四 川 师 范 大 学

本科毕业设计（论文）

基于区块链的云文件安全共享方法研究

学生姓名 邓杰

院系名称 计算机科学学院

专业名称 网络工程

班　　级 2 班

学　　号 2018110109

指导老师 冯朝胜

完成时间 2022 年 4 月 20 日

摘 要

在数据安全共享方案中，密文策略基于属性加密 (CP-ABE) 是解决数据隐私问题，并提供“一对多”数据共享的有效手段。传统的 CP-ABE 方案将共享数据密文外包给云服务器中，然而云服务器是一个不完全可信的实体，因此这种方案存在着访问权限不受限、数据隐私泄露等问题。随着分布式技术的发展，区块链和星际文件系统 (IPFS) 步入大众视野，并为去中心化存储提供了新思路。

鉴于上述问题，本文提出了一个基于区块链的云文件安全共享方案。数据所有者为共享数据指定访问策略，在加密时将密文与访问策略链接，并将密文存储于星际文件存储系统，提升共享数据隐私性；将共享数据元信息密文存储在区块链，便于用户查询和验证数据完整性；通过在以太坊上部署智能合约并设计合约函数，用户可以使用关键词密文对共享数据密文进行检索。安全性分析和实验分析表明，该方案在保护数据隐私的同时，提升了系统可靠性。

关键词: 属性基加密；以太坊；智能合约；IPFS；访问控制

Abstract

Ciphertext-policy attribute-based encryption(CP-ABE) is an effective technology to achieve ”one-to-many” encryption/decryption, solve the data privacy issues and fine-grained access control in a data security sharing solution. In traditional CP-ABE solutions, the ciphertext is stored in the cloud. However, cloud sever is not a fully trusted entity, so this scheme is faced with problems such as data privacy leakage and unrestricted access rights. With the development of distributed technology, blockchain and IPFS have entered the public view and provide a new idea for decentralized storage.

For the above problems, we propose a blockchain-based cloud file security sharing scheme. In this scheme, data owner can encrypt shared data by specifying a access policy and store it to IPFS which can improve data availability; store encrypted metadata information on the blockchain and user can access it and verify its integrity; deploy a smart contract that implements a encrypted keyword search function on shared data to ethereum. Finally, the experimental results and security analysis show that our scheme provide a reliable and feasible environment.

Keywords:ABE; Ethereum; smart contract; IPFS; access control

目 录

第 1 章 绪 论

1.1 研究背景和意义

当前已步入大数据时代，数据成为互联网的主要资产。许多电子设备如智能手机、智能手表、电脑每天都会生成大量的数据，存储这些数据需要巨大的存储空间和资源。云存储系统具有分布式的数据中心，服务器使用虚拟化技术协同工作并为使用者提供了大量的存储资源。由于云存储系统的便利性和高效性，吸引了个体、企业以及其它组织的广泛关注。许多企业将和组织将数据外包给云存储系统以减轻设备的存储负担[1]。对比本地存储方案，云存储系统具有非常多的优点例如存储开销低、灵活性高、自动更新、灾难容忍等[2-5]。此外，可以给使用者提供按需、弹性的存取服务以及便利的数据访问服务；支持用户随时随地访问数据并提供按需和按量多种付费模式。借助云存储系统，用户可以根据需要租赁和支付存储以及计算费用。然而，若云服务器遭受攻击，存储在云端的数据将面临泄露风险。现有的云存储系统以分布式的方式将数据存储在多个数据中心或者是多台服务器，但不同数据中心存储的数据高度冗余，此外还存在中心化存储面临服务不可用、明文数据面临泄露隐私等风险[6]。

因此，将数据进行加密后上传到云端是保护用户隐私的有效途径。Bethencourt 等人[7] 于 2007 年提出了支持树状结构的通用组模型的密文策略基于属性的加密 (CP-ABE)，解决了密文细粒度共享问题。目前，CP-ABE 广泛用于云数据共享。在 CP-ABE 算法中，先由数据所有者为共享数据构造一个访问策略，在加密过程中将密文链接到访问策略，并在密钥生成过程中将密钥链接到用户属性，在解密过程中只有密钥所链接的用户属性满足密文所链接的访问策略时才能从密文中恢复明文。为提高现有应用的性能，可以将星际文件系统 (IPFS)、区块链、CP-ABE 结合，提出一个基于区块链的云文件安全共享方案，为企业和用户提供安全、可靠的云存储服务。

区块链本质上是一个由链式数据块组成的分布式数据库[8]。其存储了所有的历史交易记录，交易经确认后无法更改。其不可篡改的特性由区块链系统本身保证，而与具体操作无关。因此，区块链的使用非常简单，并且与其它安全技术相比，其更加的稳定。例如文献 [9] 将区块链技术用于所有权管理以及产权追踪。文献 [10] 使用区块链处理电子医疗记录，实现了安全、不可篡改的数据共享。在云存储架构中使用区块链可以给用户提供一个更加安全的环境。

1.2 国内外研究现状

1.2.1 ABE 相关研究

许多研究者在提升加密安全性、服务质量参数、密钥管理等方面开展了大量研究工作。但传统密码学方案不能提供灵活的访问控制，文献 [11] 在文献 [12] 提出的基于身份加密 (IBE) 的基础上，进一步设计出基于属性的加密 (ABE) 算法。在文献 [11]

所提方案中，将用户密钥与属性 w 链接，将密文与属性 w' 链接，只有当 w' 与 w 的交集大于某个值 d 时才能得到从密文中恢复明文。ABE 算法又可以分为两类：密钥策略基于属性加密 (KP-ABE)[13] 和密文策略基于属性加密 (CP-ABE)[7]。KP-ABE 算法在加密过程中将密文与用户属性链接，在密钥生成过程中将用户密钥与用户属性链接。CP-ABE 算法在加密过程中将密文与访问策略链接，在密钥生成过程中将用户密钥与用户属性链接。文献 [14] 提出了一种匿名 CP-ABE 方案。其支持多个属性值但系统参数较短。文献 [15] 提出了一种在访问策略没有通配符的情况下，密文大小不会随着属性数量的增加而发生变化的 KP-ABE 方案。文献 [16] 使用虚拟属性技术并对 KP-ABE 重新解释，提出了一个从 KP-ABE 到 CP-ABE 转变的方案。在许多商业应用场景中，属性分发和管理需要多个属性权限，文献 [17] 提出多权限基于属性加密。此外，随着移动设备的广泛使用，文献 [18] 提出一种针对计算和存储资源有限的设备的外包 ABE 方案。

1.2.2 区块链相关研究

近年来，去中心化加密货币 (例如 Bitcoin[19]，Ethereum[20]，Zcash[21] 等) 十分火热，其底层所使用的区块链技术备受关注。如今，区块链在去中心化供应链、基于身份的 PKI、去中心化物联网、去中心化存储等领域发挥着广泛应用。文献 [22] 使用比特币将数据与其数据所有者相关联，将元数据存储在区块链上，以供公众查询和恢复。文献 [23] 将区块链应用于多服务系统架构，为用户提供保护隐私的认证机制和高效的撤销过程。文献 [24] 将区块链与无许可频谱接入技术结合，提高了非实时数据传输的安全性。文献 [25] 针对智能城市环境中的车载导航系统，提出了一种基于区块链的智能交通系统隐私保护模型。文献 [26] 将区块链应用于医疗系统，使用基于属性的签名方案来保护用户隐私。

1.3 主要贡献

本文提出了一个基于区块链的云文件安全共享方案。该方案使用了密文策略属性基加密，以太坊，星际文件存储系统等技术。数据所有者将访问策略链接到密文并将用户属性链接到用户密钥，实现灵活的数据共享。此外，将元数据的密文信息存储在以太坊上，借助区块链来验证数据的完整性。通过对共享数据建立关键词密文索引，并将索引信息存储在智能合约，为用户提供了共享数据密文搜索功能。在 Ubuntu 系统下，我们使用以太坊官方测试网络 Ropsten 对该系统方案进行了实验，对相应的性能以及开销进行了分析。

1.4 论文结构

本文主要分为七个部分，全文的结构安排如下：

第一部分是绪论部分，介绍了去中心化存储方案的背景和意义。

第二部分介绍了基于属性加密技术，概述了区块链和星际文件系统的基本原理。

第三部分介绍了基于区块链的云文件安全共享方案的整体框架。

第四部分详细介绍了基于区块链的云文件安全共享方案中算法的具体实现。

第五部分测量了方案开销和性能，并对方案的安全性进行了评估和分析。

第六部分对所提方案进行了总结，概述了方案的不足并对未来研究工作进行讨论。

第 2 章 预备知识

2.1 基于属性加密

2.1.1 双线性映射

设 p 是一个大素数，G_0 是椭圆曲线上的 p 阶乘法循环群，G_t 是阶为 p 的乘法循环群，g 为 G_0 的生成元。如果映射 $e: G_0 \times G_0 \to G_t$ 满足以下三个条件，则称映射 e 为一个双线性映射：

1. 双线性：对于任意元素 $x, y \in G_0$ 以及 $m, n \in Z_p$，有 $e(x^m, y^n) = e(x, y)^{mn}$。
2. 非退化性：至少存在一个元素 $x \in G_0$，使得 $e(x, x) \neq 1$。
3. 可计算性：对于任意元素 $x, y \in G_0$，$e(x, y)$ 在计算上是可行的。

2.1.2 访问树

访问策略可以使用一棵树来表示，如图2-1所示，可以将其称为访问树。在本文中所提及的访问策略，如果没有其它特殊说明，都是指访问树。在树中，将每个内部节点 (非叶节点) 看作一个阈值门限。如果某属性集 S 达到节点所规定的阈值，则称属性集 S 满足以该节点为根节点的子树。对于每一个内部节点 t，t 的阈值门限由两部分组成，分别是 k_t 和 num_t。第一部分 k_t 规定了节点 t 的阈值，任何满足以该节点为根节点的子树的属性 S 都必须达到所规定的阈值。第二部分 num_t 记录了节点 t 的孩子节点个数。对于任何一个内部节点，其阈值不能小于 0 并且不能超过其孩子节点个数 $1 \leq k_t \leq num_t$。当阈值为 1 时 $k_t = 1$，可以将阈值门限理解一个 OR 门。当阈值等于孩子节点数量时 $k_t = num_t$，可以将阈值门限理解一个 AND 门。对于树中的每个非内部节点 t，它们没有孩子节点 $num_t = 0$，它们的阈值 k_t 都被设置为 1。此外，每个非内部节点包都含一个属性。如果属性集 S 要达到非内部节点所规定的阈值，则 S 必须包含该节点所包含的属性。为了便于描述访问树中的操作，定义了几个函数以简化描述。对于节点 t，规定函数 $parent(t)$ 返回节点 t 的父节点，函数 $att(t)$ 返回非内部节点 t 所包含的属性。访问树中每个内部节点 t 会为每个子节点 z 分配一个 1 到 num_t 的序号用以表示子节点 z 的顺序，函数 $index(z)$ 返回节点 t 对子节点 z 分配的序号。

假设 $\mathcal{T}$ 为数据所有者指定的一访问策略。对于 $\mathcal{T}$ 中的任意节点 t，$\mathcal{T}_t$ 为 $\mathcal{T}$ 中的一棵子树，其根节点为 t。如果属性集 γ 达到节点 t 所规定的阈值，则称 γ 满足子树 $\mathcal{T}_t$，表示为 $\mathcal{T}_t(\gamma) = 1$。对于 $\mathcal{T}$ 中的非内部节点 t，只有当 $att(t) \in \gamma$ 时，则称属性 γ 满足以 t 为根节点的子树，表示为 $\mathcal{T}_t(\gamma) = 1$。对于 $\mathcal{T}$ 中的内部节点 t，使用 z 表示节点 t 的每一个子节点，对节点 z 递归执行 $\mathcal{T}_z(\gamma)$，只有当 γ 至少满足 k_t 个以 z 为根节点的子树时，则称属性 γ 满足以 t 为根节点的子树，表示为 $\mathcal{T}_t(\gamma) = 1$。图2-1展示了一棵访问树。叶子节点的属性分别是 General，Army，Op-X，Op-Y，Op-Z。内部节点如

2-of-3 表示该节点的阈值为 2，孩子节点个数为 3，因此用户属性至少要同时包含该节点的两个孩子节点的属性才能通过该节点的认证。例如拥有属性集 General，Army，Op-Y，Op-Z 的用户满足访问树要求，而拥有属性集 Army，Op-Y，Op-Z 则不满足访问树的要求。

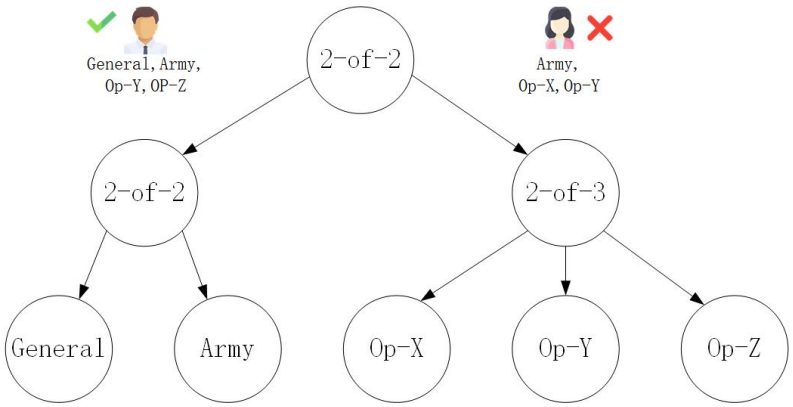

图 2-1 访问树

2.2 区块链概述

2.2.1 区块链技术

区块链是加密货币 Bitcoin 最核心的技术，最早由 Nakamoto 在 2008 年提出[19]。区块链是由网络中所有节点集体参与维护的一个包含所有历史交易记录的去中心化账本。在整个区块链网络中，不存在中心实体，单节点无法操控整个网络，这意味着网络中所有节点地位平等。图2-2描绘了区块链结构。每个区块都包含 Previous Hash、Nonce、Merkle Root Hash、Timestamp 等字段。Previous Hash 为前一个区块的哈希，通过 Previous Hash 可以将所有的区块连接在一起。Nonce 值是一个随机值，用于挖矿节点的工作量证明。Timestamp 为矿工挖矿时设置的时间戳。Merkle root 为所有交易的根哈希值，用于验证当前区块的所有交易。

2.2.2 以太坊

以太坊[20] 由 Bitcoin 发展而来，使用工作量证明 (PoW) 以及幽灵 (Ghost) 协议来解决节点之间的不一致性问题。智能合约是运行在太坊上的由一组规则组成的计算机程序。以太坊网络中所有节点都会独立执行智能合约，但这些节点最终必须在某个状态上达成一致。在以太坊中，每个节点都会维护自己的一个状态，成功执行一个交易会将状态 s_t 转化为 s_{t+1}。以太坊使用 Ether 作为货币，最小单位为 wei，1Ether 等于 10^{18}wei。

在以太坊网络上执行的每一个操作都需要消耗带宽、计算和内存资源，这些资源消耗的总量以 gas 为单位进行衡量。发送一笔交易可以触发区块链上的一系列操作，因此对于一笔交易消耗的总 gas 值是未知的。为了防止拒绝服务攻击以及恶意循环操作，交易的发送者必须为他们想执行的每一个操作而支付一定量的 gas。一个交易的

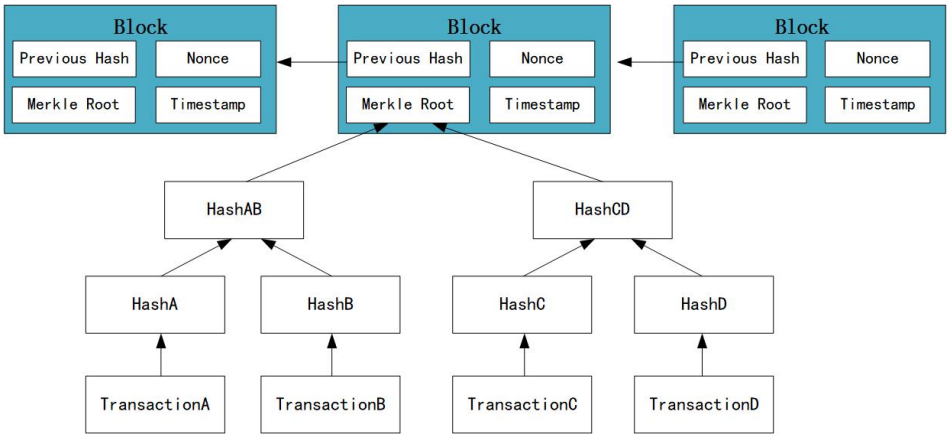

图 2-2 区块链结构

总共开销计算如下 $GasFee = GasUsed \times GasPrice$。其中 $GasFee$ 为用户需要支付的 gas 费用，$GasUsed$ 为交易所使用的 gas 开销，$GasPrice$ 为 gas 的价格。

2.3 星际文件系统

在所提方案中，使用星际文件系统 (IPFS)[27] 作为云存储平台。IPFS 是一个使用点对点通信的基于版本控制的数据存储协议，它综合了许多以往成功系统的优点，为用户提供高完整性和弹性的数据存储。IPFS 使用分布式哈希表 (DHT) 进行网络路由的寻址。当文件被上传到 IPFS，其会被分割为不同块。每个块中不仅含有文件数据还包含指向其它块的链接，但一个块最多可存储 256KB 的数据 (含链接信息)。每一个块由一个哈希值标识，由于该哈希值是通过块的内容计算得来，因此称为内容标识。块中的链接信息包含其它块的内容标识符，所有块的链接共同组成了一个默克尔有向无环图，文件使用默克尔有向无环图的根哈希值作为标识符。本文所提方案将共享数据加密后存储到 IPFS 中，而将元数据密文信息存储到以太坊。只有当密钥生成中用户密钥所链接的用户属性满足加密时密文所链接的访问策略，用户才可以从以太坊中读取元数据密文并解密得到文件标识符，进而下载文件密文最终将其解密。

第 3 章 方案设计

在本章中，我们对所提方案的整体框架进行了详细描述，并在表3-1列出了主要符号及其定义。

符号	描述
DO	数据所有者
DU	数据用户
AA	属性授权机构
PK	系统公钥
MK	系统主密钥
$HMAC$	密钥散列函数
MAC	消息认证码
TX	交易
$h_{location}$	文件位置
kw	文件关键词
CT_F	文件密文
CT_K	密钥 K 的密文
CT_{K_1}	密钥 K_1 的密文
CT_{txid}	交易 ID 的密文
CT_l	文件标识密文
CT_{md}	文件元数据密文
$index$	关键词密文索引
$token$	用于搜索的 $token$
Enc_K	AES 加密,K 为对称密钥
Dec_K	AES 解密,K 为对称密钥

表 3-1 符号含义

3.1 系统概述

3.1.1 组成实体

本文所提方案系统架构如图3-3所示，该方案包含如下五个实体：

1. DO(Data Owner，数据所有者)：拥有数据的个体、机构或其它组织。主要负责上传共享文件、指定加密所需的访问策略、部署合约。
2. DU(Data User，用户)：数据的使用者。当用户属性满足与密文所链接的策略时，用户可以解密得到数据元信息，进而获得文件明文。
3. AA(Attribute Authority，属性授权机构)：负责初始化整个系统，为用户提供注册服务并为数据所有者生成密钥。
4. 区块链：保存元数据密文信息。通过智能合约为 DO 和 DU 提供相应服务如关键字搜索、添加用户、添加关键字等。
5. IPFS：存储 DO 上传的共享数据密文，DU 可以使用文件标识下载共享文件密文。

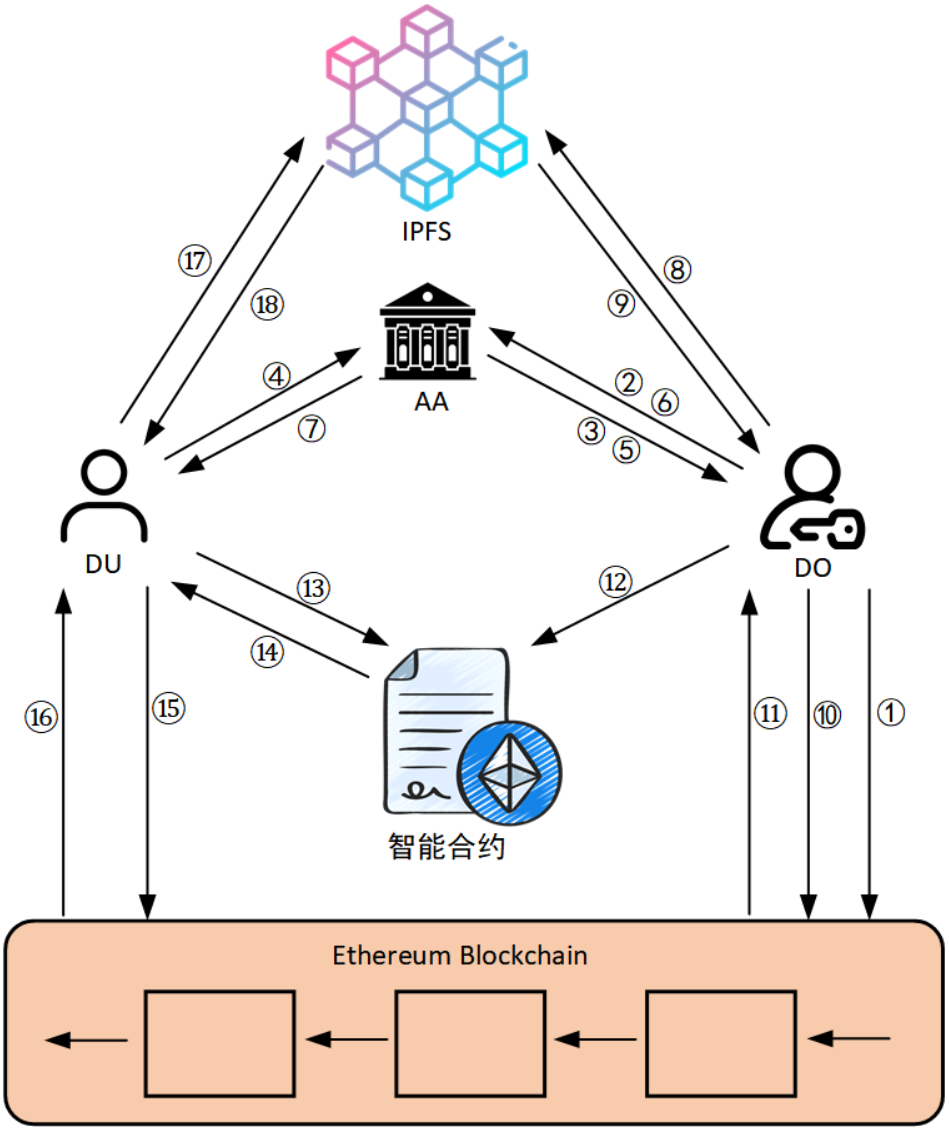

图 3-3 系统架构

3.1.2 工作流程

整个方案的工作流程如图3-3所示。整个方案包含 18 个步骤，每个编号对应的步骤具体描述如下：

1. DO 将合约部署到以太坊。部署成功后，记录合约账户地址、应用程序二进制接口、合约代码等信息。
2. DO 向 AA 发送注册请求，在注册请求中会携带如用户 ID、邮箱、合约地址、应用程序二进制接口等相关信息。
3. AA 记录 DO 的相关信息，为 DO 生成公钥 PK 和主密钥 MK。当 DO 收到密钥后，使用以太坊账户公钥将主密钥加密得到密文 CT_{msk} 并组装到交易 TX_{msk} 中发送到以太坊。
4. DU 向 AA 发送注册请求。在注册请求中，会携带 DU 的用户信息例如用户 ID、邮箱、以太坊账户地址等。

5. AA 将接收到的用户注册请求转发给相应 DO。
6. DO 根据用户信息选择对应属性 S 发送给 AA。此外，DO 将会将用户的以太坊账户地址加入到合约的白名单列表中。
7. AA 为 DU 生成用户密钥 SK_s 和 SK_d，并将合约相关信息、应用程序二进制接口、用户密钥返回给 DU。
8. DO 从共享文件 F 中选择关键词 kw，并随机生成密钥 K，使用 AES 算法和密钥 K 将文件加密得到密文 CT_F 并上传到 IPFS 中。
9. IPFS 向 DO 返回文件标识 $h_{location}$。
10. DO 首先使用 AES 算法和密钥 K 加密文件标识得到密文 CT_l。接着将访问策略 P 和密钥 K 输入到 CP-ABE 加密算法中，输出加密结果 CT_K。最后将密文 CT_l 和 CT_K 拼接得到 $CT_l||CT_K$，使用 AES 算法和随机密钥 K_1 对其加密得到密文 CT_{md}，将 CT_{md} 组装到交易 TX_{md} 中发送到以太坊。
11. DO 记录下以太坊交易 TX_{md} 的 ID 为 $txid$ 和密钥 K_1。
12. DO 为文件关键词 kw 生成消息认证码 MAC_{kw}，将 kw 与 $txid$ 进行异或得到密文 $CT_{txid} = txid \oplus kw$，将密钥 K_1 与 $txid$ 异或得到密文 $CT_{K_1} = K_1 \oplus txid$，将索引 $index = (MAC_{kw}, CT_{txid}, CT_{K_1})$ 存储到智能合约。
13. DU 使用关键词 kw 生成消息认证码 MAC_{kw}，将其作为 $token$ 发送到智能合约调用搜索函数查询相关文件。
14. 智能合约根据 $token$ 返回结果 (CT_{txid}, CT_{K_1})。
15. DU 根据智能合约返回的结果解密得到 $txid$ 以及密钥 K_1 并查找交易 TX_{md}，得到密文 CT_{md}，使用密钥 K_1 解密得到密文 CT_l 和 CT_K。
16. DU 使用用户密钥 SK_s 解密密文 CT_K 得到密钥 K，并使用密钥 K 解密密文 CT_l 得到文件标识 $h_{location}$。
17. DU 根据文件标识 $h_{location}$ 从 IPFS 中下载共享文件密文 CT_F。
18. DU 使用密钥 K 解密共享文件密文 CT_F 得到共享文件 F。

3.2 算法组成

整个方案主要由 6 个算法组成，分别是：Setup、UserRegistration、Encrypt、IndexGen、Search、Decrypt。Setup 算法用于为当数据所有者初始化整个系统。UserRegistration 算法接收用户注册请求并转发给 DO，校验用户注册所提交的注册信息，并生成用户密钥等。Encrypt 算法包含一系列加密操作，例如共享文件加密，共享文件标识加密，密钥加密等。IndexGen 算法用于关键词密文索引的生成。Search 算法可以为用户提供关键词搜索功能。Decrypt 算法包含了一系列解密操作。每个算法的详细描述如下：

Setup$(1^\lambda) \rightarrow (PK, MK)$：Setup 算法由 AA 运行，参数 λ 用于控制群的大小，输出公钥 PK 和主密钥 MK。当 AA 接收到 DO 的注册请求，其运行 Setup 函数为 DO 生成公钥和主密钥并返回给 DO。DO 接收到密钥后，使用自己的以太坊账户公钥将 MK 加密得到密文 CT_{msk} 并组装到交易 TX_{msk} 中发送到以太坊。

UserRegistration$(MK, S) \rightarrow (SK_s, SK_d)$：UserRegistration 算法由 AA 运行，使用主密钥 MK 和用户属性 S 作为参数，输出用户密钥 SK_s 和 SK_d。当 DU 向 AA 发送注册请求后，AA 将注册请求转发给对应 DO，由 DO 为用户分配属性 S 并发送给 AA，然后将 DU 的以太坊账户地址加入到合约的白名单列表中。AA 再使用主密钥和属性生成用户密钥。接着 AA 将用户密钥、合约地址、合约源码、应用程序二进制接口等信息发送给 DU。

Encrypt：Encrypt 算法由 DO 运行。该算法由三个子算法组成，分别是：FileEncrypt、KeyEncrypt、IndexGen。FileEncrypt 算法用于文件加密。KeyEncrypt 算法用于密钥加密。IndexGen 用于生成加密关键字索引。每个算法的具体描述如下：

- **FileEncrypt**$(F) \rightarrow (CT_F, K, kw)$：FileEncrypt 加密算法使用共享文件 F 作为参数，输出加密后的共享文件密文 CT_F、密钥 K、关键词 kw。DO 首先为文件 F 中生成若干个关键词索引 kw，并随机生成密钥 K，将密钥 K 和共享文件 F 输入到 AES 算法中，输出共享文件密文 CT_F，再将共享文件密文 CT_F 上传到星际文件系统，保存星际文件系统返回的文件标识 $h_{location}$。
- **KeyEncrypt**$(PK, K, h_{location}, P) \rightarrow CT_{md}$：KeyEncrypt 算法使用文件标识 $h_{location}$、访问策略 P、公钥 PK、共享文件密钥 K 作为参数，输出密文 CT_{md}。DO 首先使用共享文件密钥 K 对文件标识 $h_{location}$ 加密得到密文 CT_l。接着将密钥 K、访问策略 P 输入到 CP-ABE 加密算法中，输出密文 CT_k。最后将密文 CT_l 和 CT_K 拼接得到 $CT_l||CT_K$，使用 AES 算法和随机密钥 K_1 将其加密得到密文 CT_{md}，将 CT_{md} 组装进交易 TX_{md} 中发送到以太坊，记录交易 TX_{md} 的交易 ID 值 $txid$。
- **IndexGen**$(kw, txid, K_1, SK_s) \rightarrow index$：IndexGen 算法使用关键词 kw、用户密钥 SK_s、密钥 K_1、交易 TX_{md} 的 ID 值 $txid$ 作为参数，输出加密关键词索引 $index$。DO 首先使用用户密钥 SK_s 为关键词 kw 生成消息认证码 $MAC_{kw} = HMAC(kw, SK_s)$。接着使用关键词 kw 加密 $txid$ 得到密文 $CT_{txid} = txid \oplus kw$。最后使用交易 ID 值 $txid$ 加密密钥 K_1 得到密文 $CT_{K_1} = txid \oplus K_1$。将索引 $index(MAC_{kw}, CT_{txid}, CT_{K_1})$ 存储到合约。

TokenGen$(kw, SK_s) \rightarrow token$：TokenGen 算法由 DU 运行，使用关键词 kw 和用户密钥 SK_s 作为参数，输出搜索 $token$。DU 首先使用用户密钥 SK_s 计算关键词 kw 的消息认证码得到用于搜索的 $token = HMAC(kw, SK_s)$，接着将 $token$ 发送到合约以执行合约的搜索函数。

Search$(token, index) \rightarrow (CT_{txid}, CT_{K_1})$：Search 算法由合约运行。使用搜索 $token$ 和索引 $index$ 作为参数，输出交易 TX_{md} 的 ID 密文 CT_{txid} 以及密钥 K_1 的密文 CT_{K_1}。智能合约使用 DU 传递的参数 $token$ 在索引 $index$ 中查找，查找成功，则返回相关结果，查找失败，则返回空。

Decrypt$(CT_l, CT_k, SK_d, PK) \rightarrow F$：Decrypt 算法由 DU 运行，使用公钥 PK、用户密钥 SK_d、密钥 K 的密文 CT_k、文件标识密文 CT_l 作为输入，输出共享文件 F。

DU 根据合约返回的结果解密得到交易 TX_{md} 的 ID 值 $txid = CT_{txid} \oplus kw$ 与密钥 $K_1 = CT_{K_1} \oplus txid$。然后从以太坊中读取交易 TX_{md} 得到密文 CT_{md}，使用密钥 K_1 解密得到 CT_l 和 CT_k。如果密钥过程中用户密钥所链接的用户属性符合加密时密文所链接的访问策略时，DU 则可以从密文 CT_k 中恢复密钥 K。接着使用密钥 K 将密文 CT_l 恢复为文件标识 $h_{location}$。再根据文件标识从 IPFS 下载共享文件密文 CT_F，最终使用密钥 K 解密得到共享文件 F。

第 4 章 详细设计

4.1 算法设计

设 p 是一个大素数，G_0 是椭圆曲线上的 p 阶乘法循环群，G_T 是阶为 p 的乘法循环群，g 为 G_0 的生成元，$e: G_0 \times G_0 \rightarrow G_T$ 是一个双线性映射。定义将任意长度字符投影到 G_0 中某个元素的哈希函数 $H: \{0,1\}^* \rightarrow G_0$。定义拉格朗日系数 $\Delta_{i,L}$，其中 $i \in Z_p^*$，L 是元素为 Z_p 的集合，则有 $\Delta_{i,L(x)} = \prod_{j \neq i}^{j \in L} \frac{x-j}{i-j}$。具体构造如下：

Setup$(1^\lambda) \rightarrow (PK, MK)$：DO 向 AA 发送注册请求时，需要提供合约、身份、邮箱等信息。AA 接收到 DO 的注册请求后，运行 Setup 算法。参数 λ 决定群的大小，从椭圆曲线上选择一个阶为 p 的乘法循环群 G_0，g 为 G_0 的生成元。接着系统选择两个随机数 $\alpha, \beta \in Z_p$。公钥 PK 计算如公式1所示。

$$PK = G_0, g, h = g^\beta, e(g,g)^\alpha \tag{1}$$

主密钥为 $MK = (\beta, g^\alpha)$。AA 记录下 DO 身份信息、合约信息以及密钥信息，并将公钥和主密钥发送给 DO。DO 接收到密钥后，使用以太坊账户公钥将主密钥加密得到密文 CT_{msk} 并组装到交易 TX_{msk} 中发送到以太坊。

UserRegistration$(MK, S) \rightarrow (SK_s, SK_d)$：当 DU 向属性授权机构 AA 发送注册请求时，需要提供相关信息如用户 ID、邮箱、DU 的以太坊账户地址等。AA 收到注册请求后，将请求转发给 DO，由 DO 为 DU 分配属性 S 并发送给 AA，为了使 DU 可以访问合约，还需要将 DU 的以太坊账户地址加入到合约的白名单列表中。AA 接收到属性 S 后，使用 CP-ABE 密钥衍生算法生成用户密钥，具体过程如下：AA 遍历属性集 S 中的每个属性 $j \in S$，从 Z_p 中选择一个随机数 r_j，接着从 Z_p 中选择一个随机数 r。用户密钥 SK_d 可以通过公式2得到：

$$SK_d = (D = g^{(\alpha+r)/\beta}, \forall j \in S: D_j = g^r \cdot H(j)^{r_j}, D'_j = g^{r_j}) \tag{2}$$

用户密钥 $SK_s = K_s$, 其中 $K_s \in Z_p^*$，K_s 对于不同的用户都是固定的，可以在系统初始化时随机选择。最后 AA 向 DU 返回用户密钥 SK_s 和 SK_d、智能合约地址、应用程序二进制接口、智能合约代码。

FileEncrypt$(F) \rightarrow (CT_F, K, kw)$：DO 从共享文件 F 选择若干个关键词 kw，并随机生成用于加密共享文件 F 的 AES 密钥 K，加密共享文件得到密文 $CT_F = Enc_K(F)$，其中 Enc_K 表示使用 AES 算法加密文件 F，加密使用的密钥为 K。DO 将共享文件密文 CT_F 上传到星际文件系统并记录下文件标识 $h_{location}$。

KeyEncrypt$(PK, K, h_{location}, P) \rightarrow CT_{md}$：DO 使用 AES 算法和密钥 K 加密文件位置得到 $CT_l = Enc_K(h_{location})$。接着 DO 使用 CP-ABE 算法和访问策略 P 加密

密钥 K：使用层次遍历方法对访问树进行遍历，为所有节点各生成一个随机多项式 $q_t = a_0 + a_1 x + ... + a_{d_t} x^{d_t}$。多项式 q_t 的最高次幂 d_t 为节点 t 的阈值减一 $d_t = k_t - 1$。对于根节点 R，设 $s \in Z_p$ 为根节点多项式的常数项 $s = q_R(0) = a_0$。对于非根节点 z，自顶向下，更改节点的多项式的常数项为 $q_z(0) = q_{parent(z)}(index(z))$，其中 $index(z)$ 为 z 的序号，$parent(z)$ 为 z 的父节点。最后，让 Y 为访问树的叶子节点集合，$att(t)$ 表示叶子节点 t 的属性，则密文 CT_K 可以通过公式3计算得到。

$$CT_K = (P, \tilde{C} = Ke(g,g)^{\alpha s}, C = h^s, \forall y \in Y : C_y = g^{q_y(0)}, C'_y = H(att(y))^{q_y(0)}) \quad (3)$$

接着,DO 随机生成 AES 密钥 K_1,对密文 CT_{md} 和 CT_{K_1} 加密得到 $CT_{md} = Enc_{K_1}(CT_K, CT_l)$，将密文 CT_{md} 组装到交易 TX_{md} 并将发送到以太坊。当交易被打包后，使用 $txid$ 记录下交易 ID 并保存密钥 K_1。

IndexGen$(kw, txid, K_1, SK_s) \rightarrow index$：DO 使用文件关键词 kw、用户密钥 SK_s、交易 TX_{md} 的 ID 值 $txid$、密钥 K_1 生成加密索引 $index$。具体操作如公式 4-7 所示。

$$MAC_{kw} = HMAC(kw, SK_s) \quad (4)$$

$$CT_{txid} = txid \oplus kw \quad (5)$$

$$CT_{K_1} = K_1 \oplus txid \quad (6)$$

$$index = (MAC_{kw}, CT_{txid}, CT_{K_1}) \quad (7)$$

生成加密索引后，DO 将 $index$ 存储到智能合约，为 DU 提供文件搜索服务。

TokenGen$(kw, SK_s) \rightarrow token$：DU 计算 $token = HMAC(kw, SK_s)$，将 $token$ 作为参数调用智能合约的 Search 方法。

Search$(token, index) \rightarrow result$：智能合约首先判断 DU 是否是授权用户，如果不是则拒绝查询，否则返回搜索结果 $result = (CT_{txid}, CT_{K_1})$，其中 CT_{txid} 是 $txid$ 的密文，CT_{K_1} 是密钥 K_1 的密文。

KeyDecrypt$(CT_{md}, kw, K_s) \rightarrow (CT_k, CT_l)$：DU 获取到合约搜索结果后，首先使用公式 8-10 解密 $txid$ 和 K_1。

$$MAC_{KW} = HMAC(kw, SK_S) \quad (8)$$

$$txid = CT_{txid} \oplus kw = txid \oplus kw \oplus kw \quad (9)$$

$$K_1 = CT_{K_1} \oplus txid = K_1 \oplus txid \oplus txid \quad (10)$$

接着用户根据 $txid$ 读取交易 TX_{md} 获得密文 CT_{md}。使用密钥 K_1 解密得到密钥密文和文件标识密文 $(CT_K, CT_l) = Dec_{K_1}(CT_{md})$，其中 $Dec_{K_1}(CT_{md})$ 表示使用 AES 算法和密钥 K_1 解密 CT_{md}。

FileDecrypt$(CT_l, CT_K, SK_d, PK) \rightarrow F$：DU 首先需要使用户私钥 SK_d、公钥 PK

解密密文 CT_K。解密过程需要使用递归解法。定义函数 $DecryptNode(CT_K, SK_d, x)$。当 x 为访问树的叶子节点，$i = attr(x)$ 为叶节点 x 的属性，并且满足 $i \in S$，函数返回值如公式11所示。

$$\begin{aligned} DecryptNode(CT_K, SK_d, x) &= \frac{e(D_i, C_x)}{e(D_i', C_x')} \\ &= \frac{e(g^r \cdot H(i)^{r_i}, h^{q_x(0)})}{e(g^{r_i}, H(i)^{q_x(0)})} \\ &= e(g,g)^{rq_x(0)} \end{aligned} \tag{11}$$

如果 $i \notin S$，则 $DecryptNode(CT_K, SK_d, x) = \perp$。当 x 为访问树的内部节点 (非叶节点) 时，$DecryptNode(CT_K, SK_d, x)$ 的具体步骤如下：对于 x 的所有子节点 z，调用 $DecryptNode(CT_K, SK_d, z)$ 得到输出 F_z，函数输出 F_x 则可以通过公式12计算得到。

$$\begin{aligned} F_x &= \prod_{z \in S} F_z^{\Delta_{i,S'(0)}}, \quad \text{where} \begin{cases} i = index(z) \\ S' = index(z) : z \in S \end{cases} \\ &= \prod_{z \in S} (e(g,g)^{r \cdot q_z(0)})^{\Delta_{i,S'(0)}} \\ &= \prod_{z \in S} (e(g,g)^{r \cdot q_{parent(z)}(index(z))})^{\Delta_{i,S'(0)}} \\ &= \prod_{z \in S} (e(g,g)^{r \cdot q_x(i)})^{\Delta_{i,S'(0)}} \\ &= e(g,g)^{r \cdot q_x(0)} \end{aligned} \tag{12}$$

其中 $parent(z)$ 为 z 节点的父节点，$index(z)$ 是子节点 z 的编号，$\Delta_{i,S'(0)}$ 为拉格朗日系数。在定义完 $DecryptNode$ 算法后，我们可以定义解密算法。如果密钥生成时密钥所链接的属性集 S 满足加密时密文所链接的访问策略 P，则令 $A = DecryptNode(CT_K, SK_d, R) = e(g,g)^{rq_R(0)} = e(g,g)^{rs}$，其中 R 为访问树的根节点。最终可以通过公式13得到密钥 K。

$$K = \frac{\tilde{C}}{e(C,D)/A} = \frac{Ke(g,g)^{\alpha s}}{e(h^s, g^{(\alpha+r)/\beta})/e(g,g)^{rs}} \tag{13}$$

得到密钥 K 之后，解密文件标识 $h_{location} = Dec_K(CT_l)$。根据文件位标识从星际文件系统中下载共享文件密文 CT_F，最终解密得到共享文件 $F = Dec_K(F)$。

4.2 智能合约设计

DO 在系统初始化时需要部署一个名为 DataSharing 的合约。在这节中我们主要讨论 DataSharing 的设计。以太坊智能合约主要是由 Solidity[28] 语言编写。在合约中存在一些全局变量和函数以提供区块链的相关信息。在本文中我们主要使用了 $msg.sender$

这个全局变量，表示触发当前合约执行的交易的地址。在合约部署交易中其值为部署合约的地址，在合约函数调用中其值为触发当前函数调用的地址。

DataSharing 合约中有 3 个成员变量，这些变量主要用于记录系统当前状态，只有 DO 才有权更改这些变量。具体地，DataSharing 包含以下变量：

- $owner$：address 类型，保存了合约创建者的地址，也就是 DO 的账户地址。
- $authorizedUsers$：mapping 类型，存储授权用户的地址，将用户地址映射到一个布尔值。
- $index$：mapping 类型，存储加密索引，将加密关键词映射到一个数组。

DataSharing 还提供了几个函数用于更改成员变量，一些函数只有 DO 才有权调用，而一些函数只有合法 DU 才能调用。接下来，我们将详细讨论这些函数。

addUser($newUserAddress$)：该函数只能被合约创建者 (DO) 调用，将合法用户的账户地址加入到合约的白名单列表，参数 $newUserAddress$ 作为 DU 的账户地址。当 DU 向 AA 注册成功后，DO 将 DU 的以太坊账户地址加入到合约的白名单列表中。

Algorithm 1 addUser

Input: $newUserAddress$
Output: bool
  if $msg.sender$ is not $owner$ **then**
    return false
  end if
  $authorizedUser[newUserAddress] \leftarrow true$
  return true

removeUser($oldUserAddress$)：该函数只能被合约创建者 (DO) 调用，接收参数 $oldUserAddress$ 作为 DU 的账户地址。当系统需要撤销某个用户的权限时，DO 将用户的以太坊账户地址从授权列表中移除。

Algorithm 2 removeUser

Input: $oldUserAddress$
Output: bool
  if $msg.sender$ is not $owner$ **then**
    return false
  end if
  $authorizedUser[oldUserAddress] \leftarrow false$
  return true

addIndex($kw, txid, k_1$)：该函数只能被合约创建者 (DO) 调用，参数 kw 为加密的关键词，$txid$ 为加密的交易 ID，k_1 为加密的密钥。当 DO 需要共享文件时，从共享文件中选择若干个关键词，将其加密后和相关交易 ID 以及密钥保存到合约中。

search(kw)：该函数只能被合约的白名单列表中的用户调用，参数 kw 为加密的关键词。合约使用加密关键词 kw 找到映射的数组，并将整个数组返回给用户。

Algorithm 3 addIndex

Input: $kw, txid, k1$
Output: bool
  if $msg.sender$ is not $owner$ **then**
    return false
  end if
  $index[kw]$.append($(txid, k1)$)
  return true

Algorithm 4 search

Input: $kw, txid$
Output: result
  if $authorizedUser[msg.sender]$ is false **then**
    return false
  end if
  $len \leftarrow index[kw]$ array' length
  if len equal 0 **then**
    result $\leftarrow$ null
  else
    result $\leftarrow index[kw]$
  end if
  return result

deleteIndex(kw)：该函数只能被合约创建者 (DO) 调用，参数 kw 为加密的关键词。合约找到 kw 所映射的数组，将数组中的所有元素都清空。

Algorithm 5 deleteIndex

Input: kw
Output: bool
  if $msg.sender$ is not $owner$ **then**
    return false
  end if
  delete $index[kw]$
  return true

deleteFile($kw, txid$)：该函数只能被合约创建者 (DO) 调用，参数 kw 为加密的关键词，$txid$ 为交易 ID 的密文。合约首先使用 kw，找到 kw 所映射的数组，再遍历数组删除与 $txid$ 相同的数组元素，最后将数组元素向前移动一位。

Algorithm 6 deleteFile

Input: $kw, txid$

Output: bool

```
if msg.sender is not owner then
    return false
end if
len ← index[kw] array' length
if len equal 0 then
    return false
end if
for i ← 0 to len − 1 do
    if index[kw][i].txid equal txid then
        for j ← i + 1 to len − 1 do
            index[kw][j − 1] ← index[kw][j]
        end for
        delete index[kw][len − 1]
        break
    end if
end for
return true
```

第 5 章 性能和安全分析

5.1 性能测试

通过实验测量了所提方案的开销与性能。实验环境的详细配置如下：Ubuntu 20 系统，intel core i7-8750@2.2GHz 处理器，8GB 内存，Ropsten 测试网络。编程语言为 Java 和 Solidity。采用 JPBC 库实现了 ABE 加解密算法。对于 ABE 算法的实现，使用 JPBC 库中的 Type A 椭圆曲线上的点作为群的元素。哈希函数 H 使用的是 SHA-1，密钥散列函数 HMAC 使用的是 PBKDF2。

5.1.1 gas 开销

在所提方案中，数据所有者需要将密文组装到交易中发送到以太坊。在实验中，测量了数据所有者发送不同交易所需要的开销，在这里使用 $gasUsed$ 作为衡量单位。如表5-2所示。系统主密钥的大小是常量，不会随着属性数量的增加而增长，经测量其密文大小为 160 字节，将包含该密文的交易发送到以太坊花费为 $0.14。对于密文 CT_{md}，其由密文 CT_l 和 CT_K 加密而来，由于 $h_{location}$ 和 K 大小固定，且假设访问策略不会改变，因此 CT_{md} 也是固定大小。指定访问树的叶节点数量为 6，测量得到密文 CT_{md} 大小为 1952 字节，开销为 $0.28。

密文长度	Size(bytes)	Gas Used	Actual Cost(ether)	USD
CT_{mk}	160	23536	0.000046(ether)	0.14
CT_{md}	1952	52100	0.000091(ether)	0.28

表 5-2 密文开销

用户和数据所有者可以对智能合约进行调用，合约中函数的执行需要花费一定量的 gas。我们测量了合约中函数开销，测量结果如图表5-3和表5-4所示。

在表5-3中列出了在不同文件数量情况下，各函数的开销。创建合约只需 DO 执行一次，开销为 $9.60。在注册时，DU 的用户信息一旦校验通过，DO 就会执行 addUser 函数，开销为 $0.37。当 DO 需要从白名单列表中移除 DU 时，removeUser 会被执行，开销为 $0.20。当 DO 需要共享新文件时，addIndex 函数会被执行，开销为 $0.63。这些函数的开销是固定的，不会随着文件数量的增加而增大。

函数	Gas Used	Actual Cost(ether)	USD
dataSharing Contract create	1236191	0.00309(ether)	9.60
addUser	47763	0.000119(ether)	0.37
removeUser	25815	0.000065(ether)	0.20
addIndex	81752	0.000204(ether)	0.63

表 5-3 函数开销

文件数量	函数	Gas Used	Actual Cost(ether)	USD
1	search	34812	0.000087(ether)	0.27
	deleteFile	36293	0.000091(ether)	0.28
	deleteIndex	36917	0.000092(ether)	0.29
5	search	58374	0.000146(ether)	0.45
	deleteFile	69094	0.000173(ether)	0.54
	deleteIndex	69167	0.000172(ether)	0.53
10	search	87828	0.000220(ether)	0.68
	deleteFile	110754	0.000277(ether)	0.86
	deleteIndex	109479	0.000274(ether)	0.85
15	search	117285	0.000293(ether)	0.91
	deleteFile	152414	0.000381(ether)	1.18
	deleteIndex	149791	0.000374(ether)	1.16
20	search	146744	0.000367(ether)	1.14
	deleteFile	194074	0.000485(ether)	1.50
	deleteIndex	190103	0.000475(ether)	1.47

表 5-4 函数开销

在表5-4中列举了函数开销与文件数量有关的函数。为 5 不同的加密关键词分别添加 1，5，10，15 和 20 个文件，合约中 search、deleteFile、deleteIndex 操作所需开销如表5-4和图5-4所示。从图5-4可以看出，随着文件数量的增多，这三个函数的开销都会增大，且开销都与文件数量近似成线性关系。此外，deleteFile 函数与 deleteIndex 函数的开销非常接近。当文件数量为 5 时，search 的开销大约为 $0.45，deleteIndex 的开销大约为 $0.53。当测试 deleteFile 操作时，将每个关键词所映射数组中的第一个元素删除。这样，deleteFile 执行所需开销最大。当文件数量为 5 时，删除一个文件的开销为大约为 $0.54。

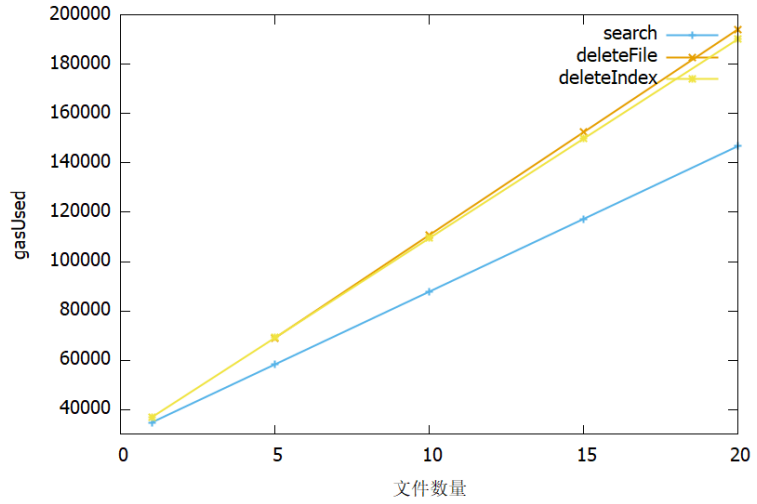

图 5-4 函数开销

5.1.2 执行效率

为了分析整个方案的性能，使用不同的属性数量对加密、解密、密钥生成的执行时间进行了测量。加密时间指的是文件加密所需时间、文件标识加密所需时间、对密钥 K 加密所需时间以及加密 CT_l 和 CT_K 所需时间之和。解密时间包含解密密文 CT_{md} 所需时间、解密密钥 K 所需时间、解密文件标识所需时间以及解密文件所需时间。密钥生成执行时间指属性授权机构为用户生成用户密钥所需时间。对于加密时间的测量，保持文件大小、文件位置不变，改变访问树中属性数量，测量结果如图5-5所示。由于计算密文 CT_K 需要对访问树中的每个属性都进行计算，因此随着访问树中属性数量的增加，加密所需时间也会增长。对于解密和生成密钥时间的测量，保持访问树中属性数量不变，改变用户密钥所链接的用户属性数量，测量结果如果5-6所示。由于计算密钥 SK_d 需要对所链接的每一个用户属性都进行计算，因此增加用户密钥所链接的用户属性数量，用户密钥生成时间也会增长。而在解密过程中，需要使用到每一个用户属性计算访问树的根节点的秘密值，因此解密所需时间也会随着分配给用户属性数量的增多而增长，但增长幅度没有生成密钥的增长幅度大。

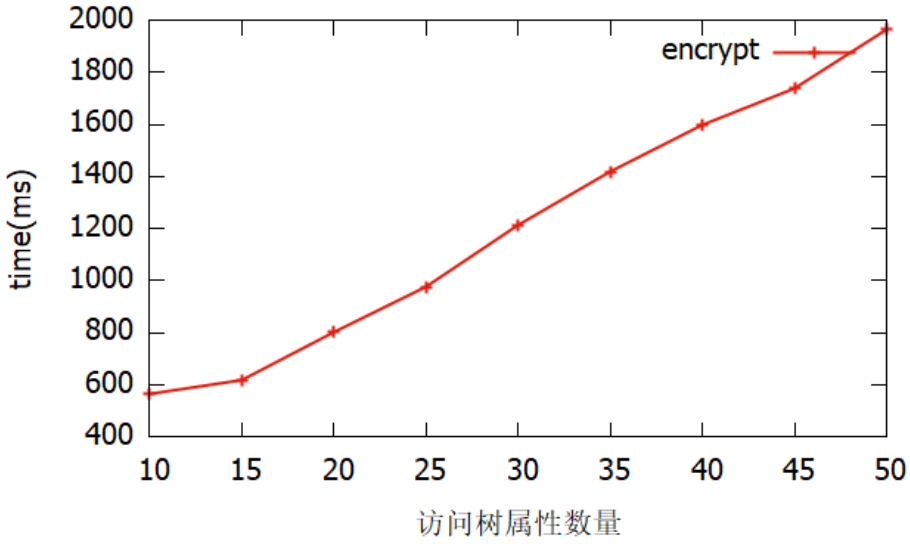

图 5-5 加密所需要时间

5.2 安全和隐私分析

本文所提方案结合星际文件存储系统 IPFS、以太坊区块链、密文策略基于属性加密，为用户数据存储以及数据共享提供了更强的隐私保护力度。在本小节，对该方案安全性和隐私性进行讨论。

5.2.1 安全性

区块链是公开的账本，所有人都可以看与数据所有者相关的交易。但由于交易中存储的是密文，即使攻击者发现了与数据所有者相关的交易，也无法破解任何密文获得有效信息。对于合约，合约存储了加密的索引信息，只要不是合法用户，就无法获得与共享文件相关的交易 ID 与密钥。假设攻击者伪装成合法用户向数据所有者注册，

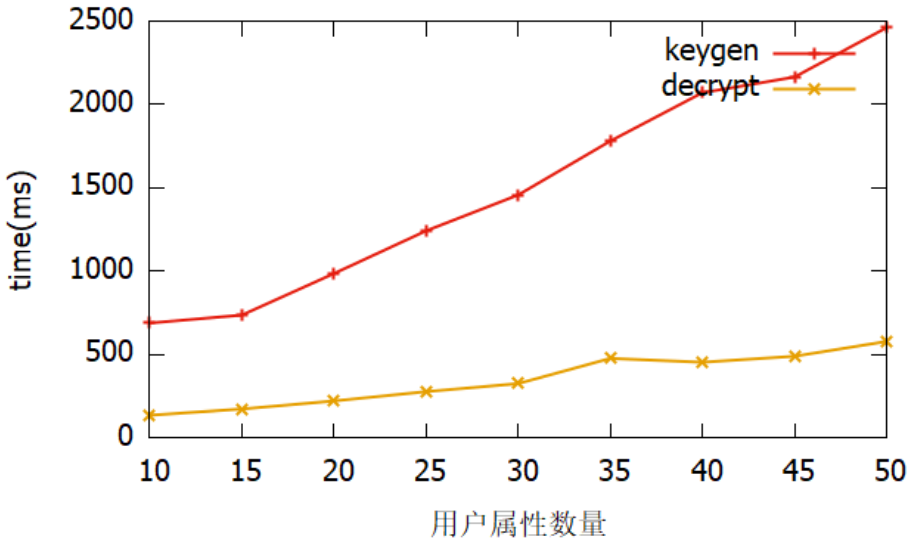

图 5-6 解密和生成用户密钥所需时间

得到密钥 SK_s。攻击者通过密钥 SK_s 可以解密并获得与文件有关的交易 ID 与密钥，但由于攻击者所分配的属性限制，攻击者只能访问其属性所能满足的共享数据密文。

5.2.2 隐私性

本文所提方案，将文件密文上传到 IPFS，防止第三方实体滥用用户数据。将经过加密的数据元信息存储在区块链，使用以太坊账户进行交易，可以有效隐藏用户的真实信息。

第 6 章 总 结

隐私和安全一直都是云存储系统中存在的两个关键问题。在本文，提出了一个基于区块链的云文件安全共享方案，将共享数据加密上传到 IPFS 中，保证数据的机密性；通过属性授权机构向用户生成密钥，只有当密钥生成时用户密钥所链接的属性满足加密时密文所链接的访问策略才能从密文中恢复明文，实现了对共享数据的细粒度共享；通过区块链存储数据元信息密文，借助区块链进行溯源以验证数据的完整性。同时，该方案实现了一个对加密关键词搜索的功能，进一步提高了整个系统的隐私性和安全性。安全性分析和实验结果表明，所提方案具有高效性和可伸缩性。对于未来的工作，我们计划在方案中添加密钥撤销功能，使得被撤销密钥的用户即使仍然保留了交易 ID 和对应密钥，也无法解密对应数据。

致 谢

时光荏苒，一眨眼便快到毕业的日子。在论文即将付梓的这一刻，感悟颇多。在这，我要诚挚的感谢那些曾经帮助过我的人。

首先，我要感谢冯朝胜老师，是冯老师为我提供的毕业设计题目，并为我以后的科研方向指明了清晰的方向。

其次，我要感谢计算机科学学院的各位老师。通过各位老师的教导，我学习到了许多新知识和研究方法，让我深受启发。

再次，我还要感谢刘彬师兄。师兄为我的毕业设计提供了解决思路，并在论文完成过程中一路给予帮助。

最后，我要感谢我未来的研究生导师和各位师兄。他们把我带入了科研的路途，为我的科研路途提供了宝贵的意见。

参考文献

[1] AZHIR E, NAVIMIPOUR N J, HOSSEINZADEH M, et al. Query optimization mechanisms in the cloud environments: a systematic study[J]. International Journal of Communication Systems, 2019, 32(8): e3940.

[2] KAANICHE N, LAURENT M. Data security and privacy preservation in cloud storage environments based on cryptographic mechanisms[J]. Computer Communications, 2017, 111: 120-141.

[3] LI Y, GAI K, QIU L, et al. Intelligent cryptography approach for secure distributed big data storage in cloud computing[J]. Information Sciences, 2017, 387: 103-115.

[4] SHEN J, GUI Z, JI S, et al. Cloud-aided lightweight certificateless authentication protocol with anonymity for wireless body area networks[J]. Journal of Network and Computer Applications, 2018, 106: 117-123.

[5] SHEN J, WANG C, LI T, et al. Secure data uploading scheme for a smart home system [J]. Information Sciences, 2018, 453: 186-197.

[6] ZYSKIND G, NATHAN O, et al. Decentralizing privacy: Using blockchain to protect personal data[C]//2015 IEEE Security and Privacy Workshops. IEEE, 2015: 180-184.

[7] BETHENCOURT J, SAHAI A, WATERS B. Ciphertext-policy attribute-based encryption[C]//2007 IEEE symposium on security and privacy (SP'07). IEEE, 2007: 321-334.

[8] 袁勇, 王飞跃. 区块链技术发展现状与展望[J]. 自动化学报, 2016, 42(4): 481-494.

[9] ALBLOOSHI M, SALAH K, ALHAMMADI Y. Blockchain-based ownership management for medical iot (miot) devices[C]//2018 International conference on innovations in information technology (IIT). IEEE, 2018: 151-156.

[10] AGBO C C, MAHMOUD Q H, EKLUND J M. Blockchain technology in healthcare: a systematic review[C]//Healthcare: volume 7. Multidisciplinary Digital Publishing Institute, 2019: 56.

[11] SAHAI A, WATERS B. Fuzzy identity-based encryption[C]//Annual international conference on the theory and applications of cryptographic techniques. Springer, 2005: 457-473.

[12] BONEH D, FRANKLIN M. Identity-based encryption from the weil pairing[C]// Annual international cryptology conference. Springer, 2001: 213-229.

[13] GOYAL V, PANDEY O, SAHAI A, et al. Attribute-based encryption for fine-grained access control of encrypted data[C]//Proceedings of the 13th ACM conference on Computer and communications security. 2006: 89-98.

[14] LI J, REN K, ZHU B, et al. Privacy-aware attribute-based encryption with user account-

ability[C]//International Conference on Information Security. Springer, 2009: 347-362.

[15] ATTRAPADUNG N, LIBERT B, PANAFIEU E D. Expressive key-policy attribute-based encryption with constant-size ciphertexts[C]//International workshop on public key cryptography. Springer, 2011: 90-108.

[16] GOYAL V, JAIN A, PANDEY O, et al. Bounded ciphertext policy attribute based encryption[C]//International Colloquium on Automata, Languages, and Programming. Springer, 2008: 579-591.

[17] CHASE M. Multi-authority attribute based encryption[C]//Theory of cryptography conference. Springer, 2007: 515-534.

[18] LI J, WANG Y, ZHANG Y, et al. Full verifiability for outsourced decryption in attribute based encryption[J]. IEEE transactions on services computing, 2017, 13(3): 478-487.

[19] NAKAMOTO S. Bitcoin: A peer-to-peer electronic cash system[J]. Decentralized Business Review, 2008: 21260.

[20] WOOD G, et al. Ethereum: A secure decentralised generalised transaction ledger[J]. Ethereum project yellow paper, 2014, 151(2014): 1-32.

[21] HOPWOOD D, BOWE S, HORNBY T, et al. Zcash protocol specification[J]. GitHub: San Francisco, CA, USA, 2016, 86.

[22] WILKINSON S, BOSHEVSKI T, BRANDOFF J, et al. Storj a peer-to-peer cloud storage network[M]. Citeseer, 2014.

[23] RAHULAMATHAVAN Y, PHAN R C W, RAJARAJAN M, et al. Privacy-preserving blockchain based iot ecosystem using attribute-based encryption[C]//2017 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS). IEEE, 2017: 1-6.

[24] FAN X, HUO Y. Blockchain based dynamic spectrum access of non-real-time data in cyber-physical-social systems[J]. IEEE Access, 2020, 8: 64486-64498.

[25] HÎRTAN L A, DOBRE C. Blockchain privacy-preservation in intelligent transportation systems[C]//2018 IEEE International conference on computational science and engineering (CSE). IEEE, 2018: 177-184.

[26] SU Q, ZHANG R, XUE R, et al. Revocable attribute-based signature for blockchain-based healthcare system[J]. IEEE Access, 2020, 8: 127884-127896.

[27] BENET J. Ipfs-content addressed, versioned, p2p file system[A]. 2014.

[28] Units and globally available variables[EB/OL]. https://docs.soliditylang.org/en/latest/units-and-global-variables.html.