

可编辑且可追责的区块链方案

赵晓琦, 张正昊, 李 勇

北京交通大学电子信息工程学院 北京 中国 100044

摘要 随着区块链所承载信息种类和应用场景的不断增加,出于信息监管、隐私保护、数据更新等方面的目的,需要对记录在区块链上的数据进行删除、更新等操作。针对这些需求,基于公开可验证秘密共享、零知识证明、变色龙哈希等技术,本文提出了一个可编辑且可追责的区块链方案。在本方案中,变色龙哈希函数将替换原始区块链中的哈希函数,由领导者将变色龙哈希的陷门密钥通过公开可验证秘密共享分发给多个用户,从而避免由某一方独自持有陷门密钥所带来的中心化问题。持有密钥份额的用户将验证网络中出现的编辑请求,并对编辑请求进行投票。当大多数用户同意进行编辑时,将通过哈希排序的方式在用户中选举出编辑者,编辑者将恢复出变色龙哈希密钥进而进行编辑。为了实现编辑过程的可追责性,全体用户都可以对编辑后的内容进行验证,监管方可以实现相关责任方的追责。本方案还通过零知识证明技术,实现了在密钥分发与验证追责阶段,可以验证密钥份额正确性。安全性分析表明方案满足陷门安全性、可编辑性、可追责性,且陷门子密钥分发时不需要经过秘密通道。仿真实验结果表明,在系统参数已经生成完毕的情况下,方案的运行时间均为毫秒量级。而参数生成算法仅执行一次,而且可以预先执行,因此参数生成算法对方案的整体运行效率影响不大,本方案依然具有较好的运行效率。

关键词 区块链; 可编辑性; 可追责性; 公开可验证秘密共享

中图分类号 TP391 **DOI号** 10.19363/J.cnki.cn10-1380/tn.2022.09.02

An Editable and Accountable Blockchain Scheme

ZHAO Xiaoqi, ZHANG Zhenghao, LI Yong

School of Electronic and Information Engineering, Beijing Jiaotong University, Beijing 100044, China

Abstract With the continuous increase of information types and application scenarios carried by the blockchain, for the purposes of regulation compliance, privacy protection, and data update, it is necessary to delete and update the data recorded on the blockchain. In response to these needs, based on technologies such as publicly verifiable secret sharing, zero-knowledge proof, and chameleon hash, this paper proposes an editable and accountable blockchain scheme. In this scheme, the chameleon hash function will replace the hash function in the original blockchain, and the leader will distribute the trapdoor key of the chameleon hash to multiple users through publicly verifiable secret sharing, thereby avoiding centralization problems caused by one party holding trapdoor keys alone. The user holding the key share will verify the edit request that appears on the network and vote on the edit request. When most users agree to edit, editors will be elected from the users by hash sorting, and the editors will restore the chameleon hash key for editing. In order to realize the accountability of the editing process, all users can verify the edited content, and the regulator can realize the accountability of the responsible party. This scheme also uses zero-knowledge proof technology to verify the correctness of the key share during the key distribution and verification accountability phase. Security analysis shows that the scheme satisfies the security, editability, and accountability of the trapdoor, and the trapdoor subkey does not need to pass through a secret channel when it is distributed. The simulation results show that when the system parameters have been generated, the running time of the scheme is in the order of milliseconds. The parameter generation algorithm is executed only once and can be executed in advance. Therefore, the parameter generation algorithm has little effect on the overall efficiency of the scheme, and the scheme still has better efficiency.

Key words blockchain; editable; accountability; publicly verifiable secret sharing

1 前言

区块链技术在信息监管、隐私保护、数据更新等方面存在切实的编辑需求^[1]。从信息监管角度出发,

由于互联网上的每个人都可以写到无许可的区块链,所以一些恶意用户可能滥用发布任意交易消息的能力^[2],存储在分类帐中的数据可能是敏感、有害或非法的。例如,区块链包含泄露的私钥^[3],甚至是违法

通讯作者: 李勇, 博士, 副教授, liyong@bjtu.edu.cn。

本课题得到国家重点研发计划资助(No. 2018YFC0832300, No. 2018YFC0832303)资助。

收稿日期: 2021-09-24; 修改日期: 2021-12-02; 定稿日期: 2022-07-15

乱纪的谣言等。这使得区块链在涉及政府记录和社交媒体等数据的领域中更广泛地应用受到限制。

从隐私保护角度出发, 欧盟于 2018 年 5 月出台通用数据保护条例(General data protection regulation, GDPR), 它不再有利于当前的区块链去记录个人数据, 比如比特币和以太坊。GDPR 将“被遗忘的权利”作为关键的数据主体权利^[4], 即数据所有者有权利从控制者处去除私人信息, 所以在遵守有关私人数据的新法规的同时如何促进区块链的广泛使用已成为一个新的挑战。

从数据更新角度出发, 区块链不可篡改这一特性限制了区块链在多领域的广泛应用, 比如金融领域、版权领域。一旦在区块链中创建了记录或者块, 传统区块链的特性保证其不可更改, 但在某些情况下应进行修改或者删除内容, 必须有一定的灵活性来应对用户的错误或事故, 以保护系统的完整性。

因此, 在确保区块链安全可信等优势的同时, 对数据进行可控编辑具有现实意义。实现区块链的可编辑性和可追责性的结合, 将进一步促进区块链实现更广泛的落地应用。为了满足区块链在信息监管、隐私保护、数据更新等方面存在的编辑需求, 本文提出可编辑且可追责的区块链方案, 在数据失效或错误时, 只要超过阈值数节点同意便可实现区块数据的合法编辑, 同时保持区块链接不变, 且可对数据合法性进行验证。若验证不通过, 则可进行相关责任方的追责。

2 相关工作

在可编辑区块链研究中, 埃森哲公司获得了可编辑区块链专利, 通过使用变色龙哈希来对链中的某个区块进行更改, 但存在编辑权中心化的问题。Ateniese 等人^[5]提出可修改区块链的概念, 该方案是基于变色龙哈希的许可区块链方案^[6], 在编辑区块时, 使用带陷门的变色龙哈希函数, 因此可以有效地计算出变色龙哈希函数的碰撞, 并在任意修改后保持哈希一致性。李佩丽等人^[7]对可修改区块链技术进行改进, 利用秘密共享方案, 将陷门分配给系统各节点, 当且仅当阈值节点联合时方可恢复陷门, 进而编辑。但是该方案需要秘密通道, 将陷门子密钥发送给链中的各个成员, 而且, 当修改后的数据出现问题时并没有考虑追责。Ashritha^[8]提出使用增强型变色龙哈希的可编辑区块链, 在该方案中, 使用了非线性秘密共享方案来防止恶意参与者共享虚假份额, 每次修改需经过发布者同意。

Marsalek 等人^[9]提出基于共识投票机制的可编辑区块链, 在投票过程需要各个节点验证编辑是否合法, 进而达成共识, 但编辑颗粒较粗, 是面向整个区块的修改。任艳丽等人^[10]基于抗碰撞哈希函数, 提出可删除区块链系统, 利用安全的门限签名方案, 在超过门限值的节点同意时, 可对单个区块的全部交易数据进行删除。但该方案无法删除指定交易数据, 也无法进行数据修改。为了支持细颗粒的可控区块链编辑, Derler 等人^[11]提出基于策略的变色龙哈希的概念, 其中满足策略的任何人都可以找到给定哈希的碰撞实现区块链的编辑。Puddu 等人^[12]提出了一种可编辑的区块链, 称为 μ 链。在 μ 链中, 交易发送方可以加密交易的不同版本, 用“变异”表示, 解密密钥在矿工之间秘密共享, 交易的未加密版本被视为活动交易。但是, 恶意用户可能会错误编辑, 甚至破坏交易的稳定性。此外, 在通过多方计算协议重建解密密钥时, μ 链还面临可伸缩性问题。

Deuber 等人^[13]提出了第一个在无许可情况下的可编辑区块链协议, 它不依赖繁重的加密协议或其他信任假设。一旦用户提出了修订建议, 该协议便开始基于共识的投票期, 并且只有在获得足够的投票批准修订之后, 该版本才会在区块链上执行。Thyagarajan 等人^[14]提出了通用协议 Reparo 来编辑区块链, 通过引入外部数据结构来存储区块内容, 保持区块结构不变。但是, 新加入的用户必须在投票期内检查所有区块, 以验证对区块链的修改。并且投票期长, 难以实际应用。任艳丽等人^[15]提出可修改的区块链方案, 引入机动因子, 从而保证签名子块不变, 满足哈希一致性, 但该方案建立在基于空间证明的区块链, 不适用于普通的区块链结构。Jia 等人^[16]提出了具有可撤销子密钥的有状态变色龙哈希(Stateful chameleon hash with revocable subkey, sCHRS), 进而在此基础上提出了一种新型的可编辑区块链构造。在该构造中存在一个半可信的监管方对编辑行为进行监管, 避免出现恶意修改的情况。同时, 用户也可以监督监管方是否正确行使权力。该方案实现了一定程度的监管功能, 但是没有研究当确实出现非法行为之后, 如何进行追责的问题。

3 系统框架

可编辑且可追责的区块链系统由三类角色组成: 用户、领导者、监管方。

(1) 用户(User): 联盟区块链中的成员, 即交易的参与方;

(2) 领导者(Leader): 联盟链中的管理者, 也是

陷门子密钥的分发者;

(3) 监管方(Regulator): 联盟链中的监管者, 具

有监管和追责的功能。

系统框架如图 1 所示:

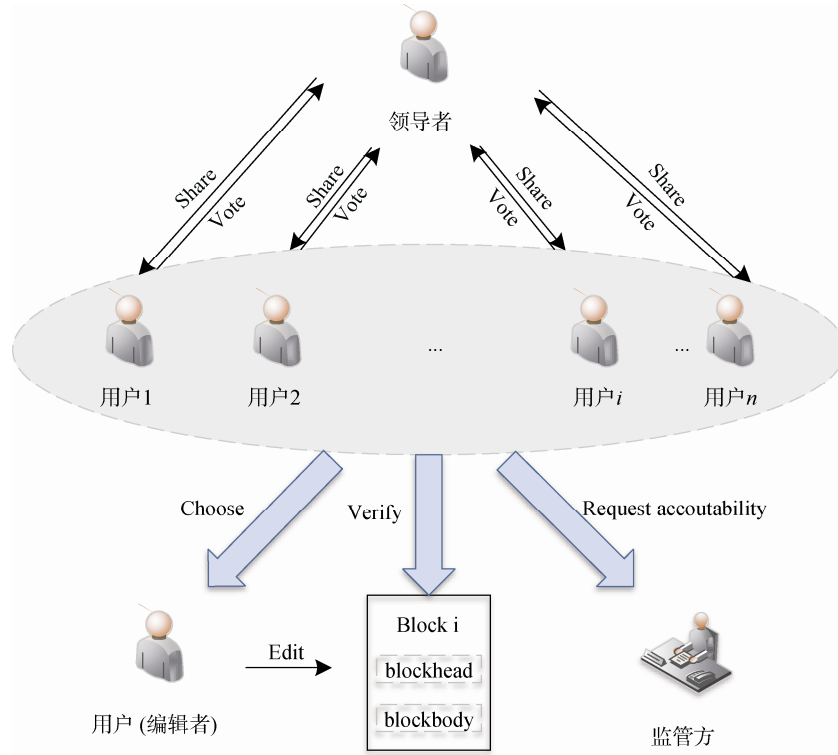


图 1 系统框架

Figure 1 System framework

具体过程如下:

- (1) 领导者分发陷门密钥给各用户;
- (2) 用户发起编辑请求, 其他用户对请求投票, 并将投票结果发送给领导者;
- (3) 在系统用户中, 选出具有编辑权的编辑者;
- (4) 编辑者对需要修改的数据进行编辑;
- (5) 用户对编辑后的区块进行验证, 是否可被接受;
- (6) 若编辑后的区块未通过验证, 则向监管方请求追责。

4 可编辑且可追责的区块链方案

4.1 方案定义

可编辑且可追责的区块链方案由秘密分享、投票、寻找编辑者、编辑、验证追责 5 个阶段构成。

(1) 在秘密分享阶段, 该阶段包含 4 个算法:

1) $\text{Setup}(1^\lambda) \rightarrow (\text{pubpara}, \text{secpa})$: 参数设置算法, 以安全参数 1^λ 作为输入, 输出公共参数 pubpara 和私有参数 secpa 。

2) $\text{Share}(S) \rightarrow (Y_1, Y_2, Y_3, \dots, Y_n, C_0, C_1, \dots, C_{t-1}, \pi_l)$: 密钥分发算法, 由 Leader 执行, 以陷门 S 作为输入,

输出 n 份加密后的陷门子密钥、 t 个承诺值和证明 π_l 。

3) $\text{Verifykey}(Y_1, Y_2, Y_3, \dots, Y_n, C_0, C_1, \dots, C_{t-1}, \pi_l) \rightarrow b$: 验证子密钥算法, 由用户执行, 以 n 份加密后的陷门子密钥、 t 个承诺值和证明 π_l 作为输入, 若验证成功, 则输出 1, 证明子密钥分发成功; 若验证失败, 则输出 0, 表明分发的子密钥错误。

4) $\text{Dec}(Y_i, \text{upk}_i, \text{usk}_i) \rightarrow (S_i, \pi_{u_i})$: 解密子密钥算法, 由用户执行, 以用户收到的加密子密钥 Y_i , 用户的公钥 upk_i 和用户的私钥 usk_i 作为输入, 输出解密后的子密钥 S_i 和相对应的证据 π_{u_i} 。

(2) 在投票阶段, 该阶段包含 3 个算法:

1) $\text{Req}(m, m') \rightarrow (\text{req})$: 请求编辑算法, 由用户执行, 以编辑的原内容 m 、编辑后的新内容 m' 作为输入, 输出编辑请求 req 。

2) $\text{Vote}(\text{req}, \text{rd}_i, b, \text{lpk}) \rightarrow (\text{enc}_i)$: 投票算法, 由用户执行, 以编辑请求 req 、用户选择的随机数 rd_i 、用户投票意见 b 、Leader 的公钥 lpk 作为输入, 其中, 若用户同意修改, 则 b 为 1, 输出加密后的投票信息 enc_i , 并将其发送至 Leader。

3) $\text{Vresult}(\text{enc}_i, \text{lsk}) \rightarrow (\text{dec}_i, \text{result}, \text{rd}_{\text{sum}})$: 结果算法, 由 Leader 执行, 以用户加密的投票信息 enc_i 、

Leader 私钥 lsk 作为输入, 输出明文 dec_i 、投票结果 $result$ 和计算所得的随机数总和 rd_{sum} 。

(3) 寻找编辑者阶段, 该阶段包含寻找编辑者算法。

$Find(rd_{sum}, upk_i) \rightarrow (hash_{min}, upk_{min})$: 寻找编辑者算法, 由用户执行, 以 Leader 所广播的随机数总和 rd_{sum} 、用户的公钥 upk_i 作为输入, 每个用户计算个人公钥和 rd_{sum} 的哈希值广播到链中。经排序后, 输出最小的哈希值 $hash_{min}$ 和所对应的用户公钥 upk_{min} , 与最小哈希值对应的用户即为具有编辑权的编辑者。

(4) 在编辑阶段, 包含两个算法:

1) $Recover(S_1, S_2, \dots, S_i(i \geq t)) \rightarrow (S)$: 恢复密钥算法, 由用户中的编辑者执行, 以子密钥 $S_1, S_2, \dots, S_i(i \geq t)$ 作为输入, 输出陷门密钥 S 。

2) $Edit(S, m, m', r) \rightarrow (r')$: 编辑算法, 由用户中的编辑者执行, 以陷门密钥 S 、编辑的原内容 m 、编辑后的新内容 m' 、以及变色龙哈希所使用的随机数 r 作为输入, 输出计算所得的新随机数 r' , 以保证哈希一致性。

(5) 验证追责阶段, 该阶段包含验证算法。

$Verify(dec_i, rd_{sum}, upk_{min}, hash_{min}, m, m', r, r', \pi_{u_i}) \rightarrow (b_1, b_2, b_3, b_4)$: 验证算法, 由用户执行, 若验证投票和随机数正确, 则 b_1 输出 1, 若不正确, 则输出 0; 若验证编辑区块信息的用户具有编辑权, 则 b_2 输出 1, 若不正确, 则输出 0; 若验证哈希一致性成立, 则 b_3 输出 1, 若不正确, 则输出 0; 若验证链中用户发送的子密钥值正确, 即验证 π_{u_i} 为真, 则 b_4 输出 1, 若不正确, 则输出 0。

4.2 安全模型

定义 1. 陷门安全性。如果满足下列条件, 则任何 $t-1$ 个用户利用他们的子密钥都不能恢复出陷门值, 任何 t 个用户利用他们的子密钥都可以恢复陷门值。

$$\Pr \left[\begin{array}{l} \text{Setup}(1^\lambda) \rightarrow (pubpara, secpa) \\ \text{Share}(S) \rightarrow (Y_1, Y_2, Y_3, \dots, Y_n, C_0, C_1, \dots, C_{t-1}, \pi_l) \\ \text{Verifykey}(Y_1, \dots, Y_n, C_0, C_1, \dots, C_{t-1}, \pi_l) \rightarrow 1 \\ \text{Dec}(Y_i, upk_i, usk_i) \rightarrow (S_i, \pi_{u_i}) \\ \text{Req}(m, m') \rightarrow req \\ \text{Vote}(req, rd_i, b, lpk) \rightarrow enc_i \\ \text{Vresult}(enc_i, lsk) \rightarrow (dec_i, result, rd_{sum}) \\ \text{Recover}(S_1, \dots, S_i(i < t)) \rightarrow S \end{array} \right] \geq 1 - v(\lambda)$$

其中, $v(\lambda)$ 为可忽略函数。

或者

$$\Pr \left[\begin{array}{l} \text{Setup}(1^\lambda) \rightarrow (pubpara, secpa) \\ \text{Share}(S) \rightarrow (Y_1, Y_2, Y_3, \dots, Y_n, C_0, C_1, \dots, C_{t-1}, \pi_l) \\ \text{Verifykey}(Y_1, \dots, Y_n, C_0, C_1, \dots, C_{t-1}, \pi_l) \rightarrow 1 \\ \text{Dec}(Y_i, upk_i, usk_i) \rightarrow (S_i, \pi_{u_i}) \\ \text{Req}(m, m') \rightarrow req \\ \text{Vote}(req, rd_i, b, lpk) \rightarrow enc_i \\ \text{Vresult}(enc_i, lsk) \rightarrow (dec_i, result, rd_{sum}) \\ \text{Recover}(S_1, \dots, S_i(i \geq t)) \rightarrow S \end{array} \right] \geq 1 - v(\lambda)$$

即任何 $t-1$ 个用户利用他们的子密钥不能恢复出陷门值的概率接近 1; 任何 t 个用户利用他们的子密钥恢复出陷门值的概率同样接近 1。

定义 2. 可编辑性。如果方案满足下列条件, 则编辑者可以对请求编辑的内容进行修改, 具有可编辑性。

$$\Pr \left[\begin{array}{l} \text{Setup}(1^\lambda) \rightarrow (pubpara, secpa) \\ \text{Share}(S) \rightarrow (Y_1, Y_2, Y_3, \dots, Y_n, C_0, C_1, \dots, C_{t-1}, \pi_l) \\ \text{Verifykey}(Y_1, \dots, Y_n, C_0, C_1, \dots, C_{t-1}, \pi_l) \rightarrow 1 \\ \text{Dec}(Y_i, upk_i, usk_i) \rightarrow (S_i, \pi_{u_i}) \\ \text{Req}(m, m') \rightarrow req \\ \text{Vote}(req, rd_i, b, lpk) \rightarrow enc_i \\ \text{Vresult}(enc_i, lsk) \rightarrow (dec_i, result, rd_{sum}) \\ \text{Find}(rd_{sum}, upk_i) \rightarrow (hash_{min}, upk_{min}) \\ \text{Recover}(S_1, S_2, \dots, S_i(i \geq t)) \rightarrow S \\ \text{Edit}(S, m, m', r) \rightarrow r' \end{array} \right] \geq 1 - v(\lambda)$$

即在编辑者恢复出陷门密钥后, 可计算出新随机数进行编辑的概率接近于 1。

定义 3. 可追责性。如果方案满足下列条件, 则可以进行追责。

令事件 B 为:

$$B = \left[\begin{array}{l} \text{Setup}(1^\lambda) \rightarrow (pubpara, secpa) \\ \text{Share}(S) \rightarrow (Y_1, Y_2, \dots, Y_n, C_0, \dots, C_{t-1}, \pi_l) \\ \text{Verifykey}(Y_1, Y_2, \dots, Y_n, C_0, \dots, C_{t-1}, \pi_l) \rightarrow 1 \\ \text{Dec}(Y_i, upk_i, usk_i) \rightarrow (S_i, \pi_{u_i}) \\ \text{Req}(m, m') \rightarrow req \\ \text{Vote}(req, rd_i, b, lpk) \rightarrow enc_i \\ \text{Vresult}(enc_i, lsk) \rightarrow (dec_i, result, rd_{sum}) \\ \text{Find}(rd_{sum}, upk_i) \rightarrow (hash_{min}, upk_{min}) \\ \text{Recover}(S_1, S_2, \dots, S_i(i \geq t)) \rightarrow S \\ \text{Edit}(S, m, m', r) \rightarrow r' \end{array} \right]$$

满足:

$$\Pr[B: \text{Verify}(dec_i, rd_{sum}, upk_{min}, hash_{min}, m, m', r, r', \pi_{u_i}) \rightarrow (1, 1, 1, 1)] \geq 1 - v(\lambda)$$

即编辑块符合编辑规则, 可被区块链接受, 不需要进行追责。

或者

$$\Pr[B: \text{Verify}(dec_i, rd_{sum}, upk_{min}, hash_{min}, m, m', r, r', \pi_{u_i}) \rightarrow (0, 1, 1, 1)] \geq 1 - v(\lambda)$$

即当 Leader 投票结果和所计算的随机数不正确时, 用户可验证到这一情况, 向监管者汇报对 Leader 进行追责的概率接近于 1。

或者

$$\Pr[B: \text{Verify}(dec_i, rd_{\text{sum}}, upk_{\text{min}}, hash_{\text{min}}, m, m', r, r') \\ \pi_{u_i} \rightarrow (1, 0, 1, 1)] \geq 1 - v(\lambda)$$

即当编辑者不具有编辑权时, 用户可验证到这一情况, 向监管者汇报对编辑者进行追责的概率接近于 1。

或者

$$\Pr[B: \text{Verify}(dec_i, rd_{\text{sum}}, upk_{\text{min}}, hash_{\text{min}}, m, m', r, r') \\ \pi_{u_i} \rightarrow (1, 1, 0, 1)] \geq 1 - v(\lambda)$$

即当编辑前后哈希不一致, 且子密钥分享正确时, 用户可验证到这一情况, 向监管者汇报对编辑者进行追责的概率接近于 1。

或者

$$\Pr[B: \text{Verify}(dec_i, rd_{\text{sum}}, upk_{\text{min}}, hash_{\text{min}}, m, m', r, r') \\ \pi_{u_i} \rightarrow (1, 1, 0, 0)] \geq 1 - v(\lambda)$$

即当编辑前后哈希不一致, 且子密钥分享不正确时, 用户可验证到这一情况, 向监管者汇报对该密钥拥有者进行追责的概率接近于 1。

5 方案设计

首先给出可编辑且可追责的区块链方案(Editable and accountable blockchain scheme, EAB)的概述, 然后给出各个算法的具体实现过程。

5.1 方案概述

在 EAB 方案中, Leader 首先将陷门作为秘密, 作公开可验证的秘密分享给链中的各个用户, 并附带证明 π_l , 以保证 Leader 正确发送各个用户的陷门子密钥。链中用户对编辑请求投票后, 利用用户公钥和 Leader 所计算的随机数进行哈希并排序哈希值, 选出具有编辑权的编辑者, 该随机数的使用保证了编辑者的随机性, 避免恶意攻击者攻击固定编辑者情况的发生。其他用户发送相应的子密钥给编辑者, 并附带相应的证明 π_u , 以保证各用户正确分享子密钥。编辑者至少收到 t 个子密钥后, 利用拉格朗日插值函数恢复出陷门密钥并进行编辑, 保证编辑前后的哈希一致性。对编辑后的区块进行验证, 并对相关责任方进行追责。

5.2 方案具体构建

EAB 方案的具体算法构建如下:

(1) 秘密分享阶段:

1) Setup(1^λ) $\rightarrow$ (pubpara, secpara): 参数设置

算法, 以安全参数 1^λ 作为输入, 输出公共参数 pubpara 和私有参数 secpara。 $\mathbb{G}$ 是阶为 q 的任意乘法群, q 为素数, p 是大素数, 且 q 与 $p-1$ 互质。 Z_q 是阶为 q 的整数群。 g 和 h 是乘法群 $\mathbb{G}$ 的两个独立生成元 $H: \mathbb{G} \rightarrow Z_q$ 。 是一个抗碰撞的哈希函数。 选择一个随机数 $x_i \in Z_q$, 令用户私钥 $usk = x_i$, 公钥 $upk = y_i = h^{x_i}$ 。 Leader 选择一个随机数 $lsk \in Z_q$, 计算 $lpk = h^{lsk}$ 。 输出 $pubpara = (\mathbb{G}, Z_q, p, q, g, h, upk, lpk)$, $secpara = (usk, lsk)$ 。

2) Share(S) $\rightarrow (Y_1, Y_2, Y_3, \dots, Y_n, C_0, C_1, \dots, C_{t-1}, \pi_l)$: 密钥分发算法, 由 Leader 执行, 具体步骤如下:

步骤一: 随机选择 $s \in Z_q$, 令陷门 $S = h^s \in \mathbb{G}$ 。 选择 $(t-1)$ 次的多项式 $f(x) = \sum_{j=0}^{t-1} a_j x^j$, 其中 $a_0 = s, a_j \in Z_q$;

步骤二: Leader 计算系数的承诺值 $C_j = g^{a_j}, j = 0, 1, \dots, t-1$;

步骤三: 对子密钥进行加密, $Y_i = y_i^{f(i)}, i = 1, 2, \dots, n$;

步骤四: 构建证据。 对于每个 $i \in 1, 2, \dots, n$, Leader 随机选择数值 $w_i \in Z_q$, 并计算 $a_{1i} = g^{w_i}, a_{2i} = y_i^{w_i}, X_i = g^{f(i)}$, 利用 X_i, Y_i, a_{1i}, a_{2i} 计算公共值 $c_1 = H(X_1 \parallel X_2 \parallel \dots \parallel X_n \parallel Y_1 \parallel \dots \parallel Y_n \parallel a_{11} \parallel \dots \parallel a_{1n} \parallel a_{21} \parallel \dots \parallel a_{2n})$, 其中 $\parallel$ 表示两个比特串的连接。 计算对每个用户的应答 $r_i = w_i - f(i)c_1 \bmod q$, 证明 $\pi_l = (c_1, r_1, r_2, \dots, r_n)$ 。

输出加密后的子密钥 $Y_1, Y_2, Y_3, \dots, Y_n$, 系数承诺值 $C_0, C_1, \dots, C_{t-1}$, 证明 $\pi_l = (c_1, r_1, r_2, \dots, r_n)$, 并广播到链中。

3) Verifykey($Y_1, Y_2, Y_3, \dots, Y_n, C_0, C_1, \dots, C_{t-1}, \pi_l$) $\rightarrow b$: 验证子密钥算法, 由用户执行, 以 n 份加密后的陷门子密钥、 t 个承诺值和证明 π_l 作为输入。 具体步骤如下:

步骤一: 用户根据 Leader 公布的系数承诺值计算 $X_i' = \prod_{j=0}^{t-1} C_j^{i^j}$;

步骤二: 在根据公开信息 y_i, Y_i, r_i , 计算 $a_{1i}' = g^{r_i} X_i^{c_1}, a_{2i}' = y_i^{r_i} X_i^{c_1}, i = 1, 2, \dots, n$;

步骤三: 用户验证等式 $c_1 = H(X_1' \parallel X_2' \parallel \dots \parallel X_n' \parallel Y_1 \parallel \dots \parallel Y_n \parallel a_{11}' \parallel \dots \parallel a_{1n}' \parallel a_{21}' \parallel \dots \parallel a_{2n}')$ 是否成立, 若成立, 则输出 1, 证明子密钥分发成功; 若验证失败, 则输出 0, 表明分发的子密钥错误。

4) Dec(Y_i, upk_i, usk_i) $\rightarrow (S_i, \pi_{u_i})$: 解密子密钥算法, 由用户执行, 以用户收到的加密子密钥 Y_i , 用户的公钥 upk_i 和用户的私钥 usk_i 作为输入, 具体步骤如下:

步骤一: 利用用户私钥 usk_i , 即 x_i , 从加密子密钥 Y_i 中计算出子密钥 $S_i = Y_i^{x_i^{-1} \bmod q} = y_i^{f(i)x_i^{-1}} = h^{x_i f(i)x_i^{-1}} = h^{f(i)}$;

步骤二: 构建证据。用户 U_i 随机选择数值 $t_i \in Z_q$, 并计算 $b_{1i} = h^{t_i}$, $b_{2i} = S_i^{t_i}$, $c_2 = H(b_{1i} \parallel b_{2i})$, $z_i = t_i - x_i c_2$, 证明 $\pi_{u_i} = (c_2, z)$ 。

输出解密后的子密钥 S_i 和相对应的证据 $\pi_{u_i} = (c_2, z)$ 。

(2) 投票阶段:

1) $\text{Req}(m, m') \rightarrow (req)$: 请求编辑算法, 由用户执行, 以编辑的原内容 m 、编辑后的新内容 m' 作为输入, 输出编辑请求 $req = (m, m')$, 广播至链中。

2) $\text{Vote}(req, rd_i, b, lpk) \rightarrow (enc_i)$: 投票算法, 由用户执行, 以编辑请求 req 、用户选择的随机数 rd_i 、用户投票意见 b 、Leader 的公钥 lpk 作为输入, 其中, 若用户同意修改, 则 b 为 1。若不同意修改, 则进行以下操作: 首先, 验证用户的编辑请求与金融交易信息无关, 验证编辑请求符合编辑策略, 不会使链中下一个块无效。若请求内容无效, 则丢弃该请求。若请求内容有效, 则针对编辑请求进行投票。若同意修改, 用户 U_i 随机选择一个数 rd_i , 计算 $h_i = H(rd_i, b, req)$, 用 Leader 公钥 lpk 加密后的密文 $enc_i = E(rd_i, h_i, lpk)$, 输出加密后的投票信息 enc_i , 并将其发送至 Leader。

若不同意修改, 则略过投票步骤。

3) $\text{Vresult}(enc_i, lsk) \rightarrow (dec_i, result, rd_{\text{sum}})$: 结果算法, 由 Leader 执行, 以用户加密的投票信息 enc_i 、Leader 私钥 lsk 作为输入, 使用 Leader 私钥解密密文 enc_i , 得 $dec_i = D(enc_i, lsk) = (rd_i, h_i)$ 。统计收到的投票信息, 收到半数以上的投票则同意修改。并计算 $rd_{\text{sum}} = \sum rd_i$, 输出明文 dec_i 、投票结果 $result$ 和计算所得的随机数总和 rd_{sum} 。

(3) 寻找编辑者阶段:

$\text{Find}(rd_{\text{sum}}, upk_i) \rightarrow (hash_{\min}, upk_{\min})$: 寻找编辑者算法, 由用户执行, 以 Leader 所广播的随机数总和 rd_{sum} 、用户的公钥 upk_i 作为输入, 计算 $hash_i = H(upk_i, rd_{\text{sum}})$, 各个用户将自己计算的 $hash_i$ 和个人公钥广播到链中, 将哈希值按从小到大的顺序排序, 输出最小的哈希值 $hash_{\min}$ 和所对应的用户公钥 $upk_{\min}$, 最小哈希值所对应的公钥即为寻找到的编辑者。

(4) 编辑阶段:

1) $\text{Recover}(S_1, S_2, \dots, S_i (i \geq t)) \rightarrow S$: 恢复密钥算法, 由用户中的编辑者执行, 以至少 t 个用户 $U_i (i \in B \subseteq \{1, 2, \dots, n\} \text{ 且 } |B| \geq t)$ 的子密钥 $S_i (i \in B)$ 作

为输入, 利用拉格朗日插值函数计算得 $S = \prod_{i \in B} S_i^{C_{Bi}}$, 其中, $C_{Bi} = \prod_{j \in B \setminus \{i\}} j / (j - i) \bmod q$, 输出陷门密钥 S 。

2) $\text{Edit}(S, m, m', r) \rightarrow r'$: 编辑算法, 由用户中的编辑者执行, 以陷门密钥 S 、编辑的原内容 m 、编辑后的新内容 m' 、以及变色龙哈希所使用的随机数 r 作为输入, 计算 $r' = (m - m' + Sr) / S$, 使得 $\text{CH}(m, r) = \text{CH}(m', r')$, $\text{CH}(\cdot)$ 表示变色龙哈希算法, 保证编辑前后哈希一致性, 输出 r' 。

(5) 验证追责阶段:

$\text{Verify}(dec_i, rd_{\text{sum}}, upk_{\min}, hash_{\min}, m, r, r', \pi_{u_i}) \rightarrow (b_1, b_2, b_3, b_4)$, 验证算法, 由用户执行, 具体步骤如下:

1) 根据 Leader 公布的明文 dec_i 和收到的随机数总和 rd_{sum} , 计算随机数总和是否正确, 投票用户的哈希值是否正确, 若验证投票和随机数正确, 则 b_1 输出 1, 若不正确, 则输出 0, 并追责 Leader;

2) 判断具有编辑权的编辑者公钥是否对应最小哈希值, 计算 $hash_{\min} = H(upk_{\min}, rd_{\text{sum}})$ 等式是否成立, 若成立, 则编辑区块信息的用户具有编辑权, b_2 输出 1, 若不成立, 则编辑者并不具备编辑权, b_2 输出 0, 并追责编辑者。同时, 为了防范拥有陷门密钥的 Leader 进行非法编辑, 也会计算 Leader 的公钥和随机数总和的 $hash$ 值, 并验证是否等于最小 $hash$ 值, 当验证不通过时, 追责 Leader;

3) 判断编辑前后的哈希一致性, 计算等式 $\text{CH}(m, r) = \text{CH}(m', r')$ 是否成立, 若成立, 则 b_3 输出 1, 若不正确, 则输出 0;

4) 验证链中用户发送的子密钥值的正确性, 即验证 π_{u_i} 是否为真, 计算等式 $c_2 = H(h^z y_i^{c_2} \parallel S_i^z Y_i^{c_2})$ 是否成立, 若成立, 则 b_4 输出 1, 若不正确, 则输出 0。在 b_3 为 0, b_4 为 1 的情况下, 追责编辑者; 在 b_3 为 0, b_4 为 0 的情况下, 追责该密钥的拥有者。

6 安全性分析

定理 1(陷门安全性). 若 Diffie-Hellman 假设成立, 则任何 $(t-1)$ 个用户利用他们的子密钥不能通过计算恢复出陷门密钥, 任何 t 个用户利用他们的子密钥能恢复出陷门密钥。

证明. 首先证明任何 $(t-1)$ 个用户利用他们的子密钥不能通过计算恢复出陷门密钥。假设 $(t-1)$ 个用户利用他们的子密钥能恢复出陷门密钥 S , 不妨设这 $(t-1)$ 个用户就是前 $(t-1)$ 个用户 $U_i (i=1, 2, \dots, t-1)$ 。即证明, 对于任给的 g^a 和 g^b , 攻击者 A 利用这 $(t-1)$ 个用户作为预言机, 能算出

$g^{\alpha\beta}$ 。

现在为攻击者设置一个模拟系统, 使得攻击者将前 $(t-1)$ 个用户作为预言机, 就能算出 $g^{\alpha\beta}$ 。具体过程如下:

(1) 攻击者 A 令 $h = g^\alpha$, $C_0 = g^\beta (= g^{f(0)})$, 隐含确定了 $f(0) = \beta$;

(2) A 随机选取 $(t-1)$ 个值: $f(1), f(2), f(3), \dots, f(t-1) \in \mathbb{Z}_q$, 则确定了 $f(x)$ 函数;

(3) A 计算前 $(t-1)$ 个 X_i 和 Y_i 的值, $X_i = g^{f(i)}$ $i = 1, 2, \dots, t-1$; $Y_i = y_i^{f(i)}$ $i = 1, 2, \dots, t-1$ 。

(4) 由于 $f(0)$ 是隐含确定的, 所以攻击者 A 无法计算出 $f(t), f(t+1), \dots, f(n)$ 。但是可以利用拉格朗日插值函数计算出剩余的 X_i 值:

$$X_i = C_0^{C_B(i,0)} \cdot \prod_{j=1}^{t-1} X_j^{C_B(i,j)} = g^{\sum_{j=0}^{t-1} C_B(i,j) \cdot f(j)} = g^{f(i)}$$

其中, $i = t, t+1, \dots, n$; $B = \{0, 1, 2, \dots, t-1\}$; 插值系数 $C_B(i, j)$ 为:

$$C_B(i, j) = \prod_{k \in B \setminus \{j\}} (i - k) / (j - k)$$

其中, $i = t, t+1, \dots, n$; $j = 0, 1, \dots, t-1$ 。

(5) 计算 $C_j (j = 1, 2, \dots, t-1)$ 。因为 $f(x) = \sum_{j=0}^{t-1} a_j x^j$, 所以得出下式:

$$\begin{pmatrix} 1 & 0^1 & 0^2 & \dots & 0^{t-1} \\ 1 & 1^1 & 1^2 & \dots & 1^{t-1} \\ \dots & \dots & \dots & \dots & \dots \\ 1 & (t-1)^1 & (t-1)^2 & \dots & (t-1)^{t-1} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \dots \\ a_{t-1} \end{pmatrix} = \begin{pmatrix} f(0) \\ f(1) \\ \dots \\ f(t-1) \end{pmatrix}$$

上述线性方程组中, 攻击者 A 知道 $f(1), f(2), \dots, f(t-1)$, 但不知道 $f(0)$, 所以无法求出 $(a_0, a_1, \dots, a_{t-1})$ 。但是 A 可以求出 g^{a_j} , 即各个 $C_j (j = 1, 2, \dots, t-1)$ 。

若上述线性方程组的系数矩阵为 J , 因为 J 是范德蒙矩阵, 所以 J 存在逆矩阵。设 J^{-1} 为

$$J^{-1} = \begin{pmatrix} a_{00} & a_{01} & \dots & a_{0,t-1} \\ a_{10} & a_{12} & \dots & a_{1,t-1} \\ \dots & \dots & \dots & \dots \\ a_{t-1,0} & a_{t-1,1} & \dots & a_{t-1,t-1} \end{pmatrix}$$

则有:

$$J^{-1} \begin{pmatrix} a_0 \\ a_1 \\ \dots \\ a_{t-1} \end{pmatrix} = \begin{pmatrix} f(0) \\ f(1) \\ \dots \\ f(t-1) \end{pmatrix}$$

即 $a_j = \sum_{i=0}^{t-1} a_{ji} \cdot f(i)$, $j = 1, 2, \dots, t-1$ 。

因此,

$$\begin{aligned} g^{a_j} &= g^{\sum_{i=0}^{t-1} a_{ji} f(i)} = (g^{f(0)})^{a_{j0}} \cdot \prod_{i=1}^{t-1} (g^{f(i)})^{a_{ji}} \\ &= C_0^{a_{j0}} \cdot \prod_{i=1}^{t-1} X_i^{a_{ji}} \end{aligned}$$

由于攻击者 A 知道 C_0 和各个 $X_i (i=1, 2, \dots, t-1)$, 所以计算得:

$$C_j = C_0^{a_{j0}} \cdot \prod_{i=1}^{t-1} X_i^{a_{ji}}, \quad j = 1, 2, \dots, t-1.$$

(6) 攻击者 A 随机选取 $w_i \in \mathbb{Z}_q$, 令用户 U_i 的公钥为 $y_i = g^{w_i}$, 加密子密钥为 $Y_i = X_i^{w_i}$, $i = 1, 2, \dots, n$ 。则 Y_i 满足要求 $Y_i = y_i^{f(i)}$, 因为:

$$Y_i = X_i^{w_i} = (g^{f(i)})^{w_i} = (g^{w_i})^{f(i)} = y_i^{f(i)}$$

在这一模拟系统中, 前 $(t-1)$ 个用户可以计算出相关信息, 根据假设这 $(t-1)$ 个用户可以计算出密钥 $h^{f(0)}$, 攻击者 A 将这前 $(t-1)$ 个用户作为预言机, 所以 A 可以得知 $h^{f(0)}$ 。由于 $h = g^\alpha$ 且 $f(0) = \beta$, 因此 A 就可以计算出 $g^{\alpha\beta}$ 。这就与 Diffie-Hellman 假设矛盾。

所以任何 $(t-1)$ 个用户利用他们的子密钥不能通过计算恢复出陷门密钥。

其次, 证明任何 t 个用户利用他们的子密钥能恢复出陷门密钥。假设 $(t-1)$ 个用户的子密钥为 $S_i (i \in B, B = \{1, 2, \dots, n\}, \text{ 且 } |B| = t)$, 有:

$$\begin{aligned} \prod_{i \in B} S_i^{C_{Bi}} &= \prod_{i \in B} h^{(f(i))C_{Bi}} = h^{\sum_{i \in B} f(i)C_{Bi}} \bmod q = h^{f(0)} \\ &= h^s = S \end{aligned}$$

其中, $C_{Bi} = \prod_{j \in B \setminus \{i\}} j / (j - i) \bmod q$ 。所以任何 t 个用户利用他们的子密钥能恢复出陷门密钥 S 。

综上, 若 Diffie-Hellman 假设成立, 任何 $(t-1)$ 个用户利用他们的子密钥不能通过计算恢复出陷门密钥, 任何 t 个用户利用他们的子密钥能恢复出陷门密钥 S , 满足陷门安全性。

定理 2(可编辑性). 当具有编辑权的编辑者收到至少 t 个子密钥, 正确恢复出陷门密钥后, 编辑者进行编辑, 并保证哈希一致性, 即具有可编辑性。

证明. 当编辑者收到至少 t 个子密钥 $S_i (i \in B, B = \{1, 2, \dots, n\}, \text{ 且 } |B| = t)$ 利用拉格朗日插值函数恢复得出的陷门密钥 S 。

由变色龙哈希的构建 $\text{CH}(m, r) = g^m h^r \bmod p$, 其中, $h = g^S$ 为陷门公钥, S 为陷门私钥, m 为消息, r 为随机数。当 $\text{CH}(m, r) = \text{CH}(m', r')$ 时, 有 $g^m h^r \bmod p = g^{m'} h^{r'} \bmod p$, 进而得出 $m + Sr = m' + Sr'$, 因此

$$r' = m - m' + Sr/S \bmod p$$

因此, 编辑者使用新随机数 r' 时, 可保证编辑前后的哈希一致性, 从而不影响前后区块中的哈希值, 因此方案具有可编辑性。

定理 3(可追责性). 当编辑后的内容广播到区块链中, 用户可验证是否可被接受, 在出现错误时, 可向相关责任方进行追责, 即具有可追责性。

证明. 若方案具有追责性, 即在用户验证到区块不被接受时, 可以追责到相关责任方。即存在以下几种情况:

情况一: 链中用户可根据 Leader 广播的公开信息, 即投票信息的明文 dec_i 、投票结果 $result$ 和随机数总和 rd_{sum} , 判断用户的投票信息 h_i 是否存在于 Leader 所公开的信息中, 计算 $rd_{sum} = \sum rd_i$ 等式是否成立, 即是否存在与链中用户串通作弊, 通过更改随机数和从而内定编辑者的可能性。若等式不成立, 运行验证算法输出(0, 1, 1, 1), 表明 Leader 所计算的随机数和不正确, 则可追责到 Leader;

情况二: 链中用户计算 $hash_i = H(upk_i, rd_{sum})$, 并进行排序, 得到 $hash'_{min}$; 计算 $hash_{min} = H(upk_{min}, rd_{sum})$, 判断 $hash'_{min} = hash_{min}$ 等式是否成立, 若不成立, 运行验证算法输出(1, 0, 1, 1), 表明编辑者的哈希值不是最小哈希值, 实际并不具备编辑权, 则可以追责到编辑者;

情况三: 链中用户判断 $CH(m, r) = CH'(m', r')$ 是否成立, 若不成立, 计算

$$\begin{aligned} c_2 &= H(h^z y_i^{c_2} \parallel S_i^z Y_i^{c_2}) \\ &= H(h^{t-x_i c_2} h^{x_i c_2} \parallel S_i^{t-x_i c_2} S_i^{x_i c_2}) \\ &= H(h^t \parallel S_i^t) \end{aligned}$$

若上式成立, 运行验证算法输出(1, 1, 0, 1), 表明链中用户发送的子密钥值为真, 则可追责编辑者; 若不成立, 则为情况四;

情况四: 运行验证算法输出(1, 1, 0, 0), 表明子密钥值为假, 则可追责子密钥值的拥有者。

综上所述, 若编辑后的内容经公共验证后不被接受, 则可根据验证算法的输出结果追究到 Leader、编辑者、子密钥的拥有者等相关责任方, 因此该方案具有可追责性。

7 方案对比及性能分析

本方案与文献[7, 10, 15]进行对比, 如表 1 所示。可以看出, 本方案实现了可编辑性和可追责性, 且不需要秘密通道, 在陷门子密钥分发时保证了安全性。

表 1 各方案对比

Table 1 Comparison of the schemes

方案	可编辑性	不需要秘密通道	可追责性
文献[7]	√	×	×
文献[10]	√	—	—
文献[15]	√	—	×
EAB	√	√	√

为了更好的验证本方案, 通过仿真实验对方案中各阶段的算法进行性能评估, 并在不同密钥长度的情况下进行时间对比。算法的运行时间为十次实验中去掉最大值和最小值后的平均运行时间。本实验用 Java 语言编写, 实验所用环境具体配置如表 2 所示。

表 2 实验环境配置

Table 2 Experimental environment configuration

硬件	版本/型号
操作系统	Windows 10
内存	8 GB RAM
CPU	Intel Core i5 2.5 GHz

本实验实现了秘密分享、投票、寻找编辑者、编辑、验证追责五个阶段的具体算法。其中, 将系统用户设置为 10 人, 阈值为 5 人, 密钥长度为 1024 比特。

方案中各阶段算法的运行时间如图 2 所示, 其中所用的密钥长度为 1024 比特, Setup 算法运行时间为 20317ms, 即 20.317s, 虽然与其他步骤相比时间较长, 但是在该算法中生成了系统参数、用户的公私钥对以及 Leader 的公私钥对, 仅执行一次, 不用反复执行。密钥分发算法的运行时间 1517ms, 即 1.517s; 寻找编辑者算法运行时间仅 630ms, 即 0.63s; 恢复密钥算法运行时间仅 6.2ms; 编辑算法运行时间 265ms; 验证算法仅需 34ms。

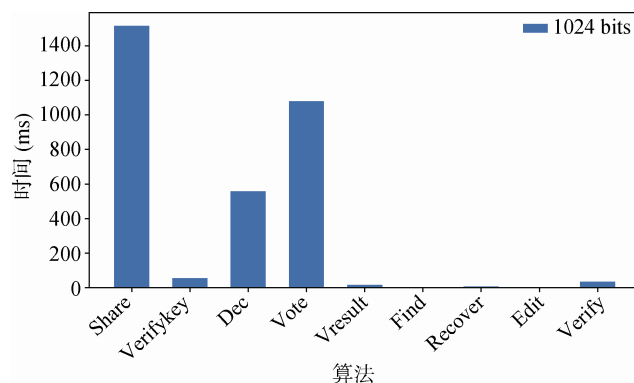


图 2 各算法的运行时间

Figure 2 Running time of each algorithm

在密钥长度分别为 512 比特、1024 比特以及 2048 比特时, 秘密分享阶段和投票阶段主要算法的运行时间如图 3 所示。图中显示, 在验证子密钥以及执行投票结果算法时, 不同比特的算法运行时间大致相同。在 2048 比特时, 解密子密钥约为 2s, 投票约为 3.7s, 投票结果算法的执行时间约为 94ms。

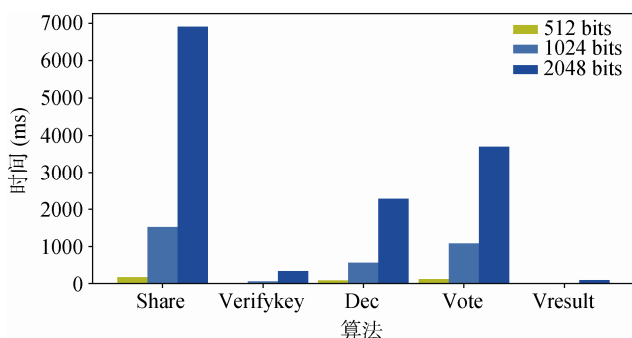


图 3 秘密分享阶段和投票阶段算法的运行时间

Figure 3 Running time of the algorithm for the secret sharing phase and the voting phase

在密钥长度分别为 512 比特、1024 比特以及 2048 比特时, 寻找编辑者阶段、编辑阶段以及验证追责阶段算法的运行时间如图 4 所示。图中显示, 在寻找编辑者、恢复密钥以及进行编辑操作时, 虽然密钥长度不同, 但是算法的执行时间相近, 在寻找编辑者的时间均不到 1ms。在 2048 比特时, 恢复密钥时间约为 8ms, 验证追责时间约为 145ms, 时间成本较低。从整体看, 这三个阶段算法的执行时间偏低, 均为毫秒级, 具有较高的效率。

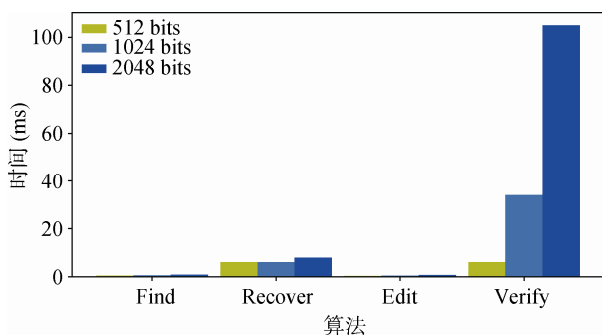


图 4 寻找、编辑以及验证追责阶段算法运行时间

Figure 4 Finding, editing and verifying the running time of the recourse phase algorithm

8 结论

本文提出了可编辑且可追责的区块链方案, 可以对区块链中的非交易数据进行编辑, 修改区块链中错误数据、更新陈旧数据, 该编辑不影响前后区块。

若该区块在编辑之后不被公共接受, 可对编辑过程中的相关责任方进行追责, 实现可追责性。该方案结合公开可验证秘密共享、零知识证明、变色龙哈希等工具, 针对编辑中心化、编辑者的选定以及编辑块的追责问题, 实现了编辑权的去中心化和编辑者的随机化以及已编辑块不能通过验证时的可追责性, 且在陷门值分发时不需要秘密通道, 满足陷门安全性。最后对方案进行了安全性证明与性能分析, 该方案满足陷门安全性、可编辑性和可追责性, 仿真实验表明方案中各算法效率较高。下一步工作将考虑编辑者的编辑权更新和方案效率提升之间的平衡和优化的问题。

参考文献

- [1] Yuan Y, Wang F Y. Editable Blockchain: Models, Techniques and Methods[J]. *Acta Automatica Sinica*, 2020, 46(5): 831-846.
(袁勇, 王飞跃. 可编辑区块链: 模型、技术与方法[J]. *自动化学报*, 2020, 46(5): 831-846.)
- [2] Matzutt R, Hiller J, Henze M, et al. A Quantitative Analysis of the Impact of Arbitrary Blockchain Content on Bitcoin[M]. *Financial Cryptography and Data Security*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2018: 420-438.
- [3] Shirriff K. Hidden Surprises in the Bitcoin Blockchain and how they are Stored: Nelson Mandela, Wikileaks, Photos, and Python Software[EB/OL]. 2014
- [4] Ibanez L D, O'Hara K, Simperl E. On Blockchains and the General Data Protection Regulation[OL]. 2018
- [5] Ateniese G, Magri B, Venturi D, et al. Redactable blockchain—or—rewriting history in bitcoin and friends[C]. *2017 IEEE European Symposium on Security and Privacy*, 2017: 111-126.
- [6] Camenisch J, Derler D, Krenn S, et al. Chameleon-Hashes with Ephemeral Trapdoors[M]. *Lecture Notes in Computer Science*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2017: 152-182.
- [7] Li P L, Xu H X, Ma T J, et al. Research on Fault-Correcting Blockchain Technology[J]. *Journal of Cryptologic Research*, 2018, 5(5): 501-509.
(李佩丽, 徐海霞, 马添军, 等. 可更改区块链技术研究[J]. *密码学报*, 2018, 5(5): 501-509.)
- [8] Ashritha K, Sindhu M, Lakshmy K V. Redactable blockchain using enhanced chameleon hash function[C]. *2019 5th International Conference on Advanced Computing & Communication Systems*, 2019: 323-328.
- [9] Marsalek A, Zefferer T. A correctable public blockchain[C]. *2019 18th IEEE International Conference on Trust, Security and Privacy In Computing and Communications/13th IEEE International Conference on Big Data Science and Engineering*, 2019: 554-561.
- [10] Ren Y L, Xu D T, Zhang X P, et al. Deletable Blockchain Based on Threshold Ring Signature[J]. *Journal on Communications*, 2019, 40(4): 71-82.
(任艳丽, 徐丹婷, 张新鹏, 等. 基于门限环签名的可删除区块链[J]. *通信学报*, 2019, 40(4): 71-82.)

- [11] Derler D, Samelin K, Slamanig D, et al. Fine-grained and Controlled Rewriting in Blockchains: Chameleon Hashing Gone Attribute-based[R/OL]. 2019
- [12] Puddu I, Dmitrienko A, Capkun S. μ Chain: How to Forget without Hard Forks[R/OL]. 2019
- [13] Deuber D, Magri B, Thyagarajan S A K. Redactable blockchain in the permissionless setting[C]. *2019 IEEE Symposium on Security and Privacy*, 2019: 124-138.
- [14] Thyagarajan S A K, Bhat A, Magri B, et al. Reparo: Publicly Verifiable Layer to Repair Blockchains[M]. *Financial Cryptography and Data Security*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2021: 37-56.
- [15] Ren Y L, Xu D T, Zhang X P, et al. Scheme of Revisable Blockchain[J]. *Journal of Software*, 2020, 31(12): 3909-3922.
(任艳丽, 徐丹婷, 张新鹏, 等. 可修改的区块链方案[J]. *软件学报*, 2020, 31(12): 3909-3922.)
- [16] Jia Y X, Sun S F, Zhang Y, et al. Redactable Blockchain Supporting Supervision and Self-Management[C]. *The 2021 ACM Asia Conference on Computer and Communications Security*, 2021: 844-858.



赵晓琦 于 2021 年获得北京交通大学信息与通信工程硕士学位。研究领域为区块链安全。Email: 18120187@bjtu.edu.cn



张正昊 于 2022 年获得北京交通大学信息与通信工程硕士学位。研究领域为区块链安全。Email: 19120172@bjtu.edu.cn



李勇 于 2007 年在中国科学院研究生院获得博士学位。现任北京交通大学电子信息工程学院副教授。研究领域为数据安全和隐私保护、区块链、应用密码学。Email: liyong@bjtu.edu.cn