面向可信联盟的区块链账本可验证修改方法研究

吕伟龙[1)] 魏松杰[1)] 于铭慧[2)] 李莎莎[1)]

[1)](南京理工大学计算机科学与工程学院 南京 210094)

[2)](南京理工大学理学院 南京 210094)

摘 要 区块链历经十年发展,成为信息技术领域最被寄予厚望的颠覆性技术之一.区块链上链数据具有不可篡改性,历史区块数据一旦确认就不能变更.这保证了区块链历史数据的完整性和可验证,但另一方面,当区块链中出现数据管理需求,即历史区块中过期或无效交易需要被压缩、违法违规信息需要被管控删除时,这种特性也将阻碍区块链对有问题的数据进行修改.类似数据管理需求在联盟链、私有链中尤其突出.传统区块链利用哈希算法的碰撞困难,实现区块和交易的完整性验证.而变色龙哈希存在一个陷门,掌握陷门可以轻松找到哈希碰撞.基于这个特性,本文将该类哈希算法的陷门交给多方管理,从而在不影响前后区块的完整性验证的情况下,实现多方共识修改交易数据的功能.本文进一步对变色龙哈希进行改进,设计了一种适用于联盟链的多中心化的账本修改方案.考虑到交易修改功能的去中心化,即变色龙哈希的陷门不应生成、存放于单个节点的问题,改进后的算法允许联盟链的所有权限节点协作生成系统的变色龙哈希公私钥.同时,为了权衡时间、空间代价与安全性,设计了多种有关私钥生成与同步的共识机制,并对它们的空间开销、通信时间、安全程度等性能进行了对比.本文考虑了一些特殊情况下区块链系统的可用性和问责性.最后,利用改进后的变色龙哈希算法构建了原型链,实现了历史交易的管控功能,描述了数据管理功能的设计细节.实验表明,本文提出的账本修改方案,其最佳区块压缩率可达30%、算法执行速率整体达到毫秒级,且具有可证明的安全性.

关键词 联盟链;变色龙哈希;可编辑区块链;账本验证;多中心化

中图法分类号 TP309 **DOI号** 10.11897/SP.J.1016.2021.02016

Research on Verifiable Blockchain Ledger Redaction Method for Trusted Consortium

LV Wei-Long[1)] WEI Song-Jie[1)] YU Ming-Hui[2)] LI Sha-Sha[1)]

[1)](*School of Computer Science and Engineering*, *Nanjing University of Science and Technology*, *Nanjing* 210094)

[2)](*School of Science*, *Nanjing University of Science and Technology*, *Nanjing* 210094)

Abstract After a decade of development, blockchain becomes one of the most disruptive innovations in information technology. Data on the blockchain has the strength of immutability. Once a historical block of data is confirmed, it cannot be revised any more. Such design guarantees the integrity and verifiability of blockchain historical data. But when there is a need of data maintenance in blockchain ledger, such as compression of expired or invalid transactions in historical blocks, removal of illegal information, immutability prevents a blockchain from redacting the data questioned. Such data maintenance requirements are particularly prominent for consortium and private blockchains. Therefore, redactable blockchain is of great value in these cases where a blockchain ledger is allowed compressing ledger data whose value fades away to save storage space, and when new regulations roll out a blockchain can adjust historical data

收稿日期:2020-03-15;在线发布日期:2021-01-09.本课题得到国家自然科学基金(61802186, 61472189)资助.**吕伟龙**,硕士研究生,计算机学会会员(A2828G),研究领域为区块链技术、密码学,E-mail:lvweilong26@126.com. **魏松杰**(通信作者),博士,副教授,研究领域为网络安全、区块链技术,E-mail:swei@njust.edu.cn. **于铭慧**,硕士研究生,研究领域为信息系统安全与应用. **李莎莎**,硕士研究生,研究领域为分布式系统、任务调度.

accordingly without forking, in order to satisfying government supervision. Traditional blockchain can guarantee the integrity verification of blocks and transactions relying on the collision difficulty of hash algorithms. While chameleon hash algorithm contains a trap door or a private key, which one may utilize to find collisions easily. When applying in blockchain, the trap door of the hash algorithm can be handed over to multiple parties for management, to support the procedure of redacting transaction data by consensus without affecting the integrity verification of the neighboring blocks. The existing ledger redaction methods simply add modification and deletion functions to the blockchain system, without good adaption in the consensus procedure. Implementation of some existing methods replies on specific block structures or blockchain architectures. These methods lack universality and compatibility across various blockchain platforms, and they are restrained from being promoted in all the mainstream blockchains. Other existing methods supporting redaction operation require the intervention of trusted third parties or a centralized single node, which undermines the decentralization of the blockchain and is likely to cause single point of failure or trust issues. In order to solve these problems, in this paper, we optimize the original chameleon hash as a multiply centralized ledger redaction method for consortium blockchain. Considering the decentralization of the transaction redaction capability, that is, the trap door of the chameleon hash should not be generated and stored at a single node, the traditional chameleon hash is improved allowing all authority nodes of the consortium blockchain cooperatively generate the chameleon hash's public and private keys. In order to evaluate time, space cost and security robustness, several private key generating and synchronizing consensus mechanisms are designed and benchmarked in all aspects. The usability and accountability of the blockchain system are considered in special cases. Finally, the improved chameleon hash algorithm is used to construct a prototype chain. We implement the management function for historical transactions and describe the design details of the function. Experiments show that, applying the ledger redaction method proposed, the best block compressing rate can reach 30%, while the algorithm executing rate achieves millisecond level, and the improved algorithm has proved safety.

Key words consortium blockchain; chameleon hash; redactable blockchain; ledger verification; multiple centralization

1 引 言

区块链是最近十年来最为令人瞩目的信息技术发展和应用模式创新之一.在区块链账本中,数据可分为两种:与历史无关的独立数据和与历史有关的非独立数据.在以比特币为代表的区块链1.0时代中,区块链系统主要是为金融交易而设计的,对于区块中的转账记录,无需设计修改操作来逆转交易,一方面是由于转账记录并非独立数据,其数值与前后区块中的交易存在一定的联系,无法简单的进行修改、删除;另一方面,正是由于区块链数据不可篡改的特性,才使用户相信并接受基于区块链的数字加密货币.但随着区块链应用场景和用户需求的变化,在以以太坊等为代表的区块链2.0时代中,用户不仅仅在区块链上进行转账交易,还会将现实世界中的有形和无形资产对象转换成数字通证并上链,实现资产标记化的目的.在这种应用背景下,可能会出现需要对区块数据进行编辑管理的情况.考虑一种场景,在基于区块链的知识产权保护平台中[1-2],用户的知识产权转换成数字通证在区块链中发布,之后若知识产权法发生变化,平台有责任修改链上数据,使得数据符合新法律的要求.

从政府监管角度来看,区块链也应该拥有特定条件下数据编辑的能力.欧盟《一般数据保护条例》[3]首次在立法层面确立了个人隐私数据的被遗忘权,这意味着,个人有权删除其在平台中的相关数据.中国《区块链信息服务管理规定》[4]中要求区块

链信息服务提供商,具备对法律禁止的信息内容进行修改、删除等操作的应急处理能力.预计未来在德国、美国等国家的法院裁决中,将认定那些存储有令人反感内容的区块链的服务提供商有罪[5].因此,在一定条件下,允许区块链对历史数据进行必要的可控、可溯、可验证的修改,符合监管要求,符合管理需求,有利于区块链系统的广泛普及.

不可篡改性是区块链众多特性之一,是指历史区块中的数据一旦经过确认就很难被更改.不可篡改性保证了区块链历史数据的完整性和可靠性,但同时这种特性也阻碍了区块链对有问题的历史数据进行修订.另外,由于目前主流的区块链大多属于只增不减的去中心化账本,链中数据量日益增多,这种特性也给系统资源和可用性带来了负面影响,不利于可持续发展.因此,需要一种链上数据可编辑的改进来适配现有区块链架构,其中最核心的是适用于区块链的账本修改方案[6].

需要指出的是,区块链魅力之一在于对等共识,即参与者之间的平等的一致性同意.区块链数据安全来源于参与者之间的共识和协作,参与者既可以形成对新的区块中数据正确性的共识,也可以实现对于历史区块中数据的修改方案的共识.因此区块链上数据安全并不囿于能否修改的争辩,而是能否支持可以被拥有足够权益的参与者所认可的、能够事前提案共识、事后广泛接受、随时可验可溯的数据修改机制.

本文提出的账本修改方案,不是对某一现有区块链系统的定制设计开发,其不依赖特定的区块链架构和区块结构,即该方案中的账本修改功能不需要以特定的区块链架构或区块数据结构为前提,具有跨区块链平台的通用性、兼容性,且能够在不影响区块链原本完整性验证的情况下修改账本,以保证对当前主流区块链系统的数据和算法的兼容性.因此本文引入了变色龙哈希算法,对其密钥管理进行了分布式设计,实现了保证原有哈希值的链上数据修改与验证.

同时,由于区块链倾向于弱化单中心性,本文还希望这种账本修改方案的整个过程都是多中心化的,即在账本修改的整个过程中,不需要中心节点或可信第三方参与.因此,本文改进了原本的变色龙哈希算法,改进后的算法允许多个节点共同生成系统的变色龙哈希公私钥.

本文的研究内容和创新成果主要有以下5点:

(1)对传统变色龙哈希算法进行了改进,允许多个节点合作生成变色龙哈希的公私钥对,保证了密钥生成、同步过程的去中心化;

(2)设计了四种变色龙哈希公私钥对的生成细粒度,分析了各个粒度的时空开销、安全性能以及适用场景;

(3)面向联盟链,利用改进后的变色龙哈希算法,设计了一种多中心化的账本修改方案,并提出了一种可编辑的联盟链架构,主要用于修改区块中独立数据;

(4)为了减少恶意权限节点作恶的可能性,设计了一种权限节点入职、离职以及保证金动态调整的共识过程,用损益机制增大恶意权限节点的作恶成本;

(5)对提出的账本修改方案进行了性能测试,主要分析并测试了方案的压缩率、算法执行速率、额外存储空间、安全性和联盟链关键阶段运行时间.

2 相关工作

当前对可编辑区块链的研究方兴未艾.区块链账本修改方案大致分为两类,基于门限环签名的账本修改方案和基于变色龙哈希的账本修改方案.

2.1 基于门限环签名的账本修改方案

任艳丽等人提出了基于门限环签名的账本修改方案[7],这种方案依赖于空间证明共识机制的区块结构.当数据过期或失效时,由一个节点提出删除提案,其他节点为这个提案投票,当同意这个提案的节点超过设定的阈值后,系统生成一条专属的删除消息;接着,同意提案的节点成为签名节点,代表整个系统生成门限环签名;最后,由提案者将生成的环签名储存于交易数据原本所在的位置,再全网广播,即可完成删除操作.这种方案实用有效,但有几点局限性:首先,其依赖于特定的区块结构,兼容性不高;另外,该方案只能删除某区块全部的交易内容,无法删除指定交易的内容,也无法编辑交易内容.相关研究人员又对文献[7]中方案进行了修改,引入机动因子的概念,修改了空间证明共识机制的区块内容,并利用陷门单向函数,允许特定阈值数的节点同意后,合法的修改区块数据[8].这种改进解决了原方案中无法删除指定交易,无法编辑交易内容的困境,但仍存在两个问题:首先,该方案仍依赖于特定的区块结构,兼容性不足;其次,为了保存修改记录以供溯源,在该方案中每笔交易的每次修改都会记录在主链中,给普通节点带来了明显的空间消耗负担.

2.2　基于变色龙哈希的账本修改方案

Krawczyk等人提出了变色龙哈希与签名算法[9]，与传统哈希算法不同，变色龙哈希算法可以人为的设下一个“陷门”或者“私钥”，掌握了这个“私钥”就可以轻松的找到碰撞. 这种算法允许区块链修改交易内容却不改变其哈希值. 但由于区块链是一个去中心化的系统，而Krawczyk等人的原算法并没有考虑密钥的去中心化生成和账本的去中心化修改，因此需要对原算法进行改进，使其可以更好的适配区块链系统.

Ateniese等人在变色龙哈希算法上优化[10]，由一个可信节点生成系统的变色龙哈希密钥，再利用秘密共享的方式将系统私钥拆分成多个私钥份额，并分配给网络中的全部节点或份额最高的几个节点；在需要进行账本修改操作时，利用多方计算将各个节点的子私钥还原成系统私钥，从而修改交易内容. 这种方案存在一个问题：虽然变色龙哈希的子私钥存放在多个节点上，账本修改过程达到一定程度的去中心化，但密钥的生成与还原都需要一个中心化的可信节点，因此该方案的去中心化程度不高.

李佩丽等人针对联盟链，在变色龙哈希算法上进行了改进[11]，由一个可信节点生成系统的变色龙哈希子密钥，当有修改需求时，再利用随机数生成协议与秘密分享技术随机挑选修改者对区块进行修改. 与Ateniese等人的方案不同，该方案从不同方面改进了变色龙哈希算法，在需要进行账本修改操作时，无需进行多方计算，即可修改交易内容. 但该方案的密钥生成还是由一个可信节点完成，账本修改过程无法摆脱中心化处理步骤.

Ashritha等人在Ateniese等人方案的基础上进一步提高[12]：利用安全性更佳的非线性秘密共享替换了原本的线性秘密共享；在修改交易过程中加入了交易内容验证机制与记账者许可机制. 但该方案并未脱离对于可信单节点的依赖.

本文所提出的方案，着重考虑交易修改过程的去中心化和对主流区块链的兼容性，在保证区块编辑功能的前提下，普适多种区块链架构，消除中心化计算步骤，与现有方案对比如表1所示.

3　变色龙哈希算法介绍与改进

在回顾变色龙哈希的计算原理后，本章将对变色龙哈希算法进行改进，以便更好的服务于本文提出的多中心化的账本修改方案. 下面依次介绍变色龙哈希的基础知识、原算法实例和本文的算法改进点.

3.1　数学原理

在变色龙哈希函数中，有2个参数：明文t，可变参数r. 在明文改变后，通过改变可变参数，可以使得$H(t,r)=H(t_2,r_2)$. 其主要利用了指数运算的以下两个性质：$g^a\times g^b=g^{a+b}$、$(g^a)^b=g^{ab}$.

(1)计算变色龙哈希值

变色龙哈希值$H(t,r)=g^t\times h^r$，其中$h=g^s$，h为公钥，s为私钥，g为公共参数，t为明文.

(2)锻造新可变参数r_2

在知道私钥s的情况下，就可以将$h=g^s$带入，得到$H(t,r)=g^t\times h^r=g^t\times g^{sr}=g^{t+sr}$与$H(t_2,r_2)=g^{t_2}\times h^{r_2}=g^{t_2}\times g^{sr_2}=g^{t_2+sr_2}$. 由于两式底数相同，又$H(t,r)=H(t_2,r_2)$，再令指数相等，即可求出新可变参数$r_2=(t+sr-t_2)/s$.

表1　区块链账本修改方案对比表

方案	可删除指定交易	可编辑交易内容	不依赖特定的区块结构	去中心化程度	应用点
任艳丽等[7]	否	否	否	多中心化	公链
任艳丽等[8]	是	是	否	多中心化	公链
Ateniese等[10]	是	是	是	半去中心化	公链
李佩丽等[11]	是	是	是	半去中心化	联盟链
Ashritha等[12]	是	是	是	半去中心化	公链
本文方案	是	是	是	多中心化	联盟链

3.2　算法简介

Krawczyk等人提出的变色龙哈希算法有5个子函数，具体描述如下：

(1)初始化Setup(λ)：输入安全性参数λ，输出公共参数$pp=\{p,q,g\}$. 其中p,q满足$p=kq+1$，g为乘法循环群Z_p^*的生成元；

(2)密钥生成KeyGen(pp)：输入公共参数pp，输出公钥h，私钥s. 其中s为在乘法循环群Z_q^*中随机挑选的元素，$h=g^s \bmod q$；

(3)哈希计算Hash(h,t,r)：输入公钥h、明文t、可变参数r，输出变色龙哈希值$CH=g^th^r \bmod p$. 其中t,r均为Z_q^*的元素；

(4)可变参数锻造Forge(s,t,r,t_2)：输入私钥s、原明文t、原可变参数r、新明文t_2，输出与t_2匹配的新可变参数r_2. 其中t,r,t_2均为Z_q^*的元素，$r_2=(t+sr-t_2)s^{-1} \bmod q$；

(5)哈希验证 Verify(h,t,r,CH):输入公钥h、明文t、可变参数r、变色龙哈希值CH,验证CH与t,r是否匹配,如果匹配输出1,不匹配输出0.

3.3 算法改进

根据上节的介绍,不难发现可以直接将变色龙哈希算法替换区块链原本的哈希函数,但在这种直接替换的方案中,密钥的生成与使用都依赖于一个可信单节点.去中心化是区块链系统优势之一,本文希望交易修改操作的整体流程尽可能的满足去中心化的特性,因此需要对原本的变色龙哈希算法进行改进.

假设联盟链中一共有m个节点,其中n个是权限节点.使系统的私钥$s=s_1+s_2+\cdots+s_n$,其中$s_1,s_2,\cdots,s_n$为每个权限节点的子私钥,而系统的公钥$h=g^{s_1+s_2+\cdots+s_n}=g^{s_1}g^{s_2}\cdots g^{s_n}=h_1h_2\cdots h_n$,即系统的公钥为所有权限节点的子公钥之积.值得注意的是,$m\in[1,n]$,当$m=1$时,密钥生成完全中心化;当$m=n$时,密钥生成完全去中心化;当m越接近n时,密钥生成的去中心化程度越高但通信代价也越高.

(1)初始化Setup(λ):输入安全性参数λ,输出公共参数$pp=\{p,q,g\}$.其中p,q满足$p=kq+1$,g为乘法循环群Z_p^*的生成元;

(2)密钥生成KeyGen(pp):各个权限节点在本地调用此函数.对于第i个权限节点来说,输入公共参数pp,输出权限节点i的子公钥h_i,子私钥s_i.其中s_i为在乘法循环群Z_q^*中随机挑选的元素,$h_i=g^{s_i}\bmod q$;

(3)哈希计算Hash(h,t,r):假设全网有n个权限节点,输入系统公钥h,即所有权限节点的子公钥之积$h_1h_2\cdots h_n$、明文t、可变参数r,输出变色龙哈希值$CH=g^t(h_1h_2\cdots h_n)^r\bmod p$.其中$t,r$均为$Z_q^*$的元素;

(4)可变参数锻造Forge(s,t,r,t_2):假设参与某交易打包的权限节点有k个,输入系统私钥s,即这k个权限节点的子私钥之和$s_1+s_2+\cdots+s_k$、原明文t、原可变参数r、新明文t_2,输出与t_2匹配的新可变参数r_2.其中t,r,t_2均为Z_q^*的元素,$r_2=\left(t+sr-t_2\right)s^{-1}\bmod q$,其中$s=s_1+s_2+\cdots+s_k$;

(5)哈希验证 Verify(h,t,r,CH):假设参与某交易打包的权限节点有k个,输入系统公钥h,即这k个权限节点的子公钥之积$h_1h_2\cdots h_k$、明文t、可变参数r、变色龙哈希值CH,验证CH与t,r是否匹配,如果匹配输出1,不匹配输出0.

3.4 改进后算法的安全性分析

原算法的安全性分析已经在文献[9]中讨论,本节中着重讨论算法改进部分的安全性.

假设全网有n个权限节点,其中第1到第i个$\left(i\in[1,n)\right)$权限节点是恶意节点,它们知晓公共参数$\{p,q,g\}$、系统公钥h、它们自己的子私钥$\{s_1,s_2,\cdots,s_i\}$.在改进后的变色龙哈希中

$$H(t,r)=g^th^r=g^t(h_1h_2\cdots h_ih_{i+1}h_{i+2}\cdots h_n)^r$$
$$=g^t(g^{s_1}g^{s_2}\cdots g^{s_i}h_{i+1}h_{i+2}\cdots h_n)^r.$$

若这些恶意节点想要在不知道其他权限节点子私钥的情况下,锻造可变参数r_2使得$H(t,r)=H\left(t_2,r_2\right)$,对其来说

$$H(t,r)=g^t(g^{s_1}g^{s_2}\cdots g^{s_i}h_{i+1}h_{i+2}\cdots h_n)^r$$
$$=g^{t+(s_1+s_2+\ldots+s_i)r}(h_{i+1}h_{i+2}\cdots h_n)^r$$

令$t'=t+(s_1+s_2+\cdots+s_i)r$,$h'=h_{i+1}h_{i+2}\cdots h_n$,则$H(t,r)=g^{t'}h'^r$,同理$H\left(t_2,r_2\right)=g^{t_2'}h'^{r_2}$.

根据上步的变量代换,不难发现其形式与原变色龙哈希算法一致,因此恶意节点们要求出r_2只能令$g^{t'}h'^r=g^{t_2'}h'^{r_2}$,由于恶意节点们只知道$g$、$t'$、$r$、$t_2'$、$h'$,恶意节点们若想求出$r_2$,该难度等同于破解原变色龙哈希算法,因此改进后的变色龙哈希算法是安全的.

4 联盟链账本修改方案设计

4.1 技术思路

区块链的完整性验证主要是依靠哈希函数实现的.记账者通过对父区块的区块头进行哈希计算,保证新的区块可以有序的连接在父区块之后;记账者通过对交易进行哈希计算,保证自己打包的交易无法被篡改.如图1所示,以比特币的区块结构为例,区块头中,对②部分哈希得到的值①,可以保证区块数据的完整性;区块体中,对④部分哈希得到的值③,可以保证交易数据的完整性[13-18].

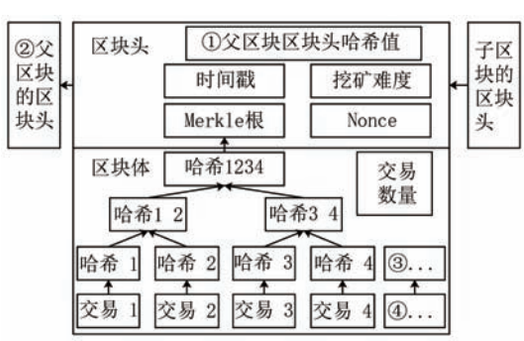

图1 比特币区块结构

为了保证方案对当前主流区块链的高度兼容性，本文希望提出的账本修改方案不依赖特定的区块链架构和区块结构，且能够在不影响区块链原本完整性验证的情况下修改账本. 本文考虑了两种方案.

(1)将对父区块数据哈希的哈希函数替换成变色龙哈希函数

这种方案下，第i个区块的区块头包含的内容$content_i$可简单分为三类：父区块的区块头内容$content_{i-1}$的变色龙哈希值CH_i、与交易内容有关的Merkle根值MR_i和其他数据w_i，即$content_i=\{CH_i, MR_i, w_i\}$. 另外，在区块头中还要记录与变色龙哈希$CH_i$匹配的可变参数$r_i$，其与$CH_i$的关系为$CH_i=\mathrm{Hash}\left(content_{i-1}, r_i\right)$.

在大多数情况下，每个区块都是无法编辑的状态. 若要修改的交易在第i个区块中，需修改第i个区块中的交易内容，此时Merkle根值MR_i变化为MR_i'，$content_i$变化为$content_i'$，若想让第$i+1$个区块仍能通过完整性验证，需要使用预先生成的变色龙哈希私钥重新锻造第$i+1$个区块中的变色龙哈希可变参数r_{i+1}'，使得r_{i+1}'与CH_{i+1}、$content_i'$匹配，也就是说符合$CH_{i+1}=\mathrm{Hash}\left(content_i', r_{i+1}'\right)=\mathrm{Hash}\left(content_i, r_{i+1}\right)$，该方案的流程如图2所示.

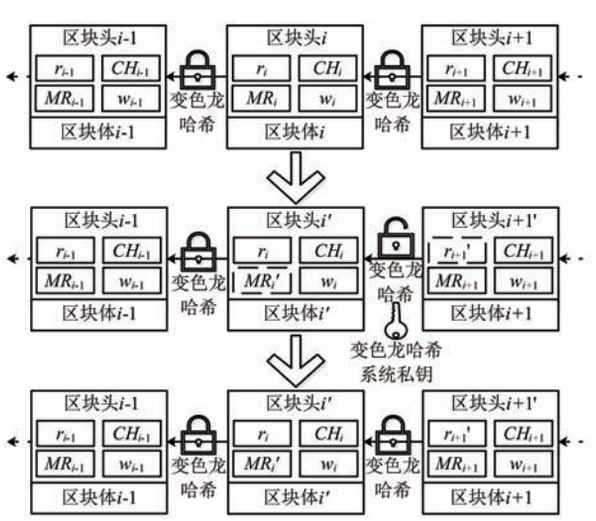

图2　替换父区块哈希方案流程

(2)将对交易数据哈希的哈希函数替换成变色龙哈希函数

这种方案下，若要修改的交易在第i个区块中，只需修改第i个区块中的交易内容，再重新锻造第i个区块中与该交易哈希相关的可变参数，从而在交易内容改变后，使该交易的哈希值不变，该方案的流程如图3所示.

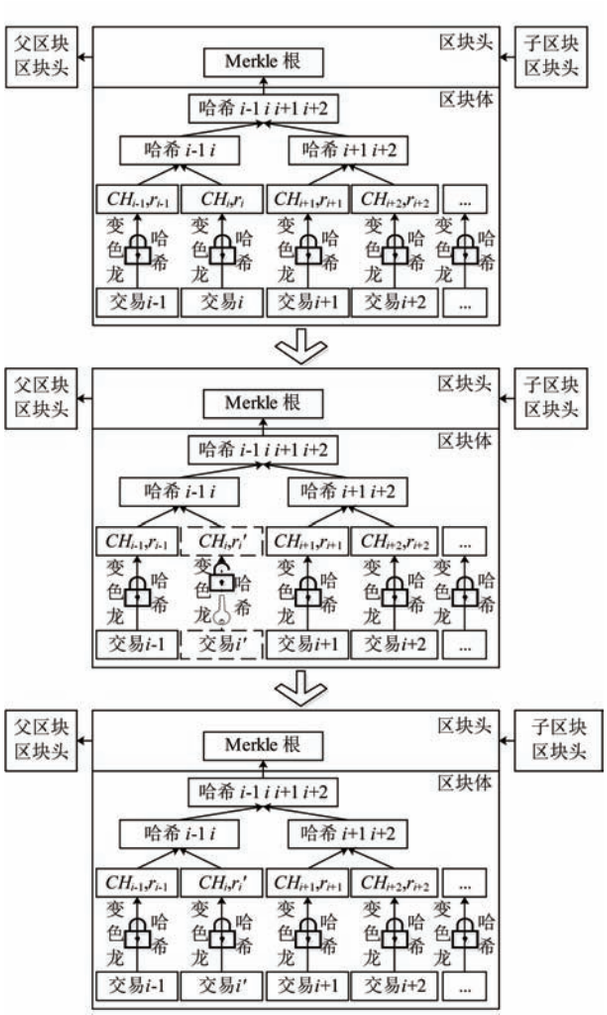

图3　替换交易哈希方案流程

具体来说，在大多数情况下，每个交易都是无法编辑的状态. 假设第j个区块的第i笔交易的哈希值为CH_i，当该交易有修改需求时，将新交易内容t_i'替换区块体中的原交易内容t_i，此时交易i变为交易i'；再利用预先生成的变色龙哈希系统私钥，锻造出与交易内容t_i'相符的可变参数r_i'，使得r_i'与t_i'、CH_i匹配.

在账本修改的过程中，第一种方案需要修改多个区块的数据，而第二种方案只需修改一个区块的数据，所以本文采用的方案是第二种方案.

4.2　变色龙哈希的密钥生成粒度

由于变色龙哈希算法引入了公私钥的概念，因此对于权限节点来说，它们需要额外的存储空间来储存与公私钥相关的数据；同时，对于区块链系统来说，公私钥的同步与广播也会带来额外的通信时间.

因此本文设计了4种密钥生成粒度来权衡算法的安全性与时间、空间代价.

(1)密钥生成粒度:交易

对于联盟链里的每笔交易,各生成一对变色龙哈希公私钥.这种粒度的优势在于,假设一个恶意节点获得了某交易相关的变色龙哈希私钥,它只能修改该交易的交易内容,而无法对其他交易进行内容修改.这种密钥生成粒度的安全性是最高的,但带来的时间、空间代价也是最高的.以以太坊为例,目前以太坊中记录大约有6亿笔交易,若使用该粒度,对于每个权限节点来说要花费大约60GB的额外存储空间代价.以交易为密钥生成粒度适用于那些修改操作极少,对交易确认时间、权限节点存储空间没有严格要求,但对修改操作的正确性与系统安全性非常看重的"不容有失型"联盟链.

(2)密钥生成粒度:区块

对于联盟链里的每个区块,各生成一对变色龙哈希公私钥,区块中的每笔交易都使用该区块的变色龙哈希公钥进行哈希值的计算.这种密钥生成粒度带来的时间、空间代价相对第一种粒度要小,以以太坊为例,现在的以太坊中大约有900万个区块,若使用该粒度,对于每个权限节点来说要花费大约500MB的额外存储空间代价.但这种密钥生成粒度也带来了一个问题:如果一个恶意节点获得了某区块的私钥,那么它将可以修改该区块中的所有交易.由于本文提出的账本修改方案是针对联盟链的,本文认为联盟链相比公有链,是一个半封闭半可信的系统,节点间互相知道,相互间拥有领域内的信任绑定和制约,同时每笔交易的修改记录都会被保存,只有拥有合法修改记录且通过完整性验证的交易才会被权限节点接受,没有未知存在的恶意节点,因此这种密钥生成粒度是可以实用的.以区块为密钥生成粒度适用于那些修改操作较少,对交易确认时间、权限节点存储空间要求不高,但对修改操作的正确性与系统安全性比较看重的"决策慎重型"联盟链.

(3)密钥生成粒度:联盟链

对于整个联盟链,只生成一对变色龙哈希公私钥,联盟链里的所有交易都使用这对公私钥计算哈希值.显然,这种密钥生成粒度带来的时间、空间代价最低,但安全性也最低.以联盟链为密钥生成粒度适用于权限节点之间完全信任,区块内存储的交易数据价值较低,但有频繁修改需求,要求系统在较短时间内做出尽可能正确的修改操作的"时间敏感型"联盟链.

(4)密钥生成粒度:时间段

考虑一种更灵活的密钥生成粒度,本文将密钥生成粒度置于时间维度.对于每个单位时间,生成一对变色龙哈希公私钥,时间段内所有区块中的交易都使用这对公私钥计算哈希值.时间单位长度可以根据物理时间划分,即每隔几个小时、几天、几个月等更换一对公私钥,也可以根据逻辑时间划分,即每生成x个区块更换一对公私钥.当$x=1$时,该方案相当于以区块为密钥生成粒度,当$x\to\infty$时,该方案相当于以联盟链为密钥生成粒度.因此这种密钥生成粒度带来的时间、空间消耗介于第二种和第三种粒度之间,安全性也介于第二种和第三种粒度之间.这种粒度的优势是,联盟链的设计者可以自己权衡安全性与时间、空间代价,灵活的设定合适其联盟链的密钥生成时间.其适用于修改操作频繁程度适中,对交易确认时间、权限节点存储空间要求适中,对修改操作的正确性与系统安全性看重程度也较为适中的"平衡性"联盟链.本文采用的正是这种密钥生成粒度.以交易、区块、联盟链以及时间段为密钥生成粒度的四种方案如图4所示.

对于第四种密钥生成粒度,由于交易打包操作需要变色龙哈希公钥,因此,必须要保证在新时间段开始前生成好新的公私钥,并将公钥广播.本文设计了一种密钥生成与广播的共识机制,在该共识机制中,每个时间段分成密钥生成时间与密钥同步时间,用于生成并同步下个时间段的密钥.在第n个时间段的密钥生成时间内,每个想要参与下个时间段交易打包的权限节点需要生成好自己下个时间段的子公私钥,并在密钥同步时间开始时广播自己的子公钥.密钥同步时间的长短需要根据网络状况具体设定,而密钥生成时间可设定为时间段内除去密钥同步时间的剩余时间.

4.3 子私钥在广播过程的保密性

根据上文的讨论可知,变色龙哈希的子私钥对于本文提出的账本修改方案十分关键,因此,需要保证子私钥在广播过程中的保密性.区块链系统在节点认证、数据加密过程中,天然采用基于非对称加密的公钥密码体制,例如比特币的ECDSA/ECC,以太坊的secp256k1.本文利用区块链系统自带的公钥密码体制,规定权限节点必须在使用接收方公钥对子私钥进行加密后,方可广播自己的变色龙哈希子私钥[19].值得注意的是,对于某个区块,参与该区块打包的权限节点一共有n个,即使攻击者通过某种方式,获得了其中m个权限节点的非对称加密私

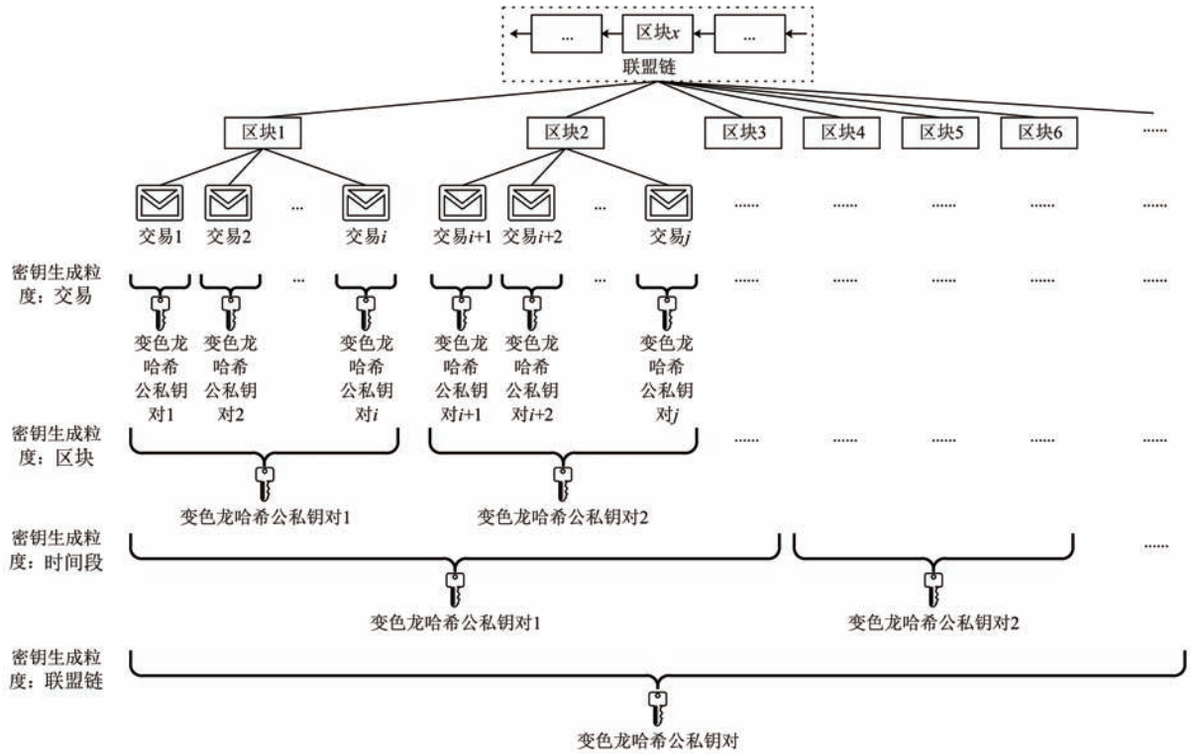

图4　变色龙哈希的密钥生成粒度

钥，再通过拦截这些节点的变色龙哈希子私钥消息，获取了变色龙哈希子私钥，只要$m \neq n$，根据改进后的变色龙哈希算法的性质，攻击者仍不能计算出变色龙哈希算法的系统私钥，从而无法修改该区块中的交易.

4.4　账本修改操作的问责性

考虑到账本修改操作的特殊性，系统需要对账本修改操作提案者的身份进行验证，并在必要的时候进行问责. 因此，本文规定，提案者必须对修改请求进行签名后[20]，才能广播修改请求. 在接收到提案请求后，其他权限节点通过比对账本修改请求与签名，验证提案者的身份，并将提案与签名作为修改记录存放到本地；当出现问题需要问责时，可以通过比对本地数据库中的修改记录找到提案者，并利用签名充当证据，对提案者进行问责. 修改提案包含的字段将在4.6节详细介绍.

值得注意的是，权限节点需要保存一定空间大小的修改记录以方便问责，虽然这给权限节点带来了额外的存储代价，但为了区块链的稳定性和问责性，权限节点有责任保存这些记录；同时，只有权限节点而非所有节点需要保存这些修改记录，对于普通节点来说，它们仍只需存储主链数据，因此，系统中普通节点的存储代价不变或相对减少.

4.5　权限节点的入职与离职

本文中的账本修改操作需要参与该交易打包的所有权限节点的变色龙哈希子私钥，考虑一种特殊情况，当某一权限节点退出联盟链，没有交出自己的子私钥，那么其参与打包的所有交易都将无法被修改. 为了解决这个问题，本文引入了保证金机制，促使权限节点离职时，主动交出它所有的变色龙哈希子私钥.

(1)权限节点入职

在节点A加入联盟链后，若想成为权限节点，A需要在智能合约内抵押一笔防止其作恶的保证金(链上通证)，并向其他权限节点提交申请；其他权限节点验证智能合约内是否有A的保证金，并审核其是否有资格成为权限节点；若同意A成为权限节点的票数超过设定好的阈值，则节点A成功成为新的权限节点. 权限节点入职的流程如图5所示. 智能合约中，权限节点入职投票的伪代码如算法1所示.

算法1.　权限节点入职投票

voteForEntry($address_{own}$, $address_A$, $isAgreed$)

输入：投票者的地址$address_{own}$，候选节点A地址$address_A$，是否同意节点A入职$isAgreed$

输出：若投票成功则输出TRUE，反之输出FALSE

1.　IF $node[address_{own}].role ==$ 权限节点THEN

2.　/*$haveVoted$用于防止节点为同一个申请多次投

票,初始值为FALSE*/
3. IF $haveVoted[address_{own}]$==FALSE THEN
4. IF $isAgreed$==TRUE THEN
5. $agreedNum_{entry}$++
6. IF$agreedNum_{entry}>threshold_{entry}$THEN
7. $node[address_A].role$=权限节点
8. END IF
9. END IF
10. $haveVoted[address_{own}]$=TRUE
11. RETURN TRUE
12. END IF
13. END IF
14. RETURN FALSE

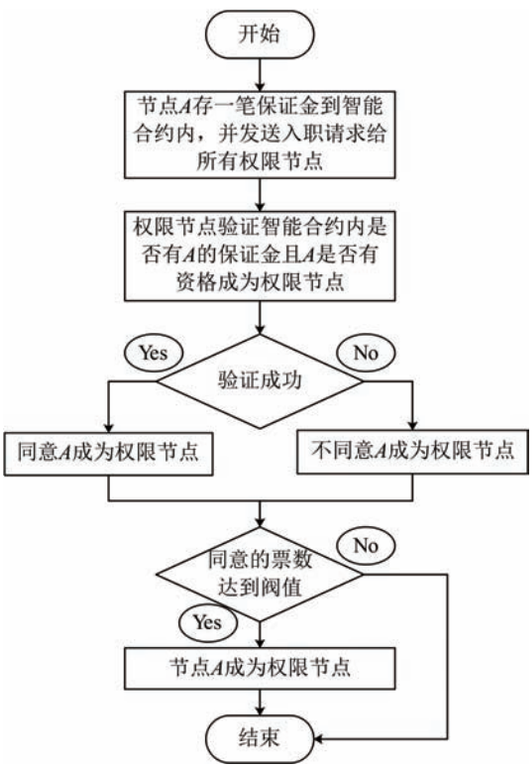

图5 权限节点的入职流程

(2)权限节点离职

若权限节点A想要退出联盟链,首先要完成离职手续,其需要向其他权限节点提出离职请求,并提供自己全部的子私钥;其他权限节点对子私钥进行验证,如果验证成功,则将其子私钥存储到本地,并同意其离职;当同意的票数超过设定好的阀值,则代表系统同意其离职,共识成立后,A取回在智能合约里的保证金,然后退出联盟链;如果权限节点A没有完成离职手续而直接消失,它将损失它在智能合约中的保证金.权限节点离职的流程如图6所示.权限节点离职投票的伪代码如算法2所示.

算法2. 权限节点离职投票

voteForExit$(address_{own}, address_A, isAgreed)$

输入:投票者的地址$address_{own}$,预离职节点A地址$address_A$,是否同意节点A离职$isAgreed$

输出:若投票成功则输出TRUE,反之输出FALSE

1. IF $node[address_{own}].role$==权限节点THEN
2. IF $haveVoted[address_{own}]$==FALSE THEN
3. IF $isAgreed$==TRUE THEN
4. $agreedNum_{exit}$++
5. IF$agreedNum_{exit}>threshold_{exit}$ THEN
6. $node[address_A].role$=离职节点
7. /*节点A完成离职后,其存放在合约内的保证金允许被提走*/
8. $node[address_A].isFrozen$=FALSE
9. END IF
10. END IF
11. $haveVoted[address_{own}]$=TRUE
12. RETURN TRUE
13. END IF
14. END IF
15. RETURN FALSE

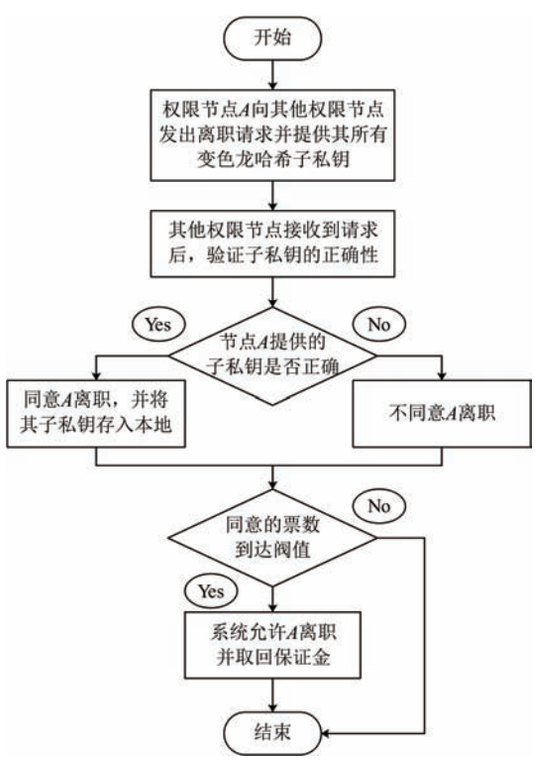

图6 权限节点的离职流程

(3)保证金动态调整

保证金机制作为一种激励和制约手段,虽无法

从根本上阻止权限节点作恶，但可以增加其作恶代价，使得权限节点实施作恶的成本远大于其作恶后的期望收益，从而有效抑制权限节点作恶的冲动. 这正如区块链中采用哈希函数验证方法可以检测数据可能存在的篡改，却无法100%保证数据不能被篡改一样（因为哈希函数存在碰撞的可能，虽然发现碰撞的计算代价极高）.

如果权限节点仅仅在入职时抵押一笔定额的保证金，那么会出现一种情况，当其所参与生成的变色龙哈希公私钥足够多时，其非法离开联盟链的不利影响可能会大于其作恶成本，即失去抵押的保证金. 因此，本文设计将保证金抵押数量定期进行调整，保证每个权限节点所掌握的变色龙哈希子私钥个数与其所抵押的保证金数量所匹配，以尽可能的放大权限节点作恶的损益. 简单来说，系统内的权限节点达成一种共识，每个权限节点需要在智能合约内抵押与其拥有的变色龙哈希子私钥数量匹配的保证金，只有遵守该共识的权限节点才可以参与之后的变色龙哈希公私钥生成过程. 该共识具体流程如下：

在权限节点A入职并抵押一笔金额为a的保证金后，其可以参与之后x对变色龙哈希公私钥的生成；当A参与超过x对变色龙哈希公私钥的生成过程后，其需要补交一笔金额为b的保证金，只有其补交了新保证金后，系统才允许其参与之后y对变色龙哈希公私钥的生成. 注意，这里所提及的a,b,x,y都是动态调整的，链上节点联盟可根据自身要求设置合理取值. 在变色龙哈希公私钥生成阶段，每个权限节点会收到其他节点传来的该阶段的变色龙哈希子公钥，所有权限节点需要检查这些子公钥是否来自于权限节点，同时调用isDepositSufficient()函数查询子公钥的生成者在智能合约内抵押的保证金是否足够，若检查通过，则将合规的子公钥保存，作为该阶段系统公钥的一部分，若检查未通过，则丢弃不合规的子公钥. isDepositSufficient()函数的伪代码如算法3所示.

算法3.　查询保证金是否充足

isDepositSufficient(*address*)

输入：需要查询的权限节点地址*address*

输出：若保证金充足则输出TRUE，反之输出FALSE

```
IF node[address].role == 权限节点 THEN
/*a为成为权限节点必须抵押的基础保证金数量，可允许权限节点参与x对变色龙哈希公私钥的生成过程*/
    IF node[address].deposit >= a THEN
/*求出权限节点最大可参与生成的变色龙哈希公私钥对数*/
maxKeyNum = ⌊(node[address].deposit − a)/b⌋ * y + x
        IF node[address].KeyNum < maxKeyNum THEN RETURN TRUE
        END IF
    END IF
END IF
RETURN FALSE
```

这种共识通过将权限节点拥有的子私钥数量与保证金金额相关联，有效的增加权限节点作恶的代价，但同时也可能导致智能合约内抵押的保证金过多，对诚实节点相对不公. 因此，本文提供了一个可选方案，该方案允许权限节点在合适的时候取回抵押的保证金：系统设置一个检查点i，假设最新块是第n块，那么所有权限节点达成共识，只有第$n-i$块到第n块可以被修改，第$n-i$块之前的所有块都无法被修改. 因为第$n-i$块之前的所有块都处于无法修改的状态，所以这些块对应的变色龙哈希子私钥失去了作用，系统便可以允许参与这些子私钥生成的权限节点取回自己对应的保证金.

本节中设计的保证金机制旨在尽可能的防止权限节点作恶，但在极端情况下，可能仍存在某些恶意权限节点出于不正当的目的，即使损失大量保证金也不交出变色龙哈希子私钥，阻止某些交易被修改. 本文认为可编辑联盟链是一种在传统联盟链上拓展了修改、压缩功能的进阶区块链，恶意节点作恶导致某些交易无法被修改、压缩，也仅仅相当于可编辑联盟链的某些区块“退化”成了传统联盟链中的区块，对可编辑联盟链系统整体的影响不大. 若这种作恶行为经常发生，不会对本文方案带来安全性或性能上的损害，只会造成个别历史数据无法修改，但仍比原始区块链具有可编辑灵活性. 况且可适当提高保证金的抵押数目，并且提高权限节点的审核门槛，来减少作恶行为的出现.

4.6　联盟链账本修改功能设计

在本节中将讨论修改联盟链账本相关功能的具体设计，主要包括联盟链初始化、交易打包与交易修改.

（1）联盟链初始化

在联盟链初始化阶段，除了进行区块链初始化的原有操作外，还需要执行变色龙哈希算法的初始化函数，得到公共参数，并将公共参数存储在创世块中.

(2)交易打包

在交易打包之前,所有权限节点需要完成变色龙哈希密钥生成与子公钥的广播操作,每个权限节点需要根据其他权限节点的子公钥计算出系统公钥.在计算出系统公钥的情况下,可以进行交易的打包操作,流程如下:

系统在权限节点中挑选出一个记账者;记账节点生成区块,对每一条交易数据t_i,随机挑选一个可变参数r_i,再利用之前已经获得的系统公钥h,执行计算哈希函数,得到该交易的变色龙哈希值CH_i,并将其和对应交易数据t_i与可变参数r_i存放在区块体中,将该区块的变色龙哈希系统公钥存放在区块头中;记账节点将打包好交易的区块连接到上一个区块上,并广播.

(3)交易修改

当某交易内容t_i需要被修改成t_i'时,由联盟链中的一个权限节点发出交易修改提案$redactTx$,并广播给联盟链中的其他参与该区块生成的权限节点进行验证.交易修改提案$redactTx$包含提案发布者信息$proposerID$、需要被修改的交易所在的区块号$blockNum$、交易编号$txNum$、修改前的交易内容t_i、修改后的交易内容t_i'、修改原因$redactReason$、提案者的签名$proposerSig$以及提案状态$txState$,即$redactTx=\{proposerID, blockNum, txNum, t_i, t_i', redactReason, proposerSig, txState\}$,其中,提案状态$txState$可分为活跃、接受与拒绝三种,初始状态为活跃.

其他权限节点收到$redactTx$后将该提案存入本地数据库中,并对该消息的正确性与合理性进行考证,正确性是指该提案是否与签名相符,合理性是指该笔交易的修改是否合理.若验证成功,则向其他权限节点广播同意该提案的回复并签名.若所有权限节点都同意该提案,则每个权限节点将该提案的状态改为接受,并将自己关于这笔交易的变色龙哈希子私钥回复给提案节点,提案节点获得全部的子私钥后,即可算出关键的系统私钥s.若有权限节点反对该提案或是该提案的生存周期已过,则每个权限节点将该提案的状态改为拒绝;

提案节点在获得全部子私钥后,执行可变参数锻造函数,得到新可变参数r_i',最后广播交易修改消息,即被接受的修改提案和新可变参数r_i';

普通节点收到交易修改消息后,通过执行哈希验证函数,验证r_i',t_i'与CH_i是否匹配,若匹配则更新本地区块信息;权限节点收到交易修改消息后,先查询本地数据库中是否有该交易被修改的合法记录,再进行交易的完整性验证,若验证成功,则更新本地区块信息.交易修改操作的流程如图7所示.

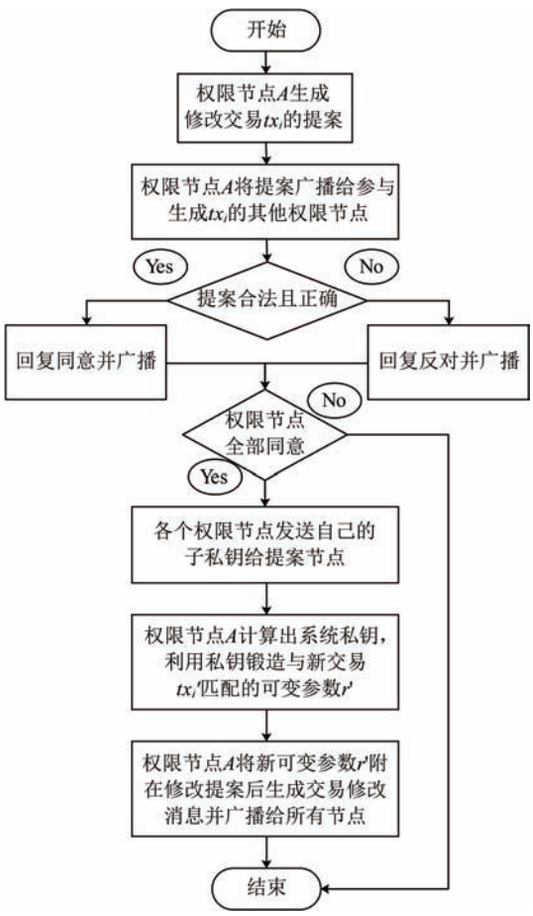

图7 交易修改操作流程

5. 性能分析与实验结果

在本章中将从方案压缩率、算法执行速率、权限节点的额外存储空间、安全性能与联盟链关键流程运行时间五个方面进行性能分析和测试.为了具体评估方案的性能,本文在以太坊1.8.21版本的基础上搭建了联盟链,所有的实验都在配置为Intel Core i7-8700 CPU、16GB内存、1TB硬盘的主机上测试.

5.1 方案压缩率

本联盟链中的区块数据,除了包含交易内容与交易哈希,还包括区块的其他数据,在对交易进行删除操作后,交易内容被删除,但交易哈希与可变参数以及区块的其他数据仍存在于区块中.本文中的压缩率$compressionRatio$概念,是指删除后的新区块大

小$size_{newBlock}$与未删除时的原区块大小$size_{oldBlock}$之比，即

$$compressionRatio = \frac{size_{newBlock}}{size_{oldBlock}}$$

某区块的压缩率$compressionRatio$的取值范围为$\left[k\%, 100\%\right]$，$k \in [\min, 100]$. 显然，当某区块未被修改时，其压缩率为100%；当某区块所有交易都被删除时，压缩率为$k\%$. 当区块中原本不存在交易时，k取100；当区块内的交易达到系统对于区块内交易大小定义的极限时，$k\%$取最小值min，但这个最小值min不低于0.

本文以以太坊区块数据为标准构造实验数据，在以太坊中，区块头大小为508 byte，一个区块平均拥有大约100笔交易，平均一笔交易大约占230 byte空间. 本次实验对5个拥有100笔交易的区块进行交易删除操作，分别删除其中1笔、10笔、50笔、全部交易. 通过区块大小统计，对比原区块大小与删除后的区块大小之比，从而大致估算方案的压缩率. 方案平均压缩率测试结果如表2所示. 理论上，不管删除多少笔交易，区块头大小不会发生变化，这是由于交易数据只存于区块体中，与区块头无关；区块体的大小随着被删除的交易变多而逐渐变小，直到等于$(size_r + size_{CH}) \times txNum$，其中$size_r$为可变参数$r$的大小、$size_{CH}$为变色龙哈希$CH$的大小、$txNum$为交易数. 在本次实验中，$size_r = 32$ byte、$size_{CH} = 32$ byte、$txNum = 100$个，因此删除全部交易后，区块体仍剩余6400 byte，与理论一致. 在本账本修改方案中，删除交易数与平均压缩率的关系为：

$$compressionRatio = \frac{size_{oldBlock} - txNum \times size_{tx}}{size_{oldBlock}}$$

表2　方案压缩率测试表

删除100笔交易中的x笔	平均区块头大小	平均区块体大小	平均删除交易数与原交易数之比	平均压缩率
0	508 byte	22677 byte	0%	100%
1	508 byte	22511.0 byte	1%	99.3%
10	508 byte	20856.5 byte	10%	92.1%
50	508 byte	15036.6 byte	50%	67.0%
100	508 byte	6400 byte	100%	29.8%

由于区块头数据大小恒定，区块体中的可变参数r、变色龙哈希值CH不会删除，因此当区块体足够大，即交易数目足够多时，方案的压缩率将低于30%.

5.2　算法执行速率

算法的执行时间与联盟链的交易修改效率有紧密联系. 执行速率$executionTime$概念是指算法结束时间$endTime$与算法开始时间$beginTime$的差值，即：

$$executionTime = endTime - beginTime$$

本文在安全参数为256，即变色龙哈希值与变色龙哈希公私钥的比特位为256，联盟链权限节点数为5，每2个区块为一个时间段，区块平均交易数为10的情况下，进行了十次实验. 每次实验重新从创世块开始生成可编辑联盟链，随机记录5个时间段中各个权限节点的变色龙哈希子密钥生成时间，5个区块共50笔交易的哈希时间，20笔交易的可变参数锻造时间，5个区块的交易验证时间. 算法执行速率测试结果如表3所示. 由结果可知，改进后的变色龙哈希算法总体执行速率为毫秒级，其中最耗时的是初始化函数，平均值为781.1ms，其他函数基本都在1 ms以内. 由于初始化操作只需要在联盟链初始化时执行一次，因此改进后的变色龙哈希算法对联盟链的影响较小.

表3　算法执行速率测试表

函数名	执行次数	最大值	最小值	平均值	方差
Setup()	10	1188 ms	376 ms	781.1 ms	81497.9 ms^2
KeyGen()	250	0.249 ms	0.143 ms	0.165 ms	0.001 ms^2
Hash()	500	0.389 ms	0.200 ms	0.266 ms	0.003 ms^2
Forge()	200	1.173 ms	0.536 ms	0.734 ms	0.042 ms^2
Verity()	500	0.375 ms	0.156 ms	0.210 ms	0.006 ms^2

5.3　权限节点的额外存储空间

为了保证一定程度的安全性，使用本文中的账本修改方案时，权限节点需要在本地存储一些额外的数据，权限节点的额外存储空间大小$extraSpace$无疑也是衡量方案好坏的指标之一. 本节将根据4.2节中关于变色龙哈希公私钥生成粒度的描述，对本文提出的四种生成粒度的额外存储空间大小与相关工作中的方案[10][11][12]的额外存储空间大小进行理论对比分析.

(1)本文提出的四种密钥生成粒度

假设变色龙哈希子公钥之积的大小为a，子私钥之和的大小为b，区块编号的大小为c，联盟链中一共有n个区块，交易编号的大小为d，平均一个区块中有m笔交易，时间段编号的大小为e，联盟链已经进行到第k个时间段. 对于本文提出四种密钥生

成粒度,它们的额外存储空间分别为:

①备选方案A:以交易为密钥生成粒度.每个权限节点在本地记录区块编号$blockNum$、交易编号$txNum$、交易对应的所有权限节点的子公钥之积h和该交易对应所有已公开的权限节点子私钥之和s_{public},即$\{blockNum, txNum, h, s_{public}\}$,所以该粒度下权限节点的额外存储空间:

$$extraSpace=(a+b+c+d)mn$$

②备选方案B:以区块为密钥生成粒度.每个权限节点需要在本地记录区块编号$blockNum$、该区块对应的所有权限节点的子公钥之积h和该区块对应所有已公开的权限节点子私钥之和s_{public},即$\{blockNum, h, s_{public}\}$,所以该粒度下权限节点的额外存储空间

$$extraSpace=(a+b+c)n$$

③备选方案C:以联盟链为密钥生成粒度.每个权限节点只需要在本地记录联盟链的所有权限节点的子公钥之积h和对应的所有已公开的权限节点子私钥之和s_{public},即$\{h, s_{public}\}$,所以该粒度下权限节点的额外存储空间

$$extraSpace=a+b$$

④选用方案:以时间段为密钥生成粒度.每个权限节点需要在本地记录时间段编号$timeNum$、该时间段对应的所有权限节点的子公钥之积h和该时间段对应所有已公开的权限节点子私钥之和s_{public},即$\{timeNum, h, s_{public}\}$,所以该粒度下权限节点的额外存储空间

$$extraSpace=(a+b+e)k$$

(2)文献[10]所提方案

在方案[10]中,整个联盟链仅有一对变色龙哈希公私钥对,即该方案的密钥生成粒度为联盟链.其中变色龙哈希的系统私钥由一个中心节点生成,并通过秘密共享将私钥拆分为多个系统私钥份额再分发给其他节点.假设变色龙哈希系统公钥的大小为a,系统私钥份额的大小为f.每个节点需要在本地记录联盟链的系统公钥h和自己的系统私钥份额$share_s$,即$\{h, share_s\}$,所以该粒度下权限节点的额外存储空间

$$extraSpace=a+f$$

(3)文献[11]所提方案

在方案[11]中,整个联盟链仅有一组变色龙哈希公私钥对,即该方案的密钥生成粒度为联盟链.其中每个权限节点各自生成变色龙哈希的一对子公私钥对,并将子公钥发送给其他权限节点.假设变色龙哈希子公钥的大小为a,子私钥的大小为b,全网有x个权限节点.每个权限节点需要在本地记录联盟链的所有子公钥$\{h_1, h_2, \cdots, h_x\}$和自己的子私钥$x_i$,即$\{\{h_1, h_2, \cdots, h_x\}, x_i\}$,所以该粒度下权限节点的额外存储空间

$$extraSpace=xa+b$$

(4)文献[12]所提方案

在方案[12]中,每个区块都有一组变色龙哈希公私钥对,即该方案的密钥生成粒度为区块.其中变色龙哈希的系统私钥由一个中心节点生成,该节点通过秘密共享将私钥拆分为多个系统私钥份额再分发给其他节点.假设变色龙哈希系统公钥的大小为a,系统私钥份额的大小为f,区块编号的大小为c,联盟链中一共有n个区块.每个权限节点需要在本地记录区块编号$blockNum$,该区块的系统公钥h和对应的自己的系统私钥份额$share_s$,即$\{blockNum, h, share_s\}$,所以该粒度下权限节点的额外存储空间

$$extraSpace=(a+f+c)n$$

密钥生成粒度的安全性与时间、空间代价的对比如表4所示.在实际应用中,系统私钥s与系统私钥份额$share_s$的大小几乎一致,可认为$f=b$,那么方

表4 密钥生成粒度的安全性与时间、空间代价对比表

方案	密钥生成粒度	权限节点的空间代价	系统通信时间代价	算法运行时间代价	安全性	额外存储空间大小	适用情况
本文备选方案	交易	高	高	高	高	$(a+b+c+d)mn$	不容有失型联盟链
	区块	较高	较高	较高	较高	$(a+b+c)n$	决策慎重型联盟链
	联盟链	低	低	低	低	$a+b$	时间敏感型联盟链
本文选用方案	时间段	适中	适中	适中	适中	$(a+b+e)k$	平衡型联盟链
方案[10]	联盟链	低	低	低	低	$a+f$	未提及
方案[11]	联盟链	低	低	低	低	$xa+b$	未提及
方案[12]	区块	较高	较高	较高	较高	$(a+f+c)n$	未提及

案[10]和方案[12]与本文备选方案C与备选方案B的额外存储空间大小相同. 对于方案[11],由于其需要存放每一个权限节点的子公钥,因此该方案的额外存储空间大小大于本文备选方案C的额外存储空间大小. 而本文选用的方案,相对于备选方案与方案[10][11][12],具备更高的灵活性,更适合应用于可编辑联盟链的实践中.

5.4　安全性能

安全性能也是本账本修改方案的重要衡量标准,在本文中主要表现为对权限节点作恶的抵抗性、变色龙哈希子私钥在同步过程中的保密性、修改操作执行者身份的可验证性、账本前后向的可验证性. 下面依次分析.

(1)对权限节点作恶的抵抗性

本文中主要将权限节点作恶方式分为两类,第一类是恶意权限节点利用算法漏洞,在不知道全部变色龙哈希子私钥的情况下,修改交易内容. 正如3.4节讨论的一样,恶意权限节点无法在不知道系统完整私钥的情况下,计算出新的可变参数,从而修改交易内容;第二类是恶意节点退出联盟链时,不主动交出自己全部的变色龙哈希子私钥,以阻碍交易的修改. 正如4.5节讨论的一样,本方案引入了保证金机制,以经济的手段促使权限节点在退出时主动交出自己的子私钥.

(2)变色龙哈希子私钥在同步过程中的保密性

变色龙哈希的子私钥对于本文提出的账本修改方案十分关键,掌握了全部子私钥的节点就可以修改交易内容,因此必须保证子私钥在同步过程中的保密性,防止其他节点获得正确的子私钥. 正如4.3节讨论的一样,本文在子私钥同步过程中引入非对称加密技术,在一定程度上保证子私钥的保密性.

(3)修改操作执行者身份的可验证性

考虑到账本修改操作的特殊性,系统需要对修改操作提案者的身份进行验证,并在必要的时候进行问责. 正如4.4节讨论的一样,本文要求权限节点在提出交易修改请求时,对请求内容进行签名,这样做的优点一是方便其他权限节点验证提案者的身份,二是在需要问责时,可以利用数字签名的不可否认性,找到交易修改请求提案者进行问责.

(4)账本前后向的可验证性

由于本账本修改方案不依赖于特定的区块链架构与区块结构且未修改与区块链账本前后向验证相关的操作. 因此,对于使用本文账本修改方案联盟链来说,依旧可以和其他主流区块链一样进行账本前后向验证.

5.5　关键阶段运行时间

为了实现交易修改与压缩功能,本文在传统区块链的基础上引入了变色龙哈希算法并增加相关的共识与额外的步骤,这肯定会导致与修改操作相关的流程运行时间变长. 本节对传统区块链、基于变色龙哈希的可编辑区块链以及本文提出的联盟链进行了对比实验,主要测试了它们在链初始化、区块生成、区块同步和交易修改四个关键阶段的运行时间. 由于文献[10]与[12]中方案在算法层面与关键阶段的步骤上区别不大,因此只选取方案[10]进行对比. 本次实验中,各个链都有15个节点,每个区块平均拥有100笔交易. 可编辑联盟链关键流程运行时间测试结果如图8所示.

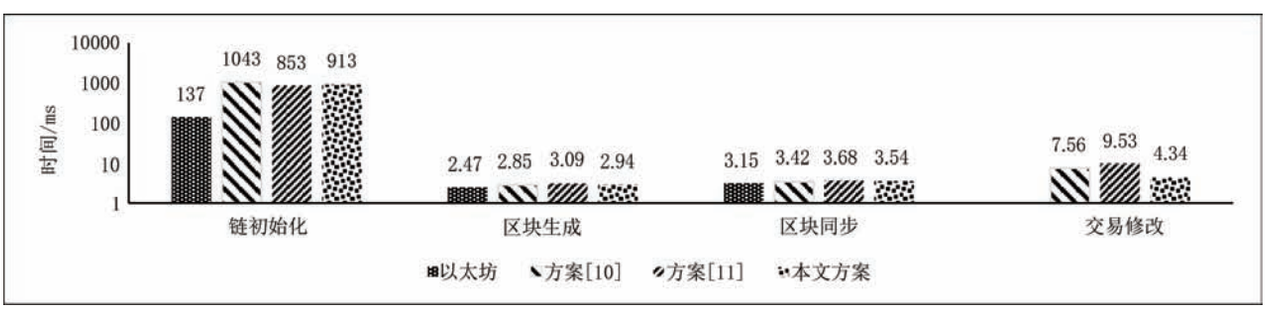

图8　关键阶段运行时间测试

在链初始化阶段,由于方案[10][11]和本文方案都基于的是变色龙哈希算法,因此需要在链初始化阶段完成变色龙哈希的初始化并将公共参数存入创世块中,相对于传统区块链会带来800 ms左右的开销;在区块生成阶段,方案[10][11]与本文方案将传统区块链中的父区块哈希或交易哈希替换成了变色龙哈希,其他步骤没有变化. 由于变色龙哈希算法比传统哈希算法的执行时间稍长,因此在此阶段基于变色龙哈希的区块链比传统区块链增加了0.5 ms左右的开销,几乎可以忽略不计;区块同

步阶段主要完成哈希验证与区块存储，方案[10][11]和本文方案由于采用了变色龙哈希算法，因此哈希验证使用的是变色龙哈希的验证子函数，该函数的运行时间略长于传统哈希算法的验证子函数，因此基于变色龙哈希的方案比传统区块链增加了0.4 ms左右的开销，对区块同步过程影响较小；在交易修改阶段，本次实验尝试将某个区块内的全部100笔交易进行删除，由于传统区块链没有交易修改功能，因此不做讨论，方案[10]需要在此阶段完成多方计算，时间开销较高，方案[11]虽不需要进行多方计算，但需要利用随机数生成协议与秘密分享技术挑选修改者，时间开销最高，本文方案无需多方计算也不需要挑选修改者，只需要利用变色龙哈希系统私钥锻造新可变参数即可完成修改，时间开销最低.通过对比实验可知，变色龙哈希算法在时间开销方面对联盟链的影响较小，本文提出的方案在链初始化、区块生成和区块同步阶段与现有其他方案差距不大，且在交易修改阶段的耗时较少.

6 应用前景

传统的区块链没有针对历史数据的修改或压缩操作，随着时间流逝，区块数据只增不减，想要存储所有的区块数据空间开销极大，对普通节点参与系统形成挑战.在实际应用中，某些区块由于时效、监管等原因，其中存储的数据失效或过期，对于区块链系统和所有用户来说，这些数据已经没有价值又占用大量的空间.如果在权限节点们的共识下，移除这些过期、无效数据，从而压缩区块大小，释放节点的存储空间，降低存储代价，意义重大.同时，传统的区块链也没有数据订正功能，如遇区块中存在违法违规信息，区块数据需要被删减；相关政策变化后，区块数据需要被更新；区块链应用平台中出现程序漏洞，错误的区块数据需要被修改，传统的区块链对此无能为力，这对区块链发展普及造成了极大阻碍，因此需要一种可编辑的区块链架构解决这些问题，而其中最核心的是适用于区块链的账本修改方案.本文提出的面向联盟链的账本修改方案的实际应用前景广阔，包含如下可能领域：

(1)防止区块链可任意储存消息的能力的滥用.如今部分区块链中包含反动淫秽内容、侵犯知识产权的材料等非法信息，用户可能会担心其设备上同步的区块链数据包含不当内容而被追责，因而不愿下载完整的区块链账本数据.故这些区块链系统中存储完整数据的全节点相对较少，导致系统安全性和数据可靠性下降.此外，如果不正当的内容不能从区块链中删除，就可能会永远影响人们的生活.当旧记录成为障碍时，添加新记录就不再成为一种选择.本文提出的账本数据修改方案可以在保持数据的共识确认、查询验证、完整一致的同时，为区块链弥补之前的能力“缺失”，通过屏蔽、删除这些非法信息，建立健康的区块链数据生态环境和应用场景系统.

(2)方便使用联盟链的企业在遇到突发情况时，用合理的手段修改历史数据.目前有不少企业级的区块链项目被提出，越来越多的公司根据自己的业务需求搭建联盟链平台.针对联盟链，研究其可编辑技术，当错误和突发情况出现时，平台就可以按照既定规则修改相应历史数据，允许企业解决人为错误，适应法律和法规要求，解决恶作剧和其他问题.

(3)为区块链提供一种新的治理规则，防止硬分叉产生.区块链缺乏治理规则，当遇到突发事件时，没有调整规则，只能通过软分叉或硬分叉来解决问题，这容易导致共识分歧和数据混乱，因此有必要建立区块链上的治理规则，提高区块链的风险抵抗能力，避免系统分裂.使用本账本修改方案的区块链，可以在不分叉的情况下，调整区块中的数据，达到数据治理的目的.

7 结 论

本文针对联盟链的修改需求，提出了一种通用的可编辑联盟链架构，实现了区块链账本的修改与压缩功能.通过对变色龙哈希算法进行改进，设计了一种多中心化且兼容当前主流区块链的账本修改方案.分析可知，本文中的账本修改方案安全可信、可控可溯，系统的通信时间与权限节点的额外存储空间大小可灵活调节.实验显示，修改后的变色龙哈希算法的执行效率可达毫秒级，区块压缩率最低可达30%，采用变色龙哈希在时间开销层面对联盟链影响较小且本文的改进方案相对于现有方案在修改操作上的耗时更少.

预计未来工作将包括如下方面：

(1)考虑密钥生成粒度为时间段的方案中，密钥生成时间和密钥广播时间的具体长短；

(2)优化变色龙哈希算法，进一步加快算法执行时间并且降低压缩率；

(3)优化权限节点的变色龙哈希密钥、修改提案存储方案，减少权限节点的额外存储空间.

参 考 文 献

[1] Yang F, Shi Y, Wu Q, et al. The survey on intellectual property based on blockchain technology//Proceedings of the 2019 IEEE International Conference on Industrial Cyber Physical Systems (ICPS). Tai Pei, China, 2019: 743-748

[2] Tsai W T, Feng L, Zhang H, et al. Intellectual-property blockchain-based protection model for microfilms//Proceedings of the 2017 IEEE Symposium on Service-Oriented System Engineering (SOSE). San Francisco,USA, 2017: 174-178

[3] Regulation P. Regulation (EU) 2016/679 of the European parliament and of the council. REGULATION (EU), 2016, 679: 2016

[4] Cyberspace Administration of China. Blockchain information service management regulations. 2019(in Chinese)
(中华人民共和国国家互联网信息办公室. 区块链信息服务管理规定. 2019)

[5] Matzutt R, Hiller J, Henze M, et al. A quantitative analysis of the impact of arbitrary blockchain content on bitcoin// Proceedings of the International Conference on Financial Cryptography and Data Security. Nieuwpoort, Curaçao, 2018: 420-438

[6] Huang K, Zhang X, Mu Y, et al. Building redactable consortium blockchain for industrial Internet-of-Things. IEEE Transactions on Industrial Informatics, 2019, 15(6): 3670-3679

[7] Ren Y, Xu D, Zhang X, Wu D. Deletable blockchain based on threshold ring signature. Journal on Communications, 2019, 40(04):71-82(in Chinese)
(任艳丽,徐丹婷,张新鹏,谷大武. 基于门限环签名的可删除区块链.通信学报, 2019, 40(04):71-82)

[8] Ren Y, Xu D, Zhang X, Wu D. A scheme of revisable blockchain. Journal of Software, 2020,31(12):3909-3922(in Chinese)
(任艳丽,徐丹婷,张新鹏,谷大武.可修改的区块链方案.软件学报,2020,31(12):3909-3922)

[9] Krawczyk H M, Rabin T D. Chameleon hashing and signatures, U.S., 2000-8-22

[10] Ateniese G, Magri B, Venturi D, et al. Redactable blockchain-or-rewriting history in bitcoin and friends//Proceedings of the 2017 IEEE European Symposium on Security and Privacy (EuroS&P). Paris, France, 2017: 111-126

[11] Li P, Xu H, Ma T, Mu Y. Research on fault-correcting blockchain technology. Journal of Cryptologic Research, 2018, 5(5): 501-509(in Chinese)
(李佩丽,徐海霞,马添军,穆永恒. 可更改区块链技术研究. 密码学报, 2018, 5(05):501-509)

[12] Ashritha K, Sindhu M, Lakshmy K V. Redactable blockchain using enhanced chameleon hash function//Proceedings of the 2019 5th International Conference on Advanced Computing & Communication Systems (ICACCS). Coimbatore, TN, India, 2019: 323-328

[13] Crosby M, Pattanayak P, Verma S, et al. Blockchain technology: Beyond bitcoin. Applied Innovation, 2016, 2(6-10): 71

[14] Gao W, Hatcher W G, Yu W. A survey of blockchain: techniques, applications, and challenges//Proceedings of the 2018 27th international conference on computer communication and networks (ICCCN). Hangzhou, China, 2018: 1-11

[15] Zou J, Zhang H, Tang Y, et al. Blockchain technical guidelines. Beijing:China Machine Press, 2016(in Chinese)
(邹均,张海宁,唐屹,等. 区块链技术指南. 北京:机械工业出版社, 2016)

[16] Nakamoto S. Bitcoin: A peer-to-peer electronic cash system. White Paper, 2008

[17] Shao Q, Jin C, Zhang S, Zhou A. Blockchain: Architecture and research progress. Chinese Journal of Computers, 2018, 41(05):969-988.(in Chinese)
(邵奇峰,金澈清,张召,等. 区块链技术:架构及进展. 计算机学报, 2018,41(05):969-988)

[18] Cai X, Deng X, Zhang L, et al. The principle and core technology of blockchain. Chinese Journal of Computers, 2021, 44(1): 84-131(in Chinese)
(蔡晓晴,邓尧,张亮,等. 区块链原理及其核心技术. 计算机学报, 2021, 44(1): 84-131)

[19] Zheng Z, Xie S, Dai H, et al. An overview of blockchain technology: Architecture, consensus, and future trends// Proceedings of the 2017 IEEE international congress on big data. Honolulu, USA, 2017: 557-564

[20] Bos J W, Halderman J A, Heninger N, et al. Elliptic curve cryptography in practice//Proceedings of the International Conference on Financial Cryptography and Data Security. Christ Church,Barbados, 2014: 157-175

LV Wei-Long, M. E. candidate. His main interests include blockchain technology and cryptography.

WEI Song-Jie, Ph. D., associate professor. His research interests include blockchain and network security.

YU Ming-Hui, M. E. candidate. Her research interests include information system security and application.

LI Sha-Sha, M. E. candidate. Her research interests include distributed system and task scheduling.

Background

The problem studied in this paper relates to the field of network security and blockchain system. As one of the most significant characteristics of blockchain, immutability refers to that historical data on a blockchain ledger cannot be changed once confirmed. On one hand, this guarantees the integrity and reliability of blockchain-stored historical data. However, on the other hand, it also disables a blockchain from correcting ledger data with mistakes, removal of invalid data, or compression of expired data, which may cause a blockchain application to fail to comply with legal requirements. For example, China, E. U. and other countries require blockchain service providers to have the emergency management capability to delete or modify illegal data in blockchain ledgers. The emerging blockchain ledger redaction method may alleviate the immutability appropriately, and allow a blockchain to compress the blocks and redact the historical data for correctness, which is highly valuable appreciable in practice.

Currently there exist a few papers researching on blockchain ledger redaction methods internationally, but with a lack of high-level architecture design. Specifically, existing blockchain ledger redaction methods can be classified into two categories: ledger redaction methods based on threshold ring signature and chameleon hash ledger redaction methods. Some of these methods are based on specific block structures or blockchain architectures, so they do not have the universality and compatibility across variant blockchain platforms. In other ledger redaction methods, the processes of the redaction operations require the intervention of some trusted third parties or a centralized single node, which undermines the decentralization of the blockchain system, and is likely to suffer single point of failure or trust issues.

This paper presents a distributed optimization approach to the original chameleon hash algorithm. By allowing multiple nodes to generate public/private key pairs for the chameleon hash, the improved algorithm ensures the highly decentralized processes of key generation and usage. Based on the improved algorithm, a completely decentralized and highly compatible ledger redaction procedure is proposed and described in detail. Furthermore, in order to evaluate time, space cost and security robustness, several private-key generating and synchronizing consensus mechanisms are designed and benchmarked in all aspects. Meanwhile, in order to further perfect the proposed method, the usability and accountability of the blockchain system are considered in special cases. Experiments and analyses show that, applying the ledger redaction method proposed, the best block compressing rate can reach 30%, the algorithm executing rate achieves millisecond level, and the improved algorithm has proved safety.

This work is partially supported by the grants from the National Natural Science Foundation of China (No. 61802186, No. 61472189).