq 2^{512}$；

q：大素数，$q|(p-1)$，$q \geq 2^{160}$；

g：$g \in_R Z^*_p$,且$g^q \equiv 1 \pmod p$；

x：A的秘密钥，$1<x<q$；

y：A的公开钥，$y \equiv g^x \pmod p$。

签名：

用户为待签消息m选取秘密随机数$1 < k < q$

定义$Sig(m,k) = (e,s)$

$$r = g^k \bmod p$$

$$e = H(r,m)$$

$$s = xe + k \bmod q$$

验证：

$ver(m,(e,s),y) = 真 \Leftrightarrow H(r,m) = e$

$$r = g^k = g^s g^{-xe}$$
$$= g^s y^{-e} \bmod p$$

Okamoto身份识别方案

TA将为协议选择下列一些参数：

p及q是两个大素数，$\alpha 1$,$\alpha 2 \in Z_p$为q阶元,对系统的所有参加者包括A，TA保密$c=\log_{\alpha 1}\alpha_2$。假定任何人计算c都是不可能的。TA选择一个签名方案和一个Hash函数。

Okamoto身份识别方案

TA向A颁布一个证书的协议为：

（1） TA建立A的身份并颁布一个识别串ID(A);

（2）A秘密地选择两个随机指数m1,m2, $1 \le m1,m2 \le q-1$，并计算 $v=\alpha_1^{-m_1}\alpha_2^{-m_2} \bmod p$，将v发送给TA;

（3）TA对（ID,v）签名，$s=Sig_{TA}(ID,v)$。TA将证书 C(A)=(ID(A),v,s)发送给A。

Okamoto身份识别方案

（1）A随机选择两个数r1,r2, $0 \le r1,r2 \le q-1$，并计算$X= \alpha_1^{r_1}\alpha_2^{r_2} \bmod p$;

（2）A将他的证书C(A)=(ID(A),v,s)和X发送给B;

（3）B通过检测$Ver_{TA}(ID,v)$=TURE来验证TA的签名。

（4）B随机选择一个数e,$1 \le e \le 2^t$,t为安全参数并将e发送给A。

（5）A计算$y_1=(r_1+m_1e) \bmod q, y_2=(r_2+m_2e) \bmod q$，并将$y_1, y_2$发给B。

（6）B验证$X=\alpha_1^{y_1}\alpha_2^{y_2} v^e \bmod p$

5. Okamoto签名体制

体制参数:

p：大素数，且$p \geq 2^{512}$;

q：大素数，$q|(p-1)$，且$q \geq 2^{140}$;

g_1, g_2：两个阶为q的元素，$g_1, g_2 \in Z_p^*$

x_1, x_2：用户A的秘密钥，两个小于q的随机数;

y：用户A的公开钥，$y = g_1^{-x_1} g_2^{-x_2} \bmod p$ 。

Okamoto签名体制

签名：

待签消息m,

选$k_1, k_2 \in_R Z_q^*$

定义$Sig(m, k_1, k_2) = (e, s_1, s_2)$

$$\begin{cases} e = H(g_1^{k_1} g_2^{k_2} \bmod p, m) \\ s_1 = k_1 + e x_1 \bmod q \\ s_2 = k_2 + e x_2 \bmod q \end{cases}$$

验证：

$ver(y, (e, s_1, s_2), m) = 真 \Leftrightarrow$

$H(g_1^{s_1} g_2^{s_2} y^e \bmod p, m) = e$

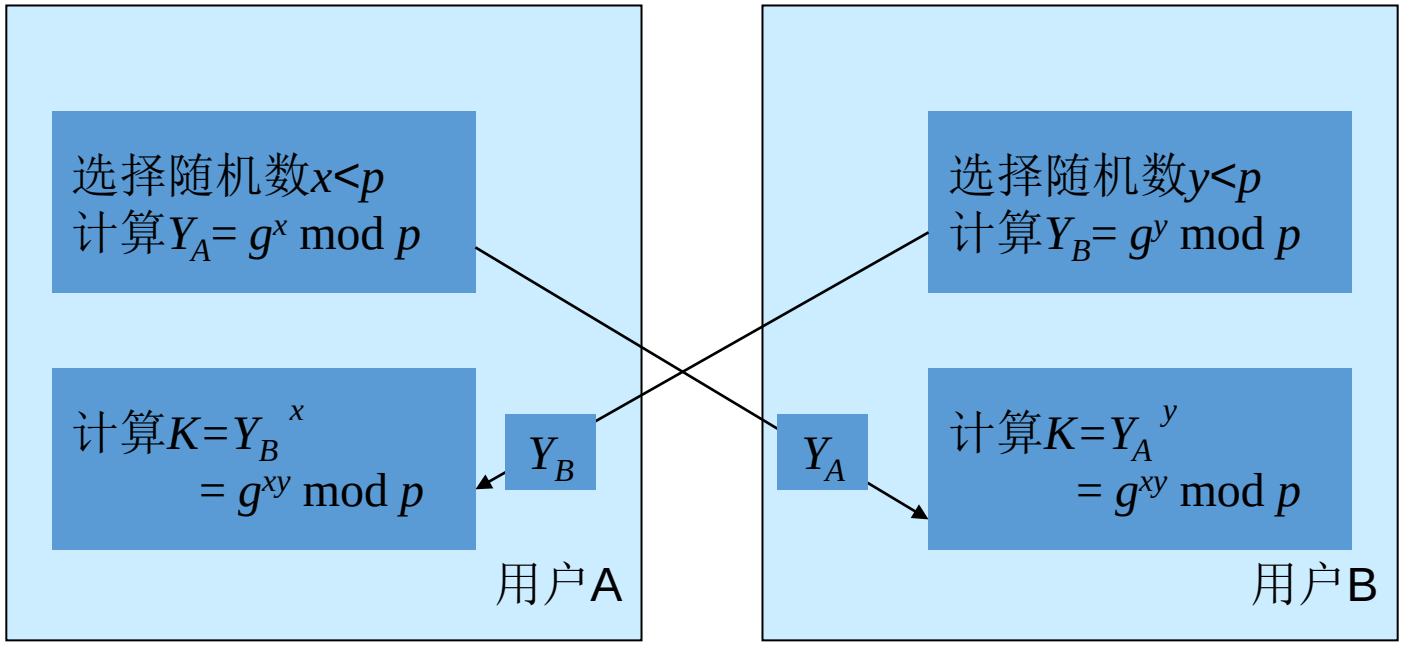
第七章　密码协议

复习要点

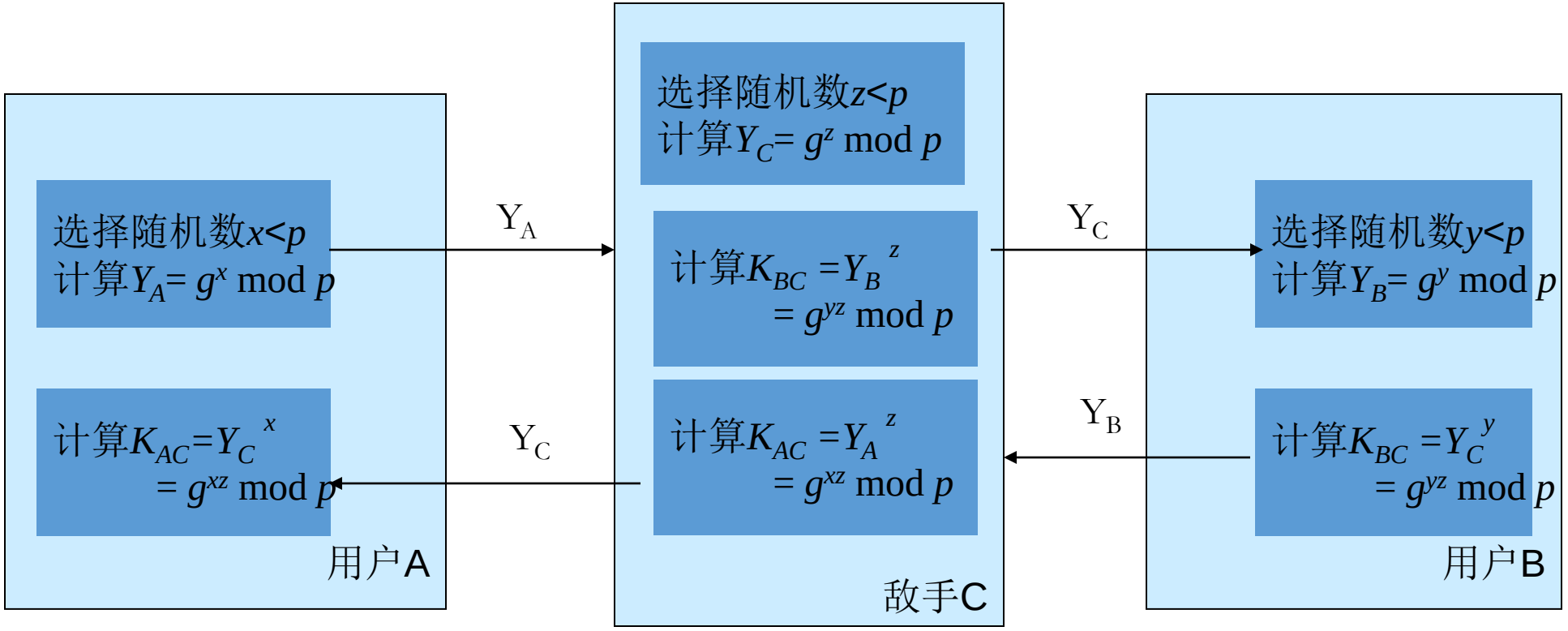
- Diffie-Hellman密钥交换原理及安全性
- Shamir门限方案
- Kerberos协议
- 一次性口令协议

Diffie-Hellman密钥交换

- W.Diffie和M.Hellman1976年提出
- 算法的安全性基于求离散对数的困难性



中间人攻击


选择随机数x<p
计算Y_A= g^x mod p
计算K_{AC}=Y_C^x
= g^{xz} mod p
用户A
Y_A
Y_C
选择随机数z<p
计算Y_C= g^z mod p
计算K_{BC} =Y_B^z
= g^{yz} mod p
计算K_{AC} =Y_A^z
= g^{xz} mod p
敌手C
Y_C
Y_B
选择随机数y<p
计算Y_B= g^y mod p
计算K_{BC} =Y_C^y
= g^{yz} mod p
用户B

门限方案的一般概念

- 秘密s被分为n个部分,每个部分称为shadow,由一个参与者持有，使得
- 由k个或多于k个参与者所持有的部分信息可重构s。
- 由少于k个参与者所持有的部分信息则无法重构s。

- 称为(k,n)秘密分割门限方案，k称为门限值。
- 少于k个参与者所持有的部分信息得不到s的任何信息称该门限方案是完善的。

- 门限方案由份额分配算法和恢复算法构成

Shamir门限方案

- 在有限域GF(q)上的Shamir秘密分割门限方案
- （1）秘密的分割
  - 设GF(q)是一有限域，其中q是一个大素数，满足$q \geq n+1$
  - 秘密s是在GF(q)\{0}上均匀选取的一个随机数，表示为$s \in_R \mathrm{GF}(q)\backslash\{0\}$
  - 令s等于常系数a_0
  - 其它k-1个系数$a_1, a_2, \ldots, a_{k-1}$的选取也满足$a_i \in_R \mathrm{GF}(q)\backslash\{0\}$ (i=1,…,k-1)
  - 在GF(q)上构造一个k-1次多项式$f(x) = a_0 + a_1 x + \ldots + a_{k-1} x^{k-1}$
  - n个参与者记为$P_1, P_2, \ldots, P_n$，其中P_i分配到的子密钥为（i, $f(i)$）

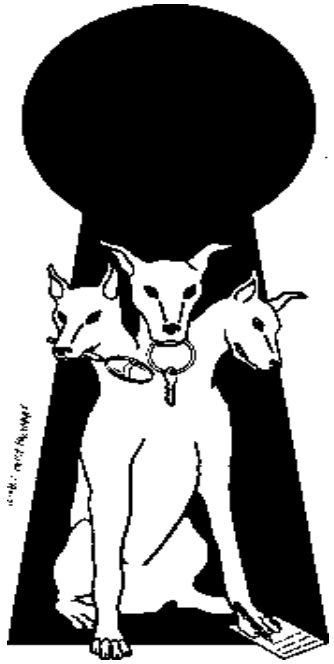
Shamir门限方案

- (2) 秘密的恢复
  - 如果任意k个参与者$P_{i1},P_{i2},\ldots,P_{ik}(1\le i_1<i_2<\ldots<i_k\le n)$要想得到秘密$s$，可计算

$$f(x)=\sum_{j=1}^{k} f(i_j)\prod_{\substack{l=1\\ l\neq j}}^{k}\frac{(x-i_l)}{i_j-i_l}\pmod q$$

$$s = f(0) = (-1)^{k-1}\sum_{j=1}^{k} f(i_j)\prod_{\substack{l=1\\ l\neq j}}^{k}\frac{i_l}{i_j - i_l}\pmod q$$

KERBEROS认证协议



- 在希腊神话中，有一个多头的狗，它有一条毒蛇的尾巴，它是地狱之门的守护者。

Kerberos

- 是美国麻省理工学院（MIT）开发的一种身份鉴别服务。
- “Kerberos”的本意是希腊神话中守护地狱之门的守护者。
- Kerberos提供了一个集中式的认证服务器结构，认证服务器的功能是实现用户与其访问的服务器间的相互鉴别。
- Kerberos建立的是一个实现身份认证的框架结构。
- 其实现采用的是对称密钥加密技术，而未采用公开密钥加密。
- 公开发布的Kerberos版本包括版本4和版本5。

Kerberos设计目标

- 安全性
  - 能够有效防止攻击者假扮成另一个合法的授权用户。
- 可靠性
  - 分布式服务器体系结构，提供相互备份。
- 对用户透明性
- 可伸缩
  - 能够支持大数量的客户和服务器。

Kerberos设计思路

- 基本思路：
  - 使用一个（或一组）独立的认证服务器（AS —— Authentication Server），来为网络中的用户（C）提供身份认证服务；
  - 认证服务器（AS），用户口令由 AS 保存在数据库中；
  - AS 与每个服务器（V）共享一个惟一保密密钥（K_v）（已被安全分发）。
- 会话过程：

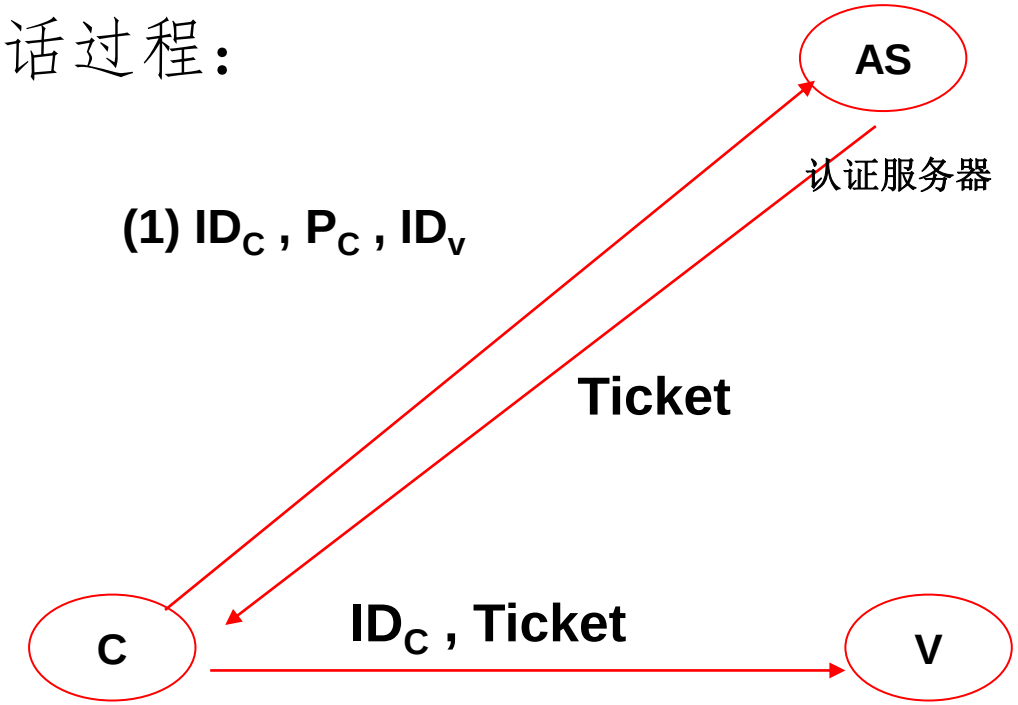
(1) $C \rightarrow AS$: $ID_C \| P_C \| ID_v$
(2) $AS \rightarrow C$: Ticket
(3) $C \rightarrow V$: $ID_C \| Ticket$

其中：

$Ticket = E_{Kv}[ID_C \| AD_C \| ID_v]$

Kerberos设计思路

- 会话过程：



搜索数据库看用户是否合法

如果合法，验证用户口令是否正确

如果口令正确，检查是否有权限访问服务器V

用与AS共享密钥解密票据

检查票据中的用户标识与网络地址是否与用户发送的标识及其地址相同

如果相同，票据有效，认证通过

Ticket = $E_{Kv}[ID_C , AD_C , ID_v]$ 用户

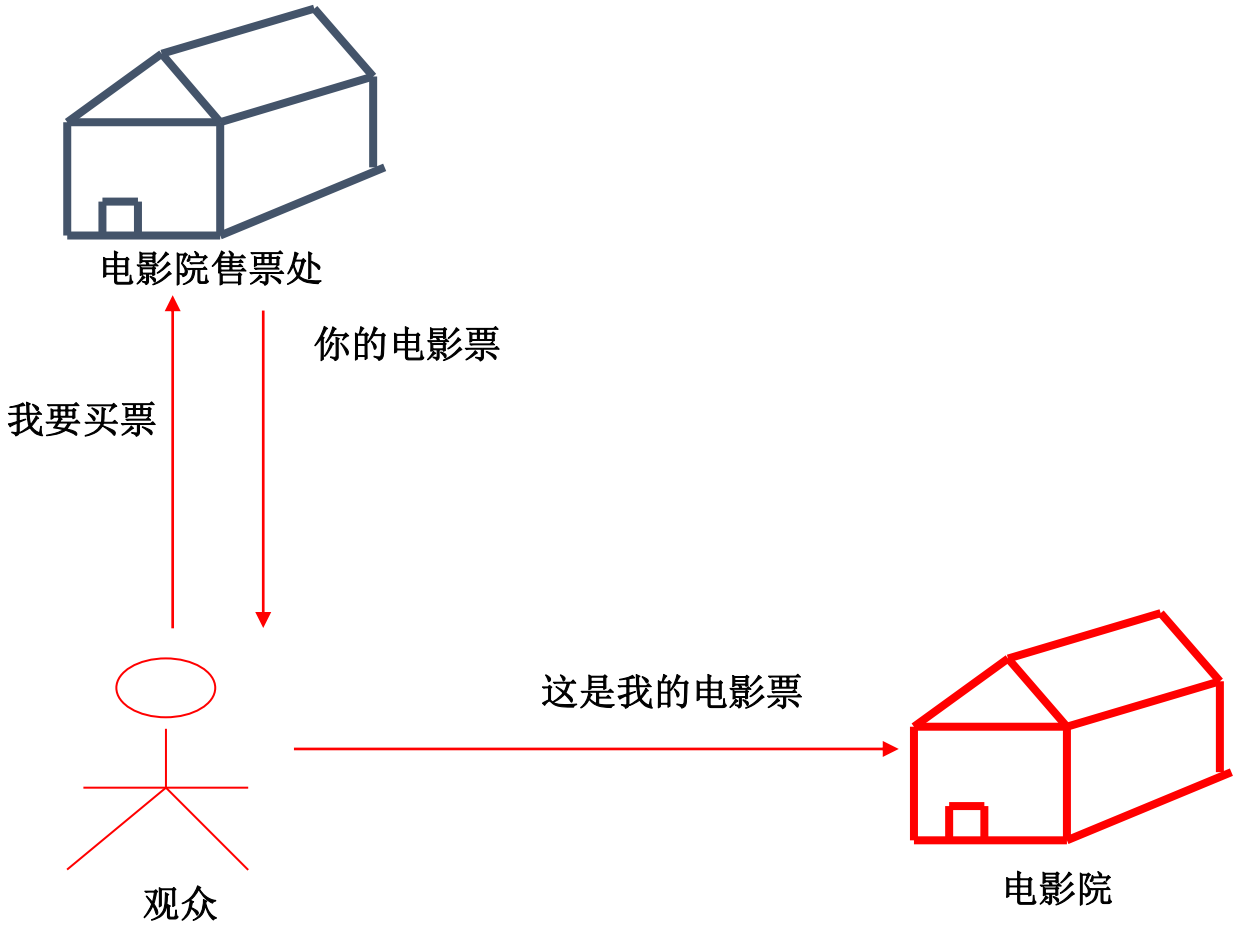
ID_C：用户C的标识

P_C ：用户口令

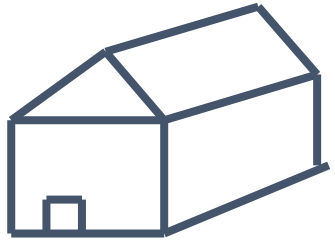
ID_v：服务器标识

AD_C：用户网络地址

Kerberos设计思路



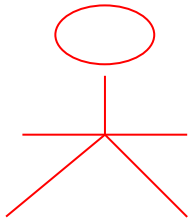
Kerberos设计思路



电影院售票处

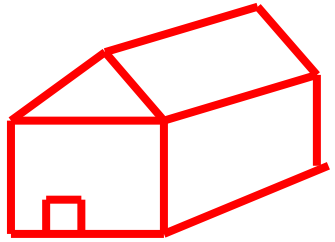
你的电影票

我要买票，
这是我的信
用卡密码



观众

这是我的电影票



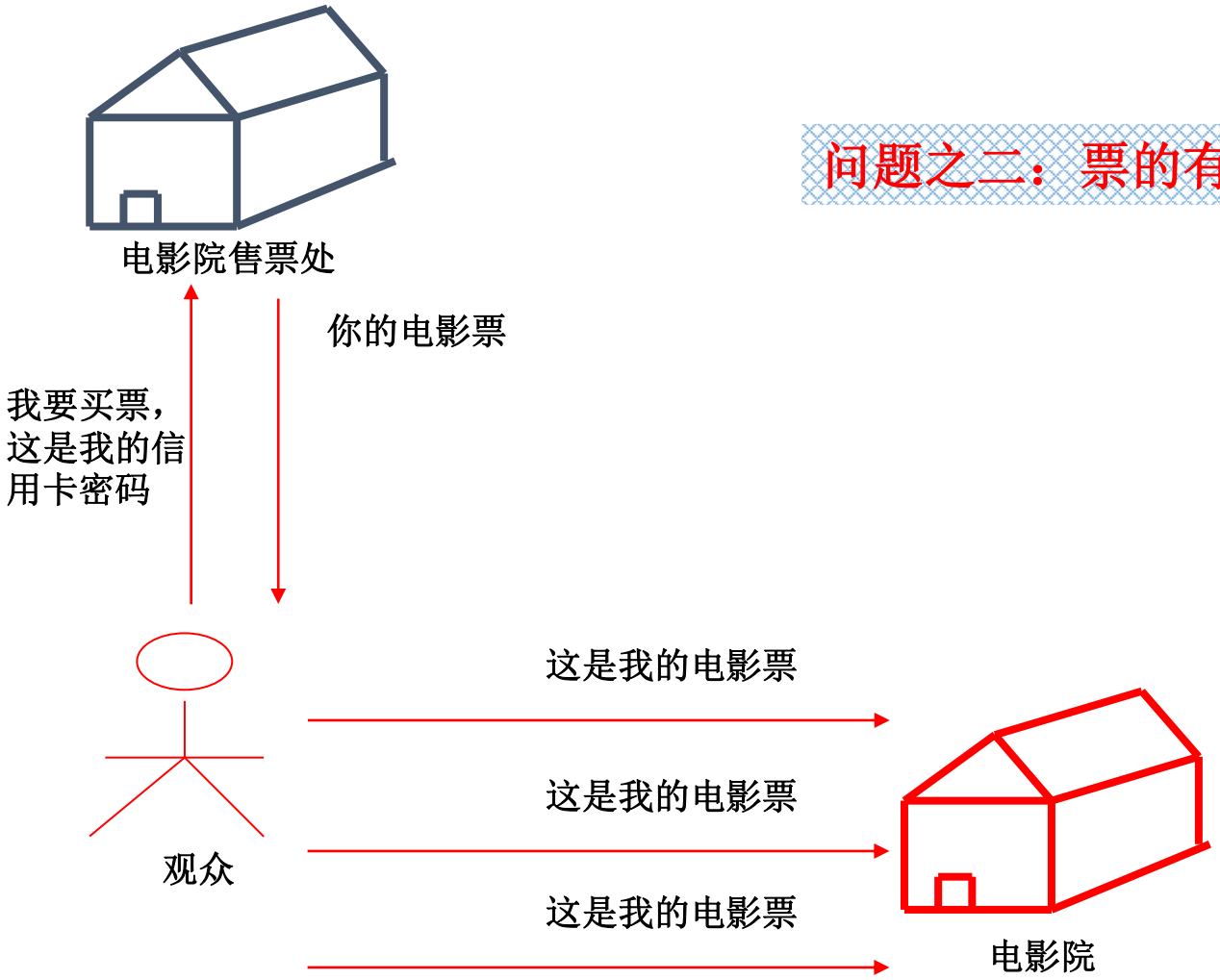
电影院

问题：如何买票

答案：出示信用卡卡号和密码

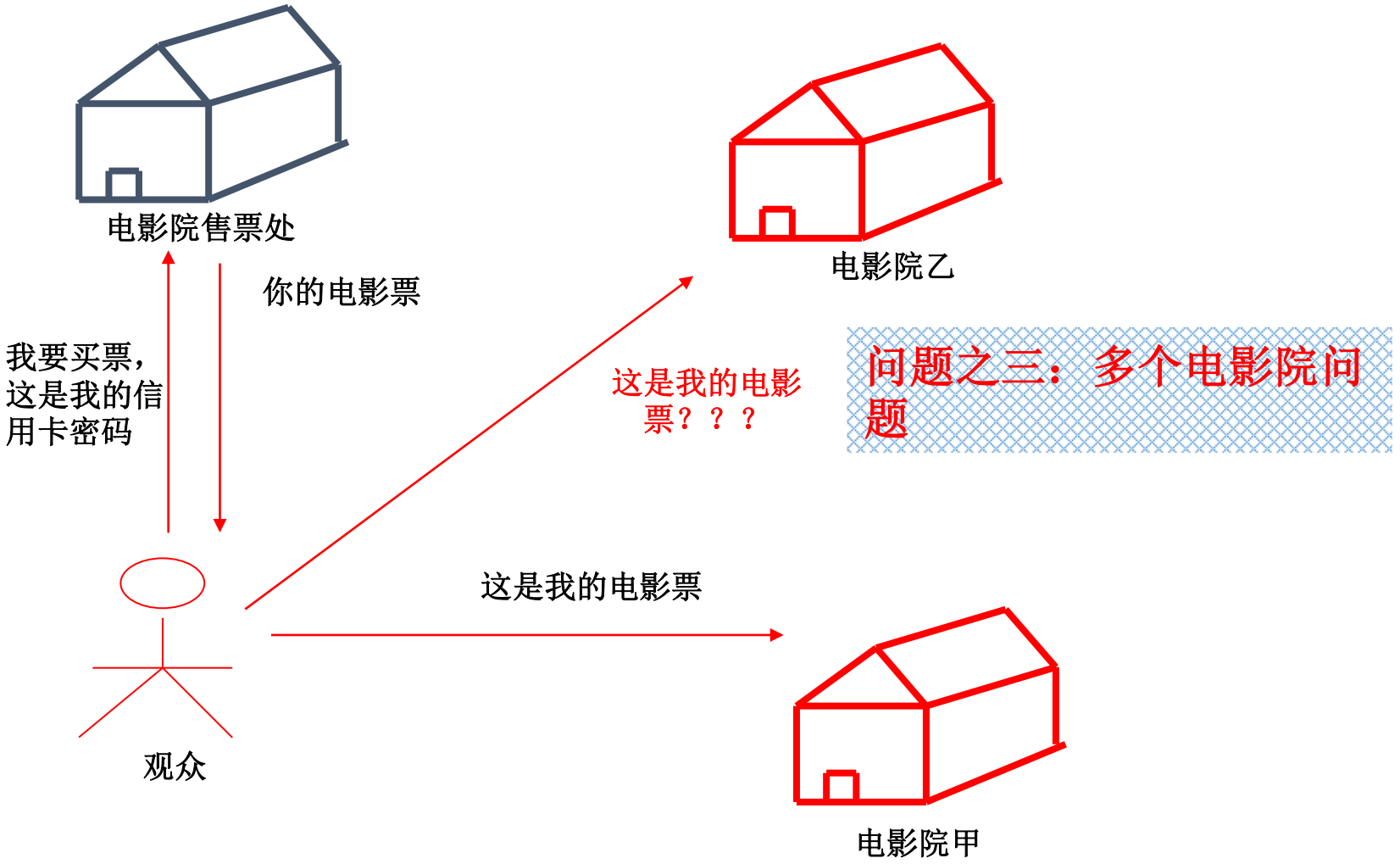
问题之一：信用卡问题

Kerberos设计思路



问题之二：票的有效期问题

Kerberos设计思路



Kerberos设计思路

上述协议问题？

上述协议的问题：

（1）口令明文传送

（2）票据的有效性（多次使用）

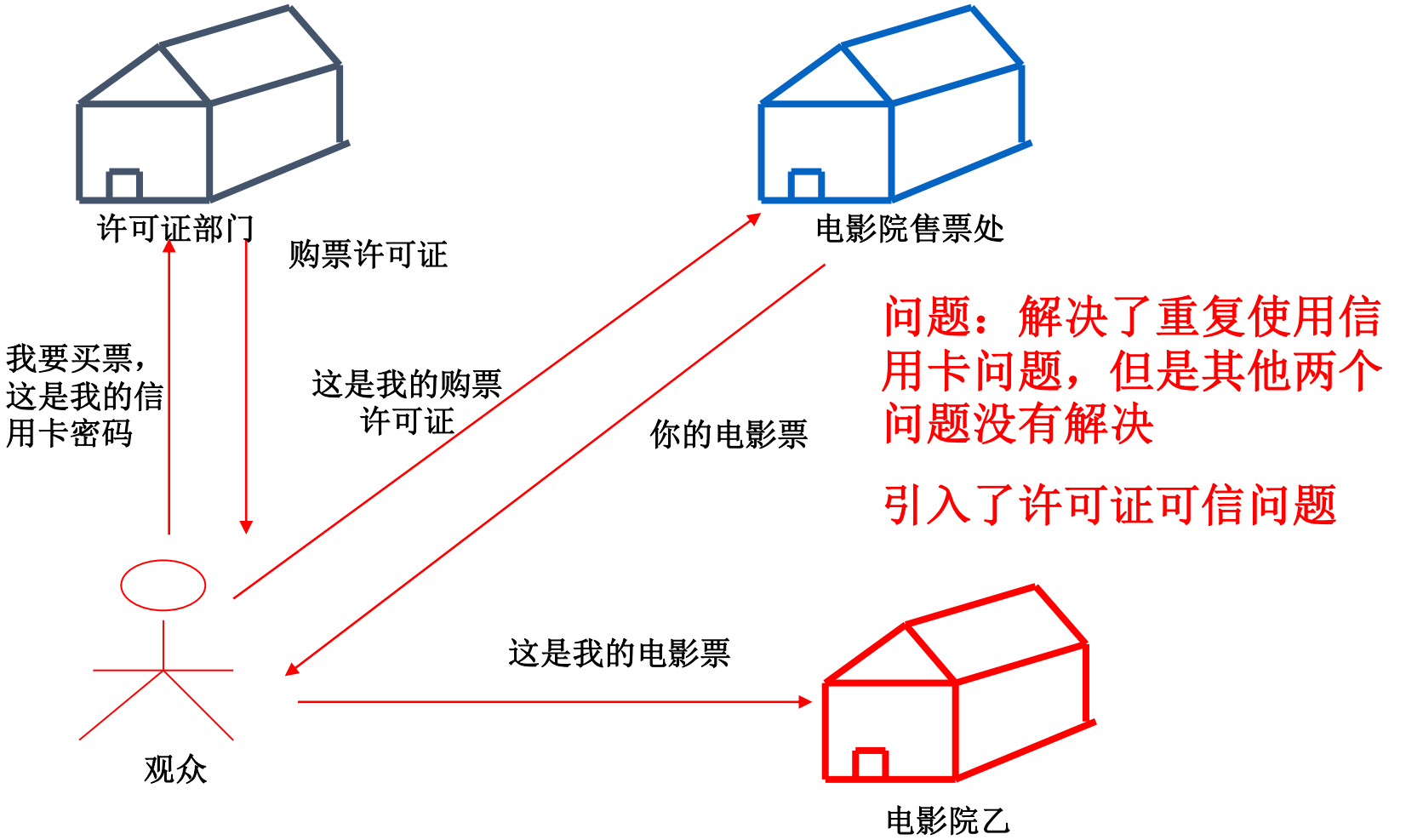
（3）访问多个服务器则需多次申请票据（即口令多次使用）

如何解决?

Kerberos设计思路

- 问题：
  - 用户希望输入口令的次数最少。
  - 口令以明文传送会被窃听。
- 解决办法
  - 票据重用（ticket reusable）。
  - 引入票据许可服务器（TGS - ticket-granting server）
    - **用于向用户分发服务器的访问票据；**
    - **认证服务器 AS 并不直接向客户发放访问应用服务器的票据，而是由 TGS 服务器来向客户发放。**

Kerberos设计思路



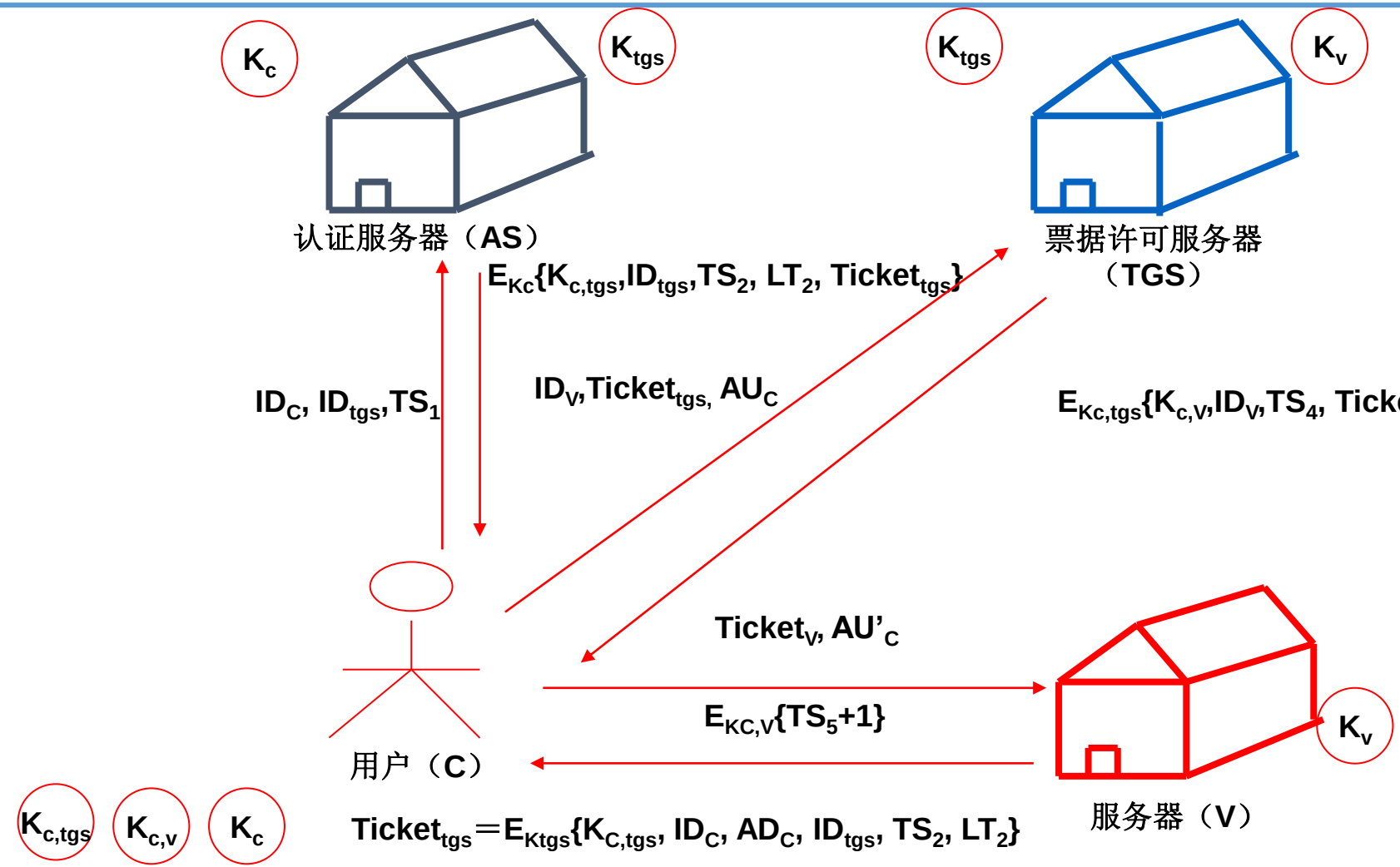
问题：解决了重复使用信用卡问题，但是其他两个问题没有解决

引入了许可证可信问题

Kerberos设计思路

- 两种票据
  - 票据许可票据（Ticket granting ticket）
    - 客户访问 **TGS** 服务器需要提供的票据，目的是为了申请某一个应用服务器的“服务许可票据”；
    - 票据许可票据由 **AS** 发放；
    - 用 $\mathbf{Ticket_{tgs}}$ 表示访问 **TGS** 服务器的票据；
    - $\mathbf{Ticket_{tgs}}$ 在用户登录时向 **AS** 申请一次，可多次重复使用；
  - 服务许可票据（Service granting ticket）
    - 是客户时需要提供的票据；
    - 用 $\mathbf{Ticket_V}$ 表示访问应用服务器 **V** 的票据。

Kerberos设计思路



$Ticket_{tgs}=E_{Ktgs}\{K_{C,tgs}, ID_C, AD_C, ID_{tgs}, TS_2, LT_2\}$

$AU_C=E_{KC,tgs}\{ID_C, AD_C, TS_3\}$　　$AU'_C=E_{Kcv}\{ID_C, AD_C, TS_5\}$

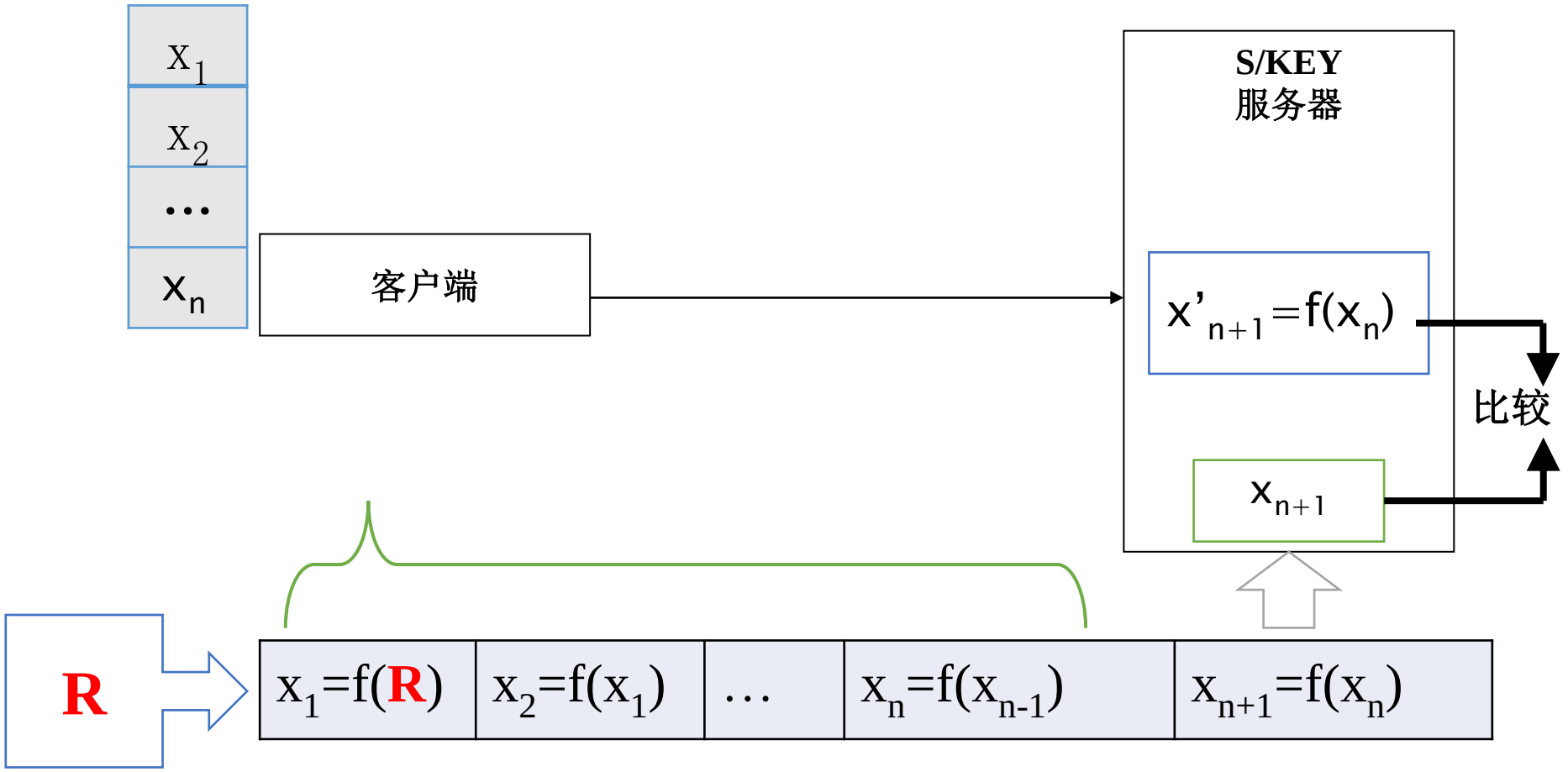
$Ticket_V=E_{KV}\{K_{C,V}, ID_C, AD_C, ID_V, TS_4, LT_4\}$

一次性口令的认证机制

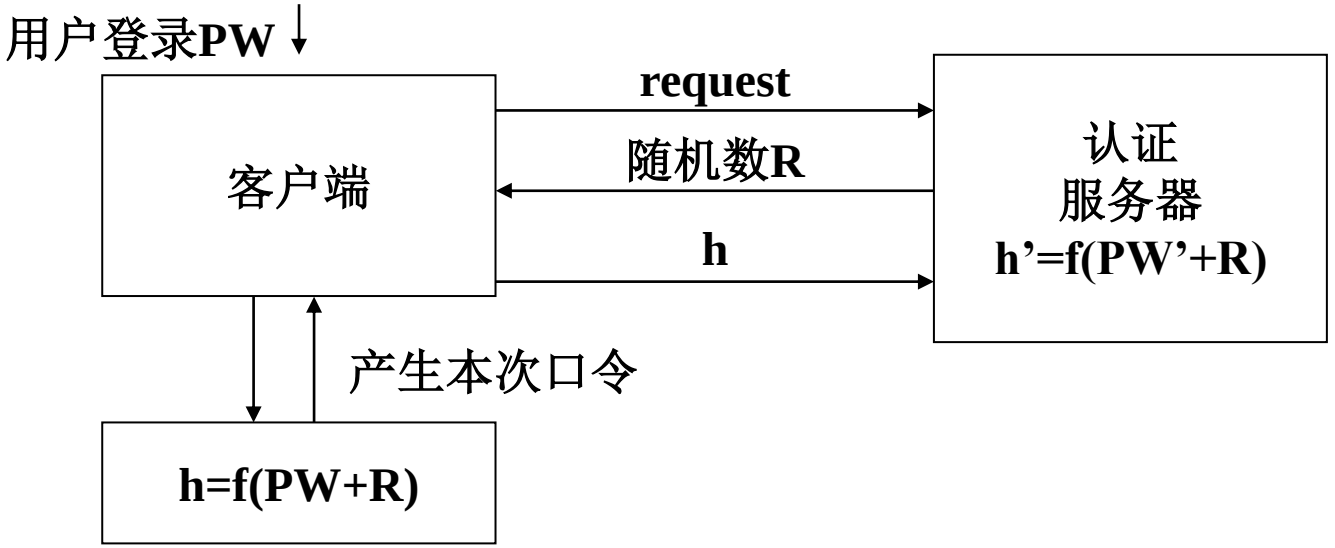
对抗重放攻击——一次性口令

- 在登录过程中加入不确定因素，使每次登录过程中传送的信息都不相同
- 不确定口令的方法：
  - 口令序列
  - 挑战/回答
  - 时间戳

口令序列S/KEY



挑战/回答



- 类似加盐，但每次认证盐不同

时间戳

- 以用户登录时间作为随机因素，如：
  - 用户计算，登录口令=hash(用户名＋口令 ＋时间)
  - 系统验证， hash(用户名＋口令 ＋时间)
- 要求双方较高时间同步准确度，一般采取以分钟为时间单位的折中办法。

感 谢 聆 听!

liaoyj@uestc.edu.cn