面向异步网络的安全分布式随机数通用构造

张宗洋[1),2)] 李 彤[1)] 周 游[1)] 陈 劳[1)]

[1)](北京航空航天大学网络空间安全学院 北京 100091)

[2)](云南省区块链应用技术重点实验室 云南 650233)

摘 要 随机数在密码学和区块链领域扮演着极其重要的角色，如密码学中的安全参数生成、共识机制中的委员会重配置以及电子投票的智能合约应用等. 近年来针对不依赖可信第三方的分布式随机数生成技术的研究受到了越来越多的关注，其中基于秘密共享的方案数量最多，但普遍通信复杂度较高，而通信复杂度较低的一些方案通常牺牲了随机数的抗偏置性和不可预测性. 另外，现有大多数方案基于同步网络模型，网络假设较强，与现实网络环境不符. 本文主要研究基于秘密共享的分布式随机数生成技术，抽象出不同方案的共性并兼容特殊性，以及弱化网络假设和优化通信开销. 具体贡献如下：（1）提出了交互的分布式随机数生成通用构造. 满足伪随机性、唯一性和鲁棒性安全目标，并使用该通用构造对一个分布式随机数生成方案进行了分析，从而更加证明其通用性；（2）设计了面向异步网络的安全分布式随机数生成方案. 将现有方案 $O(n^3)$ 和 $O(f^2n^2)$ 的通信复杂度降为 $O(fn^2)$，将 $O(fn^2)$ 计算复杂度降为 $O(n^2)$；（3）实现了分布式随机数仿真系统. 针对不同节点数进行了性能测试，测试结果表明，相同节点数下，本方案比现有异步方案的通信开销更低，性能有所提升. 最好情况下本方案比现有异步方案在通信开销方面分别降低了 11%和 47%.

关键词 分布式随机数；秘密共享；异步网络；通用构造；通信复杂度

中图法分类号 TP309 **DOI 号** 10.11897/SP.J.1016.2023.00163

A General Construction of Secure Distributed Randomness Generation for Asynchronous Networks

ZHANG Zong-Yang[1),2)] LI Tong[1)] ZHOU You[1)] CHEN Lao[1)]

[1)](*School of Cyber Science and Technology, Beihang University, Beijing* 100091)

[2)](*Yunnan Key Laboratory of Blockchain Application Technology, Yunnan* 650233)

Abstract Randomness plays an extremely important role in the field of cryptography and blockchain, such as the secure generation of protocol parameters for cryptographic schemes, committee reconfiguration in consensus mechanisms, and smart contract applications for electronic voting. However, the generation of randomness in existing applications mostly relies on a trusted third-party. Therefore, in order to improve the transparency of the randomness generation process, the research on distributed randomness generation has received more and more attention in recent years. Existing distributed randomness schemes can be divided into 6 types according to the underlying cryptographic primitives, such as verifiable random function, threshold signature, secret sharing, verifiable delay function and so on. Among them, schemes based on secret sharing are the most, but generally the communication complexity is high, and some schemes with

收稿日期：2021-12-07；在线发布日期：2022-06-16. 本课题得到国家重点研发计划（2021YFB2700200）、国家自然科学基金（61972017, 72031001, 61972310）、北京市自然科学基金（M22038, M21033, 4202037）、云南省区块链应用技术重点实验室开放课题（202105AG070005）、中央高校基本科研业务费（YWF-22-L-1039）资助. **张宗洋**（通信作者），博士，副教授，CCF高级会员，主要研究领域为区块链和密码学. E-mail: zongyangzhang@buaa.edu.cn. **李 彤**，硕士研究生，CCF学生会员，主要研究领域为区块链和密码学. **周 游**，本科生，CCF学生会员，主要研究领域为区块链和密码学. **陈 劳**，博士研究生，助理研究员，主要研究领域为网络安全和区块链. 所有作者贡献相等.

low communication complexity usually sacrifice the bias-resistance and unpredictability. In addition, most of the existing schemes are based on a synchronous network model, which are under strong network assumptions and inconsistent with the real network environment. These limitations make it difficult for the distributed randomness generation to be applied to specific protocols or real applications.

This work focuses on the distributed randomness generation technology based on secret sharing. We mainly study on the extraction of the commonalities and compatibility of different schemes, as well as the weakening of network assumptions and the optimization of communication overhead. On the basis of proposing a general structure of distributed randomness, we design and implement a secure distributed randomness generation system for asynchronous networks. Specifically, the main innovations of this paper include the following aspects:

(1) We propose a general construction for interactive distributed random beacon. Compared with non-interactive distributed randomness beacon, the general construction proposed in this paper does not rely on a complex distributed key generation protocol. It is composed of secret sharing and consensus, including initialization, node selection, secret sharing, secret recovery, randomness computation, randomness verification, and state update. This construction meets the security goals of pseudo-randomness, uniqueness and robustness. In addition, we analyze an instantiation in this paper, thereby further demonstrating the universality of the construction.

(2) We design a secure distributed randomness generation scheme for asynchronous networks. This scheme is mainly oriented to a small-scale permissioned environment. Aiming at the key problem that asynchronous binary agreement cannot be used, asynchronous verifiable secret sharing is used as a sub-module of the scheme, and a voting consensus method is combined on its basis. In addition, this paper presents the formal security proof of the pseudo-randomness, uniqueness and robustness of the scheme. In terms of complexity analysis, this scheme has achieved $O(fn^2)$ communication complexity and $O(n^2)$ computation complexity and is better than the current asynchronous schemes.

(3) We implement the distributed randomness simulation system. This system from bottom to top is the core layer, the network layer, the general layer and the service layer. On this basis, the simulation system has been tested for the computation time and communication overhead under different numbers of nodes. The results show that our scheme has lower communication overhead and the performance is improved, compared with the existing asynchronous schemes, in the case of the same number of nodes. The communication overhead of our scheme is 11% and 47% lower than the existing asynchronous schemes.

Keywords distributed randomness; secret sharing; asynchronous network; generic construction; communication complexity

1 引 言

现有应用中随机数的生成大多依赖于可信第三方机构，主要包括由美国国家标准与技术研究院（NIST）或 Random.org[①]等网站提供的随机数. 由于随机数在生成和使用过程中的不透明性，其产生的各种问题层出不穷，比如双椭圆曲线伪随机数生成器的后门泄露[1]、区块链智能合约中的抢先交易攻击（Front Running Attack）[2]等，上述情境下，随机数应用将无法保证安全性. 因此，为了提高随机数生成的透明性，近年来针对分布式随机数生成技术的研究受到了越来越多的关注.

所谓分布式随机数生成技术，就是在无可信第三方的环境下，由一组参与者自己生成公开可验证的随机数，从而防止中心机构作弊，避免单点故障，提高安全性. 然而在复杂的实际应用场景下，参与者之间往往是互不信任的，其中也会存在着若干个

① RANDOM.ORG, https://www.random.org/

恶意参与者，企图干预随机数的输出结果，无法保证安全性.

近年来，使用不同密码学技术、具有不同特点、面向不同场景的分布式随机数生成方案被提出. 例如，RandShare[3]、SCRAPE[4]和 HydRand[5]适用于数量小的节点集合，但当节点数量增多，通信复杂度会较高，可扩展性较差. RandHound[3]和 RandHerd[3]都将参与节点分为多个子集合，然而 RandHound 不提供抗偏置性，RandHerd 需要在初始化阶段执行分布式密钥生成[6]（Distributed Key Generation，DKG）协议. Dfinity[7]提供很强的抗偏置性，但同样依赖于复杂度高的 DKG 协议；Algorand[8]和 Ouroboros Praos[9]具有很好的可扩展性，但不保证抗偏置性.

可见，如何在保证安全性的前提下，设计适应复杂网络环境、具有低通信复杂度的分布式随机数生成方案，仍是具有挑战性的研究热点. 本文主要研究基于秘密共享的分布式随机数生成技术，聚焦于通用构造的研究，以及网络假设的弱化和通信开销的优化，具体贡献如下：

第一，提出了一个交互的分布式随机数生成通用构造，详细定义了系统模型和安全模型. 另外，使用该通用构造对现有方案进行实例化分析，分析结果表明该构造具备通用性.

第二，提出了一个面向异步网络的安全分布式随机数生成方案，该方案基于异步可验证秘密共享密码学原语，满足所提通用构造的安全属性，通信复杂度为 $O(fn^2)$，在异步方案中较优.

第三，实现了一个分布式随机数仿真系统，该实现包含四个核心模块，测试了不同节点数下各个算法平均计算耗时和带宽消耗，测试结果表明该仿真系统相对于现有面向异步网络的分布式随机数生成方案具备较优的通信开销.

2　相关工作

根据所依赖的底层密码学技术，现有典型的分布式随机数生成方案可以分为基于秘密共享的方案[3-5, 10-17]、基于可验证随机函数的方案[7-19]、基于门限签名的方案[7,20]和基于可验证延迟函数的方案[21-23]等类别. 其中基于秘密共享的方案数量最多，占比最大，且不断有相关改进和优化，因此是本文研究的重点，下面将介绍基于秘密共享的方案大体流程和原理.

Shamir[10]在 1979 年提出秘密共享（Secret Sharing）的概念，即一个分发者将秘密分割成多个份额并分发给参与者，原始秘密可以通过份额的多项式插值被重建. 然而秘密共享有一个很重要的前提：分发者和参与者都需要是诚实的. 为了解决可能存在恶意参与者的安全问题，可验证秘密共享（Verifiable Secret Sharing，VSS）[11]被提出. 此外，Stadler[13]在 1996 年提出公开可验证秘密共享（Public Verifiable Secret Sharing，PVSS），任意用户都可以通过公开信息验证份额的正确性. 在份额分发阶段，分发者为每个参与者计算加密份额和确保份额加密正确性的非交互式零知识证明；在秘密重建阶段，参与者通过广播解密份额和相应的证明来恢复原始秘密. 另外，Kokoris-Kogias、Malkhi 和 Spiegelman 在 2020 年提出高门限值的异步可验证秘密共享[17]（High-Threshold Asynchronous Verifiable Secret Sharing，HAVSS）来解决 VSS 和 PVSS 网络模型的局限性. 基于秘密共享的分布式随机数生成方案主要流程为：首先每个节点私下生成一个随机秘密值，并将该秘密值分割成多个份额，然后向所有节点广播份额和相应的承诺，最后在验证通过后揭示秘密值. 如果某个节点没有揭示秘密值，那么其他诚实节点可以使用收到的份额共同恢复出原始秘密，最后将多个秘密值合并后作为随机数输出.

典型的基于 VSS 的分布式随机数生成方案有 RandShare[3]和 BRandPiper[12]，其中 RandShare 是基于异步网络的，但其通信复杂度为 $O(n^3)$. BRandPiper 依赖于同步网络假设，解决了适应性敌手可以预测未来 f+1 轮随机数的问题，将可预测性限定在 1 轮内，通信复杂度为 $O(fn^2)$.

基于 PVSS 的典型方案有 RandHound[3]、RandHerd[3]，SCRAPE[4]，HydRand[5]，ALBATROSS[14]、Ouroboros[15]和 SPURT[16]等. 其中 RandHound 是客户端-服务器模型，将服务器分为多个组，并将每个组内部生成的随机数结合起来输出唯一的随机数值. RandHerd 首先使用 RandHound 公平分组，组内使用 DKG，并通过集体签名技术生成持续的随机数值. HydRand 是在 SCRAPE 的 PVSS 方案基础上，考虑了初始化和协议执行流程（包括提案、确认、投票和领导者选举阶段）. ALBATROSS 在 SCRAPE 基础上进行了计算复杂度的优化，并且是第一个具有通用可组合安全的分布式随机数生成方案. SPURT 是第一个半同步模型下的基于 PVSS 的分布式随机数生成协议，该协议首先提出基于判定性 Diffie-Hellman 问题的 PVSS 方案，并使用拜占庭容错共识机制达成所有节点间份额收发的一致性，最好情况下通信复杂度为 $O(n^2)$，最坏情况下通信复杂度为 $O(n^2\log n+n^3)$.

基于 HAVSS 的方案仅有 Kokoris-Kogias、Malkhi

和 Spiegelman 的工作[17],该方案是使用 n 个 HAVSS 实例构造弱分布式密钥生成方案，并在此基础上结合门限签名技术[24]构造出最终完美的公共随机数（Eventually Perfect Common Coin，EPCC），但该方案的通信复杂度为 $O(f^2n^2)$.

现有研究通过不同的技术路线对分布式随机数生成方案进行了探究，但在通用构造、安全性、网络假设与复杂度的权衡等方面仍存在一些未解决的问题.

问题一：缺乏关于分布式随机数生成通用构造及安全性证明研究. 针对基于秘密共享的方案，其数量占比最高，但现有相关工作均为设计具体的方案，而没有分析不同方案之间的联系，没有针对通用构造的研究，缺少方案通用模型的具体分析.

问题二：分布式随机数生成方案的网络假设较强. 目前大多数方案都在同步模型下执行，网络假设相对较强，不适用于实际应用场景. 例如在 HydRand[5]中,必须仔细选择每一轮的持续时间参数以满足同步模型假设来避免协议意外终止.

问题三：基于秘密共享的方案通信复杂度较高. 在现有基于秘密共享的方案中，通信复杂度较优的是 RandHerd[3]、RandHound[3]和 HydRand[5]方案，但 HydRand[5]牺牲了随机数的不可预测性，只能保证 f+1 轮之后的不可预测性，RandHound[3]牺牲了抗偏置性，RandHerd[3]在初始化阶段需要执行分布式密钥生成协议，且当领导者作恶时，依赖于视图转换技术. 而基于异步网络的方案产生了更高的通信复杂度和计算复杂度,达到了 $O(n^3)$ 甚至 $O(n^4)$ 的复杂度（如 RandShare[3]、EPCC[17]）. 因此，在未来的工作中不仅应考虑如何解决网络模型所带来的局限性，也要在保证安全性的基础上降低通信复杂度.

3 背景知识

本章介绍后文中将使用到的背景知识. 第 3.1 节介绍 KZG 多项式承诺方案. 第 3.2 节介绍 HAVSS 方案. 第 3.3 节介绍共识机制的安全属性.

3.1 KZG承诺方案

本小节介绍针对多项式的承诺. 多项式 $\phi(x)=\phi_0+\phi_1x+\cdots+\phi_nx^n$ 的表示方式有两种,分别是系数表示 $\phi_0,\phi_1,\ldots,\phi_n$ 和点值表示 $(x_0,y_0),\cdots,(x_n,y_n)$，可以通过快速傅里叶变换（Fast Fourier Transform，FFT）将系数表示转换到点值表示. 多项式承诺是指承诺方对点值计算承诺，最经典的为 Kate、Zaverucha 和 Goldberg[25]在 2010 年提出的承诺方案，即 KZG 承诺，主要包括四个算法 $\Pi_{\mathrm{KZG}}=(\mathrm{Setup},\mathrm{Com},\mathrm{Eval},\mathrm{Verify})$.

- $\mathrm{Setup}(1^\kappa,f)\to pp$：这是初始化算法. 输入安全参数 κ 和 f，初始化两个阶为素数 p（κ 比特安全）的群 $\mathbb{G}$ 和群 $\mathbb{G}_T$，存在对称双线性映射 $e:\mathbb{G}\times\mathbb{G}\to\mathbb{G}_T$，且 t-SDH 假设成立. 随机选取群 $\mathbb{G}$ 生成元 $g\in_R\mathbb{G}$，随机选取 $\alpha\in_R\mathbb{Z}_p^*$，计算生成 $f+1$ 元组 $\langle g,g^\alpha,\cdots,g^{\alpha^f}\rangle\in\mathbb{G}^{f+1}$，输出公共参数 $pp=(e,\mathbb{G},\mathbb{G}_T,\langle g,g^\alpha,\cdots,g^{\alpha^f}\rangle)$.
- $\mathrm{Com}(pp,\phi(x),d)\to\hat{\phi}$：这是多项式承诺算法. 输入公共参数 pp、多项式 $\phi(x)=\sum_{j=0}^{d}\phi_jx^j$ 和多项式阶数 $d\leqslant f$，输出承诺 $\hat{\phi}=\prod_{j=0}^{d}(g^{\alpha^j})^{\phi_j}$.
- $\mathrm{Eval}(pp,\phi,i)\to y$：这是多项式点值计算算法. 输入公共参数 pp、多项式 ϕ、点坐标 i、计算 $\psi_i(x)=\frac{\phi(x)-\phi(i)}{(x-i)}$，计算证据 $w_i=g^{\psi_i(\alpha)}$，输出 $y=(i,\phi(i),w_i)$.
- $\mathrm{Verify}(pp,\hat{\phi},y,d)\to1/0$：这是验证算法. 输入公共参数 pp、多项式承诺 $\hat{\phi}$、点值计算结果 $y=(i,\phi(i),w_i)$ 和多项式阶数 d，验证 $e(\hat{\phi},g)=e(w_i,g^\alpha/g^i)e(g,g)^{\phi(i)}$ 是否成立，若成立，则输出 1，否则输出 0. 上述等式计算过程为

$$\begin{aligned}&e(w_i,g^\alpha/g^i)e(g,g)^{\phi(i)}\\&=e(g^{\psi_i(\alpha)},g^{\alpha-i})e(g,g)^{\phi(i)}\\&=e(g,g)^{\psi_i(\alpha)(\alpha-i)+\phi(i)}\\&=e(g,g)^{\phi(\alpha)}\\&=e(\hat{\phi},g).\end{aligned}$$

3.2 高门限值的异步可验证秘密共享

Kokoris-Kogias、Malkhi 和 Spiegelman[17]在 2020 年提出 HAVSS,通信复杂度为 $O(n^4)$. 随后 Alhaddad、Varia 和 Zhang[26]对上述方案进行优化，提出了复杂度为 $O(n^2)$ 的 HAVSS 方案.

本小节介绍 Alhaddad、Varia 和 Zhang[26]设计的 HAVSS 方案，该方案采用两个一元多项式，其中一个为秘密分发多项式（Share Polynomial），另一个为秘密份额恢复多项式（Recovery Polynomial）. 为了后续协议描述方便，对原文方案描述做了部分变动，HAVSS 方案主要分为以下四个算法.

算法 1 主要分发秘密 s 并生成相应的多项式承诺，具体流程如下. 首先选取阶为 p 的多项式 R 且

$R(0)=s$，再选取阶为 f 的 n 个多项式 S_i 且 $S_i(i)=R(i)$，之后计算对多项式 R 和 S_i 的承诺 $\hat{R}$ 和 $\hat{S}_i$，以及每个多项式 S_j 在 i 处的取值，再根据承诺同态性计算多项式 $T_i=R-S_i$ 的承诺和在 i 点的取值，最后针对 $\hat{R}$ 和 n 个 $\hat{S}_i$ 计算根承诺 C.

算法 2.1 和算法 2.2 分别为秘密分发者消息的验证和参与节点消息的验证，依次验证多项式 S_i 和 T_i 的点值计算结果，以及根承诺的正确性.

算法 3 主要进行秘密份额的恢复，任意 $f+1$ 个有效的点值能够恢复相应秘密份额，具体流程如下所示.

算法 4 主要进行秘密的恢复，先验证所收到的秘密份额是否正确，然后验证多项式承诺 $\hat{S}_i$ 是否正确，若验证均通过，则可根据任意 $p+1$ 个有效的秘密份额恢复相应秘密.

算法 1. 秘密分发

输入：秘密 S

输出：根承诺 C 及证据

1. 随机选取阶为 p 的多项式 R，使得 $R(0)=s$；

2. 对于所有节点 $i\in[1,n]$，随机选取阶为 f 的多项式 S_i，使得 $S_i(i)=R(i)$；

3. 计算多项式承诺 $\hat{R}=\mathrm{Com}(pp,R,p)$；

4. 对于每个多项式 S_i 计算多项式承诺 $\hat{S}_i=\mathrm{Com}(pp,S_i,f)$；

5. 对于所有节点 $i\in[1,n]$，计算点值 $S_j(i)=\mathrm{Eval}(pp,S_j,i)$，其中 $j\in[1,n]$；

6. 对于所有节点 $i\in[1,n]$，计算多项式承诺 $\hat{T}_i=\mathrm{Hom}(\hat{R},\hat{S}_i,-1)$，并计算点值 $T_i(i)=\mathrm{Eval}(pp,T_i,i)$；

7. 计算根承诺 $C=\mathrm{vCom}(pp,\langle\hat{R},\hat{S}_1,\cdots,\hat{S}_n\rangle)$ 及其证据.

算法 2.1　秘密分发者消息的验证

输入：根承诺 C、多项式承诺 $\hat{R}$ 和 $(\hat{S}_1,\cdots,\hat{S}_n)$、$(S_1(i),\cdots,S_n(i))$、$(T_1(1),\cdots,T_n(n))$

输出：TRUE 或 FALSE

1. 令 $\hat{\boldsymbol{S}}=(\hat{S}_1,\cdots,\hat{S}_n),\boldsymbol{y_i}=(S_1(i),\cdots,S_n(i)),$

2. $\boldsymbol{y}=(T_1(1),\cdots,T_n(n))$；

3. 验证 $\mathrm{Verify}(pp,\hat{S}_i,S_j(i),f)=1$ 和 $\mathrm{Verify}(pp,\hat{T}_i,T_i(i),p)=1$ 是否成立；

4. 验证多项式承诺 $\hat{R}$ 和所有 $\hat{S}_i$ 是否在根承诺 C 中；

5. 验证 $T_i(i)=0$ 是否成立.

算法 2.2　参与节点消息的验证

输入：根承诺 C、多项式承诺 $\hat{S}_t$、$S_t(j)$

输出：TRUE 或 FALSE

1. 验证多项式承诺 $\hat{S}_t$ 是否在根承诺 C 中；

2. 令 $y_t=S_t(j)$，验证 $\mathrm{Verify}(pp,\hat{S}_t,S_t(j),f)=1$ 是否成立.

算法 3.　秘密份额恢复流程

输入：$(S_1(j),\cdots,S_n(j))$

输出：秘密份额 R_i

1. 令 $y_i=S_i(j)$，对至少 $f+1$ 个有效的 $S_i(j)$ 执行拉格朗日差值计算得到秘密分享多项式 S_i；

2. 计算点值 $R_i=S_i(i)=\mathrm{Eval}(pp,S_i,i)$.

算法 4.　秘密恢复流程

输入：多项式承诺 $(\hat{S}_1,\cdots,\hat{S}_n)$、$(S_1(i),\cdots,S_n(i))$

输出：秘密值 s

1. 令 $y_i^*=S_i(i)$，验证 $\mathrm{Verify}(pp,\hat{S}_i,S_i(i),f)=1$ 是否成立；

2. 验证多项式承诺 $\hat{S}_i$ 是否在根承诺 C 中；

3. 对 $p+1$ 个有效的 $R_i=S_i(i)$ 执行拉格朗日差值计算得到原秘密值 $s=R(0)$.

一个 (n,p,f)-HAVSS 方案具有如下四个安全属性，其中 $f<n/3$，$p<n-f$.

（1）正确性（Correctness）：一旦 p 个诚实节点完成秘密分享过程，那么存在一个值 z 满足（a）若秘密分发者诚实，则 $z=s$；（b）诚实节点恢复的秘密 $z_i=z$.

（2）活性（Liveness）：若秘密分发者在秘密分享阶段是诚实的，那么所有诚实节点都能完成秘密分享过程.

（3）一致性（Agreement）：若一些诚实节点完成秘密分享过程，那么所有诚实节点都能完成；如果所有诚实节点接着开启秘密恢复过程，那么所有诚实节点都能恢复秘密.

（4）隐私性（Privacy）：若秘密分发者分享了秘密 s，且少于 $p-f$ 个诚实节点开启秘密恢复过程，那么恶意节点无法得知 s.

3.3　共识机制

共识机制是区块链技术的基础和核心，其决定协议参与节点以何种方式对某些特定的数据达成一致. 经典的分布式共识主要在节点之间完成状态机复制，具备一定的容错能力，允许一定比例的节点出现宕机等故障或遭受敌手攻击.

共识机制需满足两个重要的安全属性：一致性（Consistency）和活性（Liveness）. 其中一致性是指所有诚实节点最终输出的消息是相同的，活性是指任意诚实节点收到的消息在一定时间后其他诚实节点也会收到. 在本文后续描述中，使用 Consensus (m,r) 表示共识流程，其中 m 为要共识的消息，r 表示当前轮数.

共识机制可以分为经典分布式共识[27]、授权共识、基于工作量证明的共识、基于权益证明的共识

等[28]. 本文主要涉及经典分布式共识，如 RandShare[3]、SCRAPE[4]、HydRand[5]和 SPURT[16]等方案均采用了经典分布式共识中拜占庭类共识的思想. 上述方案将协议分为多个阶段（如预准备、准备、承诺），需要节点间进行多轮交互，当且仅当收到一定数量的某类型消息后，再进入下一阶段，从而防止恶意节点干扰协议正常执行.

4 I-DRB 的通用构造与安全模型

分布式随机数生成方案从是否交互的角度可以分为交互式方案（基于秘密共享）和非交互式方案（基于 VRF 和门限签名）. David 等人[29]在 2021 年提出非交互的分布式随机数生成方案（Non-Interactive Decentralized Random Beacon，NI-DRB）的形式化定义与安全模型，但 NI-DRB 需要复杂的分布式密钥生成协议，该协议使得参与节点无法进行更新，若有新节点参与，则需要重新执行分布式密钥生成协议分配密钥. 为此，本文给出了交互的分布式随机数生成方案（Interactive Decentralized Random Beacon，I-DRB）的形式化定义、通用构造和安全模型.

第 4.1 节给出 I-DRB 的通用构造，并说明与 NI-DRB 的不同之处. 第 4.2 节介绍 I-DRB 的安全模型，即应当满足的安全目标. 此外，在附录中给出了使用 I-DRB 通用构造对一个基于 PVSS 的分布式随机数生成方案的分析.

4.1 通用构造

David 等人[29]提出由分布式可验证随机函数到 NI-DRB 的通用构造，其中分布式可验证随机函数需要每个节点使用各自私钥作为函数输入，再将输出聚合得到唯一的随机数，且需要一个共同公钥对该随机数进行验证，因此 NI-DRB 依赖 DKG 协议分配一个共同公钥和多个私钥. 考虑到 DKG 协议是针对固定节点数生成相应的私钥，因此若有新节点加入，则需重新执行该协议为新节点分配私钥.

而本文所提的 I-DRB 由秘密共享作为子方案，仅需秘密分发者随机选取秘密值并进行分发，无需公私钥参与随机数生成过程（除签名外），因此不依赖复杂的分布式密钥生成协议. 本节给出由秘密共享到 I-DRB 的通用构造，其示意图如图 1 所示.

除初始化算法外，每生成一次随机数共需要经过 6 个过程，分别是节点选取、秘密共享、秘密恢复、随机数计算、随机数验证以及状态更新. 节点选取算法可以采用轮流选取方式或根据上一轮随机数选取方式（前者是指根据节点下标顺序每轮依次选取，后者是指根据上一轮随机数结果取余得到本轮选中的节点下标，不同实例化方案使用不同的节点选取方式）. 选取的节点作为秘密分发者向 n 个节点分发秘密份额，如图 1 虚线框为从 n 个节点中选取 m 个节点作为秘密分发者. 在秘密共享算法中，上述选取的秘密分发者根据具体的秘密共享方案生成相应秘密份额.

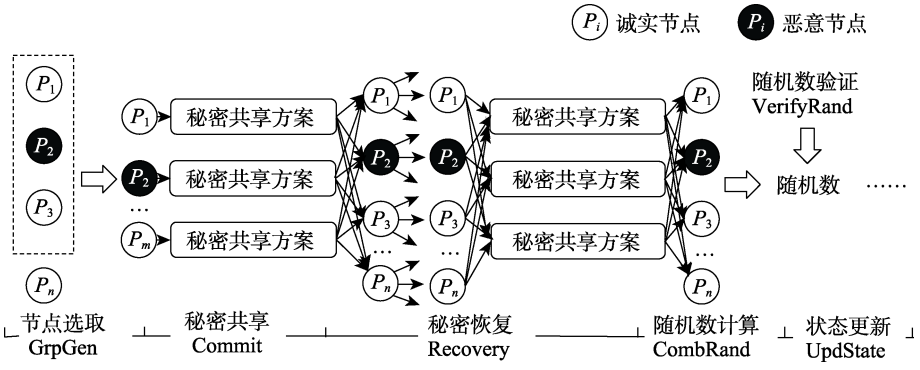


图 1 I-DRB 通用构造示意图

所有诚实节点收到一致来源的有效秘密份额后，可以对达成共识的所有秘密进行恢复，秘密共享方案保证任意少于门限值个秘密份额无法恢复原秘密值，从而保证恶意节点无法预测随机数. 在恢复完原秘密值后，所有节点计算最终随机数值，且任意外部验证者可以验证该随机数输出，若验证通过，则每个节点可以开始生成下一次随机数.

为了简化协议的符号表示，需要说明的是每一次消息的发送者同样也作为消息的接收者，即秘密分发者也会给自己分发秘密份额.

令网络中总节点数为 n，节点集合为 $\mathcal{N}=\{P_1,\cdots,P_n\}$，其中有 f 个恶意节点. I-DRB 方案 $\Pi_{\text{I-DRB}}=(\text{Init},\text{GrpGen},\text{Commit},\text{Recovery},\text{CombRand},\text{VerifyRand},\text{UpdState})$ 由秘密共享方案 $\Pi_{\text{SS}}=(\text{Setup},\text{Share},\text{Rec-}$

onstruct, Check) 和共识 Consensus(s_i^r, r) 组成，其中 r 为当前轮数，$\Pi_{\text{I-DRB}}$ 的详细描述如下.

- Init($1^\kappa, n, f$) $\to GP$：这是初始化算法. 输入安全参数 1^κ、总节点数 n 和恶意节点数 f，执行 Setup($1^\kappa, f$) $\to pp$，输出公共参数 $GP = pp$.
- GrpGen(GP, st_{r-1}, n, f) $\to \mathcal{P}^r$：这是节点选取算法. 输入公共参数 GP、状态 st_{r-1}、总节点数 n 和恶意节点数 f，选取当前轮 r 下进行秘密分发的节点子集 $\mathcal{P}^r \subseteq \mathcal{N}$，输出节点子集 $\mathcal{P}^r$.
- Commit(GP, st_{r-1}, i) $\to \hat{s}_i^r$：这是秘密共享算法. 输入公共参数 GP、状态 st_{r-1} 和节点 P_i，其中 $i \in \mathcal{P}^r$，节点 P_i 随机选取一个当前轮 r 的秘密值 s_i^r，执行 Share(s_i^r) $\to (s_{i,1}^r, \cdots, s_{i,n}^r)$. 令 $\hat{s}_i^r = (s_{i,1}^r, \cdots, s_{i,n}^r)$ 并输出.
- Recovery($GP, st_{r-1}, \hat{s}_i^{r*}$) $\to \mathcal{S}^r$：这是秘密恢复算法. 输入公共参数 GP、状态 st_{r-1} 和当前轮 r 某个秘密值对应的秘密份额集合 $\hat{s}_i^{r*} = \{\hat{s}_{i,j}^r\}_{j \in \mathcal{J}}$，其中 $f < |\mathcal{J}| \leqslant n$，每个节点执行 Reconstruct($\hat{s}_i^{r*}$) $\to s_i^r$ 恢复出原秘密值 s_i^r. 所有节点执行 Consensus(s_i^r, r) $\to \mathcal{S}^r$ 对相同秘密值集合达成共识，输出共识后的集合 $\mathcal{S}^r$.
- CombRand($GP, st_{r-1}, \mathcal{S}^r$) $\to R_r$：这是随机数计算算法. 输入公共参数 GP、状态 st_{r-1} 和已恢复的秘密值集合 $\mathcal{S}^r = \{s_i^r\}_{i \in \mathcal{P}^r}$，最终每个节点拥有相同的秘密值集合，输出 r 轮的随机数 R_r.
- VerifyRand(GP, st_{r-1}, R_r, L) $\to 1/0$：这是随机数验证算法. 输入公共参数 GP、状态 st_{r-1}、当前轮 r 的随机数 R_r 和验证辅助信息 L，其中 L 包括所有验证所需信息. 若 Check $(R_r, r, L) = 1$，则输出 1，否则输出 0.
- UpdState(GP, st_{r-1}, R_r) $\to st_r$：这是状态更新算法. 输入公共参数 GP、状态 st_{r-1} 和当前轮 r 的随机数 R_r，表示当前轮已完成随机数生成，因此状态更新为 st_r 并输出.

4.2　安全模型

为了便于给出安全目标定义，本文首先定义一系列谕言机（如算法 5.1–算法 5.3 所示），其中 rn 表示当前轮数，st 表示当前状态.

算法 5.1 用于验证恶意节点集合 C 的合法性并初始化全局变量 rn 和 st.

算法 5.2 用于更新全局变量 rn 和 st，首先验证上一轮随机数的有效性，若验证通过则更新状态变量和轮数.

算法 5.3 用于计算节点 P_i 向节点 P_j 分发的份额 $S_{i,j}$，其中 $\mathcal{P}$ 为选取的秘密分发者集合.

算法 5.1　谕言机 $\mathcal{O}^{\text{Init}(C)}$

输入：恶意节点集合 C

输出：1 或 $\perp$

1. 若 $C \subsetneq \mathcal{N}$ 或 $|C| > f$，返回 $\perp$；
2. $GP \leftarrow \text{Init}(1^\kappa, n, f)$；
3. $rn = 0, st = st_0$；
4. 返回 1.

算法 5.2　谕言机 $\mathcal{O}^{\text{Update}(R)}$

输入：随机数 R

输出：(rn, st)

1. 若 VerifyRand$(GP, st, R, L) = 0$，返回 $\perp$；
2. $x \leftarrow \text{UpdState}(GP, st, R)$；
3. $rn = rn + 1, st = x$；
4. 返回 (rn, st).

算法 5.3　谕言机 $\mathcal{O}^{\text{Share}(i,j)}$

输入：节点序号 i, j

输出：份额 $\hat{s}_{i,j}$

1. 若 $i \notin \mathcal{P}$ 或 $j \in C$，返回 $\perp$；
2. $\hat{s}_i \leftarrow \text{Commit}(GP, st, i)$；
3. 解析 $\hat{s}_i = (\hat{s}_{i,1}, \cdots, \hat{s}_{i,j}, \cdots, \hat{s}_{i,n})$；
4. 返回 $\hat{s}_{i,j}$.

David 等人[29]给出了 NI-DRB 方案的安全目标，分别是伪随机性、唯一性、鲁棒性. 本文提出的 I-DRB 方案也具备上述性质，但谕言机定义（算法 5.1-算法 5.3）和游戏定义与 David 等人的方案不同. 需要说明的是，现有方案[3-5, 7-9]中的随机数性质大多定义为可用性、不可预测性和抗偏置性，而此处的鲁棒性即可用性，伪随机性含有不可预测性的意义，另外，伪随机性与唯一性结合可以推出抗偏置性.

下面将介绍 I-DRB 伪随机性、唯一性、鲁棒性的形式化定义.

首先是伪随机性，该性质保证随机数输出与均匀分布的随机数之间是不可区分的.

定义 1.（I-DRB 伪随机性）　令网络中总节点数为 n，节点集合为 $\mathcal{N} = \{P_1, \cdots, P_n\}$，其中有 f 个恶意节点. 如果对于任意概率多项式时间内的适应性敌手 $\mathcal{A}$，存在可忽略函数 negl(·) 使得

$$\begin{aligned}\text{Adv}_{\Pi,\mathcal{A}}^{\text{PRand}}(\kappa) =& |\Pr[\text{PRand}_{\Pi,\mathcal{A}}(\kappa, 0) = 1] \\ &- \Pr[\text{PRand}_{\Pi,\mathcal{A}}(\kappa, 1) = 1]| \\ \leqslant& \text{negl}(\kappa)\end{aligned}$$

成立，那么 I-DRB 方案是 (f, n)-伪随机的，其中

$\mathrm{PRand}_{\Pi,\mathcal{A}}(\kappa,b)$，$b\in\{0,1\}$ 游戏定义如下.

- 腐化阶段：敌手 $\mathcal{A}$ 选择恶意节点集合 C，且 $|C|\leqslant f$.
- 初始化：挑战者 $\mathcal{C}$ 执行谕言机 $\mathcal{O}^{\mathrm{Init}(C)}$.
- 问询 1：敌手 $\mathcal{A}$ 询问谕言机 $\mathcal{O}^{\mathrm{Share}(i,j)}$ 和 $\mathcal{O}^{\mathrm{Update}(R)}$ 诚实节点 $P_{j\wedge}\in\mathcal{P}\setminus C$ 收到的对于某秘密 s_i 的秘密份额值 $s_{i,j}$.
- 挑战：敌手 $\mathcal{A}$ 向挑战者发送集合 U 满足 $|U|>f$，以及相应的份额集合 $\{\hat{s}_{i,j}\}_{i\in\mathcal{P},j\in U\cap C}$. 挑战者 $\mathcal{C}$ 计算 $\hat{s}_i\leftarrow\mathrm{Commit}(GP,st^*,i)$，得到 $\hat{s}_i'=\{\hat{s}_{i,j}\}_{i\in\mathcal{P},j\in U\setminus C}$. 挑战者 $\mathcal{C}$ 恢复秘密 $s_i\leftarrow\mathrm{Recovery}(GP,st^*,\hat{s}_i')$，其中 $i\in\mathcal{P}$. 令 $R^*\leftarrow\mathrm{CombRand}(GP,st^*,\{s_i\}_{i\in\mathcal{P}})$，若 $b=0$，则令 $\delta=R^*$，若 $b=1$，则令 δ 为随机值. 挑战者 $\mathcal{C}$ 进入下一轮 $rn=rn^*+1$，并更新状态 $st\leftarrow\mathrm{UpdState}(GP,st^*,\delta)$. 挑战者 $\mathcal{C}$ 发送 $(\delta,\ rn,st)$ 给敌手 $\mathcal{A}$.
- 问询 2：敌手 $\mathcal{A}$ 继续询问谕言机 $\mathcal{O}^{\mathrm{Share}(i,j)}$ 和 $\mathcal{O}^{\mathrm{Update}(R)}$ 诚实节点 $P_{j\wedge}\in\mathcal{P}\setminus C$ 收到的对于某秘密 s_i 的秘密份额值 $s_{i,j}$.
- 猜测：敌手 $\mathcal{A}$ 输出比特 b'.

其次是唯一性，该性质含义为在某轮 r 内，敌手无法输出两个不同的且均通过验证的随机数，即能保证每轮的随机数输出都是唯一的.

定义 2.（I-DRB 唯一性） 令总节点数为 n，节点集合为 $\mathcal{N}=\{P_1,\cdots,P_n\}$，其中有 f 个恶意节点. 如果对于任意概率多项式时间内的敌手 $\mathcal{A}$，存在可忽略函数 negl(·) 使得

$$\mathrm{Adv}^{\mathrm{Unique}}_{\Pi,\mathcal{A}}(\kappa)=\Pr[\mathrm{Unique}_{\Pi,\mathcal{A}}(\kappa)=1]\leqslant\mathrm{negl}(\kappa)$$

成立，那么 I-DRB 满足唯一性，其中 $\mathrm{Unique}_{\Pi,\mathcal{A}}(\kappa)$ 游戏定义如下.

- 腐化阶段：敌手 $\mathcal{A}$ 选择恶意节点集合 C，且 $|C|\leqslant f$.
- 初始化：挑战者 $\mathcal{C}$ 执行谕言机 $\mathcal{O}^{\mathrm{Init}(C)}$.
- 问询：敌手 $\mathcal{A}$ 询问谕言机 $\mathcal{O}^{\mathrm{Share}(i,j)}$ 和 $\mathcal{O}^{\mathrm{Update}(R)}$ 诚实节点 $P_j\in\mathcal{P}\setminus C$ 收到的对于某秘密 s_i 的秘密份额值 $\hat{s}_{i,j}$，且不同的敌手所收到的问询结果是一致的.
- 挑战：敌手向挑战者 $\mathcal{C}$ 发送两个随机数 R 和 R'，若 $R\neq R'$ 且 $\mathrm{VerifyRand}(GP,st^*,R,L)=\mathrm{VerifyRand}(GP,st^*,R',L)=1$，则输出 1，否则输出 0.

最后定义鲁棒性，该性质指虽然存在恶意敌手，但若能输出随机数 R，则 R 必定是正确的.

定义 3.（I-DRB 鲁棒性） 令总节点数为 n，节点集合为 $\mathcal{N}=\{P_1,\cdots,P_n\}$，其中有 f 个恶意节点. 如果对于任意概率多项式时间内的敌手 $\mathcal{A}$，存在可忽略函数 negl(·) 使得

$$\mathrm{Adv}^{\mathrm{Robust}}_{\Pi,\mathcal{A}}(\kappa)=\Pr[\mathrm{Robust}_{\Pi,\mathcal{A}}(\kappa)=1]\leqslant\mathrm{negl}(\kappa)$$

成立，那么 I-DRB 方案是满足鲁棒性的，其中 $\mathrm{Robust}_{\Pi,\mathcal{A}}(\kappa)$ 游戏定义如下.

- 腐化阶段：敌手 $\mathcal{A}$ 选择恶意节点集合 C，且 $|C|\leqslant f$.
- 初始化：挑战者 $\mathcal{C}$ 执行谕言机 $\mathcal{O}^{\mathrm{Init}(C)}$.
- 问询：敌手 $\mathcal{A}$ 询问谕言机 $\mathcal{O}^{\mathrm{Share}(i,j)}$ 和 $\mathcal{O}^{\mathrm{Update}(R)}$ 诚实节点 $P_j\in\mathcal{P}\setminus C$ 收到的对于某秘密 s_i 的秘密份额值 $\hat{s}_{i,j}$，且问询一定会收到应答.
- 挑战：敌手向挑战者发送集合 U 满足 $|U|>f$，以及相应的份额集合 $\{\hat{s}_{i,j}\}_{i\in\mathcal{P},j\in U\cap C}$. 挑战者计算 $\hat{s}_i\leftarrow\mathrm{Commit}(GP,st^*,i)$，得到 $\hat{s}_i'=\{\hat{s}_{i,j}\}_{i\in\mathcal{P},j\in U\setminus C}$. 挑战者恢复秘密 $s_i\leftarrow\mathrm{Recovery}(GP,st^*,\hat{s}_i')$，其中 $i\in\mathcal{P}$. 令 $R^*\leftarrow\mathrm{CombRand}\left(GP,st^*,\{s_i\}_{i\in\mathcal{P}}\right)$，若 $R^*\neq\perp$ 且 $\mathrm{VerifyRand}(GP,st^*,R^*,L)=0$，则输出 1，否则输出 0.

5 基于异步可验证秘密共享的分布式随机数生成方案

在第四章，本文提出了交互的分布式随机数生成 I-DRB. 该通用构造兼容了所有基于秘密共享的方案共性，由秘密共享和共识组成，是一个通用模型，为未来对于基于秘密共享的方案研究提供了指导意义. 然而对于具体方案而言，现有基于秘密共享的分布式随机数生成方案通信复杂度均较高，而少部分通信复杂度较低的方案均对安全性做出了牺牲. 除此之外，基于秘密共享的方案大多数均为同步网络假设，而极少部分面向异步网络的方案通信复杂度高达 $O(n^3)$ 或 $O(n^4)$. 考虑到具体方案依赖于不同秘密共享和共识方案的选取，因此针对同步网络模型假设强和通信复杂度较高的关键性问题，本节采用异步可验证秘密共享方案解决网络模型的局限性，使用常数级别的 KZG 承诺降低通信复杂度，并设计共识流程保证节点对接收消息达成一致. 第 5.1 节给出方案的具体算法描述，包括节点选取算法、秘密共享算法、秘密恢复算法、随机数计算算法、随机数验证算法和状态更新算法. 第 5.2 节证明所设计方案的伪随机性、唯一性和鲁棒性. 第 5.3 节分析所设计方案的通信复杂度和计算复杂度.

5.1　具体构造

本节所设计方案的目的是持续提供满足伪随机性、唯一性和鲁棒性的随机数，并且主要面向总节点数固定的许可化环境. 基于异步可验证秘密共享的分布式随机数生成方案如图 2 所示.

方案实例被标识为一个 ID 串 $ID.d_i$，也叫做标签（tag），本文为简单起见将 $ID.d_i$ 取值范围定义为 $[1,n]$. 所有消息均表示为 $(ID,\ldots)$，其中 ID 即为上文所说的协议标签. 输入动作的消息表示为 $(ID,\text{in},type,\ldots)$，输出动作的消息表示为 $(ID,\text{out},type,\ldots)$，协议消息表示为 $(ID,type,\ldots)_i$，即使用私钥 sk_i 对消息 $(ID,type,\ldots)$ 的签名，其中 $type$ 与协议具体实现有关，而共识阶段发送的协议消息仅为对多个 ID 进行签名.

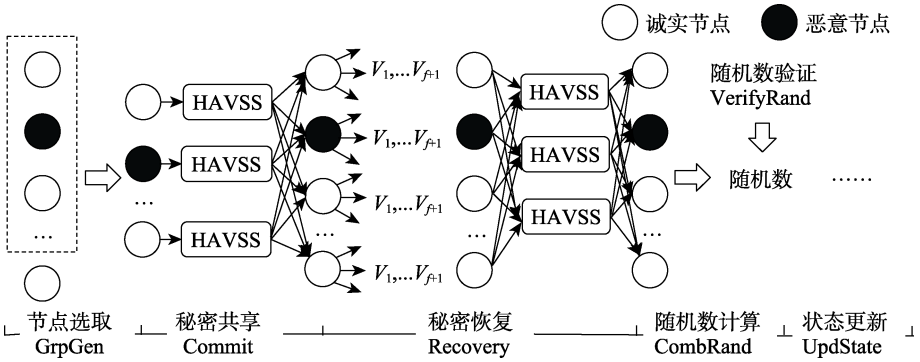


图 2　分布式随机数生成方案整体示意图

为了简化方案的符号表示，需要说明的是每一次消息的发送者同样也作为消息的接收者，即秘密分发者也会给自己分发秘密份额. 协议描述中的基本符号说明如表 1 所示，若不指明具体节点，表内符号不加下标 i，且表内符号未在上标处标识当前轮数 r.

表 1　基本符号含义表

符号	含义
R_i	节点 P_i 生成的恢复多项式
$\hat{R}_i$	节点 P_i 生成的恢复多项式的承诺
$S_{i,j}$	节点 P_i 生成的分享多项式（n 个）
$\hat{S}_{i,j}$	节点 P_i 生成的分享多项式的承诺（n 个）
C_i	节点 P_i 生成的根承诺
$T_{i,j}$	节点 P_i 生成的测试多项式（n 个）
$\hat{T}_{i,j}$	节点 P_i 生成的测试多项式的承诺（n 个）
$C_i[\cdot]$	节点 P_i 的标签集合计数器
V	标签集合候选集
$V_1^i,\cdots,V_{f+1}^i$	节点 P_i 发送的标签集合
R_r	第 r 个随机数

令网络中总节点数为 n，节点集合为 $\mathcal{N}=\{P_1,\cdots,P_n\}$，其中有 f 个恶意节点，满足 $n=3f+1$，p 为秘密恢复门限值，满足 $p=n-f$. 基于 HAVSS 方案的分布式随机数生成方案 $\Pi_{\text{HAVSS-IDRB}}=(\text{Init},\text{GrpGen},\text{Commit},\text{Recovery},\text{CombRand},\text{VerifyRand},\text{UpdState})$ 由 HAVSS 方案 $\Pi_{\text{HAVSS}}=(\text{Setup},\text{Share},\text{VerifyShare},\text{VerifyEcho},\text{RcvShare},\text{RcvSecret})$ 和共识 $\text{Consensus}(\{V_i\}_{i\in\mathcal{P}^r},r)$ 组成，从而保证每个节点是对同一组标签集合达成共识，$\Pi_{\text{HAVSS-IDRB}}$ 的详细描述如下.

（1）初始化算法 $\text{Init}(1^\kappa,n,f)\to GP$

输入安全参数 κ 的一元表示、总节点数 n 和恶意节点数 f，执行 $\text{Setup}(1^\kappa,f)\to pp$，输出公共参数 $GP=pp$.

（2）节点选取算法 $\text{GrpGen}(GP,st_{r-1},n,f)\to\mathcal{P}^r$

输入公共参数 GP、状态 st_{r-1}、总节点数 n 和恶意节点数 f，选取当前轮 r 下进行秘密分发的节点子集 $\mathcal{P}^r\subseteq\mathcal{N}$，其中 $|\mathcal{P}^r|=2f+1$，输出节点子集 $\mathcal{P}^r$.

在每次生成随机数开始时，需要选取 $2f+1$ 个节点作为本次的 HAVSS 秘密份额分发者，本文使用轮流选取的方式. 令 P_i 是参与节点列表中的第 i 个节点（下标从 1 开始），那么每一次生成随机数 $R_r(r\geqslant 1)$ 时的秘密分发者 $D_a,D_{a+1},\cdots,D_{a+2f}$ 定义为

$$D_a,\cdots,D_{a+2f}:=P_{1+(r-1)(2f+1)\bmod n},\cdots,P_{r(2f+1)\bmod n}$$

（3）秘密共享算法 $\text{Commit}(GP,st_{r-1},i)\to\hat{s}_i^r$

输入公共参数 GP、状态 st_{r-1} 和节点 P_i，其中 $i\in\mathcal{P}^r$，输出秘密份额 $\hat{s}_i^r$.

在该算法中，$2f+1$ 个秘密分发者 $D_a,D_{a+1},\cdots,D_{a+2f}$ 执行输入 $(ID.d_i,\text{in},\text{share},s_i)$，各自随机选取一个秘密值 s_i，其中 $a\leqslant i\leqslant a+2f$，$ID.d_i$ 表

示每个秘密分发者的标签（tag），s_i 为每个秘密分发者随机选取的秘密值. 然后秘密分发者 D_i 执行 3.2 节中算法 1，并向每个节点 $P_j(1\leqslant j\leqslant n)$ 发送 send 消息 $\left(ID.d_i,\text{send},set_j\right)_i$，其中 set_j 包括 D_i 选取的多项式 R_i^r 的承诺 $\hat{R}_{i,1}^r$、D_i 选取的 n 个多项式 $S_{i,m}^r$ 的承诺 $\hat{S}_i^r=(\hat{S}_{i,1}^r,\cdots,\hat{S}_{i,n}^r)$、对 n 个多项式承诺 $\hat{S}_{i,m}^r$ 和多项式承诺 $\hat{R}_i^r$ 的根承诺 C_i^r、每个多项式 $S_{i,m}^r$ 上的点值计算 $y_{i,j}^r=(S_{i,1}^r(j),\cdots,S_{i,n}^r(j))$ 和每个多项式 $T_{i,j}^r$ 上的点值计算 $y_i^r=(T_{i,1}^r(1),\cdots T_{i,j}^r(j),\cdots,T_{i,n}^r(n))$.

当节点 $P_j(1\leqslant j\leqslant n)$ 第一次收到秘密分发者发送的 send 消息后，立即执行 3.2 节中的算法 2.1，验证通过后，向其他节点 $P_t(1\leqslant t\leqslant n)$ 发送 echo 消息 $(ID.d_i,\text{echo},info_{j,t})_j$，其中 $info_{j,t}$ 包括对 n 个多项式 $S_{i,m}^r$ 和 R_i^r 的根承诺 C_i^r、多项式 $S_{i,t}^r$ 的承诺 $\hat{S}_{i,t}^r$ 和多项式 $S_{i,t}^r$ 上的点值计算 $y_{i,t}^r=S_{i,t}^r(j)$，该值助节点 P_t 恢复自己的多项式.

当节点 $P_t(1\leqslant t\leqslant n)$ 第一次收到其他节点 $P_j(1\leqslant j\leqslant n)$ 发送的 echo 消息后，执行 3.2 节中的算法 2.2，验证通过后向其他节点 $(1\leqslant k\leqslant n)$ 发送 ready 消息 $\left(ID.d_i,\text{ready},C_i\right)_t$. 在异步网络环境下，可能存在已经收到 ready 消息但还没收到 echo 消息的情况，因此当节点 $P_t(1\leqslant t\leqslant n)$ 第一次收到其他节点 $P_k(1\leqslant k\leqslant n)$ 发送的 ready 消息后，若之前没发送过 ready 消息，且收到 $f+1$ 个 ready 消息，则向其他节点发送 ready 消息 $\left(ID.d_i,\text{ready},C_i\right)_t$.

当节点收到 $2f+1$ 个关于同一个根承诺 C_i 的 ready 消息后，等待 $f+1$ 个对该根承诺 C_i 的 echo 消息到达，执行 3.2 节中的算法 3，之后执行输出 $\left(ID.d_i,\text{out},\text{shared}\right)$.

（4）秘密恢复算法 $\text{Recovery}(GP,st_{r-1},\hat{s}_i^{r*})\rightarrow s_i^r$

输入公共参数 GP、状态 st_{r-1} 和当前轮 r 某个秘密值对应的秘密份额集合 $\hat{s}_i^{r*}=\{\hat{s}_{i,j}^r\}_{j\in\mathcal{J}}$，其中 $f<|\mathcal{J}|\leqslant n$，恢复并输出原秘密值 s_i^r.

考虑到异步网络中，只能保证诚实节点的消息最终到达各节点，但具体的消息送达时间上限不确定，因此各节点间需要进行共识来确保每个诚实节点最终所收到的份额来源是一致的. 传统的异步共识需要使用异步二元共识，而异步二元共识中需要公共随机源（Common Coin），会导致循环使用随机数的问题，这是异步网络下分布式随机数生成方案的挑战. 若不使用异步二元共识，则需要重新设计适用于异步随机数生成的共识流程.

本方案参考投票共识的思想，将共识流程表示为 $\text{Consensus}(\{V_1^k,\ldots,V_{f+1}^k\}_{k\in\mathcal{P}^r},r)$，每个节点 P_k 本地维护着标签集合计数器 $C_k[\cdot]$ 和标签候选集 V，且节点每次发送的投票表示为 $V_1^k,\cdots,V_{f+1}^k$. 候选集 V 和计数器 $C_k[\cdot]$ 初始化为空. 当节点 P_k 输出 $\left(ID.d_i,\text{out},\text{shared}\right)$ 后，说明该节点已经收到标签为 $ID.d_i$ 的秘密分发者所发送的有效秘密份额，因此节点 P_k 将 $ID.d_i$ 加入到集合 V_1^k 中，继续等待不同标签的秘密份额到达. 当集合大小 $\left|V_1^k\right|=f+1$ 时，将该集合广播给全网其他节点. 若后续新收到了来自于其他秘密分发者的份额，则将该新标签加入到集合中，记为集合 $V_2^k(|V_2^k|=f+2)$ 并广播，以此类推，后续每次收到新的份额便广播新的标签集合. 因此，若 $2f+1$ 个秘密分发者均为诚实节点，那么一共会发送 $f+1$ 个标签集合，且最后一个集合 V_{f+1}^k 的大小为 $2f+1$. 图 3 所示为以节点 P_1 为例的上述秘密份额接收和标签集合发送流程（即 $k=1$）.

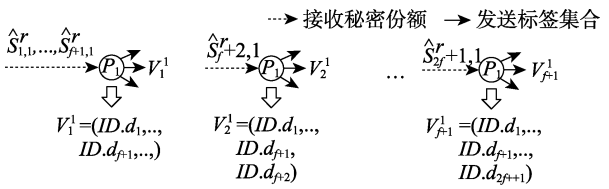


图 3 标签集合发送过程示意图

需要说明的是，收到 $f+1$ 个份额才发送第一个集合 V_1^k 的原因是，这是一个下限，只有收到 $f+1$ 个份额，才能说明其中至少有一个是诚实节点发送的，这样才能保证后续随机数计算结果的伪随机性、抗偏置性以及不可预测性. 另外，因为一共有 $2f+1$ 个秘密分发者，但恶意节点可能永远不发送份额，因此每个诚实节点最少广播 1 次集合，最多广播 $f+1$ 次不同的集合.

节点 P_j 每收到其他节点 P_k 发送的集合 V_i^k，首先判断 V_i^k 是否包含 P_k 之前所发送的集合 $V_i^{k'}$ 和候选集 V 的并集，若包含，则令 $V_i\leftarrow V_i^k$ 并将计数器 $C_j[V_i]$ 自增 1，否则直接丢弃该消息. 若某一时刻计数器 $C_j[V_i]=2f+1$，表示该节点已收到 $2f+1$ 条相同的标签集合 V_i，则将 V_i 中元素加入到候选集 V 中，并等待相应的份额到达.

令诚实节点最后一次广播的集合为 V_{final}，那么每个诚实节点的 V_{final} 一定相同，最终候选集 V 中的元素即为 V_{final} 的元素，因此所有诚实节点可以对份额来源达成共识.

当收到 $2f+1$ 条相同的 V_{final} 后，节点 P_j 将候选集 V 更新，并（1）等待相应的份额到达；（2）若份额全部到达，则执行输入 $\left(ID.d_i,\text{in},\text{reconstruct}\right)$，其中 $ID.d_i$ 应为 V_{final} 中的元素，之后开启秘密恢复

过程. 既然该诚实节点能发起秘密恢复过程，说明该节点 P_j 知道相应的分享多项式 $S_{i,j}$ 及其与根承诺 C_i 相应的证据. 因此，P_j 发送消息 $(ID.d_i, \text{reconstruct\_share}, shares_j)_j$，其中 $shares_j$ 包括多项式 $S_{i,j}^r$ 上的点值计算 $S_{i,j}^r(j)$ 和多项式 $S_{i,j}^r$ 的承诺 $\hat{S}_{i,j}^r$.

当诚实节点收到其他节点发送的 $(ID.d_i, \text{reconstruct\_share}, shares_j)_j$ 消息后，执行 3.2 节算法 4，收到 $p+1$ 个有效的秘密份额后得到集合 $\hat{s}_i^{r*}$，即可通过拉格朗日差值恢复原秘密值 s_i^r，最终输出 $(ID.d_i, \text{out}, \text{reconstructed}, s_i^r)$.

若在秘密恢复过程中，收到了延迟到达的有效的标签集合，则节点需保护现场、执行中断再恢复原执行流程. 图 4 所示为以节点 P_1 为例的秘密恢复及共识中断过程示意图. 假设节点 P_1 目前收到了 $2f+1$ 个标签集合 V_{f+1} 和 $2f$ 个标签集合 V_{final}，更新本地标签集合计数器 $C_1[V_{f+1}]$ 和 $C_1[V_{final}]$，因为前者满足 $2f+1$ 的阈值，因此将 V_{f+1} 中元素加入到候选集 V 中，等待对应的份额到达，若份额到达，则可开启秘密恢复流程.

在秘密恢复流程中，若收到了延迟到达的 V_{final}，保护当前秘密恢复的执行现场，保护的信息包括已恢复的秘密标签 $ID.d_i$ 及秘密值、正在恢复的 $p+1$ 个秘密份额及多项式承诺. 转而执行共识中断过程，更新本地标签集合计数器 $C_1[V_{final}]=2f+1$，满足阈值并更新候选集. 执行完毕后，恢复原执行现场并继续秘密恢复流程. 考虑到延迟到达的有效标签集合一定包含原候选集，因此原秘密恢复流程不受影响，仅需后续启动关于新增加标签的秘密恢复即可.

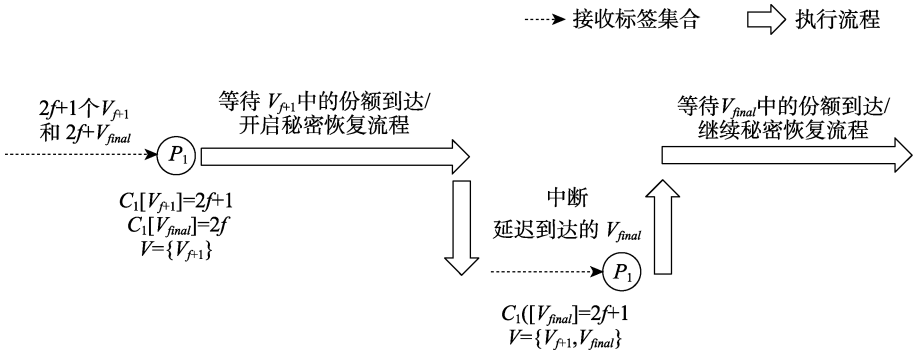


图 4　秘密恢复及共识中断过程示意图

（5）随机数计算算法 $\text{CombRand}(GP, st_{r-1}, \mathcal{S}^r) \to R_r$

输入公共参数 GP、状态 st_{r-1} 和已恢复的秘密值集合 $\mathcal{S}^r=\{s_i^r\}_{i\in\mathcal{P}^r}$，最终每个节点拥有相同的秘密值集合，输出当前轮 r 的随机数 R_r.

若 V_{final} 中所有元素所对应的秘密 s_i 均被恢复，则各节点本地计算最终的随机数值 $R_r=s_1\oplus s_2\oplus\ldots\oplus s_{2f+1}$.

（6）随机数验证算法 $\text{VerifyRand}(GP, st_{r-1}, R_r, L)\to 1/0$

输入公共参数 GP、状态 st_{r-1}、当前轮 r 的随机数 R_r 和验证辅助信息 L，其中 L 包括相关点值计算结果和承诺. 若 R_r 是正确的，则输出 1，否则输出 0.

需要说明的是，这里的验证算法指的是外部验证者对随机数的验证. 其次，I-DRB 方案的 VerifyRand 算法与 NI-DRB 方案的不同，因为后者需要在初始化过程中生成公私钥，且随机数是关于私钥的确定性函数计算结果，验证时只需通过公钥或验证密钥进行验证. 相反，前者无需密钥生成初始化过程，因此随机数结果需要借助与节点间交互的数据记录.

因此验证辅助信息 L 包括每个点值计算结果 $R_i^r(1),\cdots,R_i^r(n)$ 及其对应的承诺 $\hat{S}_{i,j}^r$，其中 $i\in\mathcal{P}^r$，$j\in[1,n]$. 验证每个点值计算是否正确，若验证通过，即可恢复出原秘密值，并异或得到最终随机数，验证完毕.

（7）状态更新算法 $\text{UpdState}(GP, st_{r-1}, R_r)\to st_r$

输入公共参数 GP、状态 st_{r-1} 和当前轮 r 的随机数 R_r，表示当前轮已完成随机数生成过程，因此状态更新为 st_r 并输出.

在随机数验证通过后，等待新的输入 $(ID.d_i, \text{in}, \text{share}, s_i)$ 发生后，更新状态 $st=st+1$，以及轮数 $rn=rn+1$.

需要说明的是，本方案虽然降低了通信复杂度，但因为使用了 KZG 承诺，需要可信初始化操作. 另外，本方案设计的共识流程使得所有节点在 f 轮之后才能达成共识，因此增加了延时.

5.2 安全性分析

在本节中，使用可证明安全方法证明本章提出的方案的安全性，包括伪随机性、唯一性和鲁棒性的证明. 假设秘密共享方案 Π_{HAVSS} 和共识 Consensus (s_i^r, r) 满足其自身安全目标，且每轮有至少一个诚实节点执行秘密共享算法.

5.2.1 伪随机性

对于 I-DRB 而言，若秘密共享方案具有隐私性，则能够保证任意不超过门限值个秘密份额无法构造原秘密值，也就是说秘密份额是均匀随机分布的，除非敌手能够区分秘密份额和和均匀分布随机值，否则敌手无法得到关于秘密的任何消息. 另外，每轮需有至少一个诚实节点执行秘密共享算法，这样才能保证 I-DRB 方案的伪随机性. 因此对于本章的具体构造而言，根据 HAVSS 的隐私性，可以证明基于 HAVSS 的分布式随机数生成方案 HAVSS-IDRB 具有伪随机性. 引理 1 在文献[26]中证明，这里只给出其定义.

引理 1.（HAVSS 隐私性[26]） 若秘密分发者分发标签为 *ID.d* 的秘密 s，并且最多 $p-f$ 个诚实节点已开启标签为 *ID.d* 的秘密恢复流程，则敌手无法获取关于 s 的任何信息.

定理 4. 若 Π_{HAVSS} 方案具有隐私性，且有至少一个诚实节点执行秘密共享算法，则 $\Pi_{\text{HAVSS-IDRB}}$ 具有伪随机性.

证明. 使用反证法，假设存在一个概率多项式时间敌手 $\mathcal{A}$ 以不可忽略的概率攻破 $\Pi_{\text{HAVSS-IDRB}}$ 方案伪随机性，则构造一个概率多项式时间敌手 $\mathcal{B}$. 攻破 Π_{HAVSS} 方案隐私性，即 $\mathcal{B}$ 同时作为 $\Pi_{\text{HAVSS-IDRB}}$ 方案伪随机性挑战者回应 $\mathcal{A}$ 的问询.

$\mathcal{A}$ 先选择恶意节点集合 C，且 $|C|\leqslant f$，$\mathcal{B}$ 将 C 发送给 HAVSS 隐私性挑战者 $\mathcal{C}$. $\mathcal{B}$ 和 $\mathcal{C}$ 各自执行初始化过程得到公开参数. $\mathcal{A}$ 向 $\mathcal{B}$ 发起 $\mathcal{O}^{\text{Update}(\cdot)}$ 问询，$\mathcal{B}$ 可以正常应答，因为仅需要公开参数. $\mathcal{A}$ 向 $\mathcal{B}$ 发起 $\mathcal{O}^{\text{Share}(\cdot,\cdot)}$ 问询，$\mathcal{B}$ 可以向 $\mathcal{C}$ 询问对于某个秘密值的份额并且把结果返回给 $\mathcal{A}$.

在挑战阶段，$\mathcal{B}$ 收到 $\mathcal{A}$ 发送的 $\{\hat{s}_{i,j}\}_{i\in\mathcal{P},j\in U\cap C}$，$\mathcal{B}$ 进行以下应答：（1）$\mathcal{B}$ 向挑战者 $\mathcal{C}$ 询问 $(st^*, \{\hat{s}_{i,j}\}_{i\in\mathcal{P},j\in U\cap C})$，得到应答值 δ；（2）$\mathcal{B}$ 使用该应答值 δ 更新当前状态 $st\leftarrow \text{UpdState}(GP, st^*, \delta)$，并增加当前轮数 $rn = rn^* + 1$，$\mathcal{B}$ 发送 (δ, rn, st) 给敌手 $\mathcal{A}$. $\mathcal{A}$ 可以继续执行问询. 最终，若 $\mathcal{A}$ 输出 b'，则 $\mathcal{B}$ 也输出 b'.

综上，$\mathcal{B}$ 向 $\mathcal{A}$ 模拟了真实攻击环境，因此 $\text{Adv}^{\text{PRand}}_{\Pi_{\text{HAVSS-IDRB}},\mathcal{A}}(\kappa) = \text{Adv}^{\text{Pri}}_{\Pi_{\text{HASSS}},\mathcal{B}}(\kappa)$，进而 $\mathcal{B}$ 攻破了 Π_{HAVSS} 方案，但是 Π_{HAVSS} 方案具备隐私性，产生矛盾，所以不存在成功攻击 $\Pi_{\text{HAVSS-IDRB}}$ 方案伪随机性的敌手 $\mathcal{A}$.

证毕.

5.2.2 唯一性

对于 I-DRB 而言，若秘密共享方案具有正确性，也就是说秘密分发者和诚实节点都是遵守协议流程，那么算法中的所有验证都通过，因此可以根据秘密恢复算法从有效秘密份额中恢复原秘密值，即每个节点收到任意超过门限值个秘密份额，那么节点均可恢复出原秘密值. 另外，共识的一致性保证所有诚实节点能够得到相同来源的秘密值集合. 而在异步网络中，唯一性弱化为最终唯一性. 因此对于本章的具体构造而言，基于 HAVSS 的正确性，和共识保证的最终一致性，可以证明随机数计算结果具有最终唯一性. 引理 2 在文献[26]中被证明，这里只给出其描述，引理 3 为共识流程的最终一致性的证明，定理 5 为分布式随机数生成方案的最终唯一性证明.

引理 2.（HAVSS 正确性[26]） 若 $p+1$ 个诚实节点完成标签为 *ID.d* 的秘密分发过程，存在一个定值 $z\in\mathbb{F}$ 使得（1）若秘密分发者在秘密分发阶段诚实分发标签为 *ID.d* 的秘密 s，则 $z=s$；（2）若诚实节点 P_i 恢复标签为 *ID.d* 的秘密 z_i，则 $z_i=z$.

引理 3.（最终一致性） 在最多 f 轮的错误输出后，诚实节点能得到相同的输出.

证明. 假设有 $f+1$ 轮两个节点的输出不同，那么存在一个输出的标签集合 V 的大小为 $f+1+f+1=2f+2$，因为一共最多 $2f+1$ 个节点参与秘密分发，出现矛盾.

证毕.

定理 5. 若 Π_{HAVSS} 方案具有正确性且共识具有最终一致性，那么 $\Pi_{\text{HAVSS-IDRB}}$ 具有最终唯一性.

证明. 使用反证法，假设存在一个概率多项式时间敌手 $\mathcal{A}$ 以不可忽略的概率攻破 $\Pi_{\text{HAVSS-IDRB}}$ 方案最终唯一性，构造一个概率多项式时间敌手 $\mathcal{B}$ 攻破 Π_{HAVSS} 方案正确性，即 $\mathcal{B}$ 同时作为 $\Pi_{\text{HAVSS-IDRB}}$ 方案最终唯一性挑战者回应 $\mathcal{A}$ 的问询.

$\mathcal{A}$ 先选择恶意节点集合 C，且 $|C|\leqslant f$，$\mathcal{B}$ 将 C 发送给 HAVSS 正确性挑战者 $\mathcal{C}$. $\mathcal{B}$ 和 $\mathcal{C}$ 各自执行初始化过程得到公开参数. $\mathcal{A}$ 向 $\mathcal{B}$ 发起 $\mathcal{O}^{\text{Update}(\cdot)}$ 问询，$\mathcal{B}$ 可以正常应答，因为仅需要公开参数. $\mathcal{A}$ 向 $\mathcal{B}$ 发起 $\mathcal{O}^{\text{Share}(\cdot,\cdot)}$ 问询，$\mathcal{B}$ 可以向 $\mathcal{C}$ 询问对于某个秘密值的份额并且把结果返回给 $\mathcal{A}$.

在挑战阶段，$\mathcal{B}$ 收到 $\mathcal{A}$ 发送的 (R, R')，$\mathcal{B}$ 将

(st^*, R, R') 发送给 $\mathcal{C}$. 最终，若 $\mathcal{B}$ 输出 1，则 $\mathcal{C}$ 也能输出 1.

综上，$\mathcal{B}$ 向 $\mathcal{A}$ 模拟了真实攻击环境，因此 $\mathrm{Adv}^{\mathrm{Unique}}_{\Pi_{\mathrm{HAVSS-IDRB}},\mathcal{A}}(\kappa) = \mathrm{Adv}^{\mathrm{Correct}}_{\Pi_{\mathrm{HAVSS}},\mathcal{B}}(\kappa)$，进而 $\mathcal{B}$ 攻破了 Π_{HAVSS} 方案，但是 Π_{HAVSS} 方案具备正确性，产生矛盾，所以不存在成功攻击 $\Pi_{\mathrm{HAVSS-IDRB}}$ 方案一致性的敌手 $\mathcal{A}$.

证毕.

5.2.3　鲁棒性

对于 I-DRB 而言，若秘密共享方案具有可验证性，也就是说在节点发送错误份额的情况下，不影响最终随机数的输出. 另外，共识的活性保证每个诚实节点最终会有输出，综上 I-DRB 满足鲁棒性. 因此对于本章的具体构造而言，HAVSS 具有的活性保证如果恶意节点不参与秘密恢复流程或发送错误份额，诚实节点也能够恢复原秘密值. 基于 HAVSS 和共识机制的活性，可以证明 HAVSS-IDRB 具有鲁棒性. 引理 4 在文献[26]中证明，这里只给出其定义.

引理 4.（HAVSS 活性[26]）　假设敌手为所有诚实节点初始化标签为 $ID.d$ 的秘密共享，并发送相应输入使之进入激活态. 若秘密分发者 P_d 是诚实的，那么所有诚实节点都能完成秘密分发过程.

定理 6.　若 Π_{HAVSS} 方案具有活性，且共识具有活性，那么 $\Pi_{\mathrm{HAVSS-IDRB}}$ 具有鲁棒性.

证明.　使用反证法，假设存在一个概率多项式时间敌手 $\mathcal{A}$ 以不可忽略的概率攻破 $\Pi_{\mathrm{HAVSS-IDRB}}$ 方案鲁棒性，构造一个概率多项式时间敌手 $\mathcal{B}$ 攻破 Π_{HAVSS} 方案活性，即 $\mathcal{B}$ 同时作为 $\Pi_{\mathrm{HAVSS-IDRB}}$ 方案鲁棒性挑战者回应 $\mathcal{A}$ 的问询.

$\mathcal{A}$ 先选择恶意节点集合 C，且 $|C| \leqslant f$，$\mathcal{B}$ 将 C 发送给 HAVSS 活性挑战者 $\mathcal{C}$. $\mathcal{B}$ 和 $\mathcal{C}$ 各自执行初始化过程得到公开参数. $\mathcal{A}$ 向 $\mathcal{B}$ 发起 $\mathcal{O}^{\mathrm{Update}(\cdot)}$ 问询，$\mathcal{B}$ 可以正常应答，因为仅需要公开参数. $\mathcal{A}$ 向 $\mathcal{B}$ 发起 $\mathcal{O}^{\mathrm{Share}(\cdot,\cdot)}$ 问询，$\mathcal{B}$ 可以向 $\mathcal{C}$ 询问对于某个秘密值的份额并且把结果返回给 $\mathcal{A}$.

在挑战阶段，$\mathcal{B}$ 收到 $\mathcal{A}$ 发送的 $\{\hat{s}_{i,j}\}_{i\in\mathcal{P}, j\in U\cap C}$，$\mathcal{B}$ 向挑战者 $\mathcal{C}$ 询问 $(st^*, \{\hat{s}_{i,j}\}_{i\in\mathcal{P}, j\in U\cap C})$. 最终，若 $\mathcal{B}$ 输出 1，则 $\mathcal{C}$ 也能输出 1.

综上，$\mathcal{B}$ 向 $\mathcal{A}$ 模拟了真实攻击环境，因此 $\mathrm{Adv}^{\mathrm{Robust}}_{\Pi_{\mathrm{HAVSS-IDRB}},\mathcal{A}}(\kappa) = \mathrm{Adv}^{\mathrm{Liveness}}_{\Pi_{\mathrm{HAVSS}},\mathcal{B}}(\kappa)$，进而 $\mathcal{B}$ 攻破了 Π_{HAVSS} 方案，但是 Π_{HAVSS} 方案具备活性，产生矛盾，所以不存在成功攻击 $\Pi_{\mathrm{HAVSS-IDRB}}$ 方案鲁棒性的敌手 $\mathcal{A}$.

证毕.

5.3　复杂度分析

复杂度包括通信复杂度和计算复杂度，其中通信复杂度是指每生成一次随机数所有节点传输的消息数量，计算复杂度是指每生成一次随机数每个节点执行指数或配对运算的数量.

本文与基于异步网络的 EPCC[17]和 RandShare[3]进行对比，对比结果如表 2 所示. EPCC 中 n 个节点作为秘密分发者向其他节点发送秘密份额，且需要发送多项式的承诺矩阵，大小为 $O(f^2)$，因此总体的通信复杂度为 $O(f^2n^2)$. RandShare 中也需要 n 个节点作为秘密分发者向其他节点发送秘密份额和对多项式系数的承诺，大小为 $O(f)$，另外需要针对每个秘密份额发送是否收到份额的投票信息，因此 n 个节点需要给其他所有节点发送 n 个投票信息，总体通信复杂度为 $O(n^3)$. 对于本方案而言，秘密共享算法中，$2f+1$ 个节点作为秘密分发者发送份额，即各自执行一个 HAVSS 实例，一个使用 KZG 承诺构造的 HAVSS 实例通信复杂度为 $O(n^2)$，那么 $2f+1$ 个节点总体的通信复杂度为 $O(fn^2)$，随机数验证过程同上. 秘密恢复过程中，每个节点发送 $O(f)$ 大小的标签集合给 n 个节点，那么 n 个节点总体的通信复杂度是 $O(fn^2)$，且每个节点向其他节点发送最多 $2f+1$ 个份额，因此 n 个节点总体的通信复杂度为 $O(fn^2)$.

表 2　本方案与其它异步网络方案的复杂度比较

方案	通信复杂度	计算复杂度
EPCC[17]	$O(f^2n^2)$	$O(fn^2)$
RandShare[3]	$O(n^3)$	$O(fn^2)$
本方案	$O(fn^2)$	$O(n^2)$

对于计算复杂度来说，EPCC 在秘密分发阶段，需要 $O(f^2)$ 次指数运算，在份额验证阶段，需要对 n^2 个点值和 n 个承诺进行验证，每次验证需要 f 次指数运算，因此总体计算复杂度为 $O(fn^2)$. RandShare 在秘密分发阶段，需要对多项式系数计算承诺，即需要 $O(f)$ 次指数计算，另外在份额验证阶段，需要对 n^2 份额进行验证，每次验证需要执行 f 次指数操作，因此总体的计算复杂度也为 $O(fn^2)$. 对于本方案而言，秘密共享算法需要计算 n 个多项式的承诺和 n 个多项式上的 n 个点值证明，因此计算复杂度为 $O(n^2)$. 秘密恢复过程需要计算拉格朗日差值，无配对或指数运算，因此计算复杂度为 $O(1)$. 在随机数验证过程中，对于一个秘密而言，需验证 n 个点值计算结果. 每次验证需要 2 次配对运算和 1 次指数运算，且一共有 $2f+1$ 个秘密值，因此计算

复杂度为 $O(fn)$，即总体的计算复杂度为 $O(n^2)$.

综上，相对于现有基于异步网络的方案，本方案在通信复杂度和计算复杂度上均有优势.

6 仿真系统实现与测试

本章介绍分布式随机数仿真系统的实现与测试结果，该系统为第 5 章具体构造的仿真实现. 第 6.1 节从系统框架和四个核心模块角度介绍系统实现，其中详细介绍了数据结构定义；第 6.2 节从性能角度对仿真系统进行两个维度的测试，并对测试结果进行详细分析.

6.1 仿真系统实现

该仿真系统使用 Golang 语言实现，由下至上分别为核心组件层、网络传输层、通用处理层以及对外服务层.

核心组件层主要负责方案所依赖的底层密码技术及共识流程的实现，包括 KZG 承诺、秘密共享、秘密验证及秘密恢复算法等. 网络传输层主要负责在节点之间提供数据传输通信功能，并向上层提供可靠的数据传输服务，包括消息结构定义、节点存储定义、通信协议选取、消息传播算法以及消息序列化/反序列化流程. 通用处理层主要负责分布式随机数方案中协议启动模块及通用功能的实现，包括节点选取算法、随机数计算算法、随机数验证算法和状态更新算法等. 对外服务层不包含业务逻辑，负责接收节点的请求，并将响应或计算结果呈现给节点，同时做一些配置文件校验的工作.

在核心组件层的密码算法模块中，KZG 承诺选取 BLS12-381 椭圆曲线，该曲线是嵌入度为 12 的 381 位的 BLS 曲线，具备 128 位安全级别，参考 bls-eth-go-binary 库[①]. 本实现主要包括 3 个核心函数，分别是承诺计算 CommitToPoly、点值证明计算 ComputeProof 和证明验证 CheckProof.

在网络传输层的消息传播模块中，节点间的信道为认证加密信道，每个节点需要生成证书及公私钥，然后节点间验证证书有效性，建立 TLS 连接，监听相应端口. 其中使用了 Golang 的 crypto/x509 和 crypto/tls 库负责证书加载和 TLS 配置初始化，并且使用 Golang 的 encoding/gob 库进行消息的序列化和反序列化.

分布式随机数仿真系统核心的数据结构包括消息类型 MessageType、节点发送的消息 ProtocolMessage 和节点本地状态 ProtocolState.

消息有四种类型，分别是广播消息（SEND）、应答消息（ECHO）、准备消息（READY）和恢复消息（RECOVER）.

节点发送的消息 ProtocolMessage 定义为消息序号 Nonce、消息广播者 Broadcaster、消息接收者 Receiver、消息类型 MessageType 和消息负载 Payload.

节点本地状态 ProtocolState 包括所收到的消息负载 Payload、消息广播者 Broadcaster、消息接收者 Receiver、消息序号 Nonce、应答消息内容 EchoMessages、准备消息内容 ReadyMessages、恢复消息内容 RecoverMessages 以及四个布尔变量（是否已发送应答消息 SentEcho、是否已发送准备消息 SentReady、是否已发送恢复消息 SentRecover、是否已完成秘密恢复 Recovered）.

6.2 性能测试

为了深入分析分布式随机数仿真系统的效率，本文测试了系统性能. 性能测试包括三部分，（1）不同节点数对方案各阶段计算耗时的影响；（2）相同节点数情况下，不同方案算法计算耗时对比；（3）不同节点数对不同方案随机数生成过程中带宽资源消耗情况的影响. 测试环境为每个节点部署在一个 Amazon EC2 t2.micro 实例上（1GB 内存，8GB 硬盘，1 个 CPU 核心和 60-80 Mbit/s 的网络带宽），操作系统为 Ubuntu 18.04，Golang 语言版本为 1.15.1.

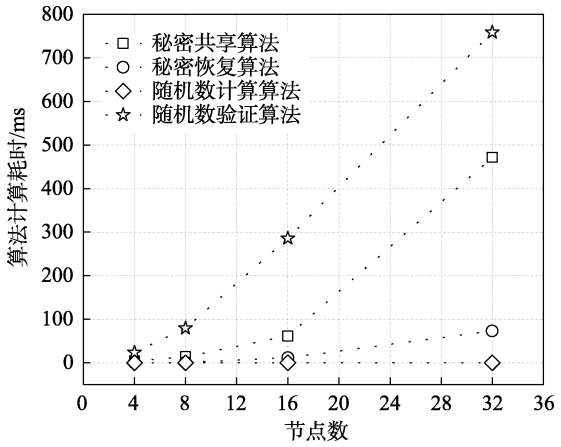


图 5 不同节点数对方案各算法计算耗时的影响

首先是方案各阶段计算耗时情况，测试结果如图 5 所示. 考虑到节点选取、状态更新和共识阶段无需密码学组件计算，因此图中仅展示剩余 4 个算法在 4、8、16、32 节点数情况下的计算耗时. 网络时延忽略不计. 从测试结果可以看到较为耗时阶段随机数验证，原因是随机数验证过程需要验证

① https://github.com/herumi/bls-eth-go-binary

$(2f+1)n$ 次点值计算结果，且因为点值从多个节点获取，因此无法进行批量验证来压缩验证时间.

另外，本文测试了 8 节点情况下本方案与其他两个异步网络方案的算法计算耗时的比较，如图 6 所示. 考虑到秘密恢复和随机数计算两个算法在不同方案中是类似的(秘密恢复均使用拉格朗日差值，随机数计算均为异或)，因此不再进行测试. 从测试结果可以看到，在秘密共享阶段，本方案比 RandShare 和 EPCC 耗时多，因为本方案需要计算 n 个多项式的 n 的点值证明，即 $O(n^2)$ 次指数运算，而 RandShare 仅需 $O(f)$ 次指数运算，EPCC 仅需 $O(f^2)$ 次指数运算. 但在随机数验证阶段，本方案的计算耗时优于其他两个方案，因为本方案在验证一个份额时是常数级别的操作.

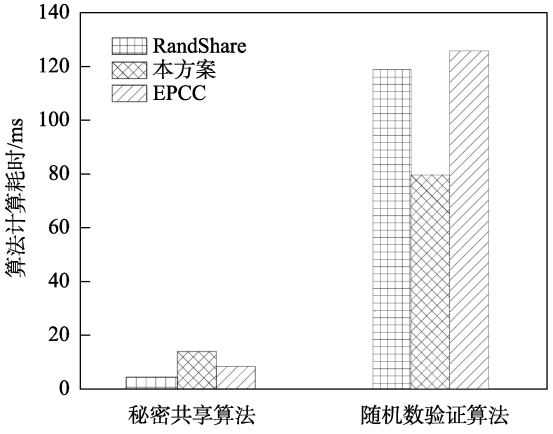


图 6　8 节点情况下算法计算耗时对比

最后本文测试了不同节点数对通信开销的影响，如图 7 所示. 此处通信开销描述为每生成一次随机数每个节点发送的字节数. 考虑到本文只针对小规模的场景，因此节点数仅选取 4 节点至 32 节点. 对于本方案而言，4 节点情况下，平均每节点收发消息数据量为 1.98KB，8 节点为 5.91KB，16 节点为 27.57KB，32 节点为 101.82KB，带宽开销变化幅度基本符合理论分析的 $O(\lambda n^2)$. 在相同节点数情况下，RandShare 和 EPCC 每个节点都会花费更多的带宽，增大消息吞吐压力，且随着节点数增多，通信开销差距越来越大. 另外对于本方案而言，非秘密分发者所消耗的带宽会低于平均值，这也从侧面反映本方案有较优的通信开销.

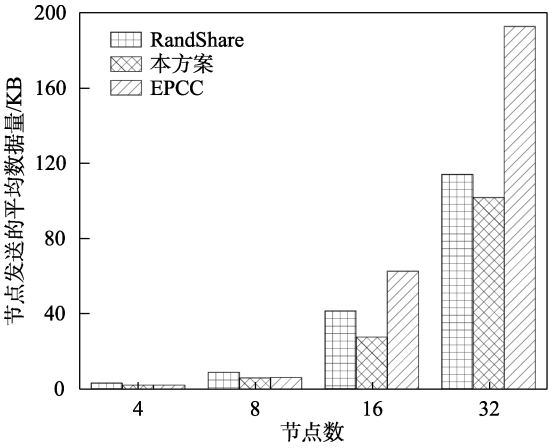


图 7　不同节点数对节点通信开销的影响

7　结　论

本文主要研究分布式随机数生成技术，根据所依赖的底层密码学技术归纳为 6 种类型，并指出现有研究存在的 3 个关键性问题. 在此基础上重点研究基于秘密共享的分布式随机数生成方案，首先提出了一个交互的分布式随机数生成 I-DRB，该构造相比于非交互的分布式随机数生成构造，不需要依赖复杂的分布式密钥生成协议. I-DRB 通用构造由秘密共享和共识组成，提取不同方案的共性，兼容不同方案的特殊性，共包含 7 个算法，满足 3 个安全目标. 其次，本文设计了一个面向异步网络的安全分布式随机数生成方案. 该方案主要面向小规模的许可化环境，针对无法使用异步二元共识的关键性问题，将异步可验证秘密共享作为方案子模块，并在其基础上结合了投票共识的方式. 另外，本文给出了该方案伪随机性、唯一性和鲁棒性的形式化安全证明. 在复杂度分析方面，本方案达到了 $O(fn^2)$ 通信复杂度和 $O(n^2)$ 的计算复杂度，在目前的异步方案中较优. 最后，本文实现了分布式随机数仿真系统. 该仿真系统使用 Golang 语言，由下至上分别为核心组件层、网络传输层、通用处理层以及对外服务层，本文针对该仿真系统进行了不同节点数下的计算耗时和通信开销性能测试，测试结果表明虽然随着节点数增多，本方案的通信开销增幅提升，但相比于现有异步方案，在相同节点数的情况下本方案的通信开销更低，性能有所提升. 本文为该领域未来的研究提供了如下方向：

（1）本文仅重点研究了基于秘密共享的分布式随机数生成方案，未来的工作可以尝试对其他类别的方案进行深入研究和优化（如基于门限签名或基于 VDF 的方案）.

（2）本文方案仅适合小规模的许可化环境. 在第 5.3 节，本研究通过对方案进行分析，发现通信复杂度从现有方案的 $O(n^3)$ 甚至 $O(f^2n^2)$ 降低到了 $O(fn^2)$. 虽然有所降低，但仍不适用于大规模的节点部署，探索异步网络下方案的可扩展性，在未来

是一个有价值和挑战性的研究方向.

参 考 文 献

[1] Bernstein D J, Lange T, Niederhagen R. Dual EC: A standardized back door. The New Codebreakers, 2016: 256-281

[2] Atzei N, Bartoletti M, Cimoli T. A survey of attacks on ethereum smart contracts//Proceedings of the 6th International Conference on Principles of Security & Trust. Uppsala, Sweden, 2017: 164-186

[3] Syta E, Jovanovic P, Kogias E K, et al. Scalable bias-resistant distributed randomness//Proceedings of the 38th IEEE Symposium on Security and Privacy. San Jose: USA, 2017: 444-460

[4] Cascudo I, David B. SCRAPE: Scalable randomness attested by public entities//Proceedings of the 15th International Conference on Applied Cryptography and Network Security. San Jose: USA, 2017: 537-556

[5] Schindler P, Judmayer A, Stifter N, et al. Hydrand: Efficient continuous distributed randomness//Proceedings of the 41st IEEE Symposium on Security and Privacy. San Francisco, USA, 2020: 73-89

[6] Gennaro R, Jarecki S, Krawczyk H, et al. Secure distributed key generation for discrete-log based cryptosystems. Journal of Cryptology, 2007, 20(1): 51-83

[7] Hanke T, Movahedi M, Williams D. Dfinity technology overview series, consensus system. https://arxiv.org/pdf/1805. 04548.pdf, 2018-05-11/2021-12-04

[8] Gilad Y, Hemo R, Micali S, et al. Algorand: Scaling byzantine agreements for cryptocurrencies//Proceedings of the 26th Symposium on Operating Systems Principles. Shanghai, China, 2017: 51-68

[9] David B, Gaži P, Kiayias A, et al. Ouroboros praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain// Proceedings of International Conference on the Theory and Applications of Cryptographic Techniques. Tel Aviv, Israel, 2018: 66-98

[10] Shamir A. How to share a secret. Communications of the ACM, 1979, 22(11): 612-613

[11] Feldman P. A practical scheme for non-interactive verifiable secret sharing//Proceedings of the 28th Annual Symposium on Foundations of Computer Science. Los Angeles, USA, 1987: 427-438

[12] Bhat A, Shrestha N, Kate A, et al. RandPiper: Reconfiguration-friendly random beacons with quadratic communication//Proceedings of the 28th ACM Conference on Computer and Communications Security. Korea, 2021: 3502-3524

[13] Stadler M. Publicly verifiable secret sharing//Proceedings of International Conference on the Theory and Applications of Cryptographic Techniques. Saragossa, Spain, 1996: 190-199

[14] Cascudo I, David B. ALBATROSS: Publicly attestable batched randomness based on secret sharing//Proceedings of the 26th International Conference on the Theory and Application of Cryptology and Information Security. Kobe, Japan, 2020: 311-341

[15] Kiayias A, Russell A, David B, et al. Ouroboros: A provably secure proof-of-stake blockchain protocol//Proceedings of the 37th Annual International Cryptology Conference. Santa Barbara, USA, 2017: 357-388

[16] Sourav D, Vinith K, Irene M I, et al. SPURT: Scalable distributed randomness beacon with transparent setup. https://eprint. iacr.org/2021/100.pdf, 2021-05-01/2021-12-04

[17] Kokoris-Kogias E, Malkhi D, Spiegelman A. Asynchronous distributed key generation for computationally-secure randomness, consensus, and threshold signatures//Proceedings of the 27th ACM Conference on Computer and Communications Security. Virtual Event, USA, 2020: 1751-1767

[18] Micali S, Rabin M, Vadhan S. Verifiable random functions// Proceedings of the 40th Annual Symposium on Foundations of Computer Science. New York, USA, 1999: 120-130

[19] Goldreich O, Goldwasser S, Micali S. How to construct random functions. Journal of the ACM, 1986, 33(4): 792-807

[20] Cachin C, Kursawe K, Shoup V. Random oracles in constantinople: Practical asynchronous byzantine agreement using cryptography. Journal of Cryptology, 2005, 18(3): 219-246

[21] Boneh D, Bonneau J, Bünz B, et al. Verifiable delay functions// Proceedings of the 38th Annual International Cryptology Conference. Santa Barbara, USA, 2018: 757-788

[22] Lenstra A K, Wesolowski B. Trustworthy public randomness with sloth, unicorn, and trx. International Journal of Applied Cryptography, 2017, 3(4): 330-343

[23] Schindler P, Judmayer A, Hittmeir M, et al. RandRunner: Distributed randomness from trapdoor VDFs with strong uniqueness//Proceedings of the 28th Annual Network and Distributed System Security Symposium. Virtual Event, 2021: 1-18

[24] Shoup V. Practical threshold signatures//Proceedings of International Conference on the Theory and Applications of Cryptographic Techniques. Bruges, Belgium, 2000: 207-220

[25] Kate A, Zaverucha M G, Goldberg I. Constant-size commitments to polynomials and their applications//Proceedings of the 16th International Conference on the Theory and Application of Cryptology and Information Security. Singapore, 2010: 177-194

[26] Alhaddad N, Varia M, Zhang H. High threshold AVSS with optimal communication complexity//Proceedings of the 25th International Conference on Financial Cryptography and Data Security. Virtual Event, 2021: 479-498

[27] Castro M, Liskov B. Practical byzantine fault tolerance// Proceedings of USENIX Symposium on Operating Systems Design and Implementation. New Orleans, USA, 1999: 173-186

[28] Liu Yizhong, Liu Jianwei, Zhang Zongyang, et al. Overview on Blockchain consensus mechanisms. Journal of Cryptologic Research, 2019, 6(4): 395–432 (in Chinese)
(刘懿中, 刘建伟, 张宗洋等. 区块链共识机制研究综述. 密码学报, 2019, 6(04): 395-432)

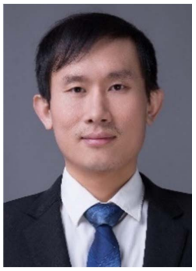
[29] David G, Jia L, Mihai O, et al. Fully distributed verifiable random functions and their application to decentralised random beacons//Proceedings of the 6th IEEE European Symposium on Security and Privacy. Vienna, Austria, 2021: 88-102

附　录.

本节使用 4.1 小节的 I-DRB 通用构造来形式化分析现有典型的基于 PVSS 的分布式随机数生成方案 SCRAPE[4].

SCRAPE[4]是第一个基于 PVSS 的分布式随机数生成方案，令网络中总节点数为 n，节点集合为 $\mathcal{N}=\{P_1,\ldots,P_n\}$，其中有 f 个恶意节点，且满足 $n=2f+1$. SCRAPE 方案 $\Pi_{\text{SCRAPE-IDRB}}=(\text{Init},\text{GrpGen},\text{Commit},\text{Recovery},\text{CombRand},\text{VerifyRand},\text{UpdState})$ 由 PVSS 方案 $\Pi_{\text{PVSS}}=(\text{Setup},\text{Distribute},\text{VerifyShare},\text{Reconstruct})$ 和共识 Consensus(P_i,r) 构成. 需要说明的是，该方案中所有节点都是秘密分发者，因此在算法 GrpGen 中，输出的节点子集即为原节点集合，另外该方案需要每个节点需要对诚实节点集合达成共识.

SCRAPE 的伪随机性可以由 PVSS 方案的秘密不可区分性[4]（Indistinguishability of Secrets）和多个诚实节点执行秘密共享算法推导出，进而保证不可预测性和抗偏置性. 另外，因为 PVSS 方案具有正确性[4]，并且诚实节点通过共识会拥有相同的诚实节点集合（即发送合法秘密值及其承诺的节点），因此最终计算的随机数也是唯一的. 最后，因为 PVSS 方案具有可验证性[4]（也可理解为活性），若敌手发送错误秘密份额或不发送打开，诚实节点会丢弃敌手的秘密份额，并与其他诚实节点共同恢复出原秘密，从而保证了随机数的鲁棒性.



ZHANG Zong-Yang, Ph.D., associate professor. His research interest focuses on blockchain and cryptography.

LI Tong, M.S. candidate. Her research interest focuses on blockchain and cryptography.

ZHOU You, B.E. candidate. His research interest focuses on blockchain and cryptography.

CHEN Lao, Ph.D. candidate, research associate. His research interest focuses on network security and blockchain.

Background

Distributed randomness generation means that a group of participants generate verifiable random numbers in an environment without a trusted third party, so as to prevent central institutions from cheating, avoid single points of failure, and improve security. The generated random numbers can be used in blockchain consensus mechanisms, smart contract applications, etc. This research belongs to the field of blockchain and cryptography.

In recent years, distributed randomness generation schemes using different cryptographic technologies have been proposed, including secret sharing, VRF, threshold signature, VDF and so on. For schemes based on secret sharing, the communication complexity is relatively high, and there is a lack of research on general construction and formal security models. For schemes based on threshold signature and VRF, the communication complexity is relatively low, but security is sacrificed, and a complex distributed key generation protocol is required during initialization. For VDF-based solutions, it depends on precise time parameters, it is difficult to apply them in practice. In addition, most of the current solutions rely on the synchronous network model, which deviates from the real network environment.

Therefore, this paper studies the scheme based on secret sharing, and focuses on the research of general construction, the weakening of network assumptions and the optimization of communication overhead. We firstly propose a general construction of interactive distributed random beacon and a formal security model, which fills the gap in theoretical research. Then, we design a distributed randomness generation scheme for asynchronous networks, reducing the communication complexity to $O(fn^2)$, and finally we implement a distributed randomness simulation system. The test results show that the communication overhead is reduced by 11% and 47% compared with the existing two asynchronous schemes. Note that our proposed scheme has two shortcomings, it need a trusted third party to setup and it need at most f rounds of consensus.

This work is supported in part by the National Key Research and Development Program of China (2021YFB-2700200), the National Natural Science Foundation of China (61972017, 72031001, 61972310), the Beijing Natural Science Foundation (M22038, M21033, 4202037), the Yunnan Key Laboratory of Blockchain Application Technology (202105AG070005) and the Fundamental Research Funds for the Central Universities (YWF-22-L-1039).

Blockchain and cryptography are main research topics of our research group. In the past few years, we have worked out many research papers that have been published in Computers&Security, Journal of Cryptologic Research and other journals.