信息安全中的密码

孙 锐

（sunrui@is.ac.cn）

2002年12月28日

主要内容

1. “密码”在信息安全中的定位
2. 一些概念
3. 对称密码
4. 公钥密码

1. “密码”在信息安全中的定位

“信息安全”概念的演变

过去（80年代）	昨天（90年代）	今天
COMSEC	**INFOSEC**	**IA**
保密： 加密 访问控制	保护： 信息的保密性 信息的完整性 信息的可用性	保护： 信息的保密性 信息的完整性 信息的可用性 检测： 系统脆弱性 入侵 响应： 入侵 病毒 恢复： 系统的可用性

基本的安全需求

- 保密性
- 完整性
- 可用性
- 可控性
- 不可否认性

安全服务（ISO /IEC 7498-2）

1. 认证（鉴别）（**Authentication**）
   （包括对等实体鉴别和数据源鉴别）
1. 访问控制（**Access Control**）
2. 数据保密（**Data Confidentiality**）
3. 数据完整性（**Data Integrity**）
4. 不可否认（**Non-repudiation**）

注：**ISO/IEC 7498-2—第二版—1994-11-15—信息技术—开放式系统互连基本参考模式：安全结构**

安全机制（ISO /IEC 7498-2）

（1）加密机制（Encryption ）
（2）数字签名机制（Digital Signature Mechanisms）
（3）访问控制机制（Access Control Mechanisms）
（4）数据完整性机制（Data Integrity Mechanisms）
（5）鉴别交换机制（Authentication Mechanisms)
（6）通信业务填充机制（Traffic Padding Mechanisms）
（7）路由控制机制（Routing Control Mechanisms）
（8）公证机制（Notarization Mechanisms）

业务	机制							
	数据加密	数字签名	访问控制	数据完整性	鉴别交换	业务流填充	路由控制	公证
对等实体身份鉴别	Y	Y	N	N	Y	N	N	N
数据原发鉴别	Y	Y	N	N	N	N	N	N
访问控制	N	N	Y	N	N	N	N	N
连接保密性	Y	N	N	N	N	N	Y	N
无连接保密性	Y	N	N	N	N	N	Y	N
选择字段保密性	Y	N	N	N	N	N	N	N
信息流保密性	Y	N	N	N	N	Y	Y	N
带恢复的连接完整性	Y	N	N	Y	N	N	N	N
不带恢复的连接完整性	Y	N	N	Y	N	N	N	N
选择字段连接完整性	Y	N	N	Y	N	N	N	N
无连接完整性	Y	Y	N	Y	N	N	N	N
选择字段无连接完整性	Y	Y	N	Y	N	N	N	N
抗抵赖,带数据原发证据	N	Y	N	Y	N	N	N	Y
抗抵赖,带交付证据	N	Y	N	Y	N	N	N	Y

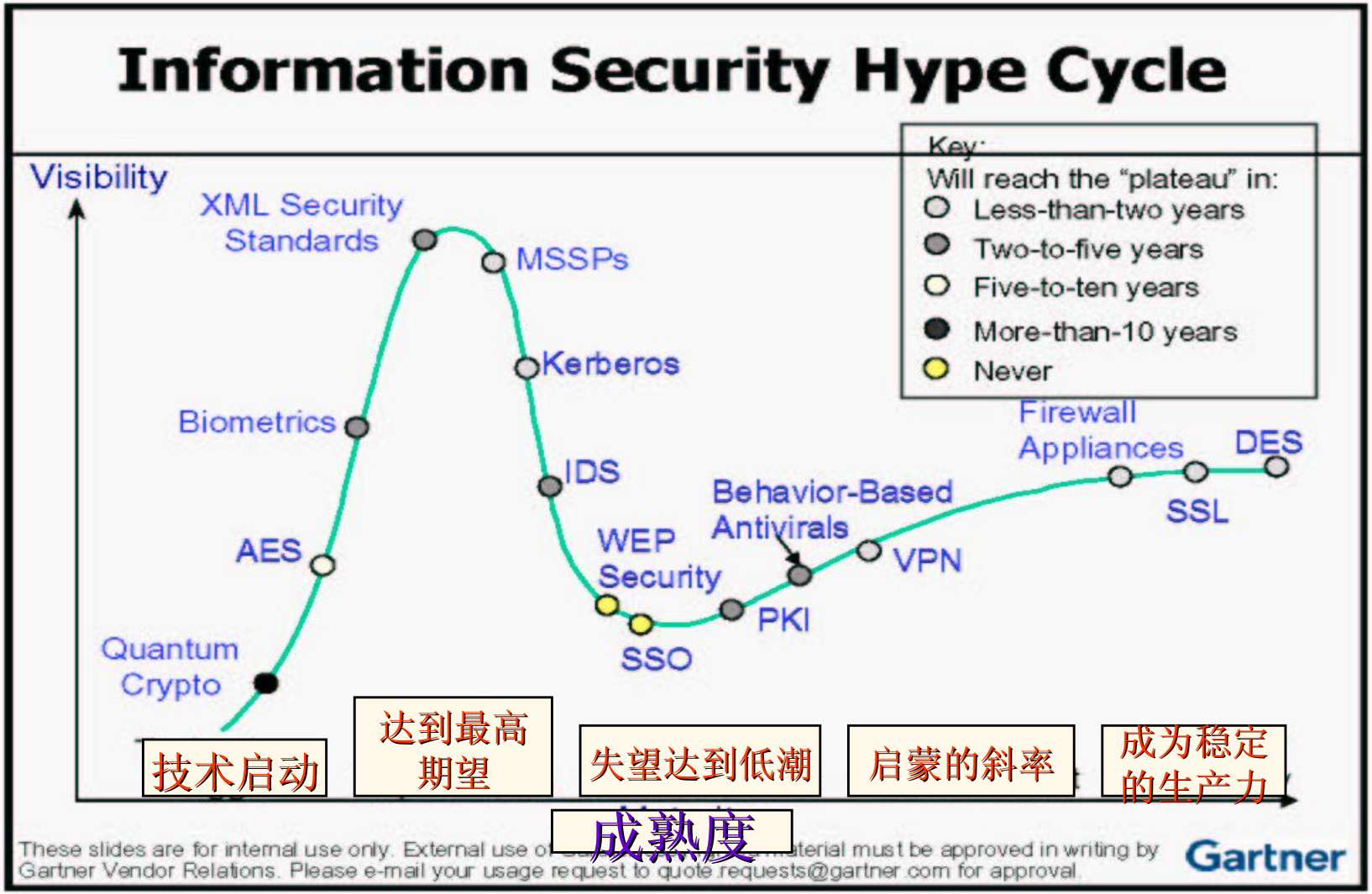
围绕“安全”的若干视角

- 安全策略——实施规程
- 基于产品——基于过程
- 静 态 防 护——动态自适应
- 风险规避——风险化解
  - 可　信——保　证
  - 集　中——分　治
- 网络安全——内容安全
- 技术属性——社会属性

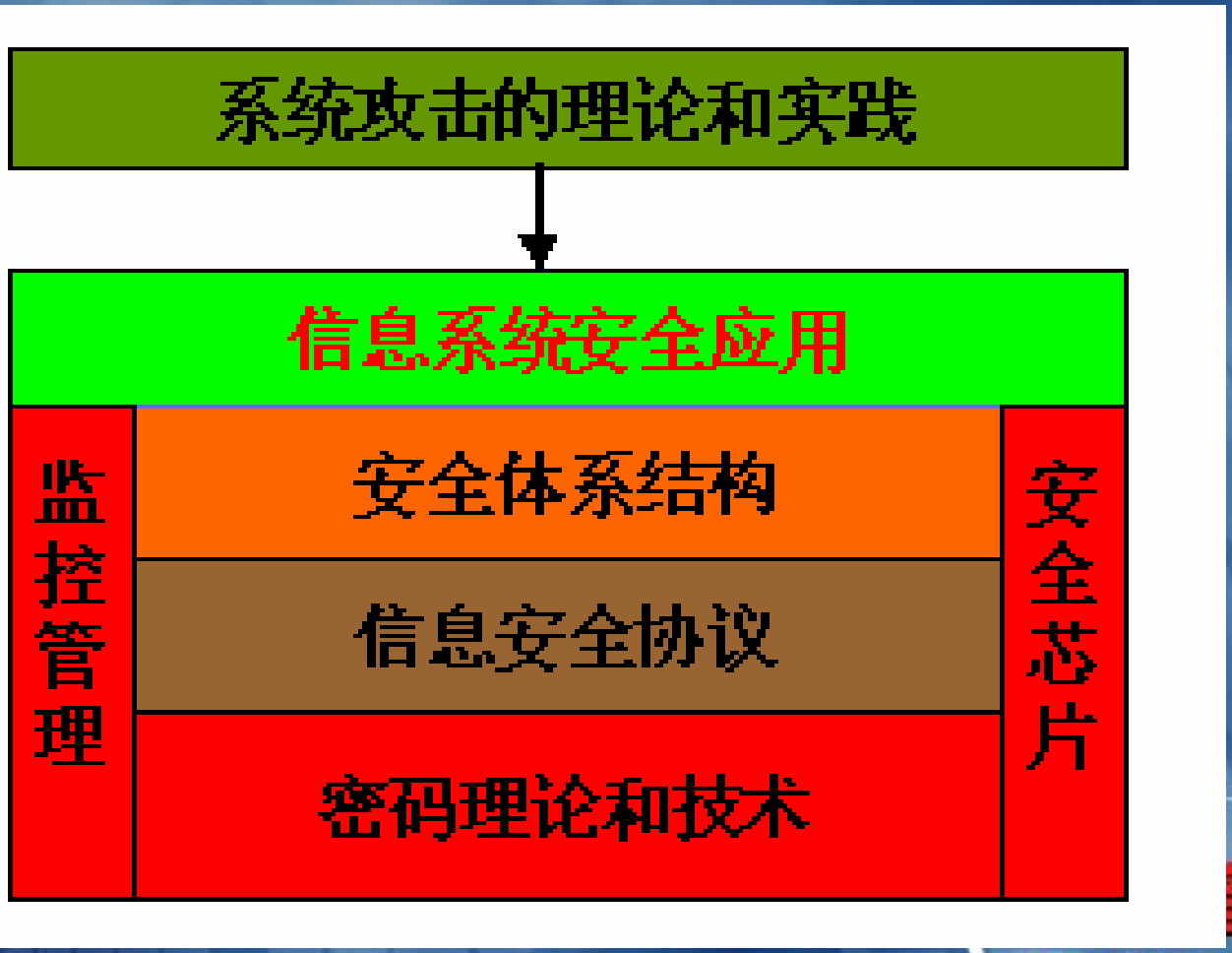
研究重点

- 信息安全的策略与模型
- 信息安全体系结构
- 安全对策的模糊理论
- 安全特性的语义学分析与形式化描述
- 安全机制及其开发方法
- *密码学与访问控制*
- 网络中安全环境的实现
- 网络免疫与可生存性
- 测评认证与安全保证
- 风险控制与安全管理

技术现状与预测


Information Security Hype Cycle
Visibility
Key:
Will reach the "plateau" in:
Less-than-two years
Two-to-five years
Five-to-ten years
More-than-10 years
Never
XML Security Standards
MSSPs
Kerberos
Biometrics
IDS
AES
WEP Security
Behavior-Based Antivirals
Firewall Appliances
DES
SSL
VPN
PKI
SSO
Quantum Crypto
技术启动
达到最高期望
失望达到低潮
启蒙的斜率
成为稳定的生产力
成熟度
These slides are for internal use only. External use of
material must be approved in writing by Gartner Vendor Relations. Please e-mail your usage request to quote requests@gartner.com for approval.
Gartner

技术要素



对技术要素的形象说明

密　码 —— 核心

安 全 协 议 —— 桥梁

安全体系结构 —— 基础

安 全 芯 片 —— 关键

监 控 管 理 —— 保障

系统攻击评测 —— 考验

2. 一些概念

- 密码学——研究密码系统或通信安全的一门学科。分为：
- 密码编码学：使得消息保密的学科。
- 密码分析学（或称密码破译学）：研究加密消息的破译的学科（精于此道的人被称为密码学家，现代的密码学家通常是理论数学家）。

三个发展阶段

- 第一个阶段：几千年前到1949年。

  密码设计与密码分析多基于直觉与经验。
- 第二个阶段：1949年到1975年。

  1949年，Shannon，“保密系统的信息理论”

  1967年，Kahn，《破译者》

  70年代初，IBM，有关密码学的几篇技术报告
- 第三阶段：1976年至今。

  1976年，Diffe和Hellman，《密码学新方向》

密码系统模型

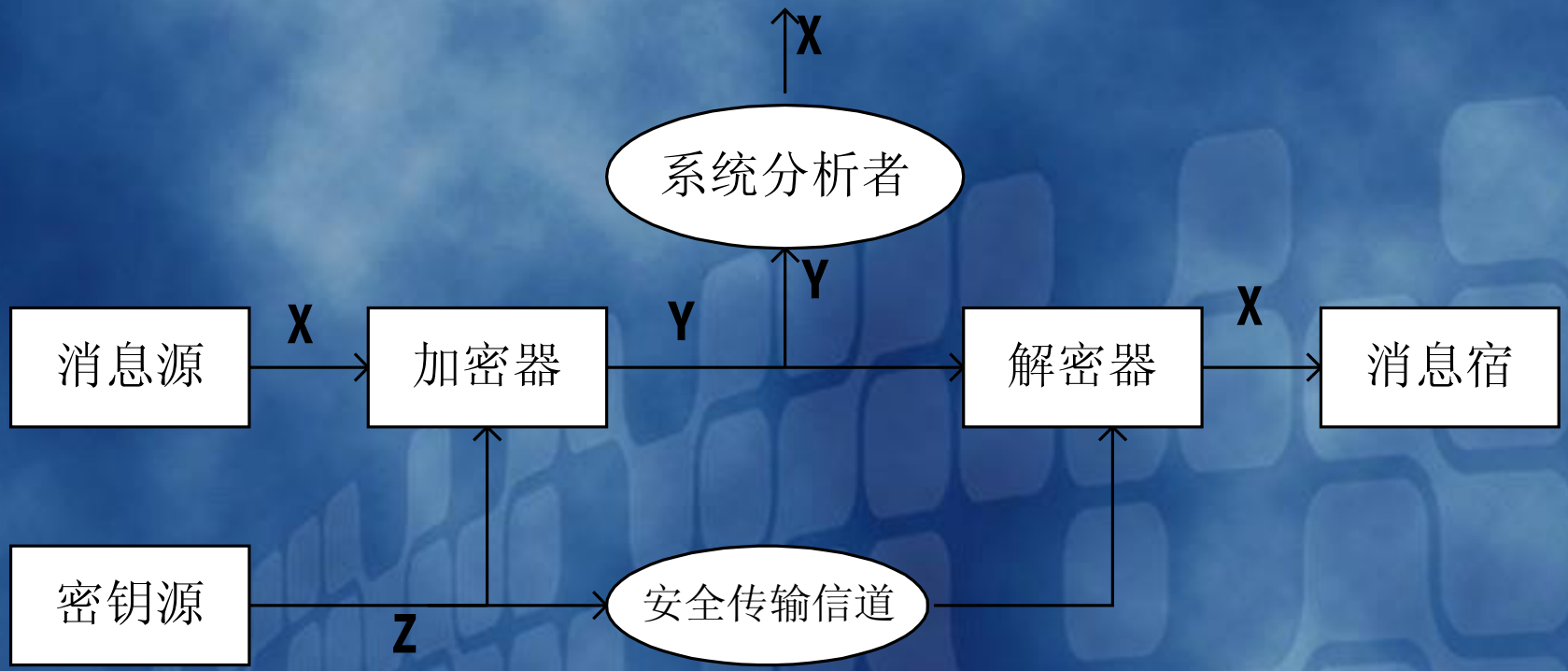
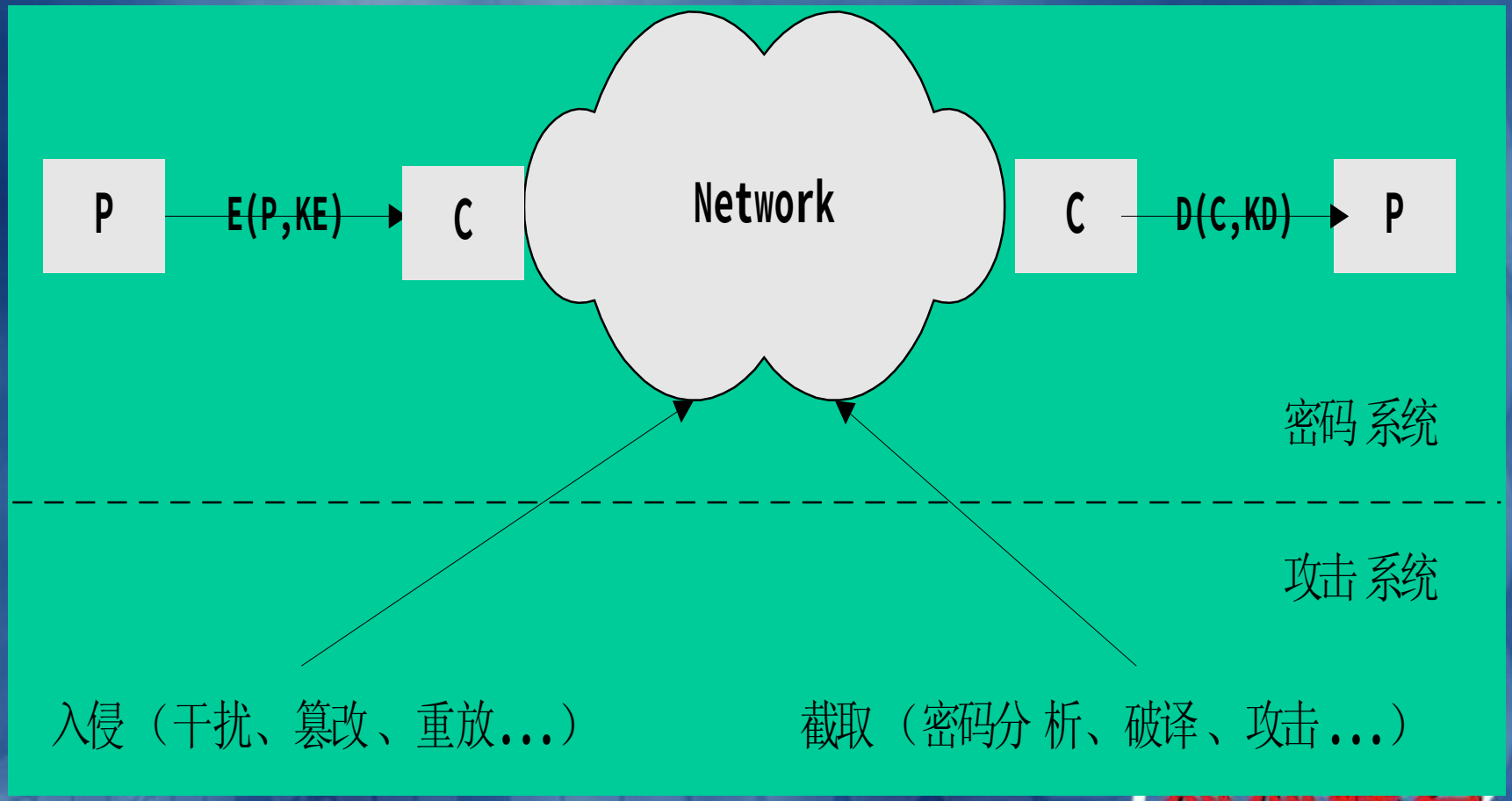


图1：香农模型

密 码 体 系



几个术语

- 明文（Plaintext）：消息源要传输的消息X（文本文件、位图、数字化的语言、数字化的视频图像等）。
- 密文（Cipher）：明文通过加密器加密后得到密文。
- 加密：记为E。其逆过程称为解密，记为D。

几个术语（续）

- 加密算法：对明文进行加密时所采用的一组规则或变换。
- 解密算法：对密文进行解密时所采用的一组规则或变换。
- 加密和解密通常都是在一组密钥的控制下进行的，分别称为加密密钥和解密密钥。
- 一个密码系统（或称为密码体制或简称为密码）由算法以及所有可能的明文、密文和密钥（分别称为明文空间、密文空间和密钥空间）组成。

密码体制

- 根据密钥的特点分为：

对称密码体制（单钥/私钥密码体制）：

加密密钥和解密密钥相同，或彼此间容易相互确定

非对称密码体制（双钥/公钥密码体制）：

加密密钥和解密密钥不同，而且从一个难于推出另一个的密码体制。

密码体制

- 根据加密方式不同，私钥密码可分为：

流密码（序列密码）：

将明文按字符逐位加密的密码体制

分组密码：

将明文分组进行加密

关于私钥密码体制

- 代表： DES（1977年，美国国家标准局颁布）
- 密码需要经过安全的密码通道由发方传给收方。

密码体制的安全性=密钥的安全性

- 优点：**1.** 安全性算法高， **2.** 加密速度快。
- 缺点：

1. 网络规模的扩大使密钥管理成为一个难点；
2. 无法解决消息确认问题；
3. 缺乏自动检测密钥泄露的能力。

关于公钥密码体制

- 代表： RSA
  （1977年，*Rivest, Shamir*和*Adleman*）
- 加密密钥和解密密钥是不同的，此时不需要安全通道来传送密码。
- 优点：1．不存在密钥管理的问题
  2．可以拥有数字签名等新功能。
- 缺点：1．算法一般比较复杂
  2．加解密速度慢

- 网络中的加密普遍采用双钥和单钥密码相结合的混合加密体制
- *加解密：采用私钥密码*
- *密码传送:采用公钥密码*
- 既解决了密钥管理的困难，又解决了加解密速度慢的问题。

密码分析

- Kerckhoff假设：

  假定密码分析者知道所使用的密码系统。

  不应该将密码系统的安全性建立分析者不知道密码系统的前提下，在设计一个密码系统时，我们的目的是在Kerckhoff假设的前提下实现安全。

- 密码分析者是在不知道密钥的情况下，从密文恢复出明文，成功的密码分析不仅能够恢复除消息的源文和密钥，而且能够发现密码体制的弱点，从而控制通信。
- 常用的分析可分为四类：唯密文攻击，已知明文攻击，选择明文攻击，选择密文攻击。

- 唯密文攻击(Ciphertext-only Attack)
- 已知明文攻击(Known-plaintext Attack)
- 选择明文攻击(Chosen-plaintext Attack)及自适应选择明文攻击(Adaptive-chosen-plaintext Attack)
- 选择密文攻击(Chosen-ciphertext Attack)

密码学的功能

- 基本功能：提供保密性，即使非授权者无法知道消息的内容，这容易理解。
- 此外，也具有以下作用：

  *鉴别：*消息接收者应该能够确认消息来源。

  *完整性：*消息接收者应该能够验证消息在传输过程中未被改变。

  *不可否认性：*发送方不可能虚假地否认他发送的消息。

一个好的密码系统应该满足的要求

- (1) 系统即使无法达到理论上不可破，也要达到实际上不可破。即：从截获的密文或已知的明文-密文对，要确定密钥或任意明文在计算上是不可行的。
- (2) 系统的保密性依赖于密钥，而非对于加密体制或算法的保密。
- (3) 加密和解密算法适用于密钥空间中的所有元素。
- （4）系统既易于实现又便于使用。

密码技术研究进展：

- 包括公钥密码、分组密码、序列密码、认证码、数字签名、Hash函数、身份识别、密钥管理等基于数学的密码技术
- 包括信息隐形等基于生物特征的鉴别技术。

新型密码：

- 量子密码(Quantum Cryptography)
- 热流密码(Heat Flow Cryptography)
- 混沌密码(Chaos Cryptography)
- 图视密码(Visual Cryptography)

其中，量子密码理论：

- 1969年Wiesner提出了共轭编码的概念
- 量子密码理论源于共轭编码概念
  - 基于单光子量信道中测不准原理的
  - 基于量子相关信道中Bell 原理的
  - 基于两个非正交量子态性质的

3. 对称密码

分组算法

- 明文为分组长度为m的序列，密文为分组长度为n的序列，加密与解密过程由密钥控制。
- 两个主要优点：易于标准化，易于实现同步。
- 局限性：分组加密不便于隐藏明文的数据模式，对重放、插入、删除等攻击方式的抵御能力不强等。
- 通过在加密过程中采用合理的记忆组件（流密码的设计思想），能够消除这些局限性。

分组算法设计原则

- ***Shannon*的混乱原则和扩散原则：**

为保证密码的安全性

混乱原则——为了避免密码分析者利用明文与密文之间的依赖关系进行破译，密码的设计应该保证这种依赖关系足够复杂。

扩散原则——为避免密码分析者对密钥逐段破译，密码的设计应该保证密钥的每位数字能够影响密文中的多位数字；同时，为了避免避免密码分析者利用明文的统计特性，密码的设计应该保证明文的每位数字能够影响密文中的多位数字，从而隐藏明文的统计特性。

乘积密码对两种或更多的简单密码进行逐次应用，是一种实现上述混乱原则和扩散原则的有效方法。

分组算法设计原则（续）

- *必须结合预定的实现方法进行考虑。*

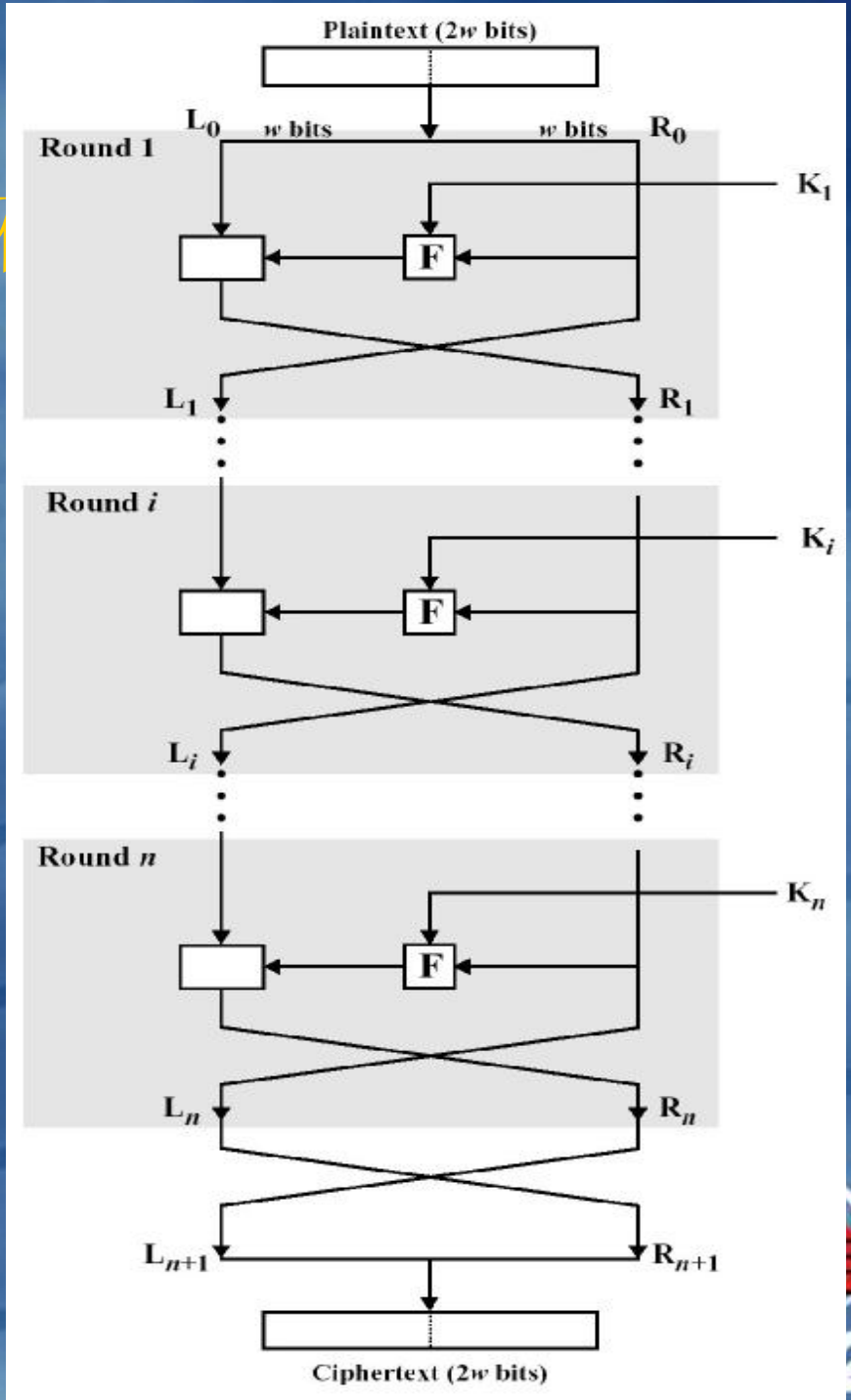
*硬件实现：高速率。*此时，加密与解密可以用同样的器件实现。为了适应超大规模集成电路的实现途径，分组密码应该具有标准的组件结构。

*软件实现：灵活、代价小。*此时，分组密码的运算在子模块上通过简单的运算进行。这些子模块的大小应该能够适应软件编程的特点，同时，这些运算也应该是软件编程容易实现的运算（例如一些标准的加法、乘法、移位等基本指令）。

Feistel密

对于一个长度为n（偶数）的分组，将其分为长度为n/2的两部分，分别记为L和R。按照如下方法定义迭代型的分组密码算法：

记**f**为任意轮函数，**Ki**为第轮使用的子密钥，第**i**轮的输出决定于第**i-1**轮的输出。



分组算法—DES

- 最初起源于美国商业部所属美国国家标准局（National Bureau of Standards-NBS）（该机构为现在的美国国家标准与技术研究所NIST的前身）早在1972年制定的一项计算机数据保护标准发展规划：

  开发一个单独的标准密码算法。

分组算法—DES

- 1973年5月15日，NBS在联邦记录1973（Federal Register1973）中发布了公开征集标准密码算法的请求，并确定了如下设计标准：

  必须提供较高的安全性；

  必须完全确定并且易于理解；

  安全性不应依赖于算法本身，而是应该依赖于密钥；

  必须对所有的用户有效；

  必须适用于各种应用；

  必须能够通过价格合理的电子器件得以实现；

  必须能够有效使用；

  必须能够得以验证；

  必须能够得以出口。

NBS最后确定的算法是在IBM提交的Lucifer密码算法基础上的一种改进。**FIPS PUB 46，1977年7月15日生效。**

分组算法—DES

- 1997年1月28日，RSA公布的13项私钥挑战赛中有一项是针对DES。
  - •Contest identifier: DES
  - •Cipher: DES
  - •Start of contest: 28 January 1997, 9 am PST
  - •State of contest: ***finished***
  - •IV: 99 e9 7c bf 4f 7a 6e 8f
  - •Hexadecimal ciphertext:

```
79 45 81 c0 a0 6e 40 a2 0b e3 33 c6 5c 93 b7 22
aa c2 61 27 ff 72 ac c6 03 68 60 2b 43 99 3c 31
9f 36 e6 ae 82 0c 5d 3f a2 3b b4 91 e5 ef ee a3
b9 92 78 d4 af 0f 66 aa f5 17 e8 64 08 88 82 8d
67 ba fe bf 78 95 60 95 31 6a e2 98 78 1c 6a f7
```

分组算法—DES

- 自DES作为联邦标准得以公布之后，美国国家安全局NSA定期对该算法进行评估，以便确定其安全性是否能够满足具体应用的需要。

- 最后一次评估在是1994年，决定1998年12月之后不再使用DES。后来，在AES的征集过程中，通过对世界范围内提交的各种新算法进行评估，已经确定了选用Rijndael算法作为高级加密算法AES。至此，DES在密码学领域中的重要使命宣告结束。

高级加密准则AES

- 1997年4月15日，NIST发起征集AES算法
- 1997年9月12日，在美国联邦登记处公布征集后选算法通知
- 2000年3月，NIST为AES公开征集保密工作模式，后讨论征集到15个，并于2001年秋在文件800-38A中公布了5个：ECB，CBC，CFB，OFB，CTR。目前有可能还要征集其他模式。

高级加密准则AES

工作模式评价指标：

- 安全性
- 性能
- 模式/执行特点

高级加密准则AES

工作模式评价指标：安全性

- 攻击——对模式攻击需要的复杂度
- 可证明安全性——在合理的假设下是否有安全性的证明结果
- 与类似模式安全性的比较
- 随机性——输出的统计特性
- 合理的数学背景

高级加密准则AES

工作模式评价指标：性能

- 计算的有效性
- 空间需求
- 可并行性
- 预处理问题

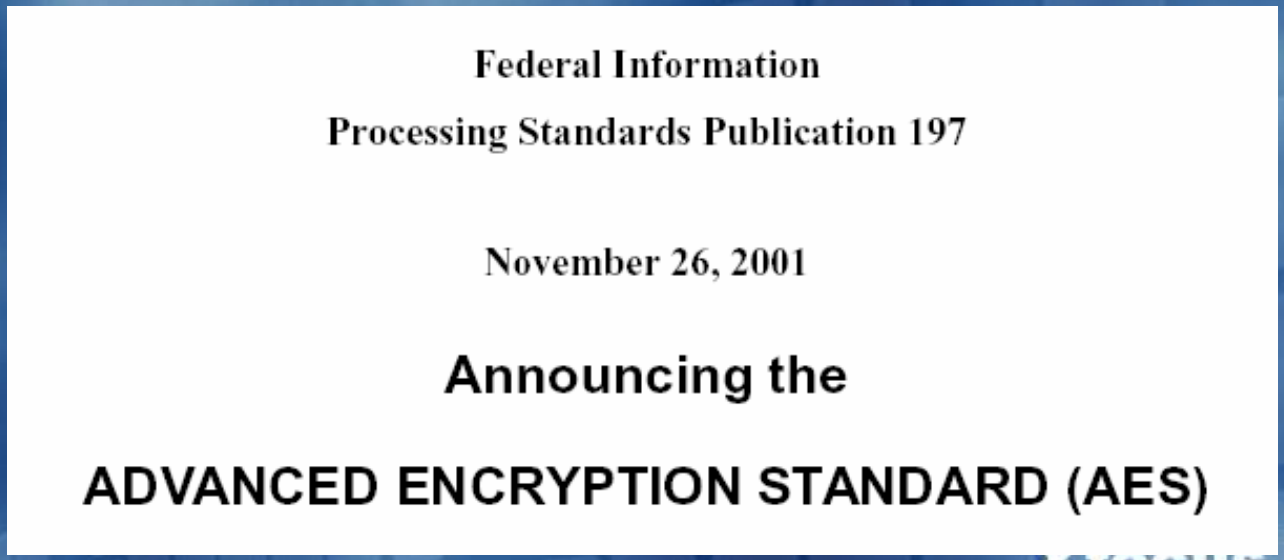
高级加密准则AES

工作模式评价指标：模式/执行特点

- 可提供的密码服务
- 灵活性
- 错误特性
- 模式本身的抗错性
- 简单

高级加密准则AES

- Rijndael 算法：已被美国国家标准技术研究所选定作为高级加密算法AES。



Federal Information

Processing Standards Publication 197

November 26, 2001

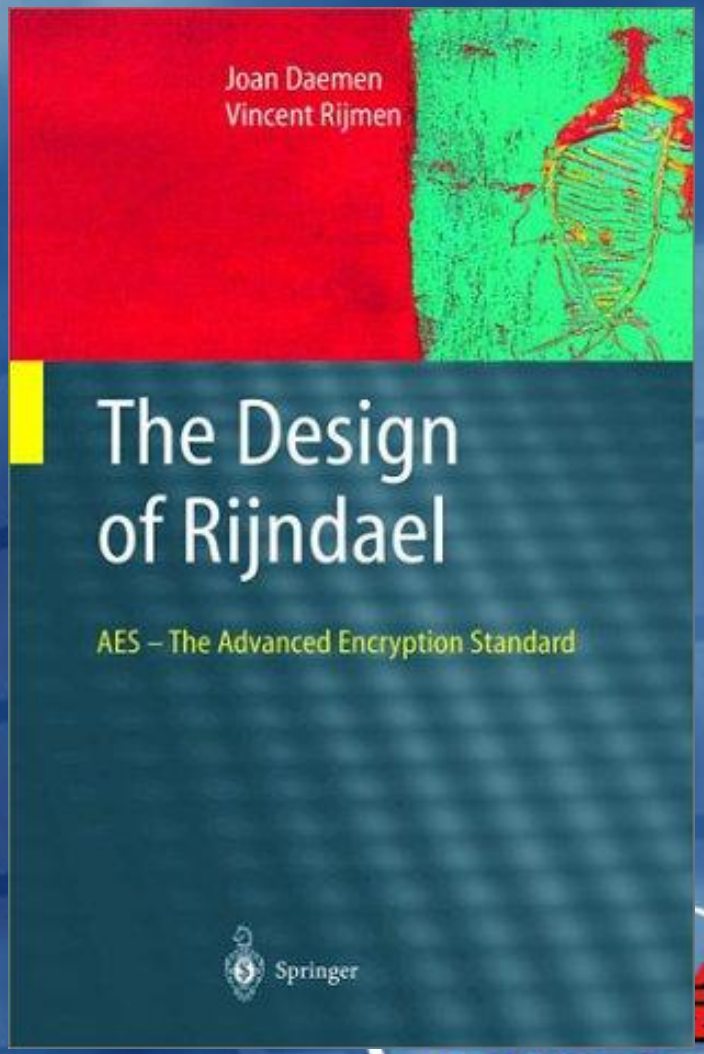
Announcing the

ADVANCED ENCRYPTION STANDARD (AES)

高级加密准则AES

- 在Rijndael算法中，分组长度和密钥长度均可依据实际需要进行选择。

- 提交给AES的Rijndael算法采用的分组长度是128bit，密钥长度为128/192/256bit，对应的轮函数分别为10/12/14轮。



高级加密准则AES

- Rijndael 算法：

由Square算法发展演变而来的一个迭代分组密码算法。

设计思想来源于宽轨迹策略。

由于宽轨迹策略的提出主要是为了使分组密码能够更好地抵抗*差分攻击*和*线性攻击*，

Rijndael算法对于这两种攻击方法具有比较好的抵抗力。

分组算法—RC 5

- 2002年9月26日，一个自发性的协作小组（ distributed.net team ）获得了RSA ***RC 5 - 64*** 私钥挑战赛的胜利。
- 几乎四年时间，
- 331,252名自愿者参与，
- $10,000 奖金。

分组算法—RC 5

- Contest identifier: RC5-32/12/5
- Cipher: RC5-32/12/5 (RC5 with 32-bit wordsize, 12 rounds, and 5*8=40-bit key)
- Start of contest: 28 January 1997, 9 am PST
- State of contest: ***finished***
- IV: 8a 16 2f 69 e8 37 98 bc
- Hexadecimal ciphertext:

```
12 35 13 64 78 d3 da 08 d9 15 ed 20 89 30 5f 50
68 5c 6c b4 bf 5b 00 38 ff 44 4e d9 d4 9b 46 16
fb 12 92 62 9b d4 7f 1a 8d 48 fe b6 63 d1 d4 c2
eb 19 0d 86 3f f4 43 75 9a 58 06 2c 8b a5 9e 6a
33 30 c3 3e a8 ab 24 25
```

分组算法—RC 5

- Contest identifier: RC5-32/12/6
- Cipher: RC5-32/12/6 (RC5 with 32-bit wordsize, 12 rounds, and 6*8=48-bit key)
- Start of contest: 28 January 1997, 9 am PST
- State of contest: ***finished***
- IV: cf 4c df 7c 08 36 ea cc
- Hexadecimal ciphertext:

```
46 de b3 1b 18 5d 4b e6 68 ef 9e 4c 72 b3 a1 44
06 fa fb f4 07 11 25 6b 41 e9 b5 8f 1a 32 b8 5a
d5 1c b6 37 e7 61 02 e9 5b 00 66 33 1c e7 64 61
12 07 d3 b6 00 56 81 8c 42 f0 51 a2 f6 45 7e d5
f6 13 2a e1 eb fd 8a 90
```

分组算法—RC 5

- Contest identifier: RC5-32/12/7
- Cipher: RC5-32/12/7 (RC5 with 32-bit wordsize, 12 rounds, and 7*8=56-bit key)
- Start of contest: 28 January 1997, 9 am PST
- State of contest: ***finished***
- IV: 7b 32 f0 8a e6 17 de 8c
- Hexadecimal ciphertext:

```
82 d3 4e a7 b3 24 86 0b c6 d8 61 5c e9 f9 e4 79
88 5c 98 f1 d2 92 4c 59 ee 47 51 31 01 3e a8 ab
d6 f0 4d c8 19 97 af 01 5e af f8 3f cd 61 b3 c2
66 89 7c 82 09 87 4d fb 07 f2 56 03 8d d5 1b 01
ca e3 41 c2 8d d7 18 1d
```

分组算法—RC 5

- Contest identifier: RC5-32/12/8
- Cipher: RC5-32/12/8 (RC5 with 32-bit wordsize, 12 rounds, and 8*8=64-bit key)
- Start of contest: 28 January 1997, 9 am PST
- State of contest: ***finished***
- IV: 79 ce d5 d5 50 75 ea fc
- Hexadecimal ciphertext:

```
bf 55 01 55 dc 26 f2 4b 26 e4 85 4d f9 0a d6 79
66 93 ab 92 3c 72 f1 37 c8 b7 0d 1f 60 11 0c 92
ae 2e cd fd 70 d3 fd 17 df b0 42 12 b9 7d cf 22
18 6b a7 15 ce 2c 84 bf ce 0d d0 4d 00 6b e1 46
```

分组算法—RC 5

- Contest identifier: RC5-32/12/9
- Cipher: RC5-32/12/9 (RC5 with 32-bit wordsize, 12 rounds, and 9*8=72-bit key)
- Start of contest: 28 January 1997, 9 am PST
- State of contest: ***ongoing***
- IV: 41 d5 97 4c 00 7a f5 f6
- Hexadecimal ciphertext:

```
e7 af fc be 5f 74 ec a6 11 a2 1f a8 8a 0a a1 76
dd 8e 01 d3 2b 31 a8 df 60 26 4f b0 16 ed 2c 71
89 fb 01 db ac 0f af b5 21 a8 d6 5e a6 0f 54 48
14 f5 06 1e 1f 21 8b 36
```

分组算法—RC 5

Ongoing

- Contest identifier: RC5-32/12/10
- Contest identifier: RC5-32/12/11
- Contest identifier: RC5-32/12/12
- Contest identifier: RC5-32/12/13
- Contest identifier: RC5-32/12/14
- Contest identifier: RC5-32/12/15
- Contest identifier: RC5-32/12/16

分组算法—RC 6

- RSA提交
- 在RC 5 基础上设计，大量使用数据依赖循环
- 分组长度128 bits
- 采用了4个移位寄存器
- 加进32 bits整数乘法（用于加强扩散特性）
- RC 6 – w / r / b

3. 公钥密码

公钥密码

- 公钥密码的最大优点：

  针对密钥管理方法的改进。

- 公钥密码算法是公钥密码体制的核心。这些算法基于不同的计算问题。
- *公钥密码体制的这种安全性理论基础只是基于复杂性理论的一种计算安全性，而非绝对的安全性。*

公钥密码

- Diffie 和Hellman在其研究中指出：

  *计算复杂性*可以被用作设计加密算法的基础，而*NP-完全问题*则是一个理想的解决途径。

  此外，由于令人满意的加密与解密过程都必须尽可能地迅速，所以更难的问题并不适合用于设计加密算法。

  这种计算安全性给密码分析者提供了破解公钥加密算法的可能性，使密码分析人员能够通过在多项式时间内猜测并检验某个密钥的方法完成破译工作。

  并非所有的NP-完全问题都可用于密码领域。

  已经找到的若干可用于密码的这类问题包括大素数分解问题和背包问题。在没有解密密钥的情况下，进行破译必须解决这些NP-完全问题。

公钥密码

- 除著名的RSA算法外，迄今为止已经研究出建立在不同计算问题之上的其它公钥密码算法有：

- 基于“子集和”难题的Merkle-Hellman Knapsack公钥密码算法（产生于1978年）。通过证明我们已经知道：除Chor-Rivest算法外的各种knapsack算法是不安全的。
- 基于代数编码系统的McEliece公钥密码算法（产生于1978年）。
- 基于有限域中离散对数难题的ElGamal公钥密码算法（产生于1985年）。
- 目前被认为安全的Knapsack型公钥密码算法Chor-Rivest（产生于1988年）。
- 基于因子分解问题的Rabin算法。
- 椭圆曲线公钥算法。

公钥算法—RSA

- RSA算法的安全性建立在大整数分解难题之上。算法所用的公钥和私钥是一对足够大的奇素数的函数。由公钥和密文恢复出明文的难度与分解两个足够大的奇素数的乘积具有同等的难度。

公钥算法—RSA

- 1999年2月2日， **Factorization of RSA-140**
- 125 SGI and Sun workstations (平均 175 MHz) and on about 60 PCs （平均 300 MHz ）.
- 8.9 CPU 年.
- 两种筛选软件：
- line sieving ： 要求 26 Mbytes/machine
- lattice sieving： 要求 about 50 Mbytes/machine

公钥算法—RSA

- 1999年2月2日，**Factorization of RSA-140**（续）

- 36.8 %　Peter L. Montgomery, Stefania Cavallar, Herman J.J. te Riele, Walter M. Lioen (C,L at CWI, Amsterdam, The Netherlands)
- 28.8 %　Paul C Leyland (L at Microsoft Research Ltd, Cambridge, UK)
- 26.6 %　Bruce Dodson (C,L at Lehigh University, Bethlehem, PA, USA)
- 5.4 %　Paul Zimmermann (L at Inria Lorraine and Loria, Nancy, France)
- 2.5 % Arjen K. Lenstra (L at Citibank, Parsippany, NJ, USA, and at the University of Sydney, Australia)

公钥算法—RSA

- 1999年8月22日， **Factorization of RSA-155**
- 数域筛法 （line sieving ， lattice sieving）
-
- Sieving: 35.7 CPU-years in total on...
- 160 175-400 MHz SGI and Sun workstations
- 8 250 MHz SGI Origin 2000 processors
- 120 300-450 MHz Pentium II PCs
- 4 500 MHz Digital/Compaq boxes

- CPU-effort： 8000 MIPS 年左右;
- 时间： 3.7 months.

公钥算法—RSA

- 1999年8月22日，**Factorization of RSA-155**（续）
- 124 722 179 relations were collected by eleven different sites, distributed as follows:
  (L: using lattice sieving code from Arjen K. Lenstra C: using line sieving code from CWI)
  - 20.1 % (3057 CPU days) Alec Muffett (L)
  - 17.5 % (2092) Paul Leyland (L,C)
  - 14.6 % (1819) Peter L. Montgomery, Stefania Cavallar (C,L)
  - 13.6 % (2222) Bruce Dodson (L,C)
  - 13.0 % (1801) Francois Morain and Gerard Guillerm (L,C)
  - 6.4 % (576) Joel Marchand (L,C)
  - 5.0 % (737) Arjen K. Lenstra (L)
  - 4.5 % (252) Paul Zimmermann (C)
  - 4.0 % (366) Jeff Gilchrist (L)
  - 0.65 % (62) Karen Aardal (L)
  - 0.56 % (47) Chris and Craig Putnam (L)

公钥算法—RSA

RSA-576	$10,000	Not Factored
RSA-640	$20,000	Not Factored
RSA-704	$30,000	Not Factored
RSA-768	$50,000	Not Factored
RSA-896	$75,000	Not Factored
RSA-1024	$100,000	Not Factored
RSA-1536	$150,000	Not Factored
RSA-2048	$200,000	Not Factored

密钥托管与恢复

- 美国政府托管加密标准（Escrowed Encryption Standard,EES）的核心。
- EES由防窜改的硬件集成块实现。每芯片有唯一序列号ID和特定密钥KIO、KID分成两部分由不同托管机构存储。安全通信时会话密钥用特定密钥加密随ID号发出，从而导致政府监听侦测并恢复其明文
- Clipper、Capstone、Skipjack

密钥恢复技术

- 用来加密的密钥保存在某一可信数据库中，如政府密钥托管
- IKE与密钥恢复技术
  - IKE密钥交换中可通过设置密钥恢复中心来长期保存密钥，但此提案被IETF否定。
  - RFC1984（IAB，Internet结构会议）中IETF明确反对使用密钥恢复技术。

Skipjack

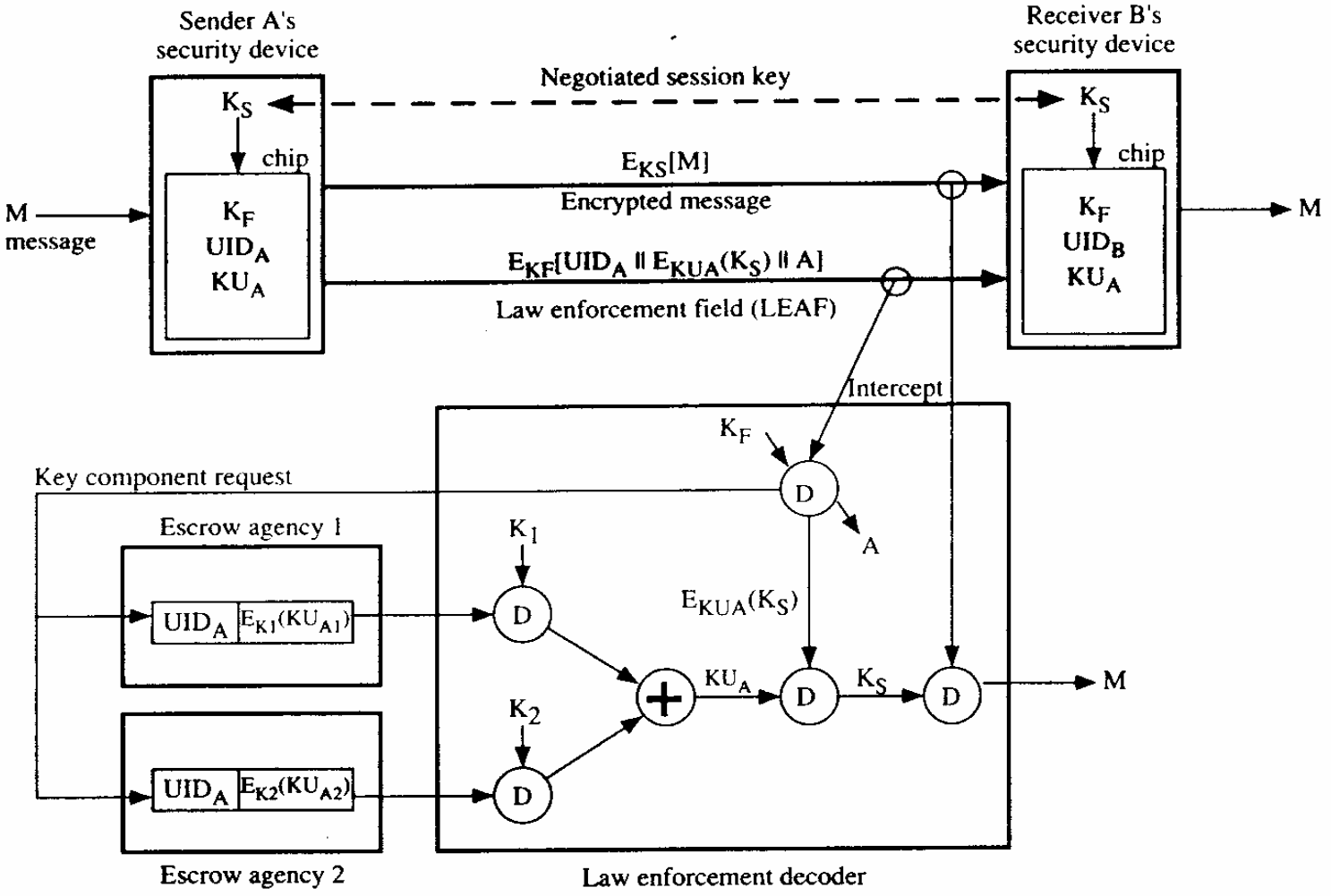


FIGURE 7.15. Secure Communications with Law Enforcement Access

Definition of the parameters:

KF = 80-bit family key.

UID = an identifier unique to a particular SKIPJAC chip.

KU = device-unique key. KU= KU1⊕ KU2

KU1, KU2 = a pair of key components.

K1, K2 = secret key-encrypting keys

Ks = session key

P = authentication code.

The LEAF that is transmitted as part of a communication has the following general format:

EKF(UIDA || EKUA(KS) || PA)

When both encrypted key components are returned, the agency can perform the following:

1. Recover key components:

   $KU_{A1} = D_{K1}[E_{K1}(KU_{A1})]$; $KU_{A2} = D_{K2}[E_{K2}(KU_{A2})]$

   where K_1 and K_2 are secret keys known only to the requesting agency.

2. Generate device-unique key:

   $KU_A = KU_{A1} \oplus KU_{A2}$

3. Recover session key:

   $K_s = D_{KU_A}[E_{KU_A}(K_s)]$

4. Recover message:

   $M = D_{Ks}[E_{ks}(M)]$

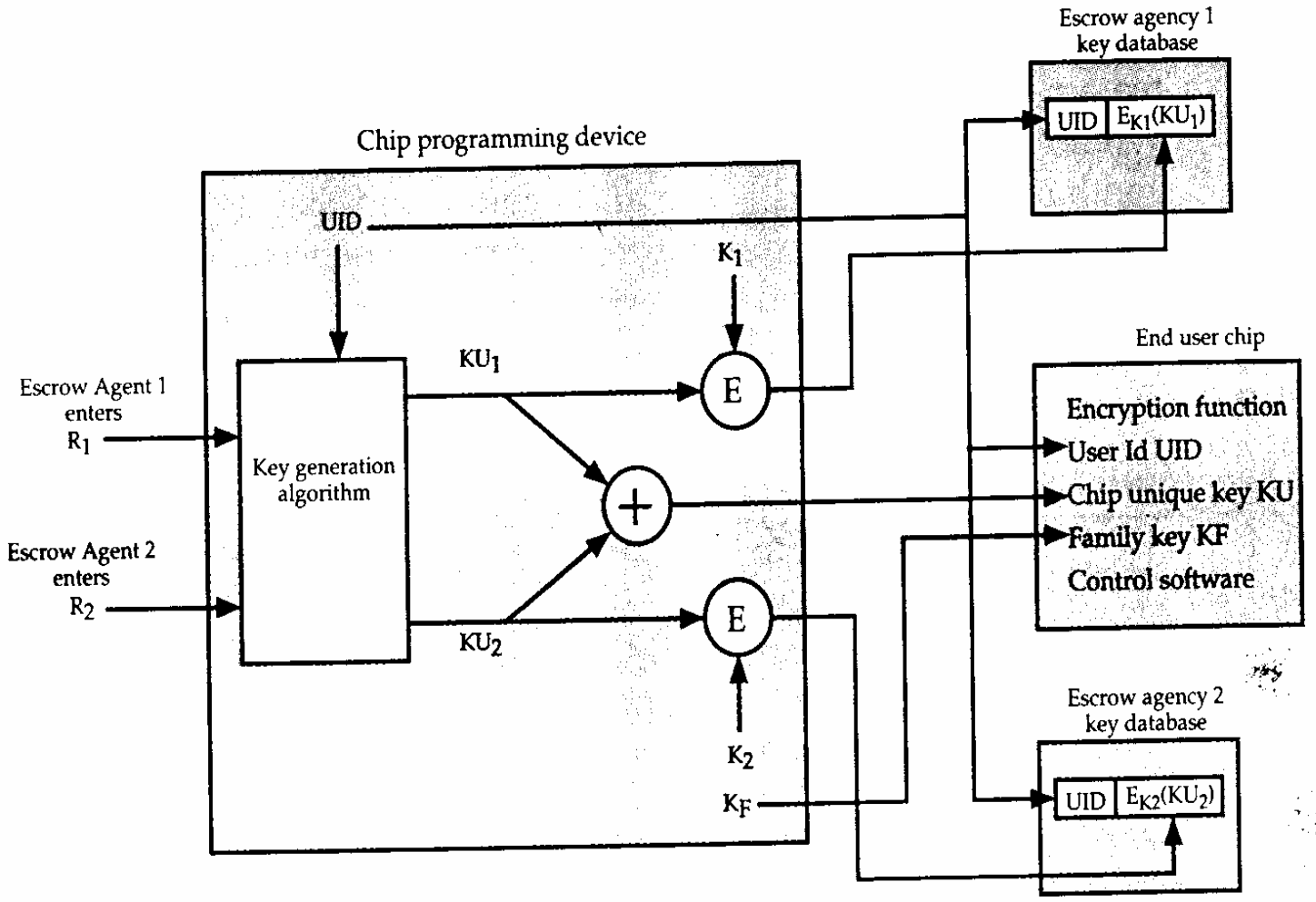


FIGURE 7.16. Chip Programming and Key Escrow

谢谢大家！