

密码技术基础

本章参考书:

Applied Cryptography, Protocols, algorithms, and source code in C (2nd Edition), Bruce Schneier 著

应用密码学 — 协议、算法与C源程序, 吴世忠、祝世雄、张文政等译

Cryptography and Network Security, Atul Kahate

LOGO



-
- 密码学在信息网络安全中的作用
 - 密码学基本概念
 - 密码模式
 - 对称加密
 - 公钥密码学
 - 单向散列函数
 - 数字签名
 - 密钥管理



密码学在信息网络安全中的作用



违反安全性的例子：

(1) 用户**A**传输一个文件到用户**B**，该文件包含了敏感的数据，这样数据必须加以保护以防泄密。没有被授权读取该文件的用户**C**可能监视该传输过程，并在传输过程中截取了该副本。 (**机密性**)



(2) 某网络管理员**D**在其管理下向一台计算机**E**传输一条消息，该消息指示计算机**E**更新一个授权文件，该文件包含了能够访问该计算机的一些新用户标识符。用户**F**中途截取了该消息，并且增加和删除一些项从而改变了该消息，然后将该消息转发给**E**。计算机**E**以为该消息是从管理者**D**接收的，因而更新了这个授权文件。 (**完整性**)

(3) 用户**F**并没有中途阻止某消息，用户**F**构造了具有它自己希望内容的消息，并将该消息传输给**E**，好象该消息来自于管理员**D**。计算机**E**接收了以为来自于管理者**D**的消息并更新了它的授权文件。 (**鉴别性**)

(4) 一个客户向一个股票代理商发出带有多个交易指示的消息。随后，该投资跌值，而该客户不承认发送了该消息。 (**抗抵赖性**)

思考：密码学不能解决什么问题？



密码学基本概念

- 密码学(cryptology)包括两个方面：密码编码学 (Cryptography) 和密码分析学 (Cryptanalytics)。
- 密码编码学就是研究对数据进行变换的原理、手段和方法的技术和科学。
- 密码分析学是为了取得秘密的信息，而对密码系统及其流动的数据进行分析，是对密码原理、手段和方法进行分析、攻击的技术和科学。



密码学基本概念Terminology



- 明文**Plain text, clear text** : 需要秘密传送的消息。
- 密文**Cipher text**: 明文经过密码变换后的消息。
- 加密**Encryption** : 由明文到密文的变换。
- 解密**Decryption**: 从密文恢复出明文的过程。
- 破译**Cryptanalysis** : 非法接收者试图从密文分析出明文的过程。
- 加密算法**Encryption Algorithm** : : 对明文进行加密时采用的一组规则。
- 解密算法**Decryption Algorithm**: 对密文进行解密时采用的一组规则。
- 密钥**Key**: 加密和解密时使用的一组秘密信息。

密码学基本概念

- 密码系统:

一个密码系统可以用以下数学符号描述:

$$S = \{P, C, K, E, D\}$$

P = 明文空间

C = 密文空间

K = 密钥空间

E = 加密算法

D = 解密算法

- 当给定密钥 $k \in K$ 时, 加解密算法分别记作 E_k 、 D_k , 密码系统表示为

$$S_k = \{P, C, k, E_k, D_k\}$$

$$C = E_k(P)$$

$$P = D_k(C) = D_k(E_k(P))$$



加密方案的安全性

■ 安全性体现在：

- 破译的**成本超过加密信息**的价值
- 破译的**时间超过该信息有用**的生命周期
- **无条件安全**：若密文中不含明文的任何信息，则认为该密码体制是安全的，否则就认为是不安全的。无论提供的密文有多少，由一个加密方案产生的密文中包含的信息不足以唯一地决定对应的明文
- 已经证明（**shannon**），达到这样高等级（完善）的安全性，仅当所用密钥的个数不少于可能的发送消息的数目才有可能。除了一次一密的方案外，没有无条件安全的算法

参考书: p4

Table 1.1: Classification of attacks

Type of Attack	Information assumed to be available to cryptanalyst
Ciphertext only	Encryption Algorithm Ciphertext to be decoded.
Known plaintext	Encryption Algorithm Ciphertext to be decoded One or more plaintext-ciphertext pairs.
Chosen plaintext	Encryption Algorithm Ciphertext to be decoded Plaintext chosen by cryptanalyst & corresponding ciphertext.
Chosen ciphertext	Encryption Algorithm Ciphertext to be decoded Ciphertext chosen by cryptanalyst & corresponding decrypted plaintext.
Chosen text	Encryption Algorithm Ciphertext to be decoded Plaintext chosen by cryptanalyst & corresponding ciphertext Ciphertext chosen by cryptanalyst & corresponding decrypted plaintext.

补充: Side channel attack

Hera He, Side Channel Cryptanalysis Using Machine Learning Using an SVM to recover DES keys from a smart card

密码安全性

- 选择密文攻击(IND-CCA)游戏. 敌手和受挑战者进行如下交互:
 - (1) 受挑战者对加密方案进行系统建立,输出公私钥对,并将公钥交给敌手
 - (2)敌手可以向受挑战者进行一些解密询问,受挑战者解密密文,返回结果给敌手
 - (3) 敌手选择两个明文 M_0, M_1 ,然后发送给受挑战者.受挑战者投掷一个公平硬币 $b \in \{0, 1\}$,对明文 M_b 加密,得到密文 C^* 并发送给敌手;
 - (4)敌手可以继续向受挑战者进行一些同步骤(2)中的解密询问,但询问密文不能为 C^* ;
 - (5) 受挑战者必须回答 0 或者 1(记为 b'),作为对密文 C^* 的猜测. 若 $b' = b$, 则敌手在该游戏中获胜.
 - 敌手在游戏中的优势定义为 $\Pr[b' = b] - 1/2$. 若上面的交互中,敌手不能进行任何解密询问,则此游戏称为选择明文攻击(IND-CPA)游戏.



- 对于一个加密方案,如果任意概率多项式时间 (PPT) 的敌手在上述游戏中的优势是可忽略的,则称该加密方案是 **IND-CCA**安全,简称 **CCA** 安全. 对应选择明文攻击游戏,称为 **IND-CPA** 安全,简称 **CPA** 安全.
- **CPA** 安全是公钥加密机制的最基本要求, **CCA** 安全是公钥加密机制更强的安全性要求.



● 算法的选择:

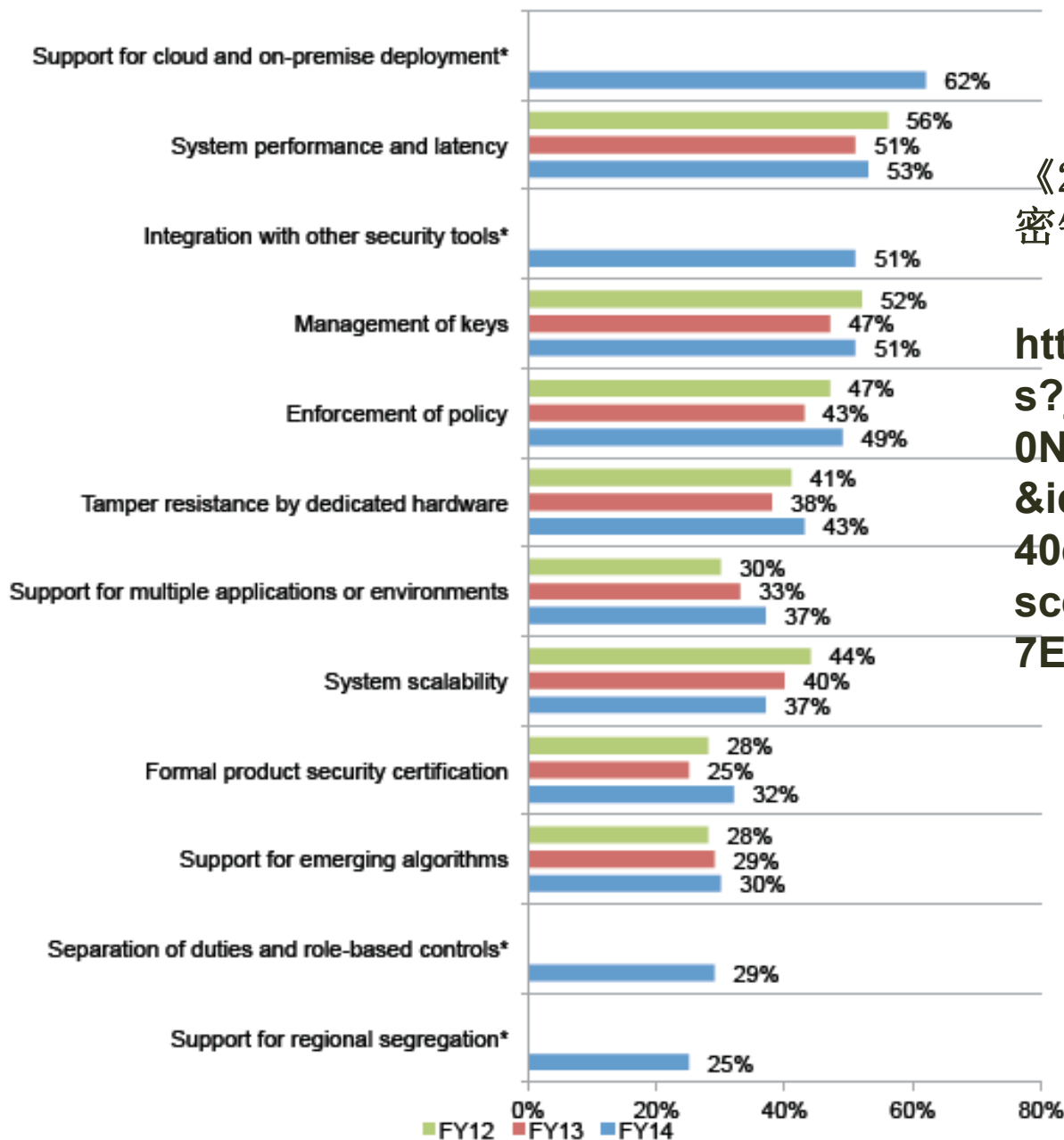
- Degree of security
- Speed: 加密与解密运算速度
- Key length: 关系到
 - Key的存储空间, 如移动设备
 - 算法的安全性
 - *Key Space* 密钥空间
- public/Private: 通常公开的算法, 经过了更多的测试
- 专利与出口限制问题

Figure 16. Most important features of encryption technology solutions

Very important response

More than one choice permitted

*Historic data is not available



《2015年全球加密技术与
密钥管理趋势研究》

http://mp.weixin.qq.com/s?__biz=MjM5Mzg0NTU0NQ==&mid=208377144&idx=1&sn=e382930f15540e0c92a7b2f9cf77af34&scene=5&srcid=RT0TG7ELNazQqYLekEq#rd

密码学发展历程

- 密码学应用：需求的变化
 - 军用，民用
 - 计算机通信应用
 - 机密性，完整性检查，认证，签名
- 密码算法分类
 - 受限制的 (**restricted**) 算法: 算法的保密性基于保持算法的秘密。
 - 基于密钥 (**key-based**) 的算法: 算法的保密性基于对密钥的保密。

密钥的生成、注入、存储、管理、分发的复杂性

密码学发展历程

- 密码学是一门古老的科学，大概自人类社会出现战争便产生了密码，以后逐渐形成一门独立的学科。密码学的发展历史大致可以分为三个阶段：
 - 在1949年之前，是密码发展的第一阶段——古典密码体制。古典密码体制是通过某种方式的文字置换进行，这种置换一般是通过某种手工或机械变换方式进行转换，同时简单地使用了数学运算。虽然在古代加密方法中已体现了密码学的若干要素，但它只是一门艺术，而不是一门科学。

古典密码学



- 已经成为历史，但被**传统密码学**所借鉴；
- 加解密都很简单，易被攻破；
- 属于**对称密钥算法**；
- 包括置换密码、代换密码等

置换密码

● 古典密码

● 置换密码

- 用加密置换去对消息进行加密

- 举例：

- 加密算法 $E = (2, 1, 4, 3)$
- 解密算法 $D = (2, 1, 4, 3)$
- 明文 $M = \text{“置换密码”}$
- 密文 $C = E(M) = \text{“换置码密”}$

● 代换密码

- 明文中的字母用相应的密文字母进行替换
- 单表代换密码
- 多表代换密码

攻击：如何进行密码分析？



代换密码（换位密码）

- 单表代换密码举例


明文: a b c d e f g h i j k l m n o p q r s t u v w x y z
密文: D E F G H I J K L M N O P Q R S T U V W X Y Z A B C

- m = “Caser cipher is a shift substitution”
- c = “FDVH DU FLSHU LV D VKLIW VXE VWLWXWLRO”

🌐美国南北战争时期（1861-1865年），军队中曾使用过的“栅栏”式密码（rail fence cipher）

Helloworld					(明文)
He	ll	Ow	or	ld	(分组)
elwrd		hlool			(重排)
Elwrdhlool					(密文)

密码学发展历程



1883年Kerchoffs第一次明确提出了编码的原则：加密算法应建立在**算法的公开**不影响明文和密钥的安全的基础上。这一原则已得到普遍承认，成为判定密码强度的衡量标准，实际上也成为古典密码和现代密码的分界线。

- 基本特点：加密和解密采用同一个密钥

let C = Cipher text, P = Plain text, k is key, $E()$ / $D()$ is the encryption/decryption function, then

$$C=E(P, k), P=D(C, k)$$

- 基本技术
替换/置换和移位

密码学发展历程

- 从1949年到1975年，是密码学发展的第二阶段。1949年Shannon发表了题为《保密通信的信息理论》的著名论文，把密码学置于坚实的数学基础之上，标志着密码学作为一门学科的形成，这是密码学的第一次飞跃。
- 然而，在该时期密码学主要用在政治、外交、军事等方面，其研究是在秘密地进行，密码学理论的研究工作进展不大，公开的发表的密码学论文很少。



密码学发展历程

- 1976年， **Whitfield Diffie**和**Martin Hellman**在《密码编码学新方向》一文中提出了公开密钥的思想，这是密码学的第二次飞跃。
- 1977年美国数据加密标准（**DES**）的公布使密码学的研究公开，密码学得到了迅速地发展。
- 1994年美国联邦政府颁布的密钥托管加密标准（**EES**）和数字签名标准（**DSS**）以及2001年颁布的高级数据加密标准（**AES**），都是密码学发展史上一个个重要的里程碑。

课外：视频**Information Security—Before & After Public-Key Cryptography**

加密体制分类

- 基于密钥的算法，按照密钥的特点分类：
 - 对称密钥算法 (**symmetric cipher**): 又称传统密码算法 (**conventional cipher**), 就是加密密钥和解密密钥相同, 或实质上等同, 即从一个易于推出另一个。又称秘密密钥算法或单密钥算法。
 - 非对称密钥算法 (**asymmetric cipher**): 加密密钥和解密密钥不相同, 从一个很难推出另一个。又称公开密钥算法 (**public-key cipher**)。• 公开密钥算法用一个密钥进行加密, 而用另一个进行解密。其中的加密密钥可以公开, 又称公开密钥 (**publickey**), 简称公钥。解密密钥必须保密, 又称私人密钥 (**private key**) 私钥。简称私钥。
 - 混合密钥体制

加密体制分类

- 按照明文的处理方法（参考书1 p132):
 - 分组密码 (**block cipher**):将明文分成固定长度的组, 用同一密钥和算法对每一块加密, 输出也是固定长度的密文。计算机软件处理时代的主流。
 - 流密码 (**stream cipher**):又称序列密码.序列密码每次加密一位的明文。序列密码是手工和机械密码时代的主流。
 - 明文 $m=m_1, m_2, \dots, m_k$
 - 随机序列 $k=k_1, k_2, \dots, k_k$
 - 密文 $c_i=m_i \oplus k_i, i=1, 2, \dots, k$
 - 解密过程与加密过程一致, $m_i=c_i \oplus k_i=m_i \oplus k_i \oplus k_i$
 - 序列密码的安全性完全依赖于随机序列的强度。
 - 移位寄存器是产生序列密码的有效方法
 - **Key**的作用, 密钥序列发生器的输出为**key**和函数

密码模式

● 以某个分组密码算法为基础,对任意长度的明文加密的方法

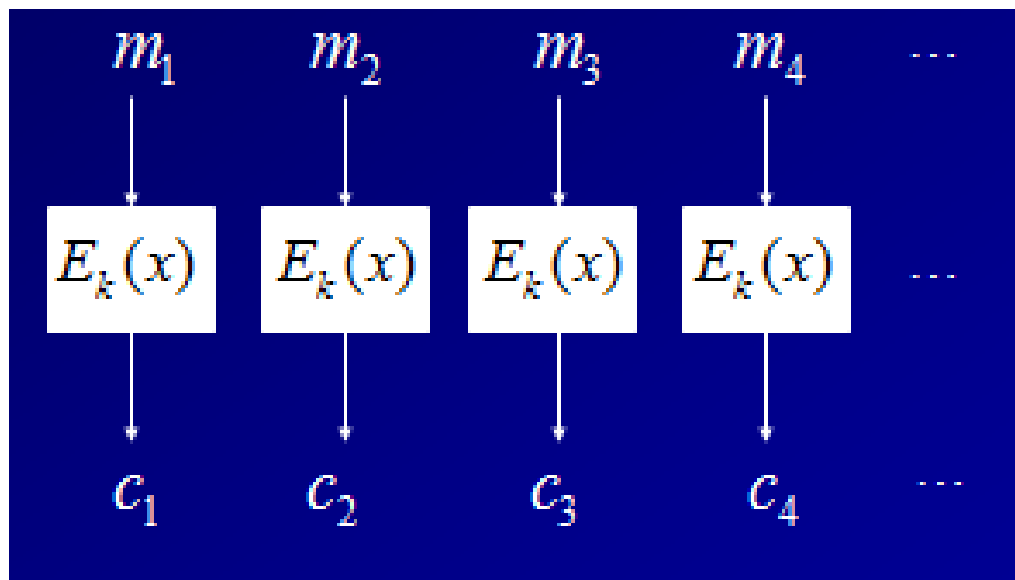
- 电码本**ECB**(**Electronic Code Book**)
- 密码分组链接**CBC**(**Cipher Block Chaining**)
- 密码反馈**CFB**(**Cipher FeedBack**)
- 输出反馈 **OFB**(**Output FeedBack**)
- 计数器模式 (**counter mode**)
- 分组链接**BC** (**Block Chaining**)
- 扩散密码分组链接**PCBC** (**Propagating Cipher Block Chaining**)
- ...

(比较: **P146**)



ECB

- 实现简单
- 不同明文分组的加密可并行处理硬件实现
- 密文中的误码不会影响其它分组的解密
- 无法恢复同步错误
- 相同明文分组对应相同密文分组,因而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。(特别当明文为结构数据时), 需增加完整性检查字段。





例：假设银行A和银行B之间的资金转帐系统所使用报文模式如下：

1	2	3	4	5	6	7	8	9	10	11	12	13
时间 标记	发送银 行	接收银 行	储户姓名1								储户账号1	存款 金额

1	2	3	4	5	6	7	8	9	10	11	12	13
时间 标记	发送银 行	接收银 行	储户姓名2								储户账号2	存款 金额

敌手C通过截收从A到B的加密消息，只要将第5至第12分组替换为自己的姓名和帐号相对应的密文，即可将别人的存款存入自己的帐号。



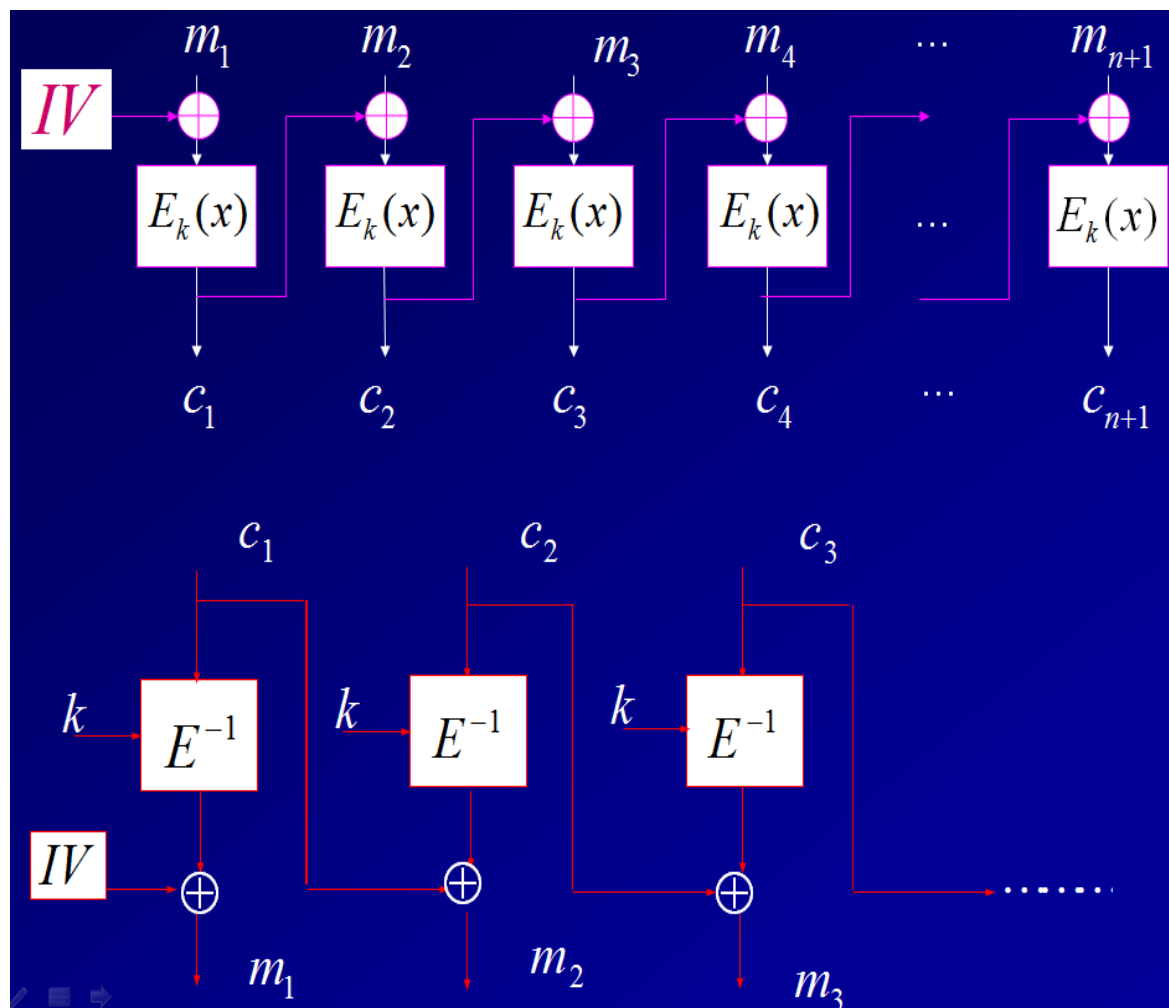
-
- **ECB应用:**
 - 单分组明文的加密
 - 各明文块间无冗余信息: 如随机数
 - 上一例中, 如何消除明文块间的冗余信息?



密码分组链接CBC

加密算法的输入是当前明文组与前一密文组的异或。

加解密如图：
初始化向量 IV 为 c_0



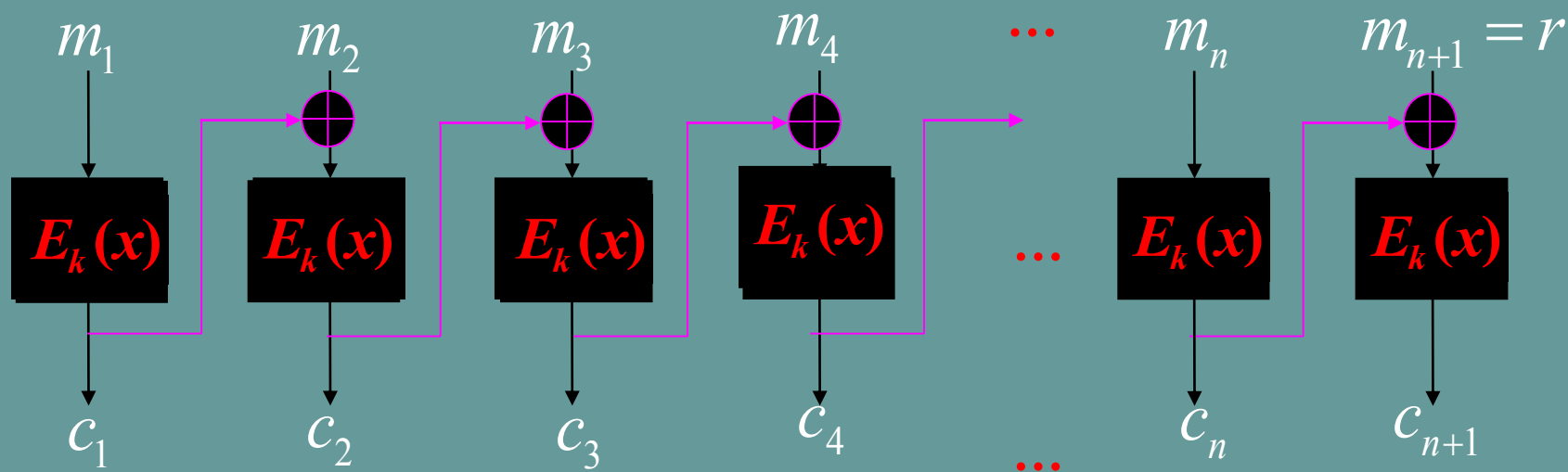


● CBC特点:

- 明文块的统计特性得到了隐蔽。
- 一位密文错误将影响两块分组的解密
- 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.
- 当密文有比特增加和减少时,无法恢复这种同步错误。
- 对于结构数据,需采用明文传送随机数据IV(initialization Vector)作为第一个分组
- 易于完成完整性检查功能



n 分组明文 $m = (m_1, m_2, \dots, m_n)$, 校验码为 $r = f_n(m_n) \oplus f_{n-1}(m_{n-1}) \oplus \dots \oplus f_1(m_1)$



- (1) 仅需对明文验证,而不需加密时,传送明文 m 和验证码 C_{n+1} ,
此时也可仅保留 C_{n+1} 的 t 个比特作为认证码;
- (2) 既需对明文验证,又需要加密时,传送密文 C 和验证码 C_{n+1}

任一 m_i 的更改都会使 C_{n+1} 改变



应用：如何防止电脑彩票的伪造问题。

方法：

(1) 选择一个分组密码算法和一个认证密钥，将他们存于售票机内；

(2) 将电脑彩票上的重要信息，如彩票期号、彩票号码、彩票股量、售票单位代号等重要信息按某个约定的规则作为彩票资料明文；

(3) 对彩票资料明文扩展一个**校验码**分组后，利用认证密钥和分组密码算法对之加密，并将得到的最后一个分组密文作为认证码打印于彩票上面；

认证过程：

执行(3)，并将计算出的认证码与彩票上的认证码比较，二者一致时判定该彩票是真彩票，否则判定该彩票是假彩票。





短块处理方法----直接扩充法



在电码本ECB模式和密码分组链接CBC模式中, 都要求明文长度是明文分组规模的整数倍. 否则就会出现最后一个明文分组是短块的情形. 这时应如何处理呢?



方法1: 对明文扩充, 使最后一个分组不是短块, 但需在文件头或最后一个明文分组中指明文件所含的字节数.

(A) 添充全0比特或其它固定比特, 或计算机内存中自然存放的数据.

(B) 添充随机数.

相对而言, 方法(A)简单, 易实现, 但安全性没有第二种方法好.



密码反馈 (CFB-Cipher Feedback) 模式



CBC模式，整个数据分组需要接收完后才能进行加密。



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 j 比特处理的CFB模式为 j 比特CFB模式。

适用范围：

适用于每次处理 j 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。

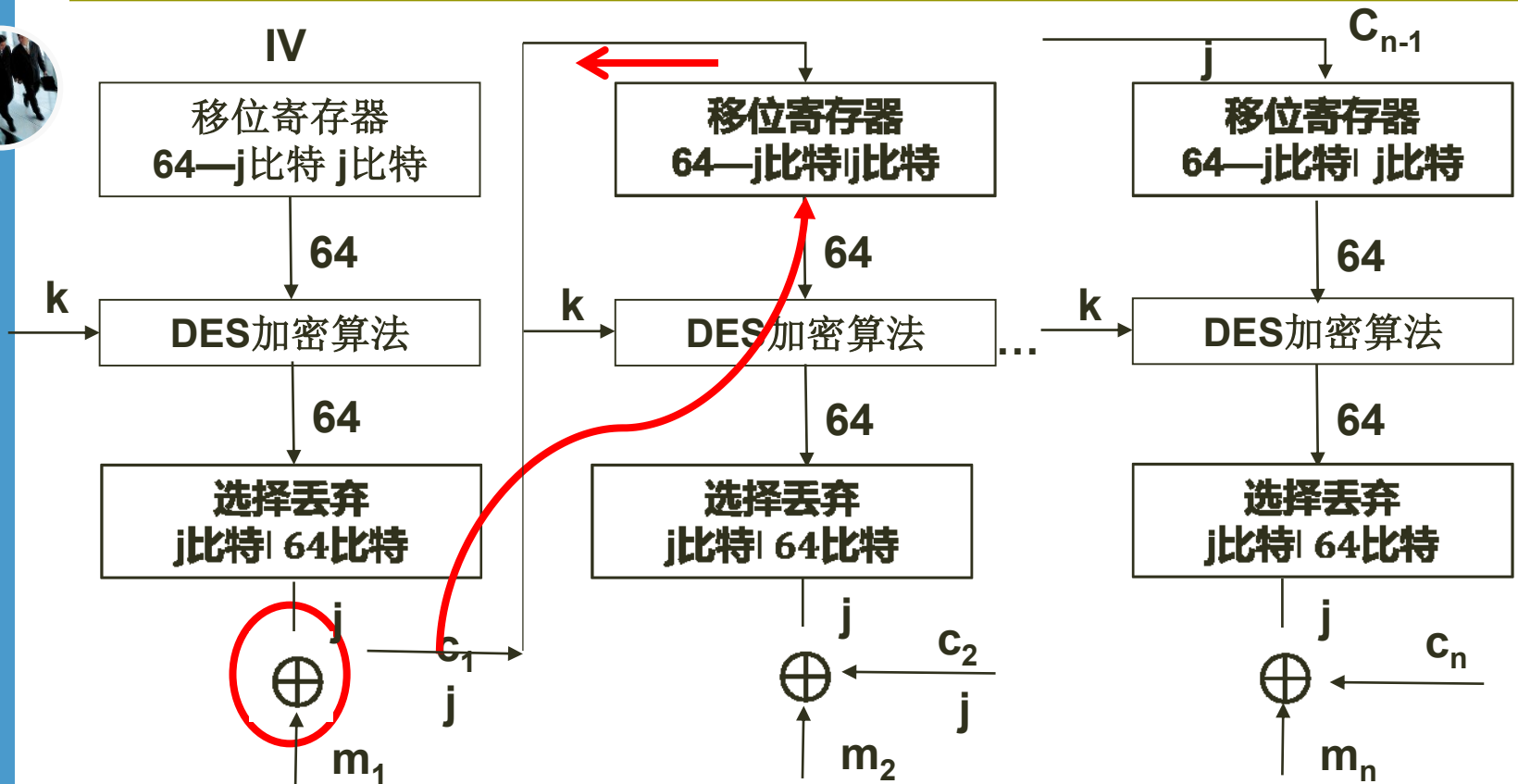
例如，数据库加密要求加密时不能改变明文的字节长度，这时就要以明文字节为单位进行加密。



若记 $IV=c_{-l+1}\dots c_{-1}c_0$, $|c_i|=j$, 则加密过程可表示为:



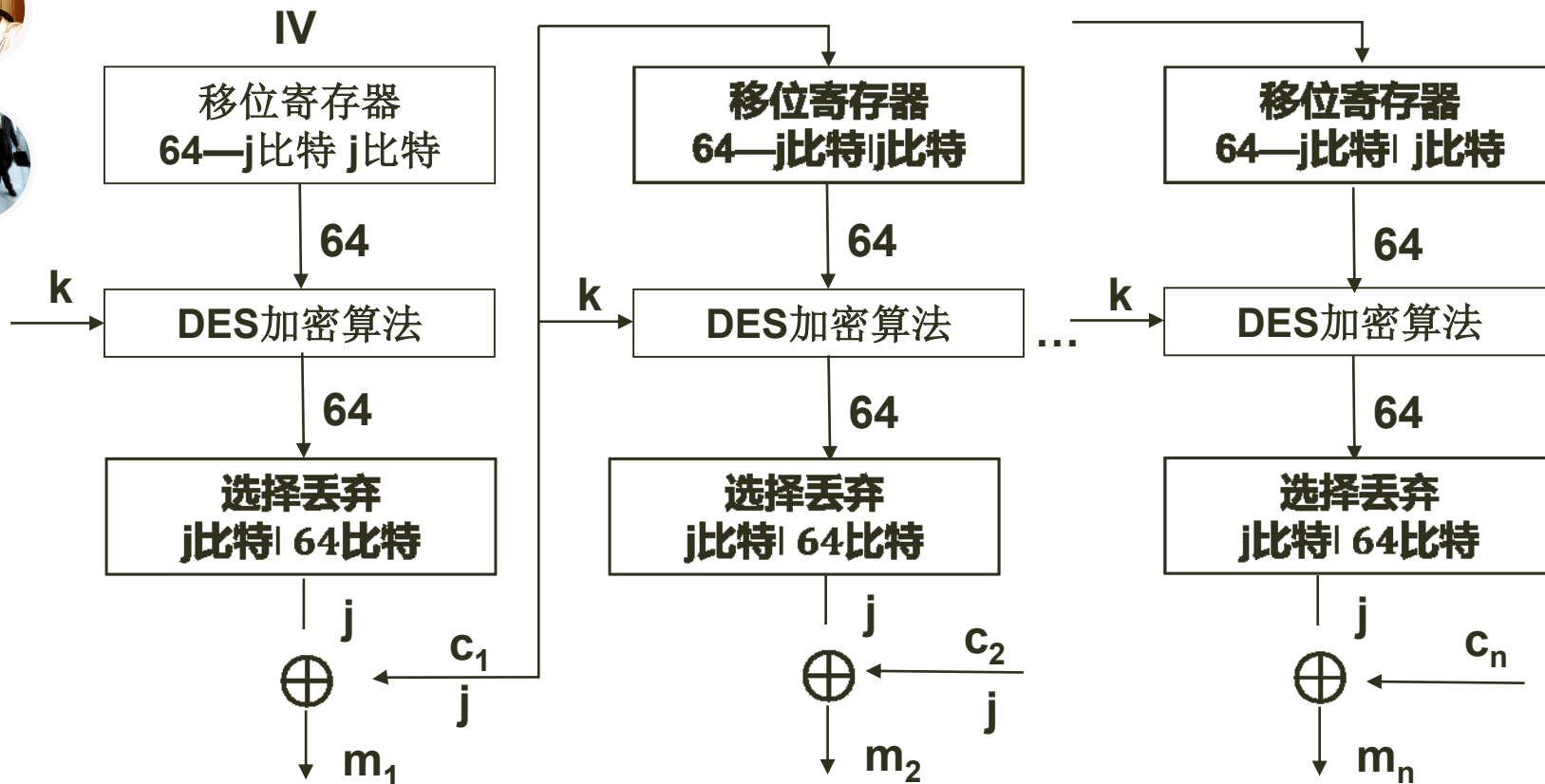
$$c_i = m_i \oplus \text{left}_j(E_k(c_{i-l} \parallel \text{IV}))$$



64位分组算法下的j位CFB加密框图



密文反馈模式的解密





优点:

- (1) 适用于每次处理 j 比特明文块的特定需求的加密情形;
- (2) 具有有限步的错误传播, 可用于完整性认证;
- (3) 可实现自同步功能:

该工作模式**本质上是将分组密码当作序列密码使用的一种方式**, DES 分组加密并不直接对明文加密, 它产生的乱数 j 可作为流加密的 **key**!

缺点: 加密效率低。

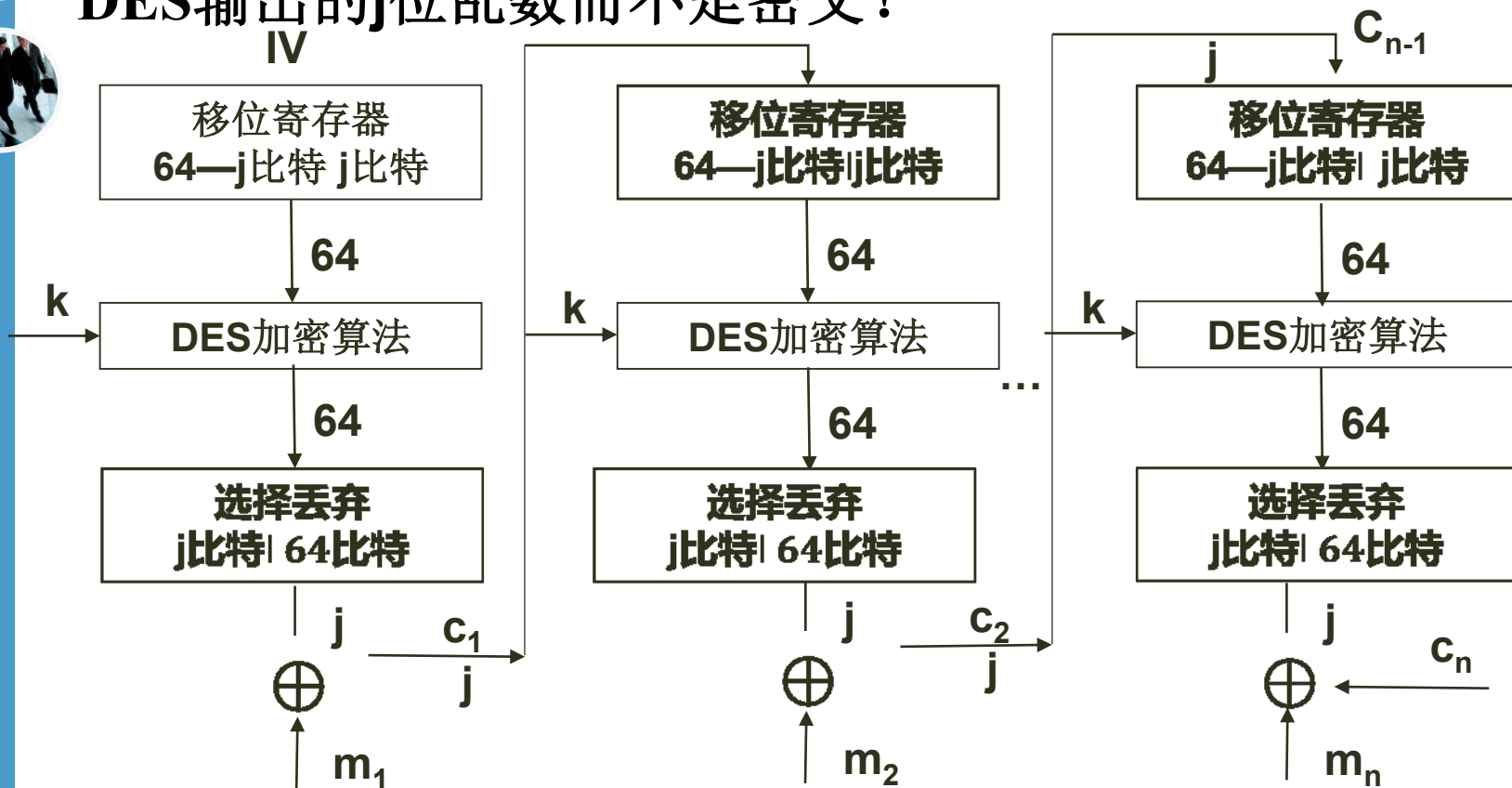




输出反馈 (OFB-Output Feedback) 模式

OFB模式在结构上类似于CFB模式，但反馈的内容是DES输出的j位乱数而不是密文！

IV



64位分组算法下的j位OFB加密框图



优点:

(1) 这是将分组密码当作序列密码使用的一种方式, 序列密码与明文和密文无关!

(2) 不具有错误传播特性!

适用范围:

(1) 明文的冗余度特别大, 信道不好但不易丢信号, 明文有误码也不影响效果的情形。如图象加密, 语音加密等。

(2) **OFB**安全性分析表明, **j**应与分组大小相同。

缺点:

(1) 不能实现报文的完整性认证。

(2) 乱数序列的周期可能有短周期现象。





总 评：

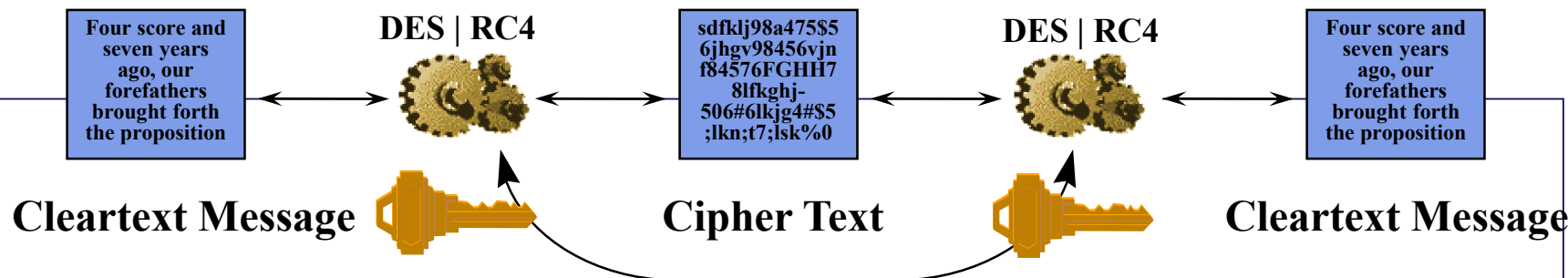
- **ECB**模式简单、高速，但最弱，易受重发和替换攻击。商业软件中仍应用，可用于无结构小数据。
- 低丢包率，低误码率，对明文的格式没有特殊要求的环境可选用**CBC**模式。需要完整性认证功能时也可选用该模式。
- 高丢包率，或低误码率，对明文格式有特殊要求的环境（如字符处理），可选用**CFB**模式。
- 低丢包率，但高误码率，或明文冗余多，可选用**OFB**模式。（但加密前先将明文压缩是一种安全的方法）

p146



对称加密

- 对称加密算法中加密和解密使用相同的密钥。
- 对称加密算法工作原理可以用下列公式表示：
 - 加密（明文，密钥）= 密文
 - 解密（密文，密钥）= 明文





对称密钥算法的优缺点



优点：

- 加解密速度快。



缺点：

- 网络规模扩大后，**密钥管理**很困难；
- 无法解决消息确认问题；
- 缺乏自动检测密钥泄露的能力。



典型算法：DES (Data Encryption Standard, 数据加密标准)



- DES是第一个得到广泛应用的密码算法；



源自IBM 1970年开发的Lucifer算法， 1977年DES被采纳为美国联邦信息处理标准FIPS-46 ；

- 高质量的数据保护，防止数据未经授权的泄露和未被察觉的修改；
- 高复杂性，破译开销超过可能获得的利益，同时便于理解和掌握；
- 安全性不依赖于算法的保密，仅以加密密钥的保密为基础；
- 实现经济，运行有效，并且适用于多种不同的应用

- 1980年美国国家标准协会ANSI正式采用这个算法作为美国商用加密算法DEA。



- 每隔五年，NSA都要对DES进行评估，并重新审议它是否适合继续作为联邦加密标准。
- 1988，1993年分别以FIPS46-1 FIPS46-2标准发布。
- DES(56)在1997年被攻破；1998年美国政府终于决定不再延用DES作为联邦加密标准，DES退出加密标准的舞台。
- 在最后的FIPS46-3标准中，EDE模式(加密-解密-加密)的三重DES成为提供更高安全级别的替代算法。
- DES是一个**分组加密算法**，它以64位为分组对数据加密。同时DES也是一个对称算法，即加密和解密用的是同一个算法。它的密钥长度是**56位**



分组密码设计准则

混淆（confusion）：用于掩盖明文和密文间的关系。在加密变换过程中使明文、密钥以及密文之间的关系尽可能地复杂化，以防密码破译者采用统计分析法，通过研究密文以获取冗余度和统计模式。

扩散（diffusion）：通过将明文冗余度分散到密文中使之分散开来。密码分析者寻求这些冗余度将会更难。（**扩散函数**，通过换位，亦称置换）

迭代结构：选择某个较为简单的密码变换，在密钥控制下以迭代方式多次利用它进行加密变换，就可以实现预期的扩散和混乱效果。（**轮函数**）

DES算法

- 明文分组M(64bits)

- 初始转置 (IP) 后

$M = (L_0, R_0)$

- 对于第*i*次迭代, 用L和R表示第 (*i*-1) 次迭代后的左右两部分(各32bit), 则

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

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

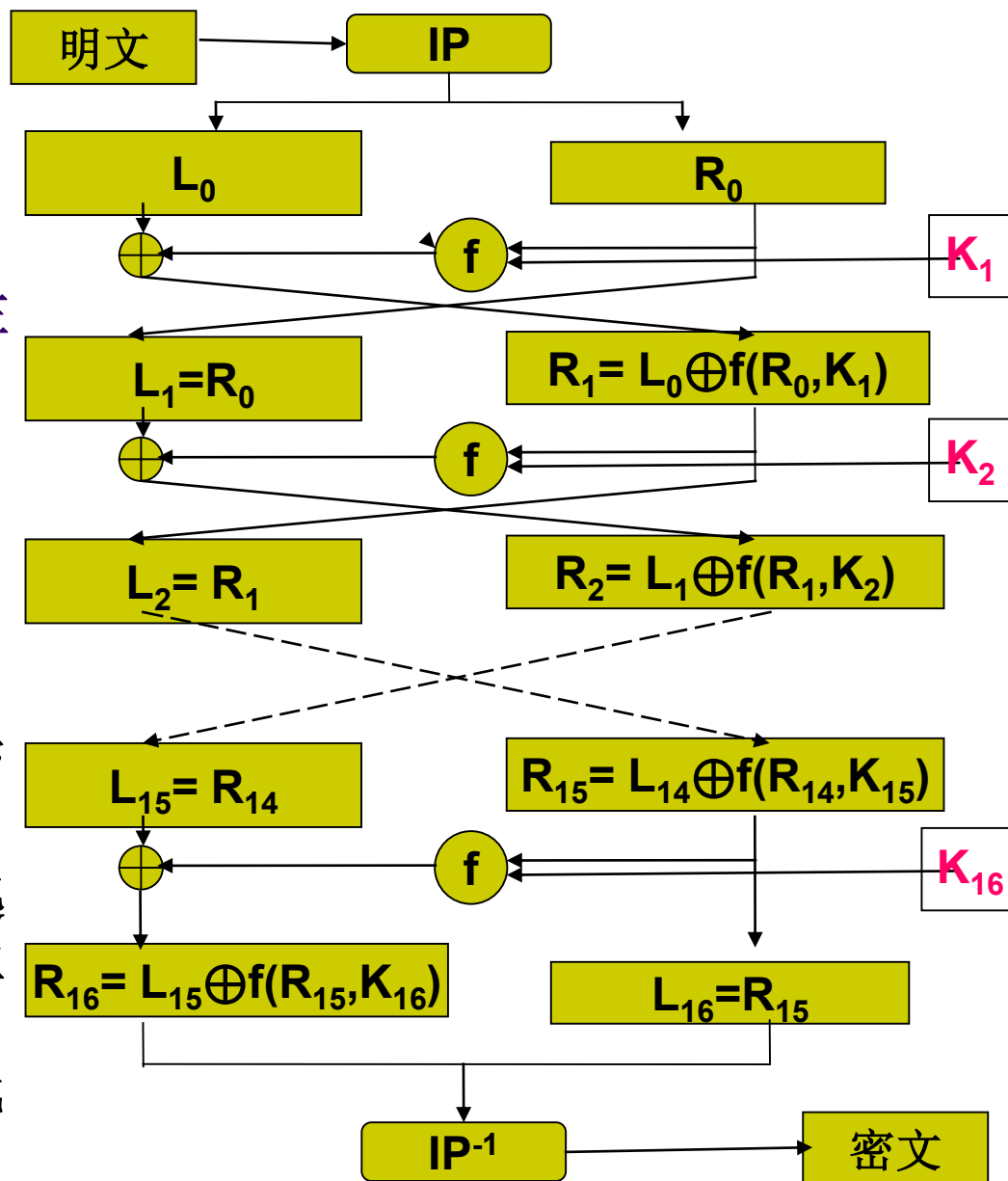
($i=2, 3, \dots, 16$)

这里 \oplus 是对应比特的模2加

- 轮函数f对两部分进行16轮 (混淆和扩散组合) 密钥控制的迭代运算。

- 将左、右两部分合在一起经过一个末置换, 就产生了密文

- 算法: 参考书1P189, 代码P441



DES加密过程

设 $m = m_1 m_2 m_3 \dots m_{64}$ $k = k_1 k_2 k_3 \dots k_{64}$ $m_i, k_i = 0, 1 \ i = 1, 2, \dots, 64$

加密过程可表达如下：

$$\text{DES}(m) = IP^{-1} \cdot T_{16} \cdot T_{15} \dots T_2 \cdot T_1 \cdot IP(m)$$

T_i 为使用 K_i 的一轮运算。

IP 置换：

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

IP 和 IP^{-1} 作用在于打乱原来输入 x 的 ASCII 码字划分的关系，

并将原来明文的校验位 $x_8, x_{16}, \dots, x_{64}$ 成为 IP 输出的一个字节。

在密码意义上作用不大，

因为输入组 x 与其输出组 $y = IP(x)$ (或 $IP^{-1}(x)$) 是已知的一一对应关系。



DES加密过程

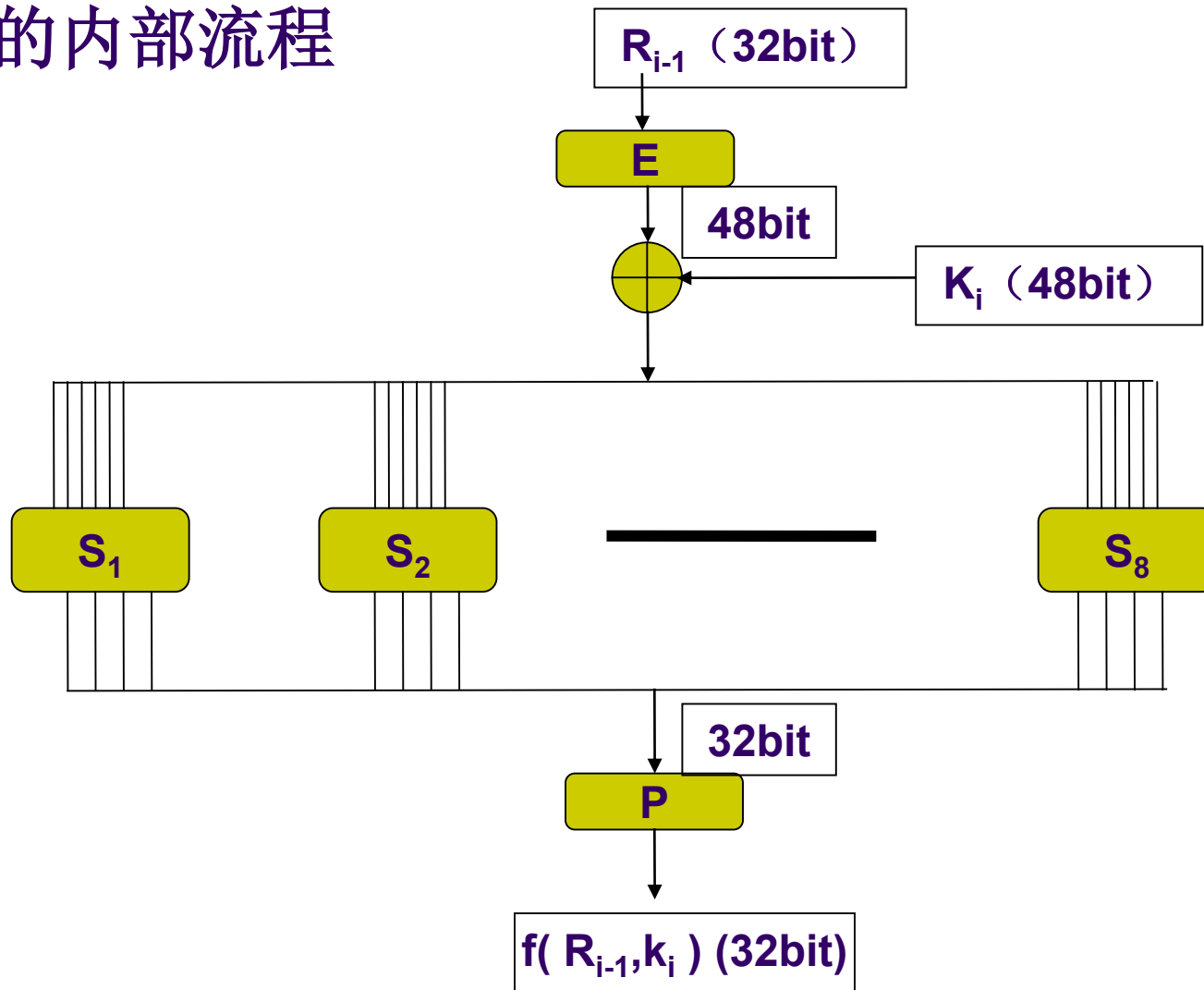
IP^{-1} 是IP的逆置换, 即 $IP^{-1} \cdot IP(m) = IP \cdot IP^{-1}(m) = m$

IP^{-1} 置换:

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



f函数的内部流程





- 扩展置换 (expansion permutation)

E table将32bit输入扩展为48bit。

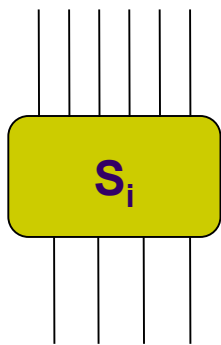
32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

原下标 $s \equiv 0$ 或 $1 \pmod{4}$ 的各比特重复一次得到的，即对原第32, 1, 4, 5, 8, 9, 12, 13, 16, 17, 20, 21, 24, 25, 28, 29各位都重复一次,实现数据扩展。



S盒

E输出的48bit顺序分成8组，每组6bit，分别通过 S_1 , S_2 , ..., S_8 盒又收缩为32bit。每个S盒的输入是6bit，输出是4bit。



S_i 盒输入 $b_1b_2b_3b_4b_5b_6$ ，在 S_i 表中找出 b_1b_6 行， $b_2b_3b_4b_5$ 列的元素 S_i （ b_1b_6 ， $b_2b_3b_4b_5$ ），即是 S_i 盒的输出。



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



置换P是将S盒的32bit输出置换为32bit。

P置换:

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

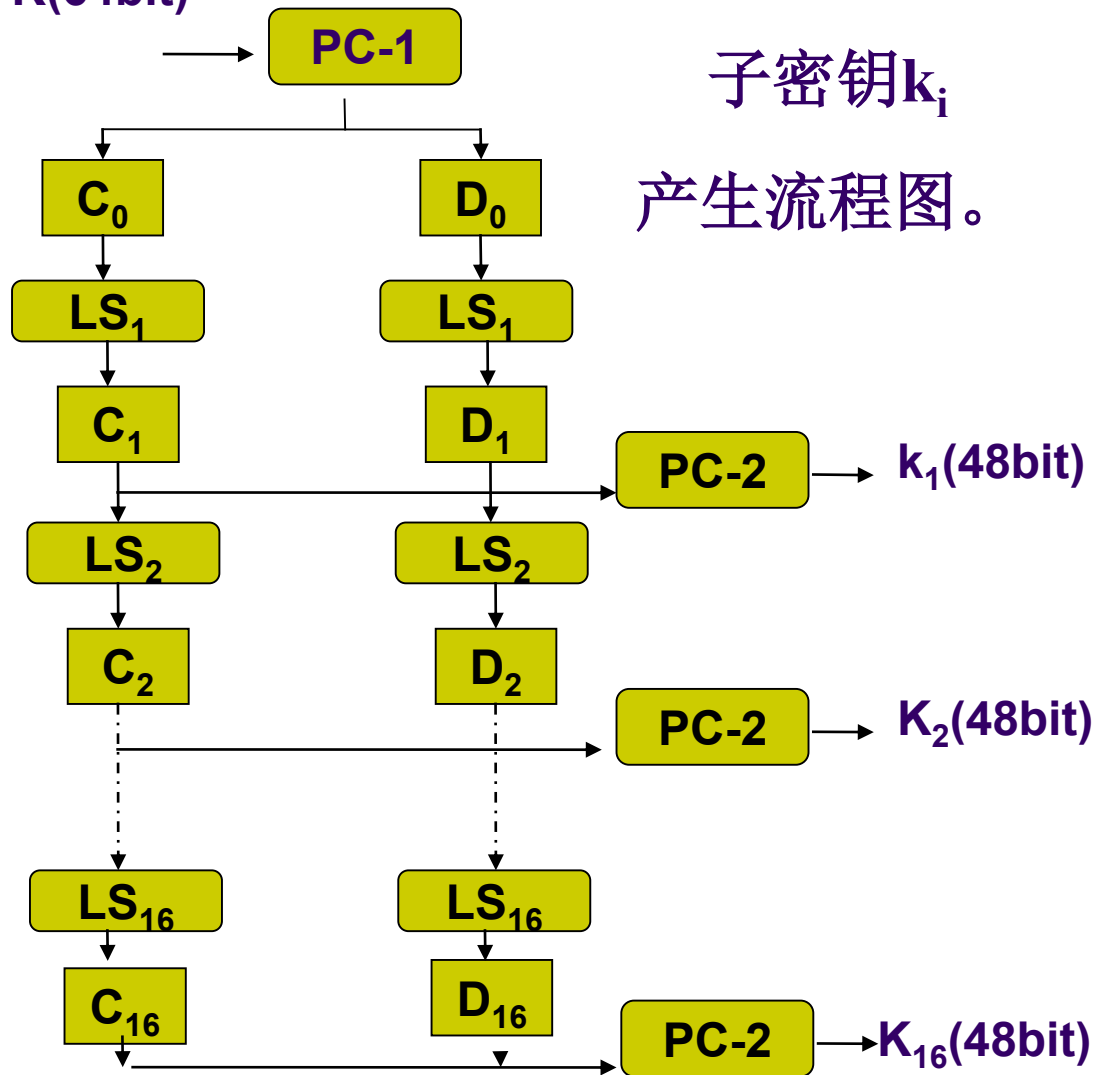


密钥编排算法

- k_i 是 64bit 密钥 k 产生的子密钥， K_i 是 48bit。

- 密钥 k 长度：56 比特，每 7 比特后为一个奇偶校验位（第 8 位），共 64 比特

$K(64\text{bit})$





置换PC-1是将输入的64bit密钥k重新排序，但忽略了k中的第8，16，24，32，40，48，56，64位。因为这8个bit在密钥k中实际作为每个字节的奇偶校验位，故DES中64bit密钥k实际只有56bit有效，即实际密钥k只有56bit。

PC-1置换:

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
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



将置换PC-1的输出56bit分为左右两半各28bit，即得到

C_0 和 D_0 。

LS_i 是将 C_i 和 D_i 循环左移，左移位数如下：

迭代顺序	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
左移位数	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1



置换PC-2是将循环左移后的56bit中选出48bit作为子密钥 k_i 。

PC-2置换：

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

可以看出，输出 k_i 缺 C_iD_i 的9, 18, 22, 25, 35, 38, 43, 54这8bit。



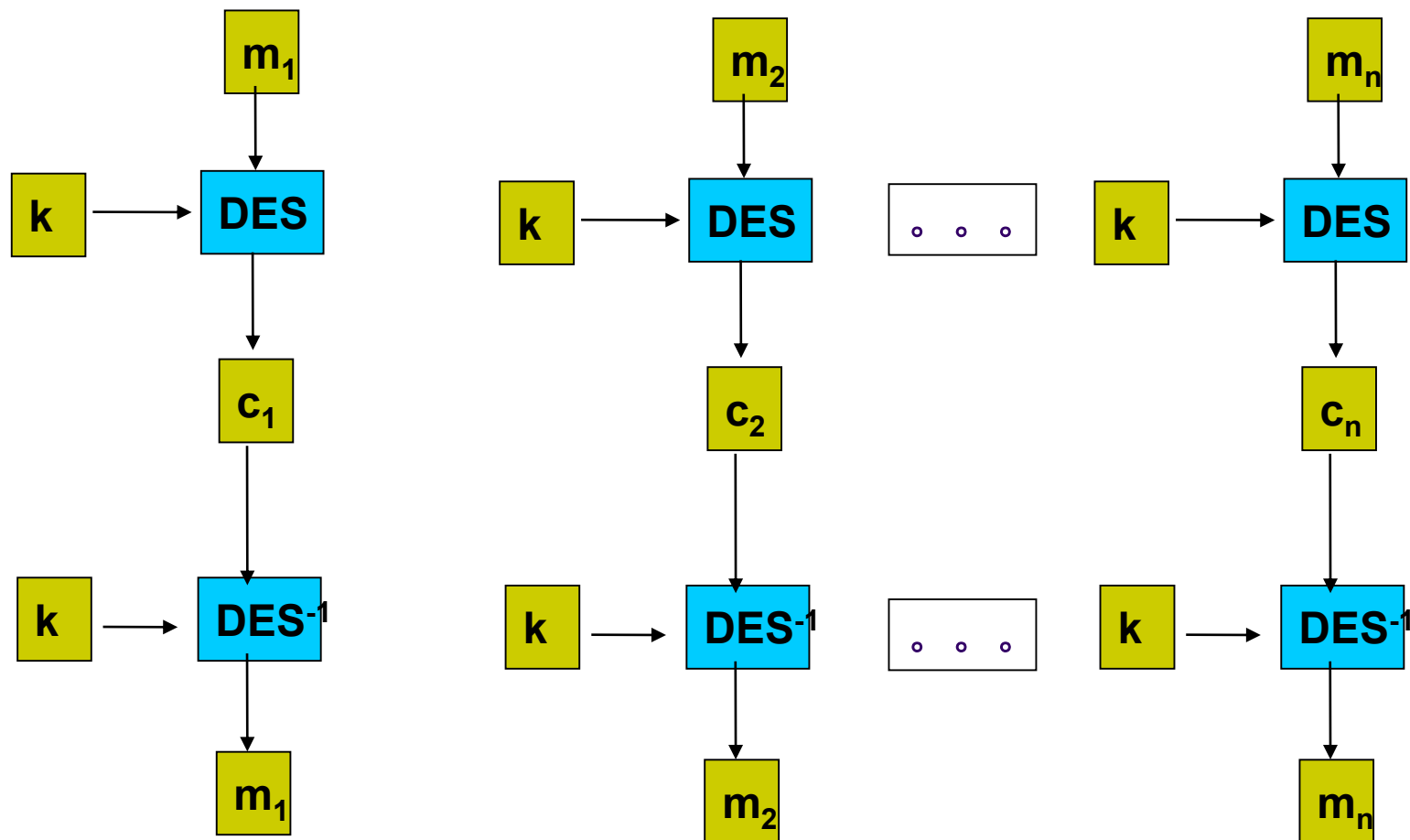
- DES的解密过程与加密过程完全相同,只不过解密中使用子密钥的顺序是 $K_{16}, K_{15}, \dots, K_1$

$$D(C) = T_1^{-1} \cdot \dots \cdot T_{16}^{-1}(C)$$

T_i 为使用 K_i 的一轮运算。

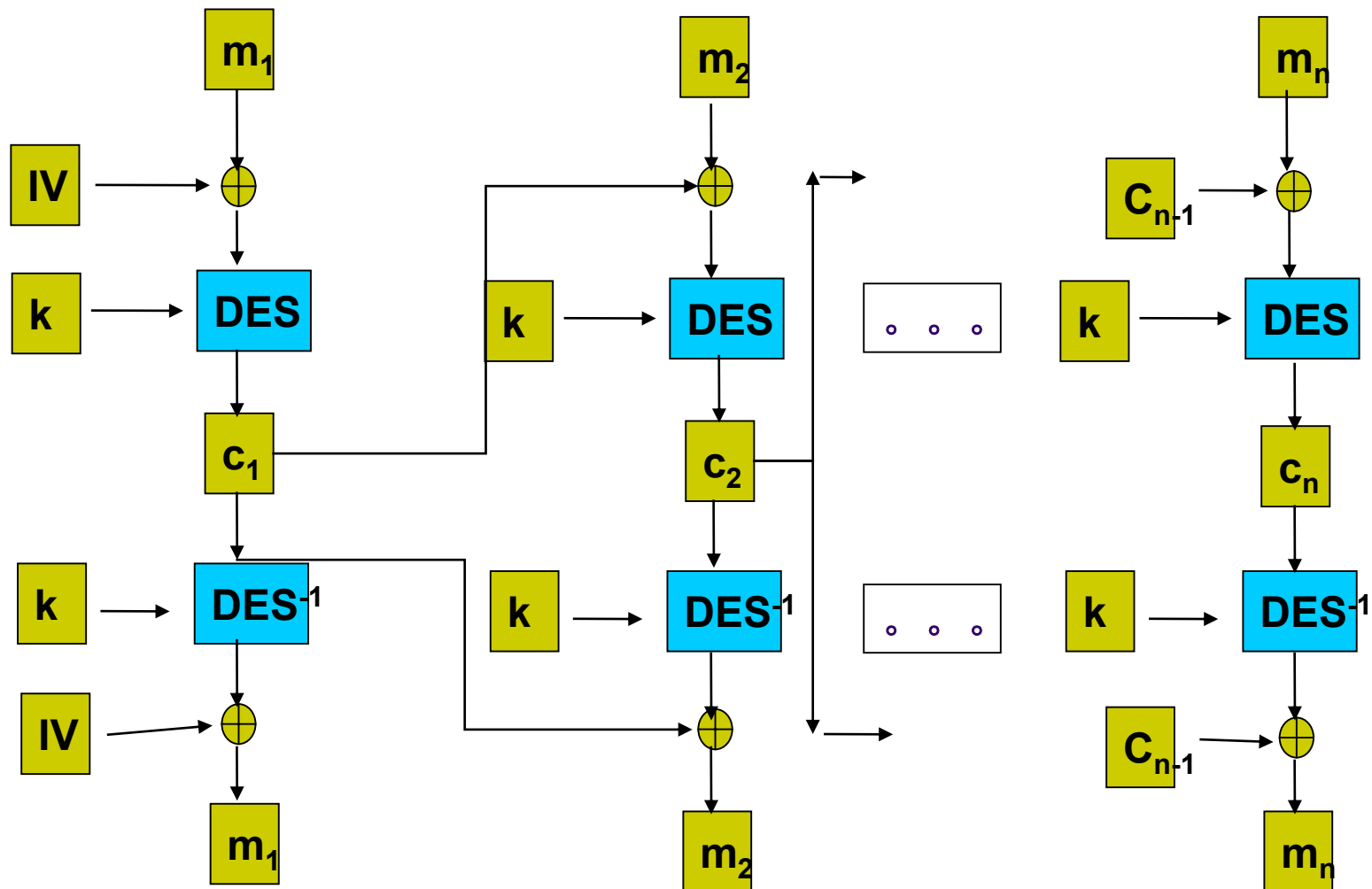


- 定义了ECB,CBC,OFB,CFB模式, 加密:
ECB,CBC,认证: CBC,CFB
- ECB模式





• CBC模式





DES的实现



DES的实现

自正式成为美国标准以来，已有许多公司设计并推出了实现**DES**算法的产品

有的设计专用**LSI**器件或芯片，

有的用现成的微处理器实现。

有的只限于实现**DES**算法，有的则可运行各种工作模式。



DES的变型

为提高安全性和适应不同情况的需求而设计了多种应用**DES**的方式。

提出了多种**DES**的修正形式，如

独立子密钥方式、**DESX**、**CRYPT(3)**、**S**盒可变的**DES**、**RDES**、**sⁿDESⁱ**、**xDESⁱ**、**GDES**等。



DES 安全性



弱密钥和半弱密钥



由于各轮密钥是通过改变初始密钥得到，
因此当初始值分成两部分，每部分都是0或都是1时，
那么算法的每一周期的密钥是相同的。
这样的**弱密钥**共有4种：全1，全0，先全1后全0，先全0后全1。
相当于只有一个子密钥，而不是16个。

还有一些密钥只会产生两个不同的子密钥，而不是16个不同的密钥。
这样就造成两个不同的密钥会有相同的加密结果。

这种情况一共有6对：

01FE 01FE 01FE 01FE 和 FE01 FE01 FE01 FE01;
1FE0 1FE0 0E1F 0E1F 和 E01F E01F F10E F10E;
01E0 01E0 01F1 01F1 和 E001 E001 F101 F101;
1FFE 1FFE 0EFE 0EFE 和 FE1F FE1F FE0E FE0E;
011F 011F 010E 010E 和 1F01 1F01 0E01 0E01;
E0FE E0FE F1FE F1FE 和 FEE0 FEE0 FEF1 FEF1.

上述“密钥对”他们对明文加密的结果是相同的。

同理，只有4个子密钥的有64个



DES的破译分析



56比特密钥太短，已抵挡不住穷尽密钥搜索



攻击

密钥量为 $2^{56} = 7.2 \times 10^{16} = 72,057,594,037,927,936 \approx 10^{17}$ (千万亿)个

若要对DES进行密钥搜索破译，分析者在得到一组明文-密文对条件下，可对明文用不同的密钥加密，直到得到的密文与已知的明文-密文对中的相符。

密钥搜索所需的时间取决于密钥空间的大小 执行一次加密所需的时间。

若假定DES加密操作需时为 $100\mu s$ (一般微处理器能实现)，则搜索整个密钥空间需时为 7.2×10^{15} 秒，近似为 2.28×10^8 年。

若以最快的LSI器件，DES加密操作时间可降到 $5\mu s$ ，也要 1.1×10^4 年才能穷尽密钥。



DES的破译分析



借助差分密码分析的思想



1993年Wiener给出一个**密钥搜索机**的详细设计方案，估计**100万美元**制造一台机器，搜索一个密钥平均大约花**3.5小时**

1997年1月28日，**RSA**数据安全公司在**RSA**安全年会上悬赏**一万美金** 破译密钥长度为**56比特**的**DES**算法。

美国克罗拉多州的程序员**Verser**从**1997年3月13日**起，用了**96天**的时间，在**Internet**上数万名志愿者的协同工作下，于**1997年6月17日**用**穷尽密钥搜索方法**成功地找到了**DES**的密钥。

1998年7月17日，电子边境基金会**EFF**使用一台**25万美元**的电脑通过穷尽密钥搜索方法在**56小时内**破解了**56比特DES**。

1999年的**RSA**会议期间，穷尽密钥搜索时间减少到不足**24小时**。



3DES简介



- DES的分组长度太短（仅64位）、密钥长度更短（仅56位），可以通过穷举（也称野蛮攻击）的方法在较短时间内破解。1978年初，IBM意识到DES的密钥太短，于是设计了一种方法，利用三重加密来有效增加密钥长度，加大解密代价，称为3DES。
- 3DES 是DES算法扩展其密钥长度的一种方法，可使加密密钥长度扩展到128位（112位有效）或192位（168位有效）。其基本原理是将128位的密钥分为64位的两组，对明文多次进行普通的DES加解密操作，从而增强加密强度。
- 许多基于Internet的应用里用到：PGP和S/MIME

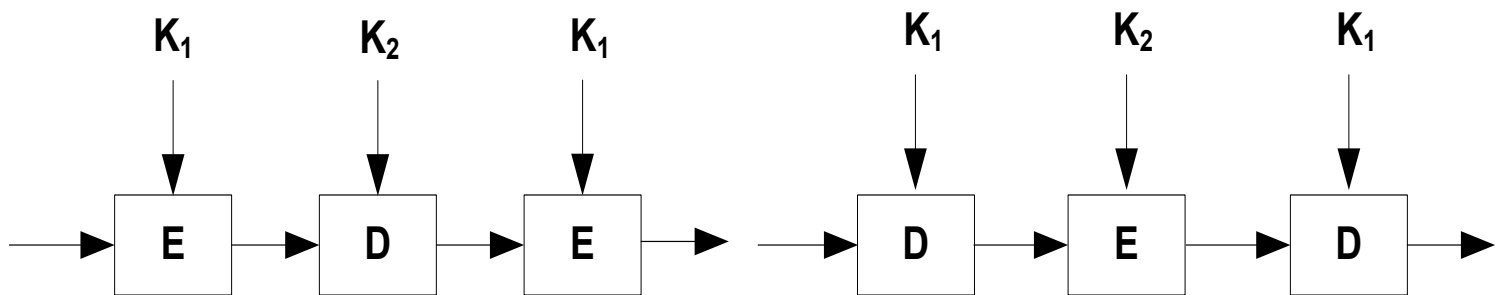


3DES的模式

- 3DES有三种不同的模式 (p253)

- DES-EEE3, 使用3个不同的密钥进行加密;
- DES-EDE3, 使用3个不同的密钥, 对数据进行加密、解密、再加密 (P207)。
- DES-EEE2和DES-EDE2, 与前面模式相同, 只是第1次和第3次使用同一密钥。

最常用的3DES是DES-EDE2。



(a) DES-EDE2的加密

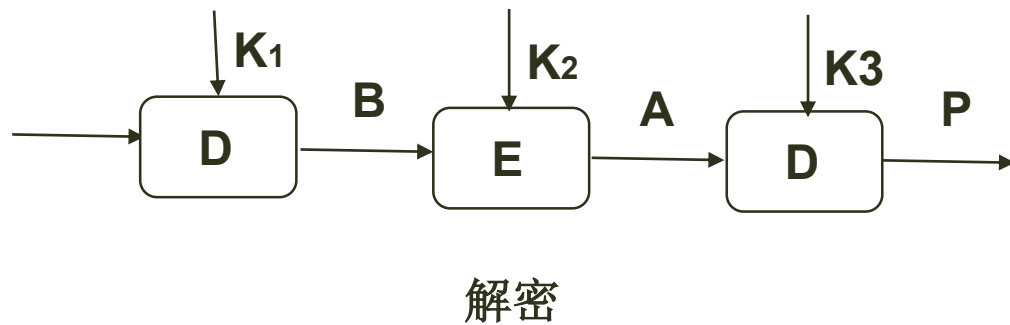
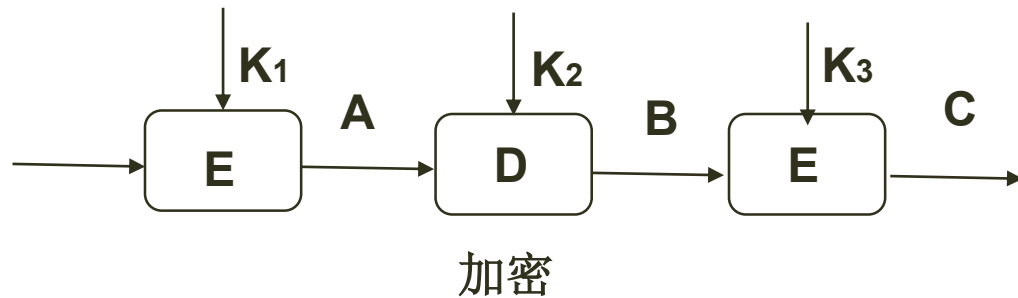
(b) DES-EDE2的解密

$$C = E_{K1}(D_{K2}(E_{K1}(P))) \Leftrightarrow P = D_{K1}(E_{K2}(D_{K1}(C)))$$

三密钥的三重DES

(Triple DES with Three Keys)

- $C = E_{K_3}(D_{K_2}(E_{K_1}(P))) \Leftrightarrow P = D_{K_3}(E_{K_2}(D_{K_1}(C)))$

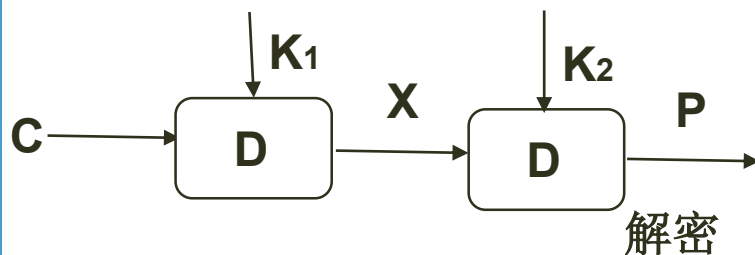
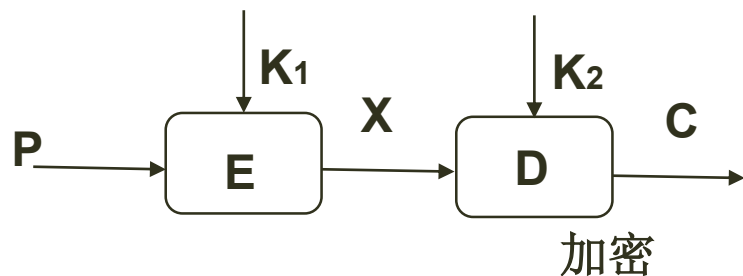




● 2DES 可以吗? Meet-In-The-Middle attack

由 $C = E_{K_2}(E_{K_1}(P))$
从图中可见

$$X = E_{K_1}(P) = D_{K_2}(C)$$



若给出一个已知的明密文对 (P, C) 做：对 2^{56} 个所有密钥 K_1 做对明文 P 的加密，得到一张密钥对应于密文 X 的一张表；类似地对 2^{56} 个所有可能的密钥 K_2 做对密文 C 的解密得到相应的明文 X 。做成一张 X 与 K_2 的对应表。比较两个表就会得到真正使用的密钥对 K_1, K_2

对二重DES的中间相遇攻击的分析

- 已知，给定一个明文 P ，经二重DES加密有 2^{64} 个可能的密文。而二重DES所用密钥的长度应是 112bits，所以原则密钥有 2^{112} 个可能。于是对给定明文 P 加密成密文有 $2^{112}/2^{64} = 2^{48}$ 种可能。于是，密文是假的比例约为 $2^{48-64} = 2^{-16}$ 。这样，对已知明文-密文对的中间相遇攻击成功的概率为 $1 - 2^{-16}$ 。
- 攻击用的代价 {加密或解密所用运算次数} $\leq 2^{56} + 2^{56} = 2^{112}$



RC系列



- RC (“Rivest Cipher”) (“Ron’s Code”) 系列
- Ron Rivest为**RSA**数据安全公司设计的一系列密码：
 - RC1从未被公开，以致于许多人们称其只出现在Rivest的记事本上；
 - RC2 没有专利,是变长密钥加密密法；(RC3在设计过程中在**RSADSI**内被攻破)；
 - RC4是Rivest在1987年设计的变长密钥的**序列**密码（P282），OFB方式工作，用于Lotus Notes, Oracle database, WTLS, skype signalling；

美国政府已允许**RC2**、**RC4**产品出口，但密钥限制为 $\leq 40\text{bit}$ 。
（假设芯片每秒能测试一百万个密钥，则仅需**12.7**天即可破译）



- **RC5**是**Rivest**在**1994**年设计的分组长、密钥长、迭代轮数都可变的**分组**迭代密码算法；
 - 一个特定的**RC5**表示为**RC5-w/r/b**，其中**w**是以比特表示的字(word)的尺寸($w=16, 32, 64$)，分组大小为 $2w$ ，**r**是轮数($0 \leq r \leq 255$)，**b**是密钥的字节(byte)长度($0 \leq b \leq 255$)
 - **RC5-32/12/5**, **RC5-32/12/6**, **RC-32/12/7**已分别在**1997**年被破译
 - **Rivest**建议**RC-32/12/16**
- **RC6**
 - **AES**的竞争算法，**RC6-w/r/b** block 128bit，key: 128,192,256bits
 - U.S. Patent 5,724,428，Patent 5,835,600



IDEA (International Data Encryption Algorithm)

- IDEA (International Data Encryption Algorithm, 国际数据解密算法) 是瑞士联邦理工学院的中国学者赖学嘉与著名的密码学专家James Massey等人1990年提出的算法。
- IDEA是对称、分组密码算法，输入的明文为64位，密钥为128位，生成的密文为64位；在DES算法的基础上发展出来的，类似于三重DES。
- IDEA是一种相对较新的算法，有坚强的理论基础，但仍应谨慎使用(尽管该算法已被证明可对抗差分分析和线性分析)；
- IDEA也存在一些弱密钥
- IDEA是一种专利算法(在欧洲和美国)，专利由Ascom-Tech AG拥有；
- PGP中已实现了IDEA；
- 2IDEA、3IDEA $C = E_{k3}(D_{k2}(E_{k1}(P)))$



AES (Advanced Encryption Standard)



高级加密标准



- **AES评选过程:** 1997年1月2日, NIST (National Institute of Standards and Technology 美国技术标准局) 发起**AES**开发活动, 目的是为了确定一非保密的、公开披露的、全球免费使用的分组密码算法, 用于保护下一代政府的敏感信息, 也希望能够成为秘密的和公开部门的数据加密标准(**DES**)
- **基本要求:**比**3DES**快而且至少和**3DES**一样安全, 分组长度为**128 bit**, 密钥长度为**128/192/256 bit**,
- **最后的5个候选算法:** Mars, RC6, Rijndael, Serpent, and Twofish
- **2000年10月2日**, NIST宣布选定Rijndael做为**AES**的建议标准。Rijndael算法的原型是Square算法, 其设计策略是宽轨迹策略(**Wide Trail Strategy**), 以针对差分分析和线性分析;
- Rijndael是迭代分组密码, 其分组长度和密钥长度都是可变的; 为了满足**AES**的要求, 分组长度为**128bit**, 密码长度为**128/192/256bit**, 相应的轮数r为10/12/14。

<http://csrc.nist.gov/encryption/aes/>



思考

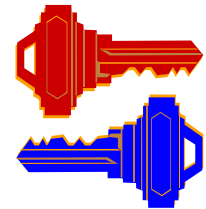


- 互联网典型应用场景：
 - C/S, B/S,
 - 通信对端未知
 - 大量的用户
 - 不安全的公共网络



公钥密码学

- W.Diffie and M.E.Hellman, New Directions in Cryptography, IEEE Transaction on Information Theory, V.IT-22.No.6, Nov 1976, PP.644-654
- 公钥密码学的出现使大规模的安全通信得以实现 – 解决了密钥分发问题;
- 非对称密码技术又称公钥密码技术, 或双钥密码技术, 即加密和解密数据使用不同的密钥。





公钥密码算法重要特性

- 加密与解密由不同的密钥完成

加密X得到Y: $Y = E_{KU}(X)$

解密Y得到X: $X = D_{KR}(Y) = D_{KR}(E_{KU}(X))$

- 知道加密算法, 从加密密钥得到解密密钥在计算上是不可行的。

$$X = D_{KR}(E_{KU}(X)) = E_{KU}(D_{KR}(X))$$

- 应用中一对密钥为:

- 秘密密钥（私钥），由使用者自己掌握使用
- 公共密钥（公钥），可以公开发布；

- 由公共密钥加密的信息必须用秘密密钥解密（必须是同一对密钥），由秘密密钥加密的信息必须用公共密钥解密。



非对称密钥算法应用原理



Alice

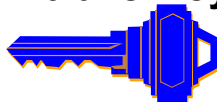


Plaintext



+

Bob's
Public Key



=



Ciphertext

+

Bob's
Private Key



=

Plaintext



Bob



分别完成什么功能？

Alice



Plaintext

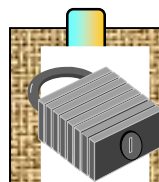


+

Bob's
Private Key



=



Ciphertext

+

Bob's
Public Key



=

Plaintext



Bob



哪里错了？

公钥算法
工作前提



公钥算法功能总结



由于公开密钥加密在计算上的巨大开销，当前主要用于：密钥交换，数字签名；



	加密	密钥交换	数字签名
RSA（Rivest, Shamir和	Y	Y	Y
ElGamal	Y	Y	Y
ECC（Elliptic Curve Curve Discrete）	Y	Y	Y
DH(Diffie-Hellman)	N	Y	N
DSA	N	N	Y



公钥密码体制的基本原理

- 单向陷门函数(trapdoor one-way function) 满足下列条件的函数 f :

(1) 给定 x , 计算 $y=f(x)$ 是容易的;

(2) 给定 y , 计算 x 使 $y=f(x)$ 是困难的。

(所谓计算 $x=f^{-1}(Y)$ 困难是指计算上相当复杂, 已无实际意义。)

(3) 存在 δ , 已知 δ 时, 对给定的任何 y , 若相应的 x 存在, 则计算 x 使 $y=f(x)$ 是容易的。

注: 1*. 仅满足(1)、(2)两条的称为单向函数; 第(3)条称为陷门性, δ 称为陷门信息。

2*. 当用陷门函数 f 作为加密函数时, 可将 f 公开, 这相当于公开加密密钥。此时加密密钥便称为公钥, 记为 Pk 。 f 函数的设计者将 δ 保密, 用作解密密钥, 此时 δ 称为秘密钥匙, 记为 Sk 。

3*. 单向陷门函数的第(2)条性质表明窃听者由截获的密文 $y=f(x)$ 推测 x 是不可行的。



Whitfield Diffie提出绝大多数公开密钥算法都基于以下三种难题之一：

1. 背包问题：给定一个互不相同数组成的集合，找出一个子集其和为N
2. 离散对数：如果 p 是素数， g 和 M 是整数，找出 x 满足 $g^x \equiv M \pmod{p}$
3. 因子分解：设 N 是两个素数的积，则
 - (a) 分解 N
 - (b) 给定整数 M 和 C ，寻找 d 满足 $M^d \equiv C \pmod{N}$
 - (c) 给定整数 e 和 C ，寻找 M 满足 $M^e \equiv C \pmod{N}$
 - (d) 给定整数 x ，判定是否存在整数满足 $x \equiv y^2 \pmod{N}$



RSA

- Ron Rivest, Adi Shamir和Leonard Adleman 1977年研制并且1978年首次发表。
- **RSA**是一种分组密码，其理论基础是一种特殊的可逆模指数运算，其安全性**基于分解大整数**的困难性。
- 已被许多标准化组织(如**ISO**、**ITU**、**IETF**和**SWIFT**等)接纳，目前多数公司使用的是**RSA**公司的**PKCS**系列；
- **RSA-155(512 bit)**, **RSA-140**于1999年分别被分解；

大素数相乘和因子分解可以被
看成一个单向函数





RSA (cont.) (p334)



设 n 是两个不同大素数之积，即 $n = pq$ ，计算其欧拉函数值 $\Phi(n)=(p-1)(q-1)$.



随机选一整数 e , $1 \leq e < \Phi(n)$, $(\Phi(n), e) = 1$. 即 e 和 $(p-1)(q-1)$ 互素

因而在模 $\Phi(n)$ 下, e 有逆元 d

$$de \equiv 1 \pmod{\Phi(n)}$$

即 $ed \equiv 1 \pmod{(p-1)(q-1)}$

取公钥为 n, e , 私钥为 d, n , (p, q 不再需要, 应该被舍弃, 但绝不可泄露)

$$E_k(x) = x^e \pmod{n}$$

定义加密变换为

解密变换为

$$D_k(y) = y^d \pmod{n}$$

e 值对RSA的加密速度影响很大, 最常用的 e 值为3, 17, 65537 ($2^{16}+1$), X.509中建议使用65537, PEM中建议使用3, PKCS#1中建议使用3或65537,

要求:

位数 (指 n 的位数): 要1024以上, 加密仍然依赖于密钥长度
 $p-1, q-1$ 有大的素因子, $p+1, q+1$ 也要有大的素因子



RSA基本结构



```
struct {  
    int pad;  
    long version;  
    const RSA_METHOD *meth;  
    ENGINE *engine;  
    BIGNUM *n;      n=p*q  
    BIGNUM *e;      公开的加密指数, 经常为65537 (0x10001)  
    BIGNUM *d;      私钥  
    BIGNUM *p;      大素数p  
    BIGNUM *q;      大素数q  
    BIGNUM *dmp1;   d mod (p-1)  
    BIGNUM *dmq1;   d mod (q-1)  
    BIGNUM *iqmp;   (inverse of q) mod p  
    int references;  
    int flags;  
    // ...  
}RSA;
```




例：如果 $p=47$ $q=71$,那么 $n=pq=3337$

加密密钥 e 与 $(p-1)(q-1)=46 \times 70=3220$ 没有公因子

随机选取 e ，如79，那么： $d=79^{-1} \bmod 3220=1019$

公开 e 和 n ，将 d 保密

加密消息 $m=6882326879666683$

首先将其分成小的组，在此例中，按三位数字一分组就可进行加密，这个消息将分成六个分组 m_i 进行加密：

$m_1=688$ $m_2=232$ $m_3=687$ $m_4=966$ $m_5=668$ $m_6=003$

第一组分组加密为： $688^{79} \bmod 3337=1570=c_1$

对随后的分组进行加密得密文：

$c=1570 \ 2756 \ 2091 \ 2276 \ 2423 \ 158$

解密消息时用解密密钥 $d=1019$ 进行相同的指数运算。因而：

$1570^{1019} \bmod 3337=688=m_1$

消息的其余部分可用同样的方法恢复出来。



RSA算法的使用

加密，签名，密钥交互

- 1.加解密
 - A的公开密钥为 (e,n) ,B对消息 m 加密
 - $c=m^e \mod n$ 给A, 只有A能解密
 - $m=c^d \mod n$
 - 特点:
 - 和A从来不认识,都可进行保密通讯,只要知道A的公钥.
 - 速度慢
 - 要求对公开密钥进行保护，防止修改和替换。



RSA算法的使用



2. 数字签名与身份认证

- A的公开密钥为 (e, n) , 私钥为 (d, n) , A对消息 m 的数字签名为: $s = H(m)^d \bmod n$,
 - $H(x)$ 为公开的散列(hash)函数. $H(m)$: 摘要

什么是数字签名?

- 任何人都可验证A对 m 的签名的有效性 $H(m) = s^e \bmod n$
- 功能: 防止非法篡改、伪造, A的抵赖与否认, 对A的假冒等。
- 要求对公开密钥进行保护, 防止修改。



RSA算法的使用

- 3.加密和数字签名同时使用
 - A的公钥为 (e_1, n_1) ,私钥为 (d_1, n_1)
 - B的公钥为 (e_2, n_2) ,私钥为 (d_2, n_2) , $n_1 > n_2$
 - A对B发消息 m ,并签名:
 - A计算 $c = ((m^{e_2}) \bmod n_2)^{d_1} \bmod n_1$
 - A将 c 给B,并告诉是A发的。
 - B验证 $m = ((c^{e_1}) \bmod n_1)^{d_2} \bmod n_2$



RSA算法的使用

● 4. 密钥交换

- A,B商量使用对称密码(**IDEA**)加密消息m,
- A,B可用**RSA**协商对称密码的密钥
- B随机产生工作密钥k,找到A的公开密钥(e,n),B对工作密钥k加密
- $c = k^e \mod n$ 给A, 只有A能解密:
 $k = c^d \mod n$
- A,B共同用k在对称密码系统(**IDEA**)下进行保密通讯



RSA安全性讨论

若 $n=pq$ 被因子分解，则RSA 便被击破。

因为若 p, q 已知，则 $\varphi(n)=(p-1)(q-1)$ 便可算出。解密密钥 d 关于 e 满足下式：

$$de \equiv 1 \pmod{\varphi(n)}$$

为了安全起见，对 p 和 q 还要求如下：

- (1) p 和 q 的长度相差不大；
- (2) $p-1$ 和 $q-1$ 有大素数因子；
- (3) $(p-1, q-1)$ 很小。

满足这些条件的素数称作安全素数。



实现中的问题

- ① 如何计算 $a^b \bmod n$
- ② 密钥产生

如何计算 $a^b \bmod n$

中间结果非常大
幂运算的效率

要点1: $(a \times b) \bmod n = [(a \bmod n) \times (b \bmod n)] \bmod n$

要点2: $a^{16} = \underbrace{a \times a \times \dots \times a}_{16 \text{ times}}$
 $= a^2, a^2, a^2, a^2$



更一般性的问题： a^m

M的二进制表示为 $b_k b_{k-1} \dots b_0$, 则 $m = \sum_{b_i \neq 0} 2^i$

$c \leftarrow 0; d \leftarrow 1$

for $i \leftarrow k$ downto 0

do $c \leftarrow 2 \times c$

$d \leftarrow (d \times d) \bmod n$

if $b_i = 1$

then $c \leftarrow c + 1$

$d \leftarrow (d \times a) \bmod n$

return d

计算 $a^m \bmod n$

$$a^m \bmod n = [\prod_{b_i \neq 0} a^{(2^i)}] \bmod n$$

$$= \prod_{b_i \neq 0} [a^{(2^i)} \bmod n]$$



密钥产生

如何找到足够大的素数 p 和 q ?

选择 e 或 d 计算另外一个

素数选取

没有产生任意的大素数的有用技术，通常的做法是随机选取一个需要的数量级的技术并检验这个数是否是素数。

传统使用试除法

- 依次用比该数平方根小的素数进行除法运算
- 只对小数有操作性

根据素数的特性使用统计素性检测

- 所有的素数满足的特性
- 但是一些伪素数也满足此特性



素数：只能被1和它本身整除的自然数；否则为合数。



每一个合数都可以唯一地分解出素数因子

$$6=2 \cdot 3$$

$$999999=3 \cdot 3 \cdot 3 \cdot 7 \cdot 11 \cdot 13 \cdot 37$$

$$27641=131 \cdot 121$$



从2开始试验每一个小于等于 $\sqrt{27641}$ 的素数。

素数检测

直接判断一个整数是否为素数是困难的

命题：如果p是奇素数，则方程

$$X^2 \equiv 1 \pmod{p}$$

只有两个解 $x \equiv \pm 1 \pmod{p}$ 。

证明： $X^2 \equiv 1 \pmod{p}$

$$\Rightarrow p \mid (x^2 - 1)$$

$$\Rightarrow p \mid (x+1)(x-1)$$

$$\Rightarrow p \mid (x+1), \text{或者 } p \mid (x-1)$$

$$\Rightarrow x+1=kp, \text{或者 } x-1=jp+1$$

$$\Rightarrow x \equiv 1 \pmod{p}, \text{或者 } x \equiv -1 \pmod{p}$$

若方程 $x^2 \equiv 1 \pmod{p}$ 存在的解不是 $x \equiv \pm 1$,则P不是素数

素数检测-- WITNESS算法

- 目前还没有一个高效的办法，实际中应用最多Miller and Rabin, WITNESS算法

WITNESS (a,n) 判断n是否为素数，a是某些小于n的整数

1. 令 $b^k b^{k-1} \dots b^0$ 为 $(n-1)$ 的二进制表示

2. $d \leftarrow 1$

3. For $i \leftarrow k$ downto 0

4. Do $x \leftarrow d$

5. $d \leftarrow (d \times d) \bmod n$

6. If $d=1$ and $x \neq n-1$ $x^2 \equiv 1 \bmod n$

7. Then return TRUE

8. if $b_i=1$

9. then $d \leftarrow (d \times a) \bmod n$ $a^{n-1} \bmod n$

10. if $d \neq 1$

11. then return TRUE

12. return FALSE

返回值：

TRUE：n一定不是素数

FALSE：n可能是素数

应用：

随机选择 $a < n$, 计算s次，
如果每次都返回FALSE
，则这是n是素数的概率
为

$(1 - 1/2^s)$

素数选取

- 选取素数的过程如下：

- ①随机选取一个奇素数 n

- ②随机选取一个整数 $a < n$

- ③执行概率素数判定测试（如：用WINESS（ a, n ））

。如果 n 没有通过检验，舍弃 n 并转到步骤1

- ④如果 n 通过了足够多次的检验，接受 n ，转到步骤2



盖莫尔(*ElGamal*)算法

- [Taher Elgamal](#)于1985年提出，很大程度上为**Diffe-Hellman**密钥交换算法的推广和变形，基于离散对数问题，
- **Elgamal**,**Schnorr**和**DSA**签名算法都非常类似。事实上，它们仅仅是基于离散对数问题的一般数字签名的三个例子。
- **Elgamal**方案未申请专利。但受到**DH**专利的制约（**DH**专利已经在1997年4月29日到期）。
- [GNU Privacy Guard \(PGP\)](#)中实现
- [Taher Elgamal](#)于19854年提出 **Elgamal**数字签名方案，其一个修改被**NIST**采纳为数字签名标准**DSS**



Elgamal (cont.)

- 选择一个素数 p ，两个随机数 g 和 x ， g 和 x 都小于 p ，计算 $y=g^x \bmod p$ ，
 - 公钥为 y, g, p
 - 私钥为 x
 - g, p 可由一组用户共享。
- Elgema加密：
 - M -待加密消息， $0 < k$ -(随机数,秘密) $< p-1$ ，计算密文 (a,b) :
 - $a = g^k \bmod p$
 - $b = y^k M \bmod p = (g^x)^k \bmod p$
 - 解密时，计算 $M = b/a^x \bmod p = M(g^x)^k g^{-xk} \bmod p = M$
- Elgema签名：
 - 计算 $a = g^k \bmod p$
 - b 满足 $M = (xa + kb) \bmod (p-1)$ ， a, b 为签名值， $b = (M - xa)k^{-1} \bmod (p-1)$
 - 验证： $y^a a^b \bmod p = g^M \bmod p$



example

- 生成密钥：使用者Alice选取素数 $p=2357$ 及 Z_{2357}^* 的生成元 $g=2$, Alice选取私钥 $x=1751$ 并计算 $g^x \bmod p = 2^{1751} \bmod 2357 = 1185$
A的公钥是 $p=2357$, $g=2$, $g^x=1185$
- 加密：为加密信息 $m=2035$, Bob选取一个随机整数 $k=1520$ 并计算 $a=2^{1520} \bmod 2357=1430$,
 $b=2035 \times 1185^{1520} \bmod 2357=697$
Bob发送 a, b 给Alice
- 解密：Alice计算 $a^{-x} \equiv 1430^{p-1-x} \equiv 1430^{605} \equiv 872 \pmod{2357}$
 $M \equiv b/a^x \equiv ba^{-x} \equiv 697 \times 872 \equiv 2035 \pmod{2357}$



ELGamal加密算法安全性

攻击**ELGamal**加密算法等价于解离散对数问题
要使用不同的随机数 k 来加密不同的信息

假设用同一个 k 来加密两个消息 m_1, m_2 , 所得到的密文分别为 $(a_1, b_1)(a_2, b_2)$, 则 $b_1/b_2 = m_1/m_2$, 故当 m_1 已知, m_2 可以很容易地计算出来。



5.椭圆曲线密码算法ECC



多数公钥密码（RSA，D-H）适用非常大的数或多项式



给密钥和信息的存储和处理带来很大的运算量

椭圆曲线是一个代替，可以用很小的尺寸得到同样的安全性

1985年，Neal Koblitz和Victor Miller分别独立地提出了椭圆曲线密码体制

ANSI、IEEE、ISO和NIST都制定了ECC标准草案



非对称密钥算法的优缺点

● 优点：

- 可以适用网络的开放性要求，密钥管理相对简单；
- 可以实现数字签名,认证鉴权和密钥交换等功能。

● 缺点：

- 算法一般比较复杂，加解密速度慢（ **RSA**算法最快的情况也比**DES**慢两个数量级，硬件实现时，比**DES**慢约**1000**倍。软件实现时比**DES**慢约**100**倍。 ）。

在公开密钥加密法应用中，生成较长密钥的技术属于尖端高科技



私有密钥法和公开密钥法比较

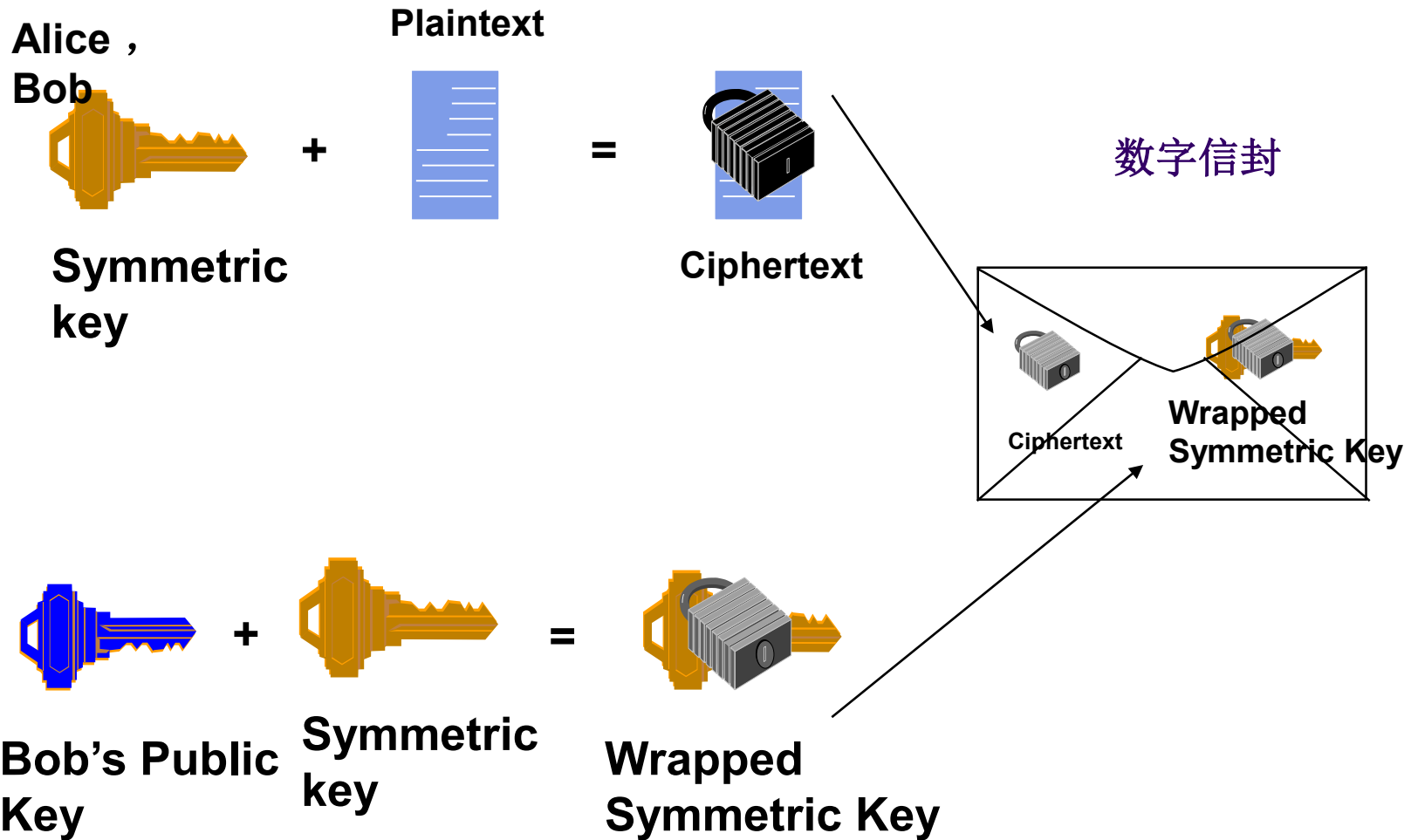


- 1) 加密、解密的处理效率
- 2) 密钥的分发与管理
- 3) 安全性
- 4) 数字签名和认证



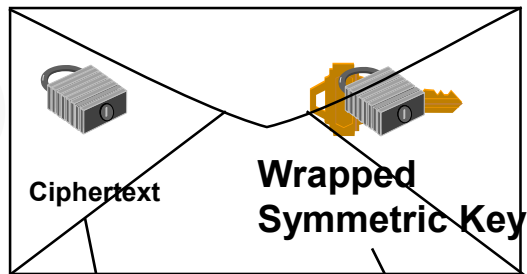
在实际业务系统中如何应用呢？

混合密钥体制：加密

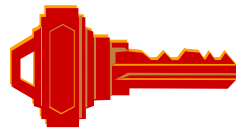


混合密钥体制：解密

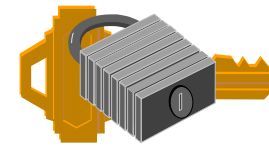
Alice or Bob?



Bob's
Private Key



+

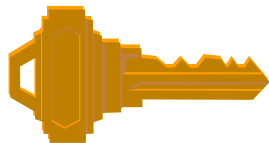


=



Wrapped
Symmetric Key

Symmetric
key



Symmetric
key

+



=

Ciphertext



Plaintext



数字信封的定义

- 在上述流程中利用接收方公开密钥对加密信息原文的密钥 P 进行加密后再定点传送，这好比用一个安全的“信封”把密钥 P 封装起来，所以称做数字信封。
- 因为数字信封是用消息接收方的公钥加密的，只能用接收方的私钥解密打开，别人无法得到信封中的密钥 P ，好像挂号信，必须有私人签章才能获得一样。
- 采用公开密钥加密法的数字信封，实质上是一个能分发、传播称密钥的安全通道。



相关开源软件及库



- **OpenSSL, SSL/TLS tool kits**, 也是最成功的加密库: 开源, 跨平台, 在windows,各种Linux,Unix, MAC OS等操作系统被广泛的使用, 很多大公司的产品底层加密库用的也是openssl,比如cisco vpn。
- **windows CryptoAPI**,对C/C++; C#, .Net: WSE中的安全库。用起来相对方便, 不需要另外的第三方的库文件, 无法跨平台, 无法获得源代码。如果想读写windows keystore, 必须用window cryptoapi。
- **NSS, firefox以及Netscape, Sun**的数字证书服务器, **LDAP**服务器底层都用的**NSS**。**NSS**的跨平台做得也很好, 它下面有一层**Netscape Portable Runtime**的支持跨平台的库。**NSS**功能也很强, 和openssl不相上下。使用起来不如 openssl方便。
- **MAC OS或iPhone**的开发: 苹果的**crypto**库
- **Java: bouncy castle**包
- **Python: M2Crypto** (A Python crypto and SSL toolkit)
- **C++: Crypto++**



相关开源软件及库



- 其它:



- **OpenSSH: FREE version of the SSH connectivity tools. SSH connectivity tools, provides secure tunneling capabilities and several authentication methods,**
- **OpenPGP implementation: GnuPG (GNU Privacy Guard或GPG)**



ABE(Attribute-Based Encryption)

- 基于属性的密码系统
 - 由IBE(identity-based encryption) 基于身份的系统发展而来： IBE直接使用用户的身份作为公钥,使得资源提供方无需在线查询用户的公钥证书
 - 一种数据访问控制方法
 - 使用属性集合作为群体的公钥
 - 用户私钥与属性集合对应，具有一定属性的用户才可读取正确明文,无需事先知道用户身份



思考



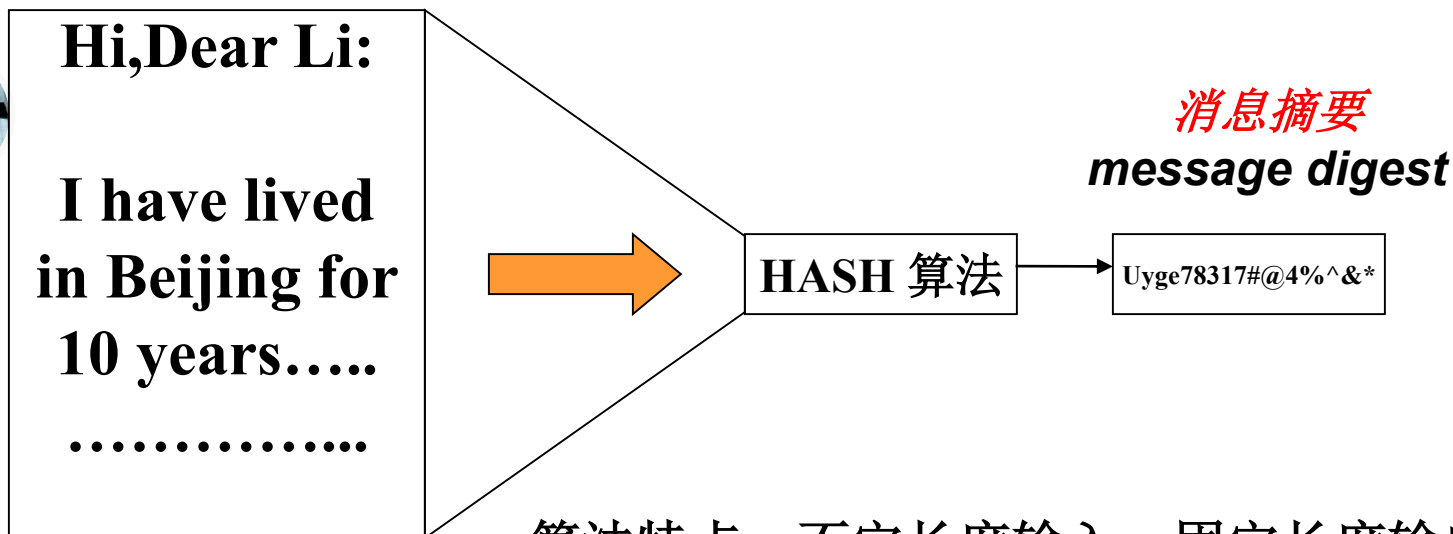
- 到这里为止，我们有了什么密码学的方法
- 完成了什么安全功能？
- 其它方面的安全性能如何保障呢？



散列 (Hash) 函数

- 函数 $y=H(x)$, 将任意长度的 x 变换成固定长度的 y
- 哈希函数有时也称为杂凑函数
- 单向Hash函数特性:
 - 1. 单向性, 任给 y , 计算 x , 使得 $y=H(x)$ 困难
 - 2. 快速性, 计算 $y=H(x)$ 容易
 - 3. 无碰撞, 寻找 $x_1 \neq x_2$, 满足 $H(x_1)=H(x_2)$ 是困难的.

One-way HASH(单向散列算法)



- 算法特点：不定长度输入，固定长度输出（MD5-16字节、SHA1-20字节）
- 输入很小的变动可引起输出较大变动
- 完全单向：
 - 已知输出无法推算出输入（无法让消息摘要不变而修改原文，**data integrity**）
 - 已知两个输出的差别无法推算出输入的差别



- 常用的hash 函数有 MD5, SHA, 及分组密码算法
 - Ron Rivest设计的MD (Standard For Message Digest, 消息摘要标准) 算法, MD5输入信息可以是任意长度, 输出是128bit.
 - MD4[Rivest 1990, 1992, 1995; RFC1320]
 - MD5是MD4的改进型[RFC1321]
 - MD2[RFC1319], 已被Rogier等于1995年攻破
 - 较早被标准化组织IETF接纳, 并已获得广泛应用
 - NIST设计的SHA (Secure Hash Algorithm) 称为安全散列算法, 是由美国政府开发的一种散列函数。它的输入是可变长度的信息, 输出是160bit的信息摘要。



SHA和SHA-1(secure Hash Algorithm)

- **NIST和NSA为配合数字签名标准DSS，设计了安全散列标准(SHS)**
- **1992年NIST制定了SHA（128位）**
- **1993年SHA成为标准（FIPS PUB 180）**
- **1994年修改产生SHA-1（160位）**
- **1995年SHA-1成为新的标准（FIPS PUB 180-1/RFC 3174），为兼容AES的安全性，NIST发布FIPS PUB 180-2，标准化SHA-256，SHA-384和SHA-512**
- **不仅仅可应用于数字签名中，它也可以应用到任何需要Hash算法的地方**
- **目前还没有针对SHA有效的攻击**
- **SHA/SHA-1采用了与MD4相似的设计准则，其结构也类似于MD4，但其输出为160bit**
 - **输入：消息长度 $<2^{64}$**
 - **输出：160bit消息摘要**
 - **处理：以512bit输入数据块为单位**



SHA与MD4和MD5的比较

	MD4	SHA	MD5
Hash值	128bit	160bit	128bit
分组处理长	512bit	512bit	512bit
基本字长	32bit	32bit	32bit
步数	48(3*16)	80(4*20)	64(4*16)
消息长	$\leq 2^{64}\text{bit}$	2^{64}bit	不限
基本逻辑函数	3	3(第2,4轮相同)	4
常数个数	3	4	64
速度		约为MD4的3/4	约为MD4的1/7



单向散列函数的抗碰撞性



散列函数的应用，它的安全性是什么意思？



抗碰撞性的能力体现出单向散列函数对抗生日攻击和伪造的能力。

- 弱抗碰撞性 (**Weak collision resistance**) :

- 对于任意给定的 M ，找到满足 $M \neq N$ 且 $H(M) = H(N)$ 的 N ，在计算上是不可行的；
- 即给定报文和摘要，其它人无法**伪造**

- 强抗碰撞性 (**Strong collision resistance**) :

- 找到任何满足 $H(x) = H(y)$ 的偶对 (x, y) 在计算上是不可行的。
- 发送方自己无法产生两个摘要相等的报文，以便**抵赖**已发送的报文
注：强无碰撞自然含弱无碰撞！



哈希函数-生日攻击

- 如果采用传输加密的散列值和不加密的报文 M ，攻击者需要找到 M^- ，使得 $H(M^-) = H(M)$ ，以便使用替代报文来欺骗接收者。
- 生日问题：一个教室中，最少应有多少个学生，才使至少有两人具有相同生日的概率不小于 $1/2$ ？
概率结果与人的直觉是相违背的。实际上只需23人，即任找23人，至少两人具有相同生日的概率 $\geq 1/2$



生日攻击实例



- **A**准备两份合同**M**和**M⁻**，一份**B**会同意，一份会取走他的财产而被拒绝
- **A**对**M**和**M⁻** 各做32处微小变化（保持原意），分别产生 2^{32} 个64位hash值
- 根据前面的结论，超过0.5的概率能找到一个经过微小变化的**M**和**M⁻**（**M1**和**M1⁻**），它们的hash值相同
- **A**提交**M1**，经**B**审阅后产生64位hash值并对该值签名，返回给**A**
- **A**用**M1⁻** 替换**M1**

结论：Hash必须足够长（ 128， 160， 224， 256， ...）



消息认证MAC (Message Authentication Code)



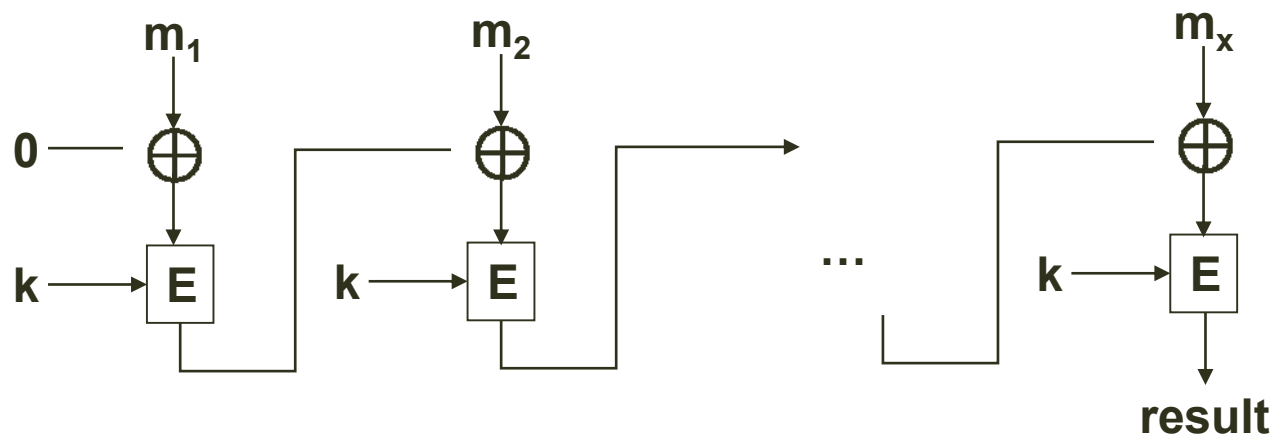
● 消息摘要Message Digest



- 在网络安全目标中，要求信息在生成、存储或传输过程中保证不被偶然或蓄意地删除、修改、伪造、乱序、重放、插入等破坏和丢失，因此需要一个较为安全的标准和算法，以保证数据的完整性。
- 消息摘要算法采用单向散列（**hash**）函数从明文产生摘要密文。摘要密文又称为数字指纹（**Digital Fingerprint**）、数据认证码DAC（**Data authentication code**）、篡改检验码MDC（**Manipulation detection code**）
- 消息的散列值由只有通信双方知道的秘密密钥**k**来控制，此时散列值称作消息认证码**MAC (Message Authentication Code)**
 - 先计算**message digest**，再对其进行加密
 - **CBC-MAC**（**ISO8731-1, ISO 9797, ANSI X9.9, IEEE802.11i**）



- **CBC-MAC (cipher block chaining message authentication code)**





数字签名



● 概念和作用



● 消息认证的局限性

- 消息认证使收方能验证消息发送者及所发消息内容是否被篡改过。当收发者之间没有利害冲突时，这对于防止第三者的破坏来说是足够了。但当收者和发者之间有利害冲突时，就无法解决他们之间的纠纷

- 信宿方可以伪造消息并称消息发自信源方
- 信源方可以否认曾发送过某消息，因为信宿方可以伪造消息，所以无法证明信源方确实发送过该消息

- 在收发双方不能完全信任的情况下，引入数字签名来解决上述问题

- 数字签名的作用相当于手写签名



数字签名的特点



- 传统签名的基本特点
 - 不可重用,与被签的文件在物理上不可分割
 - 不可抵赖,签名者不能否认自己的签名
 - 签名不能被伪造
 - 容易被验证

- 数字签名是传统签名的数字化
 - 能与所签文件“绑定”
 - 签名者不能否认自己的签名
 - 容易被自动验证
 - 签名不能被伪造



数字签名常见算法



- 普通数字签名算法

- RSA

- ElGamal /DSS/DSA

- ECDSA Elliptic Curve Digital Signature Algorithm

- 盲签名算法

- 群签名算法

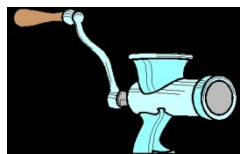
数字签名 Digital Signatures



Alice's
Private Key



Message



+ Hash

Message



Signature

Digest
Algorithm

Message



Signature

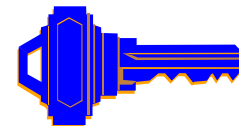


?

Hash

=

Hash



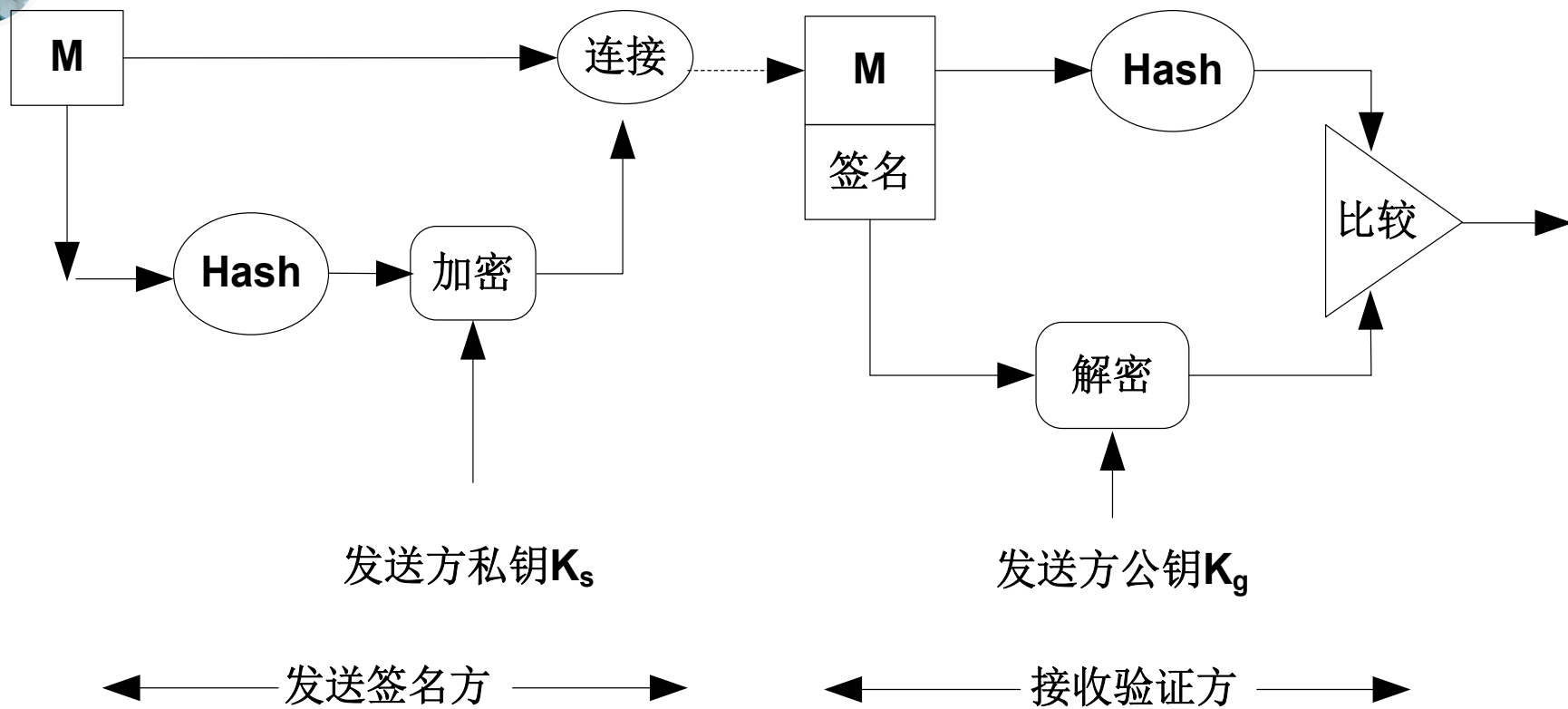
Alice's Public
Key



Signature



RSA算法的签名和验证过程





数字签名标准DSS

- 美国国家标准与技术研究所NIST发布的联邦信息处理标准FIPS186，称为数字签名标准DSS (Digital Signature Standard)
- 于1991年最初提出，1994年12月1日采纳为标准DSS，2000年发布DSS的扩充版，即FIPS 186-2
- DSS为ElGamal和Schnorr签名方案的改进，其使用的算法记为DSA (Digital Signature Algorithm)。此算法由D. W. Kravitz设计。DSS使用了SHA，安全性是基于求离散对数的困难性
- DSS和DSA是有所不同的：前者是一个标准，后者是标准中使用的算法
- 只能用于数字签名，不能用于加密或密钥分配
- DSS是一个产生签名比验证签名快得多的方案，验证签名太慢！



DSA与RSA



- 1991年8月，NIST提出将DSA用于其数字签名标准DSS。
- DSA是由NSA设计，是Schnorr和ElGamal的变型。



- 反对DSA的意见主要包括：
 - 1) DSA不能用于加密或密钥分配
 - 2) DSA是由NSA研制的，并且算法中可能存在陷门
 - 3) DSA比RSA要慢
二者产生签名的速度相同，但验证签名时DSA要慢10至40倍，其产生密钥的速度比RSA快。
 - 4) **RSA是事实上的标准**
 - 5) DSA的选择过程不公开，并且提供的分析时间不充分
 - 6) DSA可能侵犯其他专利:DSA侵犯了三个专利：Diffie-Hellman、Merkle-Hellman、Schnorr，前两个1997年已到期，Schnorr专利到2008年。
 - 7) 密钥长度太小
模数最初设置为512位，后来可以变化到1024位。

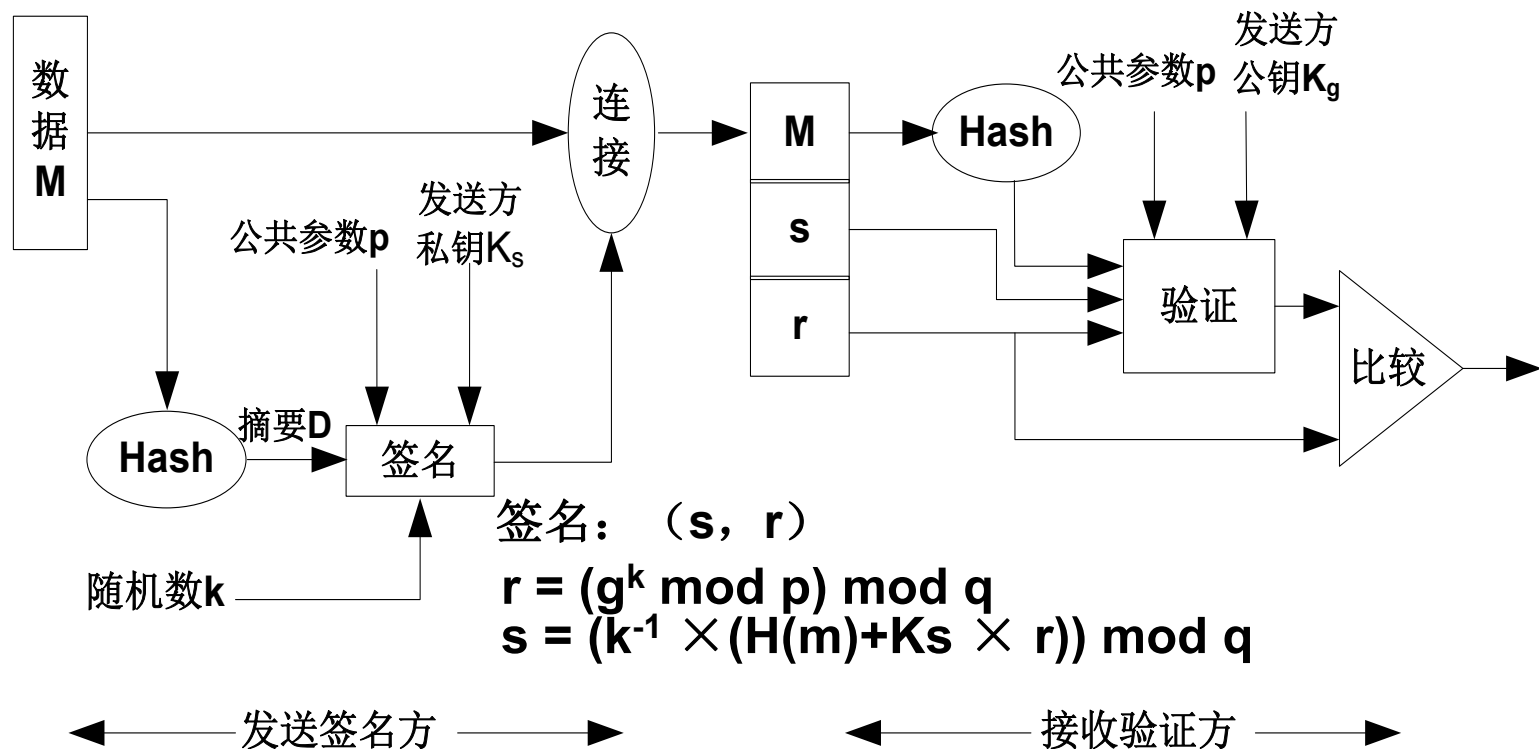


DSA (cont.)

- 全局公钥 (p, q, g) (p, g 可在一组用户中共享)
 - p : L 位长的素数, L 大于512小于1024, 是64的倍数
 - q 是 $(p-1)$ 的素因子, 为160比特的素数
 - $g = h^{(p-1)/q} \mod p$, 且 $1 < h < (p-1)$, 使得 $h^{(p-1)/q} \mod p > 1$
- 用户私钥 x : x 为 $0 < x < q$ 内的随机数
- 用户公钥 y : $y = g^x \mod p$
- 签名: 产生一个小于 q 的随机数 k
 - $r = (g^k \mod p) \mod q$
 - $s = (k^{-1} \times (H(m) + x \times r)) \mod q$
 - r, s 为 m 的签名, $H(.)$ 是 HASH 函数, DSS 指定为 SHA
- 验证:
 - $w = s^{-1} \mod q$
 - $u1 = (H(m) \times w) \mod q$
 - $u2 = r \times w \mod q$
 - $v = ((g^{u1} \times y^{u2}) \mod p) \mod q$, 如果 $v=r$ 则签名被验证。



DSS的签名和验证过程



- DSS采用的是SHA散列函数计算消息摘要D。签名方在签名时，将消息摘要D和一个随机数k一起作为签名函数的输入。签名函数使用发送方的私钥K_s和一组公共参数P (p, q, g)，产生两个输出 (s, r) 作为签名结果。验证方就是通过比较接收到的签名结果与自己计算出来的签名结果，根据它们是否相等来判断签名的有效性。



其他签名方案



- **GOST**是俄罗斯的签名方案，**1995**年开始使用。
- 离散对数签名方案：通用签名方案。**ElGamal**、**Schnorr**、**DSA**等。
- **ESIGN**是由日本**NTT**提出的数字签名方案。对于同样的密钥和签名长度，其安全性至少与**RSA**或**DSA**相同，速度要快的多。
- **ECDSA** Elliptic Curve Digital Signature Algorithm
- 近年来其他许多公钥算法不断被提出又被破译



更多的数字签名



- 不同应用的身份认证，完整性，不可抵赖需求



- 盲签名

- Chaum在1983年提出
- 一类盲签名（盲消息签名）
 - 有时需要某人对一个文件签名，但又不让他知道文件内容
 - 适应于电子选举、电子支付和电子现金协议，金融合同的签署、遗嘱签署、CA 证书的颁发等方面
- 除满足一般数字签名的不可抵赖、不可伪造外，还满足：
 1. 签名者对其所签署的消息是不可见的，即签名者不知道他所签署消息的具体内容。
 2. 签名消息不可追踪，即当签名消息被公布后，签名者无法知道这是他哪次签署的。

消息拥有者先将消息盲化，签名者对盲化的消息进行签名，消息拥有者对签字除去盲因子，得到签名者关于原消息的签名



群签名(group signature)

群签名

- 群体密码学由Desmedt于1987年提出，群签名是群体密码学中的课题，1991由Chaum和van Heyst提出，其特点有：
- 正确性Correctness: 只有群体成员才能代表群体签名；签名可被验证
- 匿名性Anonymity
 - 群体中的任意成员可以以匿名的方式代表整个群体对消息进行签名；
 - 接收到签名的人可以用单个群公钥验证群签名，但不可能知道由群体中哪个成员所签；
- 可追踪性Traceability: 发生争议时可由群体中的成员或可信赖机构识别群签名的签名者。
- 不可陷害性Non-frameability: 攻击者（包括群成员）不能伪造一个有效的签名但却追踪到一个无辜的群成员



环签名 Ring signature



- a group of entities each have public/private key pairs, $(PK_1, SK_1), (PK_2, SK_2), \dots, (PK_n, SK_n)$. Party i can compute a ring signature σ on a message m , on input $(m, SK_i, PK_1, \dots, SK_n, PK_n)$. Anyone can check the validity of a ring signature given σ, m , and the public keys involved, PK_1, \dots, PK_n .
- It should be hard for anyone to create a valid ring signature on any message for any group without knowing any of the secret keys for that group.

How to leak a secret, Ron Rivest, Adi Shamir, and Yael Tauman, ASIACRYPT 2001.



门限签名

- 门限签名是 n 个人里有 k 个人以上一起参与才能生成签名($n > k$)。 k 就是这个“限”。
- 群签名是 n 个人一起生成签名。
- 环签名主要实现的功能是匿名性，环签名和门限签名结合的门限环签名 在自组网中用途较广



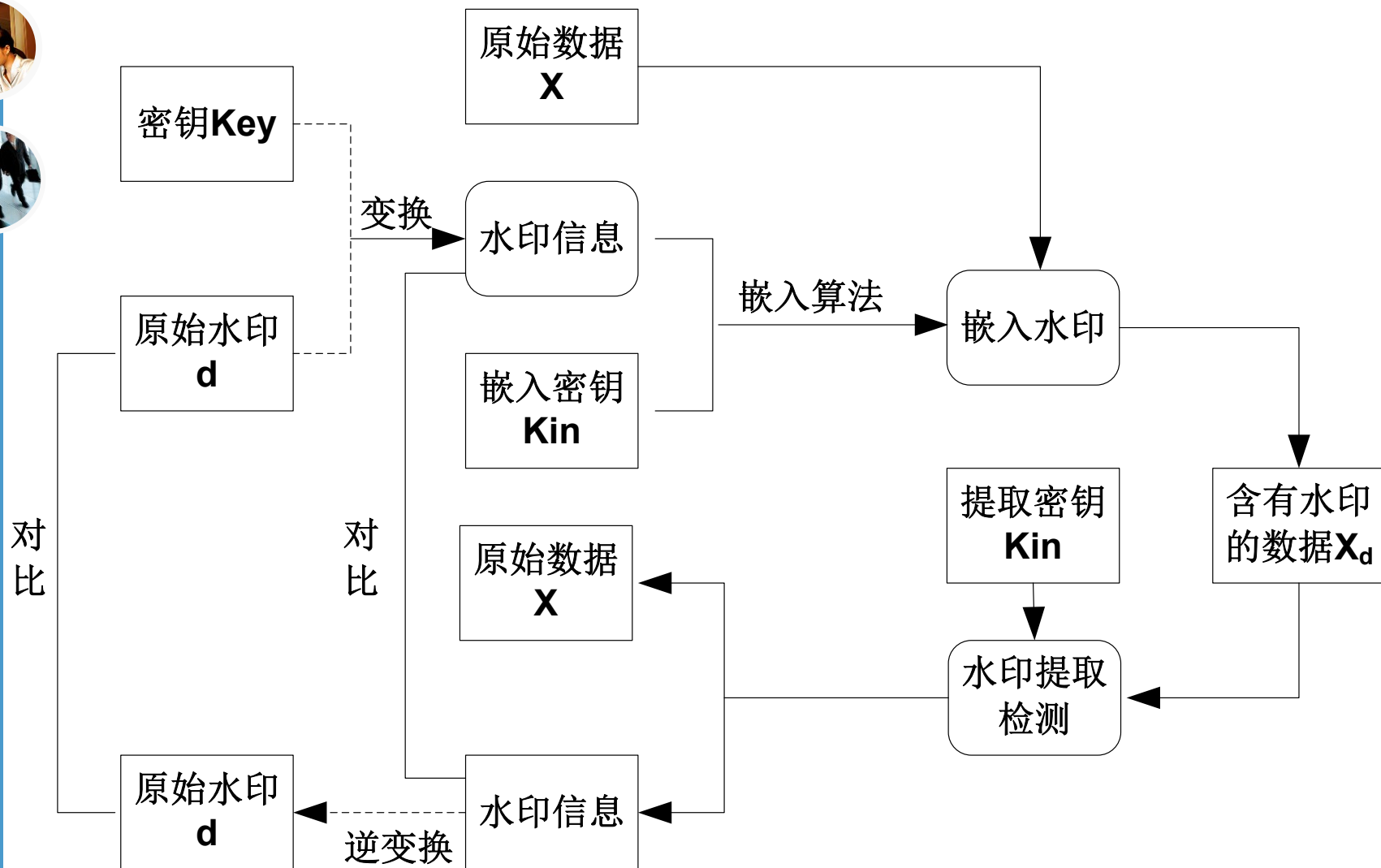
数字水印



- 数字水印（**Digital Watermark**）是指永久镶嵌在其它数据（主要指宿主数据）中具有可鉴别性的数字信号或数字模式。
- 数字水印的主要特征有：
 - 不可感知性（**imperceptible**）：包括视觉上的不可见性和水印算法的不可推断性。
 - 鲁棒性（**Robustness**）：嵌入水印必须难以被一般算法清除。也就是说多媒体信息中的水印能够抵抗各种对数据的破坏，如A/D、D/A转换、重量化、滤波、平滑、有失真压缩以及旋转、平移、缩放及分割等几何变换和恶意的攻击等。
 - 可证明性：指对嵌有水印信息的图像，可以通过水印检测器证明嵌入水印的存在。
 - 自恢复性：指含水印的图像在经受一系列攻击后（图像可能有较大的破坏），水印信息也经过了各种操作或变换。但可以通过一定的算法从剩余的图像片段中恢复出水印信息，而不需要整个原始图像的特性。
 - 安全保密性：数字水印系统使用一个或多个密钥以确保安全，防止修改和擦除。同时若与密码学进行有机的结合，对数据可起到双重加密作用。



数字水印的工作过程

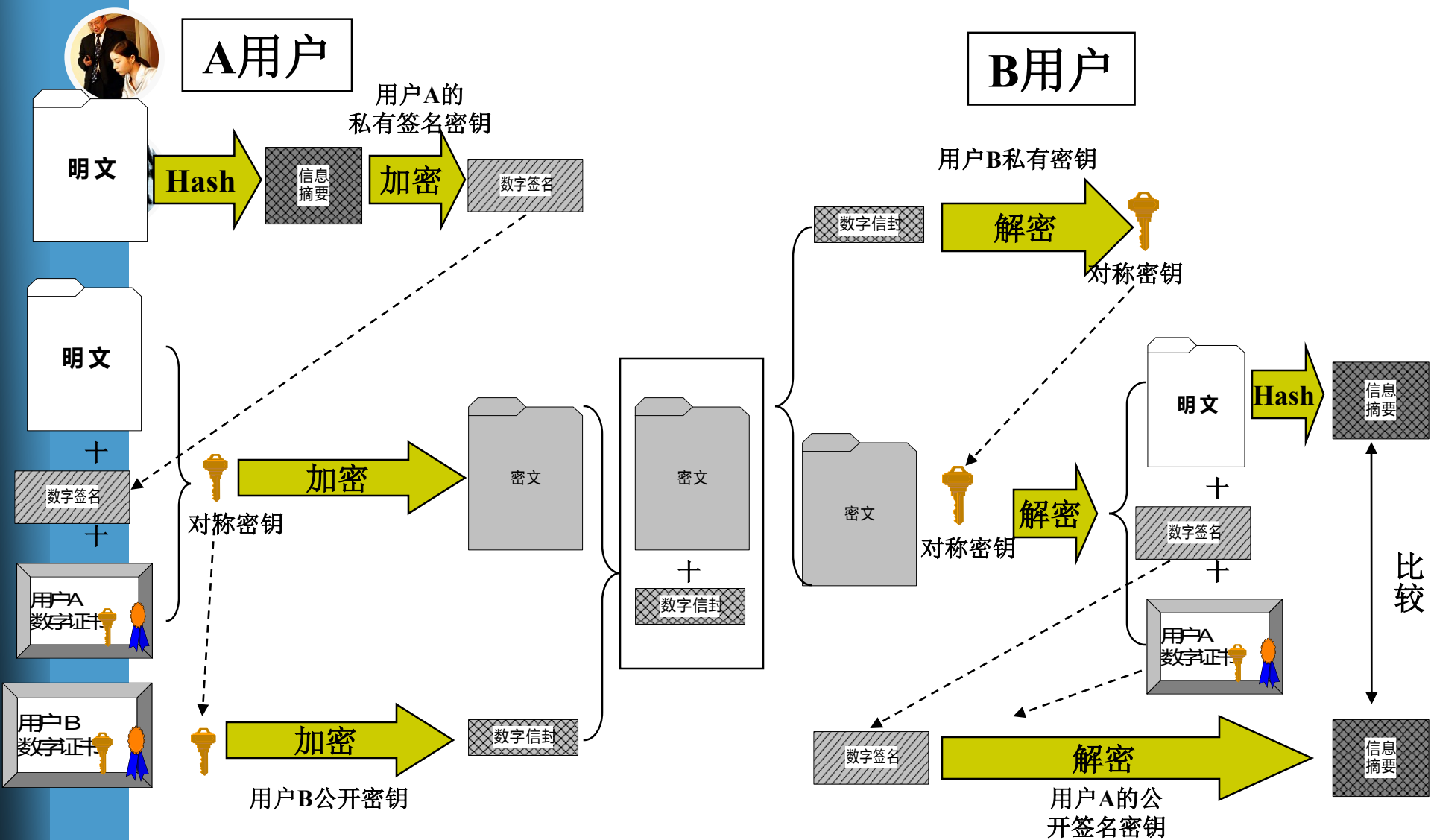




总结：几个概念

- 加密：对称加密，非对称加密特点与应用
- 消息摘要
- 消息认证码**MAC**
- 数字签名
- 数字信封
- 数字证书

总结：一个完整的数据加解密/身份认证流程





密钥管理的意义



- 密钥管理技术是信息安全的核心技术之一。包括密钥的：
 - 产生：密钥空间（减小密钥空间，如只用小写字母，可使穷举攻击难度大大下降），随机密钥（随机数发生器质量），测试弱密钥（有些特定密钥可能比其它的不安全）
 - **生成**：（参见安全协议部分，以TLS, IPSEC为例）
 - **分发**：安全性，扩展性
 - **验证**：验证密钥的发送方，传输差错检测，解密中的错误检测
 - 使用：
 - **存储**：口令保护， token等
 - 备份：密钥托管，秘密分享协议（密钥分段）
 - 保护（泄露检测，有效期，旧密钥销毁）
 - **吊销、更新**等。



密钥的组织结构——多层密钥系统



基本思想:用密钥保护密钥
一个系统中常有多多个密钥

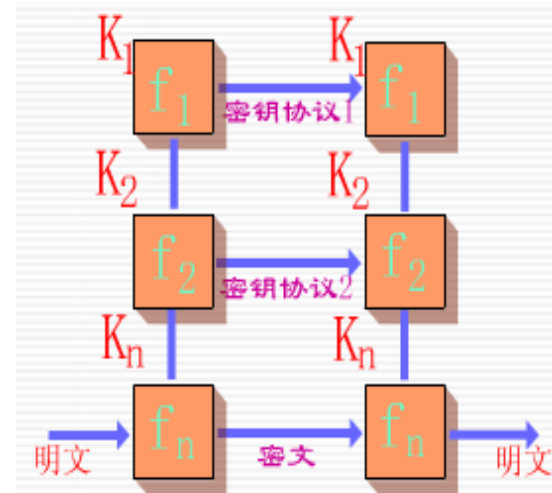


例:

会话密钥或数据加密密钥: 最底层的密钥, 直接对数据进行加密和解密;

密钥加密密钥: 最底层上所有的密钥, 对下一层密钥进行加密;

主密钥: 最高层的密钥, 是密钥系统的核心。



例: 签名key,
加密key



密钥分发(分配, 交换)



- 密钥分发是密钥管理中的一个关键因素，目前已有许多密钥分配协议，但其安全性是一个很重要的问题。
- 按分发的内容
 - 秘密密钥的分发
 - 公开密钥的分发



密钥分发中的威胁1: 消息重放

- 攻击者简单复制一条消息，以后再重新发送它；
- 可能导致向敌人暴露会话密钥，或成功地冒充其他人；

应对：在认证交换中使用**序列号**，使每一个消息报文有唯一编号。
仅当收到的消息序数顺序合法时才接受之。

问题：同步，双方必须保持上次消息的序号。



抵抗消息重放的方法



两种更为一般的方法：



➤ 1、时间戳：

- ✓ **A**接受一个新消息仅当该消息包含一个时间戳，该时间戳在**A**看来，是足够接近**A**所知道的当前时间；
- ✓ 这种方法要求不同参与者之间的时钟需要同步。
- ✓ 安全的时间服务器实现时钟同步可能是最好的方法

➤ 2、挑战/应答方式。（**Challenge/Response**）

- ✓ **A**期望从**B**获得一个新消息，首先发给**B**一个临时值 (**challenge**)，并要求后续从**B**收到的消息 (**response**) 包含这个临时值相关的正确信息。
- ✓ 不适应非连接性的应用



密钥分发中的威胁2：中间人攻击



(1) Alice将她的公钥传递给BOB。MALLORY截取这个密钥，并将自己的公钥传送给BOB（伪造）



(2) BOB将他的公钥传递给Alice。MALLORY截取这个密钥，并将自己的公钥传送给Alice

(3) 当Alice将用BOB的公钥加密消息传递给BOB时，MALLORY截取它。由于消息实际上是用MALLORY的公钥加密的，他就用自己的私钥解密。再用BOB的公钥对消息重新加密，并将它传送给BOB。

(4) BOB将用Alice的公钥加密消息传递给Alice时，MALLORY截取它。由于消息实际上是用MALLORY的公钥加密的，他就用自己的私钥解密。再用Alice的公钥对消息重新加密，并将它传送给Alice。

即使Alice和BOB的公开密钥存储在数据库中，这种攻击亦是可行的。MALLORY能够截取Alice的数据库查询，并用自己的公开密钥代替BOB的公开密钥，对BOB他可做同样的事情。（参考书1P34）

对策：使用数字签名的密钥交换



由Ron Rivest和Adi Shamir发明的**联锁协议**是阻止中间人攻击的好办法。



- (1) Alice将她的公开密钥传送给BOB
- (2) BOB将她的公开密钥传送给Alice
- (3) Alice用BOB的公开密钥加密她的消息，并将加密消息的**一半**传送给BOB（无法解密）
- (4) BOB用Alice的公开密钥加密她的消息，并将加密消息的**一半**传送给Alice
- (5) Alice将加密的另一半消息传递给BOB
- (6) BOB将Alice的两半消息合在一起，并用他的私人密钥解密；BOB将她加密的另一半消息传递给Alice
- (7) Alice将BOB两半消息合在一起，并用她的私人密钥解密。

(参考书1P34)



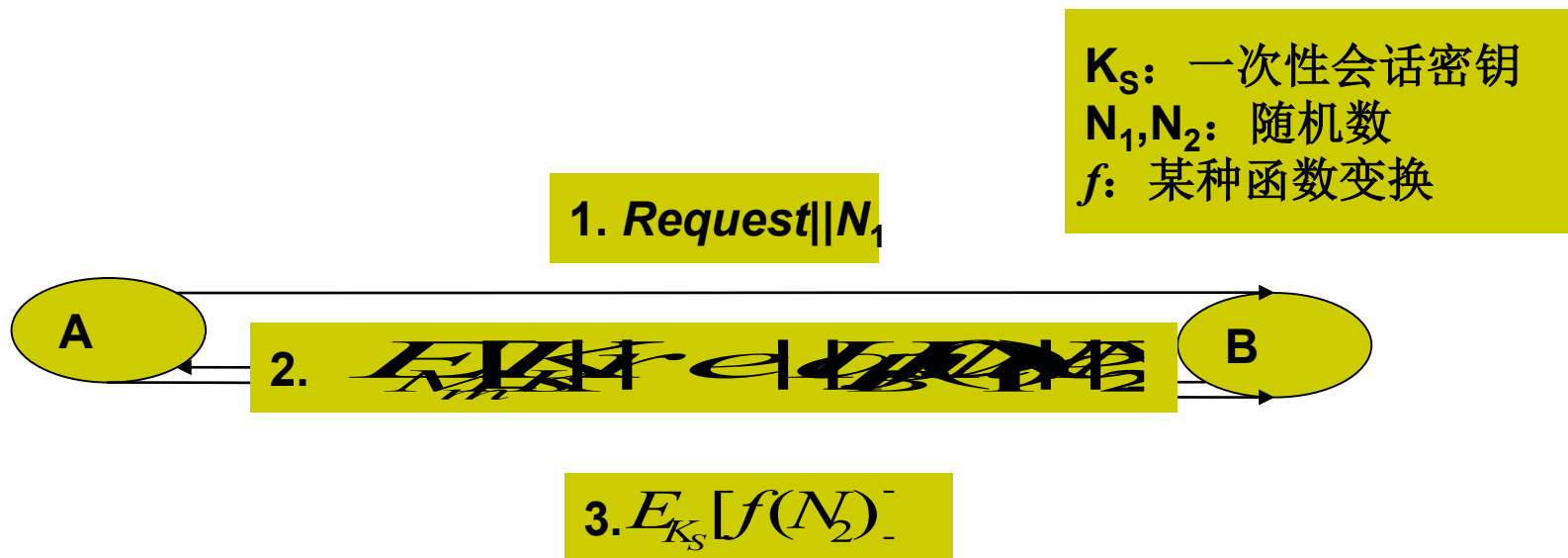
密钥分发

- 典型的自动密钥分配途径有两类：集中式分配方案和分布式（无中心的）分配方案。
 - 集中式分配是指利用网络中的“密钥管理中心”来集中管理系统中的密钥，“密钥管理中心”接受系统中用户的请求，为用户提供安全分配密钥的服务。
 - 分布式分配方案取决于它们自己的协商，不受任何其他方面的限制。



无中心的密钥控制

用户A和B建立会话密钥的过程

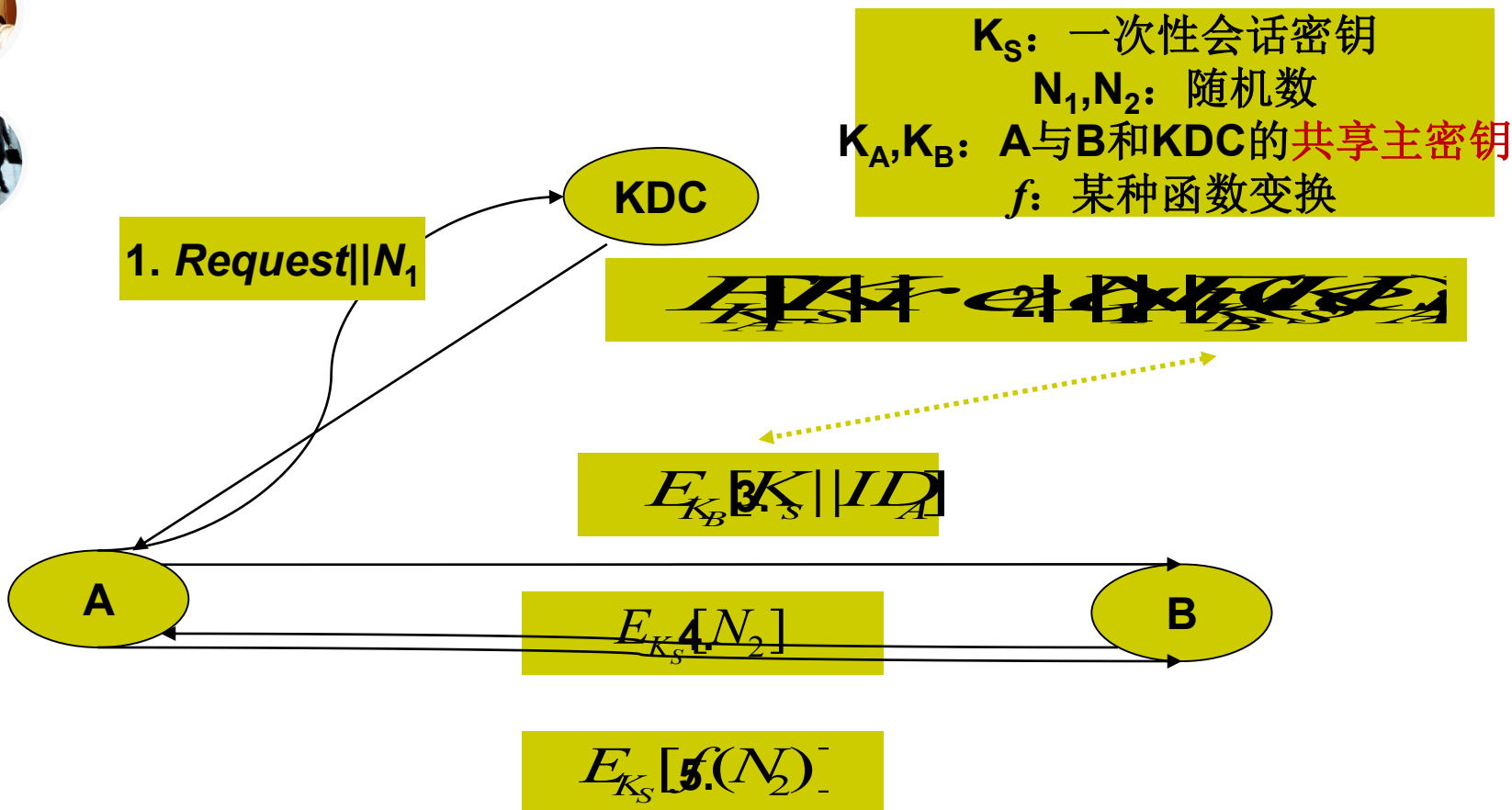


要两两拥有共享密钥，
一共需要 $n(n-1)/2$ 的密钥

$$K_m = ?$$



密钥分配中心 (KDC, Key Distribution Center)



问题：假定攻击方C已经掌握A和B之间通信的一个老的会话密钥。
C可以在第3步冒充A利用老的会话密钥欺骗B。除非B记住所有以前使用的
与A通信的会话密钥，否则B无法判断这是一个重放攻击。



- (1) **Alice**呼叫**KDC**，并请求一个与**Bob**通信的会话密钥。
 - (2) **KDC**产生一随机会话密钥，并对它的两个副本加密：一个用**Alice**的密钥，另一个用**Bob**的密钥加密。**KDC**发送这两个副本给**Alice**
 - (3) **Alice**对她的会话密钥的副本解密
 - (4) **Alice**将**Bob**的会话密钥的副本传给**Bob**
 - (5) **Bob**对他的会话密钥的副本解密
- Alice**和**Bob**用这个会话密钥安全通信。

这个协议依赖于**KDC**的绝对安全性。此协议的缺点是：

- (1) **MALLORY**可能破坏**KDC**，整个网络都会破坏。他有**KDC**与每个用户共享的秘密密钥；
- (2) **KDC**可能成为瓶颈。他必须参与每一次密钥交换，如果**KDC**失败了，这个系统就会被破坏。



公开密钥密码学的密钥交换（数字信封）

基础的混合密码系统中，**Alice**和**Bob**使用公钥算法协商**会话密钥**，并用协商的会话密钥加密数据。在一些实际的现实中，**Alice**和**Bob**签名的公开密钥可在数据库中获得。这使得密钥交换更容易，即使**Bob**从来没有听说过**Alice**，**Alice**亦能够把消息安全地发送给**BOB**。

- （1）**Alice**从**KDC**得到**BOB**的公钥。
- （2）**Alice**产生随机会话密钥，用**Bob**的公钥加密它，然后将它传给**BOB**。
- （3）**BOB**用他的私钥解密**Alice**的消息。
- （4）两人用同一会话密钥对他们的通信进行加密。



可用于广播



Alice想把消息传送给**BOB**、**CAROL**和**DAVE**

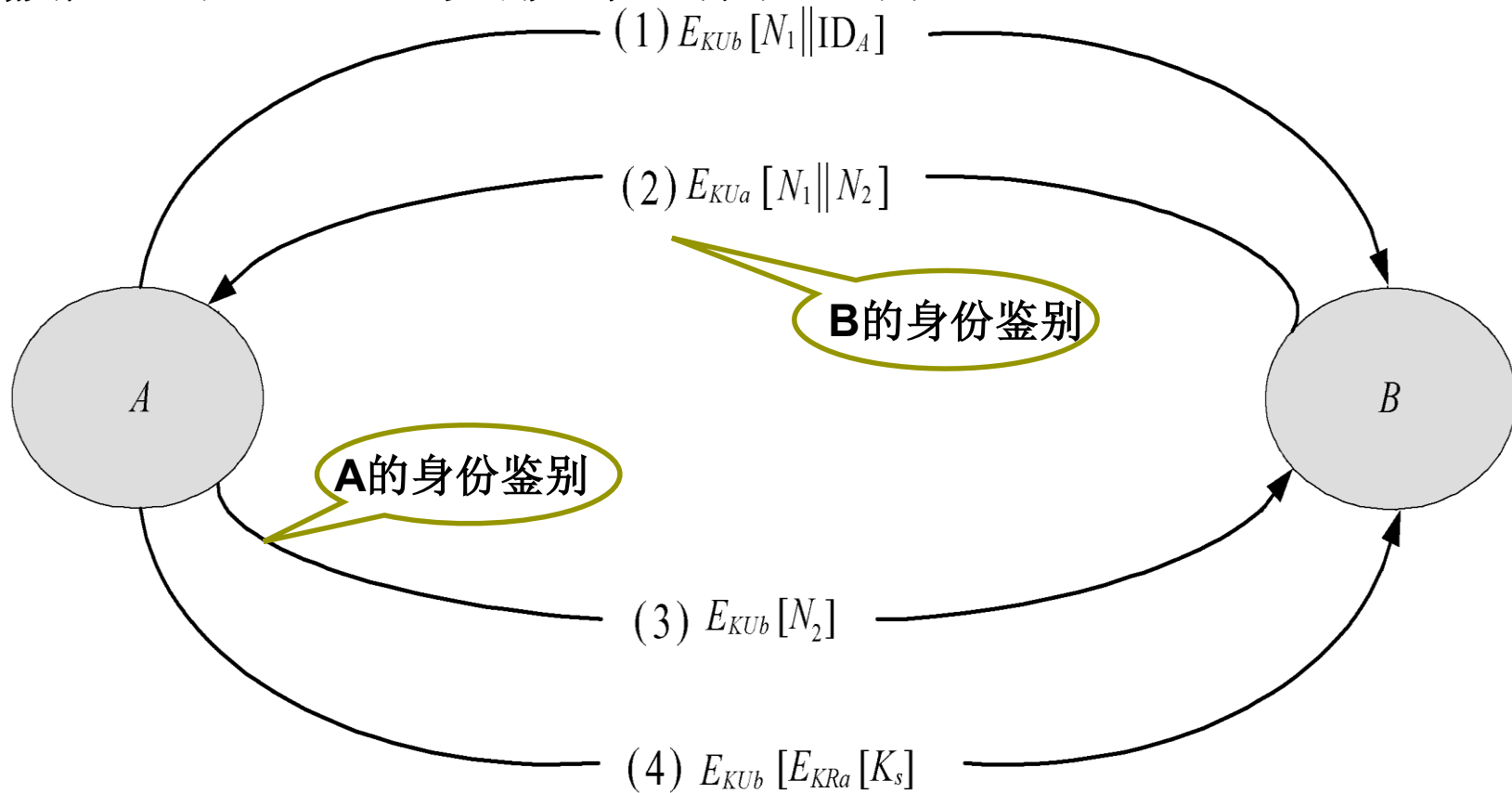


- (1) **Alice**产生一随机会话密钥**K**，并用**K**加密**M**: $E_K(M)$
- (2) **Alice**从数据库中得到**BOB**、**CAROL**和**DAVE**的公开密钥。
- (3) **Alice**用**BOB**的公开密钥加密**K**，用**CAROL**的公开密钥加密**K**，用**DAVE**的公开密钥加密**K**: $E_B(K)$ 、 $E_C(K)$ 、 $E_D(K)$
- (4) **Alice**广播加密的消息和所有加密的会话密钥传送给**BOB**、**CAROL**和**DAVE**: $E_K(M)$ 、 $E_B(K)$ 、 $E_C(K)$ 、 $E_D(K)$ 。
- (5) **BOB**、**CAROL**、**DAVE**用他们的私钥将**Alice**的会话密钥解密。
- (6) **BOB**、**CAROL**、**DAVE**用会话密钥将**Alice**的消息解密。



具有保密和身份认证的会话密钥分配（无中心）

假定**A**和**B**已经交换了公开密钥



KUb: B的公钥

N1, N2的作用: 通过私钥进行鉴权



DH密钥交换 (P368)

- 1976年Diffie和Hellman发明了DH算法，该算法是第一个公开密钥算法，其安全性源于在有限域上计算离散对数比计算指数更为困难。
- DH算法用于密钥分配，但不能用于加密或解密。
- DH算法已经申请美国和加拿大的专利，目前该专利已经到期。
- DH算法容易受到中间人攻击。



DH密钥交换

- **Alice和Bob**需要首先协商好一个大的素数 n 和 g ， g 是模 n 的本原元。 n 和 g 可以是公开的，可以在一组用户中使用。协议如下：
 - **Alice**选取一个大的随机数 x 并发送给**Bob**: $X = g^x \bmod n$;
 - **Bob**选取一个大的随机数 y 并发送给**Alice**: $Y = g^y \bmod n$;
 - **Alice**计算 $k = Y^x \bmod n$;
 - **Bob**计算 $k' = X^y \bmod n$ 。
 - $k=k'=g^{xy} \bmod n$ 是**Alice**和**Bob**协商好的密钥。
- n 和 g 的选取对系统的安全性具有很大的影响， n 应该足够大，并且 $(n-1)/2$ 是一个素数。
- **RFC 2631 Diffie-Hellman Key Agreement Method**

密钥交换不需要事先存在的基础设施



Diffie-Hellman Example



用户Alice和Bob想交换密钥:



约定素数 $p=353$ 和 $a=3$

随机选择密钥:

—A chooses $x_A=97$, B chooses $x_B=233$

计算公钥:

— $y_A = 3^{97} \bmod 353 = 40$ (Alice)

— $y_B = 3^{233} \bmod 353 = 248$ (Bob)

计算共享的会话密钥:

$K_{AB} = y_B^{x_A} \bmod 353 = 248^{97} = 160$ (Alice)

$K_{AB} = y_A^{x_B} \bmod 353 = 40^{233} = 160$ (Bob)



- 三方或多方Diffie-Hellman

- 设Alice、Bob和Carol产生秘密密钥:

- 1) Alice选取一个大随机整数 x , 并且发送给Bob $X = g^x \bmod n$;
- 2) Bob选取一个大的随机整数 y , 并且发送给Carol $Y = g^y \bmod n$;
- 3) Carol选取一个大的随机整数 z , 并且发送给Alice $Z = g^z \bmod n$;
- 4) Alice发送给Bob $Z' = Z^x \bmod n$;
- 5) Bob发送给Carol $X' = X^y \bmod n$;
- 6) Carol发送给Alice $Y' = Y^z \bmod n$;
- 7) Alice计算 $k = Y'^x \bmod n$;
- 8) Bob计算 $k = Z'^y \bmod n$;
- 9) Carol计算 $k = X'^z \bmod n$;



- 优势:

- 仅当需要时才生成密钥，减小了将密钥存储很长一段时间而致使遭受攻击的机会。
- 除对全局参数的约定外，密钥交换不需要事先存在的基础结构。

- 不足:

- **DoS**攻击：计算密集型的。受攻击者花费了相对多的计算资源来求解无用的幂系数而不是在做真正的工作。
- 重发攻击。
- 无身份认证
- 遭受中间人攻击。



选择随机数 $x < p$
计算 $Y_A = g^x \bmod p$

计算 $K = Y_c^x$
 $= g^{xz} \bmod p$

用户A

选择随机数 $y < p$
计算 $Y_B = g^y \bmod p$

计算 $K = Y_c^y$
 $= g^{yz} \bmod p$

用户B

用户C假冒 A,B

中间人
攻击

选择随机数 $z < p$
计算 $Y_c = g^z \bmod p$

计算 $K = Y_A^z$
 $= g^{xz} \bmod p$

计算 $K = Y_B^z$
 $= g^{yz} \bmod p$

用户C



Oakley算法

- Oakley算法是对**Diffie-Hellman**密钥交换算法的优化，它保留了后者的优点，同时克服了其弱点。
- Oakley算法具有五个重要特征
 - 1、它采用称为**cookie**程序的机制来对抗阻塞(**DoS**)攻击。
 - 2、它使得双方能够协商一个全局参数集合。
 - 3、它使用了当前时间（现时）来保证抵抗重发攻击。
 - 4、它能够交换**Diffie-Hellman**公开密钥。
 - 5、它对**Diffie-Hellman**交换进行鉴别以对抗中间人的攻击。

以后还会见到它



oakley可以使用三个不同的鉴别方法

- 1、 数字签名：通过签署一个相互可以获得的散列代码来对交换进行鉴别；每一方都使用自己的私钥对散列代码加密。散列代码是在一些重要参数上生成的，如用户**ID**和时戳。
- 2、 公开密钥加密：通过使用发送者的私钥对诸如**ID**和现时等参数进行加密来鉴别交换。
- 3、 对称密钥加密：通过使用某种共享密钥对交换参数进行对称加密，实现交换的鉴别。



EKE（加密密钥交换p371）



- EKE是Steve Bellovin和Michael Merrit设计的加秘密钥交换协议；
- EKE同时使用对称和公钥密码给计算机网络提供了安全性和鉴别；
- EKE使用共享的秘密密钥加密随机产生的公开密钥；
- EKE已经申请专利。

公开密钥安全的重要性



EKE (cont.)



- 设**Alice**和**Bob**共享一个公共口令**P**，利用这个协议，他们可以产生一个公共会话密钥**K**：
 - **Alice**产生一个随机公开密钥/私人密钥对，并用对称算法(**P**作为密钥): $E_p(K')$ 对公开密钥**K'**进行加密，将结果发送给**Bob**: $E_p(K')$
 - **Bob**知道**P**，他解密此消息得到**K'**，然后产生一个随机会话密钥**K**，用从**Alice**处得到的公开密钥和**P**来加密**K**，并将之发送给**Alice**, $E_p(E_{K'}(K))$
 - **Alice**解密该消息获得**K**，她产生一个随机串**R_A**，用**K**加密后发送给**Bob**: $E_K(R_A)$
 - **Bob**解密该消息得到**R_A**，他产生一个随机串**R_B**，用**K**加密后发送给**Alice**: $E_K(R_A, R_B)$
 - **Alice**解密得到**R_A** 和**R_B**，如果他得到的**R_A**与自己产生的一致，则用**K**加密**R_B**后发送给**Bob**: $E_K(R_B)$
 - **Bob**解密该消息得到**R_B**，如果他得到的**R_B**与自己产生的一致，则协议完成，可以用**K**作为会话密钥进行通信。





● 密钥管理体制主要有三种：

- 适用于封闭网的技术，以传统的密钥管理中心为代表的**KMI**机制（**Key Management Infrastructure**，密钥管理基础设施）；
- 适用于开放网的**PKI**机制（**Public Key Infrastructure**，公开密钥基础设施）；
- 适用于规模化专用网的**SPK**