第3讲 密码学基础

中国科学技术大学
曾凡平
billzeng@ustc.edu.cn

主要内容

1. 密码学概述
2. DES加密算法
3. RSA加密算法
4. 消息摘要和数字签名：实现完整性和抗抵赖的方法
5. PGP (Pretty Good Privacy)：
   - 日常工作最常用的加密工具，使用PGP产生密钥，加密文件和邮件。
6. 使用OpenSSL中的密码函数
7. Windows系统提供的密码算法
8. python语言的加密模块

第3讲 密码学基础

- 信息安全的主要目标是保护信息的**机密性、完整性和可用性**。机密性主要通过密码技术实现，而信息的完整性也直接或间接地使用了密码的相关技术，因此密码学是信息安全的基础。

- 长期以来，密码技术只在很小的范围内使用，如军事、外交、情报等部门。随着人类社会向信息社会的演进，**基于计算复杂性的计算机密码学**得到了前所未有的重视并迅速普及和发展起来。

- 在国外，密码学已成为计算机网络安全领域的主要研究方向之一。

3.1 密码学概述

- 密码学是研究如何**隐密地传递信息**的学科，其**首要目的是隐藏信息的涵义**。密码学涉及信息的加密/解密及密码技术在信息传递过程中的应用。

- 早期的密码技术的安全性基于密码算法的保密，现代的密码技术要求**密码算法公开、密钥必须保密**，**密码算法的强度基于计算的复杂性**。

- 著名的密码学者Ron Rivest（RSA密码算法的发明者之一）对密码学的解释是：**“密码学是关于如何在敌人存在的环境中通讯”**。

密码学的发展

- 密码技术的历史比较悠久，在四千年前，古埃及人就开始使用密码来保密传递消息。
- 两千多年前，恺撒就开始使用目前称为“恺撒密码”的密码系统。但是密码技术直到20世纪40年代以后才有重大突破和发展。
- 特别是20世纪70年代后期，由于计算机、电子通信的广泛使用，现代密码学得到了空前的发展。

密码学的相关学科

- 密码学相关学科大致可以分为三个方面：
  1. **密码学(Cryptology)**是研究信息系统安全保密的科学；(**密码(cipher)**是指逐个字符或者逐位地进行变换，它不涉及信息的语言结构)

  2. **密码编码学(Cryptography)**主要研究对信息进行编码，实现对信息的隐藏；(**编码(code)**则是指用一个词或符号来代替另一个词)

  3. **密码分析学(Cryptanalytics)**主要研究加密消息的破译或消息的伪造。

基于密码学的保密通信系统的模型

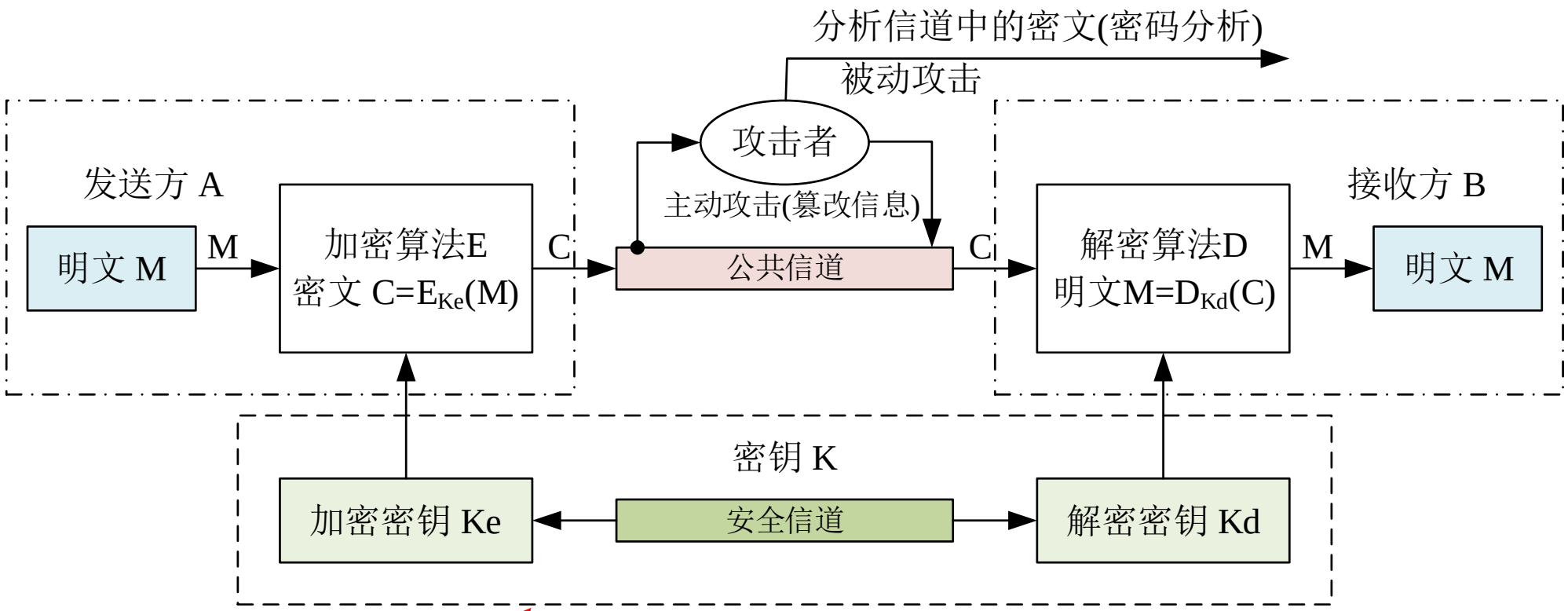


图3-1 保密通信系统的模型

目前的大部分**量子保密通信**技术：主要是通过量子技术实现**安全的密钥分发**

消息和加/解密

- 遵循国际命名标准，加密和解密可以翻译成：“Encipher（译成密码）”和“Decipher (解译密码)”。也可以这样命名：“Encrypt (加密)”和“Decrypt (解密)”。

→消息被称为明文

→用某种方法伪装消息以隐藏它的内容的过程称为加密

→加了密的消息称为密文

→把密文转变为原始明文的过程称为解密

明文、密文

- **明文**用**M** (Message，消息) 或**P** (Plaintext，明文)表示，它可能是比特流、文本文件、位图、数字化的语音流或者数字化的视频图像等。
- **密文**用**C** (Cipher) 表示，也是二进制数据，有时和M一样大，有时稍大。通过压缩和加密的结合，C有可能比P小些。

- 加密函数E作用于M得到密文C，用数学公式表示为：

$$E(M)=C$$

- 解密函数D作用于C产生M，用数学公式表示为：

$$D(C)=M$$

- 先加密、再解密，**原始的明文**将恢复出来，下式必须成立：

$$D(\ E(M)\ )=M$$

鉴别、完整性和抗抵赖性

- 除了提供机密性外，密码学需要提供三方面的功能：鉴别、完整性和抗抵赖性。
- **鉴别：**消息的接收者应该能够确认消息的来源；入侵者不可能伪装成他人。
- **完整性：**消息的接收者应该能够验证在传送过程中的消息没有被修改；入侵者不可能用假消息代替合法消息。
- **抗抵赖性：**发送消息者事后不可能虚假地否认他发送的消息。

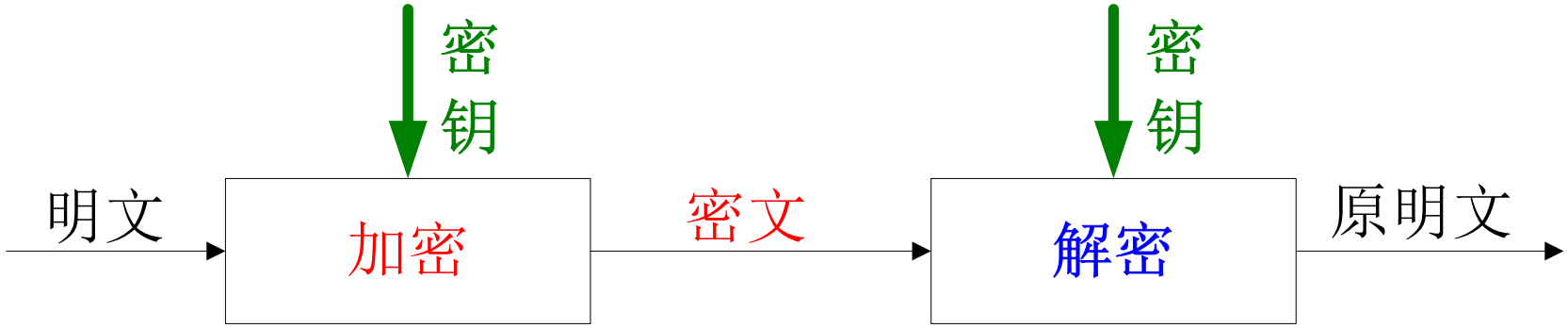
算法和密钥

- ***现代密码学要求密码算法公开，密钥保密。***
- 密钥用K表示。K可以是很多数值里的任意值，密钥K的可能值的范围叫做**密钥空间**。对称密码算法加密和解密运算都使用这个密钥，即运算都依赖于密钥，并用K作为下标表示，加/解密函数表达为：

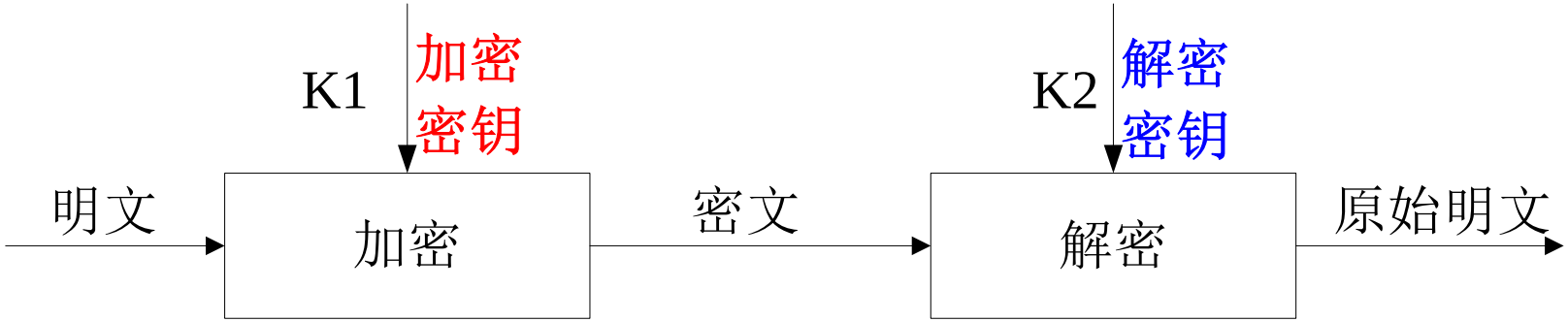
$$E_K(M)=C$$

$$D_K(C)=M$$

$$D_K(E_K(M))=M$$



不同的加密密钥和解密密钥



- 有些算法使用不同的加密密钥和解密密钥，也就是说**加密密钥K1**与相应的**解密密钥K2**不同，在这种情况下，加密和解密的函数表达式为：

  $\mathbf{E_{K1}(M)=C}$

  $\mathbf{D_{K2}(C)=M}$

- 函数必须具有的特性是：$D_{K2}(\mathbf{E_{K1}(M)})=M$

对称密码算法的基本原理

- 基于密钥的算法通常有两类：对称密码算法和公开密钥算法（非对称算法）。
- 对称密码算法有时又叫传统密码算法或单密钥密码算法，**加密密钥能够从解密密钥中推算出来**，反过来也成立。

- 在大多数对称算法中，加/解密的密钥是相同的。对称算法要求发送者和接收者在安全通信之前，协商一个密钥。**对称算法的安全性依赖于密钥的保密**，泄漏密钥就意味着任何人都能对消息进行加/解密。对称算法的加密和解密表示为：

  $E_K(M) = C$

  $D_K(C) = M$

对称密码算法的两个分支

- ***分组密码算法：*** 对明文的一组位进行加密和解密运算，这些位组称为分组，相应的算法称为分组算法。常见的分组算法有DES、DES3、IDEA、AES等。分组密码算法也称为块密码算法。

- DES分组长度为64位、密钥长度为64位；AES加密数据块分组长度为128比特，密钥长度可以是128比特、192比特、256比特中的任意一个（如果数据块及密钥长度不足时，会补齐）。

- ***序列密码（流密码）算法：*** 一次只对明文的单个位（有时对字节）运算的算法称为序列密码算法或流密码。常见的流密码有RC4、A5、SEAL、PIKE等。

公开密钥算法的基本原理

- **公开密钥算法（非对称算法）的加密密钥和解密密钥不同**，而且解密密钥不能根据加密密钥计算出来，或者至少在可以计算的时间内不能计算出来。

- 之所以叫做公开密钥算法，是因为加密密钥能够公开，即陌生者能用加密密钥加密信息，但只有用相应的解密密钥才能解密信息。**加密密钥叫做公开密钥（简称公钥），解密密钥叫做私人密钥（简称私钥）**。

- 用公开密钥K1加密表示为：$E_{K1}(M)=C$
- 用相应的私人密钥K2解密可表示为：$D_{K2}(C)=M$

安全协议

- **密码协议**：也称作**安全协议**，是使用密码学的协议，是以密码学为基础的消息交换协议，其目的是在网络环境中提供各种安全服务。
- 密码学是网络安全的基础，但网络安全不能单纯依靠安全的密码算法。安全协议是网络安全的一个重要组成部分，我们需要通过安全协议进行实体之间的认证、在实体之间安全地分配密钥或其它各种秘密、确认发送和接收的消息的不可否认性等。
- 常见的安全协议有：认证协议、不可否认协议、公平性协议、身份识别协议、密钥管理协议。
- **密码编码原则**：密码算法应建立在**算法的公开不影响明文和密钥的安全**。

3.2 对称密码技术

- 对称密码体制的加密密钥和解密密钥是相同的，其中最负盛名的是曾经广泛使用的DES和正在推广使用的AES。与公开密钥密码技术相比，其最大的优势就是**速度快**，一般用于对大量数据的加/解密。

- 数据加密标准（DES，Data Encryption Standard）是一种使用密钥加密的块密码，1976年被美国联邦政府的国家标准局确定为**联邦资料处理标准（FIPS）**，随后在国际上广泛流传开来。它基于使用56位密钥的对称算法。

3.2.1 DES算法的安全性

- DES的56位密钥过短，现在已经不是一种安全的加密方法。早在1999年1月，distributed.net与电子前哨基金会合作，就在22小时15分钟内破解了一个DES密钥。

- 随着计算机的升级换代，运算速度大幅度提高，破解DES密钥所需的时间也将越来越短。为了保证实用应用所需的安全性，可以使用DES的派生算法3DES来进行加密。3DES被认为是十分安全的，虽然它的速度较慢。另一个计算代价较小的替代算法是DES-X，它通过将数据在DES加密前后分别与额外的密钥信息进行异或来增加密钥长度。GDES则是一种速度较快的DES变体，但它对微分密码分析较敏感。

高级加密标准（AES）

- 2000年10月，在历时接近5年的征集和选拔之后，NIST选择了一种新的密码，即**高级加密标准（AES）**，用于替代DES。2001年2月28日，联邦公报发表了AES标准，从此开始了其标准化进程，并于2001年11月26日成为FIPS PUB 197标准。

- AES算法在提交的时候称为Rijndael。选拔中其它进入决赛的算法包括RC6、Serpent、MARS和Twofish。

3.2.2 DES算法的原理

- DES是一种典型的块密码，一种将固定长度的明文通过一系列复杂的操作变成同样长度的密文的算法。
- 对DES而言，块长度为64位。同时，DES使用密钥来自定义变换过程，因此只有持有加密密钥的用户才能解密密文。

- DES的密钥表面上是64位，然而只有其中的56位被实际用于算法，其余8位可以被用于奇偶校验，并在算法中被丢弃。因此，DES的有效密钥长度为56位。

- DES算法的整体结构如图3-5所示：

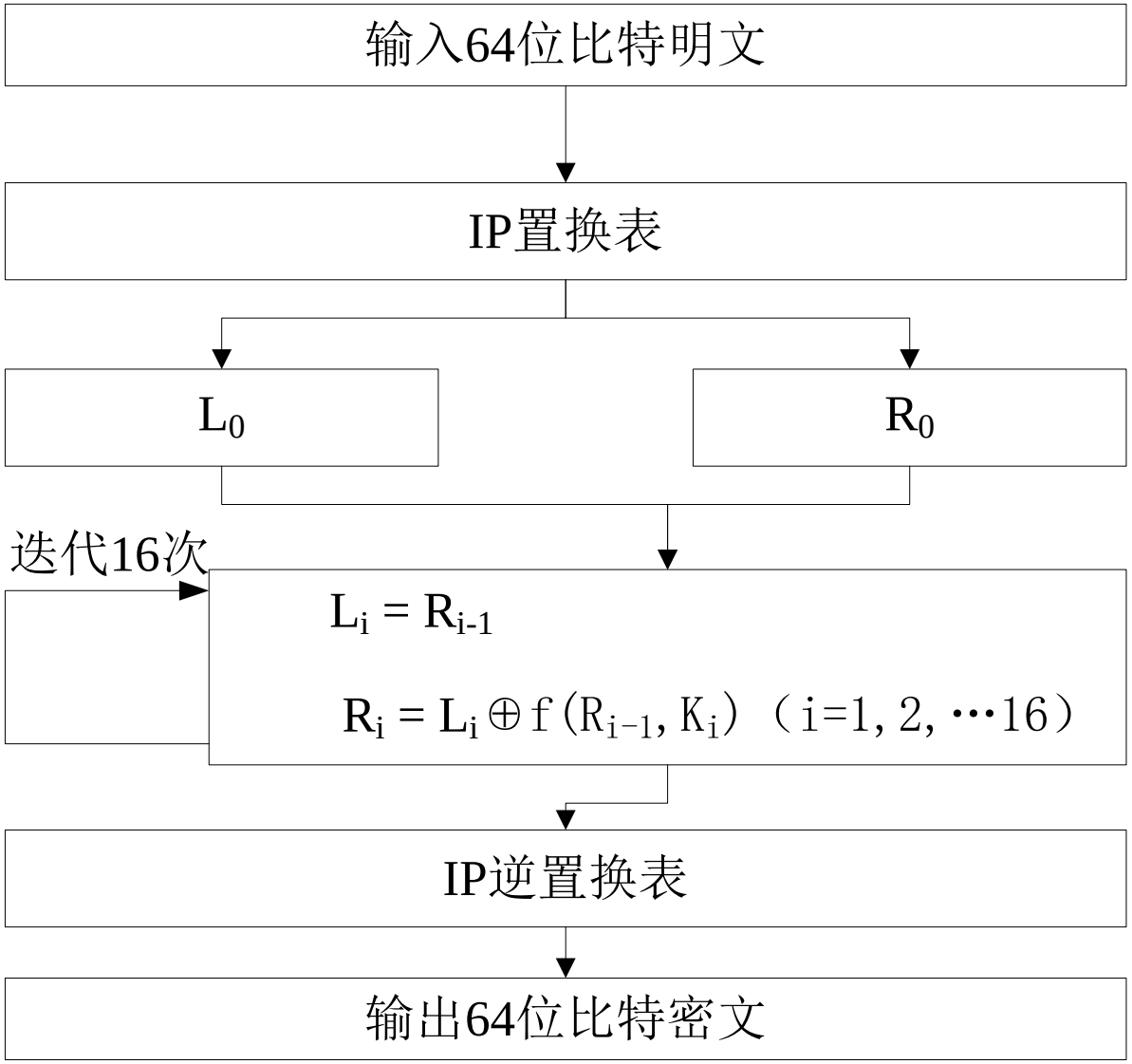


图3- 5 DES算法的整体结构

DES算法实现加密需要三个步骤:

- **第一步：变换(置换)明文**。
  - 对给定的64位比特的明文x，首先通过一个置换IP表来重新排列x，从而构造出64位比特的x_0，$x_0=IP(x)=L_0R_0$，其中L_0表示x_0的前32比特，R_0表示x_0的后32位。
- **第二步：按照规则进行16轮迭代**。规则为

  $L_i = R_{i-1}$

  $R_i = L_i \oplus f(R_{i-1},K_i)$ （i=1,2,3…16）

  经过第一步变换已经得到L_0和R_0的值，其中符号⊕表示的数学运算是异或，f表示一种置换，由S盒置换构成，K_i是一些由密钥编排函数产生的比特块。f和K_i将在后面介绍。
- **第三步：**
  - 对$L_{16}R_{16}$利用IP^{-1}作**逆置换**，就得到了密文y。

- DES算法的详细内容请参考密码学方面的专著，其具体实现的源代码请参考OpenSSL源代码:

  http://www.openssl.org https://github.com/openssl/openssl

3.2.3 DES的各种变种

- 由于DES的密钥长度仅为56比特，破解密文需要2^{56}次穷举搜索，在目前已难于保证密文的安全。
- 为了解决DES密钥长度过短的问题，可以采用组合密码技术，也就是将密码算法组合起来使用。三重DES(简写为DES3或3DES)是最常用的组合密码技术，破解密文需要2^{112}次穷举搜索，其算法如图3-6所示。

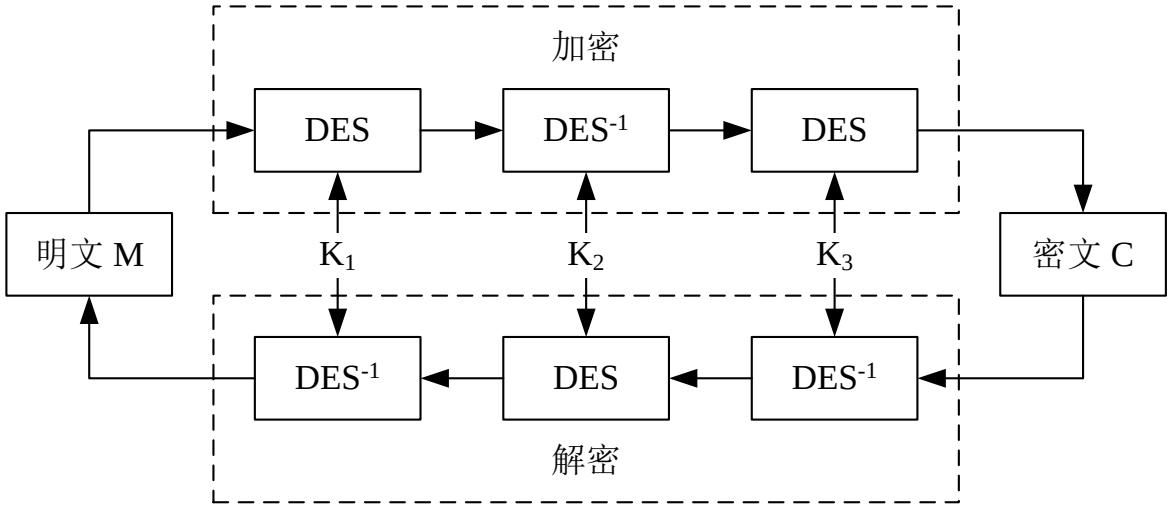


图3-6 三重DES

基于DES的其它算法还有DESX、CRYPT(3)、GDES、RDES、更换S盒的DES、使用相关密钥S盒的DES等。

3.3 RSA公钥加密技术

- 公开密钥加密（public-key cryptography）也称为**非对称(密钥)加密**，该思想最早由Ralph C. Merkle在1974年提出。之后在1976年，Whitfield Diffie（迪菲）与Martin Hellman（赫尔曼）两位学者在现代密码学的奠基论文“New Direction in Cryptography”中首次公开提出了公钥密码体制的概念。

- 公钥密码体制中的密钥分为**加密密钥**与**解密密钥**，这**两个密钥是数学相关的**，用加密密钥加密后所得的信息，只能用该用户的解密密钥才能解密。如果知道了其中一个，并不能计算出另外一个。因此如果公开了一对密钥中的一个，并不会危害到另外一个的秘密性质。公开的密钥称为**公钥（PK）**，不公开的密钥称为**私钥（SK）**。

常见的公钥加密算法

- 常见的公钥加密算法有RSA、ElGamal、背包算法、Rabin（RSA的特例）、Diffie－Hellman密钥交换协议中的公钥加密算法、椭圆曲线加密算法（Elliptic Curve Cryptography, ECC）。使用最广泛的是RSA算法（由发明者Rivest、Shmir和Adleman姓氏首字母缩写而来），ElGamal是另一种常用的非对称加密算法。

- RSA和ElGamal是同时很好地用于加密和数字签名的公开密钥算法。此类算法要求**加密、解密的顺序可以交换**，即满足以下公式：

$$\mathbf{E_{PK}(\,D_{SK}(M)\,)} = D_{SK}(\,E_{PK}(M)\,) = M$$

RSA算法

- RSA算法于1977年由Rivest、Shamir和Adleman(当时他们三人都在麻省理工学院工作)发明，是第一个既能用于数据加密也能用于数字签名的算法。RSA算法易于理解和操作，虽然其安全性一直未能得到理论上的证明，但是它经历了各种攻击，至今未被完全攻破，所以，实际上是安全的。

- 1973年，在英国政府通讯总部工作的数学家克利福德·柯克斯（Clifford Cocks）在一个内部文件中提出了一个与RSA相似的算法，但他的发现被列入机密，一直到1997年才被发表。

RSA算法

- 对极大整数做因数分解的难度决定了RSA算法的可靠性。换言之，对一极大整数做因数分解愈困难，RSA算法就愈可靠。如果有人找到了一种快速因数分解的算法，那么用RSA加密的信息的可靠性就肯定会极度下降，但找到这样的算法的可能性是非常小的，目前只有短的RSA密钥才可能被强力方式解破。到2013年为止，世界上还没有任何可靠的攻击RSA算法的方式。**只要其密钥的长度足够长，用RSA加密的信息实际上是不能被解破的**。

- 1983年麻省理工学院在美国为RSA算法申请了专利。这个专利2000年9月21日失效。由于该算法在申请专利前就已经被发表了，在世界上大多数其它地区这个专利权不被承认。

3.3.1 RSA算法描述

- **密钥计算**方法：

  选择两个大素数p和q (典型值为1024位)

  计算 $n=p\times q$ 和 $z=(p-1)\times(q-1)$

  选择一个与 z 互质的数，令其为 d

  找到一个 e 使满足 $e\times d=1 \pmod z$

  公开密钥为 (e, n)，**私有密钥为 (d, n)**

- **加密**方法：

  将明文看成比特串，将明文划分成k位的块P即可，这里k是满足$2^k<n$的最大整数。

  对每个数据块P，计算$C = P^e \pmod n$，C即为P的密文。

- **解密**方法：

  对每个密文块C，计算$P = C^d \pmod n$，P即为明文。

3.3.2 RSA算法举例

- 密钥计算：

  取$p = 3$，$q = 11$

  则有$n = p\times q=33$，$z = (p-1)\times(q-1) =(3-1) \times(11-1)=20$

  7和20没有公因子，可取$d = 7$

  解方程$7\times e = 1(\mathrm{mod}\ 20)$，得到$e = 3$

  公钥为（3, 33），私钥为（7, 33）

- 加密：

  若明文$P = 4$，则密文$C = P^e\ (\mathrm{mod}\ n) = 4^3\ (\mathrm{mod}\ 33) = 31$

- 解密：

  计算$P = C^d\ (\mathrm{mod}\ n) = 31^7(\mathrm{mod}\ 33) = 4$，恢复出原文

3.3.3 RSA算法的安全性

- 假设偷听者乙获得了甲的公钥(e,n)以及丙的加密消息C，但她无法直接获得甲的私人密钥d。

- 要获得d，最简单的方法是将n分解为p和q，这样她可以得到同余方程$d \times e \equiv 1 \pmod{(p-1)(q-1)}$并解出d，然后代入解密公式：

$$P = C^d \pmod{n}$$

- 这样就破解了密文C，导出了明文P。

RSA算法的安全性

- 但**至今为止还没有人找到一个多项式时间的算法来分解一个大的整数的因子，同时也还没有人能够证明这种算法不存在。**至今为止也没有人能够证明对n进行因数分解是唯一的从C导出P的方法，但今天还没有找到比它更简单的方法（至少没有公开的方法）。因此今天一般认为只要n足够大，那么攻击者就没有办法了。

- 目前，假如n的长度小于或等于256位，那么用一台个人电脑在几个小时内就可以分解它的因子。1999年，数百台电脑合作分解了一个512位长的n。2009年12月12日，编号为 RSA-768 （768 bits, 232 digits）数也被成功分解。这一事件威胁了现流行的1024 bit密钥的安全性，普遍认为用户应尽快升级到2048 bit或以上。

3.3.4 RSA算法的速度

- 比起DES和其它对称算法来说，RSA要慢得多。速度慢一直是RSA的缺陷，一般来说只用于少量数据加密。事实上RSA一般用于数字签名和对工作密钥的加密，对数据的加密一般采用速度更快的对称密码算法。

- RSA是被研究得最广泛的公钥算法，从提出到现在已经过了几十年，经历了各种攻击的考验，逐渐被人们接受，普遍认为是目前最优秀的公钥方案之一。

3.3.5 RSA算法的程序实现

- 根据RSA算法的原理，可以利用C语言实现其加密和解密算法。RSA算法比DES算法复杂，加/解密所需要的时间也比较长。

- 具体实现见OpenSSL的源代码 (http://www.openssl.org)

演 示

3.4 消息摘要和数字签名

- 使用高强度的密码技术可以保证数据的机密性。
- 然而，密码算法的运行速度较慢，如果数据的价值（比如卫星拍摄的视频或图像，声音等大数据）不值得用密码技术对其进行保护，而只需保证其完整性时，人们迫切需要一种技术能实现**高速的完整性鉴别**。同时，为了**防止发送信息的一方否认曾经发送信息**，也需要一种技术来鉴别信息确实发送自某个密钥持有者。

- 消息摘要和数字签名可以满足这两方面的需求。

3.4.1 报文摘要(消息摘要)

- 消息摘要的目的是将消息鉴别与数据保密分开，其基本设想是：发送者用明文发送消息，并在消息后面附上一个标签，允许接收者利用这个标签来鉴别消息的真伪。

- 用于鉴别消息的标签必须满足以下两个条件：
  - 第一，能够验证消息的完整性，即能辨别消息是否被修改；
  - 第二，标签不可能被伪造。

消息摘要

- 为了辨别消息是否被修改，可以将一个散列函数作用到一个任意长的消息m上，生成一个固定长度的**散列值H(m)**，这个散列值称为该消息的数字指纹，也称**消息摘要（message digest，MD)**。消息的发送者对发送的消息计算一个消息摘要M1，和消息一起发给接收者；接收者对收到的消息也计算一个消息摘要M2，如果M2等于M1，则验证了消息的完整性，否则就证明了消息被篡改了。

- 为了保证标签不可能被伪造，发送方可以用密码技术对消息摘要M1进行加密保护，得到加密后的消息摘要C，接收方对C进行解密恢复M1，再与消息摘要M2比较，从而判断消息的完整性。**加密后的消息摘要**也称为**消息鉴别标签**。

- 用于消息鉴别的散列函数H必须满足以下特性：

  (1)H能够作用于任意长度的数据块，并生成固定长度的输出。

  (2)对于任意给定的数据块x，H(x)很容易计算。

  (3)对于任意给定的值h，要找到一个x满足H(x)=h，在计算上是不可能的(单向性)。(这一点对使用加密散列函数的消息鉴别很重要)

  (4)对于任意给定的数据块x，要找到一个y≠x并满足H(y) = H(x)，在计算上是不可能的。(这一点对使用加密算法计算消息鉴别标签的方法很重要)

  (5)要找到一对（x，y）满足H(y) = H(x)，在计算上是不可能的。(抵抗生日攻击)

MD5和SHA

- 目前使用最多的两种散列函数是MD5和SHA序列函数。
- MD5的散列码长度为128比特。
- SHA序列函数是美国联邦政府的标准，如SHA-1散列码长度为160比特，SHA-2散列码长度为256、384和512位。

MD5的碰撞问题

- MD5的散列码长度为128比特，已经被证明是不安全的。2004年，山东大学的王小云（现为科学院院士，清华大学和山东大学的教授）第一次发现MD5算法存在碰撞的可能，并给出了实例。

```
Sequence #1
    d1  31  dd  02  c5  e6  ee  c4  69  3d  9a  06  98  af  f9  5c
    2f  ca  b5  87  12  46  7e  ab  40  04  58  3e  b8  fb  7f  89
    55  ad  34  06  09  f4  b3  02  83  e4  88  83  25  71  41  5a
    08  51  25  e8  f7  cd  c9  9f  d9  1d  bd  f2  80  37  3c  5b
    d8  82  3e  31  56  34  8f  5b  ae  6d  ac  d4  36  c9  19  c6
    dd  53  e2  b4  87  da  03  fd  02  39  63  06  d2  48  cd  a0
    e9  9f  33  42  0f  57  7e  e8  ce  54  b6  70  80  a8  0d  1e
    c6  98  21  bc  b6  a8  83  93  96  f9  65  2b  6f  f7  2a  70

Sequence #2
    d1  31  dd  02  c5  e6  ee  c4  69  3d  9a  06  98  af  f9  5c
    2f  ca  b5  07  12  46  7e  ab  40  04  58  3e  b8  fb  7f  89
    55  ad  34  06  09  f4  b3  02  83  e4  88  83  25  f1  41  5a
    08  51  25  e8  f7  cd  c9  9f  d9  1d  bd  72  80  37  3c  5b
    d8  82  3e  31  56  34  8f  5b  ae  6d  ac  d4  36  c9  19  c6
    dd  53  e2  34  87  da  03  fd  02  39  63  06  d2  48  cd  a0
    e9  9f  33  42  0f  57  7e  e8  ce  54  b6  70  80  28  0d  1e
    c6  98  21  bc  b6  a8  83  93  96  f9  65  ab  6f  f7  2a  70

Both produce MD5 digest 79054025255fb1a26e4bc422aef54eb4
```

SHA-1的碰撞问题

- On February 23, 2017, CWI Amsterdam and Google announced they had performed a collision attack against SHA-1. They had given 2 different PDF files with the same SHA-1 outputs.
- https://shattered.io/

Compared to other collision attacks

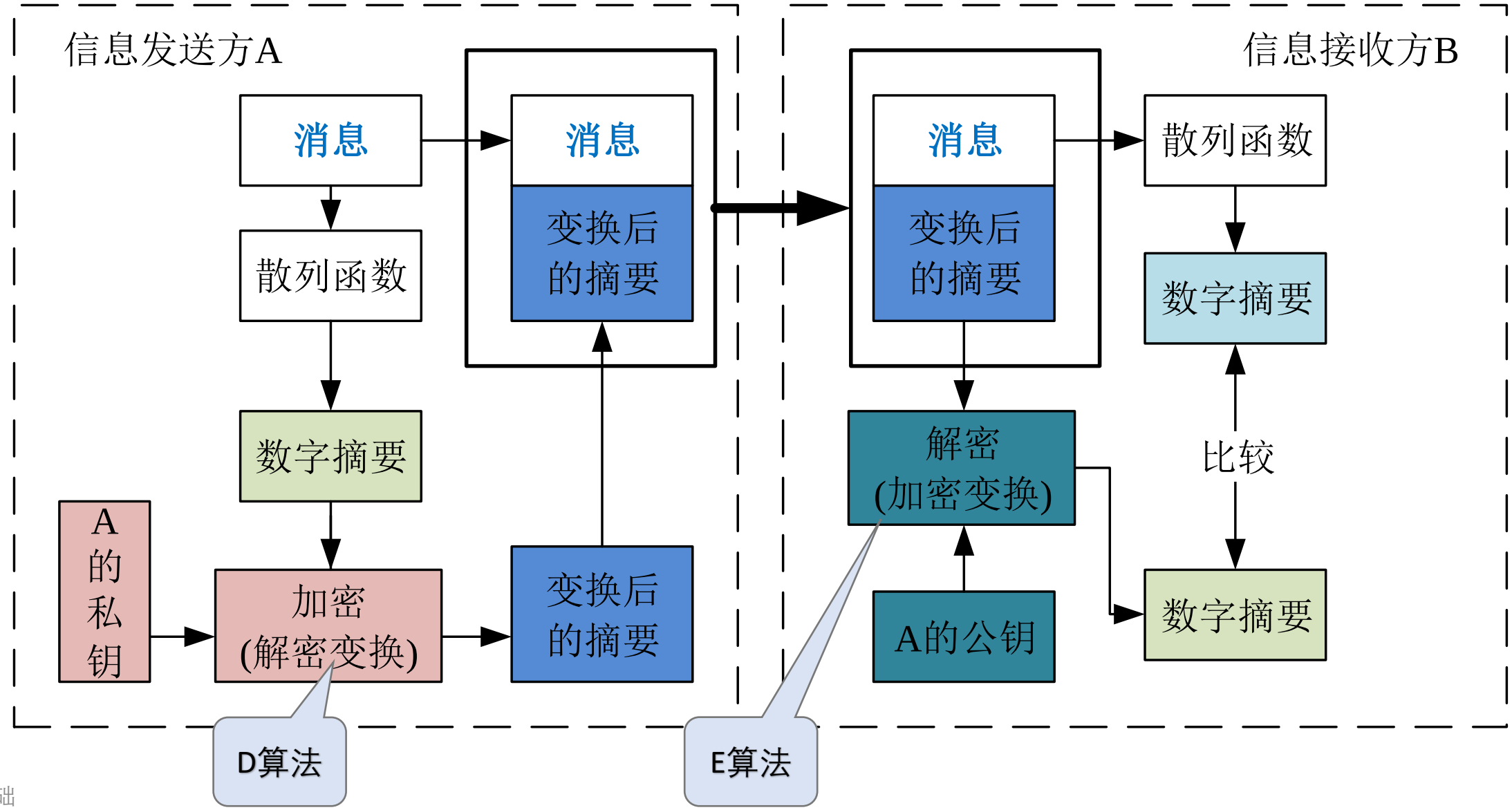
目前：
① 对于安全要求较高的应用，不能使用MD5
② 找到SHA-1的碰撞是较困难的；SHA-2是安全的

3.4.2 数字签名

- **数字签名(Digital Signature)**是指用户用自己的私钥对原始数据的消息摘要进行加密所得的数据，即加密的摘要。

- 信息接收者使用信息发送者的公钥对附在原始信息后的数字签名进行解密后获得消息摘要M1，并与原始数据产生的消息摘要M2对照，便可确信原始信息是否被篡改。这样就保证了消息来源的**真实性**和数据传输的**完整性**。

图3-7 用公钥算法实现数字签名及完整性验证



消息的保密传输

- 为了对消息进行保密传输，通常将公钥密码技术和对称密码技术结合起来使用。

① 在发送方A随机生成一个对称密码算法的密钥K。

② 然后用K对消息加密得到密文C并生成密文的数字摘要M，接着用A的私钥对M签名，将**密文C**、**用B的公钥加密的签名及K**发送给接收方B。

③ 接收方B进行相反的操作，就可以实现消息的保密传输及完整性验证。

3.5 PGP及其应用

- 为了保护电子邮件及文件的保密性，Phil Zimmermann提出了Pretty Good Privacy 加密标准，得到了广泛的应用。
- PGP（Pretty Good Privacy）是一个基于RSA公钥加密体系的邮件加密软件。

- PGP加密技术的创始人是美国的Phil Zimmermann。他创造性地把RSA公钥体系和传统加密体系的结合起来，并且在数字签名和密钥认证管理机制上有巧妙的设计，因此PGP成为目前几乎最流行的公钥加密软件包。

PGP简介

- 由于RSA算法计算量极大，在速度上不适合加密大量数据，所以PGP实际上用来加密的不是RSA本身，而是采用传统加密算法IDEA，IDEA加解密的速度比RSA快得多。
- PGP随机生成一个密钥，用IDEA算法对明文加密，然后**用RSA算法对密钥加密**。收件人同样是用RSA解出随机密钥，再用IEDA解出原文。这样的链式加密既有RSA算法的保密性(Privacy)和认证性(Authentication)，又保持了IDEA算法速度快的优势。

- PGP提供五种服务：
  - 鉴别，机密性，压缩，兼容电子邮件，分段

PGP简介

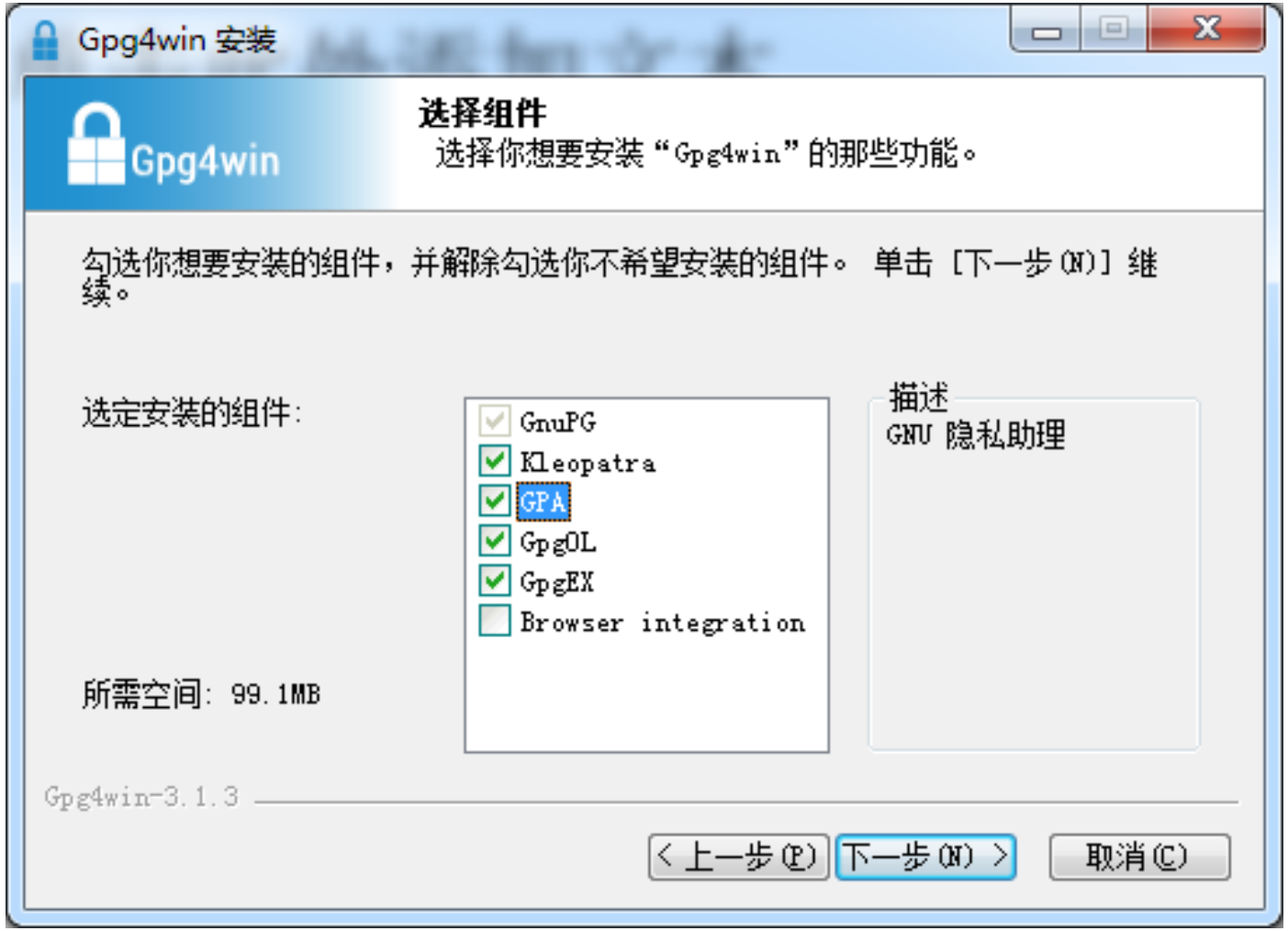
- PGP最初在Windows实现，直到PGP Desktop9.0一直为免费共享软件，后来PGP被Symantec收购，成为了收费软件。

- OpenPGP (http://www.openpgp.org/index.shtml)是源自PGP 标准的免费开源实现，目前是世界上应用最广泛的电子邮件加密标准。OpenPGP由IETF 的OpenPGP 工作组提出，其标准定义在**RFC 4880**。在Windows和Linux(Unix)下均有**免费开源**的版本。

- GunPG(The GNU Privacy Guard)是OpenPGP的最典型实现，目前支持Windows、Linux、MacOS等流行操作系统。相关软件可以从http://www.gnupg.org/ 下载。

从 http://www.gpg4win.org/ 网站下载Gpg4win的最新版本，安装界面如下(以2018年8月的Gpg4win3.1.3为例)：

注：2023-07-14的版本为gpg4win-4.2.0


Gpg4win 安装
Gpg4win
选择组件
选择你想要安装“Gpg4win”的那些功能。
勾选你想要安装的组件，并解除勾选你不希望安装的组件。 单击 [下一步(N)] 继续。
选定安装的组件:
GnuPG
Kleopatra
GPA
GpgOL
GpgEX
Browser integration
描述
GNU 隐私助理
所需空间: 99.1MB
Gpg4win-3.1.3
< 上一步(P)
下一步(N) >
取消(C)

选择安装所需的组件

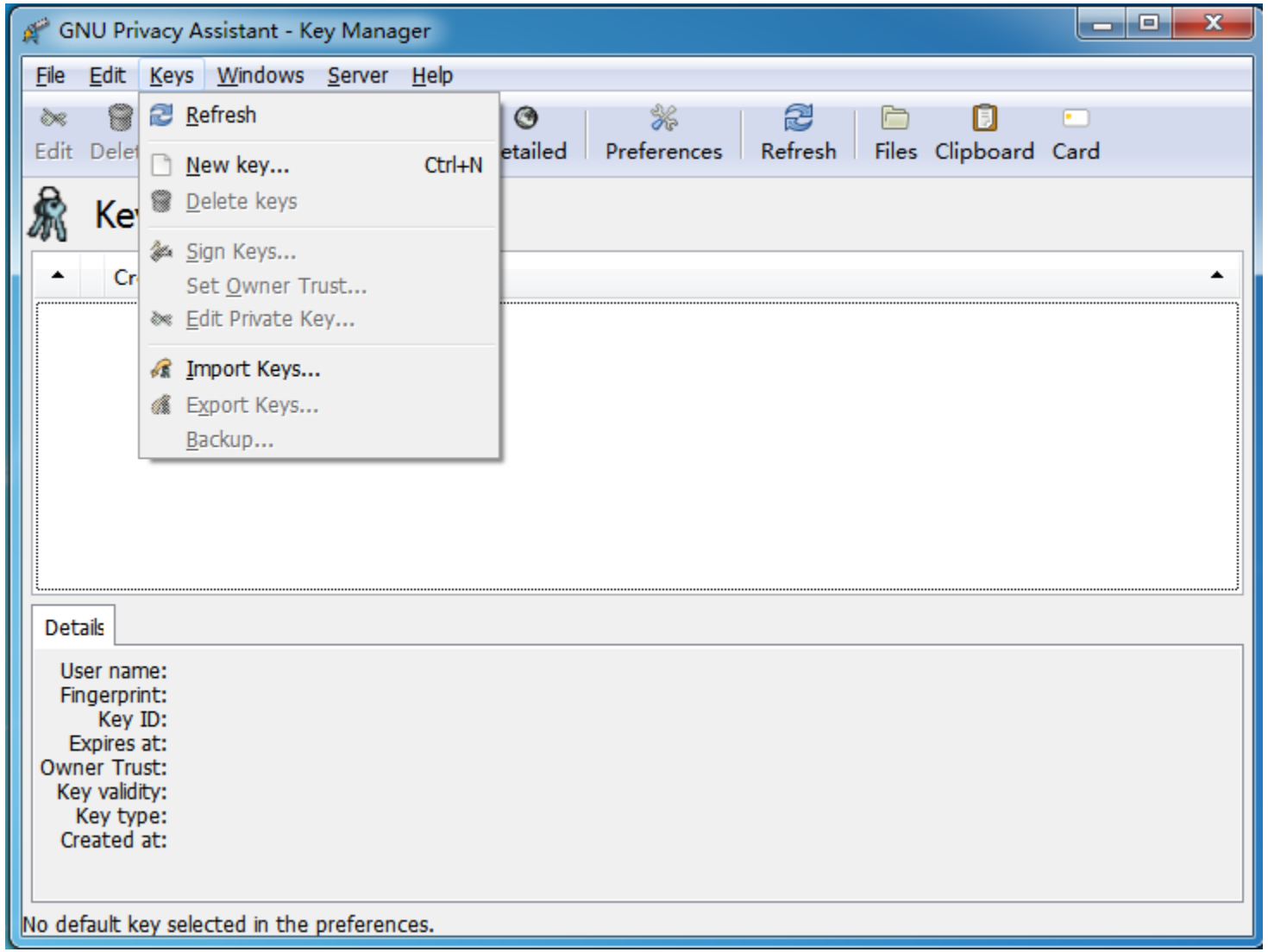
Gpg4win 安装
Gpg4win
GnuPG for Windows
正在完成“Gpg4win”安装向导
“Gpg4win”已安装在你的系统。
单击 [完成(F)] 关闭此向导。
Run Kleopatra
显示 README 文件
点击此处访问项目首页
< 上一步(P)
完成(F)
取消(C)

完成安装

Gpg4win的使用

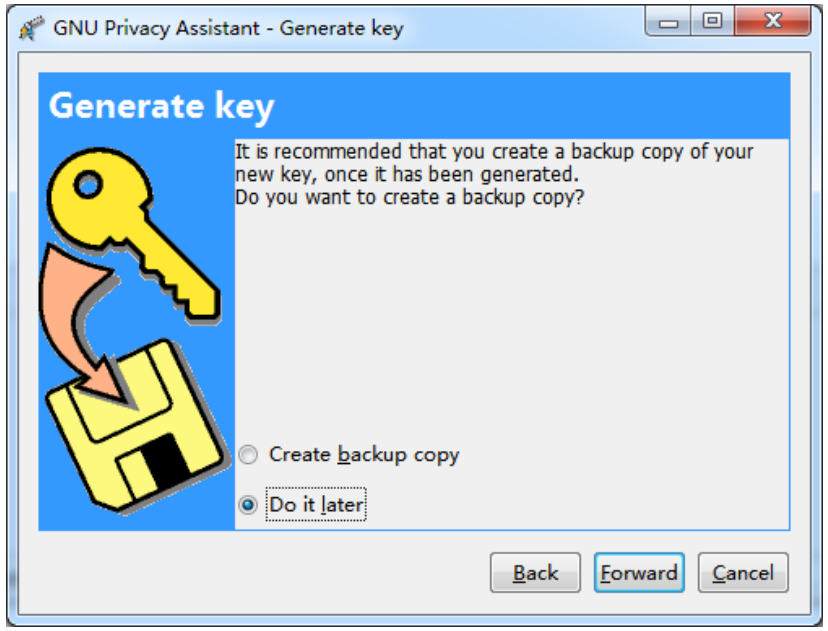
- 安装结束后认真阅读 README.en文件。按以下步骤使用加密和解密功能。

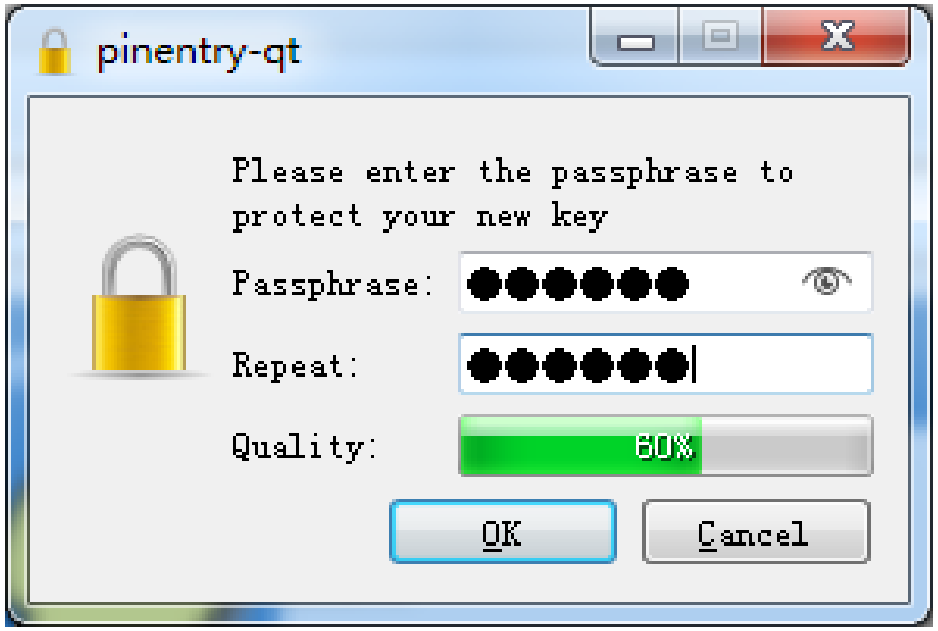
- **步骤1：产生一对RSA密钥**
  - 启动GPA（Windows7下以管理员身份运行），产生一对密钥，如下图所示。

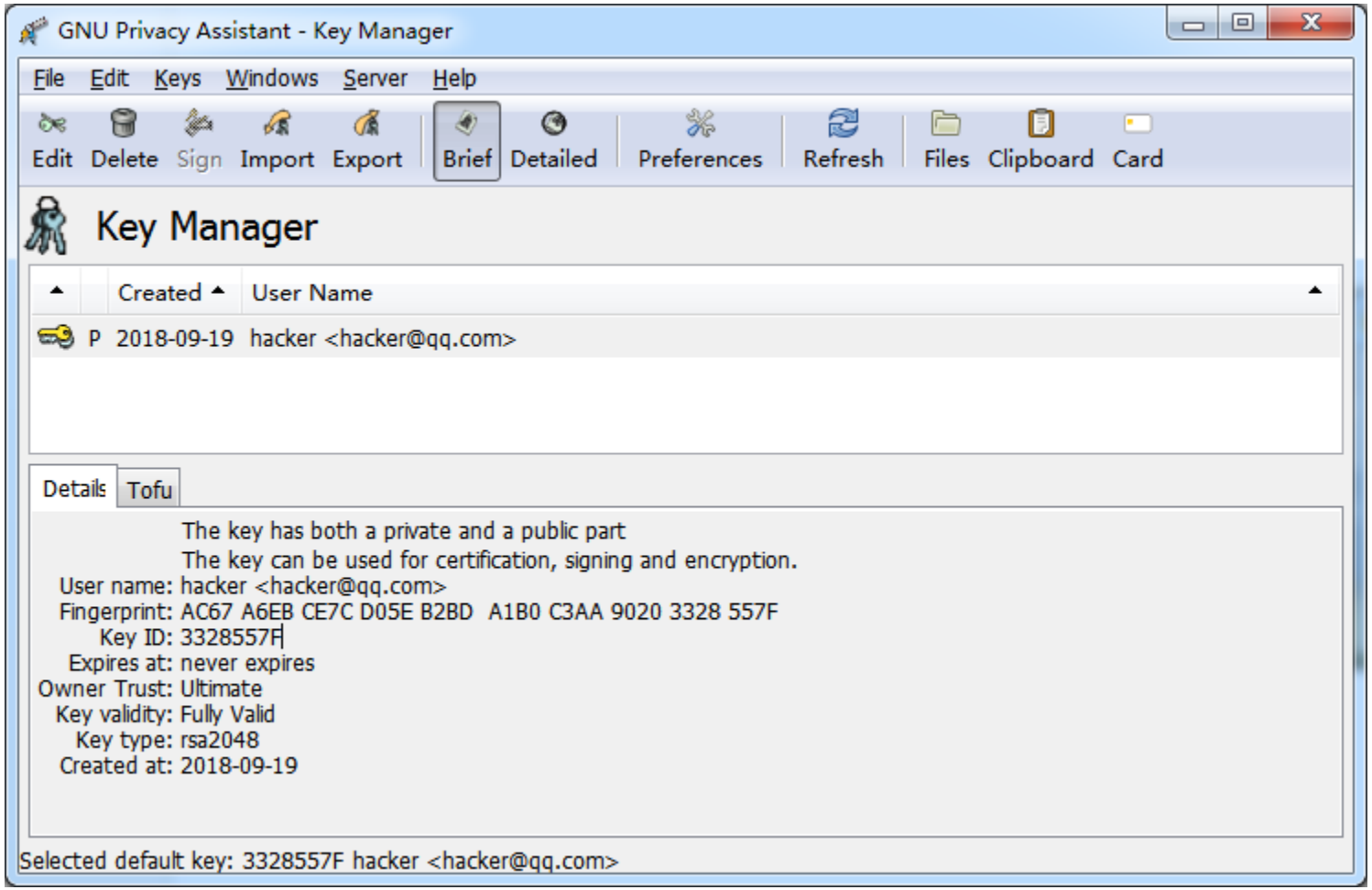

GNU Privacy Assistant - Key Manager
File
Edit
Keys
Windows
Server
Help
Refresh
New key...
Ctrl+N
Delete keys
Sign Keys...
Set Owner Trust...
Edit Private Key...
Import Keys...
Export Keys...
Backup...
Edit
Preferences
Refresh
Files
Clipboard
Card
Details
User name:
Fingerprint:
Key ID:
Expires at:
Owner Trust:
Key validity:
Key type:
Created at:
No default key selected in the preferences.

GNU Privacy Assistant - Generate key
Generate key
Please insert your full name.
Your name will be part of the new key to make it easier for others to identify keys.
Your Name:
hacker
Forward
Cancel

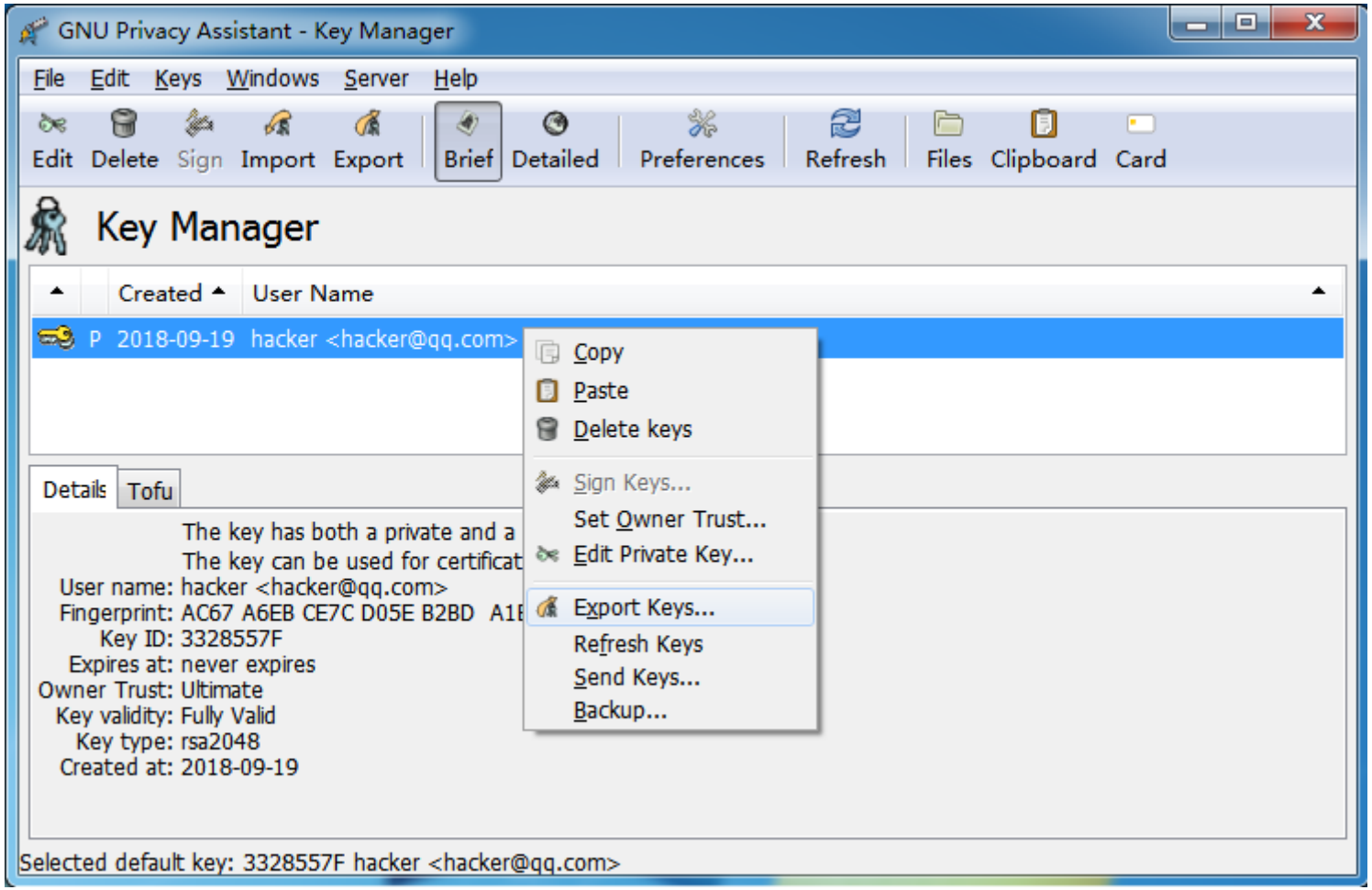
GNU Privacy Assistant - Generate key
Generate key
Please insert your email address.
Your email address will be part of the new key to make it easier for others to identify keys. If you have several email addresses, you can add further email addresses later.
Your Email Address:
hacker@qq.com
Back
Forward
Cancel


GNU Privacy Assistant - Generate key
Generate key
It is recommended that you create a backup copy of your new key, once it has been generated.
Do you want to create a backup copy?
Create backup copy
Do it later
Back
Forward
Cancel

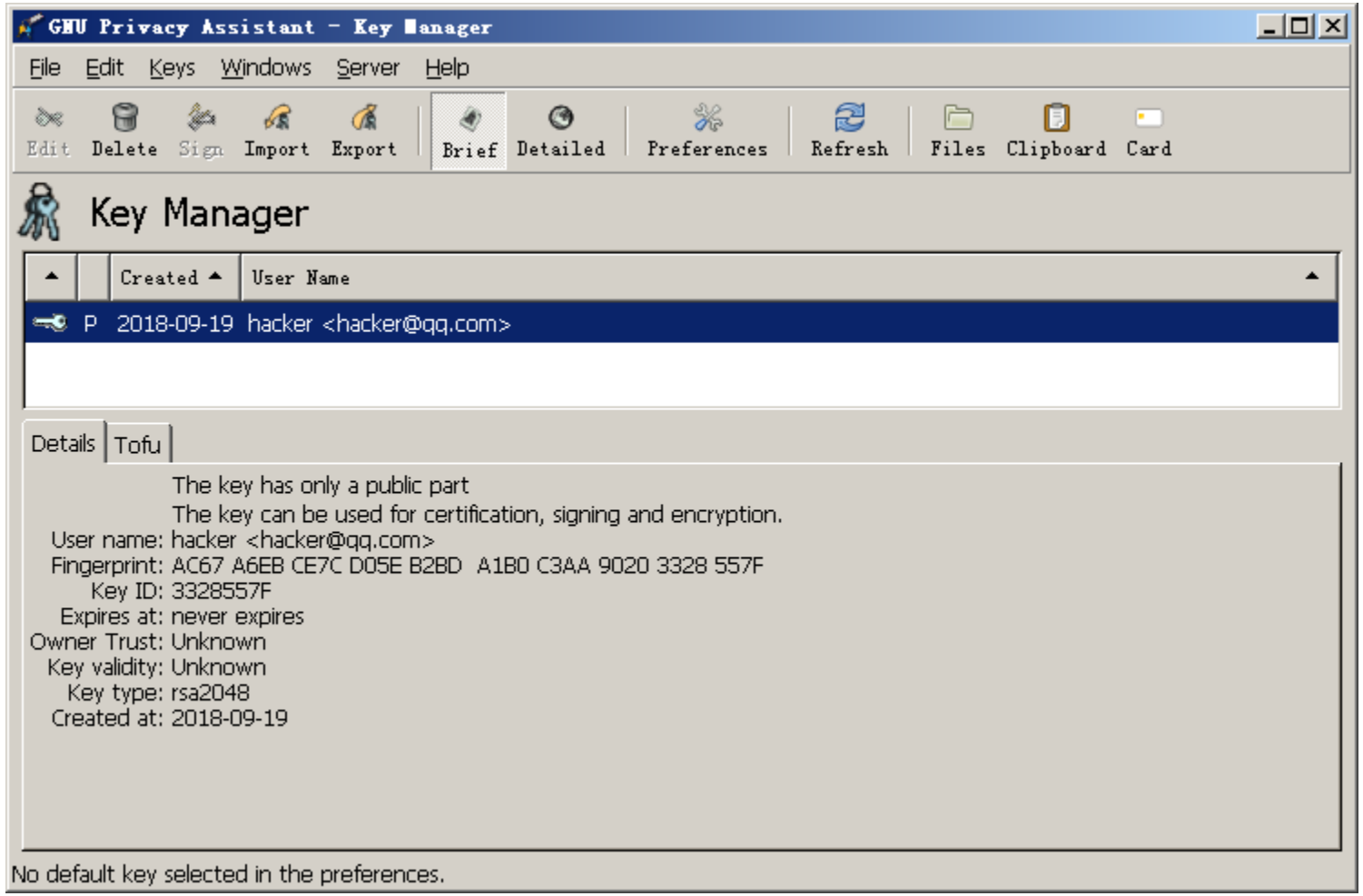

pinentry-qt
Please enter the passphrase to protect your new key
Passphrase:
Repeat:
Quality:
60%
OK
Cancel


GNU Privacy Assistant - Key Manager
File Edit Keys Windows Server Help
Edit Delete Sign Import Export Brief Detailed Preferences Refresh Files Clipboard Card
Key Manager
Created User Name
P 2018-09-19 hacker <hacker@qq.com>
Details Tofu
The key has both a private and a public part
The key can be used for certification, signing and encryption.
User name: hacker <hacker@qq.com>
Fingerprint: AC67 A6EB CE7C D05E B2BD A1B0 C3AA 9020 3328 557F
Key ID: 3328557F
Expires at: never expires
Owner Trust: Ultimate
Key validity: Fully Valid
Key type: rsa2048
Created at: 2018-09-19
Selected default key: 3328557F hacker <hacker@qq.com>

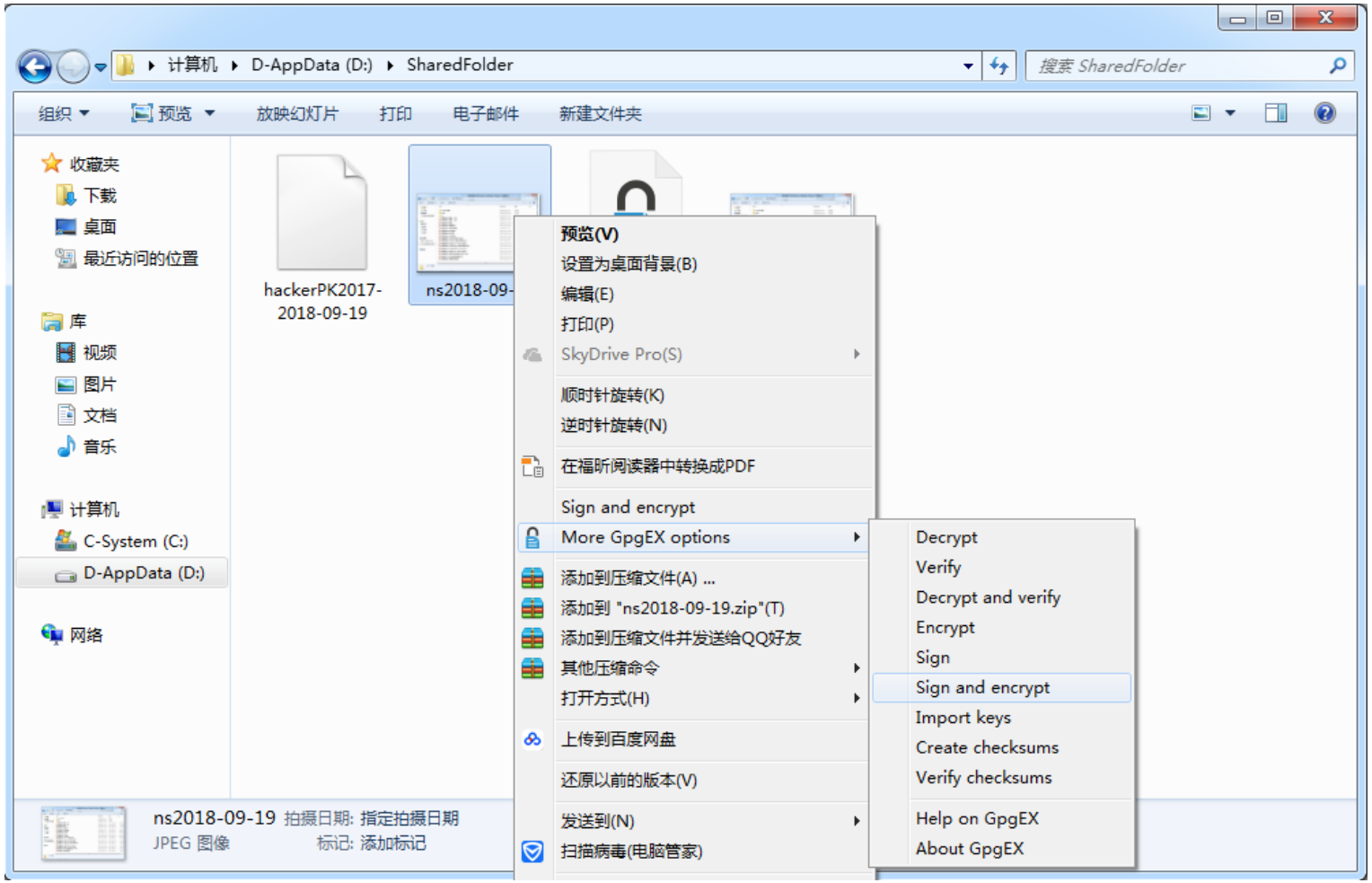
步骤2：互换公钥

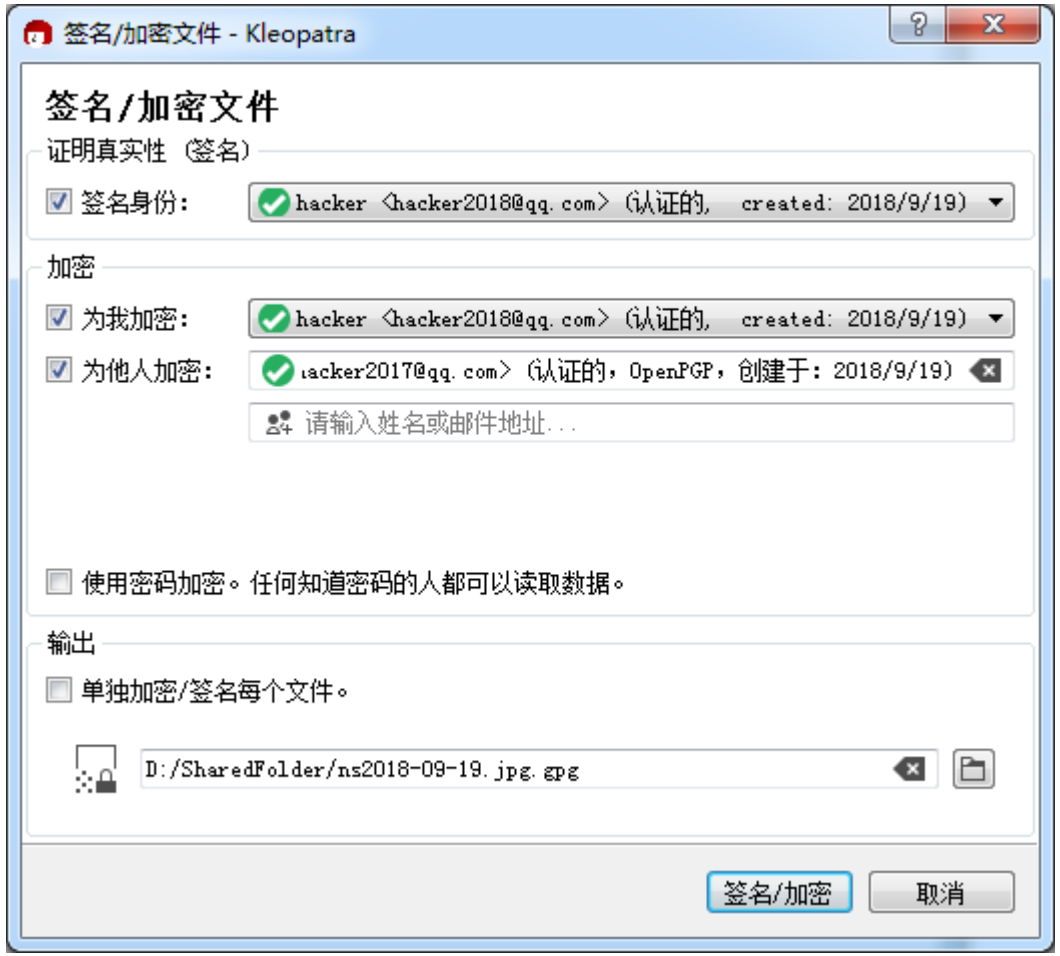

GNU Privacy Assistant - Key Manager
File Edit Keys Windows Server Help
Edit Delete Sign Import Export Brief Detailed Preferences Refresh Files Clipboard Card
Key Manager
Created User Name
P 2018-09-19 hacker <hacker@qq.com>
Copy
Paste
Delete keys
Sign Keys...
Set Owner Trust...
Edit Private Key...
Export Keys...
Refresh Keys
Send Keys...
Backup...
Details Tofu
The key has both a private and a
The key can be used for certificat
User name: hacker <hacker@qq.com>
Fingerprint: AC67 A6EB CE7C D05E B2BD A1
Key ID: 3328557F
Expires at: never expires
Owner Trust: Ultimate
Key validity: Fully Valid
Key type: rsa2048
Created at: 2018-09-19
Selected default key: 3328557F hacker <hacker@qq.com>

导入公钥


GNU Privacy Assistant - Key Manager
File Edit Keys Windows Server Help
Edit Delete Sign Import Export Brief Detailed Preferences Refresh Files Clipboard Card
Key Manager
Created User Name
P 2018-09-19 hacker <hacker@qq.com>
Details Tofu
The key has only a public part
The key can be used for certification, signing and encryption.
User name: hacker <hacker@qq.com>
Fingerprint: AC67 A6EB CE7C D05E B2BD A1B0 C3AA 9020 3328 557F
Key ID: 3328557F
Expires at: never expires
Owner Trust: Unknown
Key validity: Unknown
Key type: rsa2048
Created at: 2018-09-19
No default key selected in the preferences.

步骤3：向对方发送加密文件




签名/加密文件 - Kleopatra
签名/加密文件
证明真实性（签名）
签名身份：
hacker <hacker2018@qq.com> (认证的, created: 2018/9/19)
加密
为我加密：
hacker <hacker2018@qq.com> (认证的, created: 2018/9/19)
为他人加密：
acker2017@qq.com> (认证的，OpenPGP，创建于：2018/9/19)
请输入姓名或邮件地址...
使用密码加密。任何知道密码的人都可以读取数据。
输出
单独加密/签名每个文件。
D:/SharedFolder/ns2018-09-19.jpg.gpg
签名/加密
取消

- 点击Encrypt按钮将加密指定的文件，得到扩展名为gpg的加密文件，将该文件发送给私钥持有者。

- 私钥持有者对其解密(需要输入passphrase)后可以恢复出原文件。

实际演示

3.6 使用OpenSSL中的密码函数

- OpenSSL(http://www.openssl.org/)是使用非常广泛的SSL的开源实现，是用C语言实现的。由于其中实现了为SSL所用的各种加密算法，因此OpenSSL也是被广泛使用的加密函数库。

- 有两种方式使用OpenSSL的加/解密功能：

1. 其一是在命令行下运行OpenSSL，以适当的参数运行openssl命令，就可以实现加密和解密功能；
2. 另一种是在自己的应用程序中使用加密函数，这需要利用openssl提供的C语言接口，以函数调用的方式式使用加密函数库。

3.6.1 在命令行下使用OpenSSL

- Windows和Linux环境下的OpenSSL有相同的命令行程序名openssl。在命令行窗口下运行 **openssl**，可以列出OpenSSL支持的命令。

- OpenSSL的命令分成三类：标准命令、数字摘要命令和加密命令。

命令类别	子命令
标准命令 **Standard commands**	asn1parse, ca, ciphers, cms, crl, crl2pkcs7, dgst, dh, dhparam, dsa, dsaparam, ec, ecparam, enc, engine, errstr, gendh, gendsa, genpkey, genrsa, nseq, ocsp, passwd, pkcs12, pkcs7, pkcs8, pkey, pkeyparam, pkeyutl, prime, rand, req, rsa, rsautl, s_client, s_server, s_time, sess_id, smime, speed, spkac, srp, ts, verify, version, x509
数字摘要命令 **Message Digest commands**	md4, md5, mdc2, rmd160, sha, sha1
加密命令 **Cipher commands**	aes-128-cbc, aes-128-ecb, aes-192-cbc, aes-192-ecb, aes-256-cbc, aes-256-ecb, base64, bf, bf-cbc, bf-cfb, bf-ecb, bf-ofb, camellia-128-cbc, camellia-128-ecb, camellia-192-cbc, camellia-192-ecb,camellia-256-cbc, camellia-256-ecb, cast, cast-cbc,cast5-cbc, cast5-cfb, cast5-ecb, cast5-ofb,des, des-cbc, des-cfb, des-ecb, des-ede, des-ede-cbc, des-ede-cfb, des-ede-ofb, des-ede3, des-ede3-cbc, des-ede3-cfb, des-ede3-ofb, des-ofb, des3, desx, idea, idea-cbc, idea-cfb, idea-ecb, idea-ofb, rc2, rc2-40-cbc, rc2-64-cbc, rc2-cbc, rc2-cfb, rc2-ecb, rc2-ofb, rc4, rc4-40, seed, seed-cbc, seed-cfb, seed-ecb, seed-ofb

OpenSSL支持的命令

- OpenSSL支持的命令是相当丰富的。如果对某个命令的用法不是很清楚，可以用“openssl 命令名称 -help” (注：早期版本可能不支持）查看该命令的说明。

- 例如，如果不了解“openssl passwd”的用法，可以在命令行下输入“openssl passwd -help”，运行结果(ubuntu Linux)如下：
  - fanping@vbu32:~/work$ openssl passwd -help
  - Usage: passwd [options] [passwords]
  - ……

常用的OpenSSL的命令

功能	命令及说明
版本和编译参数	显示版本和编译参数: openssl version -a
支持的子命令、密码算法	查看支持的子命: openssl help SSL密码组合列表: openssl ciphers
测试密码算法速度	测试所有算法速度: openssl speed 测试RSA速度: openssl speed rsa 测试DES速度: openssl speed des
RSA密钥操作	产生RSA密钥对: openssl genrsa -out 1.key 1024 取出RSA公钥: openssl rsa -in 1.key -pubout -out 1.pubkey
加密文件	加密文件: openssl enc -e -rc4 -in 1.key -out 1.key.enc 解密文件: openssl enc -d -rc4 -in 1.key.enc -out 1.key.dec
计算Hash值	计算文件的MD5值: openssl md5 < 1.key 或 openssl md5 1.key 计算文件的SHA1值: openssl sha1 < 1.key

实例1

- **[实例1]**

- 密钥在文件key.txt中，用des3算法对文件test.data加密和解密，并验证其正确性。
  ① 加密为test.3des：
  openssl enc -e -des3 -in test.data -out test.3des -kfile key.txt
  ② 解密test.3des为test.dddd：
  openssl enc -d -des3 -in test.3des -out test.dddd -kfile key.txt
  ③ 验证test.dddd和原始文件test.data相同：
  openssl md5 test.dddd test.data

3.6.2 在Windows的C程序中使用OpenSSL

- OpenSSL提供了C语言接口所需的头文件、库文件和动态链接库。为了使用该接口，必须安装面向软件开发人员的软件包(安装文件较大)，并将openssl的lib和include目录添加到lib和环境变量中。对于Visual Studio C++开发平台，最简单的方法是将openssl的lib和include目录拷贝到VC目录(默认安装在C:\Program Files\Microsoft Visual Studio 9.0\VC)中，这样就不需要额外设置环境变量。

- 为了使用OpenSSL库函数，在C程序中必须包含相应的头文件，链接的时候必须加入相关的库。

3.6.3 在Linux的C程序中使用OpenSSL

- Linux系统的发行版一般预装了命令行OpenSSL程序，没有安装openssl库。为了在C程序中使用OpenSSL，需要安装openssl库。

- 在ubuntu Linux 系统中运行以下命令安装openssl库：

  sudo apt-get install libssl-dev

- 在fedora Linux 系统中切换到root，再运行以下命令安装openssl库：

  yum install openssl-devel.x86_64

  或 yum install openssl-devel.i686

AES算法的实例

- 例程： cryptoDemo.cpp // 测试AES算法的例子。

演 示

3.7 Windows系统提供的密码算法

- Windows通过CryptoAPI提供密码算法服务，支持数据的加密/解密和基于数字证书的身份认证等功能，同时也允许第三方开发符合Windows规范的密码算法。程序员只须调用相应的API函数就可以完成加密操作，而不必了解算法的实现细节。

- CryptoAPI系统架构如图3-18所示。

- CryptoAPI函数使用CSP(cryptographic service providers, 密码服务提供者)执行加密和解密、提供密钥存储和安全。CSP是独立于具体应用程序的模块，因此一个应用程序可以运行多个CSP模块。

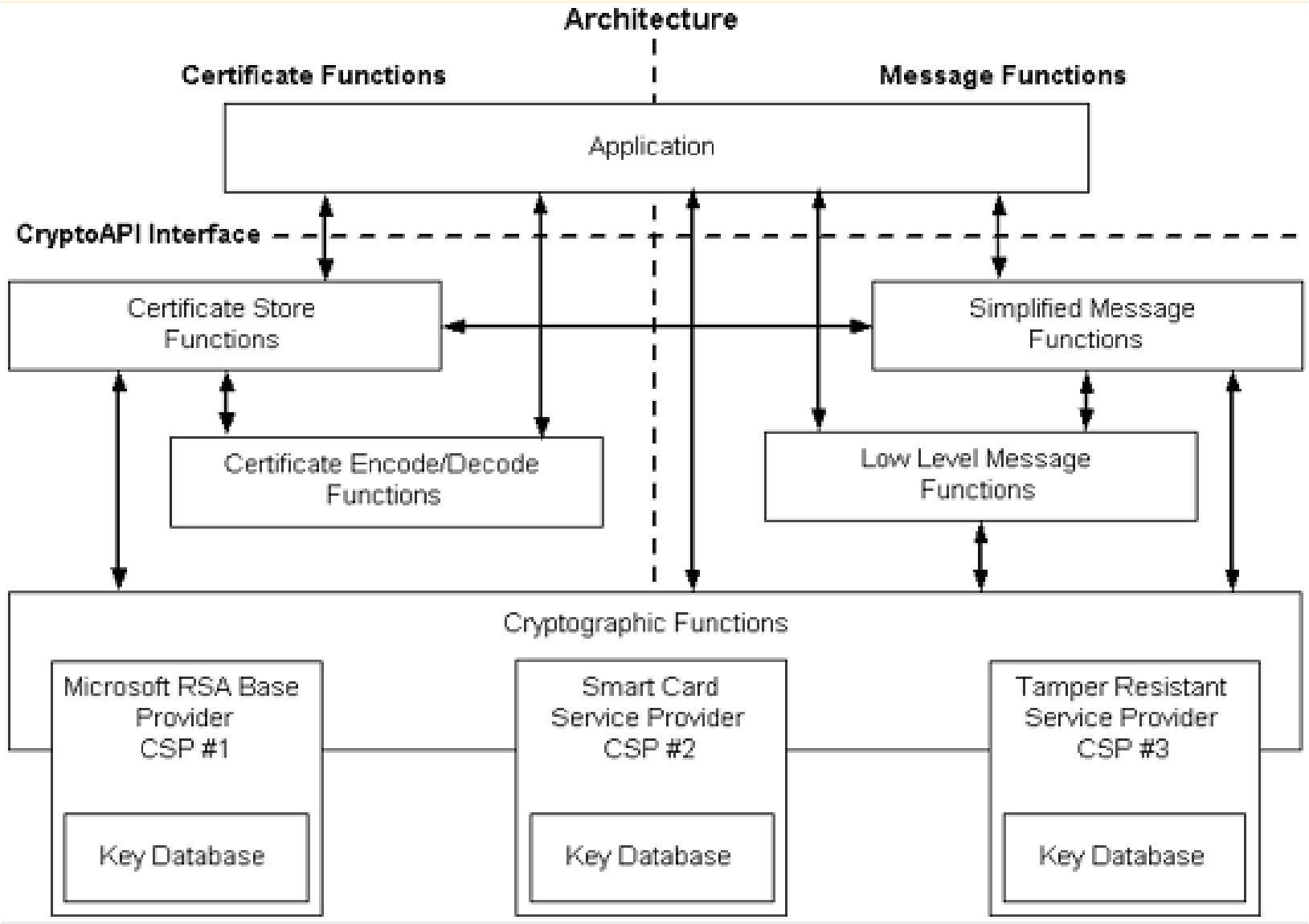


图3-18 CryptoAPI系统架构

3.7.1 密码服务提供者CSP

- CSP是真正执行加密工作的独立的模块。物理上一个CSP由两部分组成：一个动态链接库和一个签名文件。每个CSP都有一个名字和一个类型。每个CSP的名字是惟一的，这样便于CryptoAPI找到对应的CSP。

- 函数CryptEnumProviderTypes可以枚举系统中的CSP类型和该类型的名字，函数CryptEnumProviders可以枚举系统中CSP的名字和类型。例程enumerateProvidersAndTypes.cpp枚举了系统中的CSP类型、类型名和CSP名字。

密码服务提供者CSP

- 为了使用CSP提供的密码算法，首先必须调用CryptAcquireContext获得指向特定CSP的**句柄(handle)**，该句柄代表CSP提供者及对应的密钥容器。
- 对于Windows系统中的每个用户，每个CSP都有多个密钥容器，每个密钥容器由惟一的名字标识。密钥容器存储了用户的密钥，包括**签名密钥**和**密钥交换密钥**。
- 以密钥容器名字作为函数CryptAcquireContext()的参数，函数将返回指向这个密钥容器的句柄。如果该名字的容器不存在，可以用CryptAcquireContext()函数产生一个新的密钥容器。
- 使用完CSP的密钥容器后，用函数 CryptReleaseContext()释放其对应的句柄。

3.7.2 使用CSP提供的密码技术实现保密通信

- 使用CSP实现保密通信的主要过程如下:
  (1) 用CryptGenKey生成一个随机会话密钥;
  (2) 用该会话密钥加密数据;
  (3) 指定目标用户的公钥，用CryptExportKey将会话密钥导出为一个BLOB密钥，该导出的密钥是被目标用户的公钥加密了;
  (4) 发送加密的信息和加密的BLOB密钥给目标用户;
  (5) 目标用户用CryptImportKey导入BLOB密钥到其CSP。只要在步骤(3)指定了目标用户的公钥，则导入密钥时会自动解密会话密钥。
  (6) 目标用户用会话密钥解密所收到的加密信息

MSDN中的例子程序

- Microsoft的MSDN中给出了四个例子程序，演示了在应用程序中使用CSP及密码函数的方法。

- 由于例子程序较大，在此不做进一步的分析。感兴趣的读者请参考MSDN。

3.8 python语言的密码模块

- 在pipy.org有2个可用的python密码模块

  旧的pycrypto:

  - Latest version Released: Oct 18, 2013
  - https://pypi.org/project/pycrypto/

  新的pycryptodome :

  - Latest version Released: Feb 9, **2021**
  - https://pypi.org/project/pycryptodome/

- Pycryptodome的安装和使用

  安装： pip install pycryptodome

  使用AES： from Crypto.Cipher import AES

用python密码库的AES算法，实现数据的加密和解密

```
from Crypto.Cipher import AES
from Crypto.Random import get_random_bytes

key = get_random_bytes(16)
cipher = AES.new(key, AES.MODE_EAX)
ciphertext, tag = cipher.encrypt_and_digest(data)

file_out = open("encrypted.bin", "wb")
[ file_out.write(x) for x in (cipher.nonce, tag, ciphertext) ]
file_out.close()
```

```
from Crypto.Cipher import AES

file_in = open("encrypted.bin", "rb")
nonce, tag, ciphertext = [ file_in.read(x) for x in (16, 16, -1) ]

# let's assume that the key is somehow available again
cipher = AES.new(key, AES.MODE_EAX, nonce)
data = cipher.decrypt_and_verify(ciphertext, tag)
```

https://www.pycryptodome.org/en/latest/src/examples.html

谢谢！

第3讲 作业与实践

- **作业**

1. 在腾讯会议系统中，对称密码技术和公钥密码技术适合应用在哪几个阶段？说明理由。
2. 简述散列函数MD5的碰撞问题。既然MD5存在碰撞问题，为何http://mirrors.ustc.edu.cn/ubuntu-releases/16.04/仍然给出MD5值作为完整性验证的依据。
3. 简述用RSA公钥算法实现数字签名的过程。
4. 用PGP加密某个文件，如果接收该加密文件的用户为1个，加密文件的大小为24kB；如果接收该加密文件的用户为10个，请问加密文件的大小是原来的10倍(240kB)吗？为什么？

- 实践(不考核，自己练习)

1. 熟悉OpenSSL命令行程序的使用。
2. 熟悉Gpg4win的使用。
3. 熟悉Windows的CryptoAPI系统架构，阅读例子程序。
4. 试用Python语言实现AES对二进制文件的加/解密。
5. 修改例程cryptoDemo.cpp为encfile.cpp：从命令行接受3个字符串类型的参数：参数1，参数2，参数3。参数1=enc表示加密，参数1=dec表示解密；参数2为待加密、解密的文件名；参数3为密码。