小 型 微 型 计 算 机 系 统
Journal of Chinese Computer Systems
DOI:10.20009/j.cnki.21-1106/TP.2025-0125
2026 年 4 月 第 4 期
Vol.47 No.4 2026

适用于链上多元场景的门限 Schnorr 签名方案

王志伟,张振琦,朱秋池

(南京邮电大学 计算机学院,南京 210023)
E-mail:zhwwang@njupt.edu.cn

摘 要: 在区块链系统中,共识机制是确保数据一致性和系统安全性的重要环节.然而,频繁的共识场景切换会带来高频次的密钥分发需求,这一过程中,大量的计算负担会集中在参与者身上,直接影响共识效率.针对这一问题,本文提出了一种多场景适用的门限 Schnorr 签名方案.该方案通过采用分布式密钥分发协议,无需依赖强可信的第三方来完成密钥分发.在不同的应用场景下,系统中的参与者仅需重新选择身份索引,而无需更换私钥或公钥,从而显著减少了重新分发密钥所带来的时间和计算开销.此外,本文设计了一种基于布隆过滤器的去重机制,以低成本对参与成员去重.门限签名的生成条件是当单签名数量超过预设阈值时,签名聚合者能够快速生成合规的门限签名.本文从鲁棒性角度进行了深入分析,并基于离散对数假设,在随机预言模型下证明了该方案的不可伪造性.最后,通过转账和投票两种实际应用场景,并在本地 Ganache 测试网下对此方案的链上多场景适用性进行了研究.

关 键 词: 门限签名;Schnorr 签名;随机预言机模型;鲁棒性;离散对数;布隆过滤器
中图分类号: TP309 **文献标识码:** A **文 章 编 号:** 1000-1220(2026)04-0953-07

Threshold Schnorr Signature Scheme for Multi-scenario Blockchain Applications

WANG Zhiwei,ZHANG Zhenqi,ZHU Qiuchi
(School of Computer Science,Nanjing University of Posts and Telecommunications,Nanjing 210023,China)

Abstract: In blockchain systems,consensus mechanisms play a crucial role in ensuring data consistency and system security. However, frequent transitions between consensus scenarios introduce high-frequency key distribution processes,imposing substantial computational burdens on participants and directly affecting consensus efficiency. To address this issue,this paper proposes a threshold Schnorr signature scheme suitable for multi-scenario applications. By employing a distributed key distribution protocol,the scheme eliminates the need for a highly trusted third party for key distribution. In various application scenarios,participants only need to reselect their identity indexes without changing their private or public keys,significantly reducing the time and computational overhead associated with key redistribution. Additionally,this paper designs a deduplication mechanism based on a Bloom filter to ensure the uniqueness of participating members at a low cost. The threshold signature is generated when the number of individual signatures exceeds a preset threshold,allowing the signature aggregator to quickly generate a valid threshold signature. A comprehensive robustness analysis is provided, and the unforgeability of the scheme is proven under the discrete logarithm assumption in the random oracle model. Finally,the practicality of the scheme is validated in two real-world application scenarios:transfering and voting. Practical evaluations on the Ganache testnet demonstrate the scheme's adaptability to multi-scenario on-chain environments.

Keywords: threshold signature;Schnorr signature;random oracle;robustness;discrete logarithm;Bloom filter

0 引 言

区块链技术源于比特币系统,后来随着全球数字化程度的不断加深,区块链产业也迎来了飞速发展.区块链技术以其去中心化和不可篡改等特点,已经广泛应用于社会治理、教育医疗等领域,但是随着区块链应用场景的不断增加,对区块链的性能要求也越来越高,区块链性能中最核心的问题就是达成共识效率低和计算复杂度高[1].Schnorr 签名算法是一种线性的公钥密码算法,拥有更低的计算消耗,可以改善区块链钱包中存在的达成共识效率低的缺陷.

目前,随着比特币、以太坊等加密数字货币的快速发展,区块链技术收到社会各界的广泛关注,加快了区块链技术在各种实际场景下的应用.在这种背景下,加密数字货币存储与交易的需求量在持续增长,而作为存储加密数字货币的设备,区块链钱包的实际应用场景也在不断地扩大.单签钱包是当前区块链钱包应用中较为常见的一种形式,在单签钱包中用户通过公私钥对对其所拥有的加密数字货币进行维护与使用,但是一旦攻击者获取到私钥存储的位置,那么就可以利用单点攻击造成节点故障,从而窃取用户的加密数字货币.为了进一步避免单签钱包所带来的交易风险,区块链中引入了多签钱包对用户的加密数字货币进行维护,与此同时,多重签名技术由于其具有较高的安全性、可用性以及灵活性,一直是多

收稿日期:2025-03-17 收修改稿日期:2025-05-21 基金项目:国家自然科学基金项目(62372245)资助. 作者简介:**王志伟**(通信作者),男,1976 年生,博士,教授,博士生导师,CCF 高级会员,研究方向为公钥密码学;**张振琦**,男,2000 年生,硕士研究生,研究方向为密码应用;**朱秋池**,女,2000 年生,硕士研究生,研究方向为密码应用.

签钱包应用中的主要签名方案之一.

1979 年,Shamir[2]提出一种基于拉格朗日插值法的秘密共享方案,通过将秘密嵌入多项式常数项并生成份额,实现了密钥的分散管理,这是门限签名的基础. 1987 年,Desmedt 等人[3]在论文中提出首个门限签名方案,基于 Shamir 秘密共享技术将秘密值分裂成 t 个“碎片”并由相等数量的签名者持有,方案要求当签名开始时,需要至少 t 个签名者利用各自的密钥份额协作才能完成门限签名,任何小于门限值的签名者子集都无法完成签名过程,但是该方案并未解决密钥生成阶段的分布式安全问题,严重依赖可信第三方生成密钥,并且由于是基于 RSA 设计的,还存在着计算效率低、权限分配单一等问题. 1991 年,Pedersen 等人[4]提出首个基于 Schnorr 的门限签名方案,该方案首次实现分布式密钥生成(Distributed Key Generation,DKG),支持无需可信第三方的安全 DKG,但是该方案存在合谋攻击漏洞,即恶意参与者数量超过或仅仅达到门限值 t 时,攻击者可通过合谋重构群秘密多项式,从而伪造其他成员的签名. 并且该方案门限值是固定的,无法动态适应需求,而且当签名者变化时,该方案必须重新执行分发密钥. 1999 年,Gennaro 等人[5]提出一个基于 RSA 的高效门限签名方案,该方案优化了早期 RSA 方案中因分布式模数导致的计算效率低下的问题,将参与者的计算复杂度优化为 $O(n^2)$. 2000 年,Shoup 等人[6]提出一个基于 RSA 的门限签名方案,通过定期刷新密钥分片抵御长期密钥泄露攻击,弥补了传统方案在动态敌手攻击下的脆弱性,但是该方案需要多轮交互式零知识证明,带来了巨大的计算和通信代价. 2003 年,Boneh 等人[7]提出一个基于 BLS 签名的门限方案,实现了非交互式聚合门限签名,不仅将签名大小从 $O(t)$ 压缩到固定长度,而且将签名成员的本地计算复杂度降低至 $O(1)$,解决了多用户协作场景的通信开销问题,但是该方案是基于 BLS 设计的,相比较 ECDSA 和 Schnorr,单次签名验证开销是 ECDSA 的 2 ~ 3 倍.

2007 年,Zhang 等人[8]提出了动态门限值的概念,允许灵活调整门限值,但是该方案需要预先共享多个群密钥,导致密钥管理的复杂度成指数级上升. 2010 年,Maxwell 等人提出了一个基于 Waters 签名框架和分布式密钥生成协议 DKG 的新门限签名[9],此方案首次在标准模型下证明了不可伪造性,并实现了抗合谋攻击,但是它引入可验证密钥分片防止分发阶段的恶意篡改,不仅导致签名代价增大,而且群公钥长度也随成员数量线性增长. 2012 年,Wei 等人[10]结合前向安全与门限签名,设计私钥按时间片更新,同时用椭圆曲线群代替 RSA 群,将签名长度缩小,但是该方案要求密钥更新需全体签名者集合成员在线,因此不具有很强现实意义. 同年,Castagnos 等人基于签密提出了首个抗合谋攻击的(t,n)门限签名[11],其核心思想是通过冗余信息绑定机制,防止密钥分配中心伪造签名,但是该方案在签名阶段需要多个验证者协作验证,同时在初始化阶段过分依赖可信第三方,不仅存在签名实时性问题,而且还存在单点故障风险. 2017 年,Yan 等人[12]提出一个基于格的抗量子门限方案,通过格密码代替 ECC 或 RSA,同支持动态成员管理,成员加入或退出时密钥更新效率相较于传统方案提高 30%,但是格困难问题导致签名长度很大,同时本地计算代价很高. 2020 年,Komlo 等人[13]基于 Schnor 签名提出一个动态轮次门限方案,将传统方案的 3 轮交互降低至 1 至 2 轮,同时不需要所有成员同时在线,但是该方案的成员加入和退出需要复杂的协议设计,和进行单轮方案时需要进行预处理算法,这大大增加了计算代价,对多元场景的支持有限. 2023 年,Baird 等人[14]提出一个适用于动态场景的门限方案,基于任意动态可重叠的子集设计门限签名,但是该方案较多的双线性配对和计算代价. 近年来许多带复杂特性或抗量子计算机攻击的门限签名方案被提出,例如 DelPino 等人提出的基于格密码的紧致门限签名方案[21];Fischlin 等人提出的超越不可伪造特性的门限签名方案[22];Ji 等人提出的指定确认者的门限签名方案等[23]. 但是这些方案并未关注门限签名在动态多元场景的切换代价,并且由于带复杂特性,密钥分发过程带来更为复杂的时间和计算开销.

本文基于 Schnorr 签名方案,提出了一种适用于链上多元场景的门限 Schnorr 签名方案(Multi-scenario On-chain Schnorr Threshold Signature,MOSTS),方案的创新点如下:

1)门限签名方案是基于 Schnorr 签名方案设计的,拥有无双线性配对的计算优势.

2)当链上应用场景变化、或者签名集合有新成员加入时,原成员不需要重新选择密钥份额和计算公钥,只需要系统更新成员的索引集,从而降低原成员的计算代价,频繁进行密钥分发过程带来的时间和计算开销.

3)基于布隆过滤器设计一种快速定位机制,系统会将授权成员的公钥哈希插入布隆过滤器,当有新成员加入集合时,通过布隆过滤器查询,若返回“不存在”,则直接拒绝新成员,反之再用成员公钥进一步验证,避免布隆过滤器的误判.

4)文中通过额外一轮验证开销,可以抵御流氓密钥攻击,并详细证明了方案具有鲁棒性;之后基于离散对数难题和随机预言模型,证明了方案具有不可伪造性;最后通过本地搭建的以太坊测试网,证明了方案在场景切换和成员加入时拥有更低的代价.

1 预备知识

1.1 离散对数假设

定义. 对于一个群 (G,p,g)[15],定义敌手 A 的优势 Adv_G^{dl} 为:

$$Adv_G^{dl}=\Pr[y=g^x:y\leftarrow_r G,x\leftarrow_r A(y)]$$

从上面的公式中可以看出,概率的关键在于 y 值需要从群中随机抽取,敌手 A 的选取亦需要保证随机性. 如果敌手 A 至多在时间 τ 内,使得 $Adv_G^{dl}\geqslant\varepsilon$,那么说明敌手 $A(\tau,\varepsilon)$ 攻破了离散对数假设. 如果不存在这样的敌手,则离散对数假设是 (τ,ε) 困难的.

1.2 广义分叉引理

分叉引理的概念由 Stern[16]首次提出并用于盲签名安全分析. 接着,Bellare 等人[17]于 2006 年改进了该理论,提出适用于一般数字签名的广义分叉引理,而 Bagherzandi 等人[18]完成了形式化证明与系统性完善. 在分叉引理的执行过程中,挑战者 C 需在安全性证明中执行两次挑战:首次挑战的输出序列在某个查询成员前与原始交互保持一致,但该成员随后的响应值将被重构. 通过精心设计的重放策略,攻击者可诱导

挑战者暴露目标困难问题的解,从而完成安全性归约证明. 这种双阶段挑战机制有效解决了概率多项式时间算法在随机预言模型下的安全性论证难题[1]. 下面对扩展的分叉引理进行介绍. 规定 $f=(\rho,h_1,\cdots,h_{q_H})$ 是执行算法 A 涉及到的随机性向量,其中 ρ 是 A 的随机磁带,h_i 是 A 对哈希函数第 i 个查询的回答,q_H 是 A 询问哈希函数的最大次数. 设 Ω 是 f 所属的向量空间,$f|_i=(\rho,h_1,\cdots,h_{i-1})$,$A(in,f)$ 是在算法 A 中输入 in 和随机参数 f 的过程. 若 $A(in,f)$ 输出 $(J,\{out_j\}_{j\in J})$,则代表成功,其中 J 是 $\{1,\cdots,q_H\}$ 的一个子集,$\{out_j\}_{j\in J}$ 是其他输出的集合;若 $A(in,f)$ 输出 $\varnothing$,则代表算法 A 失败. 用 p 表示算法流程执行成功,即输出合法结果的概率. 对于一个给定的输入 in,扩展的分叉引理算法 GF_A 如算法 1 所示.

算法 1. 扩展的分叉引理算法 GF_A

输入: $in \xleftarrow{\$} IG, f=(\rho,h_1,\cdots,h_{qH}) \xleftarrow{\$} \omega$

1. $(J,\{\text{out}_j\}_{j\in J}) \leftarrow A(\text{in},\text{f})$
   if $=J=\varnothing$, then output fail
2. 令 $J=\{j_1,\cdots,j_n\}$ 且 $j_1\leqslant\cdots\leqslant j_n$
   For $i=1,\cdots,n$ do:
   $succ_i\leftarrow 0; k_i\leftarrow 0;$
   $k_{max}\leftarrow 8nq_H/\varepsilon\cdot ln(8n/\varepsilon)$
3. 不断重复执行以下操作直到 $succ_i=1$ or $k_i>k_{max}$;
   $f'\xleftarrow{\$}\Omega$ 满足 $f'|_{j_i}=f'|_{j_i}$
   $f''=(\rho,h_1,\cdots,h_{j_i-1},h''_{j_i},\cdots,h''_{q_H})$
   $(J'',\{out''_j\}_{j\in J''})\leftarrow A(in,f'')$
4. if $h''_{j_i}\neq h_{j_i}$ and $J''=\varnothing, j_i\in J''$
   then $out'_{j_i}\leftarrow out''_{j_i}; succ_i\leftarrow 1$
   if $succ_i=1$ for all $i=1,2,\cdots,n$
   then output $(J,\{out_j\}_{j\in J},\{out''_j\}_{j\in J''}$
   else output fail

1.3　分布式密钥分发

在(t,n)秘密共享方案中,经销商将秘密 s 分发给 n 个参与者 $P_1,\cdots,P_n$,使得任何至少有 t 个参与者的群体都可以重建秘密 s,而任何少于 t 个参与者的群体都无法获得关于秘密 s 的任何信息. 在 Shamir 的(t,n)阈值秘密共享方案中,为了将 $s\in Zq$ 分配给 $P_1,\cdots,P_n$,分发者选择一个 t−1 阶的多项式 $f(i)=a_0+a_1i+\cdots+a_{t-1}i^{t-1}$,并且存在有 $s=f(0)$,每一个参与者 P_i 的秘密值份额是 $s_i=f(i)$.

由 t 个参与者组成的任意群组 P 可以通过拉格朗日插值重构多项式:

$$f(u)=\sum_{i\in P}f(i)\lambda_i(u),\lambda_i(u)=\prod_{\substack{j\in P\\ j\neq i}}\frac{u-j}{i-j}$$

由于满足 s = f(0),所以群组 P 可以按如下方式重构秘密:

$$s=f(0)=\sum_{i\in P}f(i)\lambda_i,\lambda_i=\lambda_i(0)=\prod_{\substack{j\in P\\ j\neq i}}\frac{j}{j-i}$$

2　多场景门限 Schnorr 签名定义和安全模型

在本节中,将介绍适用于多场景的门限 Schnorr 签名方案的定义和安全模型. 其中的鲁棒性允许攻击者参入门限签名过程,但无法伪造合法签名者的份额. 不可伪造性是以攻击者输出伪造的 (R,z),其中 R 是参与签名者的承诺之积,z 是攻击者伪造的签名,然后利用 z 解决离散对数问题,达到门限模型中定义不可伪造性的目的.

2.1　方案定义

令签名者集合为 $PN=\{P_1,\cdots,P_n\}$,签名者集合为 $PT=\{P_1,\cdots,P_t\}$,满足 $n\geqslant t$.

1)密钥分发阶段(*DKG*):

(a)系统设置算法(*SetUp*):输入安全参数 λ,输出公开参数 $Par=(G,p,g,H_1,H_2)$,其中 G 为 p 阶的乘法循环群,g 是其生成元. H_1,H_2 是 2 个哈希函数.

(b)密钥生成算法(*KeyGen*):由每名签名者 P_i 生成自己的秘密值和公钥 $\{x_i,Y_i=g^{x_i}\}$,并生成一个关于秘密值的零知识证明 $\{k_i\xleftarrow{\$}Z_p^*, K_i=g^{k_i}, c_i=H_1(i,K_i,Y_i), s_i=r_i+c_i\cdot x_i\}$,最后广播三元组 (Y_i,K_i,s_i).

(c)群公钥生成算法(*GroupKeyGen*):签名者 P_i 收到其他成员的三元组后,若全部零知识验证等式 $K_l=g^{s_l}\cdot Y_l^{-c_l}$ 均成立,则计算生成群公钥 $Y=\prod_{i=1}^{n}Y_i^{\lambda_i}$,其中 $\lambda_i=\prod_{j\neq i,j\in PN}\frac{i}{j-i}$.

2)签名生成阶段(*SIGN*):

(a)承诺生成算法(*CommitGen*):签名者 P_i 随机选取随机 $r_i\xleftarrow{\$}Z_p$,计算并广播承诺 $R_i=g^{r_i}$,所以每一个参与者都可以计算得到群组集合承诺 $R=\prod_{i=1}^{n}R_i$.

(b)签名生成算法(*SignGen*):由每个签名者 P_i 计算单签名 $z_i=r_i+\lambda_i\cdot s_i\cdot c$,其中 $c=H_2(R,Y,m)$,$\lambda_i=\prod_{j\neq i,j\in PN}\frac{i}{j-i}$.

(c)聚合签名算法(*AggSign*):由签名收集者 *SA* 收集到足够的合法单签名 $|\{z_i\}|\geqslant t$ 后,计算门限签名 $z=\sum_{i\in PT}z_i$,最终的签名为 (R,z).

(d)聚合签名验证算法(*Verify*):若验证等式为 $g^z=R\cdot\prod_{i\in PT}Y_i^{\lambda i}=R\cdot Y^c$ 通过,则输出"1",反之输出"0".

3)切换场景阶段(*CHAN*):

(a)索引更新算法(*IndexGen*):为所有签名者集合成员重新选择索引,即 $\{P_i\rightarrow P_j, Y_i\rightarrow Y_j\}$.

(b):调用群公钥生成算法(*GroupKeyGen*)重新生成群公钥,之后执行签名生成阶段(*SIGN*)的算法生成门限签名.

4)添加成员阶段(*JOIN*):

(a)审核成员算法(*AuditUser*):通过布隆过滤器查询新加入成员是否是授权成员,若不通过,则拒绝新成员的加入申请;反之则将其公钥加入公钥集合 *PK*.

(b)索引更新算法(*IndexGen*):再为所有签名者集合选择新索引,即 $\{P_i\rightarrow P_j, Y_i\rightarrow Y_j\}$.

(c):调用群公钥生成算法(*GroupKeyGen*)生成新群公钥,继续执行签名生成程序(*SIGN*)的算法生成门限聚合签名.

2.2　安全模型

本方案的安全性包含正确性、鲁棒性和不可伪造性 3 个方面.

定义 1(正确性). 验证算法是正确的.

定义 2(鲁棒性). 聚合算法从合法单签名集合生成一个非法门限签名是计算不可行的. 鲁棒性的证明过程如下:

1)(*SetUp*):挑战者生成公开参数 *Par* 和一对公私钥 (sk^*,pk^*),并将公开参数和公钥 (Par,pk^*) 发给敌手 A.

2)(*Queries*):敌手 A 对消息 m 查询访问随机预言机,并

得到关于 m 的签名.

3)(*OutPut*):敌手 A 针对挑战消息 m^*,群公钥 Y,实际参与成员公钥集合 $\{Y_1,\cdots,Y_t\}$,系统全体签名者集合 $PN=\{P_1,\cdots,P_n\}$,门限签名实际参与者签名集合 $PT=\{P_1,\cdots,P_t\}$ 和伪造签名 (R,z^*).

敌手 A 攻破鲁棒性依赖于下列 3 个条件.

1) *DKG* 被正确执行,所有成员的公钥和群组公钥都被正确生成.

2) $Y^*=Y_k$ 且 $k\in PN$ 同时要求 $k\notin PT$.

3) $VerifyMul(m^*,\lambda_i,z^*,R)=0$,攻击者 A 以 Y^* 的身份查询 (m^*,k) 的签名获得 z^*,$PT=\{P_1,\cdots,P_t\}$ 是实际签名者的集合.

如果敌手 A 的成功概率是可忽略的,则本方案满足鲁棒性.

定义 3(不可伪造性). 攻击者 A 输出一个合法门限签名是计算不可行的. 不可伪造性的完整证明如下:

1)(*SetUp*):挑战者生成公开参数 Par 和一对公私钥 (sk^*,pk^*),并将公开参数和公钥 (Par,pk^*) 发给敌手 A.

2)(*Queries*):敌手 A 对消息 m 查询访问随机预言机,并得到关于 m 的签名.

3)(*OutPut*):敌手 A 针对挑战消息 m^*,群公钥 Y,实际参与成员公钥集合 $\{Y_1,\cdots,Y_t\}$,系统全体签名者集合 $PN=\{P_1,\cdots,P_n\}$,门限签名实际参与者签名集合 $PT=\{P_1,\cdots,P_t\}$ 和伪造签名 (R,z^*).

敌手 A 攻破不可伪造性性依赖于下列 3 个条件.

1) *DKG* 被正确执行,所有成员的公钥和群组公钥都被正确生成.

2) $Y^*=Y_k$ 且 $k\in PN$ 同时要求 $k\notin PT$.

3) $VerifyMul(m^*,J,\sigma^*,gpk)=1$,攻击者 A 不能询问过 (m^*,k) 的签名 z^*,其中 $PT=\{P_1,\cdots,P_t\}$ 是实际参与签名者的集合,z^* 是伪造的聚合签名.

如果敌手 A 的成功概率是可忽略的,则本方案满足不可伪造性.

3 多场景门限 Schnorr 签名方案

本文方案 MOSTS 主要由 4 个阶段组成:密钥分发(DKG),签名生成(SIGN),切换场景(CHAN),添加成员(JOIN). 下面将介绍这 4 个阶段.

3.1 密钥分发阶段(DKG)

(a)系统设置算法(*SetUp*):输入安全参数 λ,输出公开参数 $Par=(G,p,g,H_1,H_2)$,其中 G 为 p 阶的乘法循环群,g 是其生成元. $P=(P_1,\cdots,P_n)$ 是全体签名者集合,$H_1,H_2:\{0,1\}^*\to Z_p$ 是两个单向哈希函数.

(b)密钥生成算法(*KeyGen*):由每名签名者 P_i 随机挑选一个随机数 x_i 作为自己的秘密值份额,并计算自己的公钥 $Y_i=g^{x_i}$,并生成一个关于秘密值份额的零知识证明 $\{k_i\xleftarrow{\$}Z_p^*,K_i=g^{k_i},c_i=H_1(i,K_i,Y_i),s_i=k_i+c_i\cdot x_i\}$,收集公钥集合 $\{Y_1,\cdots,Y_n\}$ 并广播三元组 (Y_i,K_i,s_i).

(c)群公钥生成算法(*GroupKeyGen*):签名者 P_i 收到其他成员的三元组后,若全部零知识验证等式 $K_l=g^{s_l}\cdot Y_l^{-c_l}$ 均成立,则计算使用 *Lagrange* 插值生成群公钥 $Y=\prod_{i=1}^{n}Y_i^{\lambda_i}$,其中 $\lambda_i=\prod_{j\neq i,j\in P}\frac{i}{j-i}$.

3.2 签名生成阶段(SIGN)

预备工作:SA 表示签名聚合者,本身也是签名参与者之一,P 表示本次签名的签名者集合 $\{P_1,\cdots,P_\alpha\}$,α:$t\leqslant\alpha\leqslant n$,Y 表示群公钥,$x_i$ 表示各个签名者的密钥份额.

(a)承诺生成算法(*CommitGen*):签名者 P_i 随机选取随机 $r_i\xleftarrow{\$}Z_p$,计算并广播承诺 $R_i=g^{r_i}$,所以每一个参与者都可以计算得到群组承诺 $R=\prod_{i=1}^{n}R_i$.

(b)签名生成算法(*SignGen*):由每个签名者 P_i 计算单签名 $z_i=r_i+\lambda_i\cdot x_i\cdot c$,其中 $c=H_2(R,Y,m)$,$\lambda_i=\prod_{j\neq i,j\in P}\frac{i}{j-i}$,单签名将逐一发送给聚合者 SA.

(c)聚合签名算法(*AggSign*):*SA* 验证单签名合法性 $g^{s_i}=R_i\cdot Y_i^{c\lambda_i}$,若等式成立,则将其加入门限集合 $S\cup P_i$. 当收集到足够数量的合法单签名 $|\{z_i\}|\geqslant t$ 后,计算门限签名 $z=\sum_{i\in S}z_i$,最终的签名为 (R,z). 若签名通过验证等式,则输出"1",否则输出"0".

(d)聚合签名验证算法(*Verify*):若验证等式为 $g^z=R\cdot\prod_{i\in S}Y_i^{c\lambda_i}=R\cdot Y^c$ 通过,则输出"1",反之输出"0".

3.3 切换场景阶段(CHAN)

(a)索引更新算法(*IndexGen*):为所有签名者集合成员重新选择索引,即 $\{P_i\to P_j,Y_i\to Y_j\}$.

(b):调用群公钥生成算法(*GroupKeyGen*)重新生成群公钥,选择新的索引集和计算新群公钥的过程参考算法 2. 之后执行签名生成程序(*SIGN*)的算法生成门限签名.

算法 2. 切换场景

输入:所有成员的索引集合 *idxSet* 和公钥集合 *PK*

输出:集合的公钥 Y 和所有成员的 Lagrange 差值集合

```
1. function convertScene(PK,idxSet):
2.     change the signer index P_i→P_j
3.     for(j=0;j<n;j++):
4.         if(j≠i):
5.             λ_i=λ_i·(i/(j-i))
6.         end if
7.     end for
8.     then calculate {λ_1,…,λ_n}
9.     for(j=0;j<n;j++):
10.        Y=Y·Y_j^{λ_j}
11.    end for
12.    return{Y,λ_1,…,λ_n}
13. end function
```

3.4 添加成员阶段(JOIN)

(a)审核成员算法(*AuditUser*):通过布隆过滤器查询新加入成员是否是授权成员,若不通过,则拒绝新成员的加入申请;反之则将其公钥加入公钥集合 *PK*.

(b)索引更新算法(*IndexGen*):再为所有签名者集合选择新索引,即 $\{P_i\to P_j,Y_i\to Y_j\}$.

(c):调用群公钥生成算法(*GroupKeyGen*)生成新群公钥,重新选择新索引集和计算新群公钥的过程参考算法 3. 接

着继续执行签名生成程序(*SIGN*)的算法生成门限聚合签名.

算法 3. 加入成员

输入:新成员的公钥 Y_N

输出:群组的公钥 Y 和所有成员的 Lagrange 差值集合

```
1:function JoinNode(Y_N):
2.      access the Bloom filter use H(Y_N):
3.          if(the Bloom filter returns 0):
4.              拒绝成员加入申请
5.          else:
6.              PK∪Y_N and 集合长度加 1
7.      end if
8. end function
9. invoke algorithm 2
```

4　安全性证明

4.1　正确性

证明:MOSTS 方案的单签名验证等式和门限聚合签名验证等式都成立;

1)给定 z_j 和 Y,对消息 m,单签名 z_j 满足等式:

$$g^{z_j}=g^{r+cs_j\lambda_j}=g^r\cdot g^{cs_j\lambda_j}=R_j\cdot Y_j^{c\lambda_j}$$

2)在 S 和 Y 确定下,一定有等式成立:

$$\begin{aligned}g^z&=g^{\sum_{j\in S_2}z_j}=g^{\sum_{j\in S_2}r_j+c\sum_{j\in S_2}s_j\lambda_j}\\&=\prod_{j\in S_2}R_j\cdot\left(\prod_{j\in S_2}g^{s_j\lambda_j}\right)^c\\&=R\cdot Y^c\end{aligned}$$

4.2　鲁棒性

定理 1. MOSTS 方案满足鲁棒性.

证明:如果 MOSTS 方产生的案单签名都是合法的,假设存在攻击者 A 能攻破鲁棒性,那么表示 A 会输出一个非法的门限签名,即输出$(m^*,\{Y_i\}_{j\in T},\{z_j\},z,R,Y)$,这里 Y_k 为挑战公钥.若该方案输出了签名 z^* 是非法的,则验证等式 $g^z=R\cdot Y^c$ 被拒绝.意味着,收集的单签名集合$\{z_j\}$中存在某个合法签名 z_i,其也不能通过单签名验证等式 $g^z=R\cdot Y^c$.那么此时将出现两个情况:

1)若 $i=k$,即单签名 z_k 对应公钥 Y_k 是非法的.但其签名生成式为 $z_k=r_k+cs_k\lambda_k$,因此 z_k 不可能非法,此种情况不成立.

2)若 $i\neq k$,即单签名 z_k 非法,但单签名算法又都被正确执行,并且(*Combine*)算法执行前存在单签名的校验算法,因此这种情况不成立.

由上可知,门限 *Schnorr* 签名的生成,参与签名的用户无论是诚实的还是"*Byzantine*"叛徒"都可以在签名者集合中,而所有成员公钥、群公钥和承诺都完全公开,因此所成员都可校验他人的单签名以保证他人的签名合法.参与者生成单签名后,所有签名者都会执行分布式密钥分发协议,以得到每一个签名者的私钥、公钥和群组公钥,之后每个签名者都会选择一个随机数以计算得到承诺,并将其公开,在所有签名者都完成单签名之后,由签名汇总者收集所有单签名,在通过单签名验证等式的成员集合中随机选择 t 个成员以组成门限集合,之后完成聚合门限签名的工作.所有成员完成单签名生成算法之后会将自己的单签名广播给其他成员,由签名汇总者完成单签名验证工作,只有参与签名者的单签名通过等式,后续步骤才会执行,否则就会中断签名过程,,切换到(*Sign*)算法重新开始.只有在满足单签名通过验证的个数超过门限值 t 的情况下,才会进行(*Combine*)算法,这样保证了鲁棒性.

综上,攻击者 A 够破坏本方案的鲁棒性是计算不可行的,则 MOSTS 方案满足鲁棒性.

4.3　不可伪造性

定理 2. 在随机预言模型下,如果离散对数 DL 困难假设成立,则 MOSTS 方案满足不可伪造性.

证明:令 A 表示方案的攻击者;B 是以 A 为基础的一个算法;C 表示 DL 的挑战者.具体来说,C 模拟密钥分发和签名操作以及随机预言机查询中的诚实参与者;并且 C 接受挑战值 ω 并调用算法 B 以获得(z,z'),然后用于计算 ω 的离散对数.假设 $n=t$,A 控制 t－1 个参与者,C 模拟第 t 个诚实的参与者 P_t.接下来,模拟 C 和 A 的请求和应答过程:

B 选择群参数(G,p,g),G 是素数阶乘法循环群,以素数 q 为阶、g 为其生成元.B 挑选 Y^* 作为 P_t 的挑战公钥,并对 A 的哈希和签名查询作应答.C 的目标是求解离散对数问题,即从 $Y^*=g^{x^*}$ 中解出 x^*,这个过程利用 A 和 B 的交互过程,并利用广义分叉引理,找出分叉点,使得攻击者输出两个伪造签名,以得到结果.B 和 A 的交互过程如下:

1)对于 A 的 H_2 查询请求,B 输出一个随机向量 $f=(\rho,c_1,\cdots,c_{qH})$.

2)C 为诚实签名者 P_t 随机生成 1 个随机数 Y_t 作为诚实者的公钥,令 x_t 作为秘密值份额,有 $Y_t=g^{x_t}$作为挑战值,最后 A 还需要为 x_t 做零知识证明:$c_t,z\xleftarrow{S}Z_p;R=g^z\cdot Y_t^{-c_t};\sigma=(R,z)$.

由 F 控制的成员正常执行密钥分发协议,得到 $x_i,Y_i=g^{x_i}$,那么公钥为$\{Y_1,\cdots,Y_t\}$,群公钥为 $Y=\prod_{i=1}^n Y_i^{\lambda_i}$,$\lambda_i=\prod_{j=1,j\neq i}^n\frac{i}{j-i}$.

如果最后敌手 A 成功,那么 A 需要输出两个成功通过验证等式的门限签名 $\sigma=(R,z)$,$\sigma'=(R,z')$,根据 $z_i=r_i+\lambda_i\cdot x_i\cdot c$ 和 $z=\sum z_i$ 可得出 $z'-z=\sum\lambda_i s_i c$,所以有 $d\log(Y)=x=\frac{(z'-z)}{c}$,那么有 $x_t=d\log(Y)-\sum_{i=1}^{n-1}x_i=\frac{(z'-z)}{c}-\sum_{i=1}^{n-1}x_i$.

5　系统实现

在此节中将搭建以太坊平台实现仿真应用场景的过程,以智能合约对门限签名完成上链操作、使用公钥对门限签名合法性完成验证.再进一步的模拟两个过程,一是模拟切换场景变换,二是模拟一个新成员申请加入签名集合.

5.1　环境配置

本文使用以太坊(Ethereum)作为实验平台.以太坊要求用 Solidity 编写合约[19],使之后合约代码会被编译为 EVM 可执行的字节码,并会将结果记录在区块链上[20].本文方案在个人电脑上测试,所以选择一个兼顾以太坊开发和测试的轻量级个人区块链,即 Ganache,它部署在本地具有多种优势,其默认支持自动挖矿,交易几乎即时完成,可以极大提高开发效率.本文在 Windows11 为基础搭建以太坊的开发环境;智能合约以 Solidity 编写,链下节点以 Java 语言编写.具体来讲:使用 npm 安装 Ganache 命令行版本,在 IDEA 中通过 Ma-

ven 导入 Web3j 库链接 Ganache 节点,使用 IDEA 创建一个 Java 项目,用 Java 编写链下代码,用 Solidity 编写链上代码,最后调用 Ganache 节点接口,监控交易和合约状态.

5.2 汇款转账和电子投票应用场景分析

假设上述方案应用在于两家公司 E 和 F 的运营中,公司 E 和 F 的领导层是同一批人,即董事会成员相同. MOSTS 方案的应用场景主要涉及汇款转账和电子投票,合法的转账和投票操作都需要董事会半数以上成员同意,那么这可以看作是一个$(\frac{n}{2},n)$门限签名. MOSTS 方案的示意流程图见图 1.

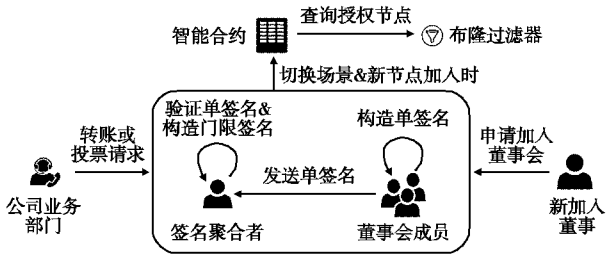

图 1 MOSTS 流程图

Fig. 1 MOSTS flowchart

操作 1(汇款转账). 当 E 公司某业务部门申请进行业务汇款操作时,需半数以上董事会成员同意,董事会成员可以根据自己意愿决定同意或不同意,同意的成员选择生成自己的单签名,并将发送给聚合者(统计秘书). 统计秘书将收集到的单签名先进行合法性验证,如果合法单签名数量低于门限值,则通知转账业务部门"董事会不同意此操作",签名中断. 否则,基于聚合签名算法($AggSign$),构造最终门限签名,并通知转账业务部门"董事会同意此操作",任何董事会成员都可以验证门限签名的正确性.

操作 2(电子投票). 如果 F 公司业务部门新发过来一个决议请求,需要董事会投票同意,此时 E 公司业务部门的汇款请求正在执行,那么只需调用智能合约,即算法 2 为董事会成员重新选择索引集合,并计算得到新的群组公钥. 之后各董事会成员按 MOSTS 方案程序算法生成,发送单签名以及聚合者验证单签名、生成门限签名的若干步操作.

操作 3(成员变化). 如果此时有一个新的成员申请加入董事会,那么根据算法 3 会根据申请者的公钥计算哈希值,并访问布隆过滤器,若布隆过滤器返回不存在,则申请成员为非授权者,可以拒接其加入. 反之,再进一步比较其公钥避免布隆过滤器的误判. 当申请成员加入董事会之后,会更新索引集合以及公钥集合 PK,原董事会其他成员不需要更新其密钥和公钥,节约了更新的时间和通信代价. 之后各董事会成员继续执行 MOSTS 方案.

5.3 切换场景性能分析

本文提出的 MOSTS 方案是基于 Schnorr 签名方案提出的,所以选择和同样基于 Schnorr 的 FROST 门限方案,且 MOSTS 方案的一个创新点是适用于链上多元场景,这一点参考了 MTS 的多元宇宙(场景)机制. 虽然近年来提出了不少带复杂特性的门限签名,但这些特性和多元场景切换无关,所以在本小节中主要和 FROST 方案和比较简洁的 MTS 方案进行比较,整体签名流程参考 5.2 小节定义的应用场景流程.

关于签名时间代价,MOSTS 方案与 FROST 方案、MTS 方案对比结果如图 2 所示,可以看出 3 种方案的时间代价随着实际签名者集合的增大而线性增加,这是因为签名者集合越大,需要生成的单签名数量也越多. 随之,单签名验证、密钥份额的零知识证明、分布式密钥分发、聚合算法的计算代价也越大. 并且从图 2 可以看出,MOSTS 方案在相等签名者集合明显优于 MTS 方案和 FROST 方案,这是因为 MTS 方案是基于 BLS 设计的,双线性配对的计算量较大,而且在 MTS 方案中还使用了密钥向量的概念,并且每一个签名者需要定义一个场景 U,引入场景 U 的主要目的是为了更方便成员同时在多个场景中进行达成共识的活动,但是代价就是成员需要在场景中定义多个公共参数和多步运算,这无疑会增加了计算代价. FROST 方案中定义了一个预处理阶段,在此阶段中会生成一些可以复用的随机数承诺,并且为了防止伪造,这些已生成的随机数承诺要跟后续的签名消息和签名集绑定,这也显著增加方案的计算代价,但是 FROST 方案是基于 Schnorr 设计的,所以 FROST 的时间代价虽然比 MOSTS 高,但是明显比基于 BLS 的 MTS 低,这在图 2 中可以得到.

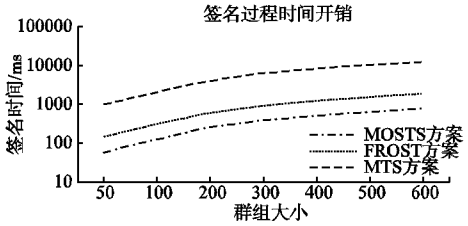

图 2 签名时间代价对比

Fig. 2 Comparison of signature time overhead

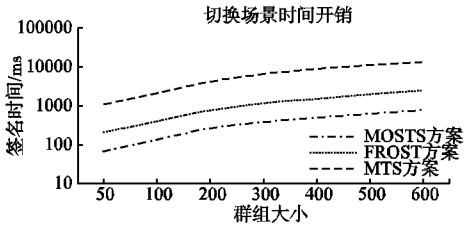

图 3 切换场景时间开销对比

Fig. 3 Comparison of time cost for switching scenes

在图 3 中主要比较了 3 种方案在切换场景时的时间开销,从图中可以看出 3 种方案本文提出的 MOSTS 依然在最优的. 其中 MTS 方案中"场景 U"的定义使得该方案可以便捷的切换场景,在不同的场景中只有极少数可忽略的公共参数需要重建,所以在该方案中切换场景基本是非交互式的,时间代价极低,图 2 和图 3 的 MTS 时间代价几乎相同. 但是由于该方案是基于 BLS 设计的门限签名,所以在计算代价上依然是 3 种方案中最高的. 而 FROST 方案虽然对场景无需作处理,需要重新进行密钥分发,不过其基于 Schnorr 签名的设计特点,使得其时间成本优于 MTS,而高于 MOSTS. MOSTS 方案给出了场景切换的算法,例如从公司 E 的业务切换到公司 F 的业务请求时,无需重新进行密钥分发,只需重新选择索引集即可,切换场景较为便捷. 所以在场景切换方面,MOSTS 方案是最优的.

在图 4 中的对比主要聚焦在成员变动时,当有一个成员申请成为签名者时,从图 4 中可以看出,MOSTS 方案依然是最优的,虽然为增加身份判定功能,需要多访问一次布隆过滤器,但是由于布隆过滤器是基于位映射设计的,而且是运行在 Redis 内存上的,所以代价可以忽略不计,又由于不需要重新执行密钥分发过程,所以时间代价依然是最小. 对于 FROST 方案而言,加入一个新成员和从公司 E 的业务切换到公司 F 的业务,都需要执行密钥分发,所以图 4 和图 3 的 FROST 的时间代价几近相同. 而在 MTS 方案中需要对新成员做的工作有,定义一个包含新成员的新场景,还需要生成新成员的密钥向量,最重要的是要更新整个方案的场景,所以时间开销上显著高于图 2 和图 3. 因此,基于 Schnorr 签名的 MOSTS 方案在此应用中的时间代价还是最优的.

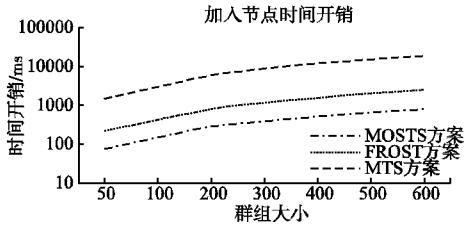

图 4　加入成员时间开销对比

Fig. 4　Comparison of node join time overhead

由上面的实验分析可知,MOSTS 方案适用于链上多元场景,拥有更低的多元场景切换成本.

6　结束语

本文结合 Schnorr 签名方案的低计算复杂度优势,提出一个适用于链上多元场景的门限方案. 当链上应用场景变化、或签名者集合有新成员加入时,原成员不需要重新选择密钥份额和计算公钥,只需要更新成员的索引,从而达到降低成员的计算压力、和频繁进行密钥分发过程带来的时间和计算开销. 此外,本文通过引入布隆过滤器设计一种快速定位机制,即系统会将授权成员的公钥哈希插入布隆过滤器,当有新成员加入集合时,通过布隆过滤器查询,若返回“不存在”,则直接拒绝新成员,反之再用成员公钥进一步验证,防止布隆过滤器的误判. 另外,文中证明了 MOSTS 方案的鲁棒性,并基于离散对数假设和随机预言模型证明了 MOSTS 方案的不可伪造性,最后,通过在本地以太坊测试网和假设现实场景下,证明了 MOSTS 方案在签名交易开销、切换场景开销、成员加入开销上都具有优越性.

References:

[1] Abdelhamid M, Sliman L, Ben D R, et al. A review on blockchain technology, current challenges, and AI-driven solutions[J]. ACM Computing Surveys, 2024, 57(3): 1-39, doi: 10.1145/3700641.

[2] Shamir A. How to share a secret[J]. Communications of the ACM, 1979, doi: 10.1145/359168.359176.

[3] Desmedt Y. Threshold cryptosystems[J]. Springer-Verlag New York, Inc, 1989: 307-315, doi: 10.1007/3-540-57220-1_47.

[4] Pedersen T P. A threshold cryptosystem without a trusted party[J]. Springer Berlin Heidelberg, 1991: 522-526, doi: 10.1007/3-540-46416-6_47.

[5] Gennaro R, Jarecki S, Krawczyk H, et al. Secure distributed key generation for discrete-log based cryptosystems[C]//International Conference on the Theory and Applications of Cryptographic Techniques, Springer, Berlin, Heidelberg, 1999: 51-83, doi: 10.1007/3-540-48910-X_21.

[6] Shoup V. Practical threshold signatures[C]//International Conference on Theory & Application of Cryptographic Techniques, Lecture Notes in Computer Science, 2000: 207-220, doi: 10.1007/3-540-45539-6_15.

[7] Boneh D, Lynn B, Shacham H. Short signatures from the weil pairing[J]. Journal of Cryptology, 2004, 17(4): 297-319, doi: 10.1007/s00145-004-0314-9.

[8] Zhang Y, Yu J, Hao R, et al. Enabling efficient user revocation in identity-based cloud storage auditing for shared big data[J]. IEEE Transactions on Dependable and Secure Computing, 2020, 17(3): 608-619, doi: 10.1109/TDSC.2018.2829880.

[9] Maxwell G, Poelstra A, Seurin Y, et al. Simple schnorr multi-signatures with applications to bitcoin[J]. Designs Codes and Cryptography, 2019, 87(4): 2139-2164, doi: 10.1007/s10623-019-00608-x.

[10] Wei J, Liu W, Hu X. Forward-secure threshold attribute-based signature scheme[J]. The Computer Journal, 2015, 58(10): 2492, doi: 10.1093/comjnl/bxu095.

[11] Castagnos G, Catalano D, Laguillaumie F, et al. Bandwidth-efficient threshold EC-DSA[C]//IACR International Conference on Public-Key Cryptography, 2020: 266-296, doi: 10.1007/978-3-030-45388-6_10.

[12] Yan Y, Zhao Y, Gao W, et al. Lattice-based threshold, accountable, and private signature[C]//Cryptographers' Track at the RSA Conference, 2024: 249-274.

[13] Komlo C, Goldberg I. FROST: flexible round-optimized schnorr threshold signatures[C]//Selected Areas in Cryptography(SAC), 2021: 27-48, doi: 10.1007/978-3-030-81652-0_2.

[14] Baird L, Garg S, Jain A, et al. Threshold signatures in the multiverse[C]//IEEE Symposium on Security and Privacy(SP), 2023: 1454-1470, doi: 10.1109/SP46215.2023.10179436.

[15] Li J, Kassem M. Applications of distributed ledger technology (DLT) and blockchain-enabled smart contracts in construction[J]. Automation in Construction, 2021, 132: 103955, doi: 10.1016/j.autcon.2021.103955.

[16] Pointcheval D, Stern J. Security arguments for digital signatures and blind signatures[J]. Journal of Cryptology, 2000, 13(3): 361-396, doi: 10.1007/s001450010003.

[17] Bellare M, Neven G. Multi-signatures in the plain public-key model and a general forking lemma[J]. Association for Computing Machinery, 2006: 390-399, doi: 10.1145/1180405.1180453.

[18] Bagherzandi A, Cheon J H, Jarecki S. Multisignatures secure under the discrete logarithm assumption and a generalized forking lemma[C]//Proceedings of the ACM Conference on Computer and Communications Security(CCS), 2008: 449-458, doi: 10.1145/1455770.1455827.

[19] Syta E, Tamas I, Visher D, et al. Keeping authorities "honest or bust" with decentralized witness cosigning[C]//IEEE Symposium on Security and Privacy(SP), 2016: 526-545, doi: 10.1109/SP.2016.38.

[20] Buterin V, Conner V, Dudley R, et al. EIP-1559: fee market change for ETH 1.0 chain[EB/OL]. Ethereum Improvement Proposals, https://eips.ethereum.org/EIPS/eip-1559, 2025-04-04.

[21] Pino R D, Niot G. Finally! A compact lattice-based threshold signature[C]//28th IACR International Conference on Practice and Theory of Public-Key Cryptography(PKC), 2025: 169-199.

[22] Fischlin M, Mitrokotsa A, Tomy J. BUFFing threshold signature schemes[C]//28th IACR International Conference on Practice and Theory of Public-Key Cryptography(PKC), 2025: 137-168.

[23] Ji Y, Zhang R, Tao Y, et al. Designated confirmer threshold signature and its applications in blockchains[J]. Cybersecurity, 2024, 7(1), doi: 10.1186/s42400-024-00256-2.