PoW共识协议与安全性分析

范磊

上海交通大学

Bitcoin Backbone 模型

- Garay, Kiayias, Leonardos in [GKL14]
  https://eprint.iacr.org/2014/765

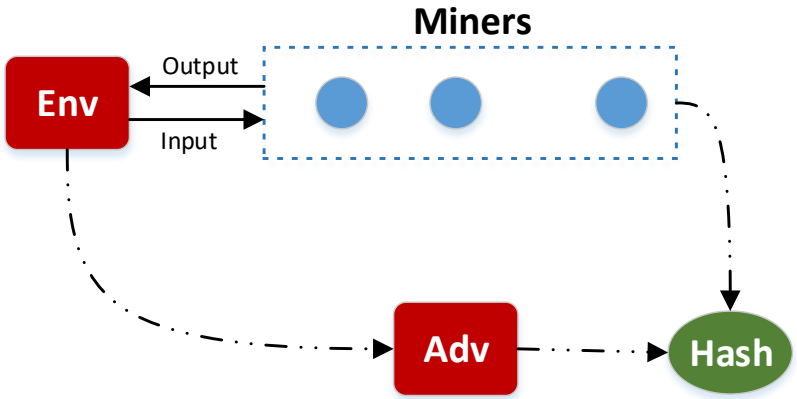
  - 第一个Bitcoin协议的正式抽象模型Bitcoin Backbone
  - 正式定义了账本协议以及安全特性
  - 第一次正式分析了比特币实现的共识特性

系统模型

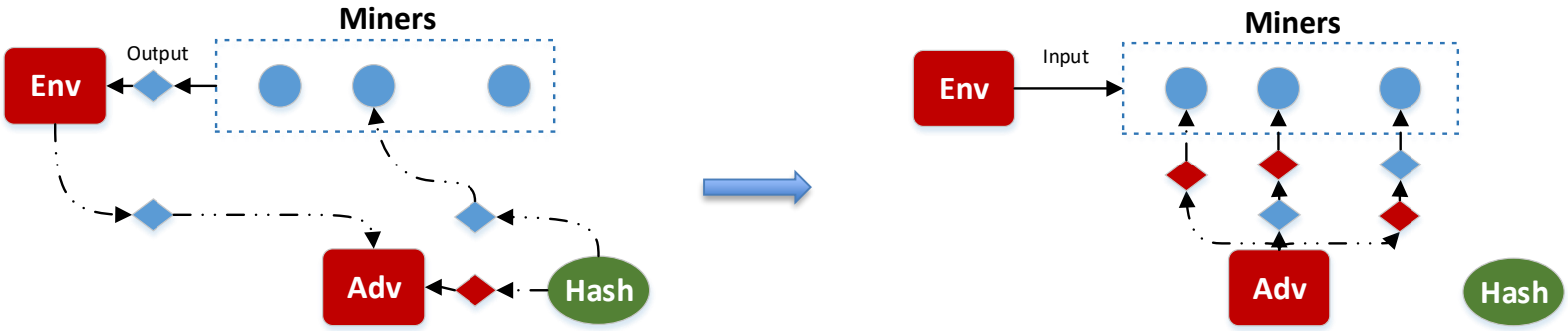
- 为了便于分析，采用同步模型，时间分割为Round。每个参与者（矿工）具有相同的计算能力（flat model）。
- 每个Round，每个参与者可以访问q次哈希函数，哈希函数抽象为随机寓言机（Random Oracle）
- 消息通过广播方式传播给其他参与者
- 攻击者可以随时访问网络（rushing）并且可以
  - 重发、插入、记录消息
  - 不能阻断诚实用户的消息

参与者模型

- 系统总共有n个参与者
- 其中t个参与者是 攻击者，攻击者可以协同工作
- 诚实参与者互相独立，仅通过广播信道传播消息
- 恶意参与者与诚实参与者的能力在证明部分描述



轮次结构

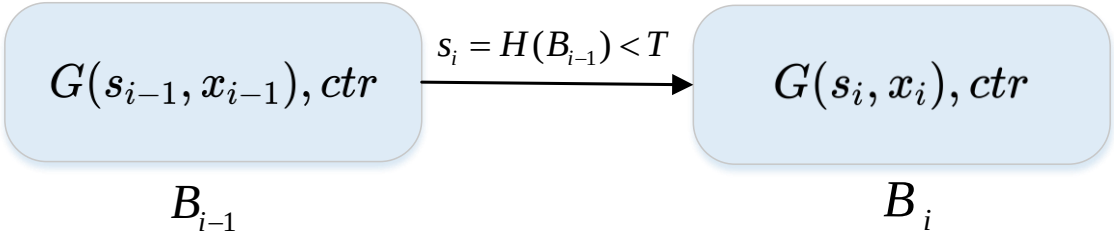


Round *i* 结束

Round *i+1* 开始

- 为了便于分析，采用同步模型，时间分割为Round
- 每轮中，参与者通过访问hash函数尝试生成新块并广播出去
- 下一轮所有用户收到广播传播的新块

Backbone协议（1）



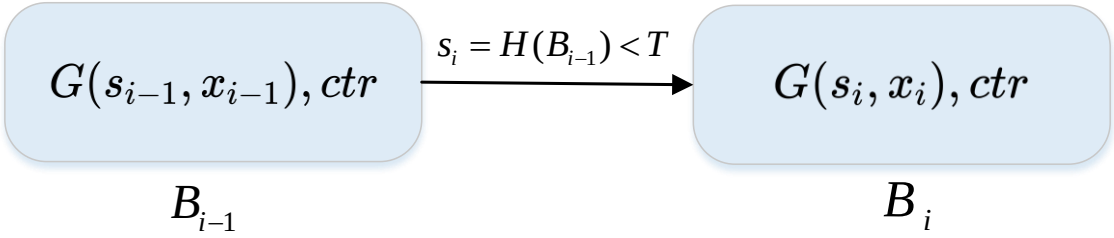
- 区块通过哈希值形成链式结构
- 结构的生成包括两个哈希函数G(),H()，分别用于内部数据压缩与工作量证明挖矿
- 三个虚化函数为上层应用提供服务:

  I()-接受应用层输入

  R()-接受网络数据

  V()-检验交易数据

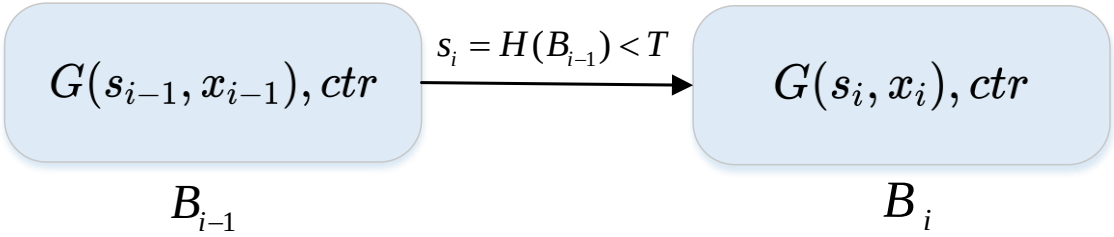
Backbone协议（2）



新区块产生：

- 诚实用户采集输入信息生成：$x_i = I(*)$
- 通过改变 $ctr = 0,1,2 \ldots\ldots$，访问RO尝试新块
- 如果满足下面条件即可生成一个有效新块：$\left(H\left(ctr; G(s_i; x_i)\right) < T\right) \wedge (ctr \leq q)$

Backbone协议（3）

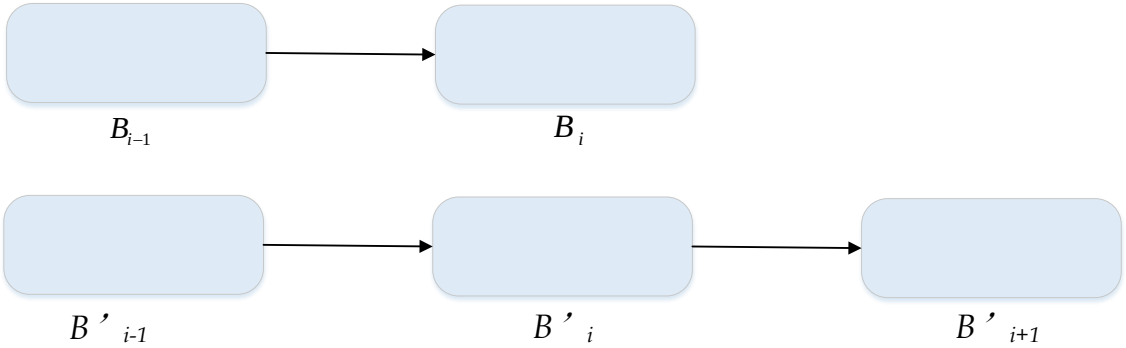


网络广播：

诚实用户如果产生了一个有效的新块，将扩展后的区块链通过广播传播给其他用户

广播信道是匿名且无认证的

Backbone协议（4）



最长区块链原则：

诚实用户选择本地的最长链作为输出

PoW函数

```
1: function pow(x, C)
2:     if C = ε then                          ▷ Determine proof of work instance
3:         s ← 0
4:     else
5:         ⟨s', x', ctr'⟩ ← head(C)
6:         s ← H(ctr', G(s', x'))
7:     end if
8:     ctr ← 1
9:     B ← ε
10:    h ← G(s, x)
11:    while (ctr ≤ q) do
12:        if (H(ctr, h) < T) then            ▷ This H(·) invocation subject to the q-bound
13:            B ← ⟨s, x, ctr⟩
14:            break
15:        end if
16:        ctr ← ctr + 1
17:    end while
18:    C ← CB                                 ▷ Extend chain
19:    return C
20: end function
```

通过工作量证明尝试生成一个新的区块

Maxvalid函数

```
1: function maxvalid(C_1, ..., C_k)
2:     temp ← ε
3:     for i = 1 to k do
4:         if validate(C_i) then
5:             temp ← max(C_i, temp)
6:         end if
7:     end for
8:     return temp
9: end function
```

在所有有效链中选择一个最长链

Validate函数

```
1: function validate(C)
2:     b ← V(x_C)
3:     if b ∧ (C ≠ ε) then                ▷ The chain is non-empty and meaningful w.r.t. V(·)
4:         ⟨s, x, ctr⟩ ← head(C)
5:         s' ← H(ctr, G(s, x))
6:         repeat
7:             ⟨s, x, ctr⟩ ← head(C)
8:             if validblock_q^T(⟨s, x, ctr⟩) ∧ (H(ctr, G(s, x)) = s') then
9:                 s' ← s                                  ▷ Retain hash value
10:                C ← C^⌈1                                ▷ Remove the head from C
11:            else
12:                b ← False
13:            end if
14:        until (C = ε) ∨ (b = False)
15:    end if
16:    return (b)
17: end function
```

验证一条链是有效的

主函数

1: **if** (init) **then**
2: $\mathcal{C} \leftarrow \varepsilon$
3: $st \leftarrow \varepsilon$
4: $round \leftarrow 1$
5: $init \leftarrow$ False
6: **else**
7: $\tilde{\mathcal{C}} \leftarrow \mathsf{maxvalid}(\mathcal{C}, \text{any chain } \mathcal{C}' \text{ found in } \textsc{Receive}())$
8: **if** $\textsc{Input}()$ contains $\textsc{Read}$ **then**
9: **write** $R(\tilde{\mathcal{C}})$ to $\textsc{Output}()$ ▷ Produce necessary output before the POW stage.
10: **end if**
11: $\langle st, x\rangle \leftarrow I(st, \tilde{\mathcal{C}}, round, \textsc{Input}(), \textsc{Receive}())$ ▷ Determine the x-value.
12: $\mathcal{C}_{\mathsf{new}} \leftarrow \mathsf{pow}(x, \tilde{\mathcal{C}})$
13: **if** $\mathcal{C} \neq \mathcal{C}_{\mathsf{new}}$ **then**
14: $\mathcal{C} \leftarrow \mathcal{C}_{\mathsf{new}}$
15: $\textsc{Diffuse}(\mathcal{C})$ ▷ Broadcast the chain in case of adoption/extension.
16: **else**
17: $\textsc{Diffuse}(\perp)$ ▷ Signals the end of the round to the diffuse functionality.
18: **end if**
19: $round \leftarrow round + 1$
20: **end if**

通过永久的循环，维持链的生成

1.找最长链

2.输入输出

3.PoW

4.广播

安全特性定义

fix
a protocol Π
a number of parties *n*, *t* of which
controlled by adversary
a predicate Q

We say that the protocol has property Q
with error ϵ if and only if

$$\forall \mathcal{A}\ \forall \mathcal{Z}\ \text{Prob}[Q(\text{VIEW}^{\Pi}_{\mathcal{A},\mathcal{Z}}(1^{\lambda})] \geq 1 - \epsilon$$

typically : $\epsilon = \text{negl}(\lambda)$

给定协议

给定参与者条件

设定某个断言（特性）

如果这个断言以可忽略的错误率成立，则该协议具有此特性

协议三个安全特性（非正式）

Common Prefix

- 任意两个诚实用户除了最后少量区块，他们的最长区块链的前缀相同

Chain Quality

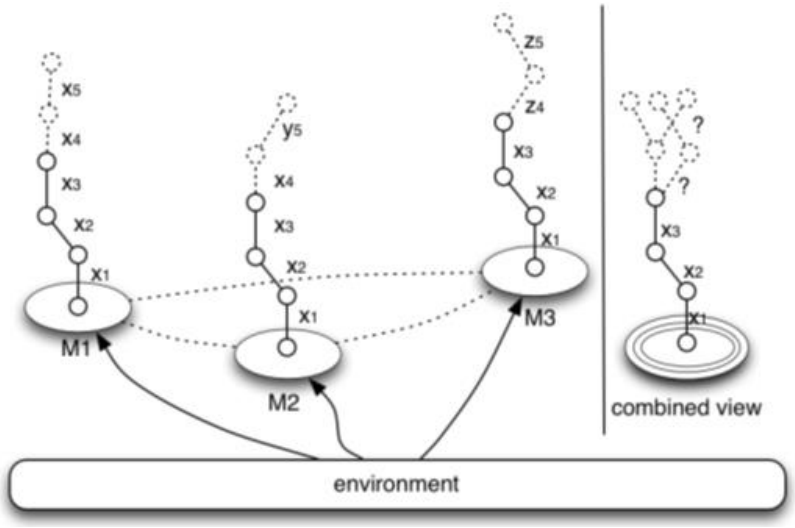
- 在任意足够长的连续区块中，一定有常数比例的区块由诚实用户生成

Chain Growth

- 诚实用户的区块链长度保持增长，并且增长速度不会过低

Common Prefix

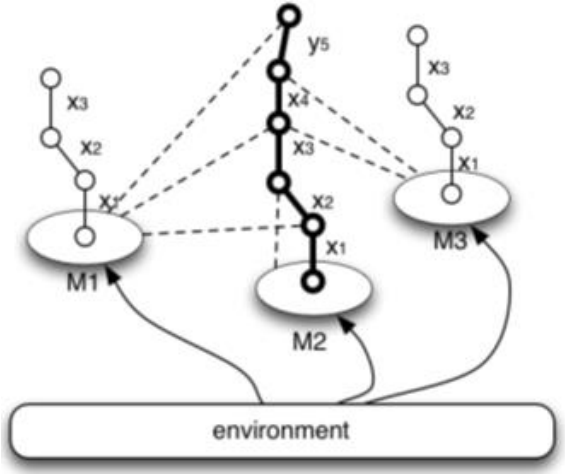
$$\forall r_1, r_2, (r_1 \leq r_2), P_1, P_2, \text{ with } \mathcal{C}_1, \mathcal{C}_2 : \ \mathcal{C}_1^{\lceil k} \preceq \mathcal{C}_2$$



Chain Quality

Parameters $\mu \in (0,1), k \in \mathbb{N}$

The proportion of blocks in any k-long subsequence produced by the adversary is less than μk

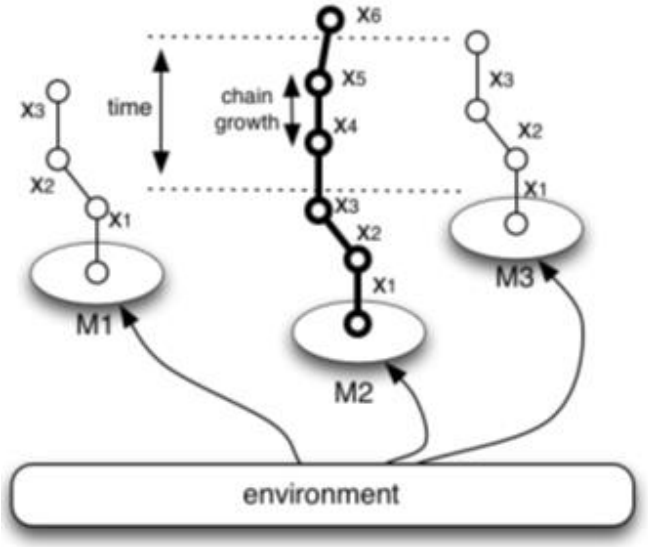


Chain Growth

Parameters $\tau \in (0,1), s \in \mathbb{N}$

$\forall r_1, r_2$ honest player P with chains $\mathcal{C}_1, \mathcal{C}_2$

$r_2 - r_1 \geq s \implies |\mathcal{C}_2| - |\mathcal{C}_1| \geq \tau s$



符号与假设

- n: 矿工总数（按照flat model假设每个人的计算能力都一样）
- t: 恶意矿工总数
- δ: 诚实矿工所占的优势 $t=(n-t)(1-\delta)$，$0<\delta<1$
- q: 每个矿工每轮可以做的RO查询数量
- p: 每次查询成功出块的概率

$$npq \ll 1$$

简单推论(1)

- 在任意连续的*s*轮中
- *X(s)*是诚实节点成功的轮次
- *Y(s)*是恶意节点成功的总次数

$$E(Y(s)) = tpqs \ll s$$

$$E(X(s)) \approx (n-t)pqs \ll s$$

$$(1-p)^{q(n-t)} \approx 1 - pq(n-t)$$

$$s - (1 - pq(n-t))s \approx (n-t)pqs$$

$$npq \ll 1$$

简单推论(2)

- 在任意连续的s轮中，大多数轮次没有新块产生

$$E(Y(s) + X(s)) \ll s$$

假设网络传播没有延迟

在大多数轮中，所有的诚实节点均在尝试延长同一条最长链

Chain Growth证明准备

- 几个观察：
  - 恶意节点无法缩短诚实节点最长链的长度
  - 任何诚实节点最长链的长度是随着时间单调递增的
  - 大多数轮次所有诚实节点具有相同的最长链
- 推论：
  - 大多数诚实节点成功轮次将带来诚实节点最长链的增长

Chain Growth证明概要

Parameters $\tau \in (0,1), s \in \mathbb{N}$

$\forall r_1, r_2$ honest player P with chains $\mathcal{C}_1, \mathcal{C}_2$

$r_2 - r_1 \geq s \implies |\mathcal{C}_2| - |\mathcal{C}_1| \geq \tau s$

在连续的s=r2-r1轮中，

$E(X(s)) \approx (n-t)pqs$

因为大多数成功的轮次将引起诚实节点最长链的增长

$|C_1| - |C_2| \approx (n-t)pqs$

Chain Quality证明准备

- 几个观察：
  - 恶意节点无法阻止诚实节点产生区块并传播给其他节点
  - 恶意节点可以并仅可以通过竞争使得诚实节点产生的区块失效
  - 链的增长速度满足前述Chain Growth特性
- 推论：
  - 如果不是因为竞争失败，诚实节点所产生的区块将留存在最长链上

Chain Quality证明概要

Parameters $\mu \in (0,1), k \in \mathbb{N}$

The proportion of blocks in any k-long subsequence produced by the adversary is less than μk

生成k个区块需要s轮，$^{k}/_{npq} < \mathrm{s} < {}^{k}/_{(n-t)pq}$，

在s轮中恶意用户最多产生$tpqs$个有效区块，同时区块链增长了$(\mathrm{n}-t)pqs$长度

最差情况下恶意节点所有区块竞争胜利，则仍有

$X' = (n-t)pqs - tpqs = (n-2t)pqs$个区块来自诚实节点

Common Prefix证明准备

- 几个观察：
  - 所有的诚实节点都会选择自己观察到的最长链
  - 链的增长速度满足前述Chain Growth特性
  - 诚实节点算力占优的假设下，恶意节点难以产生另外一条增长速度满足Chain Growth的链
- 推论：
  - 经过足够长的时间，仅有一条链增长速度达到Chain Growth

Common Prefix证明概要

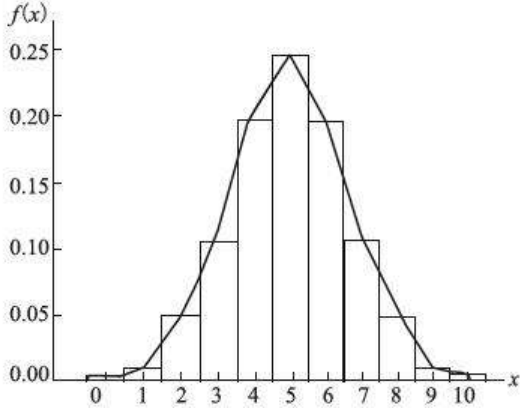
$$\forall r_1, r_2, (r_1 \leq r_2), P_1, P_2, \text{ with } \mathcal{C}_1, \mathcal{C}_2 : \mathcal{C}_1^{\lceil k} \preceq \mathcal{C}_2$$

令r_0是C_1,C_2的最后一个公共区块产生的时间，$s=r_1-r_0$, 考虑k是一个较大的参数

- 如果C_2在r_1轮时C'_2，与C_1几乎长度相同(或者长于C_1)，那么在s轮内，产生了两个满足Chain Growth的区块链，与前面的分析矛盾
- 如果C_2在r_1轮时C'_2, 长度显著小于C_1，则在r_2-r_1轮内，C_2反超C_1的概率很小，与C_2是P_2的最长链矛盾

从一般情况到极端情况

- Average Case: 按照统计概率积累足够时间得到的结果
- Worst Case: 在最坏情况下可能出现的特例
- 如：诚实用户算力是恶意用户的2倍，在统计概率下，一段时间内诚实用户生成的区块数量是恶意用户的2倍。在某个较短时期，恶意用户有概率生成的区块数量超过诚实用户。



- 安全性分析应保证在Worst Case下的安全性

从一般情况到极端情况(切诺夫不等式)

- 对于二项式分布

$$\mu = E[X]$$

$$\Pr[X \leq (1-\delta)\mu] \leq e^{-\delta^2 \mu/2}$$

$$\Pr[X \geq (1+\delta)\mu] \leq e^{-\delta^2 \mu/3}$$

在前述定理讨论中，如果s足够大，则μ足够大。Worst Case偏离Average Case的概率小于

$$e^{-\Omega(\mu)}$$

加入网络延迟

- 直觉：
- 网络延迟将造成诚实用户收到的最长链不一致
- 诚实用户在非最长链上挖矿，产生临时分叉，造成诚实用户算力的浪费
- 有限的网络延迟不会影响诚实用户收到新块，永久的网络延迟将造成用户的割裂

定理：
设诚实用户在一轮中产生一个新区块的概率为α_0，一个新区块传播到其他诚实节点的延迟为Δ，计算诚实用户在该模型下的等效出块概率α。考察一段时间t，假设在这段时间一共产生了c个区块。每次产生一个新区块，诚实用户将会浪费Δ时间，因此总体有效时间为$t - c\Delta$。因此有$c = \alpha_0(t - c\Delta)$。得到

$$\alpha = \frac{c}{t} = \frac{\alpha_0}{1 + \Delta\alpha_0}$$

PoW共识协议进一步分析

- Adaptive Difficulty
  - 自动调整挖矿难度
- Non-Flat
  - 每个矿工拥有不同的计算能力
- Rational Model
  - 经济激励下的安全性(Selfish Mining)

自私挖矿攻击

- 恶意矿工拥有一定比例的算力，当其成功挖出一个区块后，不公布该区块，而是试图在其后继续挖块
  - 诚实节点生成一个区块，则恶意矿工尝试用先挖优势与诚实区块竞争
  - 假设恶意节点所占算力为a，诚实节点所占算力为1-a

自私挖矿攻击

竞争情况	描述	收益结果	发生概率
1	恶意矿工在诚实节点出块前生成一个新块	2	a
2	恶意节点在诚实新块之前没有生成一个区块但是竞争成功	1	(1-a)/2
3	恶意节点在诚实新块之前没有生成一个区块但是竞争失败	0	(1-a)/2

- 收益的数学期望

$$2a + \frac{1-a}{2} > 1$$

$$a > 1/3$$

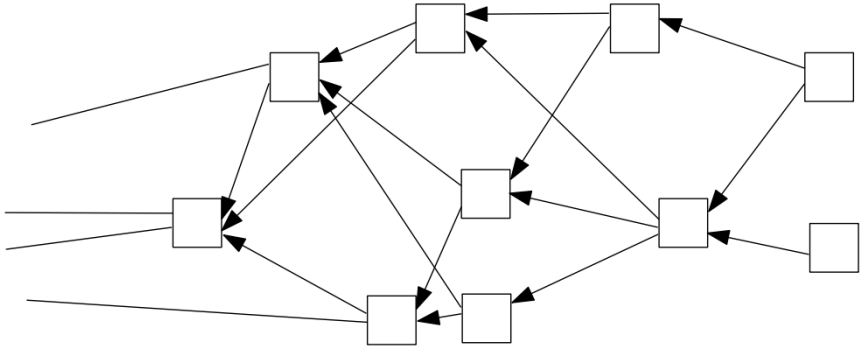
DAG共识协议——PoW的发展

动机：PoW共识中，提高出块速度将带来更多分叉，DAG可能带来更高的吞吐率，适用于弱同步网络

构造：每个区块（交易）引用不止一个前序区块，形成有向无环图

问题：如何对交易做唯一的确认与排序?

解决：通过最重子图等图算法计算确定区块顺序，算法复杂度较高

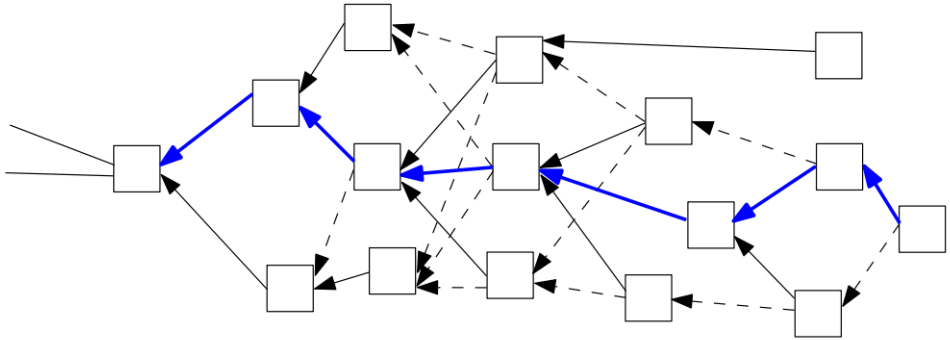


基于有向无环图（Directed Acyclic Graph, DAG）的分布式账本

DAG的进一步发展

动机：简化DAG共识算法交易排序的复杂度

构造：简化有向无环图图本身的复杂度，每个区块指向唯一的父区块，同时指向若干引用区块



基于主干链的 DAG 共识协议

谢谢

作业

- 利用(Python\Go\Rust)实现一个PoW的仿真程序，模拟一定数量的节点生成区块链的状态。
- 设置参数包括：节点数量、每个轮次出块的成功率
  - 测量区块链的增长速度
- 设置一定数量的恶意节点实施攻击
  - 测量不同恶意节点比例（10%-40%）条件下，统计分叉攻击成功的长度
  - 测量不同恶意节点比例条件下，自私挖矿收益比例