Bitcoin: A Peer-to-Peer Electronic Cash System

比特币：一种点对点的电子现金系统

From Theory to Code

An Implementation-Level Commentary

从理论到代码

一份实现级的评注

Satoshi Nakamoto

中本聪

satoshin@gmx.com

www.bitcoin.org

译者&评注：JIN

评注前言

2008 年，一份署名为“中本聪”的、仅有九页的论文《比特币：一种点对点的电子现金系统》悄然问世。它如同一道思想的闪电，划破了数字世界的夜空，不仅催生了价值万亿美元的加密货币产业，更开启了一场关于货币、信任和自由的全球性社会实验。

这份白皮书以其惊人的简洁和深刻的洞察力，描绘了一个无需信任第三方、完全由代码和共识驱动的金融系统的蓝图。然而，正如所有伟大的设计蓝图一样，白皮书为了保持其核心思想的清晰，省略了海量的工程细节。它告诉了我们“是什么”和“为什么”，却没有完全展开“怎么做”。

初衷 (Why)

这个项目，便诞生于探索“怎么做”的好奇心。当我们阅读白皮书时，无数问题油然而生：P2P 网络节点是如何发现彼此的？交易池（Mempool）如何管理待确认的交易？“数字签名链”在代码中究竟是哪个数据结构？白皮书中未曾提及的交易费，又是如何成为系统运转的血液的?

我们意识到，要真正理解比特币，就必须跨越从理论到实现的鸿沟。我们希望创建一份独特的文档——一份“代码注释版”的白皮书。它将以中本聪的原文为经，以 Bitcoin Core 的源代码为纬，将抽象的概念与具体的实现紧密编织在一起。我们的旅程最终将汇成这份详尽的注释版白皮书，希望能为和我们一样对技术充满好奇心的探索者们，提供一幅更详尽、更生动的比特币世界地图。

方法 (How)

我们的探索之旅遵循着一条独特的路径。我们没有选择传统的、自上而下的理论分析，而是采用了一种代码考古的新范式。

我们的基础素材正是 Bitcoin Core 的源代码库——这个直接继承自中本聪最初版本的、经过全球顶尖开发者十余年打磨的工程奇迹。我们对这个庞大的代码库进行了定位、解剖和分析。我们力求中立、客观，将重点放在揭示技术实现与设计权衡上，而非对不同的技术路线进行价值判断。

鸣谢 (Thanks)

知识的探索从来不是一座孤岛。这个项目的完成，离不开许多人的启发与支持。

在此，我要特别感谢我的老师 **OliviaLL**，以及我的好朋友 **立二拆四**。在无数次的讨论中，你们提出的深刻问题和独到见解，一次次点燃了我的思维火花，为这个项目注入了不可或缺的活力。

我还要感谢 Bittensor。一方面，正是 Bittensor 的创始团队及社区对激励机制和共识的

独特解读，促使我回头重新审视比特币的基石，深入研究其底层细节；另一方面，Bittensor 社区一直以来给予的各种帮助和支持，也为我的探索提供了宝贵的土壤。

最后，我们要感谢 **中本聪**，以及所有为比特币核心项目贡献过智慧和代码的开发者们。是你们构建了这座值得我们反复探索的、精妙绝伦的数字教堂。

我们希望，这份文档能成为后来者攀登这座高峰时的一根有用的登山杖。

JIN

2025 年 6 月 18 日

摘要，一种纯粹的点对点版本的电子现金，将允许在线支付直接由一方发起，支付给另一方，而无需经过金融机构。数字签名提供了部分解决方案，但如果仍需一个可信的第三方来防止双重花费，那么其主要优势也就荡然无存。我们提出一种通过点对点网络来解决双重花费问题的方案。该网络通过将交易哈希进一个持续增长的、以哈希为基础的工作量证明链中来为交易打上时间戳，从而形成一个若无重做工作量证明则不可更改的记录。

> **【评注】** 这份摘要是整篇论文的高度浓缩，它精准地提出了问题（在没有中心化信任的情况下防止双重花费）并预告了解决方案的核心支柱（点对点网络、工作量证明、链式结构）。我们后续的所有分析，都是在为这份摘要中的每一个承诺提供详尽的证据。

1. 引言（Introduction）

互联网上的商业活动，已变得几乎完全依赖于金融机构作为可信第三方来处理电子支付。虽然这套体系在大多数交易中尚能良好运作，但它仍然受困于这种基于信任的模式所固有的弊病。完全不可逆的交易实际上是不可能实现的，因为金融机构无法避免地要

居中调解争端。调解的成本增加了交易费用，限制了实际可行的最小交易规模，并杜绝了小额临时交易的可能性。同时，对于那些不可逆的服务，也失去了使用不可逆支付方式的能力，这是一种更广泛的成本。由于逆转的可能性存在，信任的需求便会蔓延。商家必须对他们的顾客保持警惕，向他们索取本不必要的更多信息。一定比例的欺诈被认为是不可避免的。当面使用实体货币可以避免这些成本和支付的不确定性，但是，目前还没有一种机制，可以在没有可信方的情况下，通过通信渠道进行支付。

我们需要的是一个基于密码学证明而非信任的电子支付系统，它允许任何有意愿的双方直接进行交易，而无需一个可信的第三方。让交易在计算上变得不可行为逆转，可以保护卖家免受欺诈，而常规的托管机制也可以被轻易地实现来保护买家。在本文中，我们提出一种解决方案，通过一个点对点的分布式时间戳服务器，来生成交易时间顺序的计算证明，从而解决双重花费问题。只要诚实节点集体控制的 CPU 算力超过任何协作的攻击者节点群体，该系统就是安全的。

【评注】

白皮书开篇即宗，直指传统金融模型的要害——对可信第三方的依赖。比特币的解决方案，是将银行的三个核心角色（托管、授权、结算）进行彻底的去中心化。

- **托管 (Custody):** 用户通过钱包自己保管资金。我们深入研究的 HD 钱包 (BIP 32/39) 技术是关键。用户的资金不与任何中介机构绑定，而是与他们自己持有的主种子和派生出的私钥绑定。私钥即所有权。
- **授权 (Authorization):** 授权是一个纯粹的、非人为干预的密码学过程。其核心是脚本系统 (src/script/interpreter.cpp)。一笔交易是否有效，取决于花费者能否提供一个有效的 scriptSig 来解锁之前输出中的 scriptPubKey。核心操作 OP_CHECKSIG 只验证签名的数学有效性。
- **结算 (Settlement):** 结算通过一个去中心化的共识机制完成。这个系统之所以能够持续运行，是因为它内建了一套精妙的经济激励机制。矿工之所以愿意投入巨大的物理成本（电力）来执行工作量证明，是因为他们可以获得区块补贴和交易手续费。这套机制确保了总有理性的参与者

愿意承担起“银行”的角色，而他们的行为完全是出于自身利益。

这一节清晰地阐明了比特币诞生的背景和其核心价值主张：摆脱对中心化第三方的依赖。但如何才能在没有中心信任的情况下，确保数字货币不被重复花费呢？这正是白皮书下一节将要解决的核心难题——双重花费（Double-Spending）。

数字信息可以被零成本地完美复制。双重花费就是指，恶意用户将同一笔“数字货币”支付给两个或多个不同的接收者。数字签名本身只能验证所有权，但无法阻止所有者创建多笔都合法签名的、花费同一个 UTXO 的交易。没有一个全局统一的账本，就无法判断哪一笔交易是唯一有效的。

比特币的解决方案不是单一的技术，而是一个由五个关键支柱协同工作的、精妙的系统。

- **公共账本 (The Public Ledger):** 比特币区块链是一个全球共享的公共账本，任何双花尝试都会被公开记录，使其无所遁形。
- **UTXO 模型 (The UTXO Model):** 比特币不采用“账户/余额”模型，而是采用 UTXO（未花费交易输出）模型。这个模型更像是实物现金。你钱包里的不是一个“100 元”的数字，而是几张实体钞票。当你需要支付 30 元时，你必须递出 50 元，然后等待找零 20 元。比特币的 UTXO 就是这些“数字钞票”，找零输出（Change Output）就是其必然结果。这种模型极大地简化了双花检测：节点只需检查这枚“数字钞票”（UTXO）是否已经被花掉即可。
- **时间戳服务器 (The Chain):** 这是防止双花的排序机制。通过在 CBlockHeader 中包含 hashPrevBlock 字段，每个区块都牢牢地锁定了其前一个区块，形成了一条不可分割的时间链，为所有交易提供了一个明确的、唯一的时间顺序。
- **工作量证明 (The Cost):** 这是为历史的“书写权”设置的物理成本。通过要求矿工不断调整 nNonce 来寻找一个低于 nBits 规定目标的哈希值，

协议确保了向公共账本添加新的一页（区块）需要消耗大量的、真实的物理资源。它将纯粹的数字信息与现实世界的**真实物理成本**绑定在一起，使得创造历史的行为变得昂贵。

- **最长链规则 (The Consensus):** 这是一个简单而强大的共识算法。在面对区块链分叉时，所有诚实的节点都遵循一个简单的规则：永远承认并扩展那条拥有最大累计工作量（nChainWork）的链作为唯一权威的历史。诚实矿工遵循此规则，不仅因为协议规定，更是出于纯粹的经济理性。如果一个矿工选择在一条较短的分叉上挖矿，他将面临沉没成本损失（已挖出的奖励作废）和机会成本损失（放弃在主链上获得未来奖励的机会）。这种设计确保了矿工的最大利益与整个网络的整体利益高度一致。

综合起来: UTXO 模型定义了什么是“双花”，公共账本使其可见，时间戳链为其排序，工作量证明使其难以伪造，而最长链规则让所有人对最终的排序达成共识。

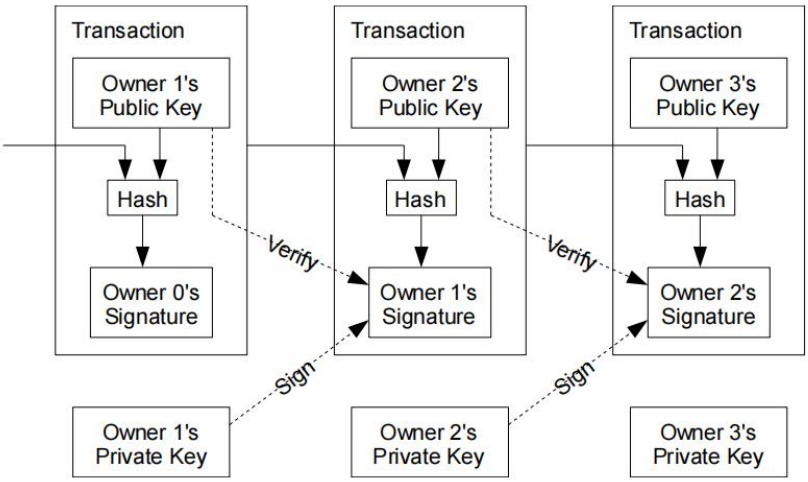
比特币的“不可变性”是一种经济学和概率论上的保证。要篡改 6 个区块前的一笔交易，攻击者必须重新计算从那个区块开始的所有后续区块的工作量证明，并且速度要超过整个诚实网络的增长速度。除非攻击者掌握了超过 51% 的全网算力，否则其成功的概率会随着区块深度的增加而呈指数级下降。这就是为什么“6 个区块确认”被广泛认为是交易达到实际上的、经济学意义上的最终确认的黄金标准。

白皮书通过五大支柱，巧妙地构建了一个去中心化的防双花系统。这里的核心在于，如何让这个“公共账本”既能被所有人信任，又没有中心化的“管账人”。解决这个问题的关键，在于如何有效地将数字交易“时间戳化”并“链接”起来，而这正是下一节“交易”和“时间戳服务器”要详细展开的内容。

2. 交易（Transactions）

我们将一枚电子货币（an electronic coin）定义为一个数字签名的链条。每一位所有者通过对前一笔交易的哈希（Hash）和下一位所有者的公钥（Public Key）进行数字签名

（Signature），并将这些信息附加在这枚电子货币的末尾，来将其转移给下一位所有者。收款人可以通过验证这些签名，来核实其所有权链条。



当然，问题在于收款人无法核实之前某位所有者没有对这枚电子货币进行双重花费。一种常见的解决方案是引入一个可信的中央权威机构，或称之为“铸币厂”，由它来检查每一笔交易是否存在双重花费。在每笔交易之后，这枚电子货币都必须返还给铸币厂来发行一枚新的电子货币，并且只有从铸币厂直接发行的电子货币，才被信任是没有被双重花费的。这种解决方案的问题在于，整个货币系统的命运都维系于运营这个铸币厂的公司，每一笔交易都必须经过他们，这与银行无异。

我们需要一种方法，能让收款人知晓之前的各位所有者没有签署过任何更早的交易。对我们的目的而言，最先发生的交易才是有效的，所以我们不关心后续的双重花费尝试。确认一笔交易不存在的唯一方法，就是知晓所有的交易。在基于铸币厂的模型中，铸币厂知晓所有交易，并由它来决定哪一笔最先到达。要在没有可信方的情况下实现这一点，交易就必须被公开宣告[1]，并且我们需要一个系统，能让所有参与者对一个单一的、记录了交易接收顺序的历史达成共识。收款人需要这样的证明：在每笔交易发生之时，大多数节点都同意它是最先被收到的。

【评注】

在上一节，我们了解了双重花费的定义以及比特币解决该问题的五大支柱。

本节将深入探讨这些支柱的“原子”——比特币如何定义和构建一笔“电子货币”的交易，并揭示其背后的密码学和脚本魔法。
白皮书将“电子货币”描述为一个概念性的“数字签名链条”。在 Bitcoin Core 的代码实现中，这个概念被一个具体、精确的数据结构所承载：
CTransaction 类。这个类不仅实现了白皮书中的核心概念，还增加了许多必要的工程细节，这些细节对于一个健壮、可演进的系统至关重要。

CTransaction 类的主要成员变量包括交易版本号 (nVersion)、交易输入 (vin)、交易输出 (vout) 和 锁定时间 (nLockTime)。

[深入代码]

CTransaction 类的定义位于 src/primitives/transaction.h。

- **交易版本号 (nVersion)** 是一个白皮书中未曾提及的关键字段。它的主要作用是为协议的升级提供支持，特别是软分叉。通过改变版本号，可以引入新的交易规则（例如，BIP 68 引入的相对时间锁就要求交易的 version 必须大于等于 2），而旧节点仍然可以兼容处理，从而确保了网络在不分裂的情况下平稳演进。
- **交易输入 (vin)** 是一个由 CTxIn 对象组成的向量（列表），直接对应白皮书中的“前一次交易”。每一个 CTxIn 对象都包含两个关键部分：一个 prevout 字段，它精确地指向某个之前交易的某个输出（UTXO）；以及一个 scriptSig 字段，即“解锁脚本”，其中包含了用以证明所有权的数字签名等数据。
- **交易输出 (vout)** 是一个由 CTxOut 对象组成的向量，对应于白皮书中创造的新“电子货币”。每一个 CTxOut 对象都包含两部分：nValue，即这个输出所代表的比特币金额；以及一个 scriptPubKey 字段，即“锁定脚本”，它定义了花费这笔钱所必须满足的条件。

- **锁定时间 (nLockTime)** 是另一个白皮书未提及的字段。它为交易增加了时间维度，允许创建在未来某个特定时间点或区块高度之后才能生效的“期货交易”或复杂的智能合约，例如用于实现延迟支付、原子交换和支付通道等复杂应用。

值得注意的是，CTransaction 的核心成员在设计上是**不可变 (Immutable)** 的。这是因为一笔交易的唯一 ID（txid）是对其完整的序列化数据进行哈希计算的结果。一旦交易被创建，其任何核心内容都不能被修改，否则 txid 将会改变，整个依赖于它的引用链条就会断裂。这种不可变性是保障系统完整和可验证性的基石。

白皮书中的“数字签名”在代码中被一个远比其字面意义更强大的脚本系统所实现。其核心逻辑位于 src/script/interpreter.cpp。这个系统定义了比特币的“可编程性”。

其核心机制在于“解锁”与“锁定”的交互。每一笔花费，都是用一个输入中的“解锁脚本” (scriptSig) 去匹配它所引用的 UTXO 中的“锁定脚本 (scriptPubKey)。验证过程非常优雅：验证程序会将 scriptSig 和 scriptPubKey 拼接在一起，然后交给一个基于栈的、非图灵完备的解释器（EvalScript`函数）来执行。

P2PKH 脚本执行流程示意：

初始栈: [ ]
执行 <Signature>: [<Signature>]
执行 <PublicKey>: [<Signature>, <PublicKey>]
执行 OP_DUP: [<Signature>, <PublicKey>, <PublicKey>]
执行 OP_HASH160: [<Signature>, <PublicKey>, <PubKeyHash_from_PublicKey>]
执行 <PubKeyHash_from_Address>: [<Signature>, <PublicKey>,

<PubKeyHash_from_PublicKey>, <PubKeyHash_from_Address>]
执行 OP_EQUALVERIFY: [<Signature>, <PublicKey>]（验证 PubKeyHash 是否匹配，如果匹配则弹出两个哈希）
执行 OP_CHECKSIG: [TRUE] （验证 Signature 是否有效，如果有效则弹出 Sig 和 PubKey，推入 TRUE）
最终栈: [TRUE]

这种脚本语言被有意地设计为非图灵完备（例如，没有循环），这保证了任何脚本的执行都将在有限时间内终止。这是一种为了系统的可预测性和安全性而做出的、对灵活性的刻意牺牲，也是比特币与以太坊等图灵完备智能合约平台最根本的区别之一。

白皮书在这一节并未提及交易费，但它是在实际系统中与矿工激励同等重要的经济动力。

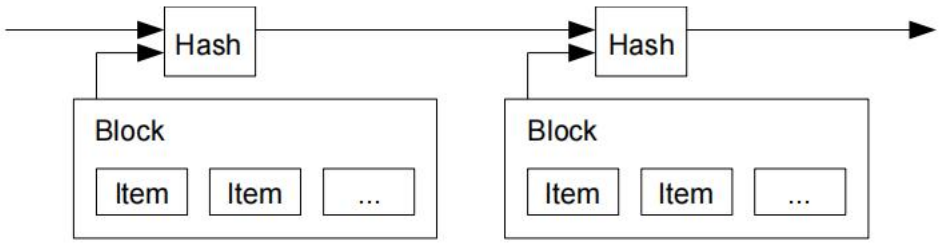
交易费在 CTransaction 结构中没有专门的字段，而是被隐式计算出来的，其公式为：交易费 = 所有输入(vin)的总金额 – 所有输出(vout)的总金额。这个差额部分，不会凭空消失，而是被成功打包这笔交易的矿工所获取，成为他们收入的一部分。这激励了矿工优先打包手续费更高的交易，形成了一个动态的、市场化的手续费竞价机制，对于网络的安全和交易的及时确认至关重要。

理解了比特币交易的“原子”结构和验证机制后，下一个问题自然浮现：如何将这些独立的交易，安全且有序地记录下来，形成一个所有人都信任的账本，以最终解决双重花费？这正是“时间戳服务器”将要回答的问题。

3. 时间戳服务器（Timestamp Server）

我们提出的解决方案，始于一个时间戳服务器。时间戳服务器的工作方式是，获取一批待打上时间戳的条目的哈希，并将该哈希广而告之，例如发布在报纸或新闻组

（Usenet）的帖子中[2-5]。显然，这个时间戳证明了，为了能被纳入该哈希的计算，这些数据在那个时间点必定已经存在。每一个时间戳都在其哈希中包含了前一个时间戳，从而形成一个链条，每一个新增的时间戳都不断地加固着它之前的所有时间戳。



【评注】

在第二节中，我们解剖了交易这个“原子”。现在，第三节将向我们展示，比特币是如何通过一个巧妙的“时间戳服务器”机制，将这些原子串联成不可变的“分子”——区块链的。

白皮书在此处用了一个非常巧妙的比喻——将哈希发布在报纸上——来阐述时间戳的核心思想：创造一个难以篡改的、带有时间顺序的公开记录。在比特币的实现中，这个“时间戳服务器”并非一个中心化的实体，而是一个由整个 P2P 网络共同维护的、去中心化的系统。

这个系统的核心是区块 (Block) 和 区块链 (Blockchain)。

区块 (Block) 作为“报纸版面”：

白白皮书中“一批待打上时间戳的条目”在比特币中就是一组交易。一个区块 (CBlock 类) 就扮演了“报纸的每日版面”的角色，它将一段时间内（大约 10 分钟）发生的多笔交易打包在一起。

[深入代码]

CBlock 类的定义位于 src/primitives/block.h。

区块头 (CBlockHeader) 作为“报纸头版”：

区块中最关键的部分是区块头 (CBlockHeader)。这是一个大小固定（80 字

节）的数据结构，它包含了该“版面”的所有元数据，相当于报纸的头版，其中最重要的字段有两个：

- nTime: 一个 Unix 时间戳，直接对应白皮书中的“时间戳”，记录了该区块被创建的大致时间。
- hashMerkleRoot: 这是对区块中所有交易进行哈希计算后得到的默克尔树根。它就是白皮书中所说的“一批条目的哈希”，作为一个唯一的、固定长度的指纹，代表了该区块包含的所有交易内容。

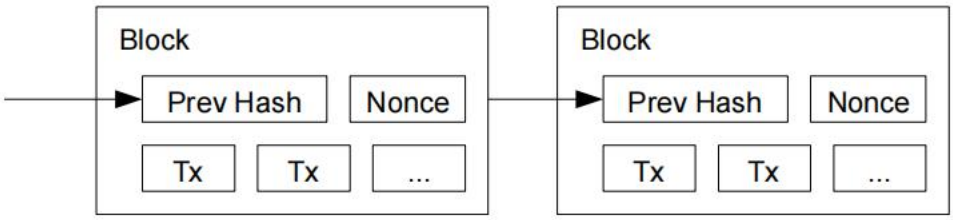
区块链 (Blockchain) 作为“链接起来的报纸”：
白皮书中“每一个时间戳都在其哈希中包含了前一个时间戳，从而形成一个链条”这一精髓，是通过区块头中的 hashPrevBlock 字段实现的。这个字段存储了其父区块的区块头哈希值。正是这个字段，像一根无法被切断的链条，将所有区块按照严格的时间顺序链接在一起，形成了区块链。
这个链式结构意味着，要篡改历史中任何一个旧区块，就必须重新计算从那个区块开始的所有后续区块的哈希值，因为每一个区块的身份（哈希值）都依赖于它前一个区块的身份。这种“层层加固”的效应，使得历史记录的篡改成本随着时间的推移而呈指数级增长，为系统的不可变性提供了基础。

这个链式结构为历史的篡改设置了障碍，但这种障碍的“强度”来自哪里？下一章“工作量证明”将为我们揭示，比特币是如何为这个链条注入真实的、不可伪造的物理成本，使其坚不可摧的。

4. 工作量证明（Proof-of-Work）

为了在点对点的基础上实现一个分布式的时间戳服务器，我们需要使用一种类似于亚当·贝克（Adam Back）的哈希现金（Hashcash [6]）所采用的工作量证明系统，而非依赖报纸或新闻组的帖子。该工作量证明机制，涉及去寻找一个特定的值，当这个值被哈希时（例如使用 SHA-256 算法），其结果哈希值会以若干个零比特位开头。所需的平均工作量，与所要求的零比特位的数量成指数级关系，而其验证过程却只需执行一次哈希运算即可。

在我们的时间戳网络中，我们通过在区块中不断递增一个随机数（Nonce）来实现工作量证明，直到找到一个值，能使得该区块的哈希满足所要求的零比特位。一旦付出了CPU 算力使其满足了工作量证明，那么在不重做同等工作量的情况下，该区块就无法被更改。



工作量证明同时也解决了在多数决策中如何体现代表权的问题。如果多数是基于“一个 IP 地址一票”的原则，那么任何能够分配大量 IP 地址的人都可以颠覆它。工作量证明的本质是“一个 CPU 一票”。多数决策由最长的链来代表，因为它投入了最大的工作量证明。如果大部分 CPU 算力由诚实节点控制，那么诚实链的增长速度将是最快的，并将超越任何竞争链。要修改一个过去的区块，攻击者将不得不重做该区块及其后所有区块的工作量证明，然后追上并超过诚实节点的工作。我们稍后将会证明，一个较慢的攻击者能够追上的概率，会随着后续区块的增加而呈指数级递减。

为了应对硬件速度的不断提升以及节点运行兴趣随时间的变化，工作量证明的难度由一个移动平均值来决定，该平均值旨在维持每小时产生一个平均数量的区块。如果区块生成得过快，难度就会增加。

【评注】

在上一节，我们了解到区块通过哈希链接形成了时间链。本节将深入探讨这种链条之所以不可篡改的根本原因：工作量证明 (Proof–of–Work, PoW)，以及它是如何将数字信息与真实世界的物理成本绑定在一起的。

白皮书在此引入了解决方案的第二个关键支柱：工作量证明（Proof–of–Work, PoW）。它为上一节中描述的“时间戳链条”的创建行为，赋予了真实的、不可伪造的物理成本。

在 Bitcoin Core 的实现中，PoW 的过程是矿工节点（通常是专门的 ASIC 矿机）对一个区块头 (CBlockHeader) 进行反复哈希计算的过程。区块头是一个 80 字节的紧凑数据结构，矿工在计算时可以调整其中的一个字段：随机数 (nNonce)。

矿工的目标是，找到一个 nNonce 值，使得对区块头进行两次 SHA-256 哈希（即 SHA256(SHA256(BlockHeader))）计算后，得到的哈希结果必须小于一个特定的目标值 (Target)。这个目标值由区块头中的另一个字段 nBits 以一种紧凑的格式所定义。一个更低的目标值，意味着哈希结果的开头必须有更多的零比特位，因此寻找一个有效哈希的难度就越大。

> [深入代码]
> 这个过程的核心验证逻辑位于 src/pow.cpp 的 CheckProofOfWork 函数中。该函数的作用非常直接：它将一个给定的区块哈希与从 nBits 解码出的目标值进行比较。如果哈希值小于或等于目标值，则证明有效。这个验证过程极其快速和廉价（只需一次哈希比较），但寻找这个有效哈希的过程却极其耗费算力，这正是 PoW“易于验证，难于生成”的非对称特性。

PoW 巧妙地解决了去中心化网络中“女巫攻击”（Sybil Attack）的难题。在传统的“一个 IP 一票”系统中，攻击者可以轻易地伪造大量身份来主导投票。而 PoW 将投票权与现实世界的物理资源——计算能力（算力）——绑定在一起，实现了“一 CPU 一票”（在现代，更准确地说是“一哈希一票”）。

一个节点在网络中的“投票权”或“影响力”，不取决于它有多少个 IP 地址或节点身份，而取决于它贡献了多少算力。白皮书中所说的“多数决策由最长的链来代表”，在代码实现中，更精确地由拥有最大累计工作量 (nChainWork) 的链来代表。nChainWork 是衡量一条链从创世区块到当前区块所投入总算力的指标。因此，网络共识追随的不是最“长”的链，而是最“重”的、凝聚了最多物理工作的链。

只要诚实节点掌握了网络中的大部分算力（即所谓的“51%”），诚实链的增长速度就会在概率上超过任何攻击者的链。要篡改历史，攻击者不仅需要重做历史区块的工作量，还必须用其有限的算力，去追赶上由全球诚实算力共同推动的主链，这在数学上很快变得不可行。

为了应对硬件的飞速发展和网络总算力的波动，比特币协议包含一个至关重要的难度调整机制。白皮书提到“移动平均值”，在代码实现中，这是一个精确的算法。

> [深入代码]
> 难度调整算法位于 src/pow.cpp。
> 该机制规定，每隔 2016 个区块，全网所有节点都会独立地根据过去 2016 个区块的实际出块时间，重新计算下一个周期的难度目标。如果过去 2016 个区块的平均出块时间小于 10 分钟，说明算力增长过快，难度就会增加（目标值会降低）；反之，如果大于 10 分钟，难度就会降低（目标值会升高）。

这个自动调整机制，像一个经济系统中的“中央银行”，但它完全由代码驱动，其目标是确保无论全网算力如何变化，区块的平均产生速度都能稳定在大约 10 分钟一个。这保证了比特币新货币的发行速度是可预测的、稳定的，并且使得交易确认时间有一个稳定的预期。

工作量证明为比特币的“不可篡改性”提供了坚实的物理基础，但一个去中心化系统如何协调如此庞大的算力，确保所有节点都能就同一条权威链达成共识呢？这就需要一个高效且健壮的“网络”机制，这正是下一章将要展开的。

5. 网络（Network）

运行该网络的步骤如下：

1. 新的交易被广播给所有节点。
2. 每个节点将收到的新交易打包进一个区块。

3. 每个节点为其区块寻找一个高难度的工作量证明。
4. 当一个节点找到了工作量证明，它就将该区块广播给所有节点。
5. 仅当区块中所有的交易都有效且未被花费时，节点才接受该区块。
6. 节点通过创建链中的下一个区块来表达它们对该区块的接受，并将这个已被接受区块的哈希作为下一个区块的前一哈希。

节点永远将最长的链视为正确的链，并会持续致力于扩展它。如果两个节点同时广播了下一个区块的不同版本，一些节点可能先收到其中一个，另一些节点则先收到另一个。在这种情况下，节点会基于它们首先收到的那个版本进行工作，但也会将另一条分支保存下来，以防它后来变长。当下一个工作量证明被找到，并且其中一条分支变得更长时，这个平局就会被打破；那些原本在另一条分支上工作的节点，此时将会切换到更长的这条链上。

新交易的广播，不一定需要到达所有的节点。只要它们能到达足够多的节点，它们迟早会被打包进一个区块中。区块的广播对于消息丢失也具有容错性。如果一个节点没有收到某个区块，它会在收到下一个区块并意识到自己遗漏了一个区块时，主动发起请求。

【评注】

前面章节详细阐述了交易的构成和工作量证明如何保障了区块的安全。本节将聚焦于这些独立运作的节点如何通过“网络”相互发现、通信、以及最终达成对唯一区块链历史的“共识”。

白皮书用六个简洁的步骤概括了网络的运行流程。在 Bitcoin Core 的代码实现中，每一步都对应着复杂的 P2P 消息传递和验证逻辑。

[深入代码]

网络的协调主要由 src/net_processing.cpp 模块负责。

交易和区块的传播 (步骤 1, 4):

白皮书描述的“广播”并非一种天真的、将完整数据直接发送给所有节点的方式。为了节省带宽和防止拒绝服务（DoS）攻击，网络采用了一种更高效的

Gossip 协议。

- 当一个节点有新的交易或区块时，它会向其对等节点发送一条小型的 inv (Inventory) 消息。这条消息只包含新数据的哈希值，相当于一个内容目录。
- 接收到 inv 的节点会检查自己是否已经拥有该数据。如果没有，它会向源节点回复一条 getdata 消息，请求获取完整的交易或区块。
- 源节点在收到 getdata 请求后，才会将完整的 tx 或 block 消息发送给请求方。这种“先通知，后索取”的机制，确保了数据只在需要时才被传输，极大地提升了网络效率。现代的 Bitcoin Core 实现中，还引入了**致密区块 (Compact Blocks, BIP 152)** 等更先进的传播机制，进一步降低了区块传播的延迟。

交易池 (Mempool) (步骤 2):

在将交易打包进区块之前，节点会将通过验证的、未确认的交易暂时存放在一个内存区域，这就是交易池 (Mempool)。

[深入代码]

Mempool 的实现位于 (src/txmempool.h/.cpp)。

Mempool 不仅仅是一个简单的暂存区，它有一套复杂的策略，用于管理交易的准入、排序（通常按手续费率）和驱逐，以确保矿工在构建区块时能最高效地选择价值最高的交易。

区块验证与接收 (步骤 5):

当一个节点收到一个新区块时，它会执行一系列极其严格的验证。

[深入代码]

验证过程由 src/validation.cpp 中的 ProcessNewBlock 等核心函数来协调。验证流程大致分为两个阶段：

- **无上下文检查 (CheckBlock):** 节点首先对区块自身进行检查，确保其结构健全，例如工作量证明有效、区块大小合规、时间戳合理等。

- **有上下文检查 (AcceptBlock):** 在区块自身有效后，节点会检查区块内容与现有区块链历史的兼容性。这包括验证区块中的每一笔交易是否合法（例如，引用的 UTXO 是否存在且未被花费），然后才将该区块连接到主链上，并更新 UTXO 集。

白皮书中的“节点永远将最长的链视为正确的链”是比特币共识的核心。在代码实现中，这更准确地说是“累计工作量最大 (nChainWork)”的链。

处理分叉与链重组 (Reorganization, "Reorg"):
当网络中出现分叉（例如，两个矿工几乎同时挖出区块）时，节点会同时追踪所有有效的分支。它会基于自己首先收到的那个区块所在的分支继续工作，但并不会丢弃另一条分支。当其中一条链积累了更多的 PoW，成为最“重”的链时，节点会执行链重组。

链重组示意：

```
旧主链: [Block A] -- [Block B] -- [Block C] (最重，当前主链)
                    |
备选链:               -- [Block B'] -- [Block C'] -- [Block D'] (成为新的最长/最重链)

---- 发生链重组 ----

新主链: [Block A] -- [Block B'] -- [Block C'] -- [Block D']
旧分支:             [Block B] -- [Block C] (被断开，交易返回 Mempool)
```

这个完全自动化的过程，确保了整个去中心化网络最终总能收敛到单一的、权威的历史记录上，从而解决了冲突。

网络的运作看似复杂，但其内在的激励机制才是保证系统持续安全运行的根本。下一章将揭示，矿工为何愿意耗费巨大的算力来维护这个网络，以及这种经济激励如何巧妙地与比特币的安全性绑定在一起。

6. 激励（Incentive）

按照约定，一个区块中的第一笔交易是一笔特殊的交易，它会生成一枚由该区块创建者所拥有的新的电子货币（coin）。这为节点支持网络提供了一种激励，也提供了一种在没有中央权威机构发行货币的情况下，将货币初始分配到流通领域的方式。这种以稳定速率不断增添新货币的方式，类似于金矿矿工消耗资源将黄金注入流通领域。在我们的案例中，被消耗的是 CPU 时间和电力。

这种激励也可以由交易费来提供。如果一笔交易的输出值小于其输入值，其差额就是一笔交易费，这笔费用会被加到包含该交易的区块的激励值中。一旦预定数量的货币全部进入流通，激励机制就可以完全过渡到仅依靠交易费，从而实现完全无通货膨胀。

这种激励机制或许有助于鼓励节点保持诚实。如果一个贪婪的攻击者能够集结比所有诚实节点都多的 CPU 算力，他将不得不在两种选择中做出抉择：一是通过窃回自己的支付款来欺诈他人，二是利用这些算力来生成新的电子货币。他应该会发现，遵守规则比破坏系统和他自身财富的有效性要更为有利可图——这些规则会让他比所有其他人加起来获得更多的新电子货币。

> **【评注】**
>
> 在了解了比特币网络的运作机制后，我们不禁要问，是什么力量驱动着无数矿工投入巨大的物理资源来维护这个看似“无主”的系统？本节将揭示比特币安全模型的经济学基石——激励机制，以及它如何将个体理性与网络整体安全巧妙地结合起来。
>
> 白皮书在此揭示了比特币安全模型的经济学基石。矿工之所以愿意投入巨大的算力和电力成本来寻找工作量证明，是因为有明确的经济回报。这种回报通过一种特殊的交易——Coinbase 交易——来实现。
>
> **Coinbase 交易**是矿工在构建区块时，自己创建并放入的第一笔（且只能有一笔）交易。它在结构上非常独特：
>
> - **特殊的输入：** 普通交易的输入必须引用一个已存在的 UTXO。而

Coinbase 交易的输入是全新的，它不引用任何先前的输出。在代码层面，其输入的 prevout 字段是一个空哈希，索引为-1。这标志着它在“凭空”创造新的比特币，是货币供应的唯一来源。

- **Coinbase Data:** 普通交易的 scriptSig 用于提供签名。而 Coinbase 交易的 scriptSig 则被称为“Coinbase Data”，矿工可以在其中写入任意数据（在一定长度限制内），通常用于标识矿池、包含额外的随机数以辅助挖矿等。

矿工通过 Coinbase 交易所获得的奖励，由两部分构成：

- **区块补贴 (Block Subsidy):** 这是协议规定新创造出来的比特币数量。这个数量被硬编码在协议中，大约每四年（每 210,000 个区块）减半一次。

  [深入代码]
  在 Bitcoin Core 的代码中，有一个专门的函数，如 GetBlockSubsidy(nHeight, consensusParams)，它会根据当前区块的高度 nHeight 来精确计算当前应得的补贴额。这个可预测的、通缩的货币政策，是比特币“数字黄金”特性的核心。
- **交易手续费 (Transaction Fees):** 正如我们在第二节注释中提到的，矿工可以获得其打包区块内所有交易的手续费总和。手续费是每笔交易的输入总金额与输出总金额之间的差额。

矿工将区块补贴和该区块内所有交易的总手续费相加，并将这个总额作为 Coinbase 交易的输出值，支付给自己指定的地址。网络中的所有节点在验证区块时，都会独立地重新计算一遍这个应得的总奖励，并检查 Coinbase 交易的输出值是否不大于这个总和。如果矿工试图给自己支付超过应得的奖励，该区块将被所有诚实节点拒绝，矿工的所有努力和成本都将付诸东流。

白皮书最后一段提出了一个深刻的博弈论观点：遵守规则比攻击系统更有利可图。

一个掌握了大量算力的“贪婪的攻击者”，面临一个理性的经济选择：

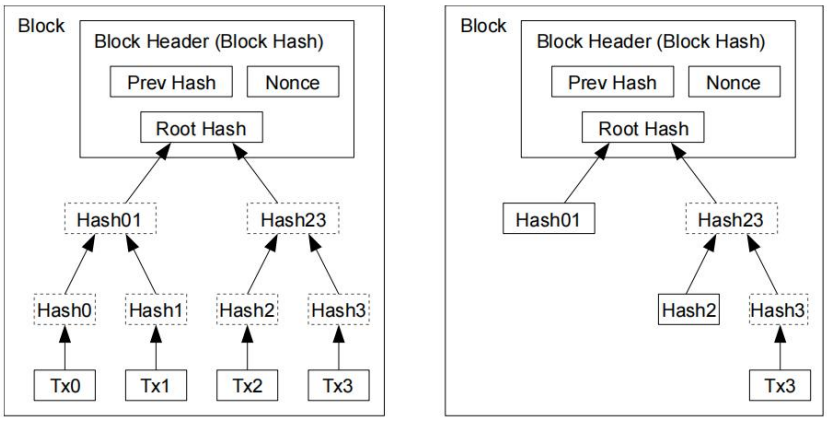
- **选择攻击：** 他可以尝试进行 51%攻击，例如通过制造链重组来双花自己的交易。这种行为会破坏市场对整个系统的信心，导致比特币价格下跌，从而损害他自己持有的比特币以及未来挖矿收益的价值。他窃取的是存量，但摧毁的是未来。
- **选择诚实挖矿：** 他可以利用其算力优势，遵循协议规则来挖矿。由于他拥有算力优势，他将比任何人都更有可能获得区块奖励。他将持续地、稳定地获得比所有其他人加起来更多的比特币。

对于一个理性的、以盈利为目的的矿工来说，长期诚实地参与游戏，分享系统增长的红利，远比通过攻击来获得一次性的、但可能导致整个系统崩溃的收益要明智得多。这种设计，将个体利益与网络整体的健康和安全巧妙地捆绑在了一起。

这种激励机制确保了网络的安全，但随着时间的推移，区块链的数据量会不断增长。如何管理这些庞大的数据，同时又不牺牲去中心化和验证能力？下一章“回收硬盘空间”将给出巧妙的答案。

7. 回收硬盘空间（Reclaiming Disk Space）

一旦一枚电子货币中最新的那笔交易被足够多的区块所掩埋，那么在它之前的那些已花费的交易就可以被丢弃，以节省硬盘空间。为了在不破坏区块哈希的情况下实现这一点，交易被哈希进一棵默克尔树（Merkle Tree [7][2][5]）中，只有树的根节点被包含在区块的哈希里。如此一来，旧的区块便可以通过剪去树的枝干来进行压缩。那些内部的哈希值无需被存储。



Transactions Hashed in a Merkle Tree　　After Pruning Tx0-2 from the Block

一个不含任何交易的区块头大约是 80 字节。如果我们假设每 10 分钟生成一个区块，那么每年产生的数据量为 80 字节×6×24×365=4.2MB。鉴于 2008 年时，计算机系统通常配备 2GB 的内存，并且摩尔定律预测当前每年增长 1.2GB，那么即便区块头必须被保存在内存中，存储也不应成为问题。

【评注】

我们已经深入探讨了交易的构成、链式时间戳的原理、工作量证明的成本以及网络的共识机制。现在，我们将目光转向一个实际问题：随着区块链的不断增长，如何有效地管理其存储空间？本节将揭示默克尔树这一密码学工具的精妙之处，以及它如何实现“回收硬盘空间”的同时不牺牲安全性。

中本聪在此展现了其非凡的远见。他预见到，随着交易量的增长，存储整个区块链的全部历史数据将对节点的硬盘空间构成巨大挑战。如果运行一个全节点变得过于昂贵，就可能导致中心化，从而损害比特币的根基。因此，他提出了一种在不牺牲安全性的前提下，为节点“减负”的机制。

这个机制的核心思想是：对于验证未来的交易而言，我们真正需要知道的，是当前所有未花费的交易输出（UTXO）的集合。那些已经被花费掉的、深埋在历史中的旧交易，对于新交易的验证来说，是不必要的。因此，理论上可以将它们从节点的存储中安全地移除。

直接删除旧交易会面临一个致命问题：它会改变区块的内容，从而导致区块的哈希值改变，进而破坏整个区块链的完整性。

为了解决这个问题，白皮书引入了一个优雅的密码学工具——默克尔树。

工作原理: 默克尔树是一种哈希树。在一个区块中，它不是将所有交易数据直接哈希，而是：

- 将每一笔交易的 ID（txid）作为树的“叶子节点”。
- 成对地哈希相邻的叶子节点，生成它们的父节点。
- 不断重复这个过程，对上一层的节点两两哈希，直到最终只剩下一个哈希值。这个最终的哈希值就是**默克尔树根 (hashMerkleRoot)**。

精妙之处: 区块头 (CBlockHeader) 中存储的不是所有交易的哈希，而仅仅是这个 32 字节的默克尔树根。这个树根就像一个摘要或指纹，它以一种密码学安全的方式，唯一地代表了该区块包含的所有交易。

如何回收空间: 有了默克尔树，节点就可以在丢弃了大部分旧交易数据后，仍然保持区块哈希的完整性。因为区块哈希只依赖于区块头，而区块头只包含了默克尔树根。只要保留默克尔树根，区块的身份就不会改变。这种丢弃旧交易数据的过程，在现代比特币核心中被称为“剪枝模式”（Pruned Mode）。运行在剪枝模式下的节点，可以只保留最近几百个区块的完整数据，而对于更早的区块，只存储区块头，从而将硬盘空间需求从几百 GB 降低到几个 GB。

除了回收硬盘空间，默克尔树更重要一个贡献是，它为下一节将要讨论的简单支付验证（SPV）提供了可能。因为有了默克尔树，一个轻客户端（如手机钱包）无需下载整个区块的交易数据，只需向全节点请求一个简短的“默克尔证明”（Merkle Proof），就能以密码学的方式验证某笔特定的交易确实存在于某个区块中。这个特性是比特币能够拥有轻量级生态系统的关键。

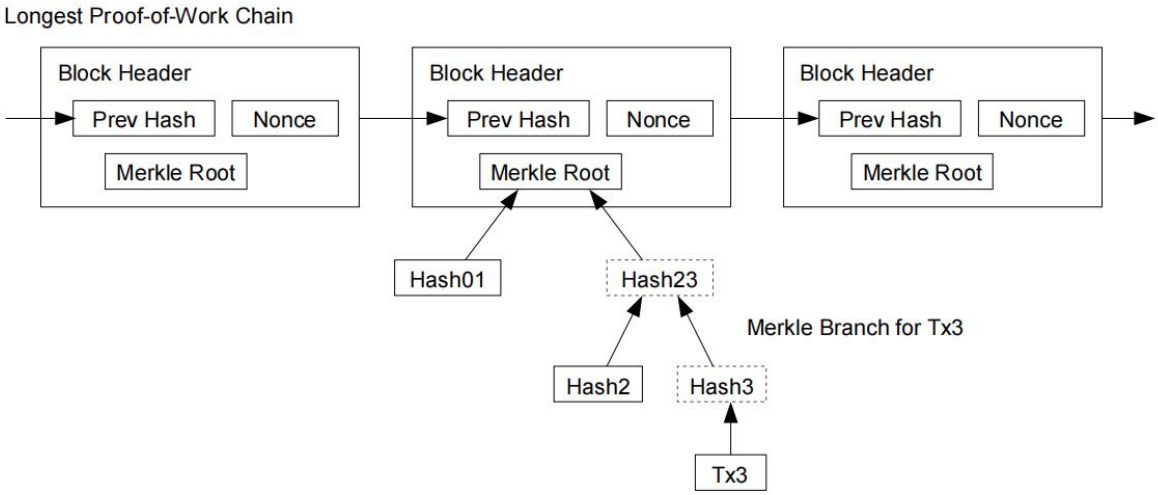
白皮书中对存储需求的估算（每年 4.2MB）在今天看来显得极为乐观，但这

恰恰反衬出中本聪在设计之初就引入默克尔树和 SPV 概念的非凡远见。他早已预料到存储会成为问题，并提前设计好了解决方案。

默克尔树不仅仅解决了存储问题，更是实现轻量级验证的关键。在下一个章节中，我们将看到 SPV 如何让普通用户在不运行完整节点的情况下，依然能够安全地验证自己的支付。

8. 简单支付验证（Simplified Payment Verification）

在不运行一个完整网络节点的情况下，验证支付是可能实现的。一个用户只需保存一份最长工作量证明链的区块头副本，他可以通过查询网络节点来获取这些区块头，直到他确信自己拥有了最长的链。然后，他需要获得将某笔交易链接到其所在时间戳区块的默克尔分支（Merkle branch）。他自己无法核实这笔交易，但通过将它链接到链中的某个位置，他可以看到一个网络节点已经接受了它，并且后续添加的区块进一步确认了整个网络已经接受了它。



因此，只要诚实节点控制着网络，这种验证就是可靠的；但如果网络被一个攻击者所压制，它就会变得较为脆弱。网络节点可以亲自验证交易，而这种简化的方法则可能会被攻击者伪造的交易所欺骗，只要攻击者能持续地压制网络。一种防范策略是，当网络节点侦测到一个无效区块时，接受它们发出的警报，从而提示用户的软件下载完整的区块和被警报的交易，以确认这种不一致性。那些频繁接收付款的商家，为了更独立的安全

性以及更快的验证，很可能仍然希望运行他们自己的节点。

【评注】

前一节我们理解了默克尔树如何优化了全节点的存储。本节将在此基础上，引出一个革命性的概念：简单支付验证（SPV）。它使得普通用户无需承担全节点的所有负担，就能安全地使用比特币。

白皮书在此提出了一个革命性的概念：简单支付验证（Simplified Payment Verification, SPV）。这是对比特币“不可能三角”中“可扩展性”问题的一种解答。其核心思想是，一个用户无需亲自验证所有事情，就可以在一定安全假设下，相信整个网络已经替他完成了验证工作。

SPV 客户端（通常指手机钱包、硬件钱包或某些桌面轻钱包）的运作模式，与运行一个全节点（Full Node）截然不同：

- **全节点:** 下载并验证从创世区块至今的每一个区块和每一笔交易。它们独立地构建整个 UTXO 集，不信任网络中的任何其他节点，只相信代码和共识规则。这是比特币网络安全和去中心化的基石。
- **SPV 客户端:** 不下载完整的区块，只下载区块头链（Block Header Chain）。由于每个区块头只有 80 字节，整个区块头链非常小（几十 MB），完全可以在资源受限的设备上存储。

SPV 客户端通过持有区块头链，就拥有了对整个网络累计工作量的“摘要”。它相信，这条凝聚了最多算力的链，其包含的交易已经被全网的诚实节点充分验证过了。这是一种信任的传递：SPV 客户端信任诚实的矿工和全节点社区。

SPV 客户端如何验证一笔与自己相关的特定交易确实存在于某个区块中呢？这就要用到我们在上一节讨论的默克尔树。

其工作流程如下：

1. SPV 客户端（例如，Bob 的手机钱包）对自己感兴趣的地址设置一个

“布隆过滤器”（Bloom Filter），并将这个过滤器发送给它所连接的全节点。

2. 当全节点收到一个包含与该过滤器匹配的交易的区块时，它会向 SPV 客户端发送两条信息：
    - 该区块的区块头。
    - 一个**默克尔证明 (Merkle Proof)**，也称为默克尔分支（Merkle Branch）。
3. 这个默克尔证明，只包含了从 Bob 那笔特定交易的哈希（叶子节点）到默克尔树根路径上所必需的、与之配对的哈希值。
4. Bob 的钱包收到这些信息后，可以独立地执行计算：它用自己的交易哈希和默克尔证明中提供的哈希值，一步步向上计算，最终得出一个默克尔树根。
5. 最后，它将自己计算出的默克尔树根，与收到的区块头中的 hashMerkleRoot 字段进行对比。如果两者完全一致，Bob 的钱包就能以极高的密码学确定性，证明他的这笔交易确实被包含在了那个区块中，并被整个网络所接受。

白皮书非常诚实地指出了 SPV 模型的安全权衡：“只要诚实节点控制着网络，这种验证就是可靠的；但如果网络被一个攻击者所压制，它就会变得较为脆弱。”

- **信任假设:** SPV 客户端的核心安全假设是，它所连接到的最长工作量证明链是由诚实的多数算力构建的。它无法独立验证区块内的所有交易是否都符合共识规则（例如，它不知道某个输入是否已经被双花）。它相信矿工不会打包一个包含无效交易的区块，因为那样会使其区块奖励作废。
- **潜在风险:** 一个拥有足够算力的攻击者（51%攻击者）可以创建一个包含无效交易（例如，一笔双花交易）的区块，并将其构建成最长链。SPV 客户端由于只检查区块头和默克尔证明，它会接受这笔无效交易，

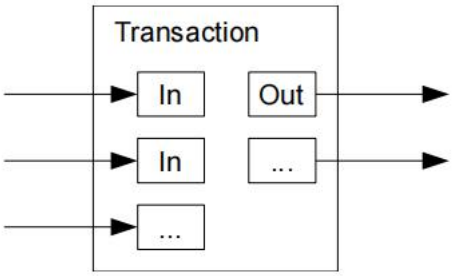
因为它看起来被合法地打包进了一个拥有有效 PoW 的区块中。而全节点则会立刻识别出该区块的无效性并拒绝它。

- **防范策略:** 白皮书提出的“接受警报”机制，在实践中演变成了 SPV 客户端连接多个不同的全节点，并交叉验证信息。如果从不同节点收到的关于最长链的信息不一致，就表明可能存在问题。尽管如此，对于需要最高安全性的场景（如商家、交易所），白皮书的建议依然是黄金法则：“可能仍然希望运行他们自己的节点”。

SPV 的出现极大地降低了用户参与比特币网络的门槛。但是，我们前面一直在讨论“电子货币”的流转和验证，还没有详细说明这些“数字钞票”如何像实体钞票一样，进行自由的“合并”和“分割”。下一节将揭示 UTXO 模型在这个方面的巧妙设计。

9. 合并与分割价值（Combining and Splitting Value）

虽然可以单独地处理每一枚电子货币，但若要为一次转账中的每一分钱都创建一笔单独的交易，那将是笨拙不堪的。为了允许价值被分割和合并，交易可以包含多个输入和多个输出。通常情况下，一笔交易要么是来自一笔较大额前序交易的单个输入，要么是合并了多个较小金额的多个输入；同时，最多会有两个输出：一个用于支付，另一个（若有）用于将找零返回给发送方。



应当指出，那种一笔交易依赖于几笔交易，而这几笔交易又依赖于更多笔交易的扇出（fan-out）情形，在这里不成问题。因为，从来没有必要去提取一份交易历史的完整独立副本。

【评注】

在前一节，我们了解了 SPV 如何让轻量级客户端安全验证交易。本节将从另一个角度深入探讨比特币交易的灵活性：如何像处理实体货币一样，在数字世界中进行价值的“合并”与“分割”，这正是 UTXO 模型的巧妙之处。

白皮书在这一节阐述了比特币交易结构的核心灵活性。其基础，正是我们在前面注释中反复强调的 UTXO（未花费交易输出）模型。

将 UTXO 理解为数字化的实体钞票或硬币，是理解本节内容的关键。你钱包里的比特币，并非一个存储在账户里的总余额数字，而是一系列离散的、具有不同“面额”的 UTXO。

分割价值 (Splitting Value):

这对应于找零 (Change) 的概念。当你用一张 100 元的钞票去买一件 30 元的商品时，你不能把钞票撕开，你必须付出整张 100 元，然后收回 70 元的找零。

在比特币中，如果你想支付 0.3 BTC，但你只有一个面额为 1 BTC 的 UTXO，你的钱包软件会自动构建一笔交易，该交易有：

- 一个输入: 你那 1 BTC 的 UTXO。
- 两个输出:
    - 一个 0.3 BTC 的输出，支付给收款方。
    - 一个约 0.7 BTC 的“找零输出”（减去交易费后），支付回给你自己的一个新地址。

这个找零输出，就是一个新的、属于你的 UTXO，你可以在未来的交易中花费它。

合并价值 (Combining Value):

这对应于凑钱的概念。当你需要支付 100 元，但钱包里只有一张 50 元、两张 20 元和一张 10 元的钞票时，你会把这些钞票凑在一起支付。

在比特币中，如果你想支付 1 BTC，但你只有两个 0.4 BTC 和一个 0.3 BTC

的 UTXO，你的钱包软件会自动构建一笔交易，该交易有：

- 三个输入: 你那两个 0.4 BTC 和一个 0.3 BTC 的 UTXO。
- 一个或两个输出:
  - 一个 1 BTC 的输出，支付给收款方。
  - 一个约 0.1 BTC 的找零输出（如果需要找零的话）。

这种包含多个输入和多个输出的交易结构，赋予了比特币 UTXO 模型与实体货币同等的灵活性，使其能够轻松应对任何金额的支付场景。

钱包软件在构建交易时，如何从一堆 UTXO 中挑选出合适的组合来支付，这是一个复杂且重要的问题，被称为“币选择”（Coin Selection）。这是一个在白皮书中没有提及，但在钱包技术的实际实现中至关重要的算法。

一个好的币选择算法，需要在多个目标之间进行权衡：

- **最小化交易费:** 交易费与交易的体积（字节数）正相关。输入和输出越多，交易体积越大。因此，算法会倾向于使用更少的输入来构建交易。
- **保护隐私:** 如果一个交易合并了大量来自不同来源的 UTXO，它可能会在区块链上暴露这些地址都属于同一个所有者。因此，算法有时会避免合并“不相关”的 UTXO。
- **避免粉尘输出:** 算法会尽量避免产生金额极小（相对于花费它的手续费而言）的找零 UTXO，因为这些“粉尘”在未来可能变得不经济而无法花费。

  [深入代码]
  现代钱包使用了各种复杂的策略，如“分支界定法”（Branch and Bound）、随机选择等，来动态地解决这个多目标的优化问题。

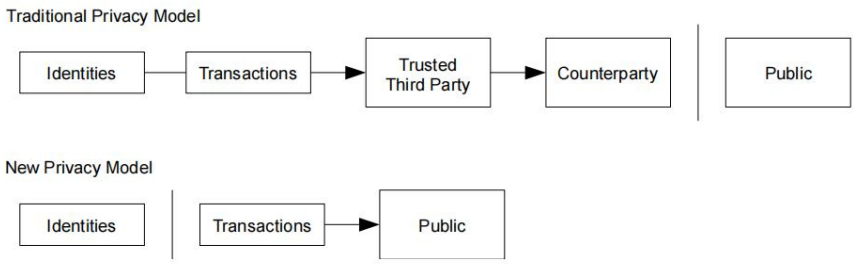
白皮书最后一段提到的“扇出（fan-out）不成问题”，是在强调 UTXO 模型的另一个优点。由于验证一笔交易只需要检查其直接引用的输入 UTXO 是否

存在于当前的 UTXO 集合中，而无需追溯这些 UTXO 的完整历史来源，因此，即使一笔交易的“祖先”关系极其复杂，也不会增加验证的难度。这保证了系统的验证效率不会随着区块链历史的增长而下降。

理解了比特币如何处理价值的合并与分割后，我们还要关注另一个重要方面：在所有交易都公开的情况下，用户如何保护自己的“隐私”？下一节将探讨比特币独特的隐私模型。

10. 隐私（Privacy）

传统的银行模型，通过将信息的访问权限限制在交易相关方和可信第三方之内，来达到一定程度的隐私。而将所有交易公之于众的必要性，则排除了这种方法。但是，隐私仍然可以通过在另一个环节切断信息流来得到保护：即保持公钥的匿名性。公众可以看到某人正在向另一个人发送一笔金额，但在没有信息将这笔交易与任何人关联起来的情况下，无法得知具体是谁。这类似于证券交易所发布的信息级别，其中每笔独立交易的时间和规模，即所谓的“盘口数据（the tape）”，是公开的，但并不会告知交易双方的身份。



作为一道额外的防火墙，应当为每一笔交易使用一对新的密钥，以防止它们被关联到同一个所有者。对于多输入的交易，某些关联仍然是不可避免的，因为它们必然揭示了其所有输入都归属于同一个所有者。这里的风险在于，如果一个密钥的所有者身份被暴露，那么通过关联分析，可能会揭示出属于该所有者的其他交易。

【评注】

我们已经深入了解了比特币交易的灵活性。然而，当所有交易都公开记录在区块链上时，用户的隐私如何得到保障呢？本节将探讨比特币的“假名”隐私模型，以及它与传统隐私方法的区别。
白皮书在此处清晰地定义了比特币的隐私模型。它不是完全匿名的，而是假名的 (Pseudonymous)。这与传统银行通过“限制信息访问”来实现隐私的方式截然不同。

公开透明的账本: 比特币选择将所有交易都记录在一个公开的、任何人都可以查阅的区块链上。这种透明性是其安全和可验证性的基础，但也意味着交易本身是无法隐藏的。

匿名的公钥: 隐私保护的环节被放在了“身份关联”上。交易的参与方不是由真实姓名或身份来标识，而是由一串字母和数字组成的地址（其背后是公钥）来代表。只要这个地址没有与用户的真实身份相关联，那么理论上，交易就是匿名的。

白皮书用了一个非常贴切的比喻：证券交易所的盘口数据 (the tape)。我们能看到在什么时间、以什么价格成交了多少股股票，但我们不知道买卖双方是谁。同样，在比特币区块链上，我们能看到哪个地址向哪个地址发送了多少比特币，但我们不知道这些地址背后的控制者是谁。

中本聪在白皮书中提出了一个至今仍然是比特币隐私保护核心原则的最佳实践：**为每一笔交易使用一个新的密钥对（即新地址）**。

为何要用新地址: 如果一个用户反复使用同一个地址来接收付款，那么所有与该地址相关的交易都会被轻易地链接在一起。通过区块链分析，任何人都可以计算出这个地址的总收入、总支出、当前余额以及所有的交易对手方。这就像在所有商业活动中都使用同一个公开的银行账号。而为每次收款都使用一个新地址，就像为每笔收入都开一个新的一次性银行账户，这会极大地增加追踪资金流动的难度。我们之前分析的 HD 钱包 (BIP 32/44)，正是为

实现这一实践提供了强大的技术支持，它可以从一个主种子中派生出几乎无限数量的地址，而无需用户手动管理。

不可避免的关联（链接性）：白皮书同样诚实地指出了这个模型的局限性。当一笔交易包含多个输入时，它必然会向全世界揭示一个信息：所有这些被花费的输入（UTXO）都属于同一个所有者。这就是所谓的“共同输入所有权启发式”（Common-input-ownership Heuristic），它是区块链分析公司追踪资金流动的最基本、最强大的工具之一。

身份暴露的风险: 最大的风险在于，一旦任何一个地址与用户的真实世界身份发生关联（例如，用户在一个需要 KYC 的交易所使用了这个地址，或者在社交媒体上公开了它），那么通过上述的链接性分析，就可能追溯出该用户拥有过的其他地址，从而揭开其更多交易历史的面纱。

自白皮书发布以来，社区为了增强比特币的隐私性，发展出了许多先进的技术和方法，这些都建立在白皮书提出的基本模型之上。

- **混币服务 (CoinJoin):** 一种将来自多个用户的输入和输出混合在一笔大交易中的技术，旨在打破“共同输入所有权”的假设，混淆资金的真实流向。
- **Taproot 升级:** 我们分析过的 Taproot 升级，通过让复杂的多签或智能合约交易在链上看起来与普通的单签名交易无异，极大地提升了隐私性。
- **二层网络 (Layer 2):** 闪电网络等二层解决方案，将大量的小额交易移到链下进行，只有最终的结算才记录在主链上，从而减少了公开暴露的交易信息。

这些发展表明，比特币的隐私是一个持续演进、不断进行攻防博弈的领域。

理解了比特币的隐私模型后，我们最后将回到其安全性——从数学角度量化这种基于概率的安全性。下一节“计算”将用严谨的数学模型，来分析攻击者

攻破网络的难度，并解释“6 个确认”这一黄金法则的由来。

11. 计算（Calculations）

我们来思考这样一种场景：一个攻击者试图生成一条比诚实链更快的替代链。即便他成功了，这也不会让系统可以被任意修改，例如凭空创造价值或拿走从不属于攻击者的钱。节点不会接受一笔无效的交易作为支付，而诚实的节点也永远不会接受一个包含无效交易的区块。一个攻击者所能尝试的，仅仅是更改他自己的一笔交易，以拿回他最近刚刚花掉的钱。

诚实链与攻击者链之间的竞赛，可以被描述为一个二项随机游走（Binomial Random Walk）。成功事件是诚实链被扩展了一个区块，其领先优势增加$+1$；失败事件是攻击者链被扩展了一个区块，差距缩小-1。

攻击者从一个给定的落后差距追赶上来的概率，类似于一个赌徒破产问题（Gambler's Ruin problem）。假设一个拥有无限信贷的赌徒从亏损状态开始，进行无限次的尝试，试图达到收支平衡。我们可以计算他最终能达到收支平衡的概率，或者说，一个攻击者最终能追上诚实链的概率，如下所示 [8]:

p = 一个诚实节点找到下一个区块的概率

q = 攻击者找到下一个区块的概率

q_z = 攻击者从落后 z 个区块的情况下最终追上来的概率

$$q_z = \begin{cases} 1 & \text{if } p \le q \\ (q/p)^z & \text{if } p > q \end{cases}$$

鉴于我们 $p > q$ 的假设，随着攻击者需要追赶的区块数量增加，该概率会呈指数级下降。由于胜算不利，如果他不能在早期幸运地向前猛冲，那么随着他落后得越来越远，他成功的机会将变得微乎其微。

现在我们来考虑，一笔新交易的接收方需要等待多久，才能充分确信发送方无法更改这笔交易。我们假设发送方是一个攻击者，他想让接收方在一段时间内相信自己已经付了款，然后在一段时间过去后，再将交易改回支付给他自己。当这种情况发生时，接收方会收到警报，但发送方希望那时为时过晚。

接收方生成一对新的密钥，并在签署前不久将公钥提供给发送方。这可以防止发送方提前准备一条区块链，即通过持续不断地运算，直到他足够幸运地领先了很多，然后在那个时刻执行交易。一旦交易被发送出去，这个不诚实的发送方就开始秘密地在一条平行的、包含其交易替代版本的链上进行运算。

接收方等待，直到这笔交易被添加到一个区块中，并且其后又链接了 z 个区块。他不知道攻击者取得了多少确切的进展，但假设诚实区块花费了每个区块的平均期望时间，那么攻击者的潜在进展将是一个泊松分布（Poisson distribution），其期望值为：

$$\lambda = z\frac{q}{p}$$

为了得到攻击者现在仍然能够追上的概率，我们将他可能取得的每一种进展量的泊松密度，乘以他从那个点能够追上的概率，然后求和…

$$\sum_{k=0}^{\infty} \frac{\lambda^k e^{-\lambda}}{k!} \cdot \begin{cases} (q/p)^{(z-k)} & if\ k \le z \\ 1 & if\ k > z \end{cases}$$

重新整理以避免对分布的无限尾部进行求和…

$$1 - \sum_{k=0}^{z} \frac{\lambda^k e^{-\lambda}}{k!} \left(1 - (q/p)^{(z-k)}\right)$$

用于运行模拟的 C 代码…

```
#include <math.h>
double AttackerSuccessProbability(double q, int z)
{
    double p = 1.0 – q;
    double lambda = z * (q / p);
    double sum = 1.0;
    int i, k;
    for (k = 0; k <= z; k++)
    {
        double poisson = exp(–lambda);
        for (i = 1; i <= k; i++)
            poisson *= lambda / i;
        sum –= poisson * (1 – pow(q / p, z – k));
    }
    return sum;
}
```

运行一些结果，我们可以看到概率随着 z 值的增加而呈指数级下降。

```
q=0.1
z=0     P=1.0000000
z=1     P=0.2045873
z=2     P=0.0509779
z=3     P=0.0131722
z=4     P=0.0034552
z=5     P=0.0009137
z=6     P=0.0002428
z=7     P=0.0000647
z=8     P=0.0000173
z=9     P=0.0000046
z=10    P=0.0000012

q=0.3
z=0     P=1.0000000
z=5     P=0.1773523
z=10    P=0.0416605
z=15    P=0.0101008
z=20    P=0.0024804
z=25    P=0.0006132
z=30    P=0.0001522
z=35    P=0.0000379
z=40    P=0.0000095
z=45    P=0.0000024
z=50    P=0.0000006
```

求解 P < 0.1% 的情况...

```
P < 0.001
q=0.10   z=5
q=0.15   z=8
q=0.20   z=11
q=0.25   z=15
q=0.30   z=24
q=0.35   z=41
q=0.40   z=89
q=0.45   z=340
```

【评注】

我们在前面各章节详细解析了比特币的各项技术机制，从交易结构到网络共识，再到隐私保护。本节将回到最初的问题：比特币的安全保证究竟有多可靠？我们将用严谨的数学语言，量化攻击者成功篡改交易的概率，从而理解为何“6 个确认”成为行业普遍接受的安全标准。

在进行数学分析之前，白皮书首先严格界定了攻击者所能尝试的目标。这是一个非常重要的前提。

攻击者不能做什么:

一个攻击者，即使拥有强大的算力，也不能凭空创造比特币，或者花费不属于他的钱。这是因为每一笔交易的有效性都由网络中的所有全节点独立地进行密码学验证。任何不符合共识规则（如无效签名、凭空创造价值等）的交易，都会被诚实节点无情地拒绝，包含这种交易的区块也同样会被拒绝。

攻击者唯一能尝试的:

攻击者唯一能做的，是尝试更改他自己的交易历史，具体来说，就是进行双重花费。他可以先向商家支付一笔款项，在商家发货后，再秘密地构建一条不包含这笔支付交易的、更长的替代链（或者将这笔支付改回支付给自己），然后广播出去，试图让网络接受他的替代链为新的事实。

白皮书将诚实链与攻击者链之间的竞赛，巧妙地建模为一个经典的概率论问题：**二项随机游走 (Binomial Random Walk)**。

模型设定:

- p: 诚实网络找到下一个区块的概率，代表了诚实算力占全网的比例。
- q: 攻击者找到下一个区块的概率，代表了攻击者算力占全网的比例 (p+q=1)。
- z: 攻击者开始攻击时，已经落后于诚实链的区块数量。

与赌徒破产问题的类比:

这个问题被进一步类比为赌徒破产问题 (Gambler's Ruin Problem)。攻击者就像一个试图从亏损 z 元的状态回本的赌徒。每一次出块，就像一轮赌局。如果诚实网络赢了（概率 p），赌徒的亏损就增加到 z+1；如果攻击者赢了（概率 q），亏损就减少到 z−1。

核心结论:

白皮书给出了攻击者最终能追上（即赌徒能回本）的概率公式：$q_z = (q/p)^z$ (在 $p>q$ 的前提下)。这个简单的公式揭示了一个深刻的结论：只要诚实算力占优（$p>q$，即 $q<0.5$），攻击者成功的概率会随着他需要追赶的区块数量 z 的增加而呈指数级下降。

为了将上述理论模型应用于实际场景（即“接收方需要等待多久”），白皮书引入了**泊松分布 (Poisson distribution)**。

为何引入泊松分布:

在现实中，区块的产生不是一个固定的时钟，而是一个随机过程。泊松分布非常适合用来描述在一段平均时间内，某个随机事件（如找到一个区块）发生的次数。

计算逻辑:

当接收方等待了 z 个区块的确认后，他不知道攻击者在这段时间内秘密挖了多少个区块。白皮书假设，攻击者挖出的区块数量 k，服从一个期望值为 $\lambda = z \cdot (q/p)$ 的泊松分布。

最终的计算公式，综合了所有可能的情况：对于攻击者可能挖出的每一个区块数 k，计算出这种情况发生的概率（泊松密度），再乘以攻击者从那个新的差距$(z-k)$追赶上来的概率，然后将所有可能的情况求和。

“6 个确认”的数学依据:

白皮书最后给出的计算结果表格，为“6 个确认”这一行业惯例提供了强有力的数学支持。例如，在攻击者拥有 10%算力（q=0.1）的情况下，当有 6 个区块确认后（z=6），他成功的概率已经下降到约 0.024%，即万分之二点四。这个概率已经低到在大多数商业场景中可以被认为是足够安全的。

这整个第十一节，用严谨的数学语言，将比特币的安全性从一个模糊的“安全”概念，量化为了一个可以计算和评估的、具体的概率值。它雄辩地证明了，只要诚实算力占据主导地位，比特币的共识机制就是稳固可靠的。

通过严谨的数学分析，我们对比特币的安全性有了量化的理解。现在，我们可以将白皮书中的所有核心概念和技术细节汇聚起来，得出最终的结论，并对整个比特币系统进行一次全面的概括。

12. 结论（Conclusion）

我们提出了一个无需依赖信任的电子交易系统。我们从常规的、由数字签名构成的电子货币框架出发，这个框架提供了对所有权的强有力控制，但在没有方法防止双重花费的情况下，它是不完整的。为了解决这个问题，我们提出了一个点对点网络，它使用工作量证明来记录一个公开的交易历史。如果诚实节点控制了大部分的 CPU 算力，那么攻击者想要更改这个历史，在计算上将很快变得不可行。该网络因其非结构化的简洁性而显得格外强大。节点可以随时离开和重新加入网络，只需将最长的工作量证明链作为它们离开期间所发生事件的证明即可。

【评注】

至此，我们已经完整地剖析了比特币白皮书的每一个核心章节，从电子现金的需求，到如何通过 UTXO、时间戳、工作量证明、P2P 网络和经济激励来解决双重花费，再到存储优化、轻量级验证和隐私考量，最后是严谨的安全性计算。结论部分，是对整个系统设计哲学的一次高度概括和重申，它将我们之前讨论的所有技术组件，再次串联成一个有机的整体。

从问题到方案的回顾:

中本聪再次强调了整个设计的逻辑起点：从一个基于数字签名的、不完整的电子货币

框架开始，其核心缺陷是无法内在地防止双重花费。随后，他重申了解决方案的几大支柱：一个点对点网络作为基础，使用工作量证明作为安全和共识机制，共同维护一个公开的、难以篡改的交易历史（即区块链）。

“计算上不可行”的安全性:

结论中再次强调，比特币的安全性不是绝对的，而是“计算上不可行 (computationally impractical)”的。这意味着篡改历史在理论上是可能的，但在实践中，只要诚实的多数算力存在，攻击者需要付出的计算成本将是天文数字，使得攻击在经济上变得毫无意义。这是一种务实而强大的安全模型。

“非结构化的简洁性”:

这是对整个系统设计美学的点睛之笔。比特币网络之所以强大（robust），恰恰在于它的“简单”。

- **无中心的结构:** 网络中没有“主节点”、“超级节点”或任何特殊角色的节点。所有全节点都遵循完全相同的规则，执行完全相同的任务。这种扁平化的、非结构化的设计，使得网络没有任何单点故障。任何节点的加入或离开，都不会对整个网络的运行产生实质性影响。
- **简单的规则:** 整个网络的协同，依赖于几条极其简单的、所有参与者都共同遵守的规则：
    - 验证所有交易和区块的有效性。
    - 将收到的有效交易和区块广播给对等节点。
    - 永远在累计工作量最大的链（最长链）上进行扩展。

      正是这些简单的、局部执行的规则，通过 P2P 网络涌现出了一个宏观上高度有序、能够达成全球共识的复杂系统。这是一种典型的“涌现秩序”（Emergent Order），也是复杂性科学的迷人体现。

结论的最后一句话，点明了系统的开放性和自我修复能力。

- **无需许可 (Permissionless):** 任何人都可以下载开源的 Bitcoin Core 软件，在不经过任何机构批准的情况下，自由地加入网络成为一个对等节点。同样，任何节点也可以随时离线。

- **同步机制:** 当一个节点重新上线时，它如何知道自己“离开期间所发生事件的证明”？它会连接到其他对等节点，请求获取区块头链，并根据“最长工作量证明链”的规则，识别出哪条是权威的历史。然后，它会下载并验证自它上次离线以来的所有新区块，最终使自己的账本状态与整个网络同步。这个过程完全是自动的，确保了即使在节点不断进出的动态环境中，网络也能保持数据的一致性和完整性。

感谢您与我一同深入探索了中本聪的这篇开创性论文。希望这份实现级的评注，能为您提供一幅更加清晰和深入的比特币世界地图。

参考文献（References）

[1] W. Dai, "b-money," http://www.weidai.com/bmoney.txt, 1998.

戴伟（W. Dai），“b-money”，http://www.weidai.com/bmoney.txt，1998 年。

[2] H. Massias, X.S. Avila, and J.-J. Quisquater, "Design of a secure timestamping service with minimal trust requirements," In 20th Symposium on Information Theory in the Benelux, May 1999.

H. Massias, X.S. Avila, and J.-J. Quisquater，“一种具有最低信任要求的安全时间戳服务设计”，见《第 20 届比荷卢信息论研讨会论文集》，1999 年 5 月。

[3] S. Haber, W.S. Stornetta, "How to time-stamp a digital document," In Journal of Cryptology, vol 3, no 2, pages 99-111, 1991.

S. Haber, W.S. Stornetta，“如何为数字文档打上时间戳”，见《密码学杂志》，第 3 卷，第 2 期，第 99–111 页，1991 年。

[4] D. Bayer, S. Haber, W.S. Stornetta, "Improving the efficiency and reliability of digital time-stamping," In Sequences II: Methods in Communication, Security and Computer Science, pages 329-334, 1993.

D. Bayer, S. Haber, W.S. Stornetta，“提升数字时间戳的效率与可靠性”，见《序列 II：通信、安全与计算机科学中的方法》，第 329–334 页，1993 年。

[5] S. Haber, W.S. Stornetta, "Secure names for bit-strings," In Proceedings of the 4thACM Conference on Computer and Communications Security, pages 28-35, April 1997.

S. Haber, W.S. Stornetta，“比特串的安全命名”，见《第四届 ACM 计算机与通信安全会议论文

集》，第 28–35 页，1997 年 4 月。
[6] A. Back, "Hashcash – a denial of service counter–measure,"
http://www.hashcash.org/papers/hashcash.pdf, 2002.
亚当·贝克（A. Back），“哈希现金——一种拒绝服务攻击的对抗手段”，
http://www.hashcash.org/papers/hashcash.pdf，2002 年。
[7] R.C. Merkle, "Protocols for public key cryptosystems," In Proc. 1980 Symposium on Security and Privacy, IEEE Computer Society, pages 122–133, April 1980.
拉尔夫·默克尔（R.C. Merkle），“公钥密码系统的协议”，见《1980 年安全与隐私研讨会论文集》，IEEE 计算机学会，第 122–133 页，1980 年 4 月。
[8] W. Feller, "An introduction to probability theory and its applications," 1957.
威廉·费勒（W. Feller），《概率论及其应用导论》，1957 年。

评注附录 (Appendices)

评注附录 A：重大协议升级 (Major Protocol Upgrades)

比特币协议并非一成不变。自白皮书发布以来，它通过一系列设计精巧的软分叉（Soft Forks）不断演进，以修复漏洞、提升性能和增加新功能。本附录将深入剖析其中最重要的几次升级。

1. 隔离见证 (Segregated Witness, SegWit)

SegWit（BIP 141, 143, 144）是比特币历史上最重要、影响最深远的一次升级，于 2017 年激活。它不仅修复了一个底层的严重缺陷，更为比特币的未来发展铺平了道路。

在 SegWit 之前，一笔交易的唯一标识符（txid）是通过对该交易的所有部分（包括签名 scriptSig）进行哈希计算得出的。问题在于，比特币使用的 ECDSA 签名算法，其签名可以有多种不同但密码学上都有效的编码方式。这意味着，任何网络节点都可以在交易被确认前，轻微改动其签名编码，从而在不改变交易逻辑的情况下，改变交易的 txid。

这个“txid 可变”的缺陷，对于依赖于稳定 txid 进行链式交易的二层协议（如闪电网络）是致命的。

SegWit 的核心思想是：签名数据（即“见证”信息）对于交易的**效果**是必要的，但对于定义交易的**内容**本身并非必要。

因此，SegWit 将签名和公钥等“见证”数据，从原始交易结构中移出，放到了一个全新的、独立的见证数据结构 (scriptWitness) 中。这个见证数据结构与交易数据一同被传输和存储，但它不参与传统 txid 的计算。

通过这个分离，txid 的计算只依赖于交易的“内容”（输入、输出、金额等），不再受签名数据可变性的影响，从而变得不可篡改。同时，系统引入了一个新的标识符 wtxid（Witness TXID），它对包含见证数据的完整交易进行哈希，用于确保见证数据自身的完整性。

SegWit 的部署是一个软分叉，它通过一个聪明的“障眼法”实现了向后兼容。一个 P2WPKH（原生 SegWit）输出的 scriptPubKey 看起来是这样的：OP_0 <20-byte public key hash>。

- **旧节点的视角:** 旧节点不理解这个模式，会认为这是一个任何人都可以花费的输出，因此不会阻止它。
- 新节点的视角: SegWit 兼容的节点看到这个模式，就会将其识别为一个 SegWit V0 脚本，并去交易的 scriptWitness 部分查找并验证签名。
  这种机制巧妙地利用了旧节点的规则，将新的、更严格的规则隐藏在了见证数据中，只有新节点才能看到并执行。

变相的区块大小提升: SegWit 引入了区块重量（Block Weight）的概念。非见证数据的每个字节计为 4 个重量单位，而见证数据只计为 1 个。由于见证数据通常占交易体积的很大一部分，这种“折扣”激励了矿工打包更多 SegWit 交易，使得一个区块在物理大小约 2MB 的情况下，可以容纳接近 4MB 的“重量”，从而实现了变相的扩容。

脚本版本控制 (Script Versioning): SegWit 引入了见证程序的版本号（如 OP_0 代表 V0）。这使得未来的脚本升级变得极其简单，开发者可以定义一个新的版本号（如 OP_1）并赋予其全新的验证规则，而无需再设计复杂的软分叉技巧。

2. Taproot (BIP 340, 341, 342)

Taproot 是继 SegWit 之后最重要的升级，于 2021 年激活。它极大地提升了比特币的隐私性、效率和智能合约的灵活性，其核心是**施诺尔签名 (Schnorr Signatures)** 和 **MAST (默克尔化抽象语法树)**。

Taproot 用施诺尔签名取代了比特币原有的 ECDSA 签名。施诺尔签名最大的优点是其线性特性。这意味着，多个参与方可以将他们的公钥聚合成一个单一的“聚合公钥”，并为一笔交易共同创建一个单一的“聚合签名”。

- **极致的隐私性:** 这是 Taproot 最革命性的贡献。无论是简单的单人支付，还是复杂的多重签名、或者包含时间锁的智能合约，只要所有参与方能够达成合作（即“协作路径”），他们就可以共同创建一个聚合签名来花费 UTXO。在区块链上，这笔花费交易看起来与最简单的 P2PKH 交易没有任何区别。只有当参与方之间出现分歧，需要执行合约的备用或争议路径时，其背后的复杂脚本才会被揭示出来。这被称为“隐藏在合作中的合约”。
- **显著的效率提升:** 对于多重签名交易，原先需要 N 个公钥和 N 个签名，占据大量区块空间。在 Taproot 下，无论多少人参与，都只需要一个聚合公钥和一个聚合签名。这大大减少了交易的体积和验证成本，尤其对于大型机构和交易所等需要复杂多签的用户来说，能节省大量手续费。
- **更强大的脚本能力 (Tapscript):** Taproot 结合了 MAST (Merkleized Abstract Syntax Tree) 的思想。一个 UTXO 可以被锁定到一棵包含多种不同花费条件（脚本）的默克尔树上。花费时，只需提供满足其中一个条件的证明（脚本本身和通往默克尔根的路径）即可。这使得开发者可以构建极其复杂、包含大量分支条件的智能合约，而链上只需存储一个 32 字节的默克尔根，极大地提升了脚本的效率和灵活性。

总结: 如果说 SegWit 为比特币的未来升级搭建了“高速公路系统”（脚本版本控制），那么 Taproot 就是在这条高速公路上行驶的第一辆“超级跑车”，它将比特币的隐私和智能合约能力提升到了一个全新的高度。

评注附录 B：比特币的演进与分叉 (Bitcoin's Evolution and Forks)

关于如何扩展比特币以承载更多用户和交易的争论，是社区历史上最重大的分歧，并直接导致了

多个著名硬分叉（Hard Forks）项目的诞生。硬分叉是一种不向后兼容的协议升级，它会创建一条全新的、与原链分离的区块链。本附录旨在简要介绍这些主要分叉项目的动机和技术差异。

1. 比特币现金 (Bitcoin Cash, BCH)

- **分叉时间:** 2017 年 8 月 1 日
- **主要动机:** BCH 的诞生源于对比特币扩容路线图的根本性分歧。其支持者（通常被称为“大区块支持者”）认为，比特币应该通过简单直接地增加基础层（Layer 1）的区块大小限制来提升交易吞吐量，以维持低廉的链上手续费。他们坚信这才是实现白皮书中所描述的“点对点的电子现金系统”愿景的正确道路。他们对 SegWit 和闪电网络等二层扩容方案持怀疑态度，认为这些方案偏离了中本聪的初衷，且过于复杂。
- **关键技术差异:**
  - **区块大小限制:** 这是最核心的区别。BCH 在分叉时立即将区块大小上限从 1MB 提高到了 8MB，后来进一步提升至 32MB，旨在直接提升链上交易容量。
  - **移除 SegWit:** BCH 社区拒绝了 SegWit 升级，并从其代码库中移除了所有相关代码。他们认为 SegWit 是一个不必要的、过于复杂的协议修改，并且其“重量折扣”的经济模型不合理。
  - **不同的难度调整算法:** 为了在算力较低时能够稳定出块，BCH 引入了一种更灵活的紧急难度调整（EDA）算法，后来又升级为更平滑的 ASERT 算法。
  - **增强的脚本功能:** BCH 引入了一些新的操作码以增强脚本功能，例如 OP_CHECKDATASIG，它允许脚本验证来自交易外部数据的签名，为预言机（Oracles）和更复杂的链上合约打开了大门。
- **哲学与愿景:** BCH 的哲学核心是大规模链上扩容。他们致力于将 BCH 打造成一种高交易量、低手续费的全球性日常交易媒介，专注于“支付”这一核心用例。他们的口号是“点对点的电子现金”，强调其作为一种实用货币的属性。

2. Bitcoin SV (BSV)

- **分叉时间:** 2018 年 11 月 15 日 (从 BCH 分叉)
- **主要动机:** BSV（Satoshi Vision，中本聪愿景）的诞生，源于其支持者认为 BCH 的发展方向仍然不够纯粹和激进，未能完全回归和实现他们所理解的“中本聪的原始协议”。他们主张

彻底解除所有协议限制，将协议“锁定”在一个稳定的版本上，让网络通过市场力量自由发展。

- **关键技术差异:**
  - **无限的区块大小:** BSV 采取了最激进的策略，完全解除了区块大小的硬上限（或将其设置到一个极大的值）。其理念是，区块大小应由矿工根据市场需求和自身处理能力来决定，而非由协议开发者硬性规定。这使得 BSV 网络上出现过 GB 级别的巨大区块。
  - **协议锁定 (Protocol Lockdown):** BSV 主张将比特币协议恢复到他们认为是中本聪最初设计的状态，并在此基础上保持极大的稳定性，不再进行大的协议规则修改。他们认为一个稳定的协议是企业和开发者在其上构建应用的基础。
  - **恢复原始脚本:** BSV 恢复了所有在比特币早期被中本聪禁用的原始脚本操作码，并极大地增加了脚本的复杂度上限，以支持任意复杂的链上应用。
- **哲学与愿景:** BSV 的愿景是“Metanet”——一个建立在 BSV 区块链之上的、全球统一的、可承载一切数据和商业活动的价值互联网。他们认为，通过无限扩容和稳定的协议，BSV 不仅可以成为货币，更可以成为整个互联网的底层数据账本，从文件存储到 token 化，再到复杂的企业级应用，都可以在链上以极低的成本完成。

评注附录 C：术语表 (Glossary)

- **51%攻击 (51% Attack):** 一种针对比特币等基于工作量证明的区块链的潜在攻击。当单个实体或协作团体控制了网络超过 50%的总算力时，他们就有能力在理论上比整个诚实网络更快地生成区块，从而可以强制进行链重组，实现双重花费等恶意行为。
- **BIP (Bitcoin Improvement Proposal):** 比特币改进提案。这是比特币社区用于提出、讨论和采纳新功能或协议变更的标准化流程。例如，SegWit（BIP 141）和 Taproot（BIP 340）都是通过 BIP 流程被接受的。
- **区块头 (Block Header):** 一个区块的元数据部分，大小为 80 字节，包含了版本号、前一区块的哈希、默克尔树根、时间戳、难度目标（nBits）和随机数（Nonce）。矿工进行工作量证明计算时，正是对区块头进行反复哈希。
- **区块补贴 (Block Subsidy):** 作为对矿工保护网络安全的奖励，每个新区块中允许被创造出来

的、全新的比特币数量。这个数量大约每四年（210,000 个区块）减半一次。

- **Coinbase 交易 (Coinbase Transaction):** 每个区块中的第一笔、也是唯一一笔特殊交易。它由矿工创建，用于领取该区块的区块补贴和所有交易的手续费。它的输入是全新的，标志着新货币的诞生。
- **共识 (Consensus):** 在一个去中心化系统中，所有参与节点就单一、权威的交易历史版本达成一致的过程。在比特币中，共识是通过工作量证明和最长链规则共同实现的。
- **双重花费 (Double-Spending):** 将同一笔数字货币（同一个 UTXO）花费两次或多次的行为。这是所有数字现金系统都必须解决的核心问题。
- **HD 钱包 (Hierarchical Deterministic Wallet):** 分层确定性钱包（BIP 32）。一种现代钱包标准，可以从一个单一的主种子（Master Seed）中，以树状结构派生出几乎无限数量的密钥和地址。用户只需备份一次主种子（通常以助记词形式），即可恢复所有资产。
- **硬分叉 (Hard Fork):** 一种不向后兼容的协议规则变更。硬分叉后，遵循新规则的节点将无法再与遵循旧规则的节点兼容，从而导致区块链分裂成两条独立的链。BCH 和 BSV 都是通过硬分叉从原链中分离出来的。
- **默克尔树 (Merkle Tree):** 一种哈希树。它通过对区块中的所有交易 ID 两两哈希，并不断向上迭代，最终生成一个单一的哈希值——默克尔树根。这个树根被记录在区块头中，可以高效地验证某笔交易是否存在于该区块中，是 SPV 的基础。
- **内存池 (Mempool):** 节点内存中用于临时存放已通过验证、但尚未被打包进区块的交易的区域。矿工会从内存池中根据手续费率等策略来挑选交易，构建新的区块。
- **矿工 (Miner):** 网络中专门负责创建新区块的节点。他们通过执行工作量证明（进行大量哈希计算）来竞争记账权，并以此获得区块补贴和交易手续费作为奖励。
- **P2P (Peer-to-Peer):** 点对点。一种网络架构，其中所有参与者（节点）都是平等的，可以直接相互通信，而无需经过一个中心化的服务器。
- **脚本 (Script):** 比特币内置的一种基于栈的、非图灵完备的编程语言。每一笔交易的解锁和锁定条件都由脚本来定义。例如，OP_CHECKSIG 就是一个验证数字签名的脚本操作码。
- **软分叉 (Soft Fork):** 一种向后兼容的协议规则变更。软分叉通过收紧规则来实现升级。升级后的节点可以理解并执行新规则，而未升级的旧节点虽然不理解新规则，但仍然认为新格式的交易或区块是有效的，因此不会导致网络分裂。SegWit 和 Taproot 都是通过软分叉实现

的。

- **SPV (Simplified Payment Verification):** 简单支付验证。一种允许轻客户端（如手机钱包）在不下载完整区块链的情况下，验证某笔交易是否已被打包进区块的方法。它通过下载区块头链和验证默克尔证明来实现。
- **交易费 (Transaction Fee):** 用户为了激励矿工将其交易打包进区块而支付的费用。它等于一笔交易所有输入总金额与所有输出总金额之间的差额。
- **UTXO (Unspent Transaction Output):** 未花费的交易输出。比特币的基本记账单位，可以被理解为钱包里的一张张“数字钞票”。每一笔交易的输入，都必须是之前某笔交易产生的 UTXO；而每一笔交易的输出，则会创造出新的 UTXO。
- **工作量证明 (Proof-of-Work, PoW):** 一种共识机制，要求参与者（矿工）执行一些有难度的、但易于验证的计算工作。在比特币中，这个工作就是寻找一个能使区块头哈希值小于特定目标值的随机数（Nonce）。PoW 将数字世界的记账权与物理世界的能源消耗绑定，从而保证了账本的安全。