# Hyperledger Fabric 共识机制优化方案

孟吴同 [1,2] 张大伟 [1,2]

**摘 要** 针对 Hyperledger Fabric 使用固定背书节点处理交易所带来的安全风险和性能瓶颈问题, 提出了一种非交互、可验证的随机化背书节点优化方案. 基于 “背书-排序-验证” 的 Hyperledger Fabric 共识模型, 引入背书节点候选集, 使用可验证随机函数随机抽取背书节点进行交易背书, 实现了可验证情况下背书节点的非交互式随机选取和背书过程的并行处理. 分析和实验表明, 优化后的共识机制具有更高的安全性和更快的交易处理速度.

**关键词** 区块链, 共识机制, 可验证随机函数, Hyperledger Fabric

**引用格式** 孟吴同, 张大伟. Hyperledger Fabric 共识机制优化方案. 自动化学报, 2020, **46**(x): 1−14

**DOI** 10.16383/j.aas.c190516

## Optimization Scheme for Hyperledger Fabric Consensus Mechanism

MENG Wu-Tong[1,2] ZHANG Da-Wei[1,2]

**Abstract** To solve the problem of the security risk and processing bottleneck caused by handling all transactions with a fixed endorsement node in Hyperledger Fabric, a non-interactive and verifiable randomized endorsement scheme is proposed. On the basis of the “endorse-sort-verify” consensus mechanism, the non-interactive and verifiable random selection of the endorsement node and the parallel endorsement process are implemented by introducing the endorsement node candidate set and using the verifiable random function to randomly extract the endorsement node to endorse each transaction. Analysis and experiments show that the optimized consensus mechanism has higher security and faster transaction processing speed.

**Key words** blockchain, consensus mechanism, verifiable random function, Hyperledger Fabric

**Citation** Meng Wu-Tong, Zhang Da-Wei. Optimization scheme for hyperledger fabric consensus mechanism. *Acta Automatica Sinica*, 2020, **46**(x): 1−14

来源于比特币[1] 的区块链融合了数学、密码学、计算机科学等多种技术, 其去中心化、不可篡改、多方维护等特性使得区块链得到了各个领域的 广泛关注. 随着人们对区块链技术的深入研究和普及推广, 区块链很可能对当今互联网产生颠覆式的影响.

区块链技术的一个核心问题是在无中心、弱信任的分布式系统中让各个节点达成共识. 所谓共识, 简单来说是让分布式系统中各 个节点对某个值或某种状态产生相同的看法[2]. 达成共识的方法就是区块链的核心技术 - 共识机制. 共识机制需要注重安全性、效率和能耗等问题[3], 从而保证区块链系统能够正 常运行. 目前主流的共识机制有工作量证明 (Proof of Work, PoW), 权益证明 (Proof of Stake, PoS)、授权权益证明 (Delegated Proof of Stake, DPoS) 等证明类共识机制, 以及实用拜占庭容错 (Practical Byzantine Fault Tolerance, PBFT) 等传统分布式系统一致性算法[4]. 不同于前面提到的几种共识机制, Hyperledger Fabric 采用 “背书-排序-验证” 的方式来达成共识[5].

Hyperledger Fabric 将应用层的信任模型同底层的共识协议解耦, 拆分共识过程为交易内容合法性验证和交易顺序一致性保证两个步骤. 背书和验证保证交 易内容的合法性, 排序保证交易顺序的一致性. 通过将这两部分进行解耦[6], 智能合约的开发者可以设计更为灵活的信任模型, Hyperledger Fabric 网络也获得了更快的交易处理速度. 但是在 Hyperledger Fabric 中, 背书节点属于关键节点, 数量较少却承担着对所有交易内容的合法性进行背书的重要任务; 并且为了便于交易客户 端识别和验证, 背书节点的身份必须公开. 在区块链网络中身份公开且承载着大量的敏感交易数据, 这必然使得背书节 点成为敌手的首要攻击目标. 此外, 由于每个背书节点都要处理所有的交易, 因此使其成为系统中交易处理的性能瓶颈, 单个背书节点的处理能

收稿日期 2019-07-07 录用日期 2019-12-15
Manuscript received July 7, 2019; accepted December 15, 2019
国家留学基金 (201807095023) 资助
Supported by China Scholarship Council (201807095023)
本文责任编委 葛世超
Recommended by Associate Editor
1. 智能交通数据安全与隐私保护技术北京市重点实验室 北京 100044 2. 北京交通大学计算机与信息技术学院 北京 100044
1. Beijing Key Laboratory of Security and Privacy in Intelligent Transportation, Beijing 100044 2. School of Computer and Information Technology, Beijing Jiaotong University, Beijing 100044

力限制了整个区块链网络的交易速度.

Hyperledger Fabric 将应用层信任同底层共识协议相解耦的共识模型非常利于构建基于业务逻辑的共识机制, 因此在实际系统中得到了广泛应用. 目前 Hyperledger Fabric 共识机制的相关研究工作在对背书节点高度信任的假设前提下多侧重于排序过程的改进, 即对交易顺序一致性进行优化[7]. 而忽视了对于背书节点和 共识机制的安全性考量. 事实上由于交易安全性高度依赖于固定身份的背书节点, 背书节点一旦被恶意攻击, 最终账本数据的正确性将无从保证. 公有链系统 所面临的开放式网络环境使得其共识机制的设计充分考虑了安全性问题, 多使用随机抽取共识节点的方法使敌手难以选择攻击目标, 从而提高攻击难度. 因此, 本文针对联盟 链中共识机制的安全性问题, 在 "背书-排序-验证" 共识模型的基础上引入非交互式可验证随机节点抽取方法, 提出了一种 Hyperledger Fabric 共识机制的优化方案:

1) 引入背书节点候选集, 使用可验证随机函数 (Verifiable Random Functions, VRF) 的随机输出特性在非交互模式下抽取背书节点, 降低背书节点中心化程度, 增加系统的抗攻击能力;

2) 交易客户端利用可验证随机函数的验证特性来完成背书节点的身份认证, 从而在节点随机抽取过程中确保背书节点身份的真实性和可验证性;

3) 将原有的一组背书节点为所有交易背书, 改进为多组背书节点为交易进行背书, 使用并行处理的方式, 提高交易处理速度.

本文结构如下: 第一节概述了区块链共识机制需考虑的关键问题、可验证随机函数 的定义以及可验证随机函数在区块链共识机制中的应用; 第二节详细介绍了 Hyperledger Fabric 的共识机制及其存在的问题; 第三节针对 Hyperledger Fabric 存在的问题提出了具体的优化方案; 第四节对优化方案的安全性和性能进行了分析; 第五节构建了优化方案的原型系统, 对方案的性能进行了对比测试; 第六节对本文工作进行了总结.

## 1 共识机制与可验证随机函数

### 1.1 共识机制

区块链技术本身是基于分布式系统的, 因此区块链技术仍然需要解决一些分布式系统中存在的问题. 其中最核心的问题是如何高效的就不同节点数据一致性达成共识, 因此需要从安全性和共识速度两个角度设计共识算法[8].

共识问题受多种系统因素的影响, 比如节点数量的多少, 是否具有拜占庭节点等. 节点数量是区块链系统去中心化程度的直观表现. 节点数量与账本的安全性成正比, 与共识难度成反比. 即节点数量越多, 账本的备份也就越多, 攻击者修改账本的难度越高, 数据越安全, 但是让这些节点的账本保持一致性的难度也越高[9]. 拜占庭节点来源于拜占庭将军问题[10], 是指系统中因运行故障 (如, 被恶意攻击) 而发送错误信息的节点. 在不具有拜占庭节点的网络中, 仅存在数据丢包和延迟等非恶意行为; 而在具有拜占庭节点的网络中, 恶意节点可能会主动发送错误信息, 以谋取利益. 在不考虑非拜占庭节点的情况下, Paxos[11], Raft[12] 等算法可以实现较为高效的共识, 但是现实中很难构建一个不存在非拜占庭节点的系统, 因此大多数共识算法基于的安全模型中都是存在拜占庭节点的.

传统分布式一致性算法一般采用协商的方式解决共识问题[13], 如 1999 年, Barbara Liskov 等提出的 PBFT[14] 算法 (Practical Byzantine Fault Tolerance, PBFT) 可以在保证活性 (Liveness) 和安全性 (Safety) 的前提下提供 (n-1)/3 的容错性 (其中 n 为节点总数). PBFT 算法共识速度快且易于实现, 但是 PBFT 算法的消息复杂度 ($O(N^2)$) 会随节点的增加而增加, 因此不适用于大规模节点的场景.

从另一个角度讲, 拜占庭节点可以理解为被敌手攻击的节点, 减少拜占庭节点的产生, 需要增加敌手的攻击难度. 为此, 一方面可以增加节点数量, 另一方面是增加出块节点的随机性[15]. 传统分布式一致性算法往往会因为节点数量的增加而带来巨大的通信量, 因此更多共识算法选择后者来保证安全性.

以工作量证明为例, 工作量证明[1](Proof of Work, PoW) 使用 "挖矿" 的方式随机选取出块者. "挖矿" 是指每个节点不断尝试解决一个求解困难但是验证容易的数学难题. 最快解决该难题的节点会获得发布下一区块的权利 (记账权) 以及系统的奖励. PoW 的随机性依赖于哈希函数值的均匀分布, 但是大量的哈希计算伴随着巨大的能源消耗, 而且这种消耗所用来解决的问题是无意义的[16]. 同时随着 "矿机" 的产生及发展, 计算能力逐渐被一些大的矿池垄断, 也对系统安全性产生了威胁. 此外 PoW 共识的效率较低, 过长的出块时间和交易确认时间难以满足现实需求.

权益证明[17](Proof of Stake, PoS) 同样是使用 "挖矿" 的方式选取出块者, 但是 "挖矿" 成功的概率与节点的权益有关, 节点持有的权益越大, "挖矿" 成功的概率越高, 这从整体上加快了区块的产出速度, 提高了共识的效率, 同时计算量不再是 "挖矿" 的主要影响因素, 从而减少了大量计算带来

的资源浪费. 但是 PoS 中存在的 “无利害攻击”, “长程攻击” 等问题使 PoS 的安全性备受质疑[18].

在 PoS 的基础上, 授权权益证明[19](Delegated Poof of stake, DPoS) 通过牺牲一定的 “去中心化” 特性以实现更高的共识效率. 每个节点可以将其持有的权益作为选票投给一名代表, 最终投票最多的前 N 个用户会构成参与共识的 “委员会”, 每一个委员轮流负责对交易进行打包, 生成区块. 由于参与共识节点数量的减少, DPoS 具有很高的交易速度. 但 DPoS 在去中心化过程中所产生的代理节点令敌手的攻击目标更加明确, 从而降低了敌手的攻击成本.

从上述共识算法可以看出, 随机选取节点可以增加共识机制的安全性, 因此随机选取方式的设计变得尤为关键, 随机选取算法既要保证结果的随机性还要具有较高的运行效率.

### 1.2 可验证随机函数

可验证随机函数 (Verifiable Random Function, VRF)[20] 是由 Silvio Micali 等人于 1999 年提出的. VRF 本质上是一类具有验证功能的伪随机函数. 对于一个特定的输入 $x$ 以及输入者的私钥 $SK$, VRF 会输出一个随机数 $v$ 以及一个证明 $proof$, 验证者可以通过输出的随机数、证明和输入这三部分验证出随机数是否是由该输入产生. 这个过程不必暴露输入者的私钥, 因此是安全的.

VRF 需要满足三个性质: 可验证性、确定性和随机性:

- 可验证性是指通过 $proof$ 可以验证出 $v$ 是 $SK$ 和 $v$ 对应的输出.
- 确定性是指在 $SK$ 和 $x$ 不变的情况下, 输出的 $v$ 也是不变的.
- 随机性是指在不给定证明 $proof$ 的情况下, VRF 的输出 $v$ 与一个随机数对于敌手来说两者之间是不可区分的.

文献 [21] 中的 VRF 具体定义如下: VRF: 设 $G$, $F$, $V$ 是多项式时间算法:

- $G$ 是一个概率性算法, 其输入是安全参数 $k$, 输出公钥 $PK$ 和私钥 $SK$;
- $F=(F_1,F_2)$ 是一个确定性算法, 输入包括 $x$ 和私钥 $SK$, 输出为 $value=F_1(x,SK)$ 和 $proof=F_2(x,SK)$, 其中 $value$ 是 VRF 输出的随机值, $proof$ 是对随机值的证明;
- $V$ 是一个概率性算法, 输入为: $(PK,x,v,proof)$, 输出为 YES 或 NO.

设 $a:\mathbf{N}\to\mathbf{N}\in\{*\};b,s:\mathbf{N}\to\mathbf{N}$ 是任意三个 poly $(k)$ 时间内可计算的函数, 并且都以一个 $k$ 的多项式为上界 (除非 $a\in\{0,1\}_*$), 其中 k 为安全参数. 我们称 (G, F, V) 是输入长度为 $a(k)$、输出长度为 $b(k)$、安全性为 $s(k)$ 的可验证的伪随机函数 (VRF), 如果满足以下几条性质:

(1) 下面两条以 $1-2^{-\Omega(k)}$ 的概率成立, 其中 $(PK,SK)\xleftarrow{R}G(1_k)$:

1) 值域的正确性:

对于任意 $x\in\{0,1\}^{a(k)}$, $F_1(SK,x)\in\{0,1\}^{b(k)}$

2) 完全可证性:

对于任意 $x\in\{0,1\}^{a(k)}$, 如果 $v=F_1(SK,x)$ 且 $proof=F_2(SK,x)$ 那么

$$Pr[V(PK,x,v,proof)=\text{YES}]>1-2^{-\Omega(k)} \quad (1)$$

(2) 唯一可证性:

对每一个 $PK$, $x$, $v_1$, $v_2$, $proof_1$, $proof_2$, 其中 $v_1\neq v_2$, 下面的不等式对于 $i=1$ 或 $i=2$ 都成立:

$$Pr[V(PK,x,v_i,proof_i)=\text{YES}]>1-2^{-\Omega(k)} \quad (2)$$

(3) 剩余伪随机性:

$T=(T_E,T_J)$ 是任意一对算法, 满足第一次输入为 $1_k$ 时, $T_E(\cdot,\cdot)$ 和 $T_J(\cdot,\cdot)$ 总共运行了最多 $s(k)$ 步, 则 T 在下面的实验中成功的概率至多为 $1/2+1/s(k)$:

1) 运行 $G(1_k)$, 得到 $(PK,SK)$.

2) 运行 $T_E^{F(SK,\cdot)}(1^k,PK)$, 得到 $(x,state)$.

3) 选择 $r\xleftarrow{R}\{0,1\}$:

①如果 $r=0$, 选择 $v=F_1(SK,x)$

②如果 $r=1$, 选择 $v\xleftarrow{R}\{0,1\}^{b(k)}$

4) 运行 $T_J^{F(SK,\cdot)}(1^k,PK)$, 得到值 $guess$.

5) 如果 $x\in\{0,1\}^{a(k)}$, $guess=r$, 并且 $x$ 没有被 $T_E$ 或 $T_J$ 作为 $F(SK,.)$ 的查询串查询过, 则 $T=(T_E,T_J)$ 成功.

### 1.3 可验证随机函数在共识机制中的应用

可验证随机函数在提出之后, 一般主要用在密钥封装[21]、数字签名等领域. 随着近年来人们对区块链技术的深入研究, 可验证随机函数的特性使其开始被应用在共识机制的设计中.

2016 年, Sivio Micali 等人提出的 Algorand[22], 将可验证随机函数与 PBFT 算法相结合. 使用可验证随机函数选取出块者, 之后采用类似于 PBFT 的协议进行通过若干轮投票, 而每一轮的投票者也是通过可验证随机函数选出的. 该共识算法具有较高的安全性和交易速度.

2016 年 Timo Hanke 等人提出的 Dfinity[23], 将全网中节点分为多组委员会, 当前负责出块的委员会中的每个成员使用基于 BLS 门限签名算法[24] 的

可验证随机函数选择出块节点, 然后阈值中继技术 (threshold relay technology) 选出其下一轮负责出块的委员会, 从而达到安全而快速生成区块的目的.

2017 年 Kiayias 等人提出的 Ouroboros[25] 共识算法在 PoS 的基础上, 使用秘密共享和追随中本聪算法选取出块节点. 算法本身虽然经过安全性证明, 但是其出块节点身份可以被提前获知, 可能导致敌手的攻击, 因此 2018 年 Ouroboros 的第二个版本 Praos[26] 使用可验证随机函数代替原有的选取算法来抽取出块节点, 增强了原有共识机制的安全性.

由此可见, 可验证随机函数已在区块链领域成为非交互模式下节点身份随机选取的重要工具.

需要注意的是, Algorand、Dfinity、Ouroboros Praox 均是公有链系统, 网络中的节点身份是对等的, 因此他们面临的问题一般是如何从全网的节点中安全的选出一个出块节点. 而 Hyperledger Fabric、Corda[27] 等联盟链系统则采用分层信任模型, 并且达成共识的过程也解耦为应用层信任和底层共识两部分, 需要具有不同职能的节点共同参与. 其中一些节点会对交易内容的合法性进行背书, 因此在共识过程中承担着重要任务, 我们称之为关键节点, 如 Hyperledger Fabric 中的背书节点和 Corda 中的 Notary 节点. 这些节点的身份一般会在区块链网络构建之初就确定下来, 因此攻击者可以直接确定攻击目标, 一旦攻击成功, 会对整个系统造成严重危害. 所以这些联盟链系统面临的问题是如何降低关键节点被攻击的风险. 通过借鉴公有链中利用可验证随机函数抽取节点的方式, 将关键节点的身份由固定变为随机抽取的, 可以使敌手的攻击目标不再明确, 这样就降低了关键节点被攻击的风险.

## 2 Hyperledger Fabric

Hyperledger 是 linux 基金会于 2015 年主导发起的开源项目, 其子项目 Hyperledger Fabric 是一个允许多方参与、开发、部署和运行区块链应用的联盟链平台[7]. Hyperledger Fabric 旨在创造一个模块化和可扩展的区块链开发框架, 为企业级区块链应用的开发提供解决方案. Hyperledger Fabric 区块链系统主要包含以下几部分组件:

链码 (Chaincode): 链码是 Hyperledger Fabric 中的智能合约, 它以代码的形式将复杂的业务逻辑固定在 Fabric 系统中. 当满足特定条件时, 链码会被执行[28].

客户端 (Client): 客户端是用户与 Hyperledger Fabric 网络的接入点, 其上会部署专有的 SDK. 用户可使用客户端发起交易请求, 即提案 (Proposal).

背书节点 (Endorser): 在 Hyperledger Fabric 中, 当客户端想要发起交易时, 首先需要为交易得到一定数量的背书 (Endorsement), 这些背书来自背书节点. 背书节点会通过运行链码模拟执行交易, 生成读写集, 之后会为该交易进行背书 (对读写集进行签名, 并附加自己的身份), 以证明该交易已经过背书节点处理[29].

排序节点 (Orderer): 在 Hyperledger Fabric 中通过多个排序节点来提供排序服务[30]. 排序服务接收来自全网的所有交易并将交易按照时间排序、打包成块. 排序服务并不参与交易的执行与验证, 因此并不关心交易的具体内容, 排序服务的目标是就交易产生的顺序达成一致的结果, 并将这个结果广播出去.

提交节点 (Committer): 提交节点是 Hyperledger Fabric 网络中账本维护的主体. 提交节点接收排序服务打包的区块, 验证区块中交易的有效性, 并据此将有效交易提交到账本. 另外, 背书节点也属于提交节点, 背书是其在维护账本基础上的额外功能.

### 2.1 Hyperledger Fabric 共识机制

如图 1 所示, Hyperledger Fabric 共识机制如下:

(1) 客户端创建一个提案, 依据背书策略将其发送给相应的背书节点. 提案包含了需要调用的链码函数及其参数、时间戳和客户端签名等信息;

(2) 背书节点会验证客户端签名, 确保提案是由已认证的客户端发出. 否则终止交易. 背书节点执行背书过程, 即根据提案使用链码模拟执行交易请求, 生成执行结果并附加背书节点的签名. 模拟执行的结果是一组基于当前账本状态的读写集, 也就是说背书过程并不会更改当前账本的状态;

(3) 客户端验证背书结果的签名, 确保其来自合法的背书节点. 否则终止交易. 由于一笔交易可能会需要多个背书节点进行背书, 客户端会收集来自不同节点的背书结果并比较这些结果中的读写集是否一致. 如果一致则根据背书结果生成交易 (transaction, 简记为 tx) 然后发送给排序服务, 否则终止交易.

(4) 当存在多个排序节点时, Hyperledger Fabric 使用 Kafka[31] 对交易进行排序. 排序节点在收到来自客户端的交易后, 会将接收到的交易发送到 Kafka 集群中, 同时会按照一定的规则从 Kafka 集群中读取一定数量的排序好的交易, 然后打包成块.

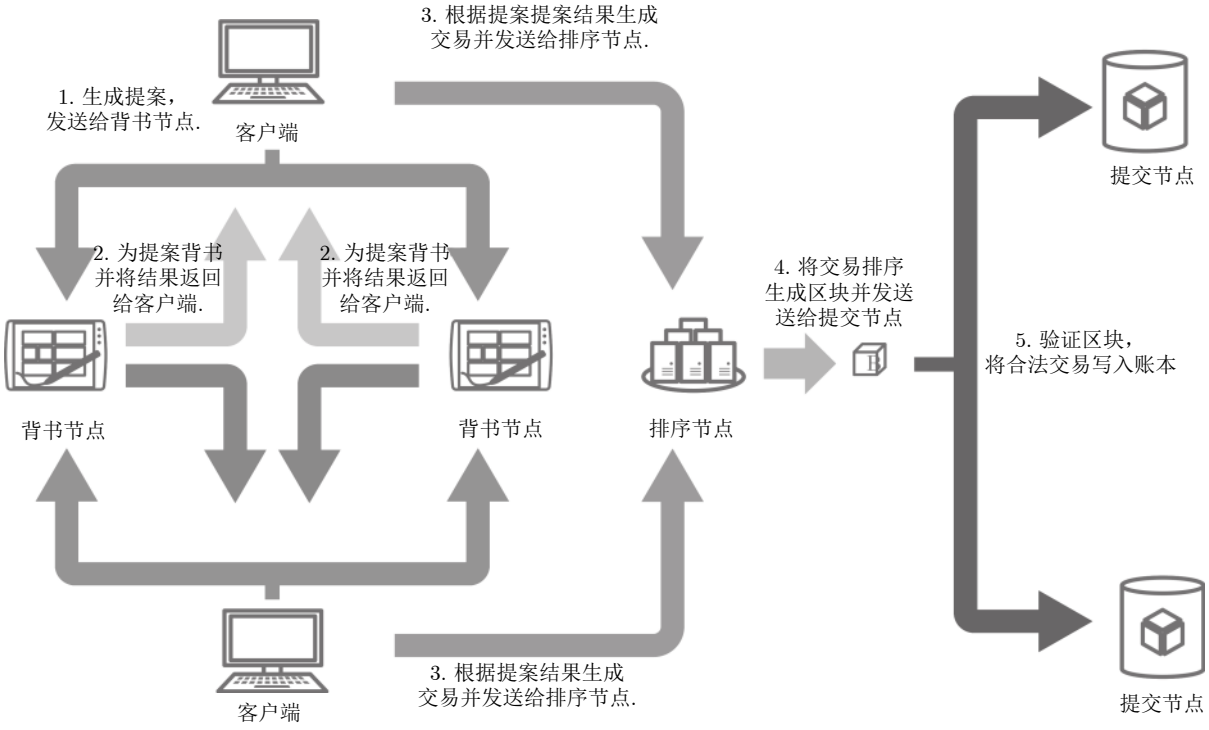


图 1 Hyperledger Fabric 共识机制示意图

Fig. 1 Diagrammatic sketch of hyperledger fabric consensus mechanism parameters

排序服务对区块签名后, 将区块分发给提交节点;

(5) 当提交节点收到区块后, 会对区块进行验证. 提交节点使用读写集合的读集部分来检查交易的有效性, 如果一笔交易通过了有效性验证, 提交节点会使用写集更新状态数据库.

当各个提交节点完成以上操作后, 可以视为对该客户端发起的交易达成了共识. 在这一过程中, 背书策略和背书过程确保了交易 (链码) 按照业务逻辑信任模型正确执行; 排序服务确保了节点对交易顺序的一致性达成共识; 提交节点根据相同的规则对每笔交易的数据进行验证, 从而在交易内容的合法性上达成共识. 由此可见, Hyperledger Fabric 通过 “背书-排序-验证” 的共识机制有效地将应用层的信任模型同底层的共识协议相解耦, 从而实现了更为灵活的信任模型和更高的交易吞吐量.

### 2.2 存在的问题

Hyperledger Fabric 将节点的身份进行划分, 在一定程度上简化了交易流程提高了交易速度, 但是也使系统产生了一些较为 “中心化” 的节点–背书节点. Hyperledger Fabric 背书的目的是为了在保证交易合法性的同时, 提高交易处理的速度[32]. 因为如果去除背书环节, 则各个节点为了达成共识, 都需要执行提案 (即调用链码), 这样会限制交易的处理速度. 而通过一部分背书节点提前去执行提案, 再让所有的节点只对结果进行检验, 则可以在牺牲一定安全性的条件下达到较高的交易处理速度[33].

但这一共识机制存在如下 2 方面的问题:

1) 首先, 背书节点在区块链网络搭建之初就已经确定且客户端必须能够识别并信任, 因此其身份必须公开且可验证. 其次, 背书节点在提案执行过程中会直接处理敏感的交易数据. 上述两点必然导致背书节点会成为攻击者的首要目标;

2) 背书节点的数量相较于全部节点数量而言占比极少, 部分系统中的背书节点数仅为个位数, 而这些节点却要处理系统中的全部交易, 这就导致了明显的交易速度瓶颈. 同时, 客户端与背书节点是多对一的关系, 当客户端产生大量交易时, 背书节点可能难以及时并发处理, 造成交易处理时间增加, 甚至导致背书节点的崩溃.

上述问题产生的根本原因在于 Hyperledger Fabric 在共识过程中缺少对背书节点的动态随机选取过程.

## 3 改进方案

为了解决 Hyperledger Fabric 共识机制在安全和性能两方面的问题, 我们在背书节点的确认中引入动态随机选取过程, 以实现节点身份的隐藏和节

点数量的动态扩展. 但在这一过程中, 同时需解决如下 2 个问题:

(1) 背书节点随机选择的过程应是非交互式的, 以降低系统的通信和计算开销, 减少选择过程对交易性能的影响;

(2) 选择后的背书节点身份必须能够被客户端识别和信任, 即选择算法应同时具有身份认证功能, 以避免节点身份伪装攻击的问题.

在基于 fabric 基本共识机制框架的基础上, 我们设计了一种非交互、可验证的随机化背书节点优化方案. 优化方案引入了背书节点候选集, 通过可验证随机函数在候选集中随机选取交易背书节点完成交易背书. 方案的改进一方面实现了交易背书前背书节点的身份隐私保护; 另一方面也动态随机扩展了交易背书节点的数量, 提高了交易处理能力.

### 3.1 优化后的共识机制

优化后的共识机制如图 2 所示:

(1) 客户端生成提案 proposal＜req, s＞$_{sig}$, 其中 req 为交易数据, 包括希望调用的 chaincode 及其参数. $s$ 为客户端选择的随机值, 作为节点身份抽取算法的种子. 客户端对 proposal 签名后将其发送给背书节点候选集. 交易发送成功后, 客户端会启动一个计时器;

(2) 各个候选背书节点收到客户端的 proposal 后, 首先根据签名验证 proposal 的完整性, 验证失败则终止交易. 候选背书节点执行背书节点身份抽取算法 $(r, proof, result) = VRF_R result(s, SK)$, 并根据 $result$ 判断自己是否为背书节点.

(3) 如果确定自己是背书节点, 执行提案并生成读写集 rw_set 以及背书结果 edm. 随后生成提案响应信息:proposal_response＜ rw_set, edm, $(r, proof)$＞$_{sig}$. 背书节点抽取算法流程如下所示:

1) 根据输入生成随机数及其证明

$(r, proof) = F(s, SK)$

2) 返回抽签结果

依据 $r$ 计算抽取结果并与阈值进行 $\lambda$ 比较, 其中 hash() 为密码杂凑算法, hashlen 是 hash 算法的输出长度, 如果 $\frac{\text{hash(r)}}{2^{\text{hashlen}}} > \lambda$, 返回 $(r, proof, yes)$; 否则返回 $(r, proof, no)$.

(4) 在计时器结束前的这段时间内, 客户端持续收集来自不同背书节点的 proposal_response 并根据签名验证 proposal_response 的完整性, 验证失败则终止交易. 使用背书节点身份验证算法 VRF_Verify() 验证该节点是否为合法的背书节点. 如果不是则丢弃其该背书结果. 在合法的 proposal_response 中, 如果大部分 (超过一半) 读写集一致, 则根据这些背书结果生成交易 tx＜r_w_set, {edm}$_k$ ＞$_{sig}$. 其中{edm}$_k$表示来自 $k$ 个合法背书

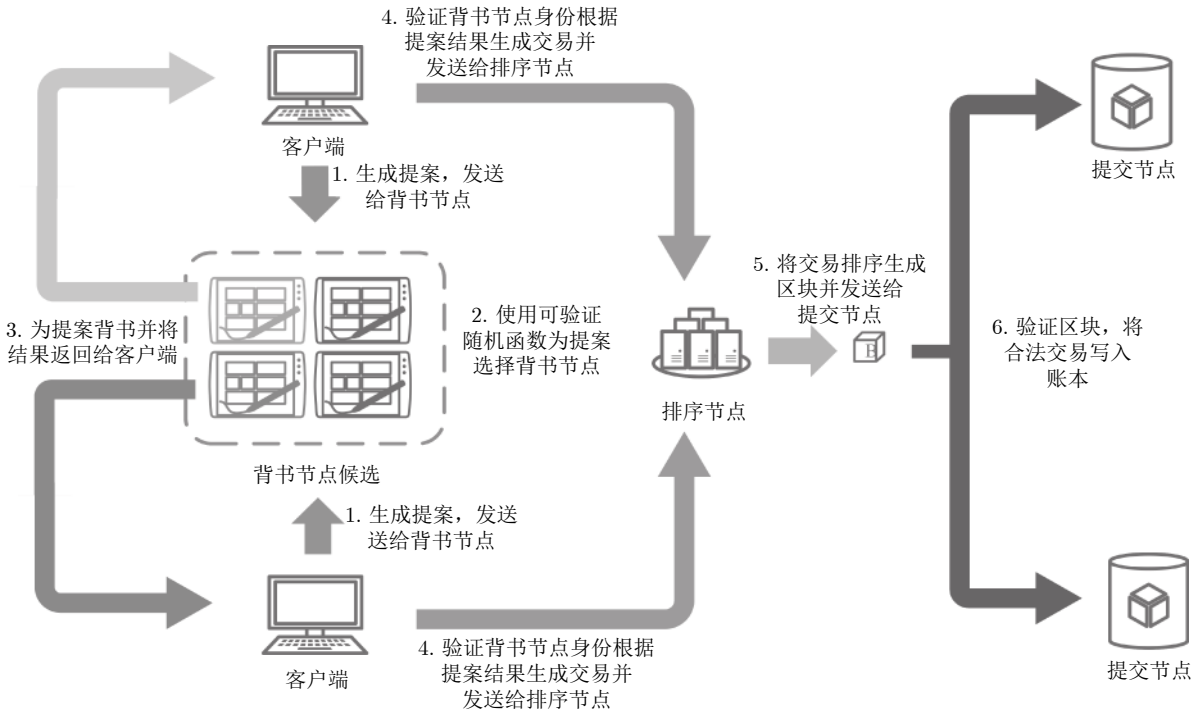


图 2 优化后的 Hyperledger Fabric 共识机制

Fig. 2 Diagrammatic sketch of optimized hyperledger fabric consensus mechanism

节点的签名. 客户端将交易 tx 签名后发送给排序节点. 背书节点身份验证算法流程如下所示:

1) 验证随机数合法性

$bool = V(s, r, PK)$

2) 返回验证结果

如果随机数合法且满足阈值条件, 认定其为背书节点. 如果 $bool$ & $\frac{\text{hash(r)}}{2^{\text{hashlen}}} > \lambda$, 返回 $yes$ ;否则返回 $no$ .

需要注意的是由于背书节点身份抽取是一个概率算法, 可能存在一笔交易没有对应的背书节点的情况, 因此需要设置适当的阈值 $\lambda$ 来降低这种情况产生的概率, 以 6.2 节中的实验为例, 背书点候选集中节点数量为 $10, \lambda = 0.4$ 时, 不产生背书节点的概率为:$\Pr = (0.4)_{10} \approx 0.01\%$, 即 10 000 笔交易中可能会有 1 笔交易没有背书节点为其背书. 同时, 如果设置的计时器超时后仍然没有收到背书节点的 proposal_response, 可以选择对 proposal 进行重发, 这样就使得不产生背书节点的概率进一步降低.

(4) 排序节点监听并接收全网所有交易, 并将交易打包成区块 block$<\{\text{tx}\}_m>_{sig}$,$\{\text{tx}\}_m$ 表示区块中包含的 $m$ 个有序交易. 排序节点对 block 签名后将其进行广播;

(5) 提交节点收到 blcok 后首先验证签名检查区块完整性, 之后对读写集进行验证, 并依此更新账本. 当各个提交节点完成以上操作后, 可以视为对该客户端发起的交易达成了共识.

## 3.2 可验证随机函数的设计

### 3.2.1 方案设计

本文依据 IRTF 提出的 VRF 标准草案[34] 设计了基于椭圆曲线的可验证随机函数, 方案设计主要分为三部分:

1) 公私钥对生成函数 $G$

设椭圆曲线的基点为 $O$ , 阶数为 $n$ , 公私钥对生成算法如下:

A1: 选择随机数 $k \in [1, n\text{-}1]$;

A2: 生成一对椭圆曲线密钥, 其中私钥为 k, 公钥为 $Y = kO$ ;

2) 随机数和证明生成函数 $F$

输入: 消息 $m$ , 私钥 $k$

输出: 随机数 $v$ , 证明 $proof$

B1: 选择随机数 $r \in [1, n\text{-}1]$; B2: 使用散列函数 h1 计算 $H = h_1(m)$ , 将消息 m 映射到椭圆曲线上一点 $H$ ; B3: 计算 $rH, rO$ ; B4: 使用函数 $h_2$ 将输入编码成一个整数 s, 并且

$$s = (rH, rO) \tag{3}$$

B5: 计算

$$t = (r - s * k) \bmod n \tag{4}$$

B6: 计算

$$V = kH \tag{5}$$

B7: 使用函数 $h_3$ 将椭圆曲线上的点编码成一个整数, 获得随机数 $value = h_3(V)$ , 证明 $proof$ 为 $(V; t; s)$ .

3) 验证函数 $V$

输入: 消息 $m'$ , 证明 $proof'$

输出: 合法性 ($valid$ or $invalid$)

C1: 使用散列函数 $h_1$ 将消息 $m'$ 映射到椭圆曲线上一点 $H'$ ;

C2: 计算

$$U_1 = t'H' + s'V' \tag{6}$$

$$U_2 = t'O + s'Y \tag{7}$$

C3: 计算

$$s' = \text{h}_2(\text{U}_1, \text{U}_2) \tag{8}$$

C4: 如果 $s = s'$, 则表明随机数有效, 验证通过, 输出 $valid$ ; 否则表明随机数无效, 验证不通过, 输出 $invalid$ .

### 3.2.2 可验证随机函数设计

由 2.2 节可知, VRF 需要满足三个性质: 可验证性、确定性和随机性. 下面将从这三个方面对设计的的可验证随机函数进行分析:

1) 可验证性

如果 $proof$ 未被篡改, 且 $m' = m$ , 则

$$H' = H,\ t' = t,\ s' = s,\ V' = V,\ P = rH \tag{9}$$

$$\begin{aligned} U_1 =& t'H' + s'V' = tH + sV = tH + skH = \\ & (t + sk)H = rH \end{aligned} \tag{10}$$

$$\begin{aligned} U_2 =& O + s'(kO) = tO + s(kO) = \\ & tG + skO = (t + sk)O = rO \end{aligned} \tag{11}$$

$$s' = \text{h}_2(\text{U}_1, \text{U}_2) = \text{s} \tag{12}$$

2) 唯一性和随机性

VRF 得到随机数的过程分为三步, 首先是使用一个从有限域到椭圆曲线上的散列函数 $h_1$ 将消息 $m$ 映射到椭圆曲线上一点 $H$ , 然后使用私钥 $k$ 计算得到 $kH$ , 最后将 $kH$ 编码成一个整数 $value$ 作为最终输出的随机数. 可以看出, 计算 $value$ 是一个确定的过程. 在这个过程中 $k$ 是不变的整数, $h_3$ 一个编码函数对于相同的输入总能保证相同的输出. 因此

*value* 的确定性和随机性取决于 $h_1$ 函数的性质. Dan Boneh[35] 等人提出了一种从有限域映射到椭圆曲线上点的散列函数构造方法, 并且证明了构造方法的正确性. 使用这种方法构造的的 $h_1$ 具有散列函数的性质[36]. 散列函数的值域唯一性保证了对于相同 $m$, $H = h_1(m)$ 也是相同的, 因此 value 也是确定的, 即满足了唯一性. 散列函数均匀分布的性质保证了 $H = h_1(m)$ 与椭圆曲线上随机一点对于敌手来说是不可区分的, 即保证了 $H$ 的随机性, 从而保证了 *value* 的随机性.

## 4 改进方案分析

### 4.1 安全性分析

改进方案的关键点在于将 Hyperledger Fabric 中的背书节点由固定配置改为随机选取. 选取方式为使用可验证随机函数, 根据输出随机数的所在区间确定候选节点是否为背书节点. 这一方案在增加身份抽取随机性、保护节点身份隐私、提供节点身份认证和降低背书被攻击风险方面都提供了更好的安全性.

首先, 由 3.2.2 节中随机性的分析可知, 可验证随机函数的输出具有良好的随机性, 因此确保了在非交互模式下抽取背书节点的随机性, 降低了背书节点的中心化程度; 同时, 在私钥未泄露的情况下, 观察者无法计算背书节点身份抽取的结果, 确保了共识过程启动时背书节点的身份隐私; 此外, 客户端可以使用可验证随机函数的公钥对节点的身份进行验证, 由 3.2.2 中可验证性的分析可知, 这一特性确保了背书节点身份的真实性和可验证性, 防止敌手冒充背书节点.

在原始方案中, 客户端会将接受到的所有背书结果进行对比, 只有全部相同的情况下才会继续交易. 因此, 即使敌手只成功攻击了一个背书节点, 客户端也会因为得到的背书结果不一致而终止交易, 即使将判定策略改为超过一半的背书结果一致则背书结果有效, 这种情况下, 背书节点中恶意节点数超过一半时, 敌手可以完全控制交易的背书结果, 从而影响交易内容正确性. 而在优化方案中, 客户端不再需要得到全部背书节点的背书结果, 使得攻击者影响交易的概率降低. 同样是攻击成功一个候选背书节点, 如果敌手试图影响交易, 则需要背书节点候选集中只产生一个背书节点, 且该节点就是被敌手攻击成功的节点, 其成功概率为:

$$Pr = \frac{1}{n}C_n^1(1-\lambda)\lambda^{n-1} = (1-\lambda)\lambda^{n-1} \tag{13}$$

其中 $\lambda$ 为抽取背书节点时的阈值, n 为背书节点集的数量. 以实验为例, 当 $\mathrm{n} = 10$, $\lambda = 0.4$ 时攻击成功一个背书节点, 攻击者可以影响交易的概率为:

$$pr_i = \begin{cases} C_k^i(1-\lambda)^i(\lambda)^{n-i}, 0 < \mathrm{i} \le \mathrm{k} \\ C_{n-k}^{i-k}(1-\lambda)^i(\lambda)^{n-i}, \mathrm{k} < \mathrm{i} < 2\mathrm{k} \end{cases} \tag{14}$$

所以控制了 $k$ 个候选背书节点的敌手, 成功影响交易的概率为:

$$\mathrm{Pr} = \sum_{i=1}^{n} pr_i \tag{15}$$

在 $n = 10$, $\lambda = 0.4$; $k$, $i \in (0, n]$ 的情况下, 敌手攻击成功的概率变化如图 3 所示:

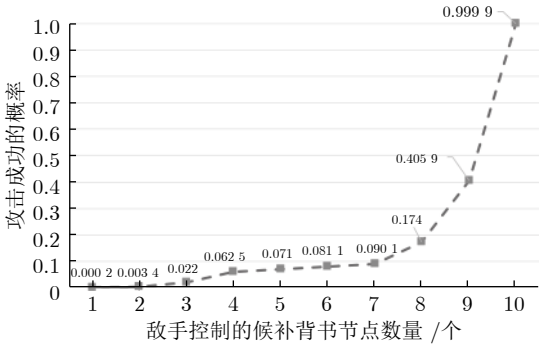


图 3 敌手攻击成功的概率

Fig. 3 Probability of successful enemy attack consensus mechanism

由图 3 可知, 敌手攻击成功的概率会随控制节点的数量的增加而提高, 并且变化趋势呈指数增长. 虽然当敌手控制了全部候选背书节点时, 攻击成功的概率会接近 100%(可能会有不产生背书节点的情况, 因此概率不是 100%, 而是 1-$\lambda^n$), 但是当敌手未完全控制背书候选节点时, 仍然可以保证敌手以一个较低的概率攻击成功, 因此可以通过设置 $\lambda$ 与 $n$, 以满足不同强度的安全需求.

### 4.2 性能分析

#### 4.2.1 交易处理速度

由 2.2 节可知, 在输入 $x$ 相同的情况下, 如果可验证随机函数的私钥不同, 则输出的随机数也不相同. 因此, 对于接受同一笔交易, 不同的节点生成的随机数是不同的, 最终背书节点候选集中部分节点会成为该节点的背书节点, 而剩余节点可能成为其他交易的背书节点. 从整体上看, 可验证随机函数的输出随机性确保了交易被均匀的分发给了所有的候选节点, 这样就减少了每个节点的工作量, 从而提高了交易的并发处理能力.

#### 4.2.2 交易延迟

交易延迟是指一笔交易从发起到确认经历的时

间. 在 Hyperledger Fabric 中, 交易延迟的时间是指从客户端发起提案, 经过背书, 排序, 最终被提交到账本所花费的时间[37]. 本文中影响交易延迟的第一个因素是优化方案增加了抽取背书节点以及验证节点身份的过程, 很明显可验证随机函数的算法效率会对交易的处理产生影响, 进而影响交易延迟, 因此需要通过实验对方案设计的可验证随机函数的算法效率进行测试. 影响交易延迟的另一个因素是优化方案对背书过程进行了调整. 在假设可验证随机函数对交易延迟的影响忽略不计的前提下, 从交易流程上看, 优化方案与原始方案的仅在背书过程中有所不同, 所以从两种方案交易处理速度可以推断出二者的交易延迟差异不大.

#### 4.2.3　通行成本

PBFT 共识算法中所有 $N$ 个节点均以广播的形式发送数据, 因此导致了 $O(N^2)$ 量级的通信成本, Hyperledger Fabric 中客户端与背书节点之间、客户端与排序节点之间是点对点形式的双向通信, 排序节点与提交节点之间是广播形式的单向通信, 因此通信成本为 $O(N)$ 量级. 从整体流程上看, 优化方案与原始方案在各个阶段的通信方式保持一致, 因此通信成本同样为 $O(N)$ 量级, 并不会带来额外的通信开销.

#### 4.2.4　与其他共识机制的比较

除了优化方案与原始方案的比较外, 本文也将对优化后的 Hyperledger Fabric 共识机制与 Algorand、Definity、Ouroboros Praos 这三种基于可验证随机函数的共识机制进行了对比分析.

由于 Hyperledger Fabric 是一个联盟链系统, 而 Algorand、Definity 和 Ouroboros Praos 则是公有链系统, 二者的应用场景、构建方法、共识原理都是不同的. Hyperledger Fabric 的共识机制采取了将应用层信任模型同底层共识协议相解耦的方式, 因此不便于直接从性能效率方面同公有链共识算法进行比较. 所以本文从算法特性的角度对这几种共识机制进行比较分析. 如表 1 所示, 本文对比分析了这几种共识机制中可验证随机函数的作用、共识原理、资源消耗以及容错能力.

表 1　优化方案与其他共识机制的对比

Table 1　Comparison of optimization schemes with other consensus mechanisms

| 共识机制 | VRF的作用 | 共识原理 | 资源消耗 | 容错能力 |
|---|---|---|---|---|
| Algorand | 出块节点的选取 | VRF+PBFT | 低 | $3f+1$ |
| Definity | 出块节点的选取 | VRF+PoS | 较高 | $2f+1$ |
| Ouroboros Praos | 出块节点的选取 | VRF+PoS | 较高 | $2f+1$ |
| 优化方案 | 背书节点的选取 | VRF+背书+排序+验证 | 低 | $F(m,t)$ |

在 Hyperledger Fabric 共识机制的应用层信任模型中背书节点是固定的且身份公开, 这是导致 fabric 背书节点易受攻击的原因. 因此 fabric 需要解决的重点问题是如何可验证地随机选取背书节点. 优化方案正是基于此问题, 使用可验证随机函数选取背书节点, 增加敌手的攻击难度. Algorand、Definity、Ouroboros Praos, 他们需要解决的问题都是如何在全网节点中安全的选取一个出块节点. 具体而言, Algorand 是在全网中选取一个出块节点, 再选取委员会对区块进行多轮投票, 最终达成共识; Dfinity 将全网中节点分为多组委员会, 在当前负责出块的委员会中选取出块节点. Ouroboros 同样是在全网中选取出块节点.

从共识原理上看, 作为公链共识算法的 Algorand、Definity、Ouroboros Praos 并无应用层信任模型的概念, 仍是在系统层共识机制的基础上进行的改进. Algorand 使用的是一个变种的 PBFT 算法 ([22] 中称为 BBA 算法), Definity 底层直接使用 PoS 算法, Ouroboros Praos 则是一个经过了安全证明的 PoS 共识模型. PBFT 和 PoS 共识机制将交易内容和交易顺序同时交给出块节点完成, 这导致 Algorand、Definity、Ouroboros Praos 共识模型是紧耦合的, 并且限制了共识机制的优化. Hyperledger Fabric 优化方案将应用层的信任模型同底层共识算法相解耦, 应用层背书策略和交易合法性验证可根据具体应用进行定制. 而对于交易顺序一致性则交给排序节点完成, 而排序节点本身并不关心交易内容是否合法, 因此可以选择不同算法去完成交易顺序的共识, 比如可以把当前的 kafka 替换为 raft 算法. 这样 Hyperledger Fabric 优化方案可以根据场景的不同, 选择不同的排序算法, 从而具有较高的灵活性.

在资源消耗方面, 基于 PBFT 的 Algorand 不存在 “挖矿” 的过程, 所以不会有大量计算消耗, 但是投票过程中的多轮交互会导致较高的通信成本. 基于 PoS 的 Dfinity 和 Ouroboros Praos 仍然会有 “挖矿” 的过程, 所以还是会导致资源浪费. Hyperledger Fabric 优化方案同样不存在 “挖矿” 的过程,

并且只有排序节点发送区块的过程是全网广播的, 其他过程都是部分节点间的交互, 因此资源消耗很少.

Algorand、Definity、Ouroboros Praos 都是支持拜占庭容错的, 他们的容错能力分别为$n=3f+1$、$n=2f+1$、$n=2f+1$. 其中,$n$ 表示全网节点数,$f$ 代表恶意节点的数量. Hyperledger Fabric 对共识模型进行了解耦, 并且具有不同的节点类型, 所以要从不同的角度去分析其容错能力. 在交易内容一致性的层面并没有提供拜占庭容错的能力, 他只支持对排序节点的一般故障容错, 如数据丢失或延迟. 对于恶意排序节点发送的错误数据, 并不具备容错能力. 而在交易内容合法性的层面, 如果客户端是恶意节点, 其伪造的交易由于不具有正确的背书最终不会验证通过, 从这一点上看, Hyperledger Fabric 对于客户端是支持拜占庭容错的. 但是如果背书节点是恶意节点, 客户端会因为收到不同的数据而终止交易, 所以这一点上看, Hyperledger Fabric 对于背书节点是不支持拜占庭容错的. 优化方案使用可验证随机函数抽取背书节点, 客户端不再接受非背书节点发送的数据, 这样就减少了交易被终止的情况. 所以从整体上而言, 优化方案提升了原始方案的容错能力. 优化方案中背书节点的容错能力取决于背书节点的数量$m$和抽取阈值$\lambda$的设定.

总的来说, 相较于其他基于可验证随机函数的共识机制来说, 优化方案具有更灵活的架构和更低的资源消耗, 并且具有高于 Fabric 原始方案的容错能力.

## 5 实验测试

### 5.1 实验设计

本文依照 Hyperledger Fabric 的架构构建了一个区块链系统. 系统对 Hyperledger Fabric 的交易流程进行了模拟, 特别的是, 系统中可以使用固定和随机两种方式选取背书节点, 即系统可以分别按照原始方案和优化方案进行交易. 本文的实验环境是一台 12 核 CPU, 16G 内存, 2T 机械硬盘的服务器. 实验使用 Docker 技术将节点以容器的形式部署在服务器上. 系统的网络拓扑如图 4 所示, 系统中有四类节点: 客户端、候选背书节点、排序节点和提交节点. 节点通过网络互连, 既可以进行广播通信, 又可以进行点对点通信. 本实验中的节点设置如下: 100 个客户端, 10 个候选背书节点, 1 个排序节点, 100 个提交节点. 客户端之间相互独立, 可以同时发起交易. 当系统使用原始方案时, 所有的候选背书节点都需要对提案进行背书, 使用优化方案时, 会增加背书节点的抽取过程 (抽取的阈值$\lambda$ =

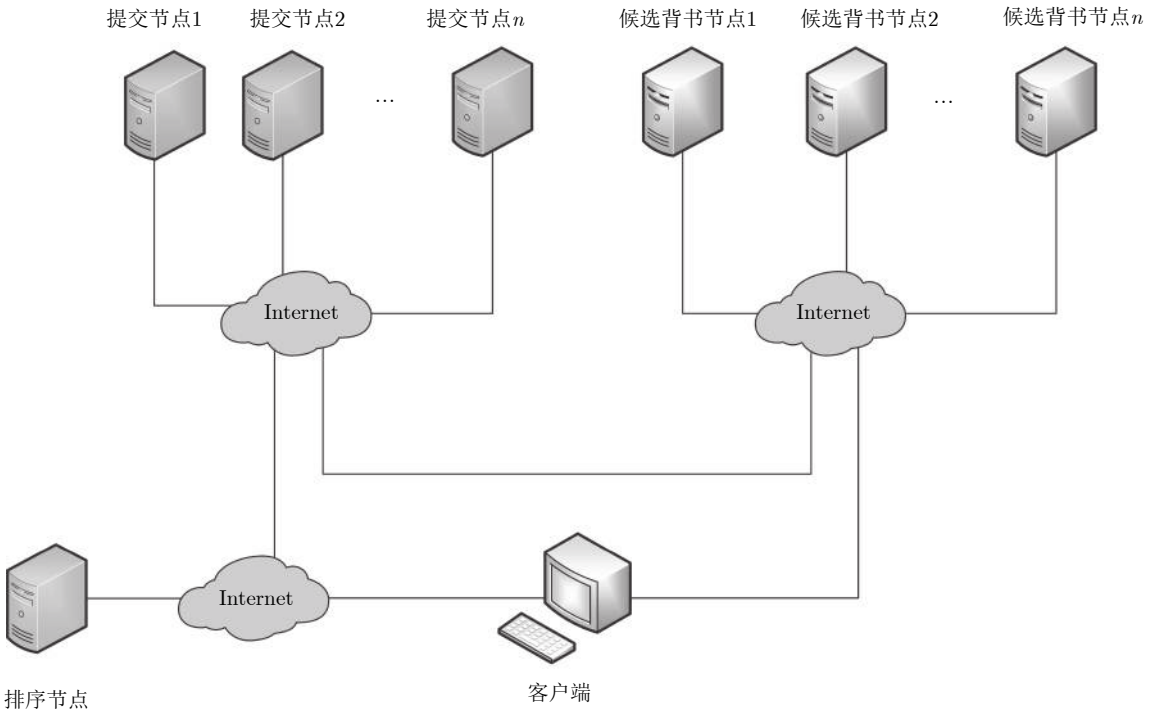


图 4 实验网络拓扑图

Fig. 4 Network topology of experiment

0.4), 客户端也会相应增加验证背书节点身份的过程. 除此之外, 原始方案与优化方案的排序和提交策略保持一致. 实验使用 Golang1.9 实现了原始方案与优化方案的业务逻辑. 其中可验证随机函数的实现基于 Golang 中的 crypto 包, 使用的椭圆曲线为 curve-p256, 使用的 hash 函数为 sha-256. 通过对两种方案的性能测试比较, 本文将在安全性、交易处理速度、交易延迟、通信成本四个方面对优化方案进行分析.

## 5.2　实验结果与分析

### 5.2.1　安全性

在安全性方面, 实验主要对恶意节点对背书过程的攻击风险方面进行了测试对比分析. 原有共识机制存在的主要问题是背书节点中只要存在一个恶意节点就会导致交易背书过程中断, 交易失败. 如 4.1 节中的分析所述, 即使将判定策略改为 “超过一半的背书结果一致则背书结果有效” , 当背书节点中恶意节点数超过一半时, 敌手仍然能够成功控制交易流程. 改进方案则通过使用 VRF 随机抽取背书节点, 增加敌手影响交易流程的难度, 降低了背书过程被攻击的风险.

为此, 测试方案选择在候选节点数 $n=10$ 和恶意节点数 $k=5$ 的条件下进行测试, 结果如表 2 所示. 实验结果表明, 对于原始方案, 背书节点中恶意节点数超过一半时, 敌手会以 100% 的概率成功控制了交易流程. 而使用优化方案后, 只有恶意节点被选为背书节点并且恶意节点数超过选中背书节点数一半以上时, 恶意攻击才能成功控制交易流程, 导致交易失败. 实验中 10 000 次交易情况下敌手攻击成功的概率为 6.86%, 与 4.1 节中的计算值 7.1% 基本接近, 由此可见优化方案中敌手攻击成功的概率明显低于原始方案, 这也证明了优化方案相较于原始方案的安全性提升.

表 2　无背书节点情况发生次数

Table 2　Frequency of nonoccurence of endorsing peer

| | 交易次数 | 敌手成功次数 | 攻击成功概率 |
|---|---|---|---|
| 原始方案 | 1000 | 1000 | 100% |
| 优化方案 | 100000 | 686 | 6.86% |

### 5.2.2　可靠性

由 4.1 节中的分析可知, 由于引入了 VRF 进行节点身份随机抽取时可能出现抽取结果为 0 个背书节点的情况, 影响系统的可靠性. 为此优化方案在客户端使用计时重传机制来解决这一问题, 当客户端在固定时间内未接收到有效的背书响应时会重新发起交易. 实验对这种情况进行了验证, 并且对使用计时重传机制后进行了再次验证, 结果如表 3 所示. 从实验结果可以看出, 当交易次数较多时, 会出现无背书节点的情况, 但是这种情况发生的概率还是很低的. 使用计时重传机制后, 无背书节点情况发生的次数明显降低, 事实上重传后仍无背书节点情况发生的概率为 $\lambda^{2n}$, 基本可以将无背书节点的情况忽略不计.

表 3　无背书节点情况发生次数

Table 3　Frequency of nonoccurence of endorsing peer

| 交易次数 | 是否使用计时重传 | 无背书节点情况发生次数 |
|---|---|---|
| 1000 | 否 | 2 |
| 100000 | 否 | 17 |
| 1000 | 是 | 0 |
| 100000 | 是 | 0 |

### 5.2.3　交易处理速度

与 Hyperledger Fabric 原始的共识机制相比, 本方案通过将交易的背书分摊给多个节点, 降低背书节点需要背书的交易数量. 考虑一种特殊情况: 假设两笔交易对应的的背书节点之间并不存在交集, 则这两笔交易可以同时进行背书, 可以视为对交易的并行处理. 不出现这种情况, 从整体而言, 每个背书节点不需要依次处理所有的交易, 而是只是其中一部分, 一批交易所需的背书时间仅取决于分配交易数最多的背书节点, 这样改进方案的背书时间相较于原有方案有所缩短, 从而提高交易的处理速度.

实验分别对原有的 Hyperledger Fabric 共识机制和优化后的共识机制进行了仿真, 通过模拟构建两种方案中的交易系统, 将两种方案的交易处理速度进行了对比. 其中系统参数设置为: 背书节点候选集数量为 10, 阈值 $\lambda$ 为 0.4. 实验结果如图 5 所示.

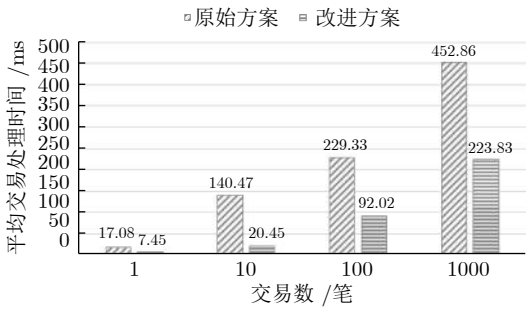


图 5　原有方案与优化方案交易时间对比

Fig. 5　The comparison of transaction time between original scheme and optimized scheme

从图 5 中的实验结果可以看出, 优化后的方案在交易处理速度方面相较于原始方案有了明显提高. 特别是当交易数较大时, 虽然随着交易数量的增加二者的交易时间都呈现上升趋势, 但是优化方案仍能比原始方案的时间少约 50%, 优化方案的交易处理速度提升明显.

### 5.2.4 交易延迟

本文针对 5.2.2 节中提到的影响交易延迟的两个因素, 分别进行了测试.

第一个是对本文设计的可验证随机函数的性能测试. 本文实现的可验证随机函数主要分为三部分, 生成密钥, 生成随机数和证明, 验证随机数和证明. 实验对方案设计的可验证随机函数进行了多次测试, 各部分算法的运行时间如表 4 所示:

表 4 可验证随机函数各部分算法运行时间

Table 4 Running time of each part of the vrf algorithm

| 算法 | 次数 | 总时间(ms) | 平均时间(ms) |
|---|---|---|---|
| 生成密钥 | 10000 | 2878.3546 | 0.2878 |
| 生成随机数和证明 | 10000 | 10395.3927 | 1.0395 |
| 验证随机数和证明 | 10000 | 12874.6190 | 1.2875 |

由实验结果可知, 可验证随机函数各部分算法的运行时间均在毫秒级, 并且相较于 5.2.2 中处理交易所需的总时间, 基本可以忽略其对交易处理速度的影响. 在此基础上, 本文对两种方案的交易延迟进行了测试, 实验结果如图 6 所示. 原始方案和优化方案的交易延迟时间都会随着交易数量的增加而增长, 这是由于交易数量超过处理上限后, 需要排队等待节点处理. 可以看出, 在相同交易数量的情况下, 优化方案的交易延迟明显低于原始方案.

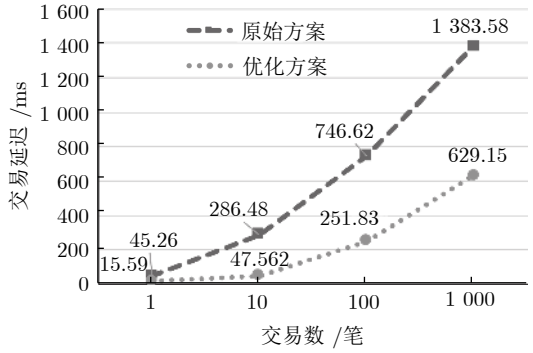


图 6 原有方案与优化方案交易延迟对比

Fig. 6 The comparison of transaction delay between original scheme and optimized scheme

### 5.2.5 通信成本

本文将一笔交易从发出到确认这段时间内, 区块链网络产生的所有报文的报文数量和报文大小作为这笔交易的通信成本. 实验分别测试了原有方案与优化方案下, 一笔交易和多笔交易时的通信成本, 实验结果如图 7 所示. 原始方案中, 每个背书节点都会将背书信息发送给客户端, 而在优化方案中, 未被选中的候选背书节点将不再执行此操作, 因此在本实验中, 优化方案的报文数量和总报文大小均小于原始方案, 这表明使用优化方案可以在一定程度上降低通信成本.

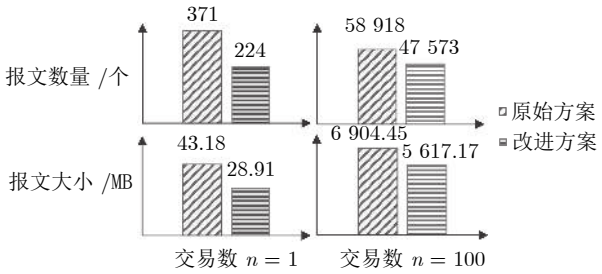


图 7 原有方案与优化方案交易成本对比

Fig. 7 The comparison of transaction delay between original plan and optimization scheme

## 6 总结

本文针对 Hyperledger fabric 原有共识机制存在的问题, 提出了一种基于可验证随机函数的背书节点随机选取的优化方案, 并对优化后的共识机制从安全性、性能等方面进行了分析比对和实验测试. 实验结果表明, 优化后的共识机制在保证通信成本基本不变的前提下, 具有更高的安全性、更低的交易延迟和更快的交易处理速度.

## References

1 Nakamoto S. Bitcoin: a peer-to-peer electronic cash system[Online], available: https://bitcoin.org/bitcoin.pdf, December 17, 2019

2 Liu Ao-Di, Du Xue-Hui, Wang Na, Li Shao-Zhuo. Research progress of blockchain technology and its application in information security. *Journal of Software*, 2018, **29**(7): 2092−2115
(刘敖迪, 杜学绘, 王娜, 李少卓. 区块链技术及其在信息安全领域的研究进展. 软件学报, 2018, **29**(7): 2092−2115)

3 Han Xuan, Yuan Yong, Wang Fei-Yue. Security problems on blockchain: the state of the art and future trends. *Acta Automatica Sinica*, 2019, **45**(1): 206−225
(韩璇, 袁勇, 王飞跃. 区块链安全问题:研究现状与展望. 自动化学报, 2019, **45**(1): 206−225)

4 Nguyen G T, Kim K. A survey about consensus algorithms used in blockchain. *Journal of Information Processing Systems*, 2018, **14**(1): 101−128

5 Elli A, Artem B, Vita B, Christian C, Konstantinos C, Angelo D. Hyperledger Fabric: a distributed operating system for permissioned blockchains. In: Proceedings of the Thirteenth

EuroSys Conference. Porto, Portugal: ACM, 2018. 30

6 Vukolić M. Rethinking permissioned blockchains. In: Proceedings of the ACM Workshop on Blockchain, Cryptocurrencies and Contracts. Abu Dhabi, United Arab Emirates: ACM, 2017. 3–7

7 Bessani A, Sousa J, Vukolić M. A byzantine fault-tolerant ordering service for the Hyperledger Fabric blockchain platform. In: Proceedings of The Workshop on Scalable & Resilient Infrastructures for Distributed Ledgers. Luxembourg City, Luxembourg: IEEE, 2018. 51–58

8 Yuan Yong, Wang Fei-Yue. Blockchain: The state of the art and future trends. *Acta Automatica Sinica*, 2016, **42**(4): 481–494
(袁勇, 王飞跃. 区块链技术发展现状与展望. 自动化学报, 2016, **42**(4): 481–494)

9 Bach L M, Mihaljevic B, Zagar M. Comparative analysis of blockchain consensus algorithms In: Proceedings of 2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO). Opatija, Croatia: IEEE, 2018. 1545–1550

10 Lamport L, Shostak R, Pease M. The byzantine generals problem. *Acm Transactions on Programming Languages & Systems*, 1982, **4**(3): 382–401

11 Lamport L. Paxos made simple. *ACM Sigact News*, 2001, **32**(4): 18–25

12 Ongaro D, Ousterhout J. In search of an understandable consensus algorithm. In: Proceedings of Usenix Conference on Usenix Technical Conference. Philadelphia, PA, USA: ACM, 2014. 305–319

13 Fan Jie, Yi Le-Tian, Shu Ji-Wu. Research on the technologies of byzantine system. *Journal of Software*, 2013, **24**(6): 1346–1360
(范捷, 易乐天, 舒继武. 拜占庭系统技术研究综述. 软件学报, 2013, **24**(6): 1346–1360)

14 Castro M, Liskov B. Practical byzantine fault tolerance. In: Proceedings of the third Symposium on Operating Systems Design and Implementation. New Orleans, Louisiana, USA: OSDI, 1999.173–186

15 Yuan Yong, Ni Xiao-Chun, Zeng Shuai, Wang Fei-Yue. Blockchain consensus algorithms: the state of the art and future trends. *Acta Automatica Sinica*, 2018, **44**(11): 2011–2022
(袁勇, 倪晓春, 曾帅, 王飞跃. 区块链共识算法的发展现状与展望. 自动化学报, 2018, **44**(11): 2011–2022)

16 Bentov I, Lee C, Mizrahi A, Rosenfeld M. Proof of activity: extending bitcoin's proof of work via proof of stake.[Online], available: http://eprint.iacr.org/2014/452, December 16, 2019

17 S. King and S. Nadal. PPCoin: peer-to-peer crypto-currency with proofof-stake(whitepaper), available: https://bitcoin.peryaudo.org/vendor/peercoin-paper.pdf, December 17, 2019

18 Li W, Andreina S, Bohli J M, Karame G. Securing proof-of-stake blockchain protocols. In: Proceedings of Cryptocurrencies and Blockchain Technology.Barcelona, Spain: Springer, 2017. 297–315

19 Bitshares. Delegated proof of stake[Online], available: https://docs.bitshares.org/en/master/technology/dpos.html, December 17, 2019

20 Silvio M, Salil V, Michael R. Verifiable random functions. In: 40th Annual Symposium on Foundations of Computer Science. New York, USA : IEEE, 1999. 120–130

21 Abdalla M, Catalano D, Fiore D. Verifiable random functions from identity-based key encapsulation. In: Proceedings of Annual International Conference on the Theory and Applications of Cryptographic Techniques. Cologne, Germany: Springer, 2009.554–571

22 Gilad Y, Hemo R, Micali S, Vlachos G, Zeldovich K. Algorand: scaling byzantine agreements for cryptocurrencies. In: Proceedings of the 26th Symposium on Operating Systems Principles. Shanghai, China: ACM, 2017.51–68

23 Hanke T, Movahedi M, Williams D. Dfinity technology overview series, consensus system. arXiv preprint arXiv: 2018, 1805.04548

24 Boneh D, Boyen X. Short signatures without random oracles. In: Proceedings of International conference on the theory and applications of cryptographic techniques. Interlaken, Switzerland: Springer, 2004. 56–73

25 Kiayias A, Russell A, David B, Oliynykov R. Ouroboros: a provably secure proof-of-stake blockchain protocol. In: Proceedings of Annual International Cryptology Conference. Paris, France: Springer, 2017.357–388

26 David B, Gaži P, Kiayias A, Russell A. Ouroboros Praos: an adaptively-secure, semi-synchronous proof-of-stake blockchain. In: Proceedings of Annual International Conference on the Theory and Applications of Cryptographic Techniques. Tel Aviv, Israel: Springer, 2018. 66–98.

27 Hearn M. Corda: A distributed ledger[Online], available: https://docs.corda.net/releases/release-V3.1/-\\static/corda-technical-whitepaper.pdf, December 17, 2019

28 Yamashita K, Nomura Y, Zhou E, Pi B, Jun S. Potential risks of Hyperledger Fabric smart contracts. In: Proceedings of 2019 IEEE International Workshop on Blockchain Oriented Software Engineering (IWBOSE). Hangzhou, China: IEEE, 2019. 1–10

29 Manevich Y, Barger A, Tock Y. Endorsement in Hyperledger Fabric via service discovery. *IBM Journal of Research and Development*, 2019, **63**(2): 1–9

30 Brandenburger M, Cachin C, Kapitza R, Sorniotti A. blockchain and trusted computing: problems, pitfalls, and a solution for Hyperledger Fabric. *arXiv preprint arXiv*, 2018, 1805.08541

31 Sukhwani H, Wang N, Trivedi K S, Rindos A. performance modeling of Hyperledger Fabric (permissioned blockchain network). In: Proceedings of 2018 IEEE 17th International Symposium on Network Computing and Applications (NCA). Cambridge, MA, USA: IEEE, 2018.1–8

32 Sukhwani H, Martínez J M, Chang X, Trivedi K, Rindos A. Performance modeling of pbft consensus process for permissioned blockchain network (hyperledger fabric).In: Proceedings of 2017 IEEE 36th Symposium on Reliable Distributed Systems (SRDS). Hong Kong, China: IEEE, 2017. 253–255

33 Baliga A, Solanki N, Verekar S, Pednekar A, Kamat P, Chatterjee S. Performance characterization of Hyperledger Fabric. In: Proceedings of 2018 Crypto Valley Conference on Blockchain Technology (CVCBT). Zug, Switzerland: IEEE, 2018. 65–74.

34 Goldberg S, Reyzin L, Papadopoulos D, Vcelak J. Verifiable random functions (VRFs)[Online], available: https://datatracker.ietf.org/doc/draft-irtf-cfrg-vrf/, February 08, 2019

35 Boneh D, Lynn B, Shacham H. Short signatures from the Weil pairing. In: Proceedings of International Conference on the Theory and Application of Cryptology and Information Security. Gold Coast, Australia: Springer, 2001. 514–532

36 Carter J L, Wegman M N. Universal classes of hash functions. *Journal of computer and system sciences*, 1979, **18**(2): 143–154

37 Nguyen T S L, Jourjon G, Potop-Butucaru M, Thai K L. Im-

pact of network delays on Hyperledger Fabric. *arXiv preprint arXiv*: 2019. 1903.08856

**孟吴同** 北京交通大学硕士研究生. 2017 年获得河北大学网络工程学士学位. 主要研究方向为区块链.
E-mail: mengwt@bjtu.edu.cn
(**MENG Wu-Tong** Master student at School of Computer and Information Technology, Beijing Jiaotong University. He received his bachelor degree of network engineering from Hebei University in 2017. His research interest covers blockchain.)

**张大伟** 北京交通大学计算机与信息技术学院副教授. 2004 年于北京航空航天大学获得通信与信息系统专业博士学位. 主要研究方向为区块链, 安全协议, 可信计算. 本文通信作者.
E-mail: dwzhang@bjtu.edu.cn
(**ZHANG Da-Wei** Associate professor at School of Computer and Information Technology, Beijing Jiaotong University. He received his Ph.D. degree of communication and information system from Beihang University in 2004. His research interest covers blockchain, security protocol and trusted computing.Corresponding author of this paper.)