# 第2章 密码技术理论

本章的目的是阐述与密码学相关的基础知识。首先给出密码基本理论与技术的概述，介绍信息安全保护的基本要素；然后讨论对称密码机制和公钥密码机制的基本原理、类型划分和加密标准等；最后介绍哈希函数、消息认证码和数字签名算法的基本概念。学习完本章后，我们就能为后面章节网络安全的学习打下基础。

## 2.1 密码基本理论与技术概述

在网络信息时代，信息安全至关重要。密码学是一门利用密码算法实现信息安全保护的关键学科。在密码学中，通信双方都可以读懂的原始消息被称作明文，而将明文通过一定方式处理后所生成的不可读的消息则被称为密文。当代信息安全保护的主要内容涉及以下几个要素：

（1）机密性

在网络信息传输的过程中，需要确保只有授权用户可获取明文信息内容，非授权用户即使获取到公开传输的数据也无法知晓明文信息内容。在密码学领域中，一般可采用加密、编码等方式实现数据的机密性。

（2）完整性

在网络信息传输的过程中，需要确保接收方收到的消息内容与发送方一致，未被其他人篡改。在密码学领域中，一般采用hash函数、数字签名等方式实现数据完整性。

（3）认证性

在网络信息传输的过程中，需要确保数据来源的有效性以及发送方实体的合法性，防止攻击者假冒发送者传输数据。在密码学领域中，一般采用消息鉴别、实体认证等方式实现认证特性。

（4）隐私性

在网络信息传输的过程中，应着重保护发送方的敏感信息，防止攻击者获取其敏感数据。在密码学领域中，一般采用加密、匿名标识等方式实现敏感数据的隐私性。

介绍完信息安全保护的关键要素后，下面介绍密码学的一些基础知识。

密码学技术根据密钥的存在与否可以分为两大类，无密钥加密技术和密钥加密技术。无密钥加密技术不依赖传统密钥机制对信息进行加密，其核心是通过数学变换或编码规则直接实现信息的加密保护，例如将数字“1234”编码为“aczx”以公开传输。而密钥加密技术，顾名思义，是指通过密钥机制实现信息的加密。本章节主要介绍密钥加密技术，具体地，根据密钥的类型，密钥加密技术一般又可以被分为两类：对称加密技术和公钥加密技术。

## 2.2 对称密码体制

对称加密体制又称为单钥密码体制，在该密码体制中，通信双方或多方的同一密钥可同时作为加密密钥和解密密钥。对称加密技术的主要应用包括通过数据加解密算法实现数据的机密性保护，以及利用哈希算法保证数据的完整性和认证性。在对称加密技术中，一般默认加密、解密、哈希等算法都是公开的，所以对称加密算法的安全性主要依赖于对称密钥，若对称密钥丢失，则该加密体制被攻破。因此，在对称加密技术的使用过程中，必须严格确保对称密钥的安全性，防止非法者获得对称密钥。图2.1显示了对称加密技术的基本工作原理。简单来说，发送方采用对称密钥加密明文消息获得密文，而接收方采用同一个密钥解密密文获得明文。在加解密算法设计合适的条件下，攻击者没有密钥是无法获得明文的。若加解密算法设计不当，也可能导致攻击者在没有密钥的情况下仍可获得明文。常用的对称加密体制包括DES、3DES、AES、IDEA等。



图2.1对称加密技术的基本工作原理

图2.2展示了对称加密技术在确保数据完整性以及认证性的一般应用流程，在该流程中发送方将明文和密钥作为哈希算法的输入值，获得相应的哈希值输出，随后发送方将明文和哈希值一并发送给接收方。接收方同样将明文和密钥作为哈希算法的输入值，并计算得到新的哈希值，最后接收方判断新的哈希值与接收到的哈希值是否一致，如果两者一致，由于仅合法发送方拥有密钥能够计算得到该哈希值且哈希函数具有单向性，此消息一定由合法的发送方发送，且明文数据没有被篡改。反之，若两者不一致，则用户认证失败或消息已遭到篡改。



图2.2对称消息验证码算法

按照对明文消息加密方式的不同，对称加密算法一般可以分为两大类：流密码和分组密钥。

### 2.2.1 流密码

流密码，也称为序列密码，该密码体制的基本思想是把明文M=m1m2…mn以位或字节为单位进行加密，一般加密过程是将明文与密钥K=k1k2…kn进行混合（如异或运算）获得密文流C= c1c2…cn=Ek1 (m1) Ek2 (m2)…Ekn (mn)，其加解密过程如图2.3所示。流密码使用一个“种子密钥”产生密钥流，通信双方共享这个“种子密钥”，按照相同的方式产生完全随机的非周期比特流或字节流。流密码的安全强度完全依赖于密钥序列的随机性和不可预测性，而实现流密码的关键技术在于保持收发两端密钥流的精确同步。流密码最早的雏形是一次一密密码技术，而完美的一次一密机制实现过程需要无限存储单元和复杂的输出逻辑函数，且任何传输中的比特丢失或错位均会导致解密失败，这对通信系统的容错性提出了苛刻要求。



图2.3 流密码加解密过程

流密码分为同步流密码和自同步流密码两种，同步流密码的密钥流的产生独立于明文和密文，而自同步流密码的密钥流的产生与密钥和密文均有一定关联，是一种有记忆变化的序列密码。

一次性密码本是最常见的流密码技术之一，其基本思想是将明文与随机序列异或产生密文。由于加密所用的密钥是完全随机的且是一次性的，所以密文也是完全随机的，攻击者无法从密文中获取任何明文相关信息。而完全随机的密钥仅在理论上可行，也就意味着一次性密码本仅在理论上无法被破译。在现实中，一次性密码本并没有被广泛使用，主要原因是一次性密码本存在以下局限性：首先，一次性密码本的安全严格依赖于密码本，密码本丢失则安全性将不存在；其次，密码本需完全随机，在实际中不可能达到；最后，密钥序列至少与明文同长且密钥只能使用一次，但是明文一般来说不是有限的，这就要求密钥序列是无限随机的，例如加密1GB数据需要1GB密钥序列，这对存储和传输提出了近乎不现实的要求。

流密码主要应用在军事、政治等领域，虽然也有部分公开设计和研究成果发表，但是流密码的相关研究成果是保密的。目前可以公开见到、较有影响力的流密码方案主要包括A5、SEAL、RC4、PIKE等。

### 2.2.2 分组密码

分组密码是当今国际商用密码的主流密码体制，其基本思想是将明文M=m1m2…mn划分成若干等长的块(m1m2…ml) (ml+1ml+2…m2l)…，然后再利用密钥K=k1k2…kl分别对每个分组进行加密，得到密文分组C=(c1c2…cl) (cl+1cl+2…c 2l)…=EK(m1m2…ml) EK(ml+1ml+2…m2l)…，解密过程与加密过程类似，利用相同的密钥K分别对每个分组进行解密，分组密码的基本加解密流程如图2.4所示。为确保所有明文消息都能够被有效加密，分组密码体制一般首先将明文消息进行填充，使其长度为固定分组长度l的整数倍。分组密码与流密码的不同之处在于，流密码加密时不一定得到相同的密文，因为明文的重复部分是使用密钥流的不同部分加密的；而对于分组密码，在一个固定密钥的作用下，对相同的明文加密，一定能得到相同的密文。目前，实际使用中，分组密码的分组长度一般为64比特、128比特或256比特。常见的分组密码有DES、3DES、AES、IDEA等。



图2.4 分组密码加解密流程

分组密码设计的两条基本准则是扩散和混淆。扩散是指密钥或明文中的任一比特应影响密文中的许多比特，这样可充分隐藏密钥或明文的数字统计特性；混淆是指密文和密钥或明文之间的依赖关系应相当复杂，最好是相互独立，这样可干扰密码分析者的分析工作。

分组密码有SP和Feistel两种网络结构。SP结构即替换-置换网络结构，将加解密过程拆分为多轮加解密基本单元的乘积，各轮所做的操作通常是一致的（最终轮可能不一样）。典型SP结构的每一轮通常包含两种类型的变换：由轮密钥控制的替换操作和与密钥无关的置换操作。其中替换操作打破明文与密钥的线性关系，实现混淆；置换操作通过位或字节的线性扩散，将单个S盒的影响扩散到整个块，实现扩散。SP结构的加解密算法通常不一致，Feistel结构则是一种特殊的SP结构，它将输入分成长度相同的左右分组，每一轮的左分组等于上一轮的右分组，右分组等于上一轮的左分组和轮函数的异或，轮函数的输入是右分组和轮密钥。Feistel结构的加解密算法是一致的，这为它的应用提供了极大的便捷。

分组密码体制的作用不仅限于机密性保护，它还可用于流密码构造器、消息认证码生成器、伪随机数生成器等重要密码系统组成部分。此外，分组密码还是单钥数字签名体制的核心组成部分。

### 2.2.3 数据加密标准（Data Encryption Standard，DES）

20世纪70年代初，由于当时加密技术存在算法不兼容、安全性参差不齐等问题，1973年美国国家标准局（National Bereau of Standards，NBS）向社会公开征集制定加密算法标准。1977年NBS正式颁布DES [1]，该加密技术曾是世界上最为广泛使用的密码体制，奠定了现代分组密码的设计框架。DES是分组密码的一种，该加密技术中明文、密文和密钥的分组长度都为64比特，其加密算法和解密算法遵循相同的Feistel网络结构。

DES的加密算法如图2.5所示，该算法主要由初始置换（Initial Permutation, IP）、16轮Feistel乘积变换、逆初始置换三个阶段组成。每一轮Fesitel乘积变换都需要一个轮密钥，这些轮密钥由DES的密钥扩展子算法生成。



图2.5 DES加密算法

初始置换IP按照预定的置换表将64比特明文输入中的各个比特进行置换，例如将原数据块的第58位放到新数据块的第1位，该操作的目的是打乱明文输入的顺序。初始置换后将输出结果分为两部分，左边的32比特为$L\_{0}$，右边的32比特为$R\_{0}$，$L\_{0}$和$R\_{0}$将作为第一轮Feistel乘积变换的输入。

16轮Feistel乘积变换的迭代过程用公式可表示为：$L\_{i}$ = $R\_{i-1}$；$R\_{i}$ = $L\_{i-1}$ XOR F($R\_{i-1}$, $K\_{i}$)，即第i-1轮的右半部分$R\_{i-1}$会作为下一轮次左半部分$L\_{i}$的输入，第i-1轮左半部分$L\_{i-1}$与F函数的异或作为第i轮的右半部分$R\_{i}$的输入。其中F函数又可分为扩展置换E、密钥异或、S盒变换、置换P四个过程。首先32比特输入$R\_{i-1}$经扩展置换E后得到48比特输出，将输出与轮密钥$K\_{i}$异或后划分为8个6比特组，并行输入对应的8个S盒得到32比特输出，最后对输出进行置换P得到F函数的输出结果。其中S盒是整个DES算法的关键步骤，DES依靠其实现非线性变换以提供更好的安全性。

逆初始置换$IP^{-1}$为初始置换IP的逆过程，以与IP置换表相逆的规则对输入比特进行置换，例如此前初始置换IP将初始输入的数据分组的第58位放到新数据块的第1位，则逆初始置换$IP^{-1}$将初始输入的数据分组的第1位放回到新数据块的第58位。逆初始置换$IP^{-1}$的输出即为加密结果。

DES的密钥扩展子算法可分为置换选择1（Permuted Choice 1，PC-1）、左循环移位、置换选择2（Permuted Choice 2，PC-2）三个过程。其中，PC-1对输入的64比特密钥删掉8个奇偶校验位后进行置换得到左右两个28比特组，每一轮按照预定的位数对两个比特组进行左循环移位，移位结束后将比特组合并输入PC-2进行压缩置换，得到48比特的轮密钥。

DES是对合运算，解密过程与加密过程完全相同，解密时只需要将输入换为密文即可，此外，DES对明文输入进行充分的置换和非线性变换，实现了密码体制设计中的扩散和混淆原则。经研究发现，DES迭代8次后输入与输出便不相关，因而在当时具有极高的安全性，但由于DES密钥空间的规模太小，当今的计算能力可以在几小时内穷举完DES中64比特的密钥空间。为防止此类攻击，后续研究人员设计了三重DES，并推出了高级加密标准AES。值得一提的是，如果DES的密钥长度为128比特，其暴力破解复杂度将达到2128次操作，那么此时使用穷举攻击攻破DES大约需要$10^{18}$年。

### 2.2.4 高级加密标准（Advanced Encryption Standard，AES）

在2.2.3节中我们提到，由于DES密钥长度仅为64比特，无法有效抵御穷举密钥搜索攻击，因此学术界希望探索出更加安全的加密标准。1997年美国国家标准与技术研究院NIST(National Institute of Standards and Technology，NIST)向社会公开征集制定高级加密标准的活动。2000年NIST采用比利时密码学家Joan Daemen与Vincent Rijmen提交的Rijndael算法[2]确立为AES。

AES加密算法的基本流程如图2.6所示，其采用128比特的数据块和可选长度（包括128比特、192比特、256比特）的密钥块，加密过程主要由一个初始轮密钥加操作和轮基本轮函数迭代组成，其中轮数与明文长度和密钥长度有关，在明文和密钥长度皆为128比特的情况下为10。相应地，AES也使用密钥产生算法针对每轮迭代生成轮密钥。



图2.6 AES加密算法

AES以字节（8比特）为基本处理单元，其基础域是有限域GF($2^{8}$)，所有操作在该域上定义。明文、密钥和中间状态都按照字节的先后顺序从上到下，从左到右排列为一个字节块。在第一轮迭代之前，需要进行初始轮密钥加，即对明文和密钥字节块中的对应字节进行异或处理。AES加密过程除最后一轮外，其余迭代轮均包含4阶段操作：字节代换、行位移、列混合和轮密钥加，最后一轮不包含轮密钥加。轮函数中的四阶段操作使输入得到充分的扩散和混淆。

字节代换即为AES的S盒变换，是AES唯一的非线性变换。不同于DES中S盒的为6位输入4位输出，而AES中S盒是8位输入8位输出。AES中的S盒变换是一种非线性置换，字节代换首先在GF($2^{8}$)上求输入字节的逆，然后按照预定的系数矩阵对得到的结果进行仿射变换。

行移位按照预定的规则表对字节块的每一行进行循环移位，比如第0行不移位，第1行左移一个字节，第2行左移两个字节。行移位变换属于置换，是线性变换，本质在于把数据打乱重排，起到了扩散作用。

列混合将字节块中每一列看作GF($2^{8}$)上的多项式，并将该多项式乘以预置不可约多项式$3x^{3}+x^{2}+x+2$，然后再模以$x^{4}+1$。同样地，列混合也属于线性变换，它混淆了字节块的每一列。

轮密钥加的过程与初始轮密钥加一致，不同的是参与异或的密钥为由初始密钥所扩展出来的轮密钥。

AES的轮密钥生成算法以字（4个字节，32位）为处理单元，该算法可分为密钥扩展和密钥选择两个阶段。在密钥扩展阶段，算法生成一个一维字数组，该数组长度取决于用户的密钥长度。在扩展密钥数组中，用户密钥放在数组最前列，其余填充字经过一定的递归生成规则得到。在密钥选择阶段，算法根据明文长度依次从扩展密钥中选出轮密钥。

AES不是对合运算，其解密算法和加密算法不同，需要分别实现正向和逆向的操作（AES解密需要使用逆S盒、逆行移位、逆列混合），解密密钥扩展也需要进行额外处理。尽管如此，AES加解密算法的基本结构是相同的，将加密算法各变换过程换为其逆过程即可得到解密算法。

 研究表明，AES密码算法设计得非常好，它适于多种环境并且高效和方便，并且能够有效抵御差分攻击。但是AES也有自己的不足之处，存在密钥扩展的非线性程度不足、列混合的扩散度不够高、无法抵抗侧信道密码分析等问题。

### 2.2.5 运行保密模式

实际应用中，分组密码不可能只针对一个分组进行操作，更多时候待加密数据包含多个分组，在待加密数据长度不是分组长度整数倍的情况下，需要对其进行填充。为了减少重复分组或关联分组所带来的密码分析安全隐患，选择合适的运行保密模式尤为重要。

NIST定义了5种基本的运行保密模式：电码本模式（Electronic Code Book，ECB）、分组链接模式（Cipher Block Chaining，CBC）、密文反馈模式（Cipher Feedback，CFB）、输出反馈模式（Output Feedback，OFB）、计数器模式（Counter，CTR），以下讨论中分组密码的明文分组长度默认为n。

ECB模式是最简单的运行保密模式，它使用相同密钥对每个单独的分组进行加解密，同样的明文分组得到的密文是一致的。如果消息中含有固定结构或重复的明文，ECB模式可能帮助密码分析者找出这种关系，并为其提供分组代换或重排的机会。因此，ECB模式比较适合短消息加密，在加密长消息时往往易受到攻击。

CBC模式解决了ECB模式的安全问题（ECB模式中相同的明文块会生成相同的密文块）。在加密时，其输入是当前明文分组与前一密文分组的异或，这样消息中相同的明文经加密后便会产生不同的密文，从而隐藏消息结构。在解密时，解密出的分组需要与前一密文分组异或才能得到最终的明文分组，由于两个相邻分组的密切耦合，当某一密文分组出现比特错误时，差错会影响当前分组和下一分组的解密，但不会影响更后面的分组的解密。同时，对于CBC模式，产生第一个密文分组需要初始向量（IV）的参与，为提高安全性，IV也需要得到保护。由于CBC模式的错误传播有限且能够并行解密，适合长消息加密。

CFB模式将分组密码转换为自同步流密码，能够实现逐比特或逐字节加密，因此不需要对消息进行填充，同CBC模式一样，CFB模式的加密过程中，每个密文块都依赖于前一次加密的密文块和密钥，并且CFB模式维护了一个初始状态为初始向量（IV）的n比特移位寄存器，假设反馈分组长度为j，每次加密时首先使用分组密钥对移位寄存器的n位内容进行加密，然后取加密结果的最左j比特与j比特明文异或得到密文，最后将移位寄存器左移j比特并将密文送入移位寄存器的最右j比特，这一过程一直重复直到所有明文加密完成为止。由于密文之间的耦合，CFB模式也会有差错传播的问题，该模式除了用于加密外，还可以结合消息验证码用于认证。

OFB模式与CFB模式基本相同，两者不同的地方在于OFB模式是直接取移位寄存器的n位加密结果中的最左j比特作为密钥流反馈给下一加密阶段，在下一加密阶段，移位寄存器左移j比特并将上一阶段反馈的j比特密钥流移入最右j比特。与CFB模式相比，OFB模式中的比特错误不会得到传播，但是却更易受到篡改攻击。因此，CFB模式适合有扰信道，而不适合认证。

CTR模式是适合并行计算的运行保密模式，其维护了一个计数器counter，在加密时对每一分组依此将counter递增，并与密钥完成分组加密，然后将加密结果与明文分组异或得到密文输出。显然，CTR模式的加密过程和解密过程是一致的。该模式具备并行、加密预处理、密文分组随机访问、可证明安全等优点。

除了上述几种基本运行保密模式以外，还有许多模式可供选择，比如认证加密模式CMAC、并行认证模式PMAC、国产的快速认证模式CCTR等。

对称密码技术快速且高效，但是仍存在以下三类安全问题：

（1）密钥分发安全性

对称密码算法中通信双方必须提前预置对称密钥，即发送方和接收方之间必须提前共享对称密钥。由于对称密码学算法的安全性依赖于对称密钥，必须防止攻击者窃取对称密钥以确保对称密钥的安全性。但是，如何确保对称密钥在通信双方或者多方中安全传输较为困难。

（2）可否认安全性

对称密码学算法中双方或者多方共享一个对称密钥，可能导致某实体假冒其他实体发送数据。例如，若实体A用共享密钥加密消息发送给C，A可声称消息被B盗取密钥伪造，因加密密钥为双方共有，无法通过密码学证据证明A是唯一发送者。此外，由于缺乏第三方可验证的签名机制，接收方C也可能篡改消息后声称来自A。

（3）密钥管理难

在对称密码体制中，需要通信的双方或者多方需维护一个共同的密钥。例如若实体A需要与实体B、C、D、E通信，则需分别和B、C、D、E共享一个长期密钥，实体A侧需存储管理多个密钥。若实体进一步增加，则会导致密钥管理所需的硬件成本与访问控制复杂度剧增

综上所述，对称密码仍存在许多场景下存在安全和效率上的局限性。为解决上述问题，1976年Diffie和Hellman提出了公钥密码体制，这是密码学史上的一个重要里程碑。

## 2.3 公钥密码体制

公钥密码体制，也称为非对称密码体制或双钥密码体制，其基本思想是利用私钥和公钥两类密钥实现消息的加解密，其中私钥由用户自身安全存储，公钥可以公开传输。公钥密码体制对于对称密码体制的优势主要体现在以下三方面：首先，在密钥管理方面，对称密码体制要求通信双方或多方安全共享统一密钥，但是这在不可信信道中存在显著挑战；而公钥密码体制通过非对称密钥机制，使得发送方仅需将公钥公开传输给接收方，公钥可经任意渠道传输，即使在不可信信道中进行传输也不会影响系统的安全性。其次，公钥密码体制具有数字签名功能，由于每个实体拥有不同的公私钥对，当一个实体采用其私钥签名明文消息后，任何持有该实体公钥的其他实体可以利用该实体的公钥验证该消息来源的真实性以及完整性，从而很好地解决了对称密码体制中的可否认性问题。最后，在公钥密码体制中，多方实体间无需像对称密码体制那样共享多个长期密钥，仅需通信各方维护自身独立的密钥对，这使得密钥管理和维护的复杂性显著降低。

综上所述，公钥密码体制能够解决对称密码体制存在的诸多问题，但是公钥密码体制也存在其局限性。对称密码学算法仅需一些简单的迭代、异或、移位、置换等操作，耗时较少。而公钥密码学则一般需要密码学中的点乘、模指数等运算，耗时较多。



图2.7 公钥密码机制及分类

主流公钥密码体制主要基于学术界公认难以解决的数论问题之上，包括大整数分解难题、有限域离散对数问题，以及椭圆曲线上的离散对数问题等。NIST进一步将公钥密码体制归类为整数因子分解密码体制、离散对数密码体制以及椭圆曲线密码体制。新兴密码学分支还有对一些其他的困难问题进行了探索，例如格理论密码学中最短向量问题和最近向量问题等新型困难问题。此外，根据公钥的不同类型，现有的公钥密码体制也可以分为三大类：基于证书的密码体制、基于身份的密码机制以及基于无证书的密码体制，图2.7展示了当前公钥密码体制的类型。

### 2.3.1 基于大整数因子分解问题的密码体系

大整数因子分解难题一般是指将一个大合数因式分解为两个未知大素数是比较困难的。例如，给定一个合数n（一般是两个素数p和q的乘积），在不知道素数p和q任何信息的情况下，求解p和q是困难的。RSA算法是最典型的基于大合数分解难题的公钥密码算法。

1978年R.Rivest、A.Shamir和L.Adleman基于大整数因子分解问题构造出了RSA算法[3]。RSA算法是典型的公钥密码机制，也是目前理论上最为成熟完善的公钥密码机制。RSA算法主要包括RSA加密算法和RSA签名算法两大功能，其实现过程分别如下：

RSA加密算法：

|  |
| --- |
| (1) 选择两个保密的大素数p,q；(2) 计算n=pq；(3) 选择一个整数e使其满足1<e<(p-1)(q-1)且gcd(e,(p-1)(q-1))=1，公开公钥(n,e)；(4) 选择私钥d，使其满足de mod ((p-1)(q-1))≡1，安全存储私钥(n,d)；(5) 加密消息M：C≡Me mod n；(6) 解密密文C：M≡Cd mod n； |

RSA签名算法：

|  |
| --- |
| (1) 选择两个大素数p,q；(2) 计算n=pq；(3) 选择一个公钥e使其不是(p-1)(q-1)的因子，公钥(n,e)；(4) 选择私钥d，使得de mod ((p-1)(q-1)) ≡1，安全存储私钥(n,d)；(5) 签名h(M)：S≡h(M)d mod n；(6) 验签S：h(M) ≡Se mod n； |

RSA算法的安全性基于大整数分解问题，其核心逻辑表现为：攻击者需要得到模数n后分解出两个大素数p和q，进而计算出(p-1)(q-1)，最终从公钥(n,e)推导出私钥(n,d)。由于当前计算技术条件下从n分解出p和 q是困难的，所以RSA一般被视为是安全的。根据NIST标准，为匹配AES-128的等效安全强度，RSA模数n的推荐长度需达到3072比特。

自RSA诞生以来，针对其密码算法研究而产生的各种攻击技术层出不穷，例如穷举攻击、重复加密攻击、共模攻击和时序攻击等，这些攻击大多是从RSA的应用层面进行攻击，此类攻击通常针对协议设计或参数选择的漏洞（如短私钥、低熵素数或共享模数等非标准场景）。在理论攻击层面，尽管在1999年有学者通过数域筛法分解了长度为512比特的RSA模数n，但针对2048或3072比特RSA模数的有效分解仍无公开突破。

作为首个实用化的公钥密码体制，RSA凭借其较为完备的理论框架和功能的扩展性，被业界普遍认为的最优秀公钥密码之一，但是RSA存在两大不可忽视的缺陷：首先，由于筛选大素数的过程需要经过多轮验证，因此RSA的密钥生成效率也会受到限制，难以做到“一次一密”（这为密码分析攻击创造了条件）；其次，RSA算法的模数位长直接关系到算法的安全性，但较长的模数导致模幂运算的计算复杂度显著上升，这会增加机器的运算量而导致算法效率较低，而且随着大整数分解领域研究的不断推进，模数长度还在不断增加。

### 2.3.2 基于有限域的离散对数密码体制

现代公钥密码体系中，离散对数问题是有限域上最核心的数学难题之一。其严格定义为：设定一个大素数p，其有限域的乘法循环群G由生成元g生成，对于任意一个整数，计算较为容易，但是通过a反推唯一的k则具有超多项式的复杂度。ElGamal是最经典的基于离散对数困难问题的公钥密码体制之一，基于ElGamal机制的加密和签名算法如下：

ElGamal加密算法：

|  |
| --- |
| (1) 生成一个大素数p, 求p的本原根g；(2) 选择私钥x, 使得1<x<p-2，安全存储私钥x；(3) 计算y=gx mod p, 公开公钥(p,g,y)；(4) 加密消息M：选择一个随机数k，使得1<k<p-2且k与(p-1)互素，计算c1=gk mod p, c2=M yk mod p；(5) 解密密文(c1,c2)：M=c1-x c2 mod p； |

ElGamal签名算法：

|  |
| --- |
| (1) 生成一个大素数p, 求p的本原根g；(2) 选择私钥x, 使得1<x<p-2，安全存储私钥x；(3) 计算y=gx mod p, 公开公钥(p,g,y)；(4) 签名消息M：选择一个随机数k，使得1<k<p-2且k与(p-1)互素，计算r=gk mod p, s=(h(M)-xr) k-1 mod (p-1)；(5) 验证签名(r,s)：计算h(M)，验证yr rs=gh(M) mod p； |

ElGamal算法被GnuPG、OpenPGP等密码工具用于数学签名和密钥协商，但仍存在其局限性。根据NIST标准，ElGamal为达到AES-128的128位安全强度，推荐使用3072位素数域和256位私钥，但这会造成密文长度扩展为明文的两倍，显著增加通信开销。ElGamal的安全优势还体现在其与RSA的对比中，相较于RSA，ElGamal在相同安全等级下可选用更短的密钥，具有更高的安全性，然而ElGamal算法单次加密需进行两次模幂运算，解密需进行一次模幂与模逆运算，这导致它的运算速度要慢于RSA。

### 2.3.3 椭圆曲线密码体制(Elliptic Curve Cryptography，ECC)

椭圆曲线密码学最早由Miller[5]和Koblitz[6]分别在1985年和1987年提出其数学基础建立在上椭圆曲线点群的离散对数问题（Elliptic Curve Discrete Logarithm Problem, ECDLP）之上。定义在有限域GF(p)上的标准椭圆曲线的方程为：y2 = (x3 + ax + b)mod p，其中p为素数，a和b满足4a3 +27b2≠0 mod p确保曲线非奇异，满足该方程的整数的点集(x, y)即为椭圆曲线上的点，该点集与无穷远点共同构成加法交换群。有限域上的椭圆曲线的核心运算包括点加运算和点乘运算，点加运算指对于椭圆曲线上的任意两点P和Q通过几何切线法则可得到R=P+Q的坐标，点乘运算是指椭圆曲线上的点P乘以整数𝑘，即k•P= P + P + ... + P，其意义为计算k•P需要对点P执行k−1次点加运算，实际可通过双倍-累加算法优化为对数级运算次数。椭圆曲线密码学的安全性基于如下两个公认的难解问题：其一为ECDLP，即给定点P和整数k，根据等式Q = k•P，很容易计算出Q，但是，给定点P和Q，计算整数k是困难的；其二为椭圆曲线Diffie-Hellman问题（Elliptic Curve Diffie-Hellman Problem，ECDHP），即给定椭圆曲线上的点P，k1•P以及k2•P，计算k1k2•P是困难的，椭圆曲线密码体制的加密和数字签名功能算法如下。

基于离散对数问题的椭圆曲线加密算法：

|  |
| --- |
| (1) 设G是有限域 GF(p) 上的椭圆曲线 E 的一个生成元，其中 E 由所有满足其方程的点和一个无穷远点组成，且E 的循环子群 E1 由 G 生成，n为素数且为生成元G的阶；(2) 选择私钥d，使得0<d<n，安全存储私钥d；(3) 计算Q=d•G 公开公钥(Q,G)；(4) 加密消息M：选择一个随机数k，使得0<k<n，计算椭圆曲线点X1(x1,y1)=k•G，点X2(x2,y2)=k•Q，计算C=Mx2 mod n；(5) 解密密文(X1,C)：首先计算d•X1=d•(k•G)=k•Q=X2，然后利用X2计算明文 M=Cx2-1 mod n； |

基于离散对数问题的椭圆曲线签名算法：

|  |
| --- |
| (1) 设G是有限域 GF(p) 上的椭圆曲线 E 的一个生成元，其中 E 由所有满足其方程的点和一个无穷远点组成，且E 的循环子群 E1 由 G 生成，n为素数且为生成元G的阶；(2) 选择私钥d，使得0<d<n，安全存储私钥d；(3) 计算Q=d•G 公开公钥(Q,G)；(4) 签名消息M：选择一个随机数k，使得0<k<n，计算椭圆曲线点R(x,y)=k•G, s=k - h(M) •dmod n；(5) 验证签名(R,s)：首先计算s•G+h(M) •Q = k•G- h(M) •d•G +h(M) •Q = k•G，验证这里得到的椭圆曲线点是否等于收到的R； |

 根据NIST定义，为实现与AES-128同等的128比特安全强度，ECC仅需256比特密钥长度，而RSA算法的密钥长度需要3072比特，并且ECC密钥规模随安全等级提升而扩展的速率显著低于RSA（384位ECC≈7680位RSA）。此外，有性能测试结果显示，ECC运算操作的计算开销远小于RSA操作的计算开销。椭圆曲线密码体制具有安全性高、密钥长度短、通信负荷低以及计算效率高等优势，近年来受到密码学研究者们的广泛关注，并在理论和技术上取得了大量的研究成果。

### 2.3.4 基于证书的公钥密码体制

公钥密码体制的密钥管理基本思想为：实体的公钥对外公开，私钥由实体自身保管。当其他方采用该实体的公钥加密数据或者采用该实体公钥验证实体签名的场景下，确保该公钥确实属于该实体至关重要。基于证书的公钥密码体制通过引入可信第三方证书颁发机构（Certificate Authority，CA），解决公钥归属验证的核心安全问题。

在基于证书的公钥密码体制中，每个用户的公钥均都绑定一个合法第三方CA签发的数字证书。公钥证书中包括用户的身份信息、公钥参数以及CA的签名等。该体制的典型应用流程如图2.8所示，第三方签发证书给发送者，发送方使用私钥对消息签名后，将签名值与证书一并发送至接收方；接收方首先验证证书的有效性，验证成功后通过证书中的公钥验证签名。在此过程中，接收方通过验证证书的有效性核实公钥的合法性，而证书则是由第三方CA颁发，因此该机制的安全性高度依赖CA的可信性。一旦CA恶意签发证书，则将导致整个网络的信任链崩塌。此外，证书生命周期管理也存在显著挑战，CA需维护证书吊销列表和在线状态协议以应对密钥泄露场景，但是吊销和恢复证书都需要一定的处理时间，这段传播延迟可能形成安全盲区。从性能角度上看，公钥证书的传递会额外占用通信资源，且公钥证书的验证耗费较多的计算开销，因此基于证书的公钥密码体制并不适用于资源受限的环境。



图2.8 CA签发公钥证书

### 2.3.5 基于身份的公钥密码体制（Identity-based Public Key Cryptography）

为了解决传统证书体制中密钥管理的复杂性，1984年，Adi Shamir提出了基于身份的公钥密码体制[7]。在该系统中，实体公钥与用户的身份信息绑定，其中用户的身份信息为身份标识信息、地址信息或者邮箱信息等具有代表性意义的信息。例如，某一通信系统中实体的公钥为用户的身份标识，当其他实体需要获取该实体的公钥时，只需获取其身份标识即可，无需额外的身份证书。而实体私钥通过相应的私钥生成中心(Privacy Key Generator，PKG) 基于主密钥与用户身份生成，并且通过安全信道发送给实体。

该机制的签名场景如图2.9所示，签名流程中发送方将身份信息发送给PKG，PKG利用身份信息和主密钥导出私钥并安全传输给发送方，随后发送方采用私钥签名消息发送给验签方。验签方仅需利用签名方公开的身份信息验证签名即可。该机制的加解密场景如图2.10所示，加密方通过身份生成公钥并加密数据，接收方向PKG申请私钥解密，全程无需公钥交换协议。



图2.9 基于PKG的签名和验签过程



图2.10 基于PKG的加密和解密过程

然而，基于身份的公钥密码体制仍存在核心缺陷：用户的私钥均是由PKG生成，即PKG掌握所有相关用户的私钥，一旦PKG被攻陷，则所有用户的私钥都将泄漏。此外，基于身份的公钥密码体制依赖双线性对或复杂椭圆曲线运算，单次签名验证耗时可达毫秒级，显著高于传统RSA算法，计算开销较大的特性使其难以适配于资源受限的设备。

### 2.3.6 无证书公钥密码体制

为消除基于身份密码体制中的私钥托管缺陷，2003年AlRiyami等人提出了无证书公钥密码体制[8]的概念，其核心机制引入了一个可信的第三方密钥生成中心（Key Generation Center，KGC）。KGC拥有系统的主密钥，并负责为用户提供和部分私钥，用户结合来自KGC的私钥与本地生成的随机参数，通过特定算法合成完整密钥对。用户公钥则由身份标识与本地参数直接构造（如设备ID+随机数哈希），无需证书签名或显式身份绑定。

与基于身份的公钥密码体制仍相比，无证书公钥密码体制中的用户私钥来自于自身和KGC，即使KGC被攻陷，用户终端的完整密钥也不会暴露给攻击者。与证书密码体制相比，用户将身份作为公钥的一部分信息，不需要证书作为载体，省去了证书颁发、存储及链式认证的过程，减少了通信开销和存储开销。此外，根据Girault的KGC可信度分级模型，基于身份的公钥密码体制中由于KGC完全掌握用户私钥，可以任意假冒用户或解密用户隐私信息因此可信度仅达到level1；基于证书的公钥密码体制中用户私钥由用户自身生成，用户可以通过证书检测出KGC的假冒攻击，从而该KGC达到最高可信度等级level3；无证书公钥密码机制中KGC未完全掌握用户私钥，虽然攻击者可以利用原公钥任意构造其他公钥并掌握相应私钥进行假冒攻击，但是用户可自主检测KGC伪造行为，故该KGC可信度也可达到Level3。

无证书公钥密码体制也存在其不足，由于没有第三方对公钥进行验证，也无法通过容易记忆的手机号、邮箱等身份标识直观验证，攻击者可以发起公钥替换，接收方只有当验签失败或解密异常后，才能发现验签的公钥或加密的公钥异常，并且该场景下用户无法区分是密钥伪造还是数据本身错误，形成归因不确定性风险。例如，用户验签失败可能是攻击者替换公钥后伪造签名，也可能是签名本身无效所致。

综上所述，无证书体制公钥管理轻便、KGC高安全可信的优势，无证书体制适用于海量设备接入，要求低带宽和存储开销，低功耗低时延的应用比如工业互联网或车联网，物联网等场景。

表2.1：三种不同公钥密码体制的优缺点对比

|  |  |  |  |
| --- | --- | --- | --- |
|  密码体制评判因素 | 基于证书的公钥密码体制 | 基于身份的公钥密码体制 | 基于无证书的公钥密码体制 |
| 证书 | 需要 | 不需要 | 不需要 |
| 可信机构 | CA | PKG | KGC |
| 存在问题 | 证书管理 | 密钥托管 | 公钥替换 |
| 密钥用途 | 公钥用于加密私钥用于解密 | 公钥用于加密私钥用于解密 | 公钥用于加密私钥用于解密 |
| 密钥生成过程 | CA生成 | PKG基于身份信息生成 | 实体和KGC各自生成一部分 |
| 公钥获取方式 | 获取证书 | 获取身份信息 | 获取部分公钥 |

 基于证书的密码体制、基于身份的密码体制以及基于无证书的密码体制的三种密码体制优缺点已总结在表2.1中。

## 2.4 Hash函数和消息认证码

哈希函数通过固定长度输出与抗碰撞性为数据生成不可逆的“数字指纹”，而消息认证码依托共享密钥实现消息来源验证与完整性保护，两者共同构筑了数据可信传输的基础层。

### 2.4.1哈希函数

哈希函数又称单向散列函数，其作为作为密码学核心基础组件之一，可以将任意长度报文转换为固定长度的输出，目前广泛应用于各种安全应用和网络协议中。典型的哈希函数包括MD5（128位输出）、SHA-1（160位输出）、SHA-2系列（224/256/284/512位输出）等。该类函数H(x)具有四大核心特征：

（1）输出长度固定

哈希函数中输入长度无论是1字节还是TB级，输出长度始终为预先设置的固定值，例如哈希函数MD5恒输出128位数据，SHA-1恒输出为160位数据，输出长度固定的特性位数据指纹的生成提供了标准化基础。

（2）单向不可逆

哈希函数的计算过程是单向不可逆的，即正向计算H(x)的值较为容易，但是反向通过H(x)直接算出x的值在计算上不可行。除非输入空间极小，则攻击者可通过暴力破解的方式（如彩虹表）获得值x。此外，理想哈希函数的输出应确保不会泄露任何关于x的统计特征。

（3）抗碰撞性

哈希函数的抗碰撞性可分为抗弱碰撞性和抗强碰撞性。抗弱碰撞性是指给定一个值x，找到值y使得H（x）= H（y）在计算上不可行；抗强碰撞性是指构造任意两个不相同的值x和y，使得H（x）= H（y）在计算上不可行。尽管从理论上碰撞必然存在（哈希函数的输入空间要远大于输出空间），但实际寻找碰撞的计算复杂度超过2n/2，对于输出长度较大的哈希函数依旧是计算不可行的，例如例如SHA-256需执行2128次运算才能找到碰撞。然而，哈希函数的抗碰撞特性也不是绝对安全的，例如MD5之前被证明是抗碰撞的，但王小云教授在2004年提出了有效的碰撞攻击方法，将其碰撞复杂度从不可计算的理论值264降到了实际可计算的224[9]。

（4）可计算性

哈希函数中，不论输入长度值x为多少，计算哈希函数H(x)都比较容易，即便处理TB级数据仍能保持线性时间复杂度。

目前针对哈希函数的攻击思路主要有两类：穷举攻击和密码分析。前者与特定算法无关，只取决于哈希的输出长度，例如2004年我国密码学家王小云就利用改进后的生日攻击攻破了MD5哈希算法；后者则依赖于相应的哈希函数算法细节，利用算法设计缺陷进行攻击，例如Google团队于2017年通过差分路径构造手段破解了SHA-1哈希算法[10]。

此外，哈希函数对输入非常敏感，其雪崩效应确保输入数据的微小变化（即使是单比特不同）也可导致输出的全局改变。正是因为哈希函数优越的压缩、抗碰撞和错误检测特性，使其可应用于消息认证码、数字签名、伪随机数生成等安全领域。

### 2.4.2消息认证码

消息认证码（Message Authentication Code，MAC）是用来确保消息合法性以及数据完整性的密码学工具，其通过生成算法产生通信双方共享的密钥，并使用该密钥对任意长度输入值进行处理，从而得到一个固定长度的认证标签，这个认证标签即为MAC，MAC的核心功能体现在以下两个方面：

（1）消息完整性：接收方通过重新计算MAC并与接收到的MAC进行比对，可以检测消息的完整性；

（2）消息合法性：密钥仅有合法通信方拥有，通过MAC可以确保消息来自合法的实体。

综合以上两点性质，可以将MAC视作为具备消息认证能力的哈希值。举例来说，通信双方Alice和Bob共享一个密钥K，为确保Alice发给Bob的消息M是合法性的，Alice使用共享密钥K计算消息M的MAC并将MAC附加于消息后发送给Bob，而Bob则可以通过相同通信密钥K验证MAC确保消息来自于合法的实体，同时确保消息完整性。

虽然MAC和哈希函数都具备类似的不可逆性与抗碰撞性，但实现MAC的方式更加多样，例如基于分组密码的MAC、基于流密码的MAC、基于公钥密码的MAC以及基于哈希函数的MAC。其中基于哈希函数的MAC因为兼容哈希算法而成为最常用的消息认证码，被称为HMAC。

然而MAC也存在其缺陷：首先，Alice和Bob必须提前安全共享密钥K，而这在大规模部署密钥管理的情况下较难实现（复杂度呈指数级增长）。其次，由于Alice和Bob同时共享密钥K，Alice可以否认其发送过消息M，存在抵赖的风险。最后，共享密钥只被Alice和Bob获悉，外部验证者无法确认消息来源于Alice还是Bob，限制了MAC在跨机构协作场景下的应用。数字签名技术通过非对称密钥机制可有效弥补上述缺陷。

## 2.5 数字签名

数字签名在哈希函数与MAC的基础上，通过非对称密钥机制实现了数据完整性与来源真实性的保障，实体私钥的存在赋予数字签名不可伪造与不可否认的特性。从简单的消息认证到合同签署、政务审批等强信任场景，数字签名以公私钥分离的机制规避了传统认证技术的安全缺陷，成为网络空间中身份与责任绑定的核心工具。

### 2.5.1 数字签名算法

在我们现实生活中，通常采用手写签名代表签名者的合法性，而在网络空间中无法实现手写签名，为模拟现实世界的手写签名功能，数字签名应运而生。数字签名常用于网络空间的身份认证，本质是通过密码学算法模拟现实中的手写签名功能，从而对某些消息进行签名，非签名方无法产生与签名方相同的签名，即攻击者无法伪造签名且签名方也无法否认已经签名过的消息。具体地，数字签名与手书签名的区别主要体现在以下两个方面[11]：

第一，两类签名的生成机制不同。由于每个人的笔迹有所不同，手书签名依赖签名者个体生物特征（如笔记力度，字形结构等）核验签名者身份；数字签名是通过某种非对称加密算法生成的二进制字符串进行签名，其合法性通过公开的验证算法来验证。出现争端时，它可通过数学证明为仲裁者提供无法抵赖的证据，其安全性远远高于主观判断的手写签名。

第二，两类签名的防复制特性不同。手书签名因其物理介质难以被精确复制；而数字签名正好相反，可被无限重复复制（例如重放攻击）。因此一般要求消息本身带有诸如时间戳、随机数等动态参数以防止签名被重复使用。此外，数字签名依托于计算机网络，其时效性也远远大于手写签名。随着计算机信息网络的迅速发展，数字签名的作用越来越重要，过去依赖于手书签名的大多业务可用数字签名来替代，是防止信息欺诈行为的重要保障。

数字签名的技术框架由Diffie和Hellman在1976年首次提出[12]，其标准流程如如图2.11所示：发送者使用哈希函数（例如SHA-256）生成消息摘要，并利用私钥加密生成签名，与原文一起发送给接收者。接收者使用发送者的公钥解密签名获取哈希值，然后使用相同的Hash函数对收到的原文独立计算哈希值，再与解密得到的哈希值对比。如果二者相同，则验证成功。这一标准流程可实现三类功能：

（1）消息完整性：如果接收方哈希值匹配成功，则说明收到的信息是完整的，在传输过程中没有被修改。

（2）认证性：由于验证签名过程中会采用发送者私钥签名，公钥验证，一般情况下，仅私钥持有者可生成有效签名，攻击者没有发送方私钥不可能产生有效的签名，该特性能够抵御伪造签名攻击和中间人攻击。

（3）不可否认性：由于接收方会采用发送方特定的公钥进行验证，一旦验证成功，则说明该消息必定是发送方发送的，即确保发送方无法抵赖已签署内容。



图2.11 数字签名工作流程

### 2.5.2 数字签名种类

随着数字签名技术的发展与应用场景的多样化，学术界研究出更多的签名方案以适配不同场景的安全与效率需求。

（1）代理签名

1996年Mambo等人提出了代理签名的概念[13]。其核心机制允许原始签名人在不泄露其私钥的情况下将其签名能力委托给代理方（例如企业高管授权法务代表签署合同）。代理方有权使用派生密钥为代表原始签名人生成签名，接收方在验证时需同时校验原始签名人的授权凭证与代理签名的有效性。代理签名的出现带来了极大的便捷，实现权责分离与权限追溯的功能，但需防止代理方权限滥用的情况。

（2）聚合签名

2003年，Boneh等人最早提出了聚合签名的概念[14]。聚合签名本质上对多用户签名进行聚合压缩并利用双线性对或格密码技术进行批量验证的签名方案。假设现有n个用户，其核心机制如下：首先，每个用户利用其私钥签名消息Mi并输出签名𝛿i；随后，将n个用户的签名𝛿i（i=1,…,n）以及身份信息等输出压缩为单个固定长度签名𝛿；最后，直接通过进行一次双线性对运算检查聚合签名𝛿的有效性以认证这𝑛个用户。显而易见，采用聚合签名机制可以有效降低通信开销和计算开销，非常适用于计算资源有限且需要同时验证大量签名的场景，例如海量物联网设备并发接入认证场景等。

（3）多重签名

1983年，Itakura和Nakamura首次提出了多重签名算法的概念[15]。在多重签名中，所有参与者对同一消息生成联合签名，其签名长度与单个签名长度相同，而验证方则可以仅用一次验证就实现了对所有参与者签名的验证，验证复杂度仅为O（1）。该签名方案可以有效降低网络资源消耗，常适用于跨国贸易多方合同签署、电子商务等场景。

（4）盲签名

1983年，Chaum首次提出了盲签名的概念[16]，该签名方案引入了盲化因子对原始消息进行混淆，允许签名者不知道签署信息的情况下对消息进行签名。该签名方案的核心机制如下：首先发送方将消息盲化使得签名者无法获取消息内容；随后让签名者对盲化后的消息进行签名；最后接收者则需通过去盲操作获取有效签名并进行验证。该签名方案主要应用于电子商务和电子投票等领域（如电子投票中选票内容对计票机构不可见）。

（5）群签名

1991年，Chaum和van Heyst等人提出了群签名的概念[17]。群签名构建了一个群体匿名体系，任何一个群成员都可以使用群主颁发的证书代表整个群组生成签名，且不会泄露个人身份信息，验证者仅需群公钥即可核验。当存在争议时，管理员可以通过追踪密钥揭露签名者身份。群签名在电子选举、电子投票、电子现金系统等方面存在广泛的应用前景（如金融审计中追踪异常交易源）。

### 2.5.3 椭圆曲线数字签名算法

椭圆曲线数字签名算法（Elliptic Curve Digital Signature Algorithm，ECDSA）由Johnson与Menezes于1999年系统化提出，其通过ECDLP构建非对称签名体系，是首个标准化椭圆曲线数字签名方案。ECDSA于1998年纳入ISO/IEC 14888-3标准，并在1999年作为ANSI X9.62标准被采纳，并于2000相继被IEEE 1363-2000与FIPS 186-4标准采纳，其签名和验签流程如下。

ECDSA签名算法：

|  |
| --- |
| 1. 选择一个随机数d作为私钥，计算D=d•P作为公钥；
2. 选择一个随机数k，计算K=k•P=(x,y)作为公钥；
3. 令 r = x mod n，如果r = 0，则返回步骤1；
4. 计算 M的哈希值得到H = Hash(M)，将H转化为一个big endian的整数e；
5. 计算s = k-1 (e + r d) mod n；
6. 输出的S =(r, s)即为签名；
 |

ECDSA验签算法：

|  |
| --- |
| 1. 计算A = s-1 e mod n；
2. 计算B = s-1 r mod n；
3. 计算P = A•P + B•D=(a,b)；
4. 如果r == a mod n，则验证正确；
 |

ECDSA算法是基于椭圆曲线上的离散对数困难问题，即使攻击者获取公钥D，也无法在多项式时间内推导出私钥d，除非攻击者已经攻破了椭圆曲线密码体系。相较于RSA签名，ECDSA在同等安全强度下密钥长度大幅缩减（如256位ECC ≈ 3072位RSA），签名速度提升3-5倍，但由于需要两次标量乘与一次点加运算，ECDSA验签效率略低。

### 2.5.4 代理签名算法

代理签名算法赋予被指定的代理签名者代表原始签名者生成有效签名的能力，该签名算法根据授权方式可分为完全委派代理签名、部分委派代理签名、授权委派代理签名三类。

在完全委派代理签名中，原始签名者直接将自己的签名密钥通过安全信道直接发送给代理签名者，这使得代理签名者生成的签名同原始签名者是一致的。完全委派代理签名虽然简单、高效，但是由于验签方无法鉴别代理签名者和原始签名者，可能会出现签名滥用、身份冒用等安全问题。

在部分委派代理签名中，原始签名者通过自己的签名密钥导出代理专用签名密钥给代理签名者，并相应地修改了验签算法，验签方只需使用原始签名者公钥即可核验代理签名的合法性。并且每个代理签名密钥不同，确保了每个代理签名具备唯一性，这就避免了完全委派代理签名存在的签名滥用、身份冒用等安全隐患。特别地，部分委派代理签名具有如下属性：

1. 不可伪造性：仅原始签名者与被授权代理可生成有效签名；
2. 可验证性：验证过程自动确认原始签名者的授权意图；
3. 可区分性：验签方能够在多项式时间内区分出代理签名和原始签名。
4. 不可否认性：代理签名者无法否认自己签名过的消息。
5. 可识别性：原始签名者可以根据代理签名判断代理签名者身份。

显然，部分委派代理签名通过密码学约束实现细粒度权限控制，比完全委派

代理签名更安全，也更灵活，成为政务多级审批等场景的主流方案。

授权委派代理签名中引入了数字证书以强化身份绑定，原始签名者签发证书以标识代理签名者的合法性，具体来分为两种授权委派方案：第一种方案中，证书主体包括代理公钥和原始签名者针对代理签名者公钥的签名，验签方能通过证书了解到该公钥获得了原始签名者的授权，从而完成对代理签名者合法性的验证。但该方案中验签方需要同时知道原始签名者和代理签名者的公钥；第二种签名方案中原始签名者生成临时密钥对，将新公钥的授权证书主体和新公钥所对应的私钥通过安全信道发送给代理签名者，通过这种方案验签方只需掌握原始签名者公钥即可。授权委派代理签名的优势在于其可以通过证书约束限制代理签名者所签发消息的类型，有效期限以及签署次数，同时保持与标准验签算法的兼容性，一种简单的基于离散对数困难问题的代理签名算法（部分委派）[13]如下：

|  |
| --- |
| 1. 基本参数。设s是原始签名者的私钥，v= gs mod p 是原始签名者的公钥。其中p是一个大素数，g是该素数域的生成元；
2. 代理密钥生成。原始签名者从1到p-2之间选一个随机数k，计算K= gk mod p，a=s+kK mod p-1；
3. 代理密钥传输。原始签名者通过安全信道将（a，K）传给代理签名者；
4. 代理密钥验证。代理签名者判断等式ga = v KK mod p 是否成立，若成立则验证通过；

(5) 代理签名。代理签名者使用a代替原始签名者的私钥s对消息进行签名，并将参数K随消息和签名一同发送给验签方；(6) 代理签名验证。验签方计算公钥v KK mod p，并基于此公钥进行验签； |

代理签名算法的选取通常依赖于用户对安全性的要求、消息长度、签名者和验签方的计算能力等因素，在实际应用中，还需考虑配套代理权限的撤销机制以构建生命周期式的安全管理体系。

### 2.5.5 聚合签名算法

聚合签名算法作为数字签名算法的扩展变种，其核心机制是将任意多个用户针对不同消息独立的签名聚合在一起完成验签。具体来说，给定n个不同用户针对不同消息的签名，聚合签名可以将这n个签名聚合成一个单独的短签名，该短签名可以让验签方相信这n个不同用户确实针对n个不同消息进行了签名，同时保障数据完整性、身份认证与不可否认性。相较于传统签名方案，聚合签名对n个签名进行了合并，因而减少了签名所需的存储空间、降低了网络传输所需的带宽、简化了验签过程的工作量，在公钥基础设施、安全路由协议等领域得到了广泛应用。Boneh等人最早提出了一种基于双线性配对的聚合签名算法[14]，其具体过程如下：

|  |
| --- |
| 1. 基本参数。$G\_{1}$和$G\_{2}$是以大素数p为阶的两个乘法循环群，$g\_{1}$和$g\_{2}$分别是这两个群的生成元，e为定义于$G\_{1}$和$G\_{2}$上的双线性配对；
2. 密钥生成。每个特定的用户从0到p-1之间选择一个随机数x作为私钥，并计算公钥v=$g\_{2}$x ，v是$G\_{2}$上的元素；
3. 消息签名。每个特定的用户针对特定的消息M，计算位于$G\_{1}$上的哈希值h = H(M)，并生成签名a=h x；
4. 消息验签。验签方判断等式e(a,$ g\_{2}$)=e(h,v)是否成立，若成立则验证通过；
5. 签名聚合。聚合签名者对n个待聚合签名作累积得到位于$G\_{1}$上的$a\_{n}$；
6. 聚合验证。聚合签名验签方判断e($a\_{n}$,$ g\_{2}$) 是否等于n个用户的e(h,v)的乘积，若成立则聚合验证通过；
 |

 上述聚合签名算法是一种无序聚合签名，即各签名能以任意顺序进行聚合，但是该方案存在两大局限：所有签名消息必须互不相同，并且双线性对计算开销大，制约了其在高吞吐场景下的应用。针对无序聚合签名的局限性，聚合签名衍生出有序聚合分支，在该签名算法中各签名只能以既定的顺序进行聚合，验签时需严格遵循该顺序执行链式验证。此类算法比较适合公钥基础设施、安全路由协议等具有严格顺序关系的应用。

### 2.5.6 多重签名算法

多重签名是一类特殊的聚合签名，后者针对多个用户的不同消息进行签名，前者则是多个用户共同完成对同一消息的签名，且不论参与用户个数，多重签名长度始终与单个签名长度保持一致。验签方可以一次性确认各用户是否对此消息完成签名，且各签名用户不能否认对于该消息的签名，确保不可否认性和细粒度认证。根据用户签名生成流程的协作模式，多重签名可以划分为串行有序多重签名，广播并行多重签名以及结构化多重签名三类。在串行有序签名中，参与者依次签名，适用于政务系统层级审批等场景；在广播并行签名中，各参与者独立生成签名片段后聚合，适用于分布式系统异步通信等场景；结构化签名中基于预定义的逻辑拓扑，例如阈值签名需要至少k个参与者协作才能产生有效签名。

 Sangjoon等人提出了一种基于RSA公钥密码体制的多重密码算法[18]，其具体过程如下：

|  |
| --- |
| (1) 基本参数。设用户i的RSA模底数为ni，公钥为$ei，$私钥为di，待签名消息为M；(2) 首用户签名。第一个用户计算签名$S\_{1}$=(2h(M)+1) n1 mod 2n1，其中h(M)是对消息求哈希，得到签名后将其与消息M发送给第二个用户；(3) 后续用户签名。后续第i个用户计算签名$S\_{i}=S\_{i-1}$di mod 2 li ni，其中li是从ni推导出来的重要参数。得到签名后将其与消息M发送给后续用户；(4) 验证多重签名。验签方按照公式$S\_{i-1}=S\_{i}$ei mod 2 li ni从最后的签名往回推导，直至推导出签名$S\_{1}$，并验证等式2h(M)+1=$S\_{1}$ e1 mod 2n1是否成立。若成立则多重签名验证通过； |

 总的来说，多重签名作为一种对多个用户同时作身份认证的高效验签方式，在分布式应用、电子商务、财产分割等领域具有优秀的应用价值。特别是近年来随着比特币等数字货币的崛起，适用于分布式系统安全的多重签名技术也引起了广泛的关注和研究。

### 参考文献

[1] Feistel, Horst, Cryptography and computer privacy, Scientific American 228(5): 15-23, 1973.

[2] Daemen, Joan, Vincent Rijmen, AES proposal: Rijndael, 1999.

[3] Rivest, Ronald L, Adi Shamir, Leonard Adleman, A method for obtaining digital signatures and public-key cryptosystems, Communications of the ACM, 21(2): 120-126, 1978.

[4] ElGamal, Taher, A public key cryptosystem and a signature scheme based on discrete logarithms, IEEE transactions on information theory, 31(4): 469-472, 1985.

[5] Miller, Victor, Use of elliptic curves in cryptography, Springer, 1986.

[6] Koblitz, Neal, Elliptic curve cryptosystems, Mathematics of computation, 48(177): 203-209, 1987.

[7] Shamir Adi, Identity-based cryptosystems and signature schemes, Workshop on the theory and application of cryptographic techniques, 1984, pp.47-53.

[8] Al-Riyami, Sattam, Kenneth G. Paterson, Certificateless public key cryptography, Asiacrypt, 2003, pp. 452-473.

[9] Wan Xiaoyun, Dengguo Feng, Xuejia Lai, Hongbo Yu, Collisions for hash functions MD4, MD5, HAVAL-128 and RIPEMD, Cryptology EPrint Archive, 2004.

[10] Stevens, Marc, Elie Bursztein, Pierre Karpman, Ange Albertini, Yarik Markov, The first collision for full SHA-1, Cryptology–CRYPTO, 2017, pp. 570-596.

[11] 现代密码学（第4版）杨波编著, 2017.

[12] Hellman, Martin, New directions in cryptography, IEEE transactions on Information Theory 2(6): 644-654,1976.

[13] Masahiro Mambo, Keisuke Usuda, Eiji Okamoto, Proxy signatures for delegating signing operation, ACM conference on Computer and communications security , 1996, pp.48-57.

[14] Dan Boneh , Craig Gentry ,Ben Lynn ,Hovav Shacham, Aggregate and Verifiably Encrypted Signatures from Bilinear Maps , International conference on the theory and applications of cryptographic techniques, 2003, pp.416-432.

[15] Itakura, Kazuharu, Katsuhiro Nakamura, A public-key cryptosystem suitable for digital multisignatures, NEC Research & Development, 71(71):474-480, 1983.

[16] Chaum David, Blind signatures for untraceable payments, Advances in cryptology, Springer, 1983, pp.199-203.

[17] Chaum, David, Eugène van Heys, Group signatures, Workshop on the Theory and Application of of Cryptographic Techniques, Springer, 1991, pp.257-265.

[18] Sangjoon Park, Sangwoo Park, Kwangjo Kim Dongho Won, Two efficient RSA multisignature schemes, International Conference on Information and Communications Security, 1997, pp.217-222.