计算机研究与发展
Journal of Computer Research and Development
DOI:10.7544/issn1000-1239.2018.20180436
55(10):2199-2207, 2018

基于区块链的云数据删除验证协议

刘忆宁[1] 周元健[1] 蓝如师[1] 唐春明[2]
[1](桂林电子科技大学计算机与信息安全学院 广西桂林 541004)
[2](广州大学数学与信息科学学院 广州 510006)
(ynliu@guet.edu.cn)

Blockchain-Based Verification Scheme for Deletion Operation in Cloud

Liu Yining[1], Zhou Yuanjian[1], Lan Rushi[1], and Tang Chunming[2]
[1](*School of Computer and Information Security, Guilin University of Electronic Technology, Guilin, Guangxi* 541004)
[2](*School of Mathematics and Information Science, Guangzhou University, Guangzhou* 510006)

Abstract Nowadays, more and more users upload their data to the cloud server, since the cloud can provide the service for users any time and at any place. Therefore, the cloud service facilitates the data usability and reduces the cost. However, the information leakage accidents have been reported frequently over the world, that is to say the cloud server is not fully trusted, and the security issue in cloud service must be paid enough attention. For example, illegal user may want to access the cloud server, and perhaps the cloud server does not delete the data according the user's requirement. In order to address these concerns, a verification scheme for deletion operation in cloud is presented using block-chain technology, which can make the deletion operation more transparent. In our scheme, the user calls the smart contract to prove his identity to the cloud server, and creates the request transaction for data deletion; then the cloud server deletes the data and generates a block chain with the evidence (evidence chain). Even if the cloud server is dishonest, the user can still verify the data deletion result. Moreover, the proposed scheme is analyzed to really achieve the public verification of data without the third-party trusted organization, the impersonation attacks resistance, and the eavesdropping attacks resistance, etc.

Key words cloud server; public verification; blockchain; data deletion; smart contract

摘 要 云服务器可以为用户提供任何时间、任何地点的服务,并极大地降低用户成本,提高使用的便利性,如今越来越多的用户将自己的数据存储在云服务器.然而,关于云存储中的安全问题不时得到披露,影响到用户对云存储的信任,因此必须足够重视云存储及云服务中的安全问题.例如未经身份验证的用户不可以访问云服务器,云服务器不按用户的要求删除数据应该能被发现并惩罚.为了解决这些问题,

收稿日期:2018-06-10;**修回日期**:2018-08-03
基金项目:国家自然科学基金项目(61662016, 61772147);广东自然科学基金基础研究重大项目(2015A030308016);广东教育厅科研团队项目(2015KCXTD014);广州市教育局协同创新重大项目(1201610005);国家密码发展基金(MMJJ20170117);桂林电子科技大学优秀研究生论文培优计划(16YJPYSS14);桂林电子科技大学研究生创新项目(2018YJCX50)
This work was supported by the National Natural Science Foundation of China (61662016, 61772147); the Major Basic Research and Cultivation Project of Guangdong Province Natural Science Foundation (2015A030308016); Project of Ordinary University Innovation Team Construction of Guangdong Province (2015KCXTD014); Collaborative Innovation Major Projects of Bureau of Education of Guangzhou City (1201610005); the National Cryptography Development Fund (MMJJ20170117); GUET Excellent Graduate Thesis Program (16YJPYSS14); and the Innovation Project of GUET Graduate Education (2018YJCX50).
通信作者:唐春明(ctang@gzhu.edu.cn)

提出了一种基于区块链的云数据删除验证协议.首先,用户通过调用智能合约向云服务器证明自己的身份,并且创建数据删除的请求交易,然后云服务器删除数据并生成一条嵌有删除证据的区块链(证据链).无论云服务器是否恶意,用户都可以验证数据删除结果.安全性分析表明:提出的协议可以在没有第三方可信机构的情况下完成数据的公开验证,同时可以抵抗窃听攻击、假冒攻击等.

关键词 云服务器;公开验证;区块链;数据删除;智能合约

中图法分类号 TP309.2

随着互联网的发展,人们把大量的数据存放在云平台,实现任何时间、任何地点的便捷使用.云存储虽然存在巨大的好处,但也存在很多的安全隐患,比如云平台可能恶意删除文件和对用户进行欺骗.当用户需要删除数据时,会向云服务器发送删除命令,服务器执行删除操作.然而,云服务器可能并未执行删除操作.因此,有必要验证云服务器是否正确地执行了删除操作.为了解决删除操作中的数据验证问题,文献[1-10]提出了多种不同类别的数据验证方案,其中公开验证最为普遍.2014 年,Sookhak 等人[11]等人对远程验证方案做了归纳,将其分为以下 3 类:

1) PDP 方案.Ateniese 等人[1]提出一种同态认证器与 RSA 签名结合的数据占有(provable data possession, PDP)方案,该方案实现了数据完整性的公开验证.然而,PDP 是静态模型,不支持存储数据的动态操作.为了实现数据的动态性,Wang 等人[12]提出了一个动态数据完整性验证方案和 Wang 等人[13]提出了一个动态数据公开验证方案.文献[12]在分布式条件下解决了数据动态存储问题并支持 TPA 审证;文献[13]在 BLS 签名技术中引入 RSA 结构,因此加强了数据的动态操作性,但又引入了隐私泄露问题;2016 年,Liu 等人提出了改进方案[14],该方案不但可以抵御来自外部的伪造攻击,还支持动态审核和多批次审批,但是没有实现多用户的审批.因为不同参数构成的用户验证数据标签不同,所以第三方审计者(third-party auditor, TPA)不能使用不同的标签线性组合审计.

2) POR 方案.Juels 等人[7]提出可检索性证明(proofs of retrievability, POR)方案.POR 方法会产生一个特殊的数据块,并将其嵌入数据中以达到检查正确性的目的.POR 和第 1 类方法 PDP 一样,不适合动态数据的存储,动态数据会使特殊数据块丢失原来的位置.随后,Cashd 等人[4]提出了动态 POR.郑等人[8]在原有的 POR 模型上实现了公平性,提出了基于 2-3 层 Merkle 树结构的公平动态 POR 协议.

3) POW 方案.POW 被称为所有权验证方案,其主要贡献在于提高数据存储的效率.

上述研究通常把验证者的角色分为公开验证和私有验证 2 种:1)公开验证意味着数据拥有者授权其他验证者去验证自己的数据是否有效;2)私有验证与公开验证相反,它是数据拥有者直接验证数据,这种方式是高效的.

虽然到目前为止已经提出很多数据完整性验证协议,但考虑到在面向物联网的云存储环境中,大多数设备的计算及存储资源有限,不适用较为复杂的运算.因此,能够高效验证物联网设备对云存储数据的完整性是至关重要的.Wang 等人[15]提出一个基于轻量级算法的高效公开数据完整性验证方案.该协议使用随机掩码技术保护用户的隐私,第三方审计人员无法由审计信息恢复用户的数据.Yu 等人[16]提出了一种开放式云存储数据验证方案.用户不再需要从云中下载数据,直接使用云中的小型认证器进行认证.该方案有效降低了通信成本和技术成本,使用户可以有效更改验证标记.Zhang 等人发现文献[16]在密钥更新阶段存在用户私钥泄露的问题.因此,Zhang 等人改进了密钥更新机制,提出了一个新的数据验证方案[17],在该方案中,新用户用私钥对原始用户进行签名形成用户的新密钥防止攻击.接着,Zhang 等人发现自己的协议不能抵御撤销用户与恶意云服务器的合谋攻击.Zhang 等人改进了文献[17],提出一个新的方案[18].他们利用代理再签名算法生成的签名密钥可以有效防止合谋攻击,实现数据的安全传输.

本文基于区块链技术,设计了面向云平台的数据删除验证协议,尤其是降低了用户对云服务器的信任假设.当云服务器发送恶意操作时,任何人都可以根据协议中的区块链验证操作结果.主要创新点包括 3 个方面:

1) 创建了一个基于区块链的远程医疗数据验证系统.如果云服务器没用诚实地删除数据,协议可以为用户提供检查服务器恶意行为的功能.而且,协议不需要任何的可信第三方的支持.

2) 在实现数据公开验证的同时,协议具有通信和计算上的高效性.

3) 构建了专门的智能合约,以实现对用户的身份验证.

1 相关基础知识

本节介绍相关的密码学基础知识,包括基于属性的签名方案(attribute based signature, ABS),Merkle 二叉树的定义和结构以及区块链技术.

1.1 基于属性的签名(ABS)

在基于属性的签名方案中,签名者首先会声明签名对应一组特定的属性或特定的访问控制结构,然后验证者对此进行验证.这类方案具有良好的匿名性,可以细粒度划分身份特征,满足分布式网络系统的要求.因此,在区块链网络的构建中,使用 ABS 签名代替传统的 ECDSA 签名(用于比特币)以保护隐私和安全性.

本文采用 Li 等人[19]提出的 ABS 结构实现交易的签名.为了描述 ABS 结构,列举出重要的符号 $c_{\max}$ 和 A,$c_{\max}$ 表示声明谓词中单调跨度程序的最大宽度,$A=Z_p^*$ (p 是安全大素数)表示属性集合.具体步骤如下:

1) $ABS.Setup$. 给定 2 个 p 阶循环群 G_1 和 G_2,对应双线性对 $e:G_1\times G_2\rightarrow G$ 和散列函数 $H:\{0,1\}^*\rightarrow Z_p^*$;然后随机选择 $g\leftarrow G_1$,$h_i\leftarrow G_2$ ($i=1,2,\cdots,c_{\max}$),$x_0,x,y,z\leftarrow Z_p^*$,计算 $Z=g^z$,$X_0=h_0^{x_0}$,$X_j=h_j^x$,$Y_j=h_j^y$,其中 $j=1,2,\cdots,c_{\max}$;最后返回公共参数 $P=(G_1,G_2,H,g,(h_0,h_1,\cdots,h_{c_{\max}}))$,主密钥 $ASK=(x_0,x,y)$ 和公钥 $APK=(X_0,X_1,\cdots,X_{c_{\max}},Y_1,Y_2,\cdots,Y_{c_{\max}},Z)$.

2) $ABS.AttrGen$. 输入主密钥 ASK,属性集 $s\in A$ 和随机选择 $K_{\text{base}}\leftarrow G_1$,然后计算 $K_0\leftarrow K_{\text{base}}^{x_0^{-1}}$ 和 $K_u=K_{\text{base}}^{(x+yu)^{-1}}$ ($u\in A$). 最后,返回签名密钥 $SK_u=(K_{\text{base}},K_0,K_u\mid u\in A)$.

3) $ABS.Sign$. 输入 (APK,SK_u,m,γ),其中 m 是消息,γ 是一个单调布尔函数.输入成立的条件:$\gamma(s)=1$,意味着对应的属性嵌入在 s 中.然后返回一个签名 σ.

4) $ABS.Ver$. 输入 (APK,σ,m,γ),输出一个布尔值,如果布尔值是 $Ture$,则表示验证成功并接收 σ,否则程序中止.

1.2 Merkle 树

Merkle 树[20]是一种散列二叉树,是一种做快速归纳和效验大数据完整性的数据结构.本文以比特币中交易存储结构为例,如图 1 所示.A,B,C,D 四个构成 Merkle 树叶的交易.在比特币中,对交易分别计算 Hash 值后,将 Hash 值存储在对应的叶子节点.然后,相邻叶子节点的 Hash 值串联哈希后生成父节点.重复同样的操作直到只剩下一个顶部节点为止,即 Merkle 根.二叉树是偶数叶子节点,如果交易个数为奇数,将最后一个交易复制一份即可.最后,用公钥签名技术对根节点签名,并且通过验证特定叶节点到根节点路径上的所有兄弟节点来验证交易.

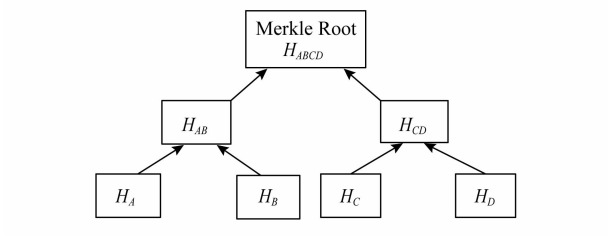

Fig. 1 Merkle tree structure

图 1 Merkle 树结构

1.3 区块链

区块链[21]源于比特币的底层技术,是一个去中心化、防篡改的分布式数据库,在没有中心集权机构的存在下,实现多方可信、价值对等的传输.在比特币系统中,如果有交易写入区块,交易数据必须由系统中的所有节点通过共识机制共同决定,并且系统内部的所有节点都无法修改.如果链上的一个区块中的交易被非法更改,其他节点会识别出非法节点,并删除这条链.区块链为每个参与者提供分散信任的账簿,每个用户或节点共同维护并存储这个完全相同的分类账簿,这确保了所有用户和相应区块链中的节点完全一致.在分类账中,交易和历史数据都会永久存在区块中,其持久性基于网络的持久性.要改变或者删除区块中的交易记录是非常困难的,除非攻击者可以篡改网络中 51% 的节点.此外,所有的系统规则都是公开和透明的,并且是经过所有节点同意的.系统中的每个节点中都存有当前网路区块链的副本.换句话说,每个节点都持有数据库的副本,这保证了数据库不能被单个节点单独操作.

目前,区块链分为 3 类:

1) 公有链.公有链缺少访问控制权限,任何人都可以随时进入系统中进行数据操作和发送交易.公有链去中心化程度高,数据也是公开的.比如比特币和以太坊.

2) 联盟链.联盟链是指由若干个机构共同参与管理的区块链,只允许系统内的机构进行读写和发送交易,共同记录交易.比如超级账本.

3) 私有链. 私有链是指写入权限集中在一个组织手里的区块链. 由于参与节点有限和可控,因此私有链交易速度快、隐私性好、交易成本低、不易被恶意攻击.

1.4 智能合约

1994 年,Szabo[22] 首次提出智能合约,定义为"执行合同条款的计算机协议". 一般智能合约可以理解为是一段部署在区块链上的程序代码. 每个智能合约都具有独特地址的数据槽,并通过事务或者交易来触发其数据管理功能.

在提出的协议中,使用智能合约来管理访问和产生交易. 具体描述如附录的算法 1 和算法 2.

2 系统模型和设计目标

2.1 系统模型

基于区块链的云数据删除验证系统结构如图 2 所示,其中包括云服务器、用户、区块链网络. 系统初始化前,系统会收集用户的属性并在区块链网络中布置智能合约. 如果用户发布交易就会触发智能合约对其身份验证. 如果用户是合法的,用户使用智能合约产生的属性密钥对交易签名和发布. 交易发布后,区块链网络中的"验证节点"会对其进行验证. 云服务器接收交易后调用设备,同时为调用的设备产生一个调用证据,并将其放入区块中形成一条区块链(证据链). 最后,云服务器将证据发送给用户,用户根据证据链验证数据删除结果是否合法. 下面我们对系统中的实体进行描述:

1) 用户. 用户可以在区块链网络中发布交易,并且可以验证云服务器实现删除操作的结果.

2) 区块链网络. 本文采用一种许可处理结构,这些许可节点分为"验证节点"(主要负责验证交易,简称 vdn)和"记账节点"(主要负责将验证的事务链接到区块链,简称 bkn)

3) 云服务器. 云服务器是存储用户数据的实体. 云服务器在区块链网络中收到用户的数据删除的请求交易后,生成删除证据并嵌入到区块链中. 最后,将证据发送给用户. 这个证据包含用户想要删除的数据请求,用户对数据请求交易的签名,服务器对删除交易的签名以及时间戳.

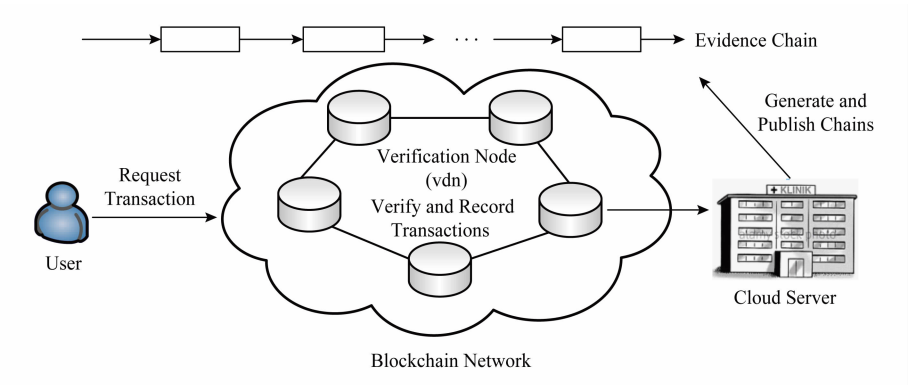

Fig. 2 Data deletion model

图 2 数据删除模型

2.2 攻击模型

本文假设云服务器是半可信的,它不会泄露我们的数据隐私,但可能不会按照协议执行删除数据的操作,并有可能返回错误结果去误导用户. 另外,本文假设区块链网络是公开的,攻击者容易获得公开信道上的传输数据. 因此,协议存在以下 3 种攻击.

1) 攻击者窃听和篡改传输在公开信道上的信息,并执行窃听攻击和重复攻击等.

2) 服务器收到用户的数据删除请求后,没有诚实按协议删除数据并返回错误的结果误导用户.

3) 在没有收到用户的数据删除情况下,云服务器为了经济利益随意删除数据.

2.3 安全目标

本文拟实现的安全目标包括:

1) 匿名性. 任何实体都不能通过云服务器产生的证据得到用户信息.

2) 防窃听攻击. 通过对公开信道的窃听,攻击者无法获得任何有用信息.

3) 防重复攻击. 过时的信息重新发送时,它将被丢弃.

4) 完整性. 用户可以检测半可信云服务器篡改数据的恶意行为.

5) 可追溯性. 如果服务器没有诚实的删除数据,用户可以通过调用证据追溯服务器的责任,并且服务器不能否定.

3 基于区块链的云数据验证协议

3.1 协议概述

本文基于区块链技术,设计了一个面向云数据删除验证协议.其中,云服务器被假设为半可信的,服务器可能不会诚实的删除数据.协议的主要过程如图3所示.用户进入系统调用智能合约进行身份验证,验证合法后,用户构建一个含有数据删除请求的交易并广播.云服务器确认交易后删除数据,并为用户产生可以对删除结果验证的证据.证据嵌入区块中形成一条可验证云服务器是否诚实删除数据的区块链(证据链).在证据链的帮助下,如果云服务器没有诚实删除数据,用户可以验证删除结果并且云服务器不能否认.

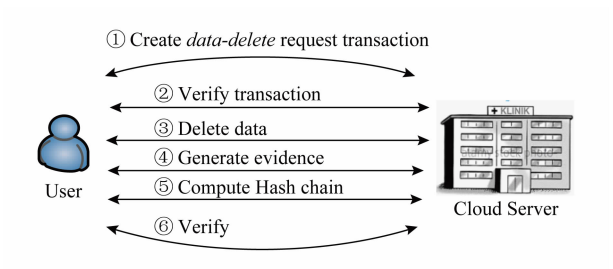

Fig. 3 Protocol Framework

图3 协议框架

3.2 本文所提方案

本文方案类似于联盟链,区块链间的节点通过共识机制达成了相互信任,同时实现了匿名性.由于区块链网络中的交易都是透明的,因此需要通过加密手段保护交易内容的机密性.所以,本文中的交易内容 $delete\text{-}request$ 是用户希望删除数据的请求集合经过 $AES.Enc$ 加密算法加密后形成的密文,例如 $delete\text{-}request=AES.Enc(delete\text{-}data_1,delete\text{-}data_2,\cdots,delete\text{-}data_n)$,其中 $delete\text{-}data_i(i=1,2,\cdots,n)$ 表示删除集合中的一个数据删除请求.tag 是文件的标签,由用户产生并上传给云服务器.$tag=H(name_f)$,其中 $name_f$ 表示删除文件的名字,只有用户知道.$H()$ 表示一种安全散列函数.

3.2.1 初始化阶段

系统管理者选择单调跨度程序的宽度 $c_{\max}$,并通过调用 $ABS.Setup()$ 算法生成公共参数 $PP(G_1,G_2,H,g,(h_0,h_1,\cdots,h_{c_{\max}}))$ 和主/公密钥对 (ASK,APK).当用户首次进入系统时,根据用户的属性调用 $ABS.AttrGen()$ 算法生成签名密钥 SK_u.

3.2.2 合约部署

将策略和交易上设计的智能契约作为输入,编译并将它们作为事务部署到块链中.如果智能合约通过验证且在区块链网络中达到了共识,它将永远记录在区块链中,并且区块链网络中所有的实体必须按照合约规则执行.

3.2.3 交易生成阶段

如果用户希望发布一个数据删除的请求交易,执行步骤如下:

1) 调用智能合约上的 $getPolicy()$ 算法,得到相对应的谓词 γ.如果用户的属性满足相应的谓词 γ,他可以发布一个合法的交易并执行下一步.

2) 用户调用属性签名算法 ABS 中的 $ABS.AttrGen()$ 算法和密钥生成算法生成属性签名密钥 SK_u 和一对公私钥 PK_i/SK_i.然后创建数据删除的请求交易 $T_x=\langle Adu,delete\text{-}request,Sig_{Reu},t_1,tag,Ads\rangle$,并触发在智能合约交易上的函数 $upTrans()$.交易 T_x 包括用户的地址 $Adu=SHA(PK_i)$,交易内容包括 $delete\text{-}request$ 和 tag,交易内容的签名 $Sig_{Reu}=\langle delete\text{-}request,t_1,tag\rangle$,交易的时间戳 t_1 和服务器的地址 $Ads=SHA(PK_s)$.最后在区块链网路中广播交易 T_x.

3.2.4 交易验证阶段

1) 验证节点 vdn(其中包括云服务器)调用智能合约交易上的 $getTrans()$,获得未验证交易.通过调用 $ABS.Ver()$ 算法验证交易中属性签名,然后,vdn 调用函数 $upTrans()$ 改变交易的状态,使其成为经过验证的合法交易.

2) 共识节点 bkn 使用 PoW 协商一致机制来链接交易.换句话说,服务器利用自己的私钥 SK_s 确认交易 T_x 后,生成一个待处理区块.如果此区块被 2/3 及以上的 bkn 节点批准,服务器才能将该块链接到区块链.

3.2.5 证书生成阶段

云服务器监视区块链网络,一旦用户发布新的有效请求,云服务器立刻响应用户的交易请求.在删除相应数据后产生一条用以验证云服务器是否诚实删除数据的区块链.为了更好地描述问题,假设用户发布的交易中需要删除数据的数量为 8,即 $1\leqslant j\leqslant 8$.

1) 收到用户发布的交易后,云服务器调用 $AES.Dec()$ 算法解密密文 $delete\text{-}request$ 获得相应的数据删除请求 $delete\text{-}data_i$,$delete\text{-}data_i$ 表示当前需要删除数据的请求消息.删除数据 $delete\text{-}data_i$

后,云服务器为其建立证据 $proof_j=(delete\text{-}data_i, Sig_{Reu}, Sig_s, t_1)$,其中 $Sig_s=(delete\text{-}data_i, tag, t_1)$,$SK_s$ 是服务器的私钥.然后,云服务器计算一个待处理区块 Hash 值 $h_i=H(h_{i-1}, ts_i, root_i)$,并用 Merkle 树结构将这些 $proof_j$ 归纳到该处理区块中,并生成根节点 $root_i$.如果此区块被 2/3 及以上的 bkn 节点批准,云服务器才能将该区块链接到区块链,其中 ts_i 表示当前区块产生的时间戳,h_{i-1} 表示 Hash 链中的前一个 Hash 值.另外,区块链形成后,服务器删除数据并建立删除证据 $proof_j$,发送 $proof_j$、区块时间戳 ts_i 给用户.这样区块链网络中的每个节点都会存储一条如图 4 所示的区块链.

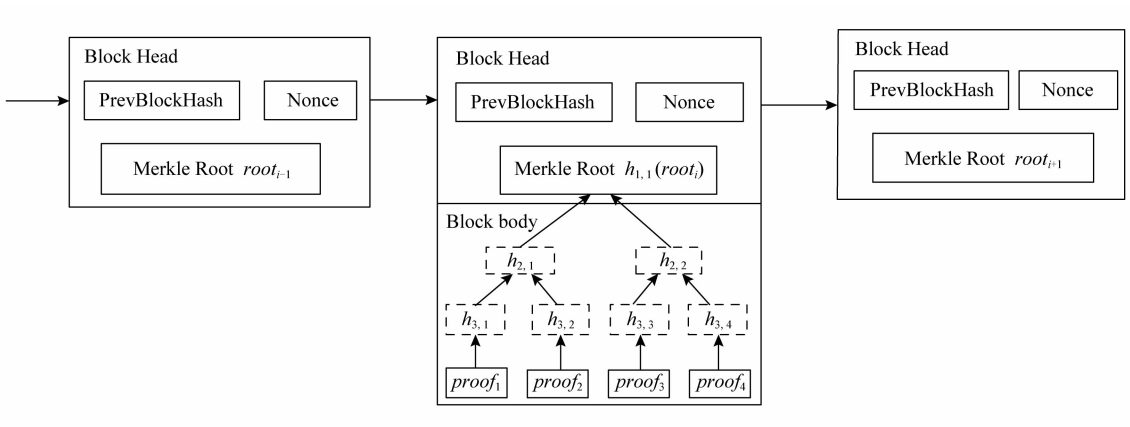

Fig. 4 "Evidence chain" structure diagram

图 4 "证据链"结构示意图

2) 用户收到最终的证据 $\tau=(proof_j, root_i, h_i, ts_i)$,通过 τ 可以验证云服务器是否诚实删除数据.

3.2.6 验证阶段

当用户接收到 τ 后,可以验证 τ 的正确性.为了不失一般性,以图 4 中的证据 $proof_4$ 作为一个例子来解释验证过程.首先,用户需要验证等式(1)~(4),确保云服务器产生的证据在区块链中.

$$h_i=H(h_{i-1}\parallel ts_i\parallel root_i); \tag{1}$$

$$h_{i+1}=H(h_i\parallel ts_{i+1}\parallel root_{i+1}); \tag{2}$$

……

$$h_{m-1}=H(h_{m-2}\parallel ts_{m-1}\parallel root_{m-1}); \tag{3}$$

$$h_m=H(h_{m-2}\parallel ts_m\parallel root_m); \tag{4}$$

其中,h_m 是最近发布的 Hash 值.

然后,用户首先计算 $h_{3,4}=H(proof_4)$,若 $h'_{3,4}=h_{3,4}$ 成立,则用户遍历 Merkle 树验证根节点.具体过程如下(如图 4 所示):

$$h'_{3,4}=H(proof_4);$$

$$h_{2,2}=H(h_{3,3}\parallel h_{3,4});$$

$$h_{1,1}=H(h_{2,1}\parallel h_{2,2});$$

……

$$root_i\stackrel{?}{=}h_{1,1}.$$

只有所有的验证和等式都是成立的,用户才可以确定服务器是诚实的以及这些证据 $proof_j$ 是可信的.

4 协议分析

4.1 安全性分析

1) 防窃听攻击

攻击者在公开信道上窃听通信信息,并且获得交易内容 $delete\text{-}request$.由于密文 $delete\text{-}request$ 的密钥是不可盗取的,攻击者无法获得任何有用的信息.因此,设计的协议能够抵御窃听攻击.

2) 防篡改攻击

假设攻击者修改了交易内容,由于攻击者不知道用户的属性,因此无法获取用户的属性密钥 SK_u.当验证节点 vdn 调用 $ABS.Ver()$ 算法时,交易无法通过验证,系统将丢弃攻击者的交易.

3) 防重放攻击

假设攻击者利用过期交易发动重放攻击,由于每个交易都含有一个独特的时间戳,通过对时间戳的检验,可以有效防止这类攻击.

4) 正确性

如果云服务器是诚实的并且能诚实的删除数据,那么用户得到的证据为 $\tau=(proof_j, root_i, h_i, ts_i)$,其中:证据 $proof_j$、根节点 $root_i$、区块前 Hash 值 h_{i-1},都是合法的.其次,由于 $h'_i=H(h_{i-1}\parallel ts_{i-1}\parallel root_i)$,因此,等式 $h'_i=h_i$ 总是成立.换句话说,h_i 总是验证阶段输出的结果.

5) 可追溯

如果云服务器不可信,用户将从 2 个方面实现对云服务器的追溯问责:

情形 1. 云服务器在没有收到删除请求交易的情况下删除数据.在这种情况下,云服务器没有删除请求的交易 $T_x=\langle Adu, delete\text{-}data_i, Sig_{Reu}, t_1, Ads\rangle$,其中 Sig_{Reu} 是基于用户属性的签名,Sig_{Reu} 只有用户可以计算,云服务器不能伪造它.因此,云服务器不能证明删除操作是否合法,必须承担相应责任.

情形 2. 云服务器根据删除请求删除数据后,用户依然可以在云中检测到删除数据的存在.对于这种情况,用户可以根据证据 $\tau=(proof_j, root_i, h_i, ts_i)$ 来检验服务器是否诚实删除数据.证据 $\tau=(proof_j, root_i, h_i, ts_i)$ 中的 $proof_j$ 包含由云服务器私钥 SK_s 的签名 Sig_s. Sig_s 只能由云服务器生成,所以用户可以证明云服务器是否诚实删除了数据.

为了便于把本文的协议与原有的协议比较,使用 Sig 表示签名生成运算,Hash 表示 Hash 运算,V 表示一次合法的签名验证.另外,我们假设 $n=\mathrm{lb}\, m$ (m 是文件的数量).如表 1 所示,提出的协议与原有协议进行比较.

Table 1 Performance Comparison

表 1 协议性能比较

Performance Index	Ref [10]	Ref [15]	Our Scheme
Third Party Audit(TTP)	Yes	Yes	No
Public Verification	Yes	Yes	Yes
Traceability	Yes	Yes	Yes
Privacy	No	Yes	Yes
Smart Contract	No	No	Yes
Compute (delete)	1Sig	1Sig	1Sig+1V
Compute (verify)	1V	1V	(n+2) Hash

4.2 性能评估

本节对所提出的协议进行实验评估,并在一个 1.8 GHz 处理器和 2 GB 内存的 Ubuntu16.04 虚拟机上使用 OpenSSL 工具实现.表 2~4 列出了协议中主要密码算法的时间开销.

如图 5 所示,在删除阶段,我们需要计算签名与验证算法.虽然,在删除阶段的计算时间开销稍大,但是删除阶段由云服务器上执行,我们可以接受较大的时间开销.验证阶段,我们只需要计算(n+2)个散列函数,大大降低了验证者的计算能力.因此,提出的协议更符合公开验证的实际情况.

Table 2 Time Cost of ABS Algorithm

表 2 ABS 算法的时间开销 s

Algorithm	Maximum Time	Minimum Time	Average Time
ABS. Setup	0.043 294	0.034 256	0.036 646
ABS. AttrGen	0.023 385	0.014 679	0.017 08
ABS. Sign	0.054 023	0.032 682	0.043 026
ABS. Ver	0.040 100	0.027 887	0.024 860

Table 3 Time Cost of AES Algorithm

表 3 AES 算法的时间开销 s

Algorithm	Maximum Time	Minimum Time	Average Time
Encryption	0.029 79	0.025 41	0.027 33
Decryption	0.027 58	0.024 41	0.025 12

Table 4 Time Cost of Hash Algorithm

表 4 Hash 算法的时间开销

Data Size/B	Time Cost/μs
16	0.253 33
64	0.412 52
256	0.912 41

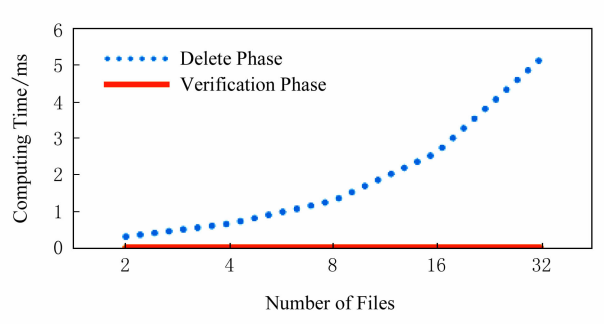

Fig. 5 Time cost for deletion and validation

图 5 删除与验证的时间消耗

5 结 论

本文设计了一个基于区块链的云数据删除验证协议,不需假设云服务器是可信的,它可能会篡改数据删除的结果并欺骗用户.与其他同类型的协议相比,本文的协议采用区块链系统保证用户可以检验云服务器的欺骗行为.如果不诚信的云服务器欺骗用户,用户可以在没有第三方可信机构的帮助下对删除证据进行检验,证明服务器的欺骗行为.

参 考 文 献

[1] Ateniese G, Burns R, Curtmola R, et al. Provable data possession at untrusted stores [C] //Proc of the 14th ACM Conf on Computer and Communications Security. New York: ACM, 2007: 598-609

[2] Ateniese G, Burns R, Curtmola R, et al. Remote data checking using provable data possession [J]. ACM Trans on Information and System Security, 2011, 14(1): 12. [2018-06-10]. https://doi.acm.org/10.1145/1952982.1952994

[3] Ateniese G, Di Pietro R, Mancini L V, et al. Scalable and efficient provable data possession [C] //Proc of the 4th Int Conf on Security and privacy in Communication Netowrks. New York: ACM, 2008: 1-10

[4] Cash D, Küpçü A, Wichs D. Dynamic proofs of retrievability via oblivious RAM [J]. Journal of Cryptology, 2017, 30(1): 22-57

[5] Chen Lanxiang, Xu Li. Research on provable data possession and recovery technology in cloud storage [J]. Journal of Computer Research and Development. 2012, 49(Suppl 1): 19-25 (in Chinese)
(陈兰香, 许力. 云存储服务中可证明数据持有及恢复技术研究[J]. 计算机研究与发展, 2012, 49(Suppl 1): 19-25)

[6] Halevi S, Harnik D, Pinkas B, et al. Proofs of ownership in remote storage systems [C] //Proc of the 18th ACM Conf on Computer and Communications Security. New York: ACM, 2011: 491-500

[7] Juels A, Kaliski J. PORs: Proofs of retrievability for large files [C] //Proc of the 14th ACM Conf on Computer and Communications Security. New York: ACM, 2007: 584-597

[8] Zheng Qingji, Xu Shouhuai. Fair and dynamic proofs of retrievability [C] //Proc of the 1st ACM Conf on Data and Application Security and Privacy. New York: ACM, 2011: 237-248

[9] Zhu Yan, Hu Hongxin, Ahn G J, et al. Efficient audit service outsourcing for data integrity in clouds [J]. Journal of Systems and Software, 2012, 85(5): 1083-1095

[10] Hao F, Clarke D, Zorzo A F. Deleting secret data with public verifiability [J]. IEEE Trans on Dependable and Secure Computing, 2016, 13(6): 617-629

[11] Sookhak M, Talebian H, Ahmed E, et al. A review on remote data auditing in single cloud server: Taxonomy and open issues [J]. Journal of Network and Computer Applications, 2014, 43(5): 121-141

[12] Wang Cong, Wang Qian, Ren Kui, et al. Ensuring data storage security in cloud computing [C] //Proc of the 17th Int Workshop on Quality of Service (2009 IWQoS). Piscataway, NJ: IEEE, 2009: 1-9

[13] Wang Qian, Wang Cong, Li Jin, et al. Enabling public verifiability and data dynamics for storage security in cloud computing [C] //Proc of the 14th European Symp on Research in Computer Security. Berlin: Springer, 2009: 355-370

[14] Liu Yang, Li Lixia. An efficient and secure public batch auditing protocol for dynamic cloud storage data [C] //Proc of the 2016 Int Computer Symp (ICS). Piscataway, NJ: IEEE, 2016: 671-675

[15] Wang Cong, Chow S M, Wang Qian, et al. Privacy-preserving public auditing for secure cloud storage [J]. IEEE Trans on Computers, 2013, 62(2): 362-375

[16] Yu Yong, Li Yannan, Au M H, et al. Public cloud data auditing with practical key update and zero knowledge privacy [C] //Proc of the Australasian Conf on Information Security and Privacy. Berlin: Springer, 2016: 389-405

[17] Zhang Xinpeng, Xu Chunxiang, Zhang Xiaojun, et al. Efficient dynamic integrity verification for big data supporting users revocability [J]. Information, 2016, 7(2): 31. [2018-06-10]. https://doi.org/10.3390/info7020031

[18] Zhang Xinpeng, Xu Chunxiang, Zhang Xinyan, et al. Efficient public auditing scheme for cloud storage supporting user revocability with proxy re-signature [J]. Journal of Computer Applications, 2016, 36(7): 1816-1821 (in Chinese)
(张新鹏, 许春香, 张新颜, 等. 基于代理重签名的支持用户可撤销的云存储数据公共审计方案[J]. 计算机应用, 2016, 36(7): 1816-1821)

[19] Li Jin, Au M H, Susilo W, et al. Attribute-based signature and its applications [C] //Proc of the 5th ACM Symp on Information, Computer and Communications Security. New York: ACM, 2010: 60-69

[20] Shamir A, Zippel R. On the security of the Merkle-Hellman cryptographic scheme (Corresp.) [J]. IEEE Trans on Information Theory, 1980, 26(3): 339-340

[21] Nakamoto S. Bitcoin: A peer-to-peer electronic cash system [J/OL]. 2008 [2018-06-01]. https://bitcoin.org/bitcoin.pdf

[22] Szabo N. Formalizing and securing relationships on public networks [J/OL]. First Monday, 1997, 2(9). [2018-06-01]. https://doi.org/10.5210/fm.v2i9.548

Liu Yining, born in 1973. Received his PhD degree in mathematics from Hubei University in 2007. Member of CCF. Currently professor in Guilin University of Electronic Technology. His main research interests include information security protocol, and data privacy.

Zhou Yuanjian, born in 1993. MEn candidate in the School of Computer Science and Information Security, Guilin University of Electronic Technology. His main research interests include information security protocol and Blockchain (237695226@qq.com).

Lan Rushi, born in 1986. Received his PhD degree in software engineering from the University of Macau. Currently assistant professor in Guilin University of Electronic Technology. His main research interests include image processing and information security (rslan2016@163.com).

Tang Chunming, born in 1972. Received his PhD degree in 2004 in Chinese Academy of Sciences. Currently professor in Guangzhou University. His current research interests are cryptography and its applications.

附录 A.

算法 A1. 交易上的智能合约.

```
Function getTrans()
{%获得交易
 if (k=1)
    {
     tx=vTrans[vTansL];
     vTrans[vTansL]=0;
     vTransL--;
    }
    else{
      tx=vnTrans[vTansL];
      vnTrans[vnTansL]=0;
      vnTransL--;
    }
 }
 Function upTrans(pm,tx,k)
 {%矿工提交合法交易
  if (k=1) then
     {
     if (pk miner) then
     {
       vTrans++;
       vTrans[vTransL]=tx;
     }
  else return 0;
  }
  else {
       uvTransL++;
       uvTrans[unTransL]=tx;
  } return 0;
}
```

算法 A2. 在调用策略上的智能合约.

```
Structure Unit[]
  {
   Predicate;
   Policy;
   Hash;
  }
Function policy()
  {%当智能合约部署后,自动调用.
   userID=sender.pk;
   Unit[];
   length=0;
   return 1;
  }
Function getpolicy(policy)
{%通过调用策略获得谓语
 if (policy t[i].policy) then
    return t[i].predicate
 else return 0;
}
Function checkpolicy (predicate, attribute)
{%验证属性
 if (t[i].hash=h(predicate)) &&
    attribute=t[i].policy) then
    return 1;
 else return 0;
}
Function uppolicy (pk, predicate, policy)
{%更新策略
 if (t[i].hash=h(predicate)) then
    if policy=null then
       Delete(t[i]);
          length--;
       else
          t[i].policy=policy;
    else
          if(policy=not null) then
          length++;
          t[length].hash=(predicate)
          t[length].policy=policy;
    return 1;
}
```