区块链 Notebook

一

1.1 区块链起源和定义

- 区块链技术通过构建区块链网络，任何达成一致的无信任双方直接交易，不需要第三方中介的参与。
- 区块链是分布式数据存储、 点对点传输、 共识机制、加密算法等计算机技术的新型应用模式。
- 区块链是什么?
    - 定义：区块链是一个分布式账本 ，一种通过去中心化， 去信任的方式集体维护一个可靠数据库的技术方案
    - 数据角度：区块链是一种几乎不可能被更改的分布式数据库， 分布式不仅体现在对数据的分布式存储， 也体现在对数据的分布式记录
    - 业务角度：区块链是多种技术的整合的结果，通过新的数据结构分布式共识机制、 哈希加密算法以及独特的运行机制， 使得去中心化的信任构想成为现实。

1.2 区块链体系结构

- 数据层
    - 区块链：每一个区块包含前一区块的哈希值， 从而形成链式数据结构
    - 不可变数据：只能添加、 不能篡改
    - 不可变数据 + 时间刻度 → 互联网加上了时间轴
- 网络层
    - 网络层封装了区块链系统的组网方式、消息传播机制和验证机制。
    - 组网方式通常采用点对点（P2P）方式
- 共识层
    - 共识层主要封装区块链系统使用的各类共识算法。
    - 常见共识算法有：工作量证明共识（PoW）、权益证明共识（PoS）、拜占庭共识机制（BFT）、授权股份证明共识机制（DPoS）。
    - 早期的比特币区块链采用高度依赖节点算力的工作量证明 (Proof of work, PoW) 机制来保证比特币网络分布式记账的一致性
- 激励层
    - 激励层是将经济因素集成到区块链技术体系中来，包括经济激励的发行机制和分配机制等，主要在公有链当中出现。
    - 比特币：
        - 系统奖励给那些创建新区块的矿工， 该奖励大约每四年减半。
        - 新创建区块没有系统的奖励时， 矿工的收益会由系统奖励变为收取交易手续费。
- 合约层
    - 封装区块链系统的各类脚本代码、 算法以及由此生成的更为复杂的智能合约。
- 应用层

四层体系架构

1.3 区块链特征

- 去中心，去信任
- 开放，共识
- 交易透明，双方匿名
- 不可篡改，可追溯

1.4 区块链分类

- 按去中心化的程度
    - 公有链
    - 联盟链
    - 私有链
- 按是否需要许可
    - 无许可区块链
    - 许可区块链

1.5 数字货币简史

区块链1.0

- 旨在解决交易速度、 挖矿公平性、 能源消耗、 共识方式以及交易匿名等问题， 参照物为比特币（BTC）
- 比特币运行机制
    1. 产生新交易。
    2. 通过P2P网络被广播到所有的参与节点。
    3. 各节点都会将新交易进行验证，并各自形成一个等待上链的区块
    4. 通过共识算法选出拥有记账权的节点。
    5. 获得记账权的矿工通过P2P网络广播它的新区块，全网其它节点核对该区块记账的正确性。
    6. 超过一定数量的节点验证新区块无误后，就可以将这个区块连接到上一个区块上组成区块链

区块链2.0

- 旨在解决数据隐私、 数据存储、 区块链治理、 高吞吐量、 域名解析、 合约形式化验证等问题，参照物为以太坊（ETH） 。
- 以太坊（英语：Ethereum）是一个开源的有智能合约功能的公共区块链平台。通过其专用加密货币以太币（Ether，又称“以太币”）提供去中心化的虚拟机（称为“以太虚拟机”Ethereum Virtual Machine）来处理点对点合约。截至2018年2月，以太币是市值第二高的加密货币，仅次于比特币。
    - 最大优势：加入了部署智能合约功能， 每个人可以根据需求发布自己的智能合约。
    - 智能合约：一种旨在以信息化方式传播、验证或执行合同的计算机协议，它允许在没有第三方的情况下进行可信交易，这些交易可追踪且不可逆转。
    - 以太坊网络中的每笔交易都需要支付一定的手续费。 无论是转账交易还是部署智能合约， 所支付的手续费越高， 该交易就越快地被打包进区块中， 这也是以太币最主要的价值。
- 数字钱包：因为拥有私钥就拥有对应地址的数字货币，因此人们把管理密钥的软件称为“钱包
    - 全节点钱包
    - 轻钱包
- 数字货币市场现状

1.6 区块链典型应用场景

- 多源身份认证
- 分布式声誉体系：分布式信用数据管理体系
- 数据协同与交换
- 分布式流程协作
- 区块链+
    - 金融科技
    - 智慧旅游
    - 品质溯源
    - 教育应用

summary

- 区块链面临的挑战
    - 技术成熟
    - 应用场景
    - 专业人才
    - 法律法规

question

- 共识算法：对系统而言，而用户只能接触合约层

二、区块链数据层（最底层）

2.1 区块结构

- 区块：每个区块由区块头和区块体组成， 区块体只负责记录前一段时间内的所打包交易信息， 区块头记录当前区块的元数据。
    - 类似于账本中的账页， 其物理存储形式可以是文件（如比特币） ， 也可以是数据库（如以太坊）
    - 创世区块：第一个区块
    - 区块高度 (Block height)是指一个区块的高度，是指在区块链中它和创世区块之间的块数
- 区块头
    - 主要封装了当前版本号(Version)、 前一个区块的地址(Pre-block) 、 Merkle根(Merkle-root)以及时间戳(Timestamp)、 当前区块的目标哈希值(Bits)、 当前区块PoW共识过程的解随机数(Nonce) 等信息，主要分为以下三类
        - 引用父区块哈希值的数据Pre-block， 将当前区块与前一区块相连， 形成一条起始于创世区块且首尾相连的区块“链条”
        - 当前区块链所有交易经过哈希运算后得到的Merkle根， 指向区块体所封装的交易
        - 由目标哈希值、 时间戳与随机数组成， 这些信息都与共识竞争相关， 是决定共识难度或者达成共识之后写入区块的信息
    - 区块的主标识符是区块头的哈希值， 是使用两次SHA256哈希算法之后的结果， 可以唯一标识一个区块。
    - 区块高度， 将区块链看成一个垂直的栈。也常用来标识一个区块， 但可能不唯一。
- 区块体
    - 包含了当前区块的交易数量和经过验证的、区块创建过程中生成的所有交易记录。

    - 交易是区块链网络中传输的最基本的数据结构，所有有效的交易最终都会被封装到某个区块中，存于区块链上
- 比特币交易示例（比萨交易）
    - 

    - 元数据
    - 交易的输入列表
        - 创世区块可能只有1个输入
    - 交易的输出列表，奖励一般放在输出的最前面，交易的所有输出之和要小于等于输入之和，差值为给矿工的奖励。包含以下两部分
        - 一部分是特定数量的比特币， 以“聪” 为单位(最小的比特币单位)
        - 另一部分是锁定脚本， 即提出支付输出所必须被满足的条件以“锁住” 这笔总额。

UTXO

- UTXO：Unspent Transaction Outputs，未使用的交易输出
    - 交易可以溯源
    - UTXO：一笔交易的输出没有任何另一笔交易的输入与之对应，则说明该输出中的比特币尚未被花费
    - 作用
        - 通过收集当前所有的UTXO， 可以快速验证某交易中的比特币是否已被花费。
        - 通过收集某人所有地址的UTXO， 可以统计他所拥有的比特币数量。
            - 比特币系统中其实并不存在“账户”，而只有“地址”（钱包）。
    - 转账的交易过程：用发起方私钥（从一个输出是发送方地址的交易中上一个UTXO）取出比特币，并用私钥对新交易进行签名。一旦交易完成，这些比特币就转到接收方的钱包地址中去。接收方钱包中新交易的未使用交易UTXO输出，只有接收方的私钥才可以打开。
    - 优点
        - 易于确定比特币的所有权，可以让双重花费更容易验证
        - 与区块链账本是完全融为一体的

Merkle树

- 比特币系统采用二叉默克尔树来组织每个区快中的所有交易。
- 可以使用默克尔路径快速校验某个区块中是否有特定的交易

通用交易类型

- 生产交易：又叫coinbase交易， 每个区块的第一笔交易都是生产新币的交易。 该交易没有输入地址， 仅有个输出地址， 其作用是将系统新
  生成的加密货币奖励给创造当前区块的矿工
- 通用地址交易：区块链系统中最常见的交易， 由N个输入和M个输出构成， 其中N,M>0。 根据N和M的不同取值， 可以进一步细分为一对一转账交易、 一对多分散交易、 多对一聚合交易和多对多转账交易。
- 合成地址交易：一类特殊的交易，其接收地址不是通常意义的地址， 而是一个以3开头的合成地址。
  - 一般是M of N模式的多重签名地址，其中1≤N≤3、 1M≤N，通常选择N=3。

2.2 区块链的运行流程

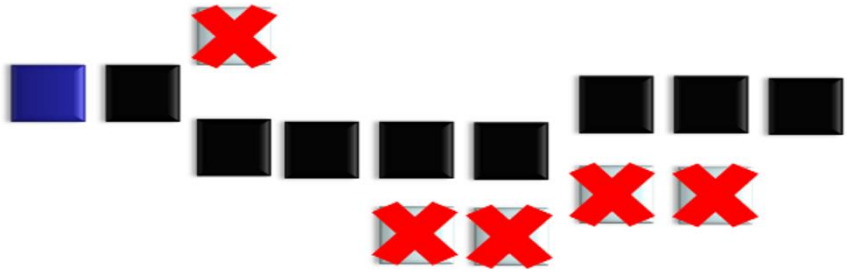
- 交易的8个步骤
  1. 源节点创建交易并验证目的节点的地址
  2. 源节点对交易进行签名加密
  3. 源节点将该交易广播至全网其他节点
  4. 全网节点接收交易并验证其有效性，直到该交易被全网大多数节点验证和接受
  5. 交易被暂存于节点内存池，并判断是否为孤立交易
  6. 交易被打包至节点本地区块中
  7. 全网共识结束后，获胜节点将其本地区块追加到主链
  8. 交易在主链上被越来越多的后续区块确认
- 交易的四个主要环节：交易生成、网络传播与验证、共识出块、激励分配
  - 交易生成：源节点创建交易，将目的节点的公钥作为交易的参数，使用自己的私钥对新交易签名。
  - 网络传播与验证
    - 比特币网络是P2P网络，使用Gossip协议进行交易的传播
    - 每个节点收到交易后都会独立对其有效性进行验证，验证通过后才会中继转发到其他节点。
    - 通过验证环节，有效抵御了恶意交易、垃圾信息的传播和拒绝服务攻击
- 交易池管理
  - 交易池：一个内存池，用于存放待确认打包的有效交易
  - 孤立交易池：暂时存放缺失父交易的子交易。
  - 交易池导致的交易拥堵和低手续费交易不能及时确认
- 交易优先级排序和手续费定价
  - 比特币系统的交易优先级公式

$$\text{交易优先级} = \frac{\sum_{\text{每个输入对应的UTXO}} (\text{UTXO交易额} \times \text{UTXO存在时间})}{\text{交易的字节长度}}$$

$$\text{交易字节长度} = 148 \times \text{输入数量} + 34 \times \text{输出数量} + 10$$

- 共识竞争和构建区块

    - 采用工作量证明(Proof of Work,PoW)共识算法， 其核心思想是通过引入分布式节点的算力竞争来保证数据一致性和共识的安全性
    - 各矿工节点基于各自的计算机算力相互竞争来共同解决一个求解复杂但验证容易的SHA256数学难题(即挖矿)， 最快解决该难题的节点将获得区块记账权和系统自动生成的比特币奖励
    - 该数学难题可表述为：根据当前难度值， 通过搜索求解一个合适的随机数(Nonce)使得区块头各元数据的双SHA256哈希值小于或等于目标哈希值。
    - 比特币系统通过灵活调整随机数搜索的难度值来控制区块的平均生成时间为10分钟左右。
- 难度和难度调整：
    - 难度：用来度量矿工成功挖到下一个区块的难易程度
    - 符合要求的区块头哈希值通常由多个前导零构成， 目标哈希值越小区块头哈希值的前导零越多， 成功找到合适的随机数并“挖” 出新区块的难度越大
- 分叉处理与主链判定
    - 分叉现象：如果多个矿工节点在同一时间段内成功搜索到符合哈希结果要求的随机数， 则这些矿工都将认为自己在共识竞争中获胜并向比特币网络中广播其构造的区块， 从而产生在同一区块高度出现多个不同的有效区块的情况， 即产生分叉现象
    - 为保证区块链系统中仅有唯一的主链， 必须定义合适的主链判定准则来从多个分叉链中选择符合条件的唯一主链。 此时不在主链上的区块将成为“孤块”， 发现孤块的矿工节点也不会得到相应的比特币奖励。
    - 区块链的形状并非单一的链条，而是一个“树状”结构。其中每个共识轮次对应的时间点上仅有唯一区块是有效的， 因而树状结构中也仅有唯一的主链条（有效的单向路径）



区块链分叉示意图

    - “最大工作量原则”作为消除区块链分叉时的主链判定规则， 该原则可体现为多种形式
        1. 如果不同分支的区块高度不同， 则选择最长区块高度的分支为主链
        2. 如果高度一致，则选择难度系数最大的分支作为主链
        3. 如果高度和难度系数均相同， 则选择接受时间最早的分支为主链
        4. 如果上述所有评判系数均相同， 则等待新区块产生并连接到某个或者多个分支、 区块高度增加后，重复步骤1-3直至选出主链。 此时， 生成新区块的节点即可对当前多个分支子链进行“投票”， 并链接至最有可能成为主链的分支子链上

2.3 数据层的关键技术

时间戳

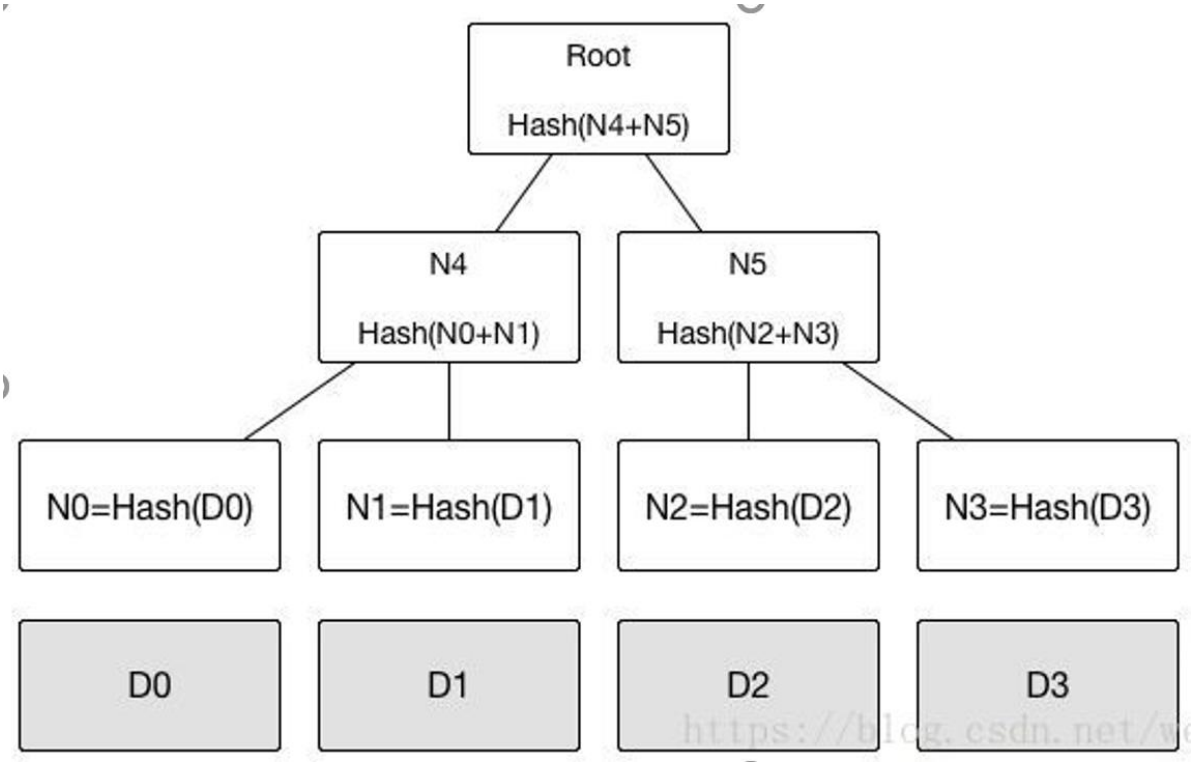
- 通俗的讲， 时间戳是一份能够表示一份数据在一个特定时间点已经存在的完整的可验证的数据。
- 功能：记录某件事情的发生日期和时间，以及证明事实存在并保证先后关系
- 基于文档时间戳
- 比特币系统的时间戳设计
    - 时间戳服务器：通过把以数据区块形式存在的一组比特币交易实施哈希运算并加盖时间戳，并在比特币网络中广播该哈希值。 这个时间戳证明在该时间这个数据一定是存在的， 因为只有数据只有在该时间才能得到相应的哈希值。

- 每个时间戳的哈希值包含了前一个时间戳， 后续的时间戳都是对之前时间戳的增强， 形成了一个**环环相扣的时间戳链条**
- 不要求强同步，保证先后顺序即可，但也不能乱改（代价很大）
- 比特币系统设计了两个防止节点恶意修改本地时间的规则
  - 时间校正
  - 拒绝接收时间戳不在此时间范围内的区块 （于前11个区块的中位数并且小于比特币节点的网络调整时间+2小时 ）

哈希函数

- 哈希函数可以在有限且合理的时间内， 将任意长度的二进制字符串映射为固定长度的二进制字符串
- 哈希函数的输出：哈希值，也称为数字摘要（Digital Digest）
- 哈希碰撞
- 抗原像性：单向性。对任意预定义的输出数据， 无法反推其输入数据。因此， 哈希函数可以看作是一类只有加密过程而没有解密过程的“单向” 加密函数。
- 抗第二原像性/若抗碰撞性：给定输入数据x1时，寻找其他不等于x1的数据x2， 使得H(x1)=H(x2)在计算上是不可行的。
- 强抗碰撞性(Collision Resistance)：寻找任意两个不同的输入x1和x2， 使得H(x1)=H(x2)在计算上是不可行的。
- 谜题友好型：对于任意n位输出y来说， 假设k是从具有较高不可预测性的高阶最小熵分布中选取的， 则无法找到有效方法可在比2的n次方小很多的时间内找到x，使得H(k|x)=y成立
  - 用来**挖矿**
- 雪崩效应：输入数据发生任何细微变化， 哪怕仅有一个二进制位不同， 也会导致输出结果发生明显改变。
- 定长/定时性：不同长度输入数据的哈希过程消耗大约相同的时间且产生固定长度的输出。
- 完整性校验
- 数据要素管理
- 共识竞争：大多数区块链系统， 特别是基于PoW共识的公有链系统， 都是利用大量的哈希函数运算来确定共识过程中获胜的矿工。 这主要是利用哈希函数的迷题友好性， 使得矿工除了付出大量算力资源执行哈希运算之外， 没有其他捷径可以对PoW共识过程进行求解。
- MD系列算法：一类较为成熟的哈希算法
  - MD：Message Digest，将输入数据映射为128位哈希值
  - MD5不具备“强抗碰撞性”
- SHA算法：安全哈希算法
  - 按照输出哈希值的长度命名
- RIPEMD算法：skip

默克尔树



- 哈希树：树结构哈希链
- 作用：快速归纳和校验区块数据的存在性和完整性
- 优点
  - 区块头仅需包含根哈希值
  - 支持“简化支付验证”， 即在不运行完整区块链网络节点的情况下也能对交易数据进行校验。（轻节点）
- 在比特币系统的应用
  - Hash函数caiyongSHA256算法
  - 二叉Hash树（交易数量为奇数时，最后一个交易重复）
  - 默克尔路径，可快速验证某个区块是否存在指定交易
- 简化支付验证（SPV）
  - 具体步骤

非对称加密

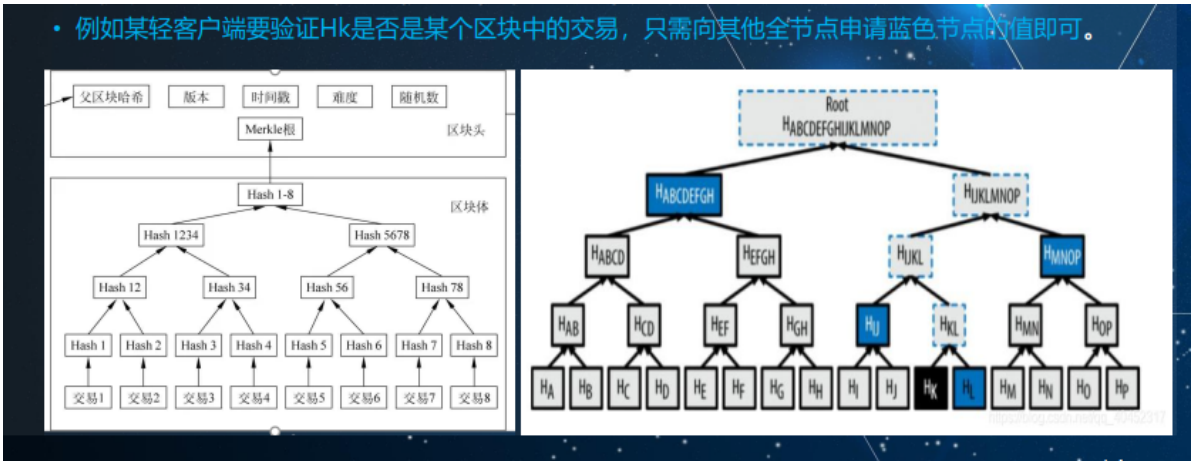
- 对称加密也称为单密钥加密
- 非对称：一堆公钥和私钥
- 非对称加密中含有一个密钥对， 公钥和私钥。私钥由一方安全保管，不轻易外泄，而公钥可以发送给任何人
  - 常用：RSA
  - 比特币：椭圆加密算法
- 在比特币系统中的应用
  - 私钥证明了用户对于账户的所有权， 如果用户想要使用某个账户中的比特币， 只有拥有该账户对应的私钥， 才能如愿使用。 在登录认证的场景中， 用户输入私钥信息， 客户端使用私钥加密登录信息后发送给服务器， 服务器接收后采用对应的公钥解密认证登录信息
  - 在比特币交易中用户使用私钥对交易进行签名， 交易信息广播后， 验证节点通过公钥对信息进行解密， 从而确保信息是由A发送的
  - 区块链网络充分利用了非对称加密的特性， 一是使用其中一个密钥加密信息后， 只有对应另外一个密钥才能解开；二是公钥可以向其他人公开， 公钥不能逆推出私钥， 保证了账户安全性。

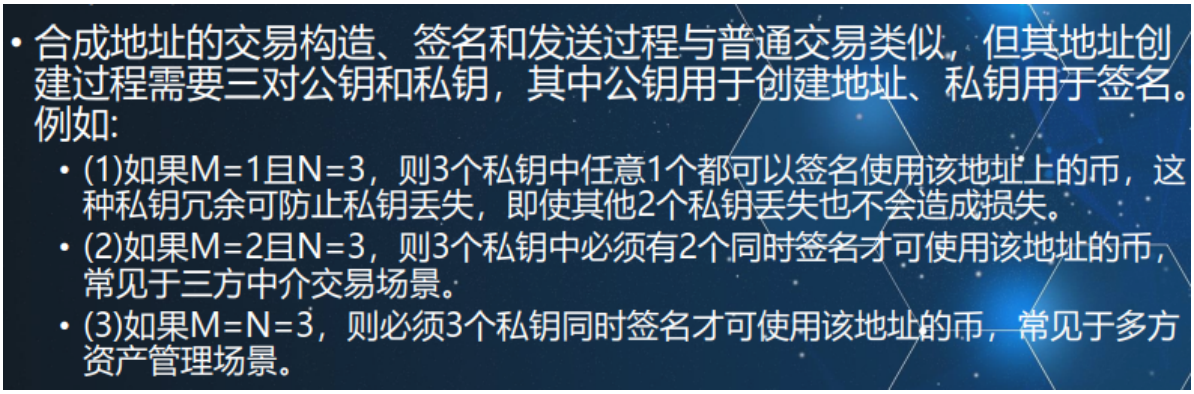
数字签名

- 数字签名(Digital Signature)是一种证明数字消息、 文档或者资产的真实性的数学方案， 其作用：
  - 身份认证， 使接收者有理由相信其接收到的内容是由已知的发送者发出的
  - 不可抵赖， 发送者无法否认其曾经发送过
  - 完整性， 该内容在传输过程中未被算改(完整性)
- 模型
- 流程
  - 签名Sign阶段
  - 验证Verify阶段
- 多重签名
  - 常用于多个参与者对某个消息、 文件和资产同时拥有签名权或者支付权的场景。
  - 应用：银行的联名账户
  - 多重签名场景通常需要N个参与者之中至少有M个参与者联合签名， 其中N≥M≥1。当N=M=1时,多重签名退化为传统的单人签名。
  - 根据签名过程的不同分为两类：有序多重签名和广播多重签名。前者， 签名者之间的签名次序是一种串行的顺序， 而后者， 签名者之间的签名次序是一种并行的顺序。
  - 比特币系统一般采用"N选M" 的形式， 即该多重签名地址共有N个私钥， 至少需要其中M个私钥共同签名才能从这个地址中转账。

question

？转账的交易过程：用发起方私钥（从一个输出是发送方地址的交易中上一个UTXO）取出比特币，并用私钥对新交易进行签名。一旦交易完成，这些比特币就转到接收方的钱包地址中去。接收方钱包中新交易的未使用交易UTXO输出，只有接收方的私钥才可以打开。





难度控制十分钟和之前讲的交易时间有关系吗？解决数学问题的时间减短，生成新区块时间减少->交易速度增加？难度降低的话，系统危险上升（×）？

- 是在系统建立初期就决定了的，之后很难再进行修改；只有在掌握了51%以上的节点之后才有可能随意篡改，和数学问题难度无关
- 尽量避免分叉；（同步优先，增加块长度也可能影响同步时间，不可取）
    - 区块是并发产生的，并且具有广播延迟，高出块率会导致更多的分叉
    - 分叉浪费网络和处理资源，分叉降低安全性
- 最长链规则不再安全
    - 无效分叉多，主链长度变短，攻击难度变小
- 同步账本的时间应该远小于出块时间，否则就会增加分叉的概率

三、区块链网络层

3.1 P2P网络

- 什么是P2P网络?
    - 对等网络或对等计算机网络，也称“点对点” 或“端对端” 网络， 是一种在对等网络实体（Peer）之间分配资源、 任务和工作负载的分布式应用框架；
- P2P网络的特点
    - 可扩展性强
    - 健壮性好
    - 高性价比
    - 私密性
    - 均衡性
- P2P网络的分类
    - 混合式对等网络：不是完全去中心化的
    - 无结构对等网络
    - 结构化对等网络
- P2P与覆盖网络（Overlay Network）
    - P2P网络通常构建在更底层的物理网络之上， 并为特定应用提供支持，是典型的覆盖网络
    - 覆盖网络（OverLay Network）：建立在另一个网络上、 并为更高层应用提供支持的中间层网络
    - 覆盖网络的作用：
        - 使得上层应用无需过多考虑与网络有关的对等实体发现、 直接通信、 数据安全、 资源定位、 网络标识及其分配、 节点加入与退出、 负载均衡等问题， 将精力集中在业务功能实现上。
        - 使得上层应用可以控制应用层流量
    - 区块链中节点之间的连接即使用Overlay
    - 底层使用TCP
- P2P网络应用经典案例

3.2 比特币区块链网络

- 比特币网络是一个典型的P2P网络，包括系统的组网方式、消息传播机制和验证机制
- 比特币网络节点-由多种类型的节点组成
  - 功能集合
    - 网络路由器 N
    - 完整区块链 B
    - 矿工 M
    - 钱包 W
  - 每个区块链都参与全网路由，同时也可能包含其他功能
- 全节点和轻节点
  - 全节点：拥有完整的、 最新区块链数据的节点称为“全节点”， 这样的节点能够独立自主地校验所有交易
  - SPV节点/轻量级节点：只保留区块头数据， 通过“简易支付验证” 方式完成交易验证的节点称为“SPV节点” 或“轻量级节点”， 它们没有区块链的完整拷贝
- 比特币网络的组网方式
  - 新区块链节点加入区块链网络的五种方式
    - 地址数据库：节点第一次启动时无法使用这种方式
    - 通过命令行指定
    - DNS种子
    - 硬编码地址
    - 通过其他节点获得
- 节点发现过程
  - 用户比特币程序启动时，并不知道任何活跃的全功能节点的IP地址
  - 为了接入网络和发现这些地址，程序会向DNS地址(种子)发出查询请求
  - DNS服务器返回的响应包（DNS的A记录）中会包含一个或多个全功能节点的IP地址

3.3 数据传播协议

- 节点间采用TCP协议，端口：8333
- 通用的区块链网络包括如下核心场景
  - 节点入网建立初始连接
  - 节点地址传播发现
  - 矿工、全节点同步区块数据
  - 客户端创建一笔交易
  - 矿工、全节点接受交易
  - 矿工、全节点挖出新区块，并广播到网络中
  - 矿工、全节点接收广播的区块
- 消息
  - 不同节点间信息传输的基本单位，协议体现为消息格式的约定和时序
  - 基本格式：消息头+消息体

  - 校验码不具有纠错功能

- 常用消息类型
  - 参考知乎
- 过程
  1. 建立初始连接
  2. 地址广播及发现
  3. 同步区块数据
     - 全节点：同步区块数据
     - SPV节点：同步区块头

     初始区块下载IBD
     - 块优先
       - Checkpoint功能，：Checkpoint就是指定一个区块高度的区块哈希必须等于某个哈希值
       - 优点：简单
       - 缺点：IBD节点的所有下载都依赖于单个同步节点和块发送的顺序性 ，会存在速度限制、重复下载、磁盘空间浪费、高内存使用
     - 头优先
       - 先尝试下载链中区块的描述头结构，然后以并行方式连接多个网络节点下载区块
       - 最大1024个区块的下载移动窗口
  4. 交易广播
     - 比特币的新块广播：当一个矿工节点发现新的区块后，它需要将此区块在全网尽可能大的范围内广播
       - 主动推送
       - 区块中继
  5. 检测节点存活
     - ping消息用于确认接收方是否仍处于连接状态。通过发送ping消息，可以检测节点是否存活。
     - 接收方通过回复pong消息， 告诉发送节点自己仍然存在。
     - 默认情况， 任何超过20分钟未响应ping消息的节点会被认为该节点已经从网络中断开。
  6. Gossip传播协议

3.4 数据验证机制

- 数据验证清单
  - 区块大小在有效范畴内
  - 区块数据结构（语法）的有效性
  - 区块至少含有一条交易
  - 第一交易是coinbase交易，有且仅有一个
  - 区块头部有效性
  - 区块内交易有效性

3.5 矿池网络协议

- Getwork协议：最早的挖矿协议，现已禁用
- Getblocktemplate协议
  - 让矿工自行构造区块
  - 搜索空间巨大
- Stratum协议

    - 最常用

3.6 区块链分叉

自然分叉

- 机器共识过程产生的临时分叉
- 固有的节点地域分布、 网络传输延迟， 造成节点接收新区块存在一定时间差异。当两个不同节点近乎同时发掘出新区块A、 B并进行广播的时候， 就会造成后继区块链分叉的发生

人为分叉

- 人的共识失败产生的分叉（BIP，Bitcoin Improvement Proposal（比特币改进建议））
    - 软分叉：向前兼容的分叉。 新规则下产生的区块可被未升级的旧节点所接受， 旧节点只是无法识别、 解析新规则。 新、 旧版本互相兼容， 软分叉对整个系统的影响较小
    - 硬分叉：不向前兼容的， 旧版本节点不会接受新版本节点创建的合法区块，于是新旧版本节点开始在不同的区块链上运行， 由于新旧节点可能长期并存，不像软分叉是临时的， 硬分叉是有可能会长期存在的， 分叉链的存活在于其算力的大小。一般会经历如下几个阶段
        - 软件分叉：新客户端发布
        - 网络分叉
        - 算力分叉
        - 链分叉
- 社区分叉
- 对于数字货币持有者来说，硬分叉会让他们额外增加一笔财富（分叉链 Token）

question

四、区块链共识层

4.1 分布式系统模型与共识

分布式系统模型

- 分布式系统及其特点
    - 并发性
    - 缺乏全局时钟，没有一个全局正确的时间来协调各组件的行为
    - 组件故障的独立性
- 分布式系统的系统模型
    - 结构模型
        - 客户/服务器结构
        - 对等结构（客户/服务器模型的变种）
    - 基础模型
        - 对体系结构模型中公共属性的一种更为形式化的描述。包括：交互模型、 故障模型和安全模型
        - 交互模型
            - 进程之间通过消息传递进行交互，实现系统的通信和协作功能：有较长的时间延迟；时间是进程间进行协调的基本的参照， 在分布式系统中， 很难有相同的时间概念。

        - 独立进程之间相互配合的准确性受限于上面两个因素
        - 根据 时间 进行分类
            - 同步系统：每一条消息会在已知的时间范围内确定被接收到；实际很少见
            - 异步系统：实际较常见
            - 部分同步系统
    - 故障模型：定义可能出现的故障形式， 为分析故障带来的影响提供依据
        - 崩溃故障
            - 崩溃
            - 故障-停止
            - 故障-恢复
        - 遗漏故障
            - 发送遗漏故障
            - 信道遗漏故障
            - 接收遗漏故障
        - 时序故障
            - 时钟故障
            - 节点性能故障
            - 信道性能故障
        - 拜占庭故障：随机故障
            - 节点可能任意地、错误地，甚至恶意地执行某些未经许可的动作
    - 安全模型
        - 目的：提供依据， 以此分析系统可能收到的侵害， 并在设计系统时防止这些侵害的发生
- 分布式一致性（Consistency）
    - 定义
    - 一致性分类
        - 强一致性
        - 弱一致性
        - 最终一致性：弱一致性的特例
- 共识（Consensus）
    - 共识描述了分布式系统中多个节点之间，彼此对某个状态达成一致结果的过程。
      在实践中，要保障系统满足不同程度的一致性，核心过程往往需要通过共识算法来达成
    - 共识和一致性常被认为是等价的
- 一致性和共识的细微区别
    - 共识侧重的是分布式节点达成一致性的过程和算法， 是一种手段。
    - 一致性侧重于节点共识过程最终达成的稳定状态， 描述的是结果状态。
    - 达成某种共识并不意味着就保障了一致性（指强一致性）。只能说共识机制，能够实现某种程度上的一致性。

FLP和CAP定理

- **FLP定理**
    - 在含有多个确定性进程的异步系统中， 只要有一个进程可能发生故障， 那么就不存在协议能保证有限时间内使所有进程达成一致
- **CAP定理**
    - 网络服务不可能同时保证如下三个特点，最多只能保持两个：
        - 一致性：（**C**onsistency）：指强一致性，分布式系统中的所有数据备份在同一时刻必须保持同样的值。

        - 可用性：（**A**vailability）：集群的部分节点出现故障，系统仍可以处理用户请求，即所有读写请求可在一定时间内得到响应，不会一直等待。
        - 分区容错性：（**P**artition-tolerance）：出现网络分区、不同分区的节点间无法互相通信时被分隔的节点仍能正常对外服务。即允许丢失任意多的从一个节点发往另一个节点的消息
    - 大多数网络系统需要满足分区容错性，故一致性和可用性只能兼顾（无法同时满足）
        - 工程实践中，一般会适当放宽对特定性质的假设，例如放宽强一致性为弱一致性、最终一致性。如互联网数据库，用以作大数据处理和分析
        - 在分布式数据库系统中，一般要保证强一致性
    - 区块链系统的设计也必须遵从CAP定理
        - 以比特币为代表的大多数公有链通常牺牲强一致性，同时满足最终一致性（可能出现分叉，但最后要确定主链）、可用性和分区容错性。
        - 某些联盟链或私有链可能会牺牲可用性来满足强一致性和分区容错性。

拜占庭将军问题

- 节点不可靠、信道可靠的同步系统

> **两军问题**
>
> - 节点可靠，信道不可靠
> - 经典情形下两军问题是不可解的，并不存在一个能使蓝军一定胜利的通信协议。
> - TCP协议三次握手，是两军问题的工程解。

- 问题模型
    - 假设
        - 一个网络中存在n个节点，其中第i个节点发出的消息记为vi
        - 每个节点都会监听其他节点发送的消息，即$v1, v2, \dots, vn$
        - 网络中存在m个恶意节点
    - 求解问题：在存在恶意节点的网络中，诚实节点能对决策问题达成一致
    - 求解条件
        - 一致性：
            - 全局角度：每个诚实节点必须接收到相同的消息集合$v1, v2, \dots, vn$
            - 单个节点角度：无论节点i是否诚实，任意两个诚实节点所保存的其消息均为vi
        - 正确性：（单个节点角度）若节点i是诚实的，其他诚实节点必须以它发送的消息作为vi
    - 针对单个节点设计求解算法
- 口头消息算法

    - 法流程：OM(m)， m>0

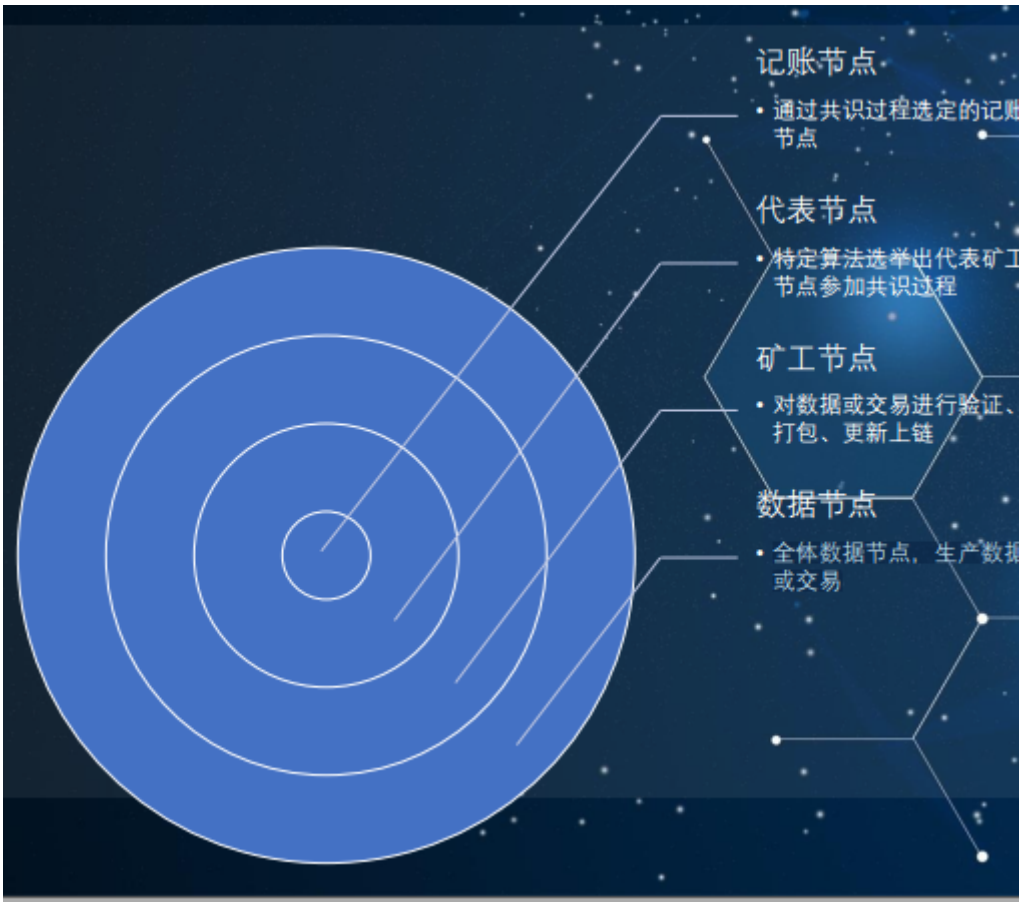
1. 主节点向每个从节点发送消息v
2. 对任意从节点i， 其接收的消息v记为vi，其作为主节点运行OM(m-1)向剩余n - 2个从节点发送vi
3. 对任意i和j≠ i， 令vj为OM（m-1)中从节点i从节点j接收的消息， 从节点采用消息majority（v1， v2， ……， vn-1)，类似投票

- 问题：当三个节点中存在一个非诚实节点时， 诚实节点的行为会不确定
- 带数字签名的拜占庭将军问题：可以通过消息剔除不诚实节点，容错能力较高

- 区块链解决拜占庭问题
  - 规则
  - 优势
    - 信息里每个将军都要签名验证身份， 如果有将军篡改了消息， 大家就能看到是哪些将军篡改了消息。
    - 尽管有不一致的消息。 但只要一个消息得到了6 名或 6 名以上将军的同意， 那么大家就达成了共识
  - 劣势：十分钟才能发起一个消息
- 拜占庭模型→公有链使用

共识过程的主流模型

- 记账节点：
- 代表节点：特定算法选举出代表矿工节点参加共识过程 （矿工太多了，全部达成公式不太现实）
- 矿工节点：对数据或交易进行验证、打包、更新上链
- 数据节点：全体数据节点，生产数据或交易



- 公识过程
  - 选主共识
    - 选主：选出记账节点

        - 造块：记账节点打包并广播
    - 主链共识
        - 验证：（涉及拜占庭问题）
        - 上链：记账节点将新区块添加到主链 。如果有分叉，需根据共识算法中的主链判别标准确定主链。

共识算法的分类

- **算法共识**
    - 研究在特定的网络模型和故障模型的前提下，如何在缺乏中央控制和协调的分布式网络中确保一致性，其实质是一种“机器共识”。
    - 后续提到的区块链共识算法均指的是“算法共识”。
- > **决策共识**
    >
    > - 研究无中心的群体决策中，如何就最优的决策达成一致的问题，例如关于比特币系统扩容问题和分叉问题的社区讨论与路线选择，其实质是“人的共识”
- 共识算法的分类
    - 选主策略：选举类、证明类、随机类、联盟类、混合类
    - 容错类型：拜占庭容错（Byzantine Fault Tolerance ，BFT）、非拜占庭容错
    - 部署方式：公有链、私有链、联盟链
- 选举类共识：联盟类和公有链常用。矿工节点在每一轮共识过程中通过投票选举的方式选出当前轮次的记账节点,首先获得半数以上选票的矿工节点将会获得记账权 。 例如 Paxos 和Raft 等
- 证明类共识：也可称为 Proof of X 类共识,即矿工节点在每一轮共识过程中必须证明自己具有某种特定的能力, 证明方式通常是竞争性地完成某项难以解决但易于验证的任务, 在竞争中胜出的矿工节点将获得记账权。 例如 PoW 和 PoS 。
- 随机类共识：矿工节点根据某种随机方式直接确定每一轮的记账节点。 例如Algorand 和 PoET
- 联盟类共识：即矿工节点基于某种特定方式首先选举出一组代表节点， 而后由代表节点以轮流或者选举的方式依次取得记账权。 这是一种以代议制为特点的共识算法，例如 DPoS 等 。

共识简史

4.2 分布式一致性算法

Paxos算法

- Paxos算法解决的问题是在一个可能发生消息延迟、丢失、重复的分布式系统中如何就某个值达成一致，保证不论发生以上任何异常，都不会破坏决议的一致性。（非拜占庭故障）
- Paxos算法将分布式系统的节点分为三种角色：
    - 提议者（Proposer） 负责向acceptor发起提案
    - 接受者（Acceptor） 负责响应提案， 对提案进行回应以表示自己接受提案
    - 学习者（Learner） 不参与前面的决策过程， 只从别人那里学习已经确定的、 达成一致的提案结果
    - 一个节点可以同时拥有这三种身份， 也可以只有部分身份。
- 如果一个提案被半数以上Acceptor接受， 它就被选定了（ Chosen） ， 并由Learner负责执行选定的提案。
- 提案
- 约束条件
    - 对提案值Value：如果没有Value被提出，就不应该有Value被选定。
        - 只有被提出的值Value才可以被选定

        - 只有一个值Value可以被选定
        - 除非一个值Value被选定，否则它不会被执行
    - 对于提案： 如果只有一个提案被提出的话,那么这个提案应该被最终选定， Acceptor必须能够接受多个不同的Value
- 算法流程
    - Phase1
        1. 准备 （Prepare）
        2. 承诺（Promise）：
    - Phase2
        1. 请求接受（Accept Request）
        2. 接受（Accepted）
    - 当获得半数以上Accepted返回后， 该提案被选定， 并提交Learner执行。Learner可以通过三种方式获取被选定的提案值value
        - Acceptor每接受一个提案， 就将该提案发送给所有Learner
        - Acceptor每接受一个提案， 就将该提案发送给主Learner；当提案值被最终选定后， 再由主Learner发送给其他Learner
        - Acceptor每接受一个提案， 就将该提案发送给一个Learner集合， 该集合中的每个Learner都可以将选定的提案值发送给所有的Learner。
- 过程：一轮Paxos只对一个值达成共识
    - Accept本地记录
        - minProposal：自身响应的提案id最大prepare请求的提案id
        - acceptedProposal：自身响应的accept请求中提案编号最大的提案id
        - acceptValue 自身响应的accept请求中提案编号最大的提案值
    -

Raft算法

-

4.3 主流区块链共识算法

4.4 共识算法新进展

question

2. If term == currentTerm, votedFor is null or candidateId, and candidate's log is at least as complete as local log, grant vote and reset election timeout

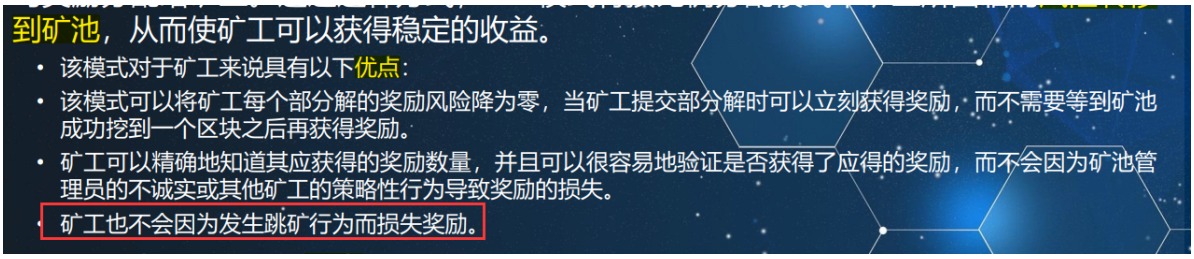
---

Implementation:
1. Return failure if term < currentTerm
2. If term > currentTerm, currentTerm ← term 这不是我选的leader
3. If candidate or leader, step down
4. Reset election timeout
5. Return failure if log doesn't contain an entry at prevLogIndex whose term matches prevLogTerm
6. If existing entries conflict with new entries, delete all existing entries starting with first conflicting entry
7. Append any new entries not already in the log
8. Advance state machine with newly committed entries

P32 3f+1 <=n是什么意思

五、区块链激励层

question



演示

- 目前比较火的交易所：币安、coinbase
- 各种货币背后的经济模型