(19)中华人民共和国国家知识产权局

(12)发明专利

(10)授权公告号 CN 109508552 B
(45)授权公告日 2021.04.30

(21)申请号 201811330536.4

(22)申请日 2018.11.09

(65)同一申请的已公布的文献号
申请公布号 CN 109508552 A

(43)申请公布日 2019.03.22

(73)专利权人 江苏大学
地址 212000 江苏省镇江市京口区学府路302号

(72)发明人 王良民 孙世璞 姜顺荣 余春堂 段梦杰 谢晴晴 邢玉萍 朱会娟

(74)专利代理机构 南京华恒专利代理事务所(普通合伙) 32335
代理人 宋方园

(51)Int.Cl.
G06F 21/60(2013.01)
G06F 21/62(2013.01)
G06Q 20/06(2012.01)
G06Q 20/38(2012.01)
G06Q 30/06(2012.01)
G06Q 40/04(2012.01)

(56)对比文件
CN 108462568 A,2018.08.28
CN 106982205 A,2017.07.25
US 2018322587 A1,2018.11.08
SHANGPING WANG 等.A Blockchain-Based Framework for Data Sharing With Fine-Grained Access Control in Decentralized Storage Systems.《IEEE Access》.2018,
simmel_.Storj:区块链在云存储上的应用.《CSDN》.2017,
yuanchaoknightt.门罗币基础技术介绍.《CSDN》.2017,

审查员 陈丽娜

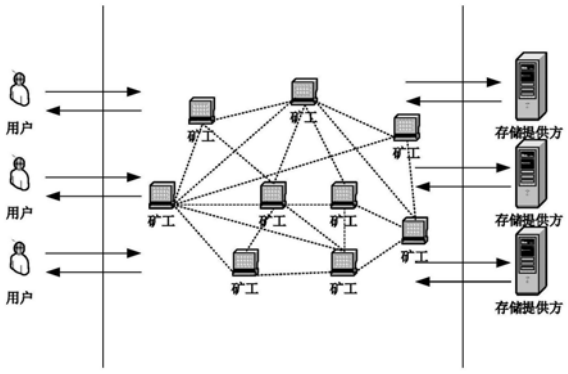
权利要求书4页 说明书12页 附图3页

(54)发明名称

分布式云存储系统的隐私保护方法

(57)摘要

本发明公开一种分布式云存储系统的隐私保护方法,包括用户、矿工、存储提供方,实现系统初始化、文件上传、文件分享、文件下载、文件删除等操作。本发明将分布式云存储系统与区块链技术结合,并将文件所有权处理过程作为交易处理,实现分布式云存储系统中文件所有权处理过程的隐私保护,包括发送方、接收方、交易内容的隐藏。另外,本发明在保证用户隐私的同时,能够有效地抵制重放攻击、数据伪造攻击。



CN 109508552 B

1.一种分布式云存储系统的隐私保护方法,其特征在于:所述分布式云存储系统包括用户、矿工和存储提供方;所述用户是使用系统服务的实体,包括数据所有者μ_i和数据使用者μ_j,数据所有者μ_i将数据外包给存储提供方,并使用系统服务进行下载和删除操作,同时数据所有者μ_i将文件F_i分享给数据使用者μ_j;矿工和存储提供方为用户提供云存储服务,其中,矿工负责打包区块和维护区块,存储提供方将自己空闲的硬盘空间出租给网络组成分布式的存储空间;另外,矿工和存储提供方通过提供服务获得代币奖励;

隐私保护方法具体包括以下步骤:

(1)系统初始化:输入公共参数(1,G),1为基点的素数阶,G为椭圆曲线的基点;用户选取随机数a∈[1,1-1],b∈[1,1-1]组成私钥对(a,b),a≠b;同时,用户计算A=aG,B=bG,作为公钥对(A,B);另外,用户计算ripemd160(sha256(A,B))作为其标准地址,同时也作为其ID;

$1=2^{252}+27742317777372353535851937790883648493$;

(2)文件上传,即数据所有者μ_i要将文件F_i上传到分布式存储系统;步骤(2)的详细过程如下:

(2.1)数据所有者μ_i选取哈希函数$H:\{0,1\}^* \to \mathbb{Z}_p$,并计算文件$F_i$的哈希值$hash_i=H(F_i)$,然后数据所有者$\mu_i$向系统广播上传请求:

$$\mu_i \to *: upload, hash_i, ID_{\mu_i}$$

其中,$hash_i$为文件F_i的哈希值,ID_{μ_i}为数据所有者μ_i的标准地址;

(2.2)当系统接收到数据所有者μ_i的上传请求,系统中的矿工开始执行工作量证明算法POW来争取记账权;执行完POW算法后,假定矿工$Node_j$获取到记账权;然后矿工$Node_j$通知数据所有者μ_i开始上传文件:

$Node_j \to \mu_i$:upload,$hash_i$

(2.3)当数据所有者μ_i接收到上传指令,数据所有者μ_i在客户端侧使用同态加密算法AES256-CTR加密文件F_i:$F_i'=Enc(F_i)$,其中加密密钥为文件哈希$hash_i$;然后,μ_i在客户端侧对加密后的文件F_i'进行分块得到文件碎片$\{shard_1, shard_2, ..., shard_n\}$,其中每个碎片大小为8M,不足8M的空间用0填充;接着数据所有者μ_i求取各碎片哈希$\{hash_{sh1}, hash_{sh2}, ..., hash_{shn}\}$并建立Merkle Tree,用于文件审计;最后$\mu_i$将文件碎片分散地存储在系统中,并在分布式哈希表DHT中生成文件索引;同时,矿工$Node_j$将文件元数据metadata打包进区块,其中metadata=$\{hash_i, MerkleRoot_i\}$;矿工节点$Node_j$通知数据所有者μ_i开始打包交易:

$Node_j \to \mu_i$:transaction,$hash_i$

(2.4)当数据所有者μ_i接收到交易打包指令,μ_i开始生成文件上传交易Tx_k,上传交易中交易发送方和交易接收方都是数据所有者μ_i;

上述(2.4)中文件上传交易生成的具体过程如下:

(2.4.1)μ_i选取随机数$r_k \in [1,1-1]$;然后μ_i计算隐匿地址$P_k=\mathcal{H}_s(r_k A_{\mu_i})G+B_{\mu_i}$,交易公钥$R_k=r_kG$,其中$(A_{\mu_i}, B_{\mu_i})$为$\mu_i$的公钥对,$\mathcal{H}_s:\{0,1\}\to\mathbb{F}_q$为密码散列函数;

(2.4.2)μ_i选取随机数$x_k \in [1,1-1]$作为私钥,并计算对应的公钥$PK_k=x_kG$,密钥镜像$I_k=x_k\mathcal{H}_p(PK_k)$,其中,$\mathcal{H}_p: E(\mathbb{F}_q)\to E(\mathbb{F}_q)$为确定性哈希函数;

(2.4.3)μ_i使用对称加密算法计算$Enc_{ID_{\mu_i}}(R,hash_i,MAC_k)$作为交易内容$Proof_k$,用于授予交易接收方文件所有权,并可用于文件所有权验证,其中$MAC_k = MAC_{ID_{\mu_i}}(R,hash_i)$,R为随机数,$ID_{\mu_i}$为$\mu_i$的标准地址,同时作为对称加密密钥;

(2.4.4)μ_i将R_k,P_k,I_k以及$Proof_k$打包进交易Tx_k,然后μ_i计算环签名σ_k,对交易Tx_k进行签名并发送到网络;网络中任何一方都可以验证交易签名,并且不会泄露交易发送方的信息;

(2.4.5)矿工节点$Node_j$验证交易并将交易打包进区块;此处,数据所有者μ_i作为交易接收方向系统发送交易Tx_k信息P_k,I_k,$Proof_k$来验证μ_i对于文件F_i的所有权,并用于消费;

(3)文件删除,即数据所有者μ_i要删除其对于文件F_i的所有权;

(4)文件分享,即文件F_i的数据所有者μ_i授予数据使用者μ_j文件F_i的所有权;

(5)文件下载。

2.根据权利要求1所述的分布式云存储系统的隐私保护方法,其特征在于:所述步骤(3)中数据所有者删除文件所有权的详细过程如下:

(3.1)首先,数据所有者μ_i向系统广播删除请求:

$\mu_i \rightarrow *: delete, hash_i, P_k, ID_{\mu_i}$,

其中,$hash_i$为文件F_i哈希;P_k是能够证明μ_i对文件F_i的所有权的交易的隐匿地址,ID_{μ_i}为数据所有者μ_i标准地址;

(3.2)当系统接收到数据所有者μ_i的删除请求,系统中的矿工开始执行工作量证明算法POW来争取记账权;执行完POW算法后,假定矿工节点$Node_j$获取到记账权;

(3.3)矿工节点$Node_j$验证μ_i对文件F_i的所有权;

(3.4)当数据所有者μ_i通过所有权验证以后,矿工节点$Node_j$从地址为P_k的交易获取其密钥镜像I_k,并将I_k添加到交易失效名单Blacklist,并向全网矿工节点广播;网络中的矿工节点接收到广播以后更新交易失效名单Blacklist;

上述所有权验证方法为:

输入:文件哈希$hash_i$, 交易地址P_k, 数据所有者μ_i标准地址ID_{μ_i}.

输出:所有权验证结果.

1: $Node_j$ 根据地址 P_k 找到对应交易 Tx_k，获取交易内容 $\mathrm{Pr}oof_k$，密钥镜像 I_k；

2: **if** $I_k \notin Blacklist$ **then**

3: $Node_j$ 使用解密函数计算：$\mathrm{Pr}oof_k'=Dec_{ID_{\mu_i}}(\mathrm{Pr}oof_k)=\{\mathrm{R}_k', hash_i', MAC_k'\}$；

4: $Node_j$ 计算 $MAC_k' = MAC_{ID_{\mu_i}}(R_k', hash_i')$；

5: **if** $MAC_k^* = MAC_k'$ **then**

6: 返回 1;

7: **else**

8: 返回⊥;

9: **end if**

10 **else**

11: 返回⊥.

12: **end if**

其中，R_k'，hash_i'，MAC_k'为解密Proof_k'之后获取的内容，其含义分别对应Proof_k中的R_k、hash_i、MAC_k。

3.根据权利要求1所述的分布式云存储系统的隐私保护方法，其特征在于：所述步骤(4)中数据所有者μ_i分享文件F_i给数据使用者μ_j的详细过程如下：

(4.1)首先，μ_i向系统广播分享请求：

$$\mu_i \to *: share, hash_i, P_k, ID_{\mu_i}, ID_{\mu_j},$$

其中，hash_i为文件F_i哈希；P_k是能够证明μ_i对文件F_i的所有权的交易的隐匿地址；ID_{μ_i}, ID_{μ_j}分别为数据所有者μ_i和数据使用者μ_j的标准地址；

(4.2)当系统接收到μ_i的分享请求，系统中的矿工开始执行工作量证明算法POW来争取记账权；执行完POW算法后，假定矿工节点Node_j获取到记账权；

(4.3)矿工节点Node_j验证数据所有者μ_i对文件F_i的所有权；

(4.4)μ_i通过所有权验证以后，Node_j向μ_i发送分享指令：

$$Node_j \to \mu_i : share, hash_i, ID_{\mu_j}$$

(4.5)当μ_i接收到指令，μ_i开始生成文件分享交易Tx_τ，其中，μ_i作为交易发送方，μ_j作为交易接收方；

(4.6)μ_j作为交易接收方，检查每一个新生成区块中的交易，提取交易中的隐匿地址P和交易公钥R′，使用自己的私钥对(a_{μ_j}, b_{μ_j})计算 $P' = \mathcal{H}_s(a_{\mu_j} R')G + b_{\mu_j} G$，判断 $P' \stackrel{?}{=} P$，其中，$\mathcal{H}_s: \{0,1\} \to \mathbb{F}_q$为密码散列函数；由于 $a_{\mu_j} R' = (a_{\mu_j} r')G = r' A_{\mu_j}$，$b_{\mu_j} G = B_{\mu_j}$，如果$\mu_j$为该交易的接

收方，则等式成立；反之，不成立；

(4.7)当μ_j找到交易Tx_τ之后，μ_j计算$x_\tau=\mathcal{H}_s(a_{\mu_j}R_\tau)+b_{\mu_j}$；$\mu_j$使用$x_\tau$恢复交易$Tx_\tau$，并在消费交易$Tx_\tau$时使用$x_\tau$作为交易私钥。

4.根据权利要求3所述的分布式云存储系统的隐私保护方法，其特征在于：所述步骤(4.5)中文件分享交易生成的方法为：

(4.5.1)交易发送方μ_i选取随机数$r_\tau\in[1,l-1]$，并获取交易接收方μ_j的公钥对(A_{μ_j},B_{μ_j})；然后μ_i计算隐匿地址$P_\tau=\mathcal{H}_s(r_\tau A_{\mu_j})G+B_{\mu_j}$，交易公钥$R_\tau=r_\tau G$，其中，$\mathcal{H}_s:\{0,1\}\rightarrow\mathbb{F}_q$为密码散列函数；

(4.5.2)交易发送方μ_i使用对称加密算法计算$Enc_{ID_{\mu_j}}(R,hash_i,MAC_\tau)$作为交易内容$Proof_\tau$，用于授予交易接收方文件所有权，并可用于文件所有权验证，其中$MAC_\tau=MAC_{ID_{\mu_j}}(R,hash_i)$，R为随机数，加密密钥为$\mu_j$的标准地址$ID_{\mu_j}$；

(4.5.3)交易发送方μ_i选取随机数$x_\tau\in[1,l-1]$，同时也作为签名私钥，并计算对应的公钥$PK_\tau=x_\tau G$，密钥镜像$I_\tau=x_\tau\mathcal{H}_p(PK_\tau)$，其中，$\mathcal{H}_p:E(\mathbb{F}_q)\rightarrow E(\mathbb{F}_q)$为确定性哈希函数；

(4.5.4)交易发送方μ_i将R_τ，P_τ，I_τ以及$Proof_\tau$打包进交易Tx_τ，且μ_i计算环签名σ_τ，对交易Tx_τ进行签名并发送到网络，网络中任何一方都可以验证交易签名，并且不会泄露交易发送方的信息；

(4.5.5)矿工节点$Node_j$验证交易并将交易打包进区块。

5.根据权利要求1所述的分布式云存储系统的隐私保护方法，其特征在于：所述(5)的详细过程如下：

(5.1)数据所有者μ_i向系统广播下载请求：

$\mu_i\rightarrow *:download,hash_i,P_k,ID_{\mu_i}$，

其中，$hash_i$为文件F_i的哈希；P_k为能够证明μ_i对文件F_i的所有权的交易的隐匿地址，ID_{μ_i}为μ_i标准地址；

(5.2)当系统接收到μ_i的下载请求，系统中的矿工开始执行工作量证明算法(POW)来争取记账权；执行完POW算法后，假定矿工节点$Node_j$获取到记账权；

(5.3)矿工节点$Node_j$验证μ_i对文件F_i的所有权；

(5.4)μ_i通过所有权验证以后，$Node_j$在DHT中查询文件实际存储地址$addr_i$，并将下载地址$addr_i$返回给μ_i：

$Node_j\rightarrow\mu_i:hash_i,addr_i$；

(5.5)μ_i使用下载工具将文件从地址$addr_i$恢复到本地。

分布式云存储系统的隐私保护方法

技术领域

[0001] 本发明涉及一种分布式存储技术，具体涉及一种分布式云存储系统的隐私保护方法。

背景技术

[0002] 随着云存储技术的发展，用户可以很方便的将数据外包到云中，并在云中对数据进行分享、下载、修改等操作。然而传统的云存储模型几乎唯一的依赖于中心化的云服务器来提供服务，这种基于客户端-服务端信任的模式具有一些固有弱点，很容易受到攻击，比如中间人攻击、恶意软件攻击、中心化的服务器也可能因技术故障导致数据不可用以及用户隐私的泄露。分布式云存储可以较好的解决这些问题。常见的分布式云存储系统有HDFS、Ceph、Pangu等。分布式云存储系统去除了中心化的服务器，保证了数据的可靠性，同时，还具有存储容量大、高吞吐量、服务高可用、高效运维、低成本等优点。

[0003] 另外，区块链技术的快速发展，使得越来越多的应用将区块链使用到分布式云存储中。区块链的引入，为系统提供了奖励机制，使得更多的用户参与到系统中。进一步的，可以使用区块链的防篡改功能，为云存储系统提供便利，增加安全性，这一改进使得基于区块链的分布式云存储系统越来越流行，常见的基于区块链的分布式云存储系统有Storj，IPFS等。

[0004] 基于区块链的分布式云存储系统没有中心化的服务器。因此，具有中心化的云存储模型所不具备的一些优点：首先，没有了中心化的服务器，因服务器故障以及服务器安全漏洞造成的数据不可用问题大大降低；其次，采用用户侧加密保证了数据安全性，采用可恢复性证明保证了数据的完整性；另外，开放的存储市场可以降低存储成本，并且在抵制审查制度、外部干预、非授权访问等方面具有一定的优势。

[0005] 以Storj为例，分布式云存储的存储过程：Storj鼓励用户将自己空闲的硬盘空间出租给网络，组成分布式的的存储空间，这样的用户被称为农户，相当于比特币网络中的矿工。当用户要上传文件时，首先，由用户在客户端对文件进行分块、加密，然后分散地存储到网络，并使用分布式哈希表(DHT)来存储文件碎片的位置信息。文件碎片存取能更好的保护数据安全性，因为没有一个农户拥有完整的副本。其次，为保证文件可用性，Storj提供了可恢复性证明以及冗余策略。另外，Storj采用区块链来记录信息，而不是采用中心化的数据库。区块链上并不存储文件内容，而是存储文件的元数据，包括文件的哈希、merkle根以及其他必要信息。最后，Storj提供了一种奖励机制，为区块链矿工和提供存储空间的农户提供代币奖励。然而，Storj采用中本聪式的区块链，账本是开放的，每个人都可以看到里面的每一笔交易以及交易的踪迹，因此存在隐私泄露问题。

[0006] 关于区块链数据隐私保护问题，已有一些工作。DASH使用混币技术来提供支付的保密性，通过将不同的交易混合然后分发给接收者，以此实现交易的匿名。ZCASH使用零知识证明技术，保证了只有那些拥有查看密钥的人才能看到交易的内容。用户拥有完全的控制权，他们可自行选择向其他人提供查看密钥。Monero对交易进行完全的隐藏，可以对交易

发送方、接收方、交易内容进行隐匿。对于分布式云存储的隐私保护方案，目前已有一些工作，主要是对分布式云存储的奖励机制进行隐藏，实现分布式云存储过程中代币交易过程的隐私保护，分别对交易发送方、接收方、交易内容进行隐藏。对于交易发送方，采用环签名进行隐藏；对于交易接收方，采用隐匿地址技术；对于交易内容，采用混币方法。但是，该方案只对分布式云存储的代币交易过程进行隐私保护，对于文件存储过程，隐私泄露问题依旧存在。

发明内容

[0007] 发明目的：本发明的目的在于解决现有技术中存在的不足，提供一种分布式云存储系统的隐私保护方法，本发明实现分布式云存储过程中的隐私保护，不仅实现交易过程的隐私保护，而且实现存储过程中用户相关、文件相关的隐私保护。

[0008] 技术方案：本发明的一种分布式云存储系统的隐私保护方法，所述分布式云存储系统包括用户、矿工和存储提供方；用户包括数据所有者μ_i和数据使用者μ_j，数据所有者μ_i使用系统服务将数据外包给存储提供方，进行下载和删除操作，同时数据所有者μ_i将文件分享给数据使用者μ_j；矿工和存储提供方为用户提供分布式云存储服务，其中，矿工负责打包区块和维护区块，存储提供方将自己空闲的硬盘空间出租给网络组成分布式的存储空间；另外，矿工和存储提供方通过提供服务获得代币奖励；

[0009] 隐私保护方法具体包括以下步骤：

[0010] (1)系统初始化：输入公共参数(l,G)，l为基点的素数阶，G为椭圆曲线的基点；用户选取随机数a∈[1,l-1]，b∈[1,l-1]组成私钥对(a,b)，a≠b；同时，用户计算A＝aG，B＝bG，作为公钥对(A,B)；另外，用户计算ripemd160(sha256(A,B))作为其标准地址，同时也作为其ID；

[0011] $l=2^{252}+27742317777372353535851937790883648493$；

[0012] (2)文件上传，即数据所有者μ_i要将文件F_i上传到分布式存储系统；

[0013] (3)文件删除，即数据所有者μ_i要删除其对于文件F_i的所有权；

[0014] (4)文件分享，即文件F_i的所有者μ_i授予数据使用者μ_j文件F_i的所有权；

[0015] (5)文件下载。

[0016] 进一步的，所述步骤(2)的详细过程如下：

[0017] (2.1)数据所有者μ_i选取哈希函数$H:\{0,1\}^*\rightarrow\mathbb{Z}_p$，例如SHA-256，并计算文件$F_i$的哈希值$hash_i=H(F_i)$，然后数据所有者$\mu_i$向系统广播上传请求：

[0018] $$\mu_i\rightarrow *:upload,hash_i,ID_{\mu_i}$$

[0019] 其中，$hash_i$为文件F_i的哈希值，ID_{μ_i}为数据所有者μ_i的标准地址；

[0020] (2.2)当系统接收到数据所有者μ_i的上传请求，系统中的矿工开始执行工作量证明算法POW来争取记账权(即交易打包权)；执行完POW算法后，假定矿工$Node_j$获取到记账权；然后矿工$Node_j$通知数据所有者μ_i开始上传文件：

[0021] $Node_j\rightarrow\mu_i:upload,hash_i$

[0022] (2.3)当数据所有者μ_i接收到上传指令，μ_i在客户端侧使用同态加密算法AES256-CTR加密文件F_i：$F_i'=Enc(F_i)$，其中加密密钥为文件哈希$hash_i$；然后，μ_i在客户端侧对加密

后的文件F_i'进行分块得到文件碎片{$shard_1$,$shard_2$,...,$shard_n$},其中每个碎片大小为8M,不足8M的空间用0填充;接着数据所有者μ_i求取各碎片哈希{$hash_{sh1}$,$hash_{sh2}$,...,$hash_{shn}$}并建立Merkle Tree,用于文件审计;最后,μ_i将文件碎片分散地存储在系统中,并在分布式哈希表(DHT)中生成文件索引;同时,矿工$Node_j$将文件元数据(Merkle根、文件哈希值等)打包进区块,其中metadata={$hash_i$,$MerkleRoot_i$};矿工$Node_j$通知数据所有者μ_i开始打包交易:

[0023] $Node_j \rightarrow \mu_i$:transaction,$hash_i$

[0024] (2.4)当数据所有者μ_i接收到交易打包指令,μ_i开始生成文件上传交易Tx_k,上传交易中交易发送方和接收方都是数据所有者μ_i。

[0025] 进一步的,所述步骤(2.4)中文件上传交易生成的具体过程如下:

[0026] (2.4.1)μ_i选取随机数$r_k \in [1,l-1]$;然后μ_i计算隐匿地址$P_k = \mathcal{H}_s(r_k A_{\mu_i})G + B_{\mu_i}$,交易公钥$R_k = r_k G$,其中$(A_{\mu_i}, B_{\mu_i})$为数据所有者$\mu_i$的公钥对,$\mathcal{H}_s : \{0,1\} \rightarrow \mathbb{F}_q$为密码散列函数;

[0027] (2.4.2)μ_i选取随机数$x_k \in [1,l-1]$作为私钥,并计算对应的公钥$PK_k = x_k G$,密钥镜像$I_k = x_k \mathcal{H}_p(PK_k)$,其中,$\mathcal{H}_p : E(\mathbb{F}_q) \rightarrow E(\mathbb{F}_q)$为确定性哈希函数;

[0028] (2.4.3)μ_i使用对称加密算法计算$Enc_{ID_{\mu_i}}(R, hash_i, MAC_k)$作为交易内容$Proof_k$,用于授予交易接收方文件所有权,并可用于文件所有权验证,其中$MAC_k = MAC_{ID_{\mu_i}}(R, hash_i)$,R为随机数,$ID_{\mu_i}$为$\mu_i$的标准地址,同时作为对称加密密钥;

[0029] (2.4.4)μ_i将R_k,P_k,I_k以及$Proof_k$打包进交易Tx_k,然后μ_i计算环签名σ_k,对交易Tx_k进行签名并发送到网络;网络中任何一方都可以验证交易签名,并且不会泄露交易发送方;

[0030] (2.4.5)矿工节点$Node_j$验证交易并将交易打包进区块;

[0031] 其中,数据所有者μ_i作为交易接收方向矿工节点$Node_j$发送交易Tx_k信息P_k,I_k,$Proof_k$来验证μ_i对于文件F_i的所有权,并用于消费(分享、删除、下载操作)。

[0032] 进一步的,所述步骤(3)中数据所有者删除文件所有权的详细过程如下:

[0033] (3.1)首先,数据所有者μ_i向系统广播删除请求:

[0034] $\mu_i \rightarrow *: delete, hash_i, P_k, ID_{\mu_i}$,

[0035] 其中,$hash_i$为文件F_i哈希;P_k是能够证明μ_i对文件F_i的所有权的交易的隐匿地址,ID_{μ_i}为数据所有者μ_i标准地址;

[0036] (3.2)当系统接收到μ_i的删除请求,系统中的矿工开始执行工作量证明算法POW来争取记账权;执行完POW算法后,假定矿工节点$Node_j$获取到记账权;

[0037] (3.3)矿工节点$Node_j$验证μ_i对文件F_i的所有权;

[0038] (3.4)当数据所有者μ_i通过所有权验证以后,矿工节点$Node_j$从地址为P_k的交易获取其密钥镜像I_k,并将I_k添加到交易失效名单Blacklist,并向全网广播;网络中的矿工节点接收到广播以后更新交易失效名单Blacklist;

[0039] 上述所有权验证方法为:

输入:文件哈希 $hash_i$，交易地址 P_k，数据所有者 μ_i 标准地址 ID_{μ_i}.

输出:所有权验证结果.

1: $Node_j$ 根据地址 P_k 找到对应交易 Tx_k，获取交易内容 $\mathrm{Pr}oof_k$，密钥镜像 I_k;

2: **if** $I_k \notin Blacklist$ **then**

3: $Node_j$ 使用解密函数计算：$\mathrm{Pr}oof_k'=Dec_{ID_{\mu_i}}(\mathrm{Pr}oof_k)=\{\mathrm{R}_k', hash_i', MAC_k'\}$;

[0040] 4: $Node_j$ 计算 $MAC_k' = MAC_{ID_{\mu_i}}(R_k', hash_i')$;

5: **if** $MAC_k^* = MAC_k'$ **then**

6: 返回 1;

7: **else**

8: 返回⊥;

9: **end if**

10: **else**

[0041] 11: 返回⊥.

12: **end if**

[0042] 进一步的,所述步骤(4)中数据所有者μ_i要分享文件F_i给数据使用者μ_j的详细过程如下:

[0043] (4.1)首先,μ_i向系统广播分享请求:

[0044] $\mu_i \rightarrow *: share, hash_i, P_k, ID_{\mu_i}, ID_{\mu_j}$,

[0045] 其中,$hash_i$为文件F_i哈希;P_k是能够证明μ_i对文件F_i的所有权的交易的隐匿地址;ID_{μ_i}, ID_{μ_j} 分别为数据所有者μ_i和数据使用者μ_j的标准地址;

[0046] (4.2)当系统接收到μ_i的分享请求,系统中的矿工开始执行工作量证明算法POW来争取记账权;执行完POW算法后,假定矿工节点$Node_j$获取到记账权;

[0047] (4.3)矿工节点$Node_j$验证μ_i对文件F_i的所有权;

[0048] (4.4)μ_i通过所有权验证以后,$Node_j$向μ_i发送分享指令:

[0049] $Node_j \rightarrow \mu_i : share, hash_i, ID_{\mu_j}$。

[0050] (4.5)当μ_i接收到指令,开始生成文件分享交易Tx_τ,其中,μ_i作为交易发送方,μ_j作为交易接收方;

[0051] (4.6)μ_j作为交易接收方,检查每一个新生成的区块中的交易,提取交易中的隐匿地址P和交易公钥R′,使用自己的私钥对(a_{μ_j}, b_{μ_j})计算 $P' = \mathcal{H}_s(a_{\mu_j} R')G + b_{\mu_j} G$,判断 $P' \stackrel{?}{=} P$,

其中，$\mathcal{H}_s:\{0,1\}\to\mathbb{F}_q$为密码散列函数；由于$a_{\mu_j}R'=(a_{\mu_j}r')G=r'A_{\mu_j}$，$b_{\mu_j}G=B_{\mu_j}$，如果$\mu_j$为该交易的接收方，则等式成立；反之，不成立；

[0052] (4.7)当μ_j找到交易Tx_τ之后，μ_j计算$x_\tau=\mathcal{H}_s(a_{\mu_j}R_\tau)+b_{\mu_j}$；$\mu_j$使用$x_\tau$恢复交易$Tx_\tau$，并在消费交易(分享、下载、删除)$Tx_\tau$时使用$x_\tau$作为交易私钥。

[0053] 进一步的，所述步骤(4.5)中文件分享交易生成的方法为：

[0054] (4.5.1)交易发送方μ_i选取随机数$r_\tau\in[1,l-1]$，并获取交易接收方μ_j的公钥对(A_{μ_j},B_{μ_j})；然后μ_i计算隐匿地址$P_\tau=\mathcal{H}_s(r_\tau A_{\mu_j})G+B_{\mu_j}$，交易公钥$R_\tau=r_\tau G$，其中，$\mathcal{H}_s:\{0,1\}\to\mathbb{F}_q$为密码散列函数；

[0055] (4.5.2)交易发送方μ_i使用对称加密算法计算$Enc_{ID_{\mu_j}}(R,hash_i,MAC_\tau)$作为交易内容$Proof_\tau$，用于授予交易接收方文件所有权，并可用于文件所有权验证，其中$MAC_\tau=MAC_{ID_{\mu_j}}(R,hash_i)$，R为随机数，加密密钥为$\mu_j$的标准地址$ID_{\mu_j}$；

[0056] (4.5.3)交易发送方μ_i选取随机数$x_\tau\in[1,l-1]$，同时也作为签名私钥，并计算对应的公钥$PK_\tau=x_\tau G$，密钥镜像$I_\tau=x_\tau\mathcal{H}_p(PK_\tau)$，其中，$\mathcal{H}_p:E(\mathbb{F}_q)\to E(\mathbb{F}_q)$为确定性哈希函数；

[0057] (4.5.4)交易发送方μ_i将R_τ，P_τ，I_τ以及$Proof_\tau$打包进交易Tx_τ，进一步的，μ_i计算环签名σ_τ，对交易Tx_τ进行签名并发送到网络，网络中任何一方都可以验证交易签名，并且不会泄露交易发送方；

[0058] (4.5.5)矿工节点$Node_j$验证交易并将交易打包进区块。

[0059] 本发明中，Tx_k为文件上传交易，Tx_τ为文件分享交易，Tx_k的交易发送方和接收方都是数据所有者μ_i，Tx_τ的交易发送方是μ_i，交易接收方是μ_j。

[0060] 进一步的，所述(5)的详细过程如下：

[0061] (5.1)数据所有者μ_i向系统广播下载请求：

[0062] $\mu_i\to *:download,hash_i,P_k,ID_{\mu_i}$，

[0063] 其中，$hash_i$为文件F_i的哈希；P_k为能够证明μ_i对文件F_i的所有权的交易的隐匿地址，ID_{μ_i}为数据所有者μ_i标准地址；

[0064] (5.2)当系统接收到μ_i的下载请求，系统中的矿工开始执行工作量证明算法(POW)来争取记账权；执行完POW算法后，假定矿工节点$Node_j$获取到记账权；

[0065] (5.3)矿工节点$Node_j$验证μ_i对文件F_i的所有权；

[0066] (5.4)μ_i通过所有权验证以后，$Node_j$在DHT网络中查询文件实际存储地址$addr_i$，并将下载地址$addr_i$返回给μ_i：

[0067] $Node_j\to\mu_i:hash_i,addr_i$；

[0068] (5.5)μ_i使用下载工具将文件从地址$addr_i$恢复到本地。

[0069] 本发明中环签名生成方法如下：

算法 2 环签名生成算法

输入: 环大小n，交易公钥R_τ，签名公钥PK，密钥镜像I.

输出: 环签名σ.

1: μ_i随机选取n个用户公钥，构成子集$S'=\{PK_1,...,PK_n\}$;

2: μ_i选取随机数s，其中$0\le s\le n$;

3: **for** $i=0,...,n$ **do**

4: μ_i选取随机数q_i，ω_i，其中，$1\le q_i\le l$，$1\le\omega_i\le l$

5: **if** $i\neq s$ **then**

[0070]

6: μ_i计算$L_i=q_iG+\omega_iPK_i$

7: μ_i计算$M_i=q_i\mathcal{H}_p(PK_i)+\omega_iI$

8: **else**

9: μ_i计算$L_i=q_iG$

10: μ_i计算$M_i=q_i\mathcal{H}_p(PK_i)$

11: **end if**

12: **end for**

13: μ_i计算$c=\mathcal{H}_s(R_\tau,L_1,...,L_n,M_1,...,M_n)$

14: **for** $i=0,...,n$ **do**

15: **if** $i \neq s$ **then**

16: μ_i 计算 $c_i=\omega_i$

17: μ_i 计算 $m_i=q_i$

18: **else**

[0071]

19: μ_i 计算 $c_i=c-\sum_{i=0}^{n} c_i \mod l$

20: μ_i 计算 $m_i=q_i-c_i x$

21: **end if**

22: **end for**

23: 返回 环签名 $\sigma=(I,c_1,...,c_n,m_1,...,m_n)$

[0072] 本发明中环签名验证方法为：

算法 3 环签名验证算法

输入:环签名 $\sigma=(I,c_1,...,c_n,m_1,...,m_n)$,交易公钥 R_τ .

输出: 环签名验证结果.

1: **for** $i=0,...,n$ **do**

2: $Node_j$ 计算 $L_i'=m_iG+c_iPK_i$

3: $Node_j$ 计算 $M_i'=m_i\mathcal{H}_p(PK_i)+c_iI$

[0073]

4: **end for**

5: **if** $\sum_{i=0}^{n} c_i=\mathcal{H}_s(R_\tau,L_0',...,L_n',M_0',...,M_n') \mod l$ **then**

6: 返回 1

7: **else**

8: 返回 ⊥

9: **end if**

[0074] 有益效果:与现有技术相比,本发明具有以下优点：

[0075] 1、本发明将分布式存储技术与区块链技术相结合,在实现安全有效的分布式云存

储的同时，将文件所有权处理过程作为区块链交易过程的处理，并进一步实现分布式存储中的隐私保护，包括交易发送方、接收方以及交易内容的隐私保护，保证了只有真正的接收者才能精确定位到该笔交易，进而获取文件所有权，并进一步的进行文件处理过程(文件分享、下载、删除等操作)。

[0076] 2、本发明所述方案可有效抵制数据伪造攻击：在本发明的系统模型中，恶意用户可能对系统发起数据伪造攻击，即没有文件所有权的用户试图通过向系统发送不属于他的交易，来骗取文件所有权。为应对数据伪造攻击，本发明设计一种所有权验证方案，保证了只有拥有正确标准地址的授权用户，才能完成所有权认证。由于用户的标准地址只对矿工和交易双方公开，因而可以抵制数据伪造攻击。

[0077] 3、本发明所述方案可有效抵制重放攻击：在本发明的系统模型中，恶意用户可能对系统发起重放攻击，即被取消文件所有权的用户，可能使用其原有的交易来向系统证明其所有权。为应对重放攻击，本发明的方案中使用交易失效列表Blacklist来记录失效交易。当用户被取消文件所有权时，对应的交易会被添加到交易失效列表中。进一步的，在进行所有权验证之前，系统首先检查该交易是否在交易失效列表中，如果交易存在于失效列表中，则无法通过所有权验证。因而可以抵制重放攻击。

附图说明

[0078] 图1是本发明实施例中的系统结构图；

[0079] 图2是本发明实施例中的区块结构图；

[0080] 图3是本发明实施例中的交易结构图；

[0081] 图4是本发明实施例中的交易生成示意图；

[0082] 图5是本发明实施例中的交易查找示意图；

[0083] 图6是实施例中不同环大小情况下，环签名生成和环签名验证的时间开销关系图。

具体实施方式

[0084] 下面对本发明技术方案进行详细说明，但是本发明的保护范围不局限于所述实施例。

[0085] 如图1所示，本发明的一种分布式云存储系统的隐私保护方法，所述分布式云存储系统包括用户、矿工和存储提供方；用户包括数据所有者μ_i和数据使用者μ_j，数据所有者μ_i将数据外包给存储提供方，进行下载和删除操作，同时数据所有者μ_i可以将文件分享给数据使用者μ_j；矿工和存储提供方为用户提供分布式云存储服务，其中，矿工负责打包区块和维护区块，存储提供方将自己空闲的硬盘空间出租给网络组成分布式的的存储空间；另外，矿工和存储提供方通过提供服务获得代币奖励；

[0086] 隐私保护方法具体包括以下步骤：

[0087] 步骤一、系统初始化：输入公共参数(l,G)，l为基点的素数阶，G为椭圆曲线的基点；用户选取随机数a∈[1,l-1]，b∈[1,l-1]组成私钥对(a,b)，a≠b；同时，用户计算A＝aG，B＝bG，作为公钥对(A,B)；另外，用户计算ripemd160(sha256(A,B))作为其标准地址，同时也作为其ID；

[0088] $l=2^{252}+27742317777372353535851937790883648493$；

[0089] 步骤二、文件上传，即数据所有者μ_i要将文件F_i上传到分布式存储系统；步骤(2)的详细过程如图4所示：

[0090] (2.1)数据所有者μ_i选取哈希函数$H:\{0,1\}^* \to \mathbb{Z}_p$，例如SHA-256，并计算文件$F_i$的哈希值$hash_i=H(F_i)$，然后数据所有者$\mu_i$向系统广播上传请求：

[0091] $\mu_i \to *: upload, hash_i, ID_{\mu_i}$

[0092] 其中，$hash_i$为文件F_i的哈希值，ID_{μ_i}为数据所有者μ_i的标准地址；

[0093] (2.2)当系统接收到数据所有者μ_i的上传请求，系统中的矿工开始执行工作量证明算法POW来争取记账权(即交易打包权)；执行完POW算法后，假定矿工$Node_j$获取到记账权；然后矿工$Node_j$通知μ_i开始上传文件：

[0094] $Node_j \to \mu_i$:upload,$hash_i$

[0095] (2.3)当数据所有者μ_i接收到上传指令，μ_i在客户端侧使用同态加密算法AES256-CTR加密文件F_i：$F_i'=Enc(F_i)$，其中加密密钥为文件哈希$hash_i$；然后，μ_i在客户端侧对加密后的文件F_i'进行分块得到文件碎片$\{shard_1, shard_2, ..., shard_n\}$，其中每个碎片大小为8M，不足8M的空间用0填充，如图2所示；接着数据所有者μ_i求取各碎片哈希$\{hash_{sh1}, hash_{sh2}, ..., hash_{shn}\}$并建立Merkle Tree，用于文件审计；最后，μ_i将文件碎片分散地存储在系统中，并在分布式哈希表(DHT)中生成文件索引；同时，矿工$Node_j$将文件元数据(Merkle根、文件哈希值等)打包进区块，其中$metadata=\{hash_i, MerkleRoot_i\}$；矿工$Node_j$通知数据所有者$\mu_i$开始打包交易：

[0096] $Node_j \to \mu_i$:transaction,$hash_i$

[0097] (2.4)当数据所有者μ_i接收到交易打包指令，开始生成上传交易Tx_k，上传交易中交易发送方和接收方都是数据所有者μ_i。

[0098] 如图3所示，步骤(2.4)中交易生成的具体过程如下：

[0099] (2.4.1)μ_i选取随机数$r_k \in [1, l-1]$；然后μ_i计算隐匿地址$P_k=\mathcal{H}_s(r_k A_{\mu_i})G+B_{\mu_i}$，交易公钥$R_k=r_kG$，其中$(A_{\mu_i}, B_{\mu_i})$为$\mu_i$的公钥对，$\mathcal{H}_s:\{0,1\} \to \mathbb{F}_q$为密码散列函数；

[0100] (2.4.2)μ_i选取随机数$x_k \in [1, l-1]$作为私钥，并计算对应的公钥$PK_k=x_kG$，密钥镜像$I_k=x_k\mathcal{H}_p(PK_k)$，其中，$\mathcal{H}_p: E(\mathbb{F}_q) \to E(\mathbb{F}_q)$为确定性哈希函数；

[0101] (2.4.3)μ_i使用对称加密算法计算$Enc_{ID_{\mu_i}}(R, hash_i, MAC_k)$作为交易内容$Proof_k$，用于授予交易接收方文件所有权，并可用于文件所有权验证，其中$MAC_k=MAC_{ID_{\mu_i}}(R, hash_i)$，R为随机数，$ID_{\mu_i}$为$\mu_i$的标准地址，同时作为对称加密密钥；

[0102] (2.4.4)μ_i将R_k，P_k，I_k以及$Proof_k$打包进交易Tx_k，然后μ_i计算环签名σ_k，对交易Tx_k进行签名并发送到网络；网络中任何一方都可以验证交易签名，并且不会泄露交易发送方；

[0103] (2.4.5)矿工节点$Node_j$验证交易并将交易打包进区块；

[0104] 其中，数据所有者μ_i作为交易接收方向矿工节点$Node_j$发送交易Tx_k信息P_k，I_k，$Proof_k$来验证μ_i对于文件F_i的所有权，并用于消费(分享、删除、下载操作)。

[0105] 步骤三、文件删除，即数据所有者μ_i要删除其对于文件F_i的所有权，详细过程如下：

[0106] (3.1)首先，数据所有者μ_i向系统广播删除请求：

[0107] $\mu_i \rightarrow *: delete, hash_i, P_k, ID_{\mu_i}$，

[0108] 其中，$hash_i$为文件F_i哈希；P_k是能够证明μ_i对文件F_i的所有权的交易的隐匿地址，ID_{μ_i}为数据所有者μ_i标准地址；

[0109] (3.2)当系统接收到μ_i的删除请求，系统中的矿工开始执行工作量证明算法POW来争取记账权；执行完POW算法后，假定矿工节点$Node_j$获取到记账权；

[0110] (3.3)矿工节点$Node_j$验证μ_i对文件F_i的所有权；

[0111] (3.4)当数据所有者μ_i通过所有权验证以后，矿工节点$Node_j$从地址为P_k的交易获取其密钥镜像I_k，并将I_k添加到交易失效名单Blacklist，并向全网广播；网络中的矿工节点接收到广播以后更新交易失效名单Blacklist；

[0112] 上述所有权验证方法为：

[0113]

输入:文件哈希 $hash_i$， 交易地址 P_k， 数据所有者 μ_i 标准地址 ID_{μ_i}.

输出:所有权验证结果.

1: $Node_j$ 根据地址 P_k 找到对应交易 Tx_k， 获取交易内容 $\Pr oof_k$， 密钥镜像 I_k；

2: **if** $I_k \notin Blacklist$ **then**

3: $Node_j$ 使用解密函数计算： $\Pr oof_k' = Dec_{ID_{\mu_i}}(\Pr oof_k) = \{R_k', hash_i', MAC_k'\}$；

[0114]

4: $Node_j$ 计算 $MAC_k' = MAC_{ID_{\mu_i}}(R_k', hash_i')$；

5: **if** $MAC_k^* = MAC_k'$ **then**

6: 返回 1;

7: **else**

8: 返回⊥;

9: **end if**

10: **else**

11: 返回⊥.

12: **end if**

[0115] 步骤四、文件分享，即文件F_i的所有者μ_i授予数据使用者μ_j文件F_i的所有权，详细过程如下：

[0116] (4.1)首先，数据所有者μ_i向系统广播分享请求：

[0117] $\mu_i \rightarrow *: share, hash_i, P_k, ID_{\mu_i}, ID_{\mu_j}$，

[0118] 其中，$hash_i$为文件F_i哈希；P_k是能够证明μ_i对文件F_i的所有权的交易的隐匿地址；ID_{μ_i}, ID_{μ_j}分别为数据所有者μ_i和数据使用者μ_j的标准地址；

[0119] (4.2)当系统接收到μ_i的分享请求，系统中的矿工开始执行工作量证明算法POW来争取记账权；执行完POW算法后，假定矿工节点$Node_j$获取到记账权；

[0120] (4.3)矿工节点$Node_j$验证μ_i对文件F_i的所有权；

[0121] (4.4)μ_i通过所有权验证以后，$Node_j$向μ_i发送分享指令：

[0122] $Node_j \rightarrow \mu_i : share, hash_i, ID_{\mu_j}$。

[0123] (4.5)当μ_i接收到指令，开始生成交易Tx_τ，其中，μ_i作为交易发送方，μ_j作为交易接收方；

[0124] (4.6)如图5所示，μ_j作为交易接收方，检查每一个区块中的交易，提取交易中的隐匿地址P和交易公钥R′，使用自己的私钥对(a_{μ_j}, b_{μ_j})计算$P' = \mathcal{H}_s(a_{\mu_j} R')G + b_{\mu_j} G$，判断$P' \stackrel{?}{=} P$，其中，$\mathcal{H}_s : \{0,1\} \rightarrow \mathbb{F}_q$为密码散列函数；由于$a_{\mu_j} R' = (a_{\mu_j} r')G = r' A_{\mu_j}$，$b_{\mu_j} G = B_{\mu_j}$，如果$\mu_j$为该交易的接收方，则等式成立；反之，不成立；

[0125] (4.7)当μ_j找到交易Tx_τ之后，μ_j计算$x_\tau = \mathcal{H}_s(a_{\mu_j} R_\tau) + b_{\mu_j}$；$\mu_j$使用$x_\tau$恢复交易$Tx_\tau$，并在消费交易(分享、下载、删除)$Tx_\tau$时使用$x_\tau$作为交易私钥。

[0126] 上述步骤(4.5)中交易生成的方法为：

[0127] (4.5.1)交易发送方μ_i选取随机数$r_\tau \in [1, l-1]$，并获取交易接收方μ_j的公钥对(A_{μ_j}, B_{μ_j})；然后μ_i计算隐匿地址$P_\tau = \mathcal{H}_s(r_\tau A_{\mu_j})G + B_{\mu_j}$，交易公钥$R_\tau = r_\tau G$，其中，$\mathcal{H}_s : \{0,1\} \rightarrow \mathbb{F}_q$为密码散列函数；

[0128] (4.5.2)交易发送方μ_i使用对称加密算法计算$Enc_{ID_{\mu_j}}(R, hash_i, MAC_\tau)$作为交易内容$Proof_\tau$，用于授予交易接收方文件所有权，并可用于文件所有权验证，其中$MAC_\tau = MAC_{ID_{\mu_j}}(R, hash_i)$，R为随机数，加密密钥为$\mu_j$的标准地址$ID_{\mu_j}$；

[0129] (4.5.3)交易发送方μ_i选取随机数$x_\tau \in [1, l-1]$，同时也作为签名私钥，并计算对应的公钥$PK_\tau = x_\tau G$，密钥镜像$I_\tau = x_\tau \mathcal{H}_p(PK_\tau)$，其中，$\mathcal{H}_p : E(\mathbb{F}_q) \rightarrow E(\mathbb{F}_q)$为确定性哈希函数；

[0130] (4.5.4)交易发送方μ_i将R_τ，P_τ，I_τ以及$Proof_\tau$打包进交易Tx_τ，且μ_i计算环签名σ_τ，对交易Tx_τ进行签名并发送到网络，网络中任何一方都可以验证交易签名，并且不会泄露交易发送方；

[0131] (4.5.5)矿工节点$Node_j$验证交易并将交易打包进区块。

[0132] 步骤五、文件下载的详细过程如下：

[0133] (5.1)数据所有者μ_i向系统广播下载请求：

[0134] $\mu_i \rightarrow * : download, hash_i, P_k, ID_{\mu_i}$，

[0135] 其中，$hash_i$为文件F_i的哈希；P_k为能够证明μ_i对文件F_i的所有权的交易的隐匿地址，ID_{μ_i}为数据所有者μ_i标准地址；

[0136] (5.2)当系统接收到μ_i的下载请求，系统中的矿工开始执行工作量证明算法(POW)来争取记账权；执行完POW算法后，假定矿工节点$Node_j$获取到记账权；

[0137] (5.3)矿工节点$Node_j$验证μ_i对文件F_i的所有权；

[0138] (5.4)μ_i通过所有权验证以后,$Node_j$在DHT网络中查询文件实际存储地址$addr_i$,并将下载地址$addr_i$返回给μ_i:

[0139] $Node_j \rightarrow \mu_i: hash_i, addr_i$;

[0140] (5.5)μ_i使用下载工具将文件从地址$addr_i$恢复到本地。

[0141] 实施例

[0142] 为评估本发明的性能表现,以如下实施例测量部分算法的时间开销。

[0143] 为评估系统开销,实施过程采用基于CryptoNote协议的开源DigitalNote平台,并且根据系统需求,完成相关算法实现。为了评估系统开销,在实施过程中在本地部署测试链实验环境,其中,本地部署了三个矿工节点以及三个钱包终端进行实验,并对系统主要步骤的主要算法的时间开销进行分析,并进行时间开销的测量。其中,T_{File}为客户端文件处理的时间开销,包括文件哈希操作时间、文件分块操作时间、文件分块上传时间;T_{Tx}为用户侧生成交易操作的时间,包括计算目的地址操作的时间、计算环签名操作的时间、发送交易操作的时间;另外,T_{del},T_{chk},T_{POW},T_{DHT}分别为删除交易操作的时间、交易验证操作的时间、文件所有权验证操作的时间以及DHT查询时间。对各阶段时间开销分析总结见表1。

[0144] 表1系统时间开销分析表

[0145]

阶段	实体	计算开销
文件上传	数据所有者 μ_i	$T_{File}+T_{Tx}$
	矿工 $Node_j$	T_{chk}

[0146]

文件删除	矿工 $Node_j$	T_{del}
文件分享	数据所有者 μ_i	T_{Tx}
	矿工 $Node_j$	$T_{chk}+T_{POW}$
文件下载	矿工 $Node_j$	$T_{chk}+T_{DHT}$

[0147] 另外,本实施例的算法时间开销如表2所示。

[0148] 表2主要算法时间开销

[0149]

算法	时间开销
生成目标地址	36ms
发送交易	31ms
创建交易	0.0093ms
查找交易	22ms
生成密钥镜像	0.164ms

[0150] 另外,如图6所示,为不同环大小情况下,环签名生成时间和环签名验证时间。

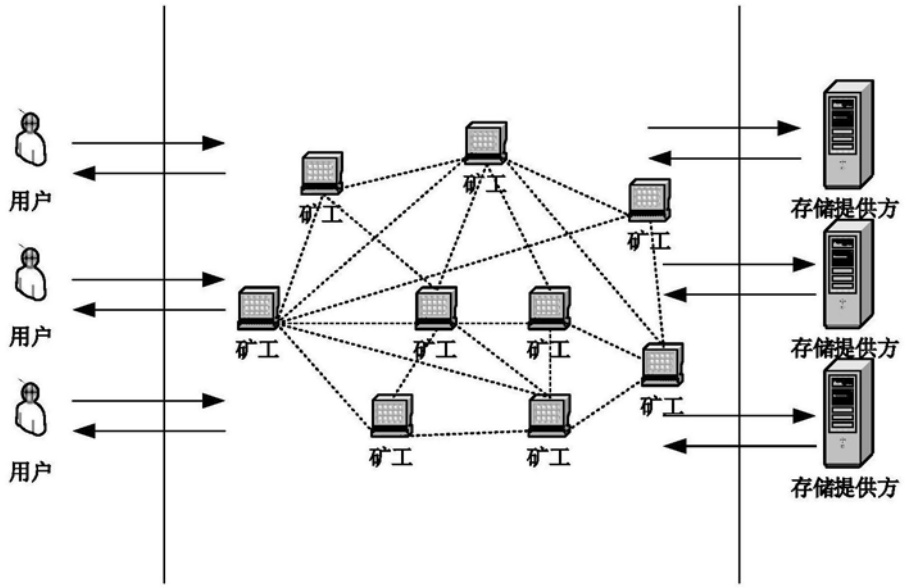

用户
用户
用户
矿工
矿工
矿工
矿工
矿工
矿工
矿工
矿工
矿工
存储提供方
存储提供方
存储提供方

图1

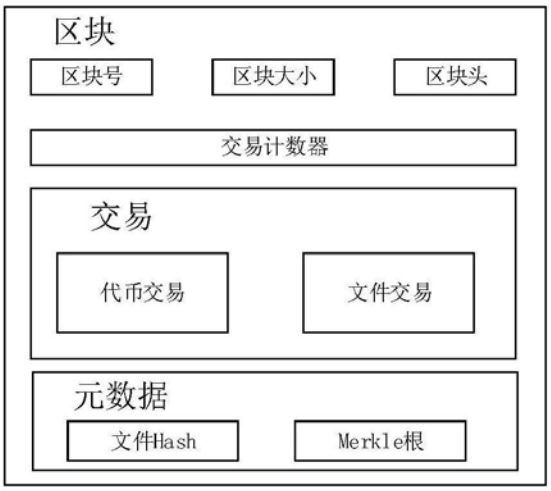

区块
区块号
区块大小
区块头
交易计数器
交易
代币交易
文件交易
元数据
文件Hash
Merkle根

图2

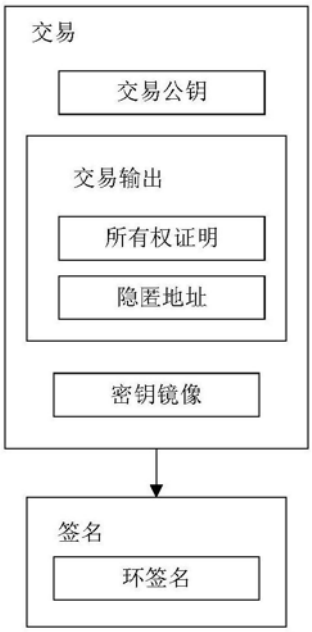

交易
交易公钥
交易输出
所有权证明
隐匿地址
密钥镜像
签名
环签名

图3

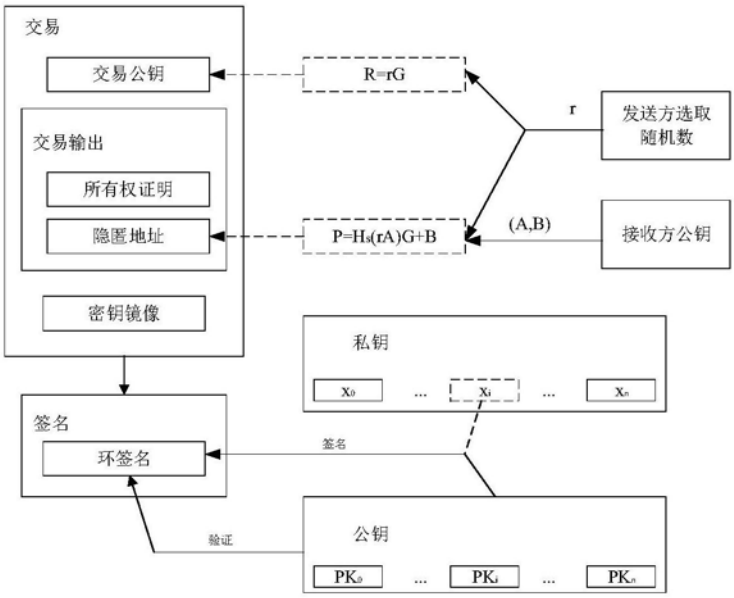

交易
交易公钥
交易输出
所有权证明
隐匿地址
密钥镜像
R=rG
r
发送方选取
随机数
P=Hs(rA)G+B
(A,B)
接收方公钥
私钥
X0 ... Xi ... Xn
签名
环签名
签名
验证
公钥
PK0 ... PKi ... PKn

图4

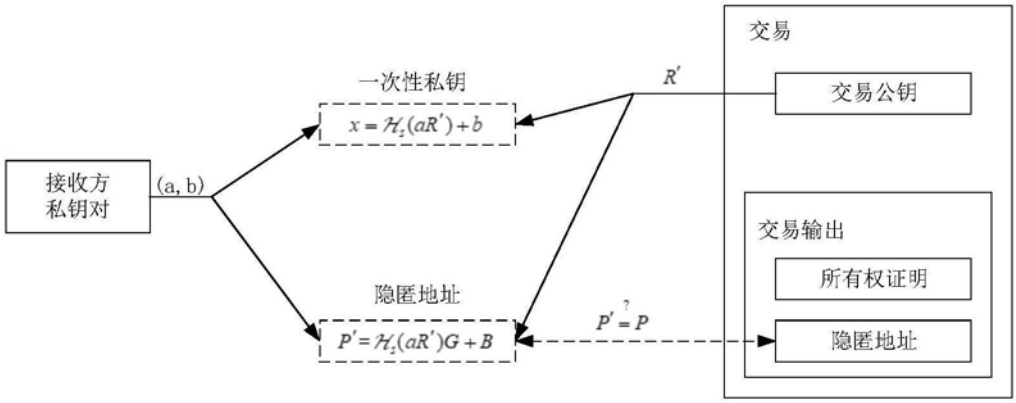

交易
一次性私钥
$x = \mathcal{H}_2(aR') + b$
R'
交易公钥
接收方
私钥对
(a, b)
交易输出
所有权证明
隐匿地址
$P' = \mathcal{H}_2(aR')G + B$
$P' \overset{?}{=} P$
隐匿地址

图5

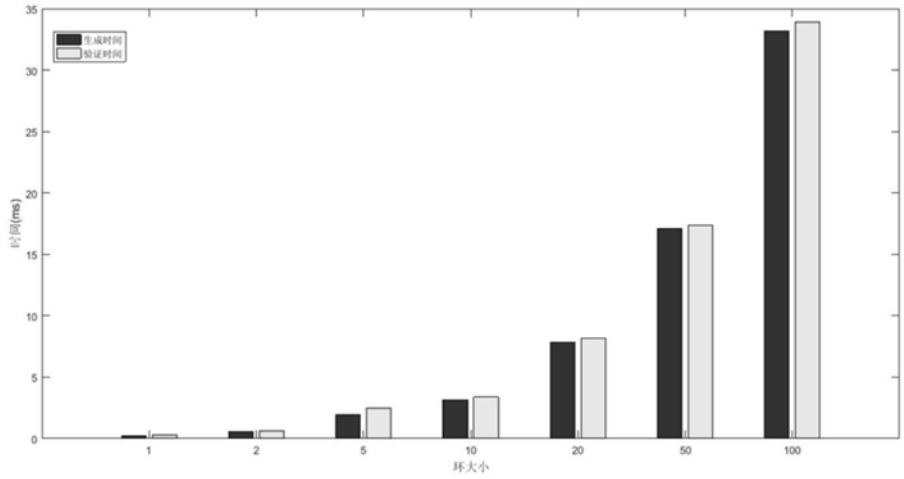

生成时间
验证时间
时间(ms)
0
5
10
15
20
25
30
35
1
2
5
10
20
50
100
环大小

图6