An Introduction to Blockchain & Consensus
区块链与共识机制

By Christopher Goes

State machines & distributed ledgers
状态机和分布式账本

- State machine状态机
  - State **S**状态 S
  - Transaction **T**交易T
  - Update function **U = S → T → S**更新机制U = S→T→S
  - Can model any state which can change可以模拟任何可以改变的状态
    - Currency, financial products, identity, record-keeping, virtual worlds
    - 货币，金融产品，身份，记录保存，虚拟世界
- Why distribute a ledger?为什么要将账本进行分布式？
  - Centralized ledger: data (**S**) and ruleset (**U**) controlled by single entity
  - 集中式账本：由单个实体控制的数据（S）和规则集（U）
    - Bank, trusted party, for-profit company银行，可信方，营利性公司
    - These ledgers are common goods!这些都是常见的账本！
  - What if instead, everyone could keep the ledger?如果相反，每个人都可以保留 账本？
    - Anyone could read and write **S**任何人都可以 读写状态S
    - Permissionless innovation, transparency, auditability
    - 无需任何许可的创新，透明度，可审计性

- Explain what a state machine is:解释状态机是什么：
  - State type S状态类型S.
  - Transaction type T交易类型T.
  - Update function S -> T -> S更新功能S - > T - > S.
- Explain why state machines are a useful model:解释为什么状态机是一个有用的模型：
  - Can encapsulate any state of interest which we want the ability to update可以封装我们想要更新的任何感兴趣的状态
- What do we use ledgers for?我们用账本做什么？
  - Currency货币
  - Finance金融
  - Identity身份
  - Personal record-keeping个人记录保存
  - Virtual world state (gaming)虚拟世界状态（游戏）
- Why would we want to distribute a ledger?我们为什么要分布式记账？
  - Presently, most ledgers are centralized: data (the state) & ruleset (the transition function) are in the control of a single entity目前，大多数分类账是集中的：数据（状态）和规则集（转换函数）在单个实体的控制之下
  - What if everyone could keep the ledger?如果每个人都可以保留 账本怎么办？
    - Anyone could read and write state任何人都可以读写状态
    - Permissionless innovation无需许可的创新
    - Transparency, auditability
    - 透明度，可审计性

The double-spend problem双重花费问题

- Let's distribute the ledger!让我们将账本分布式
  - Known participant set已知参与者集合
  - Everyone sends each other every transaction每次交易每个人都会互相进行
  - Example: a currency
  - 例如:货币
- How do we order transactions?我们如何给交易排序？
- What if a participant sends conflicting transactions?如果参与者发送冲突的交易怎么办？
  - Tries to pay two people twice with the same money
  - 尝试用同样一笔钱对两个人都进行支付
- This is the "double-spend problem"这是"双重花费问题"
  - We need function **F** to choose between two histories我们需要函数F来从两个历史中进行选择

- Let's implement a naive distributed ledger让我们实现一个天然的分布式 账本
  - Known participant set已知参与者集合
  - Start with the same state从相同的状 态开始
  - Everyone has all the state每个人都 拥有所有的状 态
  - Everyone sends each other every transaction每次交易每个人都会互相发送
- Problems!问题！
  - How do we order transactions? What if only a certain order "works"? What if we receive them in a different order?我们如何为交易排序？ 如果只有某个 订单"有效"怎么 办？ 如果我 们以不同的顺序收到它 们怎么 办？
  - What if a participant sends different conflicting transactions to different peers?如果参与者向不同的 对端发送不同的冲突交易怎么 办？
- Explanation of how these are all the same problem - the "double-spend problem" (so named for currencies)解释这些都是同 样的问题 - "双重花 费问题"（以货币命名）
- List of what we need我们需要的清 单
  - Function to pick between two "histories" - orderings of transactions在两个"历史"之间选择的函数 - 交易的顺序
  - If all validators run this function, we can be confident that we agree!如果所有验证器都运行此功能，我 们可以确信我 们同意！
  - Now we need to find such a function!
  - 现在我们需要找到这样的功能！

Byzantine Generals拜占庭将军

- A group of generals command the Byzantine army一组将军指挥着拜占庭军队
  - Each must choose whether to attack or retreat每个将军都必须选择是攻击还是撤退
    - Could be any choice of mutually exclusive action可以是任何互斥行动的选择
  - Can only send messages to other generals只能向其他将军发送消息
    - Messages may be arbitrarily delayed or forged消息可以任意延迟或伪造
- If they do not agree, catastrophe!如果他们没有达成一致，则是浩劫！
  - Attack will be a failure, retreat will be a rout攻击将是失败，撤退将是溃败

- Explain Byzantine generals problem (more formal definition)解释拜占庭将军的问题（更正式的定义）
  - Generals, Byzantine army将军，拜占庭军队
- Define “Byzantine-fault tolerance”

- 定义“拜占庭容错”

Byzantine-fault-tolerance拜占庭容错

- Generals might be malicious, but we can authenticate messages将军可能是恶意的，但我们可以验证消息
  - **h** honest generals, **d** dishonest generals
  - h个诚实的将军，d个不诚实的将军
  - Honest generals send only the same message诚实的将军只发送相同的消息
  - Dishonest generals may send different messages to different generals不诚实的将军可能会向不同的将军发送不同的信息
  - Generals wait for messages before deciding what to do将军做决定之前等待消息
  - Total nodes **n**, **n = h + d**
  - 总节点数n，n = h + d
  - Worst case: **h/2** generals want to attack, **h/2** want to retreat最坏的情况：h / 2将军想要攻击，h / 2想要撤退
    - Traitorous generals tell the first half to attack, second half to retreat叛国将领告诉前半部分将军进攻，另一半将军撤退
    - **h/2 + d** nodes agree on attack, **h/2 + d** nodes agree on retreat
    - h / 2 + d节点同意攻击，h / 2 + d节点同意撤退
    - Must require threshold **t > h/2 + d**, **t <= h** (liveness)
    - 必须要求阈值t> h / 2 + d，t <= h（存活数）
      - Simplifies to **h > h/2 + d ⇒ h > 2d** (over ⅔ of nodes must be honest)
      - 简化为h> h / 2 +d⇒h> 2d（超过⅔的节点必须诚实）

- Explain ⅔ threshold解释⅔阈值
- Define “Byzantine-fault tolerance”定义“拜占庭容错”

Blockchain, first try: proof-of-authority
区块链，首次尝试：权威证明

- Voting-based BFT consensus
- 基于投票的BFT共识
- Fixed-size, known-identity validator (general) set
- 固定大小的已知身份验证器（通用）集
- Fork choice rule **F分叉选择规则F.**
  - Whether or not a history has been signed by ⅔ of validators
  - 由⅔的验证人验证历史记录是否被签名
- Problems问题
  - ⅔ honest assumption ⅔诚实的假设
    - Old keys could sign an old history旧秘钥可以签署旧的历史
    - Multiple valid histories, breaks consensus多个有效的历史，打破共识
  - Targetable目标性
    - Validators can be DoSed (generals assassinated)验证者可以是DoSed（将军被暗杀）
  - Static静态的
    - Validators may unfairly extract rent from network users验证者可能会不公平地从网络用户那里提取租金

- Fixed, known validator set
- Majority honest assumption
- Rent-seeking, targetable

Dynamic membership & public networks
动态成员和公开网络

- Design goals设计目标
  - Dynamic validator set动态验证者集合
    - Anyone can join if they want只要愿意，任何人都可以加入
    - Bad actors can be removed坏成员可以被移除
  - Anonymous validator set匿名验证者集合
    - Identities aren't required to validate验证过程不需要身份证明
    - Different kinds of identity: cryptographic keys, public reputation, IP address不同种类的身份证明：加密密钥，公共信誉，IP地址
      - Proof-of-authority requires all three
      - 权威证明时以上三个都需要

- Things we might want
  - Dynamic validator set
    - Anyone can join
    - Bad actors can be kicked by the protocol (somehow)
  - Anonymous validator set
    - Validators don't need to reveal their identities to do their job
    - Validators can't be DDOSed

Sybil attacks
西比尔袭击

- What if validators don't have an identity?如果验证者没有身份怎么办？
  - Anyone can masquerade as another validator任何人都可以伪装成另一个验证人
  - Anyone can pretend to be multiple different validators任何人都可以伪装成多个不同的验证人
  - One dishonest node could break consensus一个不诚实的节点可能会打破共识
- Central party could verify - but that's what we want to avoid 中心化方案可以进行身份核实 - 但这是我们想要避免的

- Identity in digital systems is hard!
- We can't use a CAPTCHA

Rational-actor assumptions
理性行为者的假设

- How to model the actors?如何为参与者建模？
  - Honest - behaves in best interest of protocol诚实 - 符合协议的最佳利益
  - Rational - behaves in own best interest理性 - 表现出最大的利益
    - Weaker assumption!弱者假设！
- Often care about behaviour over time经常关心一段时间内的行为
- Enables making statements about protocol safety允许发表有关协议安全性的声明
  - If and only if ⅔ validators are honest, proof-of-authority is safe
  - 当且仅当⅔验证者诚实时，权威证明才是安全的

- How do we model the actors in a consensus network?
- One option: honest / honest majority - the protocol is safe iff. some number of actors behave the “right” way (in best interest of the protocol)
- Another option: rational / fraction rational - the protocol is safe iff. some fraction of actors behave the “rational” way (in their own best interest)
  - This is a weaker assumption - preferable!

Blockchain, second try: proof-of-work
区块链，第二次尝试：工作证明

- Need to tie a verifiable scarce resource to voting on a history需要将可核查的稀缺资源与历史投票关联起来
  - In proof-of-authority, this scarce resource is identity在权威证明中，这种稀缺资源就是身份
- One option: hash function **H一种选择：哈希函数H.**
  - Takes some time to compute需要一些时间来计算
  - Randomly generates output based on input根据输入随机生成输出
  - Hashcash, 1997 - originally used to prevent email spam
  - Hashcash，1997年 - 最初用于防止垃圾电子邮件

- We need a scarce resource
- Hashcash, basics of hash functions
- Original use: email spam prevention
- Idea: commit a scarce resource (work) to one history
- Fork choice rule: choose the history with the most work

Nakamoto consensus
中本共识

- Each validator must commit "work" to one history每个验证者必须将"工作"提交给一个历史记录
  - Input for **H** - commitment to single history输入H - 对单一历史记录的承诺
  - Output of **H** must be below some value H的输出必须低于某个值
    - Value can be changed to require more or less computation可以更改值以需要更多或更少的计算
- Deployed in practice!在实践中部署！
  - Bitcoin, running for almost 10 years比特币，运行了近10年
- Problems问题
  - Speed limitations (many "forks")速度限制（许多"分叉"）
  - Consumes too much energy消耗太多能量
  - Requires synchrony需要同步
    - Pick **n** confirmations选择n确认

- Explain "chain of blocks" (each block incorporates commitment to past chain recursively)
- Economic opportunity cost to mining on the "wrong" fork
- Successfully deployed in practice! (Bitcoin)
- Problems
  - Limits speed (otherwise can't pick a canonical fork)
  - Negative externality (more energy than Denmark!)
  - Weak light client verification

Blockchain, third try: proof-of-stake
区块链，第三次尝试：权益证明

- Scarce resource: in-protocol unit稀缺资源：协议单元
  - Validators control some amount of this resource验证者控制一定量的此资源
  - Sign histories, weighted by how much they can control根据他们可以控制的程度权重来签署历史记录
- Can be used with voting-based BFT or Nakamoto consensus
- 可以与基于投票的BFT或Nakamoto共识一起使用
  - Vote each round on a new block在每一个新区块上进行每一轮投票
    - Fork choice rule **F** - chain approved by ⅔ of stake叉子选择规则F - 链由⅔股权所有人批准
  - Subset of validators votes on a new block验证者的子集对新块进行投票
    - Fork choice rule **F** - chain with most votes 叉选择规则F –链接最多投票的

- Use a scarce in-protocol unit (token)
- Sign histories with this token
- Can work with BFT or Nakamoto consensus

The nothing-at-stake problem
无利害关系问题

- Signing is not a scarce resource at all!签署根本不是稀缺资源！
  - Stakeholders can sign different histories without penalty利益相关者可以签署不同的历史而不受处罚
- OK if always online (synchronous)如果总是在线(同步)则没问题
  - Simply reject any new histories简单地拒绝任何新的历史
- Unsafe if offline (asynchronous)脱机时不安全(异步)
  - No way to know which history is canonical无法知道哪个历史是规范的

- How do we distinguish between two different histories?
- Signing is free!
- Stakeholders can sign multiple histories without penalty
- If you're always online, you can track the longest chain (synchronous)
- If you connect to the network later on, no way to know which history is canonical (asynchronous safety)
  - Messages can be delayed, so this could be anyone

Bonding & Slashing粘合和削减

- Solution: identify & punish misbehaving validators解决方案：识别并惩罚行为不当的验证人
  - Signing multiple histories is fault-attributable签署多个历史记录是错误属性
    - We know which validators did it!我们知道哪些验证人做的
  - Validators must "bond" valuable resource验证者必须"绑定"宝贵的资源
    - Lock it up for a period of time将其锁定一段时间
    - Put it "at risk" if they commit a protocol fault如果他们提交协议错误，将其置于"风险"
  - Misbehaving validators are slashed行为不端的验证员被削减
    - Commit a protocol fault - valuable resource is lost提交协议错误 - 丢失了宝贵的资源
- Signing now a scarce resource under rational-actor assumption现在在理性行为者假设下签署稀缺资源
  - Validators signing two histories will be caught and punished签署两份历史的验证人将被抓住并受到惩罚

- Consensus participants must "bond" value: lock it up for a period of time
- If a fault is discovered, they get "slashed": value deleted
- Period of time value is locked up determines degree of subjectivity
- What constitutes a protocol fault can be defined in-protocol
- This provides a similar "economic guarantee" to PoW - signing many histories has a cost (if discovered), and the cost may be very high

Special Q&A Sessions

If you have questions about the presentations, the topics, or Cryptium Labs, you can do two things:
如果你对展示内容，题目或者Cryptium 实验室，你可以采取如下两个措施:

Option A) Ask the question in public (must be in English) at the end of every presentation topic by raising your hand, we will answer them immediately on stage.
选项A : 在每个培训之后，请立即举手公开体温，我们会立即回答这个问题(但是需要用英文)
选项B :在我们的微信群中提问。如果是英文的，我们会在演讲过程中直接回答。如果是中文的，我们会收集这些问题并汇总成中文FAQ并在微信群中分享。

Option B) Write your question in our WeChat group! If it's in English, we will answer them during the event in public, if it's in Chinese, we will collect them and compile FAQ in Chinese and share with the WeChat group.

Cryptium Labs

Valid until 9/16 and will update upon joining group


CRYPTIUM LABS