第3章

区块链核心密码学技术

本章学习目标

- 了解哈希函数的来源以及特征，掌握 SHA256 算法的原理。
- 了解 Merkle 树的概念及其在区块链中的应用，掌握 Merkle 树的构造及检索过程。
- 了解常见的数据加密技术及其日常应用，掌握相关原理。
- 了解三大数字签名算法的原理及优缺点，具备应用相关算法解决实际问题的能力。
- 了解 PKI 的基本概念，掌握 PKI 关键技术的应用。

3.1 哈希算法

哈希算法作为区块链的主要技术之一，在区块连接成链、数据加密等多个方面被广泛应用。

3.1.1 哈希函数简介

1. 定义

哈希函数是哈希算法的实现，是一种公开函数，将任意长度数据 m 创建为较短的、固定长度的数字“指纹”，函数表示为 $H(m)=h$。其中，$H(\ )$即为哈希函数，又称为散列函数、杂凑函数，h 为哈希值，即数据 m 对应的数字“指纹”。

2. 特点

哈希函数具有如下 5 种重要的特性。

（1）输入长度可变，输出长度固定。对于任何长度的输入数据块，哈希函数的输出值长度固定。例如，SHA256 哈希函数，输入值可为任意长度的数据，输出值为 256 位的 0、1 组合。

（2）正向计算速度比较快。对任意数据 m，可在有限时间内计算出 h，使得 $h=H(m)$，且计算速度较快，这个特性能够保证加密与验证的速度。

（3）单向性。对任意给定的哈希值 h，没有办法反推算出 m，使得 $H(m)=h$，即逆向困难。

（4）防碰撞性。对任意给定的 m，找到一个 $n(n\neq m)$，使得 $H(m)=H(n)$，在计算上不可行；找到任意一对不同的 m 与 n，使得满足 $H(m)=H(n)$，在计算上也是不可行的。

（5）谜题友好性。对于 $h=H(m)$，无法从输入数据 m 判断出输出的结果，即无法通过控制输入值 m 获取想要的输出值 h。

3. 应用现状

哈希函数在公共密钥密码学中占据重要地位，在信息安全领域有着非常广泛和重要的应用，例如，认证算法构造、口令保护、比特承诺协议、随机数生成及数字签名算法等。其中，MD(Message Digest)系列密码算法和 SHA(Secure Hash Algorithm)系列密码算法以及国内的 SM3 算法是哈希算法中最著名、最具代表性的算法。在实际应用中，数字摘要、哈希攻击与防护是哈希算法的重要应用领域。2004 年，密码学专家王小云教授和她的团队，将 MD4 算法的碰撞攻击复杂度时间降到手工可计算，将寻找 MD5 算法实际碰撞的复杂度和 SHA-1 算法碰撞的理论复杂度分别降为 2^{39} 和 2^{63}，这使得众多学者开始怀疑 MD 系列算法和 SHA 系列算法的安全强度，逐渐开展研究新的哈希算法，寻找安全程度更高的哈希算法。

3.1.2 SHA256 算法

SHA 算法是一个密码散列函数家族，由美国国家安全局设计，并由美国国家标准技术研究所(NIST)发布为联邦数据处理标准(FIPS)。

1. 简介

SHA 算法又称为安全散列算法(Secure Hash Algorithm)，SHA 家族可分为 SHA-1 和 SHA-2 两大类。其中，SHA-1 在许多安全协定中被广泛使用，曾被作为 MD5 算法的后继者，但其安全性如今被广大密码学专家质疑；SHA-2 又包括 SHA-224、SHA-256、SHA-384 和 SHA-512 四类，对应输出结果的长度分别为 224 位、256 位、384 位、512 位。

作为哈希算法家族的一个成员，SHA 算法同样具备哈希算法的特性。比特币采用 SHA256 算法，该算法属于 SHA-2 系列，在中本聪发明比特币时被公认为是最安全、最先进的算法之一。比特币系统在生成地址中有一个环节使用了 REPID-160 算法，其他涉及哈希运算的地方使用 SHA256。

对于任意长度的输入值消息，经 SHA256 算法计算均会生成一个 256 位的二进制输

出值，即为哈希值，也称作消息摘要，输出结果的空间范围为 2^{256}，约为 $1.157\ 92\times10^{77}$。例如，将“BlockChain”作为输入值，经过哈希函数 SHA256 计算之后，得到的哈希值为“3a6fed5fc11392b3ee9f81caf017b48640d7458766a8eb0382899a605b41f2b9”。已知原输入值数据及其输出值，想找到一个具有相同输出值的数据（即伪造数据）非常困难，只能依靠暴力破解，这就是比特币挖矿需要消耗算力的根本原因。根据输入数据，通过 SHA256 计算出输出值非常容易，且对输入值数据进行任何改动，所得到的输出值均有较大区别。

2. SHA256 算法描述

SHA256 算法的消息摘要为 256 位，相当于一个长度为 32B 的二进制数组，通常用一个长度为 64 的十六进制字符串表示，其中，1B 的长度为 8 位，一个十六进制的字符长度为 4 位。

SHA256 加密过程包括以下四个步骤。

步骤 1：设置初始化常量：8 个哈希初值，64 个哈希常量。

（1）对自然数中前 8 个质数（2，3，5，7，11，13，17，19）平方根的小数部分，取前 32 位，生成 8 个哈希初值。

$H_0^{(0)}=0$x6a09e667

$H_1^{(0)}=0$xbb67ae85

$H_2^{(0)}=0$x3c6ef372

$H_3^{(0)}=0$xa54ff53a

$H_4^{(0)}=0$x510e527f

$H_5^{(0)}=0$x9b05688c

$H_6^{(0)}=0$x1f83d9ab

$H_7^{(0)}=0$x5be0cd19

（2）对自然数中前 64 个质数（2，3，5，7，11，13，17，19，23，29，31，37，41，43，47，53，59，61，67，71，73，79，83，89，97，…）立方根的小数部分，取前 32 位，生成 64 个哈希常量，记为 K_0，K_1，…，K_{63}。

步骤 2：对输入消息进行补位，补位后的长度为 512 位的倍数。

（1）按式（3-1）计算补位长度 L_{f1}。

$$(L_{m0}+L_{f1}+L_{f2})\bmod 512\equiv 0 \tag{3-1}$$

其中，L_{m0} 为输入消息 M_0 的长度，$L_{f2}=64$。

（2）对消息进行补位。

将消息 M_0 后面拼接 1 个“1”、$(L_{f1}-1)$个“0”，以及扩展为 64 位的 L_{m0} 值。

步骤 3：对补位后消息按 512 位大小进行分解，再将每个 512 位（32 位×16）的消息块扩展为 2048 位（32 位×64）。

（1）将补位后的消息分解成 n 个消息区块 $M^{(0)}$，$M^{(1)}$，…，$M^{(n-1)}$，每个消息区块为 512 位。

其中，$M^{(i)}$ 为第 i 个消息块，每个消息块由 16 个 32 位大端序字构成，记为 $M_0^{(i)}$，

$M_1^{(i)},\cdots,M_{15}^{(i)}$。

（2）按式(3-2)将16个字赋值给$W_0,W_1,\cdots,W_{15}$。

$$W_j=M_0^{(i)}\quad(j=0,1,\cdots,15)\tag{3-2}$$

（3）按式(3-3)将消息块扩展为64字。

$$W_j=\sigma_1(W_{j-2})+W_{j-7}+\sigma_0(W_{j-15})+W_{j-16}\quad(j=16,17,\cdots,63)\tag{3-3}$$

其中：

$$\sigma_0(W_{j-2})=S^7(W_{j-15})\oplus S^{18}(W_{j-15})\oplus R^3(W_{j-15})\tag{3-4}$$

$$\sigma_1(W_{j-2})=S^{17}(W_{j-2})\oplus S^{19}(W_{j-2})\oplus R^{10}(W_{j-2}\text{ 右移 }10\text{ 位})\tag{3-5}$$

S^m 为循环右移m位，R^n 为右移n位。

步骤4：加密计算哈希摘要。

（1）用8个哈希初值$H_0^{(0)},H_1^{(0)},\cdots,H_7^{(0)}$初始化临时变量$A$、$B$、$C$、$D$、$E$、$F$、$G$、$H$。

（2）对每个消息区块进行64次迭代，生成消息区块的哈希摘要。

$$H^{(i)}=H^{(i-1)}+C_{M^{(i)}}(H^{(i-1)})\quad(i=0,1,\cdots,n-1,j=0,1,\cdots,7)\tag{3-6}$$

其中，$H^{(i)}$和$H^{(i-1)}$分别为第i和第$i-1$个消息区块的哈希摘要，“C”代表SHA256的压缩函数(具体实现见图3.1)，“M”代表SHA256压缩函数的输出值对2^{32}取模，“+”代表将两个数字加在一起。

（3）最后一个消息块消息$H^{(n-1)}$即为最终的哈希摘要。

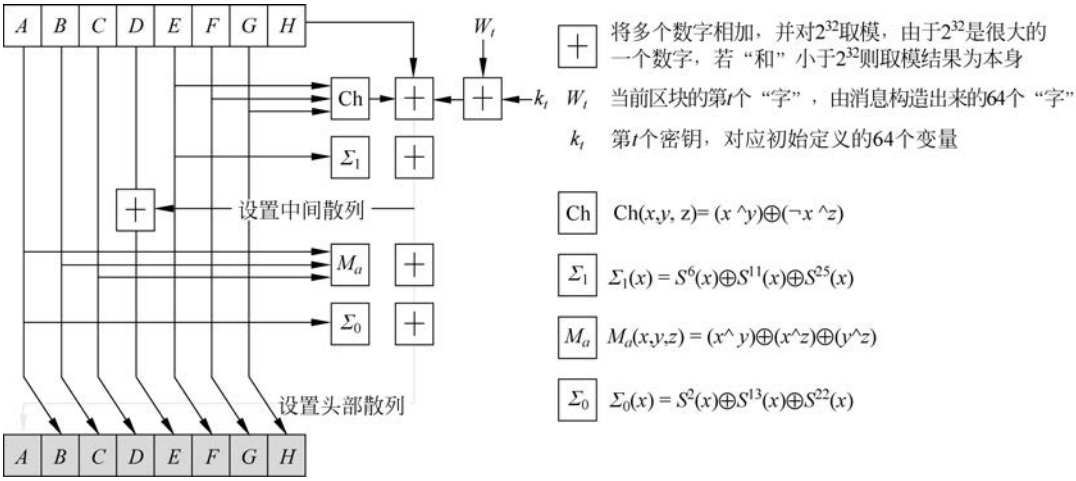

图3.1 SHA256压缩函数调用图

3.2 Merkle树(默克尔树)

在计算机领域，Merkle树常用来进行数据的完整性验证，在分布式环境下进行数据完整性验证时，Merkle树大大减少了数据传输量，降低了计算复杂度，区块链交易以Merkle树的形式存储在交易体中，通过Merkle树根可快速验证区块体中交易数据的完整性。

3.2.1　Merkle 树基本概念

1. 定义

Merkle 树是一类基于哈希值的二叉树或多叉树，其叶子节点上的值通常为数据块的哈希值，而非叶子节点上的值是将其子节点的值进行组合，之后再进行哈希计算得到的哈希值。

如图 3.2 所示，Merkle 哈希树叶子节点 C、D 分别为数据块 001 和 002 的哈希值，非叶子节点 A 存储值是将其子节点 C、D 的值进行组合之后的哈希值，这类非叶子节点的哈希值被称作路径哈希值，而叶子节点的哈希值是实际数。

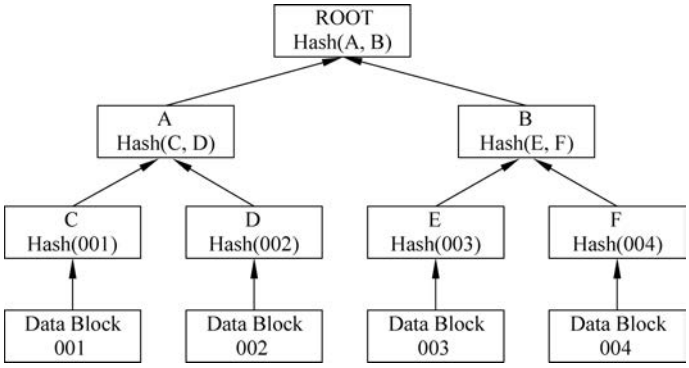

图 3.2　Merkle 哈希树

基于 Merkle 的结构特性，当上述数据从节点 A 传到节点 B 时，为了保证数据完整性，需要验证节点 A 和节点 B 上 Merkle 树根节点的值是否一致。若一致，则表示数据在传输过程中没有发生改变；若不一致，则说明数据在传输过程中被修改。当数据被修改时，通过 Merkle 树可快速定位找到被篡改的节点，时间复杂度为 $O(\log(n))$，其中，比特币的轻量级节点所采用的 SPV 验证便是利用 Merkle 树的这一特性。

2. SPV 验证

SPV(Simplified Payment Verification，简单支付验证)指不运行完全节点也可验证支付，用户只需保存所有区块头即可。SPV 属于区块链中支付验证体系的一种，该方式无须下载新区块的所有数据，只需保存区块头部数据，即可实现简便、快速支付的验证。

SPV 属于区块链支付验证，而不是交易验证，它只负责判断交易是否已得到区块链中节点的共识验证，以及得到了多少次确认。SPV 验证流程如下。

步骤 1：计算待验证支付交易的哈希值。

步骤 2：节点从网络中获取所有区块头信息，保存到本地。

步骤 3：节点从区块链获取待验证支付交易对应的 Merkle 树哈希认证路径。

步骤 4：根据哈希认证数据，与保存在本地的 Merkle 树比较，定位到包含本次交易的

区块。

步骤 5：验证该区块头信息是否包含在已知最长链中。

步骤 6：根据区块头位置，确认已经确认的次数（比特币网络中确认一笔交易，需至少 6 次确认）。

简而言之，比特币中的节点在打包新区块时，会对区块里所有交易进行验证，且一笔交易需得到 6～7 次的确认，以保证交易最后的完成。在使用 SPV 时，只要判断一笔交易是否在主链的某个区块里出现过，即可证明该交易之前是否被验证过。

3.2.2 Merkle 树构造过程

如图 3.3 所示，Merkle 树构造过程采用自底向上的方式。首先，对底层的数据块（Data Block）执行哈希运算，生成节点 0 层的值；然后，依次对相邻两个节点串联后执行哈希运算形成上层节点的值，按此规则一直递归，直到生成一个唯一节点，即顶层节点（根节点）的哈希值，Merkle 树就构造好了。如果节点层对应的节点数是偶数，可以两两节点的值串联；但如果节点层对应的节点数是奇数，从左到右，节点的值两两串联执行哈希运算，必然会有一个孤节点的值，此时对它进行复制后串联再执行哈希运算。

步骤 1：对数据块做哈希运算，生成节点 0 层。

步骤 2：若节点数大于 1，将相邻两个节点串联后执行哈希运算，如果遇到单个节点则复制后执行哈希运算，生成上层节点的值，重复步骤 2，直至生成 Merkle 树根。

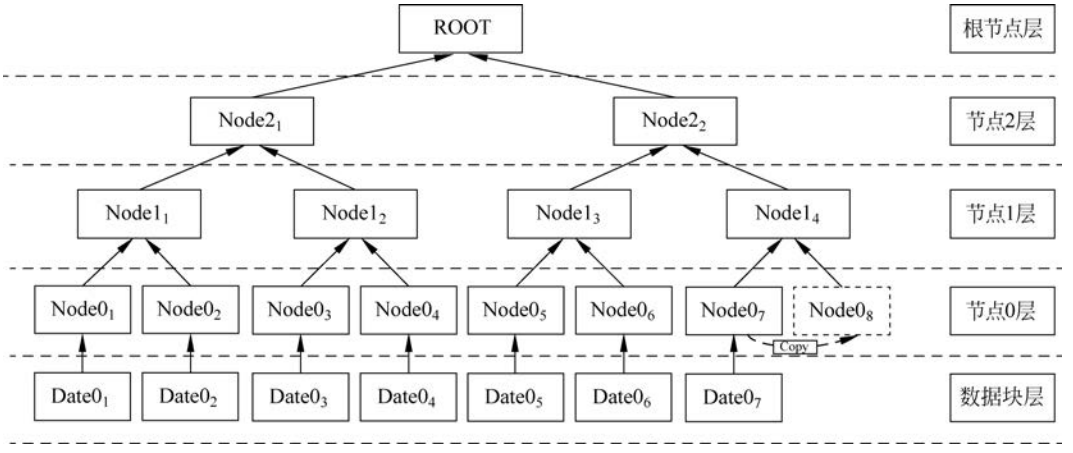

图 3.3 Merkle 树构造

从图 3.3 可以看出计算创建 Merkle 树的复杂度为 $O(n)$，即需要执行 $O(n)$次哈希运算，其中，n 为数据块的数量，Merkle 树的树高为 $\log_2 n+1$。

3.2.3 Merkle 树检索过程

假设有 A 和 B 两台机器，在相同目录下有 8 个文件，分别是 f_1、f_2、f_3、…、f_8。如图 3.4 所示，文件创建时，每个机器均构建了一棵 Merkle 树。叶子节点 Node7 的值为 Hash(f_1)，即 f_1 文件的哈希值，而其父节点 Node3 的值为 Hash(V7，V8)，Root 节点的

值其实是所有叶子节点值的唯一特征。

当机器 A 中的文件 5 与机器 B 中的不一致时，通过两个机器中 Merkle 树的相关信息检验比较的过程如下。

步骤 1：比较两个机器中 V0 是否相同，如果不同，则检验其子节点 Node1 和 Node2 的值。

步骤 2：检验结果，V1 相同，V2 不同，则继续检验 Node2 的子节点 Node5 和 Node6 的值。

步骤 3：检验结果，V5 不同，V6 相同，则继续检验 Node5 的子节点 Node11 和 Node12 的值。

步骤 4：检验结果，V11 不同，V12 相同，因 Node11 为叶子节点，则检验完毕，获取其目录信息。

通过分析可得，上述过程的理论复杂度为 $\log n$。

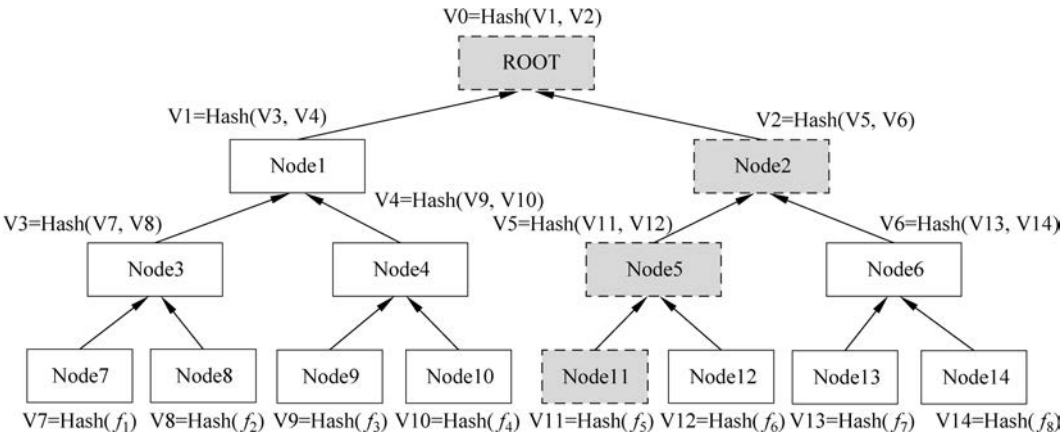

图 3.4　机器构造的 Merkle 树

3.3　数据加密技术

密码学是区块链和加密货币的重要应用技术，密码学技术解决了区块链网络中数据的安全性及完整性问题，有效保证了数据的不可篡改，是区块链高度可信的重要理论基础。

3.3.1　非对称加密

非对称加密，即在应用的过程中，需要两个不同的密钥来进行加密操作和解密操作，而这两个密钥分别为公开密钥（Public Key，简称公钥）和私有密钥（Private Key，简称私钥），公钥公开，私钥保密。

非对称密钥主要应用于机密数据传输，如图 3.5 所示，针对接收方 B 生成一对密钥，发送方 A 得到 B 公钥之后，将需要加密的明文信息进行加密处理，得到对应的密文，之后

将其发送给接收方 B；接收方 B 在收到发送方 A 发送的密文之后，使用对应的私有密钥（B 私钥）对密文进行解密处理，即可得到对应的明文。

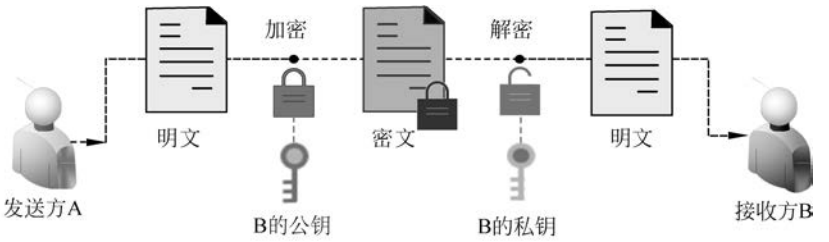

图 3.5　非对称加密技术实现机密数据传输

如图 3.6 所示，为非对称加密算法在数据传输中的另一种应用场景：发送方 B 使用自己的私钥对原文信息进行数字签名，将数字签名与原文信息一同发送给接收方 A；接收方 A 在收到数字签名与原文之后，使用发送方 B 提供的公钥对数字签名进行解密验证，若验证成功，则表明上述信息是由发送方发送的，并非他人冒充或更改过的信息。这就是常用的数字签名技术。

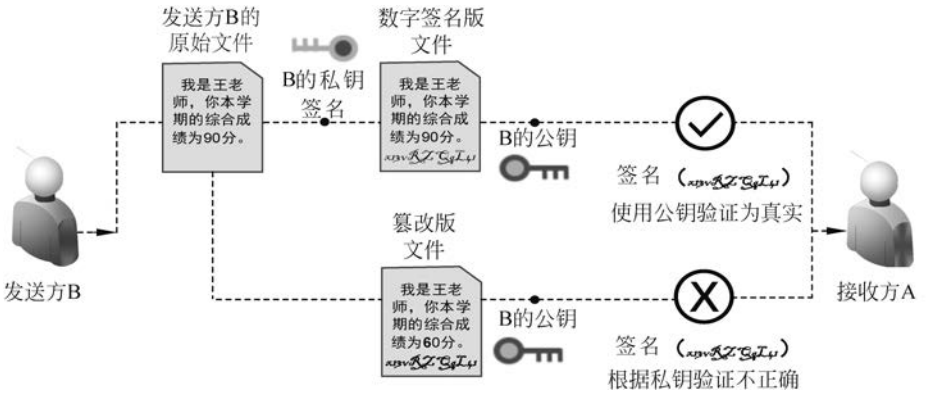

图 3.6　非对称加密技术实现数字签名

非对称加密算法公、私钥分开存储，无须安全通道来分发传递密钥，保证了密钥的安全性。但密钥生成及解密过程效率较慢，加密强度略逊于对称加密算法。

代表性的非对称加密算法有 RAS、ElGamal、椭圆曲线算法、SM2。

3.3.2　数字签名

类似于纸质合同上的书面签名，数字签名是只有信息的发送者才能生成的无法伪造的一段数字串，主要用于确认合同内容和身份证明。数字签名既可以证实某数字内容的完整性，又可确认数字内容的准确来源。

数字签名的典型场景是确认传送消息来源的准确性，当用户 A 通过信道发给用户 B 一份文件，用户 B 如何才能确定所收到的文件为用户 A 发送的原始版本呢？首先，用户 A 可先对文件内容进行摘要提取，然后用自身私钥对摘要进行数字签名，之后将文件和

签名一同发给用户 B；用户 B 收到信息之后，采用用户 A 的公钥解密数字签名，得到对应的摘要，并采用相同的方式对文件内容进行摘要提取，对比两个摘要，若一致，则说明该文件确实是用户 A 发送的原版文件，且文件内容并没有被修改。

理论上讲，所有的非对称加密算法均可实现数字签名功能，实践中常用算法包括 1991 年 8 月 NIsT 提出的数字签名算法（Digital Signature Algorithm，DSA）和安全强度更高的椭圆曲线算法（Elliptic Curve Digital Signature Algorithm，ECDSA）等。

除普通的数字签名应用场景外，针对一些特殊的安全需求，研究人员创造了一系列特殊的数字签名技术，包括盲签名、多重签名、群签名、环签名等算法。

3.3.3　时间戳技术

时间戳是指 1970 年 01 月 01 日格林尼治时间 00:00:00（北京时间 1970 年 01 月 01 日 08:00:00）到现在时间的总秒数。在区块链系统中，新区块在生成时，均会被打上时间戳，各区块依照生成时间的先后顺序相连成为区块链，独立节点通过 P2P 网络建立联系，为信息数据的记录形成了一个去中心化的分布式时间戳服务系统。例如，比特币网络中，大约每 10min 产生一个新区块，并被盖上时间戳，广播到全网各个节点，各节点均有一份新区块的完整信息，包括时间戳，形成了一个分布式时间戳系统。时间戳的应用使得区块链交易数据的篡改难度按时间的指数级增加，越早的记录越难被篡改，即区块链运行时间越久，篡改难度越高。

在区块链结构中，每个区块由区块头和区块体两部分构成，区块体存储相关的区块链交易数据，交易数据通过哈希计算以 Merkle 树的形式存储，Merkle 树根的哈希值作为本区块中所有交易数据的摘要，被存入区块头中。区块头中除 Merkle 树根之外，还包括前一个区块摘要、本区块时间戳、区块高度等信息。

时间戳在区块链中扮演着公证人的角色，比传统的公证制度更加可信，其次，时间戳技术本身的意义便是为了证明先后顺序，这使得区块链技术在产权保护、数据存在等方面有着天然的技术优势。

在版权存证场景中，创作者经常遇到版权纠纷事件，例如，用户 A 想发表一篇作品，同时又担心署名遭到侵犯，造成版权纠纷，通过区块链技术配合时间戳技术，可在作品发表之前进行链上数据存证，将作品盖上时间戳，生成独一无二、不可篡改的具有时间证明的哈希证明，通过技术手段实现作品的版权归属证明。当发生版权纠纷时，通过查询链上的数据，便可为版权纠纷提供具有法律效力的存证证明。

3.3.4　零知识证明

1. 定义

零知识证明（Zero Knowledge Proof）指在不向验证者提供任何信息的情况下，证明者能够使验证者相信某个论断正确。在 20 世纪 80 年代初由 S. Goldwasser、S. Micali、C. Rackoff 等提出，早于区块链技术的诞生。

2. 分类

零知识证明技术主要可分为交互式和非交互式两种。

零知识证明协议的基础是交互式的，它要求验证者不断对证明者所拥有的“知识”进行一系列提问，证明者通过回答一系列问题，让验证者相信证明者的确知道这些“知识”。然而，这种简单的方法并不能使人相信证明者和验证者都是真实的，两者可以提前串通，以便证明者可以在不知道答案的情况下依然通过验证。

非交互式零知识证明不需要交互过程，避免了串通的可能性，需要额外的机器或程序来确定实验的顺序。

3. 示例

示例 1：用户注册。用户在系统注册时，系统不会保存用户的密码明文，而是保存密码的哈希值；用户在登录系统时，只需输入注册时的密码，系统便会根据输入密码产生哈希值，并与系统数据库中保存的哈希值进行比对。若一致，则系统认为当前登录用户知道该账号的密码，用户在不需要告诉网站密码的情况下，便证明了自己的身份。

示例 2：在校大学生实习证明。学生 A 要去公司实习，公司要求学生 A 在校所有功课无挂科，学生 A 虽然所有功课都没挂科，但并不想让公司看到自己的成绩单，于是 A 委托学校开了一个证明，以证明其在校所有功课全部及格，通过该证明，学生 A 成功加入了公司。这样，学生 A 在没有暴露自己确切考试成绩的情况下，向公司证明了自己满足公司要求。

4. 应用

在区块链应用中，通过零知识证明技术，可对交易双方的地址、交易细节等信息进行隐藏，保证相关数据的隐私性。

目前，在 ZCASH（大零币）中，零知识证明技术被用来证明交易的有效性，其方法为 zk-SNARK（zero knowledge Succinct Non-interactive ARgument of Knowledge）。ZCASH 对交易记录上的交易双方信息和金额数量进行加密隐藏，使矿工在不知道交易细节的情况下，仍然可以对交易进行验证。

在 ZCASH 中，交易采用一种基于 UTXO（Unspent Transaction Output，未消费交易输出），被称为 NOTE（支票）的新方式。NOTE 代表当前账户对资产的支配权，与 UTXO 不同，账户余额的存储方式不再是“未消费的交易输出”，而是“未被作废的 NOTE”；NOTE 由所有者公钥 PK、所拥有金额 v 和唯一区分支票的序列号 r 组成，具体表示如式（3-7）所示。

$$\mathrm{NOTE}=(\mathrm{PK},v,r) \tag{3-7}$$

ZCASH 交易分为透明地址（t-addresses）交易和隐藏地址（z-addresses）交易两类。

透明地址交易的输入、输出地址等信息是可见的 NOTE 信息，地址以“t”开头；隐藏地址交易的输入地址、输出地址、金额等信息是隐藏的，地址以“z”开头。当隐藏地址之间发生交易时，相关信息也会存储在公有区块链上，对外可见，手续费也会支付给矿工，但

交易地址、资金数额以及备注字段等信息都会利用零知识证明技术加密，故公众无法得知交易的具体信息。

3.4　数字签名算法

3.4.1　RSA 数字签名算法

RSA(Rivest Shamir Adleman)数字签名算法是常见的数字签名算法之一。RSA 公开密钥密码体制是一种使用不同的加密密钥与解密密钥，“由已知加密密钥推导出解密密钥在计算上是不可行的”密码体制。

其算法过程为：用户 A 对明文 m 用解密变换，式(3-8)如下。

$$s^{\circ}D_k(m)=m^d \bmod n \tag{3-8}$$

其中，d，n 为用户 A 的私人密钥，不对外公开；用户 B 收到用户 A 的签名后，用用户 A 的公钥和加密变换得到明文，式(3-9)如下。

$$E_k(s)=E_k(D_k(m))=(m^d)^e \bmod n \tag{3-9}$$

因

$$de^{\circ}1 \bmod j(n) \tag{3-10}$$

即

$$de=lj(n)+1 \tag{3-11}$$

根据欧拉定理：$m^{j(n)}=1 \bmod n$

故：

$$E_k(s)=m^{lj(n)+1}=[m^{j(n)}]^e m=m \bmod n \tag{3-12}$$

若明文 m 和签名 s 一起送给用户 B，则用户 B 可确信信息是用户 A 发送的。同时，用户 A 也无法否认发送过该信息，因为除 A 本人外，其他人都无法由明文 m 产生 s。因此，RSA 数字签名方案是可行的，但 RSA 数字签名算法因计算方法本身同构，造成签名易被伪造和计算时间长的弱点，因此，在对文件签名之前，需对消息做 MD5 变换。

MD5 函数是一种单向散列函数，可将任意长度的消息压缩成 128 位的消息摘要，利用 MD5 的单向性和抗碰撞性，可实现加密信息的完整性检验。此外，该函数的设计不基于任何假设和密码体制，执行速度快，是一种被广泛认可的单向散列算法。

3.4.2　DSA(数字签名算法)

DSA(Digital Signature Algorithm，数字签名算法)是一种公开密钥算法，它不能用作加密，只用作数字签名。DSA 使用公开密钥，为接受者验证数据的完整性和数据发送者的身份，也可被第三方用于确定签名和验证所签数据的真实性。DSA 的安全性基于解离散对数的困难性，该类签字标准具有较大的兼容性和适用性，是网络安全体系的基本构件之一。DSA 以 RSA 算法为基础，遵循“私钥签名，公钥验证”的原则，Java 6 提供了 DSA 的 SHA1withDSA 的实现，bouncycastle 扩展提供了其他的 DSA 实现方式。

3.4.3 ECDSA

ECDSA(Elliptic Curve Digital Signature Algorithm,椭圆曲线数字签名算法)是微软操作系统及办公软件的序列号验证算法,具有速度快、强度高、签名短等优点,于1999年作为ANSI标准,并于2000年成为IEEE和NIST标准。

ECDSA可看作椭圆曲线对先前基于离散对数问题(Discrete Logarithm Problem,DLP)的密码系统模拟,是群元素由素域中的元素数转换为有限域上的椭圆曲线上的点。椭圆曲线密码体制的安全性,是基于椭圆曲线离散对数问题(Elliptic Curve Discrete Logarithm Problem,ECDLP)的难解性,椭圆曲线离散对数问题远难于离散对数问题,椭圆曲线密码系统的单位比特强度也远高于传统的离散对数系统。因此,在使用较短密钥的情况下,椭圆曲线密码学(Elliptic Curve Cryptography,ECC)可达到与离散对数系统(Discrete Logarithm,DL)相同的安全级别,使得计算参数更小、密钥更短、运算速度更快、签名也更短,故椭圆曲线密码尤其适用于处理能力、存储空间、带宽及功耗受限的场合。

3.5 PKI

3.5.1 PKI简介

在非对称加密中,公钥可以通过证书机制进行保护,管理和分发证书则可以通过PKI保障。PKI(Public Key Infrastructure,公钥基础设施),是一个包括硬件、软件、人员、策略和规程的集合,用来实现基于公钥密码体制的密钥和证书的产生、管理、存储、分发和撤销等功能。PKI体系是计算机软硬件、权威机构及应用系统的结合,它为实施电子商务、电子政务、办公自动化等提供了基本的安全服务,从而使那些彼此不认识或距离很远的用户能通过信任链安全地交流。

密码学上,公开密钥基础建设借助数字证书颁发机构(CA)将用户的个人身份跟公开密钥连接在一起。每个证书中心用户的身份必须是唯一的。连接关系通过注册和发布过程创建,取决于担保级别,连接关系由CA的各种软件或在人为监督下完成。PKI确定连接关系的这一角色称为证书注册机构(Registration Authority,RA)。RA确保公开密钥和个人身份连接,可以防抵赖。

可信赖的第三者(Trusted Third Party,TTP)常被用来指证书中心。PKI有时被错误地拿来代表公开密钥密码学或公开密钥算法。

3.5.2 PKI组成

PKI主要由证书颁发机构(CA)、证书注册机构(RA)、证书库、密钥备份及恢复系统、证书废除处理系统、PKI应用接口等部分组成。

证书颁发机构(Certification Authority,CA)是PKI的核心,可对任何一个主体的公钥进行公证,通过签发证书将主体与公钥捆绑在一起。

证书注册机构(Registration Authority,RA)是CA面对用户的窗口,负责接收用户

的证书申请、审核用户的身份以及向用户发放证书等工作。

证书库是证书的集中存放地，用户可从此处获得其他用户的证书，构造证书库可采用WWW、FTP等技术，一般采用X.500系列标准格式，并配合LDAP(Lightweight Directory Access Protocol，轻量级目录访问协议)目录服务管理用户信息。

密钥备份及恢复系统专门用于因用户丢失解密密钥而无法解密合法数据的情况，该机制必须由可信的机构完成，且密钥的备份与恢复只能针对解密密钥，签名私钥不能作备份操作。

证书废除处理系统主要用于废除无用的证书，处理方式一般是将证书列入证书黑名单(Certificate Revocation List，CRL)，CRL一般存放在目录系统中。

PKI应用接口主要为各种各样的应用提供安全、一致、可信的与PKI交互的方式，确保建立起来的网络环境安全可靠，并降低管理成本。

3.5.3　PKI相关标准

1. X.509

X.509是密码学中公钥证书的格式标准。X.509证书已应用在包括TLS/SSL在内的众多Internet协议以及非在线应用场景中，例如，电子签名服务。X.509证书包含公钥、身份信息和签名信息。对于一份经由可信证书签发机构签名或者可通过其他方式验证的证书，其所有者可用证书及相应的私钥来创建安全的通信，对通信文档进行数字签名。

2. X.500

X.500是一套已被国际标准化组织(ISO)接受的目录服务系统标准，定义了一个机构如何在全局范围内共享其名字和与之相关的对象。X.500是层次性的，其中的管理域(机构、分支、部门或工作组)可提供域内的用户和资源信息。在PKI体系中，X.500被用来唯一标识一个实体，该实体可以是机构、组织个人或一台服务器。X.500被认为是实现目录服务的最佳途径，具有信息模型、多功能和开放性等优势，但X.500的实现需要较大的投资，并且比其他方式速度慢。

3. LDAP

LDAP在X.500基础上发展起来，用于Internet和Intranet之间实现网络信息快速访问的目录访问协议，采用简化的目录结构和目录访问方法，提供了不同目录、不同用户和目录之间进行信息交流的标准化方法。按照此协议实现的服务器为LDAP服务器，是CA的重要组成部分，可存储PKI证书系统产生的新证书及证书撤销列表。

3.5.4　PKI关键技术

1. 数字证书

数字证书是一段包含用户身份信息、用户公钥信息以及身份验证机构数字签名的数

据；是一个经证书认证中心数字签名的包含公开密钥拥有者信息以及公开密钥的文件；是各类终端实体和最终用户在网上进行信息交流及商务活动的身份证明，在电子交易的各个环节，交易的各方都需验证对方数字证书的有效性，从而解决彼此之间的信任问题。用户公钥信息可保证数字信息传输的完整性，而数字签名可保证数字信息的不可否认性。

2. CA 及信任链

CA 认证中心作为权威的、可依赖的、公正的第三方机构，专门为各种认证需求提供数字证书服务，例如，颁发证书、废除证书、更新证书、验证证书、密钥备份与恢复等。

在向 CA 申请证书时，需用 CA 的私钥对整个证书的签名摘要做非对称加密，即证书可通过 CA 的公钥去解密获取证书的签名摘要。再次用相同的摘要算法对整个证书做签名时，若得到的签名和证书中的签名一致，则说明该证书是可信的。

同理，中介证书也可通过该方式证明其可信，该流程称为信任链（Chain of Trust），即：A 绝对相信 B（A＞B），B 绝对相信 C（B＞C），等于 A 绝对相信 C（A＞C）。信任链验证流程如下。

（1）客户端得到服务端返回的证书，通过读取得到服务端证书的发布机构（Issuer）。

（2）客户端去操作系统查找该发布机构的证书，若不是根证书，则继续递归下去，直到拿到根证书。

（3）用根证书的公钥去解密验证上一层证书的合法性，之后，拿上一层证书的公钥去验证更上层证书的合法性，一直递归回溯。

（4）最后，验证服务器端证书是否可信。

3. 证书发放流程

证书的发放过程具体可分为以下六个步骤。

（1）通过管理员终端，录入用户申请信息。

（2）审核受理终端审核提交证书申请。

（3）签名 CA 服务器向密钥管理服务器索取密钥对。

（4）密钥管理服务器向签名 CA 服务器返回密钥对。

（5）签名 CA 服务器签发证书并发布。

（6）用户通过管理员终端下载证书并制证。

3.5.5 PKI 功能

PKI 应用非常广泛，通过管理密钥和证书，为各种应用提供安全保障，提供网络信任基础，可以提供身份认证、数据完整性验证、数据机密性、操作不可否认性、时间戳五种安全服务。

1. 身份认证

身份认证就是证实被认证对象的真实性与有效性的过程。例如，某机构通过 CA 申请认证证书之后，向互联网用户公开该证书，其他用户若接受，则将证书存储在其计算机

中。当该机构在发布数据时，带上数字签名，签名可以是一个明文字符串 X，经过该机构私钥加密后的结果为 Y，其他用户在收到数据之后，通过该机构公开证书中的公钥对签名中的 Y 进行解密得到 Z，此时，如果 Z 和 X 相等，则可以判定该数据确实是由该机构发布的，即完成了对该机构的身份认证。

2. 数据完整性验证

数据的完整性是防止非法篡改信息，如修改、复制、插入、删除等。确保交易双方接收到的数据与原数据完全一致。数据在网络中传输的时候，会遇到各种安全问题，如被劫持、篡改等。接收方收到发送方利用私钥签名后的数据，用公钥进行验证，如果验证通过，则表示数据未被篡改。信息劫持者没有私钥，对劫持数据的篡改就无法通过公钥的验证。

3. 数据机密性

数据的机密性是对需要保护的数据加密，保证数据在传输和存储过程中不被未授权人获取。假定数据发送者 A 要给数据接收者 B 发送私密数据。A 使用 B 的公钥进行数据加密，A 接收到数据后用私钥进行解密后才能获取正确的数据。未经授权者即使截获数据，也只是一串无意义的乱码。

4. 操作不可否认性

操作不可否认性是指在传输数据时必须携带含有自身特质、别人无法复制的信息，防止交易发生后对行为的否认。这就类似我们在进行交易时签订合同，需要加盖公章和签名一样。在网络上进行交易数据传输时，也需要用 CA 发放的私钥进行数字签名，接收交易数据者通过公钥验证，就确保所接收到的交易数据是由私钥持有者所发送的。

5. 时间戳服务

安全时间戳是 PKI 提供的附加功能，时间戳将电子文档和权威时间值通过加密方式关联，证明电子文档从该时间值开始存在。例如，某学者撰写了一部作品，基于版权保护意识，他为该作品申请了时间戳服务，作品与时间戳可以在将来某个发生版权纠纷的时间证明作品的版权归属。

小结

本章主要讲解了在区块链技术中所涉及的相关密码学知识，主要包括哈希函数、Merkle 树、数据加密技术、数字签名算法以及 PKI 等技术。

本章以区块链架构中由外到内用到的密码学技术为顺序。首先，本章详细介绍区块的哈希函数，具体介绍了常用的 SHA256 哈希算法；之后，对区块体中用到的 Merkle 树技术进行了简要介绍，便于读者清楚地理解区块交易存储原理；接下来，介绍了区块链中所应用的常见数据加密技术，主要包括非对称加密、数字签名、时间戳以及零知识证明等技术；然后，简要介绍了常见的几种数字签名算法，包括 RSA、DSA 以及 ECDSA 算法；

最后，对 PKI 及其应用进行了简要介绍，使读者可充分了解 PKI 的特点及其在区块链中的重要作用。

习题

3.1 请描述哈希算法的具体原理。
3.2 请描述 Merkle 树的基本概念、构造过程以及检索过程。
3.3 请描述非对称加密算法的具体原理。
3.4 请比较对称加密与非对称加密算法的区别和联系。
3.5 数字签名和时间戳技术在超级账本 Fabric 中有何作用？
3.6 描述零知识证明技术的原理。
3.7 理解 RSA、DSA 以及 ECDSA 数字签名的算法原理。
3.8 请描述 PKI 技术在超级账本 Fabric 中的作用。
3.9 请描述 PKI 的具体功能。
3.10 描述密码学技术概念以及列举另外一种密码学技术。