计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.210400027

基于改进 P2PKHCA 脚本方案的比特币密钥更新机制

向阿新[1,2]　高鸿峰[1,3]　田有亮[1,2,4]
1 贵州大学计算机科学与技术学院　贵阳 550025
2 贵州大学密码学与数据安全研究所　贵阳 550025
3 贵州大学网络与信息化管理中心　贵阳 550025
4 公共大数据国家重点实验室　贵阳 550025
(Xax18885248968@163.com)

摘　要　比特币是最成熟的公有链应用系统之一,用户密钥是比特币所有权确定过程的关键,比特币的安全由用户密钥的安全管理所保证,密钥的遗失会导致大量的用户资产流失,因此实现流失资产找回是亟待解决的问题。针对以上问题,提出了基于改进 P2PKHCA(具有条件匿名的支付到公钥哈希)脚本方案的比特币密钥更新机制。首先,通过引入密钥生命周期和随机数改进 P2PKHCA 方案中的密钥生成算法,以解决其存在的密钥泄露问题;其次,提出两个新的操作符 OP_KEYUPDATE 和 OP_TSELECTION 来设计新的密钥更新脚本,以实现比特币系统的用户密钥更新;最后,基于密钥更新脚本构造两种密钥更新方案,使得密钥更新脚本适用于不同的密钥更新应用需求。对密钥更新机制进行的安全性分析和性能分析表明,所提机制在有效完成用户密钥更新的前提下,能够实现比特币系统中流失比特币的找回。

关键词:区块链;比特币;密钥更新;脚本方案;流失比特币找回
中图法分类号　TP3;TP311.13

Key Update Mechanism in Bitcoin Based on Improved P2PKHCA Script Scheme

XIANG A-xin[1,2],GAO Hong-feng[1,3] and TIAN You-liang[1,2,4]
1 College of Computer Science and Technology,Guizhou University,Guiyang 550025,China
2 Institute of Cryptography & Date Security,Guizhou University,Guiyang 550025,China
3 Network and Information Management Center,Guizhou University,Guiyang 550025,China
4 State Key Laboratory of Public Big Data,Guiyang 550025,China

Abstract　Bitcoin is one of the most mature public chain application systems,the user key is the critical factor to the process of determining the ownership of Bitcoin,the security of Bitcoin is guaranteed by the safe management of the user key,and the loss of the key will lead to the loss of a large number of user assets. So it is an urgent problem to recover the lost assets. This paper proposes a key update mechanism in Bitcoin based on the improved P2PKHCA (pay-to-public-key-hash-with-conditional-anonymity) script scheme to solve above problems. Firstly,the key generation algorithm in the P2PKHCA scheme is improved by introducing the key life cycle and random number to solve its key leakage problem. Secondly,the two new opcodes,OP_KEYUPDATE and OP_TSELECTION,are proposed to design the new key update script to realize the user key update of the Bitcoin system. Finally,two types of key update schemes based on the key update script are constructed to make the script suitable for the requirements of different key update applications. The security analysis and performance analysis of the key update mechanism show that the proposed mechanism realizes the recovery of lost Bitcoins in the Bitcoin system on the premise of the effective completion of update of user's key.

Keywords　Blockchain,Bitcoin,Key update,Script scheme,Recovery of lost Bitcoins

到稿日期:2021-04-01　返修日期:2021-08-09
基金项目:国家自然科学基金(61662009,61772008);贵州省科技重大专项计划(20183001);国家自然科学基金联合基金重点支持项目(U1836205);贵州省科技计划项目(黔科合基础[2019]1098,ZK[2021]一般 331,ZK[2021]一般 325);贵州省高层次创新型人才项目(黔科合平台人才[2020]6008);贵阳市科技计划项目(筑科合[2021]1-5)
This work was supported by the National Natural Science Foundation of China(61662009, 61772008),Science and Technology Major Support Program of Guizhou Province(20183001), Key Program of the National Natural Science Union Foundation of China(U1836205), Science and Technology Program of Guizhou Province([2019]1098,ZK[2021]general 331,ZK[2021]general 325),Project of High-level Innovative Talents of Guizhou Province([2020]6008) and Science and Technology Program of Guiyang([2021]1-5).
通信作者:高鸿峰(hfgao@gzu.edu.cn)

1 引言

2008年,化名为“中本聪”的学者在其发表的一篇名为《比特币:一种点对点的数字货币》[1]的白皮书中首次提出了区块链的概念。区块链技术是一种分布式存储技术,其具有不可篡改、去中心化、去信任和共享数据库等特点。经过12年的研究与发展,区块链技术在医疗、金融和物联网等[2-4]领域被大力探索,相关的理论型文章及应用型文章相继发表。比特币作为目前公有链应用最成熟的区块链系统之一,被给予巨大的期望。虽然比特币系统至今一直运行稳定,但是其始终面临着一个关键问题:流失比特币的找回。因为比特币交易的完成需要用户经过严格的私钥签名验证过程,除了持有者之外,无人可以使用其私钥。因此,如果用户密钥遗失,将导致该密钥地址下的比特币永久流失。比特币系统通常被称为通货紧缩的系统,在比特币总量2100万保持不变的情况下,预计到2140年将不会有新的比特币发放。比特币流失使得比特币可用总数减少,从而影响比特币系统的长久正常运行。

比特币独特的匿名性使得传统的流失资产找回方法不再适用,虽然其匿名性使得网络中的节点可实现去信任的数据交换,但是完全匿名性将促使恶意攻击者频繁地通过分散的比特币地址进行恶意交易,从而滋生犯罪,而且无法对其进行身份追踪,因此Ouimet[5]强调了比特币系统中用户匿名性和可追踪性共存的必要性。针对这一公开问题,Li等[6]通过引入身份管理者(如当地银行、凭证机构等可信机构)和防篡改设备,提出了一种具有可追踪性的比特币脚本方案P2PKHCA,从用户密钥构造的角度实现对比特币中恶意交易用户的身份追踪。该方案有效解决了比特币系统中用户匿名性和可追踪性的权衡问题。另外,在比特币技术拓展研究方面,已有研究对比特币脚本、智能合约设计、用户身份的可追踪和比特币区块链扩容等[7-12]方面进行了探索。为了防止比特币永久遗失(因用户遗忘密钥或者用户死亡等情况导致其地址下的比特币不能再使用),在保证比特币匿名性的前提下实现用户密钥的安全管理是目前比特币系统亟需解决的挑战性难题之一。

比特币的安全由密钥的安全所保证,因此用户密钥的安全管理是比特币系统的研究热点。目前比特币系统的密钥管理研究主要针对用户私钥的安全存储,即对钱包的研究。钱包大致分为冷存储和热存储两种,冷存储主要是指将私钥存储在离线的设备中,不接入互联网,但是其使用和携带都不太方便且易遭受物理攻击;热存储是将私钥放在个人电脑内,相当于在线存储,虽然使用很方便,但是易遭到黑客的攻击。然而,无论密钥怎样进行安全的存储,都存在“存储时间越久密钥泄露或遗失的可能性越大”的问题。因此如何在密钥遗失之后找回比特币是当下比特币系统亟需解决的问题。密钥的更新方法可以使密钥向下一个版本演进,并实现对当前版本密钥及用户身份的追踪认证,从而完成旧密钥资产到新密钥资产的转移。

本文的工作重心是实现流失比特币的找回,通过设计适用于比特币系统的密钥更新方案,完成对密钥的定期更新或请求更新,并基于Li等[6]提出的P2PKHCA脚本方案中的用户身份可追踪性,实现密钥遗失后找回流失的比特币。本文的主要贡献有以下3个方面:

(1) 构造适用于比特币密钥更新的密钥生成算法。引入密钥生命周期 T_e 和随机数 l 两个变量来改进Li等[6]提出的P2PKHCA脚本方案中的密钥生成算法,解决其存在的密钥隐私泄露问题。

(2)设计新的密钥更新脚本。基于改进的P2PKHCA脚本方案设计两个新的脚本操作符,即密钥更新操作符OP_KEYUPDATE和密钥生命周期判断操作符OP_TSELECTION,并基于上述两个新操作符设计密钥更新脚本,实现比特币系统中用户密钥的定期更新。

(3)基于密钥更新脚本构造比特币系统中的两种密钥更新方案,使得密钥更新脚本适用于两种不同的更新现实场景,从而解决因密钥遗失而造成的比特币流失问题。

本文第2节阐述了关于比特币技术拓展和密钥管理方法的相关工作;第3节介绍了本文涉及的基础知识;第4节详细阐述了比特币系统中密钥更新机制的设计细节;第5节对提出的密钥更新机制进行了安全性和性能分析;最后对全文进行总结。

2 相关工作

比特币自2008年被提出以来就因其独有的特性而被广泛研究与应用。在比特币系统的扩展研究方面,Eskandari等[7]提出了一个关于比特币的密钥管理的评估框架,其虽然初步引发了人们对比特币系统中密钥管理可用性限制的思考,但是仅对密钥管理进行了调查评估,因此想要获得创新性突破仍需进一步的调查研究;Ittay等[8]提出了一种新的区块链协议(Bitcoin-NG),解决了比特币块链协议中固有的可伸缩性限制问题,在交易吞吐量和延迟方面得到巨大提升,其次引入新的度量以量化比特币类区块链协议的安全性和有效性;Möser等[9]提出一种比特币脚本语言的拓展原语covenants,其功能是限制资金的使用,同时基于covenants提出两种新的安全结构vaults和poison transactions,旨在解决比特币私钥安全可靠的存储和惩戒双花攻击;O'Connor等[10]在Möser等[9]提出的原语covenants中引入新的脚本操作符CAT和CHECKSIGFROMSTACK,加强实现covenants的部署,以解决其中存在的脚本规范问题;Poulami等[11]提出一种在比特币系统中真正实用的复杂智能合约框架(FASTKITTEN),重点是在比特币网络下以低成本执行任意复杂的智能合约,解决比特币系统不支持智能合约的限制和实现简单智能合约存在的设计复杂及开销较大等问题;Yu等[12]调查并分析了现有的链上/链下扩容方案,提出适应社区的比特币可行扩容路线方案。

在密钥管理方法研究方面,Brengel等[13]对比特币系统中用户密钥的泄露进行了调查研究,并通过两种不同的方式识别密钥的泄露,从而得出ECDSA随机数的重复使用是目前比特币系统一直存在的问题,而随机数的重复使用会引起弱密钥问题,从而导致密钥易遭到泄露攻击;Li等[14]构建了重复数据删除系统模型和数据加密密钥更新模型,重点在于解决数据重复存储问题,其中密钥更新方法主要分为PKI发

起密钥更新请求和数据存储服务器主动更新密钥两种，但是仍存在密钥的安全存储问题；Hong 等[15]将基于属性加密的方法应用到移动多媒体传感网络中，同时将时间分割为离散时间段绑定到密钥更新的环节中，解决传统系统密钥频繁更新引起的资源消耗大等问题，从而提升整个系统的性能；Tian 等[16]提出一种基于区块链的安全密钥管理方案（BC-EKM）来保证动态无线传感器网络的可信度，不仅解决了因动态特性引起的密钥安全管理问题，而且减少了传统分布式密钥管理带来的高资源消耗，最终实现安全性强度与资源消耗的平衡；Li 等[17]提出一种用于云数据安全审计的密钥更新和身份验证者演进机制，以保证存储文件的零知识隐私，该机制利用代理单向重签名的思想，有效地解决了密钥更新问题，同时实现了数据的隐私保护和完整性审计；Athmani 等[18]为了解决异构无线传感器网络中密钥安全分配的问题，提出一种高效的动态认证和密钥管理方案，在提升安全性的同时防止密钥被捕获；Nicolas 等[19]针对当下密码货币、区块链钱包或系统中密钥管理存在的鲁棒性不足和遭受恶意攻击等问题，阐述了密钥管理的发展历史和隐匿地址及其重要性，并提出改进的隐匿地址方法，目的是在增强区块链系统中密钥管理的鲁棒性的同时不增大交易容量；Stanislaw 等[20]在数据存储方面提出一种可更新的遗忘密钥管理系统，其基于遗忘伪随机函数设计了一种密钥管理方案，相比传统基于包装的密钥管理系统，保证了密钥传输的无条件安全和可验证性，通过引入可更新加密的概念提出了更有效和更安全的方法来解决密钥更新问题；Albakri 等[21]提出一种基于二元多项式的许可区块链密钥管理方案，旨在减少实体传输和存储的信息量，该方案需要预部署，即实体被预先加载令牌，极大地提高了密钥生成的效率，但其缺乏撤销机制，还有待改进。

3 预备知识

3.1 比特币脚本

比特币脚本就是比特币的工程控制语言，它是堆栈式且线性的，意味着每个指令只被执行一次，无法循环执行。常见的比特币交易是通过某人的签名去获得他在前一笔交易中获得的资金，在这种情况下，交易的输出必须描述为“凭借哈希值为 X 的公钥以及这个公钥持有者的签名，才可以获得这笔资金”，这实际上就是最常见的比特币脚本。比特币脚本包括输出脚本（scriptPubKey）和输入脚本（scriptSig），二者结合应用才能完成比特币交易。

图 1 给出了两个脚本的结合案例（其中虚线上方为输入脚本，虚线下方为输出脚本），这是目前比特币系统应用最广泛的 P2PKH（pay-to-public-key-hash）脚本方案。

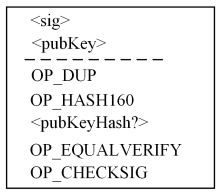

图 1 P2PKH 脚本范例

Fig. 1 Script example of P2PKH

表 1 阐述了本文所涉及的比特币脚本工作语言的指令或值及其功能。

表 1 比特币脚本中的指令或值及其功能

Table 1 Instructions or values and their functions in the Bitcoin script

指令或值名称	功能
〈sig〉	P2PKH 方案中的签名
〈pubKey〉	P2PKH 方案中的节点公钥
OP_DUP	复制堆栈顶端数据
OP_HASH160	计算哈希函数两次：第一次用 SHA-256，第二次用 RIPEMD-160
〈pubKeyHash〉	P2PKH 方案中节点公钥的哈希值
OP_EQUZALVERIFY	判断输入值是否相同，相同则返回真值，不同则返回假值并且交易作废
OP_CHECKSIG	P2PKH 方案中检查输入的签名是否有效
〈pseudonym〉	P2PKHCA 方案中的节点公钥
〈casig〉	P2PKHCA 方案中的签名
OP_TRACEID	追踪用户真实身份
OP_KEYUPDATE	本文中用于更新节点密钥
OP_TSELECTION	本文中用于判断密钥是否过期
OP_EQUAL	判断输入是否相同，相同则返回真值，不同则返回假值并且交易作废
OP_CHECKCASIG	P2PKHCA 方案中检查输入的签名是否有效
OP_SWAP	堆栈顶的前两项交换位置
OP_DROP	取出堆栈顶元素

执行比特币脚本只会产生两个结果：要么成功执行，在这种情况下交易有效；要么脚本执行出现错误，在这种情况下，整个交易无效且不能记入区块链。在堆栈语言里面执行一个脚本，我们只需要一个堆栈来垒积数据，不需要分配任何内存和变量。比特币脚本语言主要有两类指令：数据指令（尖括号中的数据）和工作码指令（以字母 OP 开头的数据）。数据指令主要是将数据堆到堆栈顶部；工作码指令主要是使用堆栈顶部的数据作为输入值计算一个函数或算法。

图 2 给出了 P2PKH 方案的执行堆栈状态，图中底部元素为脚本指令，上方为执行指令之后的堆栈状态。

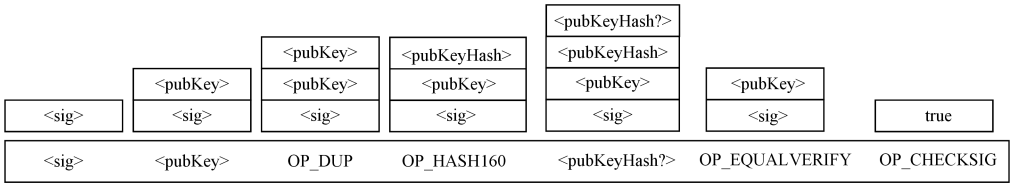

图 2 P2PKH 方案的执行堆栈状态图

Fig. 2 Execution stack-state diagram of P2PKH scheme

3.2 P2PKHCA 方案

P2PKHCA 方案是由 Li 等[6]提出的，其主要包括锁定及非锁定脚本和身份追踪脚本两个部分，目的是实现对比特币恶意交易中节点身份的追踪，但是需要牺牲一定的节点匿名

性以实现对比特币交易中恶意行为的监控，即实现有条件的匿名。脚本1描述了锁定及非锁定脚本为完成交易而进行的正确性验证，脚本2描述了身份追踪脚本以实现对恶意交易节点的真实身份追踪。

脚本1 〈casig〉〈pseudonym〉OP_DUP OP_HASH160 〈PseHash〉OP_EQUAL OP_CHECKCASIG

脚本2 〈casig〉〈pseudonym〉OP_SWAP OP_DROP OP_TRACEID

以上脚本的执行过程类似3.1节中P2PKH方案的堆栈执行过程。本节主要详细阐述P2PKH方案中密钥生成算法的具体构造过程。

首先，由身份管理者(如银行等可信机构)生成系统参数，使$\{a,b,G,P,q,Y_{cpk},spk,H_1,H_2,H_3,H_4\}$公开可见，将$\{X_{cpk},ssk\}$秘密地保存。其中$a,b\in F_P$($P$为大素数)，$G$为加法群，$P$是$G$的生成元，$q$为$G$的阶，主组合私钥$X_{cpk}=(x_1,x_2,\cdots,x_n)$($x_i\in Z_q^*$为随机数)，主组合公钥$Y_{cpk}=(y_1,y_2,\cdots,y_n)$($y_i=x_i\cdot P,i\in[1,n]$)，系统私钥$ssk\in Z_q^*$为随机数，系统公钥$spk=ssk\cdot P$，哈希函数$H_1:\{0,1\}^*\rightarrow Z_q$，$H_2:G\rightarrow Z_q$，$H_3:\{0,1\}^*\times\{0,1\}^*\times\{0,1\}^*\rightarrow Z_q$，$H_4:G\rightarrow Z_q$。同时，身份管理者将$\{ssk,X_{cpk},PIN,rid\}$预载到防篡改设备中，其中$rid$为用户节点的真实身份，$PIN$为用户的个人识别符。

然后，防篡改设备生成用户密钥，详细过程如下。

(1)选取时间戳T，计算：

$$h_1=H_1(rid\parallel T)$$

$$sk=\sum_{i=1}^{n}b_i\cdot x_i(\mathrm{mod}\ q)$$

$$pse1=\sum_{i=1}^{n}b_i\cdot y_i(\mathrm{mod}\ q)$$

$$h_2=H_2(spk\cdot sk)$$

其中，b_i为h_1中的第i位比特值。

(2)计算$pse2=h_2\oplus rid$。

(3)获取$pse=\{pse1\parallel pse2\parallel T\}$，计算$h_3=H_3(pse)$。

(4)计算$\tau=sk+ssk\cdot h_3(\mathrm{mod}\ q)$。

最后，将$\{pse,\tau\}$返回给用户，其中pse作为用户公钥〈pseudonym〉，τ作为用户私钥。

本文采用P2PKHCA方案作为参考的可执行脚本方案，主要原因是其涉及对用户真实身份的追踪，方便密钥更新的身份验证和新旧密钥的关联验证，另外，相较于目前比特币系统中应用最广泛的P2PKH方案，P2PKHCA方案在算法性能方面更优。

3.3 组合公钥算法

组合公钥的密钥生成算法最早是由Zhang等[22]提出的，后来Li等[23]提出该算法存在密钥隐私的安全漏洞，目前Li等[6]又将其引入比特币密钥生成算法中，旨在利用椭圆曲线中点加运算优于点乘运算的思想来改进目前通用的P2PKH脚本方案，同时实现身份追踪，以及解决原组合公钥算法存在的安全漏洞。Zhang等[22]提出的组合公钥算法的密钥生成过程如下：

(1)系统初始化。给定阶数为q的加法循环群G，其中P是G的生成元。首先，统一选取n个秘密元素$x_i\in Z_q^*$($i=1,2,\cdots,n$)；然后，将其构成系统主组合私钥$X_{cpk}=(x_1,x_2,\cdots,x_n)$，从而得到系统主组合公钥$Y_{cpk}=(y_1,y_2,\cdots,y_n)$，其中$y_i=x_i\cdot P,i\in[1,n]$；最后，选取哈希函数$H_1:\{0,1\}^*\rightarrow\{0,1\}^n$并公开。

(2)会话密钥生成。首先，给定某个用户身份$ID\in\{0,1\}^*$；然后，计算用户私钥$x_{ID}=\sum_{i=1}^{n}h_i\cdot x_i(\mathrm{mod}\ q)$；最后，计算用户公钥$y_{ID}=\sum_{i=1}^{n}h_i\cdot y_i(\mathrm{mod}\ q)=x_{ID}\cdot P$，其中$h_i$为$H(ID)$的第$i$位比特值。

4 比特币系统的密钥更新机制

4.1 密钥更新机制的概述

本文中比特币系统的密钥更新机制(其架构见图3)由用户节点UN、身份管理者IM、比特币系统BS和防篡改设备TPD 4个实体组成。UN指比特币系统中所有用户节点；IM具有对比特币系统中节点的身份认证和对防篡改设备TPD进行预部署等功能；TPD具有密钥更新、对旧密钥签名以完成比特币向新密钥地址转账以及销毁旧密钥等功能(该实体一经部署，其内部结构流程将不可更改)；BS是部署其他实体的环境，同时其具有收集待更新用户密钥的功能。

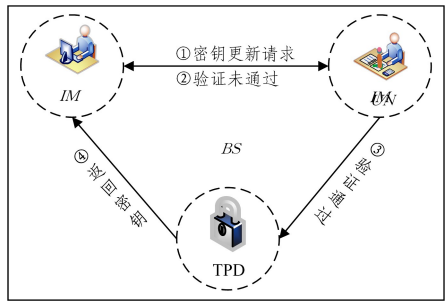

图3 比特币系统密钥更新机制

Fig. 3 Key update mechanism in Bitcoin system

本文提出的比特币密钥更新机制主要由两部分组成，分别是设计密钥更新脚本与构造应用方案。

(1)设计密钥更新脚本的目的是实现对用户密钥的定期更新，以完成对流失比特币的找回，从而维持比特币系统长久稳定地运行。首先，针对目前性能最优的P2PKHCA脚本方案存在的密钥隐私泄露安全问题，引入随机数替换时间戳的思想来提升其安全性，克服该方案因时间戳冗余而引起的单点故障问题；其次，为了实现对用户密钥的定期更新，引入密钥生命周期的概念，设计密钥更新脚本的密钥生命周期判断操作符，实现对密钥更新状态的精确判断；最后，为了执行对密钥的高效更新且切合比特币系统中单个用户多个密钥的情形，通过将密钥生成算法中部分公式保持初始计算状态的思想，减少冗余部分的重复运算以实现对比特币密钥的高效更新。

(2)构造应用方案的目的是将密钥更新脚本应用于实际场景，以实现对真实应用场景下流失比特币的找回，主要包括系统定期更新和用户请求更新两个场景。其中系统定期更新指在一定期限之后，系统会自动筛出满足密钥更新条件的用户对象并对其密钥进行更新；用户请求更新将分为两类情况

进行阐述，分别是用户密钥未遗失和已经遗失，具体思路将在4.2节进行阐述。

4.2 密钥更新机制的设计

本节首先对 P2PKHCA 中密钥生成算法进行改进；然后设计新的密钥更新操作符 OP_KEYUPDATE 和密钥生命周期判断操作符 OP_TSELECTION，并设计 P2PKHCA 的密钥更新脚本；最后分别构造两种密钥更新方案，即系统定期更新和用户请求更新。

4.2.1 P2PKHCA 的密钥更新脚本

(1)P2PKHCA 中密钥生成算法的改进

根据 Li 等[23]对 Zhang 等[22]提出的组合密码漏洞分析思想来分析 P2PKHCA 中密钥生成算法，能够了解到该算法存在元素冗余，从而造成其本身也存在密钥隐私泄露漏洞，因此本节将对 P2PKHCA 中密钥生成算法进行改进，具体分析和改进思想如下。

结合 Li 等[23]提出的漏洞分析思想，我们得出其密钥生成算法中 h_1 部分的时间戳 T 与用户公钥 pse 部分的时间戳 T 冗余，因此该算法存在严重的安全性漏洞。如果 UN 获得 pse 部分的 T 和身份 rid，则能算出 h_1，因此 Li 等[6]并没有对 Zhang 等[22]提出的组合公钥算法中引入用户身份 ID 计算散列值 $H(ID)$ 部分进行改进。另外对于 IM，其拥有系统主私钥 ssk 和主组合私钥 X_{cpk}，P2PKHCA 脚本方案最初的构造思想是：即使 IM 拥有 ssk 和 X_{cpk}，IM 也不能知道关于用户私钥 τ 的任何信息，但是基于以下安全性分析过程可知，该系统模型存在两个明显的问题：

1)IM 的存在会造成单点故障攻击，即如果 IM 被攻破，整个系统的用户密钥隐私将被泄露，从而导致系统无法正常运行；

2)基于第一个问题可以得出 T 的冗余将引起 τ 被暴露的风险。

本节对 P2PKHCA 脚本方案中密钥生成算法部分的安全性分析如下。

首先，IM 能够根据其拥有的 T，rid 和 X_{cpk} 计算出组合会话私钥；其次，IM 根据 pse 计算散列值 h_3'；最后，根据 sk'，h_3' 和 ssk 计算 τ'，IM 则能够知道所有关于 UN 公私钥的信息。因此，如果 IM 被敌手攻破，则其拥有的主私钥 ssk 和主组合私钥 X_{cpk} 就会被泄露，然后基于如下的计算过程就会造成 UN 的 τ 泄露给敌手，从而使得整个比特币系统存在密钥隐私泄露的风险。其中安全性分析的具体计算过程（描述 IM 模拟计算 τ 的过程）如下：

$$h_1' = H_1(T \parallel rid)$$

$$sk' = \sum_{i=1}^{n} b_i \cdot x_i \pmod q$$

$$h_3' = H_3(pse)$$

$$\tau' = sk' + ssk \cdot h_3' \pmod q$$

基于以上的安全性分析，本文对原密钥生成算法的改进思想是：将 h_1 部分的时间戳替换为随机数，使得其不与 pse 部分的 T 产生冗余，即 $h_1 = H_1(l \parallel rid)$（其中 l 为随机数），避免 IM 存在单点故障问题，从而保证敌手就算攻破 IM，也不能根据通过它获得的 ssk 和 X_{cpk} 计算出 UN 的 τ，防止其地址中比特币被盗取；同时，为了方便对密钥进行更新时条件的判断，本文对原密钥生成算法中的 pse 部分进行了补充，即 $pse = \{pse1 \parallel pse2 \parallel T \parallel T_e\}$（其中 T_e 为密钥生命周期，用于判断密钥是否达到可更新的条件），并在系统初始化环节将 $H_3:\{0,1\}^* \times \{0,1\}^* \times \{0,1\}^* \rightarrow Z_q$ 修改为 $H_3:\{0,1\}^* \times \{0,1\}^* \times \{0,1\}^* \times \{0,1\}^* \rightarrow Z_q$。

(2)OP_TSELECTION 操作符的设计

密钥过期是本文实现密钥更新的基本条件，但是不包括用户在密钥未遗失的情况下进行请求密钥更新的场景。相比原密钥生成算法中 h_1 部分的 T 被冗余处理，其 pse 部分的 T 则被保留，目的是更加方便地进行密钥生成周期的判断。其次，对 pse 部分补充密钥生命周期 T_e，目的是便于设计密钥更新方案的判断条件。算法1描述了新的密钥生命周期判断操作符 OP_TSELECTION(整个密钥生命周期的判断方法可以由 BS 或 IM 调用执行)的设计细节，其主要功能是判断网络中节点密钥是否达到更新条件。

算法 1 OP_TSELECTION

输入：pse
输出：真或假
1. 开始
2. 开始过程(BS 或 IM 执行)：
3. 输入 pse
4. 从 pse 中获取 T 和 T_e
5. 选取实时时间戳 T′
6. if 检查((T′−T)≥T_e)为真 then
7. return 真
8. else
9. return 假
10. end if
11. 结束过程
12. 结束

操作符 OP_TSELECTION 的具体设计思路为：首先，从 pse 中获取密钥生成时间戳 T 和密钥生命周期 T_e；然后，选取实时时间戳 T'，将上述已获取元素代入 $(T'-T) \geqslant T_e$ 中进行判断；最后，如果结果为真，说明密钥过期且返回真值，否则未过期且返回假值。本文提出的密钥更新方案正是以操作符 OP_TSELECTION 的判断结果作为基础条件，具体的方案构造细节将在4.2.2节中阐述。

(3)OP_KEYUPDATE 操作符的设计

在 P2PKHCA 中引入密钥更新脚本的关键在于设计新的密钥更新操作符 OP_KEYUPDATE，本节将基于前文改进的密钥生成算法，设计安全可靠的密钥更新操作符 OP_KEYUPDATE，从而实现比特币系统中用户密钥的可更新。同时，为了保证密钥更新的耗能低于密钥生成的耗能，需要将密钥生成算法中部分参数保持最初计算状态并预载到 TPD 中，以减少这部分参数的重复计算，从而保证密钥更新算法的能源消耗低于密钥生成算法的能源消耗，因此 UN 更倾向于对密钥进行更新来获得新的密钥，而不是根据密钥生成算法来获得新的密钥。算法2描述了 OP_KEYUPDATE 操作符的设计细节，在执行之前 TPD 已被 IM 预载的内容有 $\{PIN, rid, ssk, X_{cpk}, T_e, sk\}$，其中 sk 和 T_e 为本文新增的部分，表示

在密钥更新中不需要重复计算参数 sk 和 T_e。因此相比密钥生成算法，密钥更新算法将减少对 h_1，h_2，$pse2$，$pse1$ 和 sk 重复计算而产生的消耗。

操作符 OP_KEYUPDATE 的具体设计思路为：首先，由 UN，BS 或 IM 执行过程 1，主要是将满足密钥更新条件的 UN 信息（PIN，rid 和 pse）输入并等待 TPD 回应；然后，由 TPD 执行过程 2，该过程为根据 UN，BS 或 IM 预载到 TPD 的 UN 信息判定其真实性和正确性，如果匹配验证通过，则提取 pse 中的元素 $pse1$ 和 $pse2$，并选取实时时间戳 T'（为本文密钥更新的关键所在，即通过时间戳的更新实现密钥的更新）和提前预载的 T_e，从而完成对 pse' 的计算；接着，对 pse' 进行 H_3 哈希运算得到 h_3'，并将其与 TPD 中提前预载的 sk 和 ssk 一同代入公式 $\tau'=sk+ssk\cdot h_3' \pmod q$ 计算得到用户新私钥 τ'；最后，TPD 将 $\{pse',\tau'\}$ 返回给 UN，并由 UN 执行过程 3 完成密钥 $\{pse',\tau'\}$ 的获取与存储。

算法 2 OP_KEYUPDATE

```
输入：PIN，rid，pse
输出：更新后的密钥对{pse′，τ′}
1. 开始
2. 开始过程 1(UN，BS 或 IM 执行)：
3.    输入 PIN，rid，pse
4.    等待 TPD 响应
5. 结束过程 1
6. 开始过程 2(TPD 执行)：
7.    if 检查(rid)为真 then
8.       if 检查(PIN)为真 then
9.          从 pse 中获取 pse1 和 pse2
10.         选取实时时间戳 T′
11.         计算 pse′={pse1 ‖ pse2 ‖ T′ ‖ Te}
12.         计算 h3′=H3(pse′)
13.         计算 τ′=sk+ssk · h′3(mod q)
14.         返回{pse′，τ′}给 UN
15.      end if
16. end if
17. 结束过程 2
18. 开始过程 3(UN 执行)：
19.   UN 从{pse′，τ′}中获取 pse′作为公钥〈pseudonym〉′
20.   将 τ′保存作为私钥
21. 结束过程 3
22. 结束
```

(4)密钥更新脚本

根据前文提出的新操作符 OP_TSELECTION 和 OP_KEYUPDATE 设计本文在 P2PKHCA 脚本方案中引入的新的密钥更新脚本：

脚本 3 〈pseudonym〉 OP_TRACEID OP_SWAP OP_TSELECTION OP_KEYUPDATE。

脚本 3 的具体设计思想为：首先，UN 或 BS 输入旧公钥〈pseudonym〉，并从中获取 $pse1$ 和 $pse2$；然后，使用 OP_TRACEID 得到 rid 完成用户身份的追踪验证；接着，使用 OP_TSELECTION 判断其密钥是否满足更新条件；最后，通过 OP_KEYUPDATE 完成 UN 旧密钥的更新，得到新的公私钥对 $\{pse',\tau'\}$，其中 pse' 为新的公钥〈pseudonym〉′。密钥更新脚本的堆栈执行状态如图 4 所示。

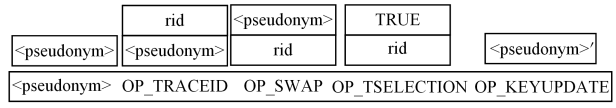

图 4 密钥更新脚本执行状态图

Fig. 4 Execution stack-state diagram of the key update script

4.2.2 密钥更新方案

本节将结合比特币密钥更新的系统机制架构（见图 3）和 4.2.1 节提出的密钥更新脚本构造两种特殊应用场景下的密钥更新方案，分别为系统定期更新方案和用户请求更新方案，主要目的是通过密钥更新实现遗失密钥地址情况下的比特币找回。

(1)系统定期更新方案

系统定期更新方案的构造思想是随着比特币版本更新而进行 UN 密钥更新，即如果比特币版本需要升级，则系统自动进行 UN 过期密钥的收集更新。算法 3 描述了系统定期更新的具体构造细节，其中完成了 UN 旧密钥地址到其新密钥地址的比特币交易，从而保证 UN 遗失密钥后能够随着比特币系统版本升级找回遗失密钥地址情况下的比特币。该方案的具体构造过程如下：

步骤 1 BS 调用算法 2 对系统中所有已知且未销毁的 UN 公钥集 $pses=\{pse\_1,pse\_2,\cdots,pse\_n\}$ 判断其密钥是否达到可更新的条件，其中 pse_i 为节点 UN_i 的公钥 $pse(i=1,2,\cdots,n)$。如果节点密钥过期，则将 $pse\_i(i=1,2,\cdots,n)$ 安全传输到 IM 并等待其响应以进行身份判断；否则忽略该密钥。

步骤 2 IM 接收到 UN_$i(i=1,2,\cdots,n)$ 的 pse 后，调用脚本 2 实现对 pse 的身份追踪得到 rid，然后将其与 pse 安全传输到 TPD 并等待其响应。

步骤 3 TPD 调用算法 1 对 UN 进行密钥更新得到其新密钥对 $\{pse',\tau'\}$，然后调用本文已改进的密钥生成算法，得到 UN_$i(i=1,2,\cdots,n)$ 更新前的旧密钥对 $\{pse,\tau\}$。

步骤 4 TPD 调用脚本 1 实现新旧密钥的交易验证，以将 $\{pse,\tau\}$ 地址的比特币转移到 $\{pse',\tau'\}$ 地址，然后将 $\{pse,\tau\}$ 销毁并向比特币网络发布销毁声明。

算法 3 系统定期更新

```
输入：pses={pse_1，pse_2，…，pse_n}
输出：新密钥对集
1. 开始
2. 开始过程 1(BS 执行)：
3.    收集公钥集 pses={pse_1，pse_2，…，pse_n}
4.    for 遍历(pse_i∈pses，i=1，2，…，n) then
5.       调用算法 2 判断其密钥生命周期并得到返回值 re1
6.       if 检查(re1)为真 then
7.          输入 pse_i
8.          等待 IM 响应
9.       end if
10.   end for
11. 结束过程 1
12. 开始过程 2(IM 执行)：
```

13. 调用脚本 2 实现对 pse_i 的身份追踪并得到 UN_i 的身份 rid
14. 输入 rid 和 pse_i
15. 等待 TPD 响应
16. 结束过程 2
17. 开始过程 3(TPD 执行)：
18. 调用算法 1 完成密钥更新得到新密钥对{pse′,τ′}
19. 调用改进的密钥生成算法得到旧密钥对{pse,τ}
20. 调用脚本 1 将{pse,τ}地址下的比特币转移到{pse′,τ′}地址
21. 销毁{pse,τ}
22. 结束过程 3
23. 结束

(2)用户请求更新方案

比特币用户的密钥更新请求方案分为以下两种情况。

1)用户密钥未遗失的情况。这类情况描述 UN 想要通过密钥更新获得新密钥从而完成新的交易，因为本文提出的密钥更新算法的时间消耗比密钥生成算法的时间消耗更少(通过 5.2 节的性能分析可知)。通过旧密钥生成时预载到 TPD 里面的参数(如 sk 等)，密钥更新时减少了该部分参数运算的冗余度，并且对 4.2.1 节密钥生成算法的改进(即对因 h_1 部分冗余的 T 所引起的 IM 单点故障问题进行改进)也解决了椭圆曲线中随机数的弱随机性问题(即比特币系统存在使用相同随机数计算密钥的情况)，因此相比通过密钥生成算法生成新密钥，用户更倾向于通过耗能更少且更安全的密钥更新算法生成新密钥。算法 4 描述了当用户密钥在未遗失情况下请求更新方案的具体构造细节。该方案的具体构造过程如下：

步骤 1　UN 调用 Li 等[6]的签名算法 ICAS 生成签名〈casig〉，然后将其与 PIN 和 pse 安全传输到 IM 并等待其响应。

步骤 2　IM 首先通过 pse 验证签名〈casig〉是否正确。然后从 pse 中获取时间戳 T、实时时间戳 T' 和密钥生命周期 T_e。最后调用算法 2 判断其密钥生命周期是否满足可更新的条件，如果正确且满足，说明 UN 密钥处于未遗失且未过期的情况，则执行步骤 3；如果正确但不满足，说明 UN 密钥过期但未遗失，则执行步骤 4；如果不正确，则忽略该密钥。

步骤 3　TPD 调用算法 1 完成旧密钥对$\{pse,\tau\}$的更新，得到新密钥对$\{pse',\tau'\}$，因为通过步骤 2 的判断了解到 UN 密钥并未过期，所以只需完成密钥更新并返回$\{pse',\tau'\}$给 UN，并不需要进行遗失密钥下新旧地址的比特币转移。

步骤 4　TPD 调用算法 1 完成旧密钥对$\{pse,\tau\}$的更新，得到新密钥对$\{pse',\tau'\}$；然后调用本文已改进的密钥生成算法得到 UN 更新前的$\{pse,\tau\}$；接着调用脚本 1 实现新旧密钥的交易验证，以将$\{pse,\tau\}$地址的比特币转移到$\{pse',\tau'\}$地址；最后将$\{pse,\tau\}$销毁并向比特币网络发布销毁声明。其中，虽然用户密钥未遗失但是已过期，所以需要由 TPD 执行$\{pse,\tau\}$地址的比特币转移。

算法 4　用户请求更新(密钥未遗失的情况)

输入：PIN，pse 和〈casig〉
输出：新密钥对{pse′,τ′}
1. 开始
2. 开始过程 1(UN 执行)：
3. 调用签名算法 ICAS 生成签名〈casig〉
4. 输入 PIN，pse 和〈casig〉
5. 等待 IM 响应
6. 结束过程 1
7. 开始过程 2(IM 执行)：
8. 通过 pse 调用验签算法验证〈casig〉得到返回值 re2
9. 调用算法 2 判断其密钥生命周期并得到返回值 re3
10. if 检查(re2) 为真 then
11. 输入 PIN，pse，re3
12. 等待 TPD 响应
13. end if
14. 结束过程 2
15. 开始过程 3(TPD 执行)：
16. if 检查(re3) 为假 then
17. 调用算法 1 完成密钥更新得到新密钥对{pse′,τ′}
18. else
19. 调用算法 1 完成密钥更新得到新密钥对{pse′,τ′}
20. 调用改进的密钥生成算法得到旧密钥对{pse,τ}
21. 调用脚本 1 将{pse,τ}地址下的比特币转移到{pse′,τ′}地址
22. 销毁{pse,τ}
23. end if
24. 结束过程 3
25. 结束

2)用户密钥已遗失的情况：这类情况是 UN 在遗失密钥后想恢复遗失密钥地址下的比特币，以防止自己的资产损失。因此需要 UN 向 IM 申请旧密钥更新，通过 TPD 找回旧密钥地址下的比特币。算法 5 描述了当 UN 在遗失密钥的前提下发出对旧密钥更新请求的构造细节，该方案的具体构造过程如下：

步骤 1　UN 将 PIN 和 pse 安全传输到 IM 并等待其响应。

步骤 2　IM 通过输入值(即未输入 UN 的签名)判断 UN 处于密钥已遗失并请求更新旧密钥的情况，因此调用脚本 2 实现对用户身份的验证得到身份 rid；然后调用算法 2 完成对 UN 密钥生命周期状态的判断；如果验证通过，则将 rid 安全传输到 TPD 并等待其响应。

步骤 3　TPD 首先调用算法 1 完成旧密钥对$\{pse,\tau\}$的更新，得到新密钥$\{pse',\tau'\}$；然后调用本文已改进的密钥生成算法得到旧密钥对$\{pse,\tau\}$；接着调用脚本 1 实现新旧密钥的交易验证，以将$\{pse,\tau\}$地址的比特币转移到$\{pse',\tau'\}$地址；最后将$\{pse,\tau\}$销毁并向比特币网络发布销毁声明。

算法 5　用户请求更新(密钥已遗失的情况)

输入：PIN，pse
输出：新密钥对{pse′,τ′}
1. 开始
2. 开始过程 1(UN 执行)：
3. 输入 PIN，pse
4. 等待 IM 响应
5. 结束过程 1
6. 开始过程 2(IM 执行)：
7. 调用脚本 2 完成对 UN 的身份确认得到 rid
8. 调用算法 2 完成密钥周期判断得到返回值 re4
9. if 检查(re4)为真 then
10. 输入 rid

11. 等待 TPD 响应
12. end if
13. 结束过程 2
14. 开始过程 3(TPD 执行):
15. 调用算法 1 完成密钥更新得到新密钥对{pse′,τ′}
16. 调用改进的密钥生成算法得到旧密钥对{pse,τ}
17. 调用脚本 1 将{pse,τ}地址下的比特币转移到{pse′,τ′}地址
18. 销毁{pse,τ}
19. 结束过程 3
20. 结束

5 安全性分析和性能分析

5.1 安全性分析

P2PKHCA 脚本方案的安全性分析的重点在于其签名算法 ICAS 的安全性。类似地,本文提出的密钥更新机制的安全性分析主要关注密钥更新操作符 OP_KEYUPDATE 和改进密钥生成算法的安全性,其分析思路主要是基于椭圆曲线离散对数难解性问题(ECDLP)和碰撞阻力阐述该密钥更新机制能够抵抗各种不同攻击方法,以评估其安全性。

ECDLP:给定椭圆曲线里面的两个随机点 P 和 Q,计算满足等式 $Q=P\cdot x$ 的 x 是困难的。

碰撞阻力:如果无法找到两个值 x 和 $y(x\neq y)$ 使得 $H(x)=H(y)$,则称哈希函数 H 具有碰撞阻力。

(1)前向安全和后向安全

前向安全和后向安全的概念最早是由 Anderson[24] 提出,目的是防止因密钥泄露而给密码系统带来安全威胁。在密码系统中前向安全或后向安全的含义是:某个时间段的密钥泄露并不会影响之前或之后时间段的密码体制安全性。而前向或后向安全密码技术之所以能够保证密钥泄露后密码体制的前向或后向安全性,主要是由于密码系统引入了密钥更新的概念,因为密钥更新算法采用的是单向函数,使得某个时间段的密钥泄露也仅仅只能获取该时间段往后或往前的密钥,并不能恢复该时间段往前或往后的密码,从而保证了密码系统的前向或后向安全性。

根据以上定义可知,本文的密钥系统也具有前向安全性,其正是通过密钥更新算法的单向性来保证的。如果第 i 个时间段的密钥对$\{pse_i,\tau_i\}$被泄露,那么想要获取第 $i-1$ 个时间段的密钥对$\{pse_{i-1},\tau_{i-1}\}$是困难的。具体证明过程为:如果敌手 adv 想要获取第 $i-1$ 个时间段的密钥对$\{pse_{i-1},\tau_{i-1}\}$,那么系统中固定的 sk 和 ssk 是计算 τ_{i-1}的关键因素。在基于 IM 不存在单点突破问题的前提下,虽然敌手 adv 能够通过 pse_i 计算出第 i 个时间段的$h_3'=H_3(pse_i)$,但是基于 ECDLP 的难解性,根据等式 $\tau_i=sk+ssk\cdot h_3^i(\mathrm{mod}\ q)$计算出 sk 和 ssk 实际上是困难的,因此能够确定本文的密码算法具有前向安全性。

类似地,本文的密钥系统也具有后向安全性。如果第 i 个时间段的密钥对$\{pse_i,\tau_i\}$被泄露,那么基于 ECDLP 的难解性,想要获取第 $i+1$ 个时间段的密钥对$\{pse_{i+1},\tau_{i+1}\}$也是困难的。

(2)抗单点故障攻击

根据 4.2.1 节对原密钥生成算法的改进,本节将基于原密钥生成算法的安全性分析思路(即阐述其容易遭受单点故障攻击)来分析 4.2.1 节中改进的密钥生成算法是抗单点故障攻击的,即假设虽然 IM 被 adv 攻破,但用户密钥仍具有隐私性。具体的分析过程如下:

1)假设 adv 攻破了 IM,那么其将获得 IM 持有的所有私有参数,包括 ssk、X_{cpk}、UN 的 rid 和 PIN,再通过 BS 能够获得 H_1,$pse=\{pse1\parallel pse2\parallel T\parallel T_e\}$,$H_2$,$H_3$ 和 P 等公开参数。

2)虽然 adv 拥有 IM 所有的私有参数,但是不能够再利用密钥生成算法中 $h_1=H_1(l\parallel rid)$和 pse 的 T 冗余来计算 h_1,因此不能通过持有的 X_{cpk} 计算出临时会话私钥 sk,进而不能够计算出 UN 的 $\tau=sk+ssk\cdot h_3(\mathrm{mod}\ q)$(因为其中 sk 是未知的),最终保证了用户私钥的隐秘性。另外,虽然 $pse1=sk\cdot P$,但是基于 ECDLP 的难解性,想要通过已知的 $pse1$ 和 P 计算出 sk 的值是困难的。

3)最终 adv 可以通过计算得到的数值仅有 $h_2=H_2(pse1\cdot ssk)$和 $h_3=H_3(pse)$,同时根据上一步的分析,因为无法获取 sk 的值,所以不可能计算出 UN 的私钥 τ。因此 UN 的密钥隐私得到了保护,证明本文提出的改进密钥更新算法能够抵抗单点故障攻击。

(3)抗弱随机性

弱随机性问题是最初的比特币系统版本存在的问题,主要是椭圆曲线中随机数的弱随机性问题,原因是虽然随机数的选择范围很广,但不可避免地会存在使用相同随机数计算用户密钥的情况,因此它也是比特币系统的漏洞之一。该安全问题在比特币社区中已经广为人知,学者们积极响应并在 2013 年应用 RFC6979[25] 的更新进行解决。但是 Wang 等[26] 又指出比特币系统中仍存在弱随机性问题,所以在关于随机数应用的部分,如果引入了密钥更新这个概念,则需要减少关于随机数的使用频率,这就是本文在 OP_KEYUPDATE 操作符中要保证系统私钥 ssk、主组合私钥 X_{cpk} 和 h_1 中的 l 保持初始计算状态的原因,另外也是为了减少密钥更新时不必更新部分重复计算的能源消耗。基于 IM 不存在单点故障的问题和基于哈希函数特性中碰撞阻力的描述,ssk、X_{cpk} 和 h_1 中 l 等随机数的最初值状态维持更能够符合本文密钥更新机制的设计。基于以上分析,我们可以认定,本文的密钥更新机制能够抵抗椭圆曲线中随机数的弱随机性问题。

(4)抗组合公钥漏洞

组合公钥漏洞最初是由 Li 等[23] 针对基于组合公钥的身份密码算法提出,其漏洞分析思想为:首先通过发送 n 个用户身份 ID 给密钥生成中心,产生 n 个相对应的密钥对$\{pk,sk\}$,然后通过散列函数 H 与 n 个 ID 计算出 n 个对应$\{pk,sk\}$的散列值 h,最后通过计算公式 $sk=\sum_{i=1}^{n}b_i\cdot x_i(\mathrm{mod}\ q)$(其中 b_i 为 h 的第 i 位的比特值,x_i 为随机选取的主组合私钥)、h 和获取的私钥 sk 建立一个 n 元方程,从而形成 n 个 n 元方程,通过$n\times n$ 的矩阵运算得出所有的 $x_i(i=1,2,\cdots,n)$值,因此得出 Zhang 等[22] 提出的基于组合公钥的身份密码算法不能保护 UN 密钥隐私的结论。但是本文提出的改进密钥生成算法使得 adv 不再能够通过发送多个身份 rid 建立 n 元方程组

求得系统主组合私钥 X_{cpk}，从而防止 τ 的泄露，因此本文改进的密钥生成算法能够抵抗 Li 等[23]提出的组合密钥陷阱。具体的分析过程如下：

1)假设 adv 发送 n 个 rid 给 TPD 获取到 n 个密钥对 $\{pse,\tau\}$；

2)adv 由于不知道改进密钥生成算法中 h_1 中 l 的任何信息，因此无法计算 h_1 的值，更加无法计算 sk 的值；

3)基于前两步的分析，可以知道 adv 不能建立关于 X_{cpk} 的 n 元方程组，从而不能获得关于 X_{cpk} 的任何信息。

因此本文提出的改进密钥生成算法是抗组合公钥漏洞的。

5.2 性能分析

本节主要对文中提及的各种算法进行时间消耗评估，包括基于平均时间消耗对改进的密钥生成算法进行性能评估并与原密钥生成算法进行对比，以及方案的复杂度分析与对比。本实验采取的语言是 python 语言，运行环境为 windows 10 的 64 位操作系统，并部署算法于 python 3.7。其中椭圆曲线采用 curve type NIST256p[27]，选取 SHA-256 作为安全散列函数。本节将分别对各种算法测试 3000 次，以获取每种算法的平均时间，准确评估每种算法的时间消耗，从而得到合理的性能结果。

实验结果中的二元组(如：(3 000，80 677.666 908 261 86))，其第一个元素表示算法执行的次数，第二个元素表示该算法在总次数内执行的总时间。

图 5 给出了原密钥生成算法和改进密钥生成算法之间的时间消耗对比，即分别对本文 4.2.1 节(1)部分的改进密钥生成算法和文献[6]的密钥生成算法进行编程实现，并执行 3000 次得到各自的总时间消耗，从而绘制出该时间消耗斜率对比图。从图 5 可以看出，本文改进的密钥生成方案的平均时间消耗约为 26.89 ms，比原密钥生成方案的 25.56 ms 慢了约 1.33 ms。

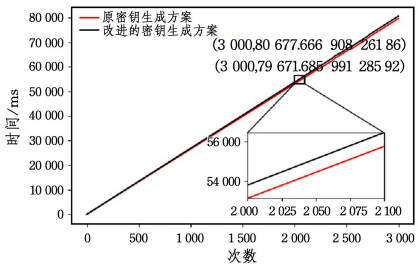

图 5 原密钥生成算法与改进密钥生成算法的时间消耗对比

Fig. 5 Comparison of time consumption between original key generation algorithm and improved key generation algorithm

图 6 给出了 OP_KEYUPDATE 和改进密钥生成算法的时间消耗对比，即分别对本文 4.2.1 节(3)部分的算法 2 和 4.2.1 节(1)部分的改进密钥生成算法进行编程实现，并执行 3000 次得到各自的总时间消耗，从而绘制出该时间消耗斜率对比图。从图 6 可以看出，OP_KEYUPDATE 的平均时间消耗约为 0.27 ms，相比改进的密钥生成算法，OP_KEYUPDATE 的时间消耗约是其时间消耗的 1/10。

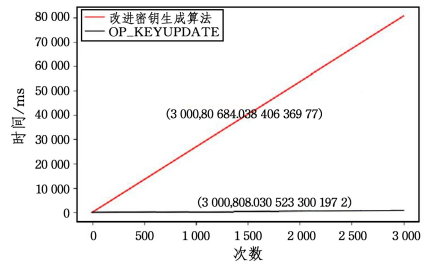

图 6 密钥生成算法与 OP_KEYUPDATE 的时间消耗对比

Fig. 6 Comparison of time consumption of key generation algorithm and OP_KEYUPDATE

图 7 给出了 OP_TSELECTION 的时间消耗，即对本文 4.2.1 节(2)部分的算法 1 进行编程实现，并执行 3000 次得到其总的时间消耗，从而绘制出该时间消耗斜率图。从图 7 可以看出，OP_TSELECTION 的平均时间消耗约为 0.0026 ms，就整体而言可以忽略不计。

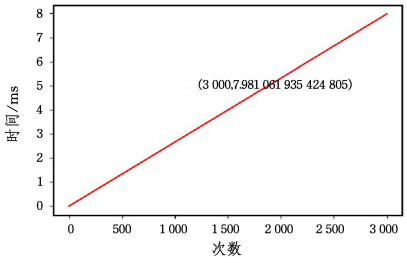

图 7 OP_TSELECTION 的时间消耗

Fig. 7 Time consumption of OP_TSELECTION

图 8 给出了文中涉及的各种算法的时间消耗对比，即分别计算或引用图 5、图 6、图 7 和文献[6]中 OP_TRACIED (OP_TRACIED 的时间消耗约为 24.90 ms)及 OP_SWAP (OP_SWAP 的时间消耗可忽略不计)的平均时间消耗，从而绘制出该平均时间消耗柱状图。从图 8 可以看出，密钥更新脚本和原密钥生成算法之间的时间差大于原密钥生成算法和改进密钥生成算法之间的时间差，因此对于重新生成密钥，用户更偏向于时间消耗更少的密钥更新，虽然改进的密钥生成算法略慢于原密钥生成算法，但整体而言，引入新元素密钥生命周期 T_e(其大小为 8 字节)而增加的 0.34 ms 时间消耗是可以接受的，因为执行两次原密钥生成算法的时间大于执行一次改进密钥生成算法和一次密钥更新脚本的时间。

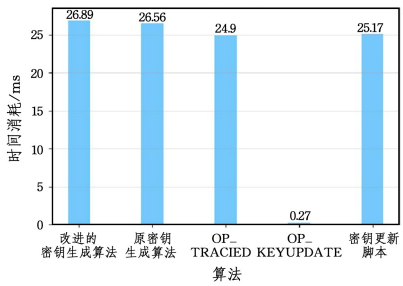

图 8 各种算法的时间消耗柱状图

Fig. 8 Histogram of time consumption of each algorithm

通过文献[6]和上述实验可知，OP_HASH160 的时间消耗约为 0.41 ms，〈casig〉的生成时间约为 24.33 ms，OP_TRACIED 的时间消耗约为 24.90 ms，OP_DUP，OP_EQUAL 和

OP_DROP 的时间消耗可忽略不计，OP_SWAP 的时间消耗几乎为 0。因此，脚本 1 的时间消耗约为 106.92ms，脚本 2 的时间消耗约为 24.90ms。图 9 给出了 4.2.2 节中系统自动更新方案在不同数量节点时的时间消耗情况。从图 9 可以看出，节点数量越多，时间消耗的增长速率就越大，但是对单个节点完成遗失密钥找回操作的平均时间消耗约为 0.16s，这相较于比特币系统中交易确认的时间消耗是可以接受的。

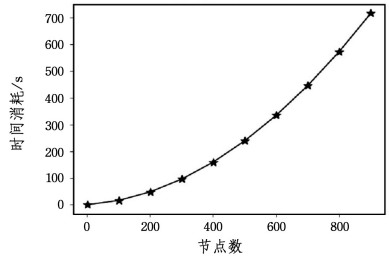

图 9 系统自动更新方案的时间消耗

Fig. 9 Time consumption of automatic system update scheme

图 10 给出了 4.2.2 节中用户请求更新方案在不同数量节点时的时间消耗情况。从图 10 可以看出，算法 4 比算法 5 的时间消耗更多(算法 4 的单个节点密钥找回的平均时间消耗约为 0.24s，算法 5 的平均时间消耗约为 0.16s)，增长速率也更大，因此在用户密钥过期的情况下，密钥的丢失可以提高整个系统的运行效率。

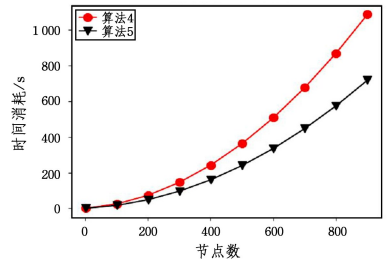

图 10 用户请求更新方案的时间消耗

Fig. 10 Time consumption of user requirement update scheme

表 2 列出了 4.2.2 节中提出的密钥更新方案与其他相关密钥更新方案的复杂度分析对比。其中，本文假设有 n 个节点用户，k 表示 h_1 的比特位或者组合密钥元组中的密钥个数；文献[14]中，n 表示边缘节点的数量，m 为上传文件的块数，l 为文件块的长度；文献[15]中，n 表示移动多媒体传感器节点的数量。4.2.2 节的密钥更新方案的复杂度分析过程如下。

(1)通信复杂度

1)算法 3 中 BS 将满足条件的公钥发送给 IM 所产生的最大通信复杂度为 $O(n)$，IM 将其传输到 TPD 所产生的最大通信复杂度也为 $O(n)$。因此算法 3 产生的通信复杂度为 $O(2n)$。

2)算法 4 和算法 5 中 UN 传输消息到 IM，再传输到 TPD，所产生的最大通信复杂度都为 $O(1)$，因此算法 4 和算法 5 产生的通信复杂度都为 $O(2)$。

综上可得，4.2.2 节密钥更新方案的通信复杂度为 $O(n)$。

(2)计算复杂度：

1)算法 3 中 BS 执行阶段的计算复杂度为 $O(n)$，IM 执行阶段的计算复杂度为 $O(n)$，TPD 对 n 个公钥执行更新操作产生的计算复杂度为 $O(n\times k)$。

2)算法 4 和算法 5 中针对单个用户执行的验签、条件判断操作产生的计算复杂度为 $O(1)$，执行密钥更新操作的计算复杂度为 $O(k)$，因此算法 4 和算法 5 产生的计算复杂度为 $O(k)$。

综上可得，4.2.2 节密钥更新方案的计算复杂度为 $O(n\times k)$。

从表 2 可以看出，在假设 k，m，l 大致相等的前提下，本文提出的密钥更新方案在复杂度方面具有较大的优势。

表 2 密钥更新方案的复杂度分析

Table 2 Complexity analysis of key update schemes

指令或值名称	通信复杂度	计算复杂度
文献[14]	$O(3n)$	$O(n\times m\times l)$
文献[15]	$O(2n)$	$O(n)$
本文	O(2n)	$O(n\times k)$

结束语 本文的主要目的是通过实现用户密钥的定期更新来完成对用户比特币资产的保护，以防止因用户遗忘密钥或突然死亡等状况而导致其密钥地址下比特币永久遗失的现象发生。首先，针对比特币系统中存在的密钥更新难问题，通过优化目前性能最优且具有可追踪性的 P2PKHCA 脚本方案构造密钥更新脚本；其次，通过结合两种实际应用场景完成比特币系统的密钥更新机制设计并实现流失比特币的找回。针对机制的效果而言，一方面，相比原 P2PKHCA 脚本方案，本文改进的 P2PKHCA 脚本方案更具安全性，虽然在密钥生成的时间消耗较长，但是对于该方案的整体密钥更新需求而言，增加的消耗时间是可以接受的；另一方面，对比分析其他相关密钥更新方案的复杂度发现，本文的密钥更新方案在复杂度方面具有较大的优势。最后，未来研究工作可能是构造区块链应用系统中通用的密钥管理方案，目前下一步工作是实现对密钥的安全托管，进而实现丢失密钥的可信恢复。

参 考 文 献

[1] NAKAMOTO S. Bitcoin: a peer-to-peer electronic cash system [OL]. (2008-10-31) [2020-10-30]. http://bitcoin.org/bitcoin.pdf.

[2] TSUNG-TING K, HYEONEUI K, LUCILA O M. Blockchain distributed ledger technologies for biomedical and health care applications[J]. the American Medical Informatics Association, 2017, 24(6): 1211-1220.

[3] CHUKWU E, GARG L. A systematic review of blockchain in healthcare: frameworks, prototypes, and implementations[J]. IEEE Access, 2020, 8: 21196-21214.

[4] TIAGO M, FERNÁNDEZ-CARAMÉS, PAULA F. A Review on the Use of Blockchain for the Internet of Things[J]. IEEE Access, 2018, 6: 32979-33001.

[5] OUIMET S. Bitcoin dominance rate hits 50% for first times [OL]. [2020-09-28]. https://www.coin-desk.com/bitcoin-dominance-rate-hits-50-for-first-time-in-2018/.

[6] LI L, LIU J Q, CHANG X L, et al. Toward conditionally anonymous Bitcoin transactions: A lightweight-script approach[J]. In-

formation Sciences,2020,509:290-303.
[7] ESKANDARI S,CLARK J,BARRERA D,et al. A First Look at the Usability of Bitcoin Key Management[C]// Workshop on Usable Security. 2015.
[8] ITTAY E,ADEM E G,EMIN G S,et al. Bitcoin-NG:A Scalable Blockchain Protocol[C]// Symposium on Networked Systems Design and Implementation. 2016:45-59.
[9] MÖSER M,EYAL I,SIRER E G. Bitcoin Covenants[C]// International Conference on Financial Cryptography & Data Security. 2016:126-141.
[10] O'CONNOR R ,PIEKARSKA M. Enhancing Bitcoin Transactions with Covenants[C]// International Conference on Financial Cryptography & Data Security. 2017:191-198.
[11] POULAMI D,LISA E,TOMMASO F,et al. FastKitten:Practical Smart Contracts on Bitcoin[C]// USENIX Security Symposium. 2019:801-818.
[12] YU H,ZHANG Z Y,LIU J W. Research on Scaling Technology of Bitcoin Blockchain[J]. Journal of Computer Research and Development,2017,54(10):2390-2403.
[13] BRENGEL M,ROSSOW C. Identifying Key Leakage of Bitcoin Users[C]// International Symposium on Recent Advances in Intrusion Detection. 2018:623-643.
[14] LI J,LI T,LIU Z,et al. Secure Deduplication System with Active Key Update and Its Application in IoT[J]. ACM Transactions on Intelligent Systems and Technology,2019,10(6):1-21.
[15] HONG H,SUN Z. Achieving secure data access control and efficient key updating in mobile multimedia sensor networks[J]. Multimedia Tools and Applications,2017,77(4):4477-4490.
[16] TIAN Y,WANG Z,XIONG J,et al. A Blockchain-Based Secure Key Management Scheme with Trustworthiness in DWSNs[J]. IEEE Transactions on Industrial Informatics,2020,16(9):6193-6202.
[17] LI Y,YU Y,YANG B,et al. Privacy preserving cloud data auditing with efficient key update[J]. Future Generation Computer Systems,2018,78(PT. 2):789-798.
[18] ATHMANI S,BILAMI A,BOUBICHE D E. EDAK:An Efficient Dynamic Authentication and Key Management Mechanism for heterogeneous WSNs[J]. Future Generation Computer Systems,2019,92:789-799.
[19] NICOLAS T C,REBEKAH M. Stealth Address and Key Management Techniques in Blockchain Systems[C]// International Conference on Information Systems Security & Privacy. 2017:559-566.
[20] STANISLAW J,HUGO K,JASON K R. Updatable Oblivious Key Management for Storage Systems[C]// ACM Conference on Computer and Communictions Secu-rity. 2019:379-393.
[21] ALBAKRI A,HARN L,MADDUMALA M. Polynomial-based Lightweight Key Management in a Permissioned Blockchain [C]// IEEE Conference on Communications and Network Security. 2019:1-9.
[22] ZHANG R,LIU J,HAN Z,et al. An IBE scheme using ECC combined public key[J]. Computers & Electrical Engineering,2010,36(6):1046-1054.
[23] LI X,QIAN H,ZHOU Y. Pitfalls in identity-based encryption using an elliptic curve combined public key[J]. Applied Mathematics Letters,2012,25(8):1111-1113.
[24] ANDERSON R. Two Remarks on Public-Key Cryptology [OL]. [2020-09-27]. http://www.cl.cam.ac.uk/user/rja14/.
[25] THOMAS P. Deterministic Usage of the Digital Signature Algorithm (DSA) and Elliptic Curve Digital Signature Algorithm (ECDSA)[J]. Request for Comments,2013,6979:1-79.
[26] WANG Z,YU H,ZHANG Z,et al. ECDSA weak randomness in Bitcoin[J]. Future Generation Computer Systems,2020,102:507-513.
[27] BROWN M,HANKERSON D,LÓPEZ J,et al. Software Implementation of the NIST Elliptic Curves Over Prime Fields[C]// Proceeding of the 2001 Conference on Topics in Crytology:The Cryptographer's Track at RSA. 2001:250-265.

XIANG A-xin, born in 1996, postgraduate. His main research interests include cryptography and blockchain technology.

GAO Hong-feng, born in 1975, associate professor. His main research interests include network and information security.