融合可验证随机函数的改进 Raft 共识算法

杨　州，周创明
（空军工程大学防空反导学院，西安，710051）

摘要　传统的 Raft 算法容易因受到恶意攻击而导致效率低下。针对 Raft 算法在执行过程中可能被攻击者探测到 Leader 身份进而发动攻击的问题，利用可验证随机函数的随机性和非交互式特性，改进了原有 Raft 算法，隐藏了选主过程中 Leader 的身份的同时避免了原算法的多轮选举问题。提出的改进 Raft 算法的详细设计了可验证随机函数的实现并将其融入到选主过程中，并且分析了改进方案的系统安全性和稳定性，对比了不同共识算法在各需求场景下的表现。仿真试验表明：改进后的算法在选主过程中隐藏了被选中节点的身份，在保证系统稳定性的前提下提高了系统安全性，并且在系统节点越来越多时提高选主的时间效率的表现更好。

关键词　Raft；可验证随机函数；区块链；分布式系统一致性；共识算法
DOI　10.3969/j.issn.2097-1915.2023.06.014
中图分类号　TP393　　**文献标志码**　A　　**文章编号**　2097-1915(2023)06-0104-08

An Improved Raft Consensus Algorithm for Integrating ECVRF

YANG Zhou，ZHOU Chuangming
（Air and Missile Defense School，Air Force Engineering University，Xi'an 710051，China）

Abstract　Raft consensus algorithm is a reliable consistency algorithm implementation in distributed systems，but the traditional Raft algorithm is easy to make efficiency low because of malicious attacks. Aimed at the problem that the leader identity in Raft algorithm may be detected by attackers in the process of execution and then attacked，the randomness and non-interactive characteristics of verifiable random functions are utilized for improving the original Raft algorithm，and hiding the leader identity in the leader election process，avoiding the multi-round election problem in the original algorithm. This paper designs an implementation of verifiable random functions in detail，and incorporates it into the leader election process. The paper also analyzes the system security and stability of the improved scheme，compares the performance of different consensus algorithms in various requirement scenarios. The simulation experiments show that the improved algorithm hides the identity of the elected node in the leader election process，improves the system security on the premise of ensuring system stability，and also performs still more in improving the time efficiency of leader election with the increase of the number in system nodes.

Key words　Raft；verifiable random function；block chain；distributed system consistency；consensus algorithm

收稿日期：2023-05-19
基金项目：国家自然科学基金（61806219，61703426，61876189）；陕西省自然科学基金（2021JM-226）；陕西省高校科协青年人才托举计划（20190108，20220106）；陕西省创新能力支撑计划（2020KJXX-065）
作者简介：杨　州（1997－），男，江苏连云港人，硕士生，研究方向为区块链技术。E-mail：yz_afeu@163.com
通信作者：周创明（1967－），男，湖南益阳人，副教授，博士，研究方向为智能信息处理、信息安全。E-mail：179820572@qq.com

引用格式：*杨州，周创明. 融合可验证随机函数的改进 Raft 共识算法[J]. 空军工程大学学报，2023，24(6)：104-111. YANG Zhou，ZHOU Chuangming. An Improved Raft Consensus Algorithm for Integrating ECVRF[J]. Journal of Air Force Engineering University，2023，24(6)：104-111.*

2008 年中本聪发表《Bitcoin：A Peer-to-Peer Electronic Cash System》[1]标志着以比特币为代表应用的区块链技术的诞生。区块链是密码学、数学、计算机科学等多学科的交叉研究领域，其具有的组织体系去中心化、链上数据不可篡改等特性引起了学界和产业界的广泛关注和研究，被认为是实现互联网价值的关键技术。

共识算法是区块链底层的核心技术，用于在分布式系统中使所有节点达成共识。从管理权限和数据读写权限角度区块链可分为公有链、联盟链和私有链。公有链是最符合区块链最初设想的概念实现，公有链系统中所有节点都可以参与到整个共识过程，是完全去中心化的。目前公有链的共识算法主要分为以算力竞争出块权的工作量证明（proof of work，POW）[1]、以已持有的数字货币量和时间竞争出块权的权益证明（Proof of Stake，POS）[2]和以自身拥有的权益来投票选举代表生产区块的委托权益证明（DPOS）[3]。然而正是由于公有链的去中心化特性，使其在隐私保护和政策监管等方面存在天然缺陷，并不适用于大部分的社会应用场景。与公有链对应的是私有链，它是完全封闭的、中心化的，由某个公司或组织创建，记录其内部的数据信息，共识和数据读取权限掌握在少数内部节点手里。私有链的这些特性虽然保护了内部数据的安全且在共识效率上也远远高于公有链和联盟链，但是其封闭性也决定了它难以适用于大部分的社会应用场景[4]。联盟链介于公有链和私有链之间，即“半开放、半去中心化”，由若干公司或组织联合创建，只限于联盟成员参与[5]。数据读取权限和共识机制等均需遵循联盟链中的规定进行运作，并按照准入规则接纳外部节点的加入。联盟链的半开放、半去中心化、规则清晰等特性，使其具有易于政策监管、权限可控等优势，相比公有链和私有链，更适用于大部分的社会应用场景。目前联盟链流行的共识算法分为两类：一类是非拜占庭容错共识，这类共识机制不考虑恶意节点的攻击行为，但可以容忍系统崩溃、节点网络故障等情况，如 Zookeeper[6]、Paxos[7]、Raft[8]等。另一类是拜占庭容错[9]共识，这类共识机制会考虑恶意节点攻击、故意回复虚假消息、串通作恶等情况，如 Pbft 共识算法[10]。选用哪类共识机制要根据具体的应用场景，综合考虑系统的复杂性与安全性进行折中设计。

本文分析了联盟链中常用的非拜占庭容错类的共识算法——Raft 共识算法中存在的安全问题和多轮选举问题，通过引入可验证随机函数（verifiable random function，VRF）[11]对原算法进行改进，提高了 RAFT 算法在 Leader 选举过程中的安全性和效率。

1　Raft 算法与可验证随机函数

1.1　Raft 算法

分布式系统的一个关键问题是如何保证系统中各节点的存储数据的一致性[12]。经典的解决方案是利用状态机保证数据复制的一致性。Paxos 算法是 Leslie Lamport 于 1998 年提出的一种基于消息传递的分布式一致性算法。Paxos 算法通过消息传递机制来就分布式网络中的某个提议达成一致。算法通过一系列日志项的复制达成整个网络的一致性，具有很快的执行效率，但是该算法对于学习者来说非常难理解，且没有适用的搭建实用系统的基础，因此难以在实践中落地。为了解决这些问题，Diego Ongaro 和 John Ousterhout 于 2014 年提出了 Raft 算法。提出 Raft 算法就是为了简化 Paxos 算法，同时 Raft 算法在一致性、效率、安全等方面与 Paxos 算法相比基本一致。Raft 系统中的角色有领导人（Leader）、跟随者（Follower）和候选人（Candidate）3 种。与 Paxos 算法相比，Raft 算法增加了对日志更新方式和选举过程的限制。首先更新日志的操作必须是连续的；其次是在 Leader 选举时只有拥有最新、最全日志的节点才有资格被选为 Leader[13]。

Raft 算法达成系统一致性的主要工作过程如下：

1）Leader 选举。节点在当选为 Leader 后会按照固定的时间间隔向其他 Follower 发送心跳日志（空的附加日志）以标识自己在系统中仍正常工作。一旦有 Follower 在设定的时间内没有接收到心跳日志，即可认为当前任期的 Leader 发生了故障，Follower 会转变为 Candidate，进行下一任期的 Leader 选举。

2）日志复制。成功选举出 Leader 后，客户端将请求发送到系统中的任一节点。若 Leader 接收到请求，则将该日志复制到系统中的所有 Follower 节点；若 Follower 接收到请求，则会将该请求转发给 Leader，确保始终只由 Leader 与客户端交互。在日志复制过程中，如果 Follower 节点崩溃、运行缓慢或网络丢包，Leader 会不断地重试直到所有的 Follower 最终都存储了最全、最新的日志条目。最后由 Leader 将执行结果返回给客户端。

3）安全性。Raft 算法的安全性主要体现在节点状态机安全方面。在系统中如果有任何一个节点已经应用了一个确定的日志条目到它的状态机中，那么其他节点就不能在同一个日志索引位置应用一

个不同的指令。

Raft 算法中节点 3 种角色状态的转换如图 1 所示。

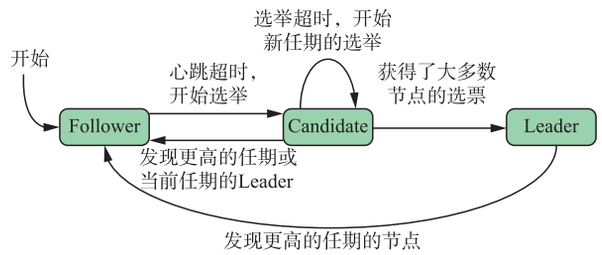

图 1 Raft 算法中角色转换示意图

但是这样的执行过程也存在问题：①可能会导致在选举中产生投票分歧，即多个 Candidate 获得了相同且均是最高的票数，这样系统就不会选出唯一的 Leader，只能宣告本次选举失败，然后将任期加 1，重新发起选举。Raft 算法为了解决这个问题采用了随机的选举超时时间策略，具体为 Candidate 会在固定的时间区间内随机选择选举的超时时间，这样可以保证通常只会有 1 个 Candidate 发生选举超时，进而向其他节点请求选票，且保证在其他节点发生超时前赢得选举，并发送心跳包。然而随着系统的节点数增多，各节点随机的超时时间重复的几率也会增加，仍然会存在投票分歧问题，导致要进行多轮选举才会有 Leader 当选。这会使选举过程的通信量和时间增加，影响系统运行效率。②Raft 算法在选举出 Leader 后会不断向其他节点发送心跳日志，频繁的节点间通信可能会使 Leader 节点的身份暴露给恶意节点，招致恶意攻击，如分布式拒绝服务攻击(distributed denial of service，DDoS)[14]。

为了解决这些问题，学者们进行了许多研究。荣宝俊等[15]提出了 FL_Raft 模型，基于联邦学习技术筛选出高性能节点组并根据权益值计算出唯一不变的领导者节点，提高了集群的效率和稳定性；邹贤等[16]提出基于信用评分改进 Raft 共识机制，利用信用评分的方式替代节点投票避免选举纠纷，提高 Raft 选主的速度和对恶意节点的适应能力；Huang 等[17]在私有链上提出了一个 Raft 网络分裂的模型，并利用此模型预测网络分裂的时间和概率，以此优化 Raft 共识算法中的参数；Rong 等[18]基于联邦重构技术，对 Raft 节点特征数据集进行训练、更新和评估，将性能较好的节点构建为 Leader 候选委员会，提高选举质量和速度，还设计了半异步缓冲机制和抵御恶意节点攻击的策略，解决了联合聚合的不一致性和安全性问题；Du 等[19]提出了一种多策略 Leader 选举机制，允许节点在认为 Leader 存在性能问题时主动触发新一轮的 Leader 选举，并在指定的优先级队列中替换这个 Leader，该机制可以在不增加选举时间开销的情况下提高系统的吞吐量。不过，以上研究对 Raft 算法的安全性方面关注较少，本文将进一步讨论如何借助可验证随机函数提高 RAFT 算法的安全性和运行效率。

1.2 可验证随机函数

1999 年 Silvio Micali 等[10](verifiable random functions，VRF)，提出可验证随机函数。VRF 是基于公私钥密码学和零知识证明技术所构建的具有可验证功能的，非交互式的伪随机函数。对于一个特定输入信息 m 和证明者的私钥(secret key，SK)，VRF 会生成一个随机数 r 以及 r 的证明 π，验证者可以通过 m、r、π 和证明者持有的公钥(public key，PK)来验证 r 是否由 PK 的持有者根据 m 所产生。在这个过程中不用暴露证明者的私钥，同时由零知识证明技术保证验证者除了知道“r 是由 PK 的持有者根据 m 所产生的”这个信息外，得不到其他任何有价值的信息，保障了证明者和输入 m 的安全。VRF 的执行流程如下：

1)用户首先通过非对称加密算法申请一组公私钥对(PK，SK)，PK 为公钥，SK 为私钥。

2)传入 m 和 SK，VRF_Proof 函数会生成一个 r 和一个可对 r 进行零知识证明的 π。

$$\pi=\mathrm{VRF\_Proof}(m,SK) \tag{1}$$

3)VRF_Verify 函数通过 m、π 和 PK 验证该证明 π 是否由原始输入 m 和证明人的 PK 所产生的，是返回 true，否返回 false。

$$\mathrm{VRF\_Verify}(m,\pi,PK)=\mathrm{true} \tag{2}$$

VRF 具备以下 3 个性质：

1)可验证性。验证者通过 π 可以验证出 r 是由 m 和证明者 SK 所生成的；

2)随机性。在不确定 π 的情况下，r 和其他任意一个随机数对于攻击者来说是不可区分的；

3)确定性。VRF 在 π 和 SK 不变的情况下，输出的 r 也是不变的。

可验证随机函数在提出后，多用于数字签名[20]等领域。随着可验证随机函数在各组件上的关键技术突破和区块链技术的深入研究与发展，越来越多的新链在设计共识算法时都会引入可验证随机函数来随机抽取节点，从而达到隐藏关键节点的目的，降低被恶意节点攻击风险。

2 试验方案设计

本节按照实际中的系统应用详细设计了 VRF 方案和融合了 VRF 的改进 Raft 算法流程。

2.1 VRF 方案设计

本试验的 VRF 方案是按照 IETF 拟定的 VRF

标准草案[21]设计的。草案的实现方案有 2 套体系，一套是基于 RSA 算法的(RSA full domain Hash VRF，RSA-FDH-VRF)，另一套是基于椭圆曲线的(elliptic curve verifiable random function，ECVRF)。2 套实现体系都能满足可信唯一、可信防碰撞和完全伪随机特性。不过基于 RSA 实现的 VRF 要起到足够的安全性，需要 RSA 的密钥长度比较长，这点限制了其在很多场景下的应用。而椭圆曲线加密因其密钥长度小、安全性能高，逐渐成为首选的非对称加密实现方案。本实验的 ECVRF 方案设计主要包括以下 3 个函数部分：

1)生成一组公私钥对。设椭圆曲线在有限域 F 上，阶数为素数 n，取椭圆曲线上一点 O，公私钥对生成算法如下：

步骤 1　选择一个随机数 x，$0<x<n$；

步骤 2　生成一对非对称密钥，其中私钥即为 x，公钥为 $Y=xO$（这里的乘法不是常用的系数与坐标乘法，而是椭圆曲线上对应的乘法规则）

2)生成随机数及其证明。

输入　私钥 x，信息 m，公共混淆值 salt_value(一个 8 位字符串，在输入内容的任意位置插入用于加强安全性，如果输入的密码套件已经实现了该部分则可不使用此值)。

输出　随机数 r，证明 π。

步骤 1　使用 $f_1()$将信息 m 映射到椭圆曲线上一点：

$$H=f_1(\text{salt\_value},m) \tag{3}$$

步骤 2　使用 $f_2()$函数将 H 转换为字符串：

$$h=f_2(H) \tag{4}$$

步骤 3　选择一个随机数 x，$0<x<n$；$\beta=xH$

步骤 4　随机数 r 由函数 nonce_generation(x，h)生成；

步骤 5　使用 $f_3()$函数得到整数

$$c=f_3(Y,H,\beta,rO,rH) \tag{5}$$

步骤 6　计算：

$$s=r-cx \bmod n \tag{6}$$

步骤 7　使用拼接函数将($f_2(\beta)$，$f_4(c,c\text{Len})$，$f_4(s,n\text{Len})$)拼接得到证明 π(nLen 取值为满足 $2^{(8n\text{Len})}>n$ 的最小整数，cLen 的取值是 nLen/2 或者接近它)。

3)验证随机数及其证明。

输入　公钥 Y，信息 m，证明 π，公共混淆值 salt_value；

输出　正确性(true or false)。

步骤 1　使用 $f_1()$将信息 m 映射到椭圆曲线上一点：

$$H=f_1(\text{salt\_value},m) \tag{7}$$

步骤 2　计算：

$$u=cO+sY \tag{8}$$

$$v=c\beta+sH \tag{9}$$

步骤 3　使用 $f_3()$函数得到整数：

$$c'=f_3(Y,H,\beta,u,v) \tag{10}$$

步骤 4　如果 $c=c'$，则通过验证，输出 true；否则验证不通过，输出 false。

表 1　VRF 方案设计中各公共函数及其作用

函数	功能
$f_1()$	将字符串映射到椭圆曲线上一点
$f_2()$	将椭圆曲线上一点转换为字符串
$f_3()$	将输入编码为一个整数
$f_4()$	将整形转换为字符串类型

2.2　改进 Raft 算法设计

原 Raft 算法的问题一方面在于 Leader 选举完成后要不断向其他节点发送心跳日志，这样可能会导致 Leader 身份暴露给攻击者，另一方面节点数量增多会增加产生投票分歧的概率而影响系统效率。针对以上问题，本实验利用可验证随机函数生成的随机数以及工作过程中的非交互式特性来改进原算法。改进算法的流程设计如下：

步骤 1　当前系统利用信息 m(可以为当前时间与任期的组合)计算得到一个随机整数值 Z，Z 在本轮选举中对系统的所有节点都是可见且唯一的。

步骤 2　选举开始，系统中的节点按照随机顺序利用信息 m_2 计算得到随机整数 Z'，比较 Z 与 Z' 的大小。若 $Z'<Z$ 满足条件，当前节点继续执行步骤 3；若 $Z'>Z$，不满足条件，由下一节点重新执行步骤 2。

步骤 3　满足条件的当前节点即是本轮选中的 Leader。Leader 首先将 Z' 与自己的私钥输入可验证随机函数的生成随机数与证明模块函数，得到 r 和 π，再将需要给其他节点验证的必要参数 Y，m_2，π 和本轮要更新的日志项一起广播给系统中所有 Follower 节点。

步骤 4　Follower 在收到 Leader 发来的信息后，将 m_2、π 与 Leader 的公钥 Y 输入可验证随机函数的验证随机数与证明模块函数得到 c'。如果 $c=c'$，则随机数有效，验证通过，则将信息中的日志项同步到本地数据库中，并且向 Leader 反馈已接受信息；否则随机数无效，验证不通过，不接收此日志项，并且向 Leader 反馈错误信息。

步骤 5　Leader 在收到大部分 Follower 的接受信息后即可向客户端反馈日志写入成功，本轮日志复制完成，下一轮从步骤 1 开始重复执行所有

流程。

改进方案流程图如图 2 所示。

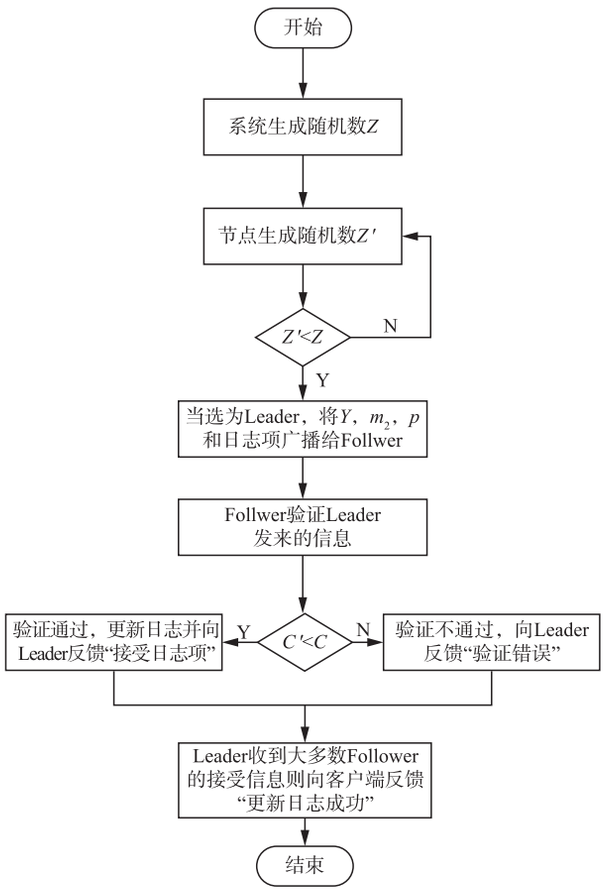

图 2 改进方案设计流程图

3 改进方案分析

3.1 系统安全性分析

Raft 算法通过在 Leader 选举时增加一些限制来增强安全性，并且给出了领导人完整特性(leader completeness property)[8]的简要证明，保证了每一个状态机会按照相同的顺序执行相同的指令。前文设计的改进算法利用可验证随机函数产生的随机数来确定哪个节点被选为 Leader，同时由于可验证随机函数的非交互性，Follower 在收到复制指令前并不会知道当前系统的 Leader 具体是哪个节点。当 Follower 收到 Leader 的日志复制指令后则会根据指令来源节点给出的 VRF 证明通过验证函数得到结果，由算法保证得到验证结果就立即判断是否接受最新日志。当结果为 true 则将最新日志同步到本地数据库中并且向 Leader 反馈已同步信息；结果为 false 则不接收此日志项并且向 Leader 反馈未通过验证信息。Leader 在收到大多数节点的同步信息后给客户端返回写入成功的信息，此轮日志复制结束。通过上述策略可以在日志复制操作前隐藏 Leader 节点的身份，增强系统的安全性。

3.2 系统稳定性分析

Raft 算法在系统发生网络分割(network partiton)时会造成脑裂(brain split)问题。系统中路由节点或者某些节点在网络故障的情况下，会将系统分割成若干个彼此独立的集群，而在选举超时后拥有多数节点的集群会产生新的 Leader，导致系统中存在多位 Leader 的情况，出现脑裂问题，如图 3 所示。脑裂问题可能会导致数据的覆盖丢失。Raft 算法通过任期(Term)属性来解决这个问题，Term 初始设为 1，此后每次选举出新 Leader 则 Term 加 1。同时规定每轮任期只能有一个 Leader。这样在发生网络分割情况下，节点多的集群产生新的 Leader 的任期必定比其他集群 Leader 的任期大，而且其他集群的 Leader 即使收到客户端的写请求并将日志复制到其他节点后也会因为得不到大多数节点的回应而无法告知客户端写入成功。在这样的策略下，当网络被修复后，其他集群的节点包括 Leader 节点会发现系统中存在任期更大的 Leader，则都会转为 Follower 并同步当前系统的最新最全日志数据。本文的试验保留了 Raft 节点的任期属性以及在选举 Leader 时的限制条件，所以改进后的试验方案能保证在出现网络分割情况下系统仍然稳定。

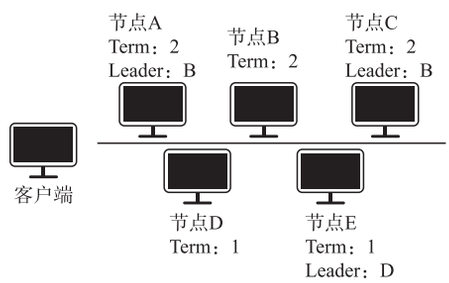

图 3 脑裂情况示意图

4 试验测试

4.1 试验环境介绍

试验环境为 Windows 10 操作系统；AMD Ryzen 5 600 H，3.30 GHz；内存为 16 GB，DDR4；2 T 固态硬盘；使用 Golang1.17 实现系统的业务逻辑；可验证随机函数实现方面基于 Golang 的 crypto 包，椭圆曲线采用 ed25519，Hash 函数采用 sha-256 以及 sha-512。

4.2 可验证随机函数测试

试验首先对 VRF 进行性能测试，本试验实现的 VRF 主要包括 3 个模块：生成公私钥对、生成随机数及其证明、验证随机数及其证明。VRF 各模块的算法运行时间如表 2 所示。

由测试结果可以看出，本试验设计的 VRF 各模块算法的运行时间都在 ms 级。将这样时间消耗的算法添加在 Raft 算法的 Leader 选举过程中，不

会增加原算法时间上的复杂度。

表 2　VRF 各模块的算法运行时间

函数模块	运行次数	总共用时/ms	平均用时/ms
生成公私钥对	10 000	647.010 9	0.064 7
生成随机数及其证明	10 000	4 513.202 3	0.451 3
验证随机数及其证明	10 000	5 463.327 1	0.546 3

4.3　选主用时测试

对系统存在 50～200 节点分别进行选主时间测试，测试结果见图 4 所示。

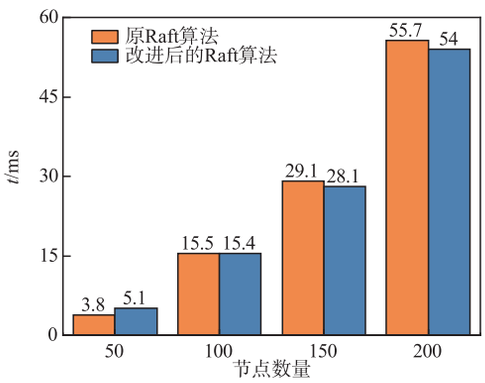

图 4　原算法与改进算法选主时间对比

由图 4 可知，原 Raft 算法和本试验改进的 Raft 算法在节点增多的情况下选主用时都是逐渐上升的。但在节点增多的情况下原算法更有可能出现因投票分歧而影响系统的执行效率的问题。在节点越来越多的情况下，改进后的算法选主用时比原 Raft 算法越来越少的原因在于规避了原算法可能出现的因投票分歧而导致选举超时重新进行选举的问题，因此，在系统节点越来越多时改进的 Raft 算法执行效率会越来越高。

4.4　脑裂情况下选主用时测试

使用 Mininet 搭建了仿真网络，在该仿真网络中对系统可能会出现的脑裂情况进行模拟，并分别对原 Raft 算法和改进后的 Raft 算法在脑裂情况下选主用时进行测试，测试结果见图 5。

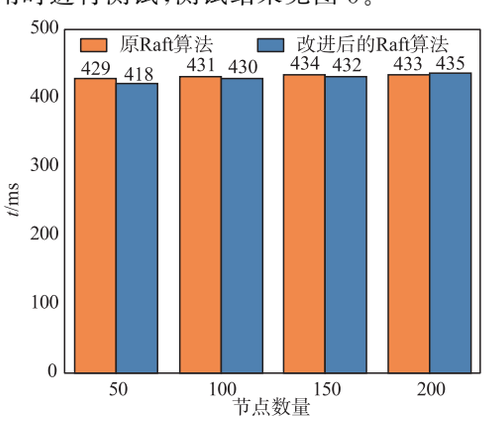

图 5　原算法与改进算法在脑裂情况下的选主时间对比

由图 5 可知，整体的选主时间较没发生脑裂情况时提高了许多，这是由于网络正常运行时，Raft 算法中的候选人能够相互交流，并通过投票来选举出新的 Leader。然而，在脑裂的情况下，可能会出现分区中的候选人无法与其他分区中的节点进行通信的问题，会导致无法达成多数选票，无法选举出 Leader。具体来说，如果一个候选人发送选举请求并等待投票，但没有足够的节点能够响应其请求，那么该候选人将无法获得多数选票，选举将无法完成。在这种情况下，Raft 算法会等待一段时间，然后重试选举过程，直到有足够的节点能够参与选举。因此，脑裂会导致选举 Leader 的时间延长，具体取决于发生脑裂情况的程度和持续时间。

而原 Raft 算法和改进后的 Raft 算法在脑裂情况下的不同节点数的选主用时很接近的原因是每个节点的选举超时时间是随机生成的，并且每个节点都会在超时后成为候选人并广播选举请求。如果脑裂情况发生，节点之间的通信可能会受到影响，导致某些节点无法接收到其他节点的选举请求，这些节点将不会投票给其他节点，并且不会增加任何节点的票数。因此，因为每个节点都是独立进行选举的，即使节点数量不同，每个节点的选举用时也差不多。另外，节点之间的通信也受到了网络分区的模拟，这意味着有些节点可能无法与其他节点进行通信。这会导致节点无法接收到其他节点的心跳消息，从而无法成为领导者。因此，即使节点数量不同，选举用时也可能相似。

4.5　DDOS 攻击时选主用时测试

同样的，我们在仿真网络中模拟了 DDOS 攻击，并分别测试了原 Raft 算法和改进后的 Raft 算法在 DDOS 攻击下的表现，测试结果见图 6。

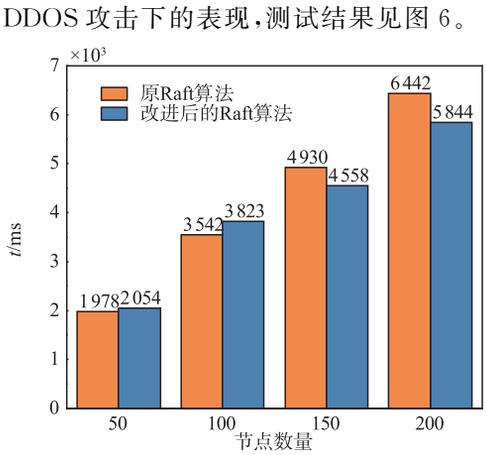

图 6　原算法与改进算法完成共识时间对比

由图 6 可知，改进后的 Raft 算法在 DDOS 攻击下明显比原 Raft 算法表现更优，主要原因有：

1）提高了选主过程的效率。改进后的 Raft 算

法引入了可验证随机函数，使选主过程更高效。可验证随机函数可以帮助减少选举过程中的不必要的消息传输和等待时间，从而降低了选主所需的时间。

2）降低了网络传输负载。DDoS 攻击可能会导致网络拥塞，使得消息传输延迟增加。改进后的 RAFT 算法通过在选举过程中减少消息传输的数量，减轻网络传输负载，进而减少选主的用时。

3）提高消息认证和安全性。通过引入可验证随机函数，增加了消息认证和安全性，因为可验证随机函数可以帮助验证消息的真实性和完整性。这可以防止攻击者发送伪造的消息来破坏 Raft 算法的正常运行。

4.6 完成共识用时测试

分别对系统存在 50～200 节点进行共识完成时间测试，测试结果见图 7。

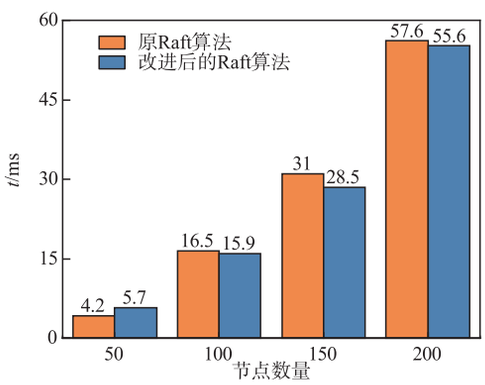

图 7 原算法与改进算法完成共识时间对比

由图 7 可知，原 Raft 算法和本实验改进的 Raft 算法在节点增多的情况下完成共识用时同样是逐渐上升的。第 4.6 节中试验用时与第 4.3 节试验用时相差不大，这也符合在第 4.2 节测试出的 VRF 各模块用时很少的结果，符合试验的总体预期。这里要注意的是，本节完成共识用时不是指所有日志复制成功所用时间，而是系统内节点验证通过了 Leader 发送的身份验证信息且没有错误反馈所用的时间。后续的日志复制等操作所用时间受各节点的网络环境和设备性能等因素的影响。

4.7 稳定性测试

分布式系统由于各个节点间都是独立的，经常会遇到某个节点或一群节点因为各种原因而失去联系的情况，所以设计算法也需要对系统进行可靠性测试。本实验分别测试了有 5 个节点和 10 个节点的系统中若干节点失联情况下系统的可靠性，测试结果见图 8。

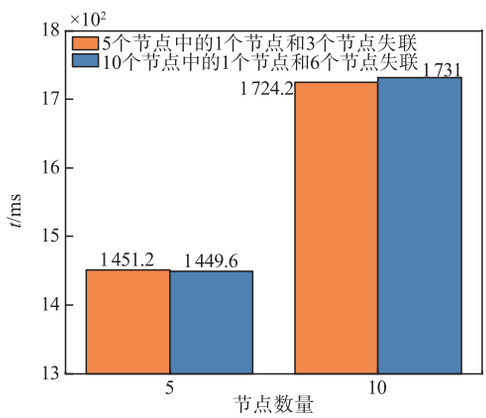

图 8 系统稳定性测试

试验测试了系统在失去若干节点联系情况下的运行状态，测试了从失去联系到节点重联所用的时间。这里的实验结果与是否应用可验证随机函数并无关联，实验设计此部分是为了验证加入可验证随机函数是否会影响系统的稳定性。结果表明系统在失去若干节点联系到节点重联的过程中仍能保持稳定的运行状态。

4.8 不同共识算法的对比

设计各种共识算法是为了适应不同的应用场景，表 3 为联盟链中常用的共识算法在各需求场景下的表现比较。

表 3 不同共识算法在各需求场景下的表现对比

共识算法	一致性	易于学习和实现	性能	容错性	抗 DDOS 攻击	可靠性
ZooKeeper	弱	弱	强	较强	弱	强
Paxos	强	较强	较弱	弱	较弱	弱
PBFT	较弱	较弱	弱	强	强	较强
Raft	较强	强	较强	较弱	较强	较弱
本文算法	较强	强	较强	较强	强	强

由表 3 可知，本文通过在 Raft 算法中引入可验证随机函数，在保留了原 Raft 算法优点的基础上减少了选举过程中信息通信量，减轻了网络流量负载，并且利用可验证随机函数的加密与解密机制帮助验证消息的真实性和完整性，提高了系统的整体可靠性。本文改进后的 Raft 算法在非拜占庭容错类共识算法中综合表现最优。

5 结语

本文分析了目前联盟链常用的 Raft 算法在运行过程中存在的安全隐患，在保留原算法系统安全

性和稳定性的前提下利用可验证随机函数的随机性、非交互性和零知识证明技术对算法进行改进。在选举 Leader 过程中隐藏关键节点的身份来提高安全性并且规避了原算法中存在的因投票分歧导致的选举超时问题。通过详细的试验和分析结果表明，改进后的 Raft 算法具有更高的安全性和可靠性，在系统的选主用时、完成共识用时等方面也都有更优秀的表现。通过对比其他共识算法表明本文算法在非拜占庭容错类共识算法中综合表现最优。但是试验设计在整体性考虑上还有一些局限，不仅要考虑选主过程的改进，还需要对日志复制过程也有响应改进。未来的工作将进一步对可验证随机函数对 Raft 算法整体的改进效果进行研究，并且使用更多的安全方法进一步提高系统的安全性和可靠性。

参考文献

[1] NAKAMOTO S. Bitcoin：A Peer-to-Peer Electronic Cash System[EB/OL]. (2009-03-01)[2023-05-01]. https://bitcoin.org/bitcoin.pdf.

[2] 王群，李馥娟，倪雪莉，等. 区块链共识算法及应用研究[J]. 计算机科学与探索，2022，16(6)：1214-1242.

[3] 邵奇峰，金澈清，张召，等. 区块链技术：架构及进展[J]. 计算机学报，2018，41(5)：969-988.

[4] 代闯闯，栾海晶，杨雪莹，等. 区块链技术研究综述[J]. 计算机科学，2021，48(S2)：500-508.

[5] 孟昊同，张大伟. Hyperledger Fabri 共识机制优化方案[J]. 自动化学报，2021，47(8)：1885-1898.

[6] PATRICK H，MAHADEV K，FLAVIO P J，et al. ZooKeeper：Wait-Free Coordination for Internet-Scale Systems[C]//Proceedings of the 2010 USENIX conference on USENIX Annual Technical Conference. Washington D C：USENIX Association，2010.

[7] LESLIE L. The Part-Time Parliament[J]. ACM Transactions on Computer Systems，1998，16(2)：133-169.

[8] ONGARO D，OUSTERHOUT J K. In Search of an Understandable Consensus Algorithm[C]// Proceedings of the 2014 USENIX Conference on USENIX Annual Technical Conference. New York：ACM，2014：305-320.

[9] LAMPORT L，SHOSTAK R，PEASE M. The Byzantine Generals Problem[J]. ACM Transactions on Programming Languages & Systems，1982，4(3)：382-401.

[10] YANG C，LIU M，WANG K，et al. Review on Variant Consensus Algorithms Based on PBFT[C]//Artificial Intelligence and Security. Singapore：Springer，2020：37-45.

[11] MICALI S，RABIN M，VADHAN S. Verifiable Random Functions[C]// 40th Annual Symposium on Foundations of Computer Science. Piscataway：IEEE Computer Society，1999：6431101.

[12] NGUYEN G T，KIM K. A Survey about Consensus Algorithms Used in Blockchain[J]. Journal of Information Processing Systems，2018，14(1)：101-128.

[13] 吴奕，仲盛. 区块链共识算法 Raft 研究[J]. 信息网络安全，2021，21(6)：36-44.

[14] AGRAWAL N，TAPASWI S. Defense Mechanisms Against DDoS Attacks in a Cloud Computing Environment：State-of-the-Art and Research Challenges[J]. IEEE Communications Surveys and Tutorials，2019，21(4)：3769-3795.

[15] 荣宝俊，郑朝晖. FL_Raft：基于联邦学习模型的选举共识方案[J]. 计算机科学，2023，50(11)：364-373.

[16] 邹贤，沈力，张伟，等. 基于信用评分的电力交易区块链改进 RAFT 共识机制[J]. 南方电网技术，2022，16(6)：132-139.

[17] HUANG D，MA X，ZHANG S. Performance Analysis of the Raft Consensus Algorithm for Private Blockchains[J]. IEEE Transactions on Systems，Man，and Cybernetics：Systems，2020，50(1)：172-181.

[18] RONG B，ZHENG Z. FRCR：Raft Consensus Scheme Based on Semi Asynchronous Federal Reconstruction[J]. IEEE Transactions on Network and Service Management，2022，19(4)：3822-3834.

[19] DU Z，QU Z，FU Y，et al. Multi-Strategy Based Leader Election Mechanism for the Raft Algorithm[J]. Concurrency and Computation：Practice and Experience，2023：e7734.

[20] KALBHOR S，GAIKWAD A，BHISE K，et al. A Survey on Digital Signature[J]. International Journal of Emerging Technology and Advanced Engineering，2015，5(1)：533-536.

[21] GOLDBERG S，REYZIN L，PAPADOPOULOS D，et al. Verifiable Random Functions (VRFs)[EB/OL]. (2022-06-16)[2023-05-01]. https://datatracker.ietf.org/doc/draft-irtf-cfrg-vrf.

（编辑：徐楠楠）