****

**本科生毕业论文（设计）**

|  |  |
| --- | --- |
| 题目： | 基于TEE和区块链的 |
|  | 隐私保护计算平台 |

|  |  |
| --- | --- |
| 姓名 | 王明业 |
| 学号 | 17343107 |
| 院系 | 计算机学院 |
| 专业 | 软件工程 |
| 指导教师 | 郑子彬 教授 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 2021 | 年 | 4 | 月 | 10 | 日 |

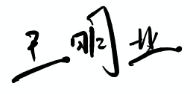
|  |
| --- |
| 基于TEE和区块链的隐私保护计算平台 |
| A Privacy-Preserving Computing Platform based on TEE and Blockchain |

|  |  |
| --- | --- |
| 姓名 | 王明业 |
| 学号 | 17343107 |
| 院系 | 计算机学院 |
| 专业 | 软件工程 |
| 指导教师 | 郑子彬 教授 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 2021 | 年 | 4 | 月 | 10 | 日 |

**学术诚信声明**

本人郑重声明：所呈交的毕业论文（设计），是本人在导师的指导下，独立进行研究工作所取得的成果。除文中已经注明引用的内容外，本论文（设计）不包含任何其他个人或集体已经发表或撰写过的作品成果。对本论文（设计）的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。本论文（设计）的知识产权归属于培养单位。本人完全意识到本声明的法律结果由本人承担。

作者签名：

日 期： 2021年 4月 10日

摘要

TEE（Trusted Execution Environment）即可信执行环境，它保证加载到其中的代码和数据在保密性和完整性方面受到保护[1]。区块链本质是一个数据结构，储存于区块链中的信息具有不可伪造、可追溯和公开透明的特征[2][3]。本文给出一种计算平台的设计，该平台使用当下流行的加密方法如RSA[4]、AES[5]，并结合TEE和区块链技术，可以全程保护用户的隐私。

**关键词**：TEE、可信执行环境、区块链、隐私保护、服务计算、计算平台、代码运行平台

Abstract

TEE (Trusted Execution Environment) guarantees that the confidentiality and the integrity of the code and data loaded into it are protected. The essence of the blockchain is a data structure, and the information stored in the blockchain has the characteristics of unforgeability, traceability and transparency. This paper presents a design of a computing platform. The platform which uses popular encryption methods such as RSA and AES, and integrates with TEE and blockchain technology, can protect the privacy of users throughout the process.

**Keywords**: TEE (Trusted Execution Environment), Blockchain, Privacy Preserving, Service Computing, Computing Platform, Codes Running Platform

目录

[一、 绪论 4](#_Toc69394324)

[二、 TEE、区块链和密码学技术在本平台的使用 5](#_Toc69394325)

[（一） TEE工具 5](#_Toc69394326)

[（二） 区块链工具 6](#_Toc69394327)

[（三） 密码学技术 7](#_Toc69394328)

[1. 非对称加密算法RSA 7](#_Toc69394329)

[2. 对称加密算法AES 8](#_Toc69394330)

[3. 散列算法SHA-256 8](#_Toc69394331)

[三、 整体架构与流程 9](#_Toc69394332)

[（一） 整体架构 9](#_Toc69394333)

[（二） 流程详解 9](#_Toc69394334)

[1. 用户端与密钥管理器进行密钥传递的一系列过程 10](#_Toc69394335)

[2. 用户端发送代码文件给文件接收器的一系列过程 12](#_Toc69394336)

[3. 文件接收器发送代码文件给代码运行器的一系列过程 14](#_Toc69394337)

[4. 代码运行器将加密运行结果和运行信息发送给区块链记录的一系列过程 15](#_Toc69394338)

[5. 用户端向区块链记录请求信息的一系列过程 16](#_Toc69394339)

[四、 UML建模 18](#_Toc69394340)

[（一） 时序图 18](#_Toc69394341)

[五、 参考文献 18](#_Toc69394342)

[附录 A. Word域操作 19](#_Toc69394343)

[六、 致谢 22](#_Toc69394344)

# 绪论

当下的一些计算任务，如深度学习，需要很高的算力[6]，个人电脑的配置无法满足在可接受时长内完成这些计算任务并得到结果，很多用户会使用大公司的云计算平台运行这些计算任务。比较流行的一种云计算服务平台做法是：在物理机器上运行虚拟机，用户与虚拟机实例进行交互，用户对虚拟机有完整的访问和操作权限[7]，使用体验等同于以管理员身份在使用一个操作系统。例如，虚拟机实例的操作系统是Ubuntu，那么用户的使用体验等同于以管理员身份在使用一个完整的Ubuntu操作系统。并且，公司监听虚拟机的使用以记账收费。

这样的云计算服务平台做法可能会有以下问题：

在物理机器上运行多个虚拟机，虚拟机的运行会占用资源。考虑到机器最主要的功能是运行计算任务，那么相对于计算任务占用资源，运行虚拟机占用资源可以看成是一种浪费。

小企业或者个人，如果有空闲计算资源想供他人使用，由于技术限制，他们可能无法像大公司一样在物理机器上运行多个相互隔离的虚拟机实例供用户使用，并进行监听与记账收费。并且，小企业或个人的空闲计算资源可能不够充裕，无法支持运行多个虚拟机并同时运行计算任务。

对于不熟悉系统操作的人员，面对一个虚拟机实例，需要花费时间精力去学习虚拟机系统的使用。而他们想要的服务，只是上传代码文件到服务器运行，然后得到结果。

本文设计的基于TEE和区块链的隐私保护计算平台，其基本运作流程是：平台在TEE中运行，用户上传计算任务（代码）到平台。平台令用户代码在TEE中执行，并得到运行结果以及运行过程信息。而后平台将加密的运行结果和运行信息存入区块链中，区块链可公开访问，用户访问区块链内容以获取运行结果。

本文设计的平台，只用在物理机器上运行一个带有在TEE中执行代码的功能的系统，然后在该系统中执行一套程序。这套程序会生成并管理密钥，接收客户端发来的代码，然后执行代码完成计算任务，最后把计算结果以及计算过程信息记录在区块链上。加密算法、散列算法等密码学技术保证了客户端和服务端数据传输的保密性；在TEE中执行程序保证了计算任务运行时的保密性；区块链上储存加密后的计算结果和计算过程信息，保证了计算结果的不可篡改性、保密性和匿名性，实现了全程的隐私保护。

本文设计的基于TEE和区块链的隐私保护计算平台只用在物理机器上运行一个操作系统，不会有运行多个虚拟机占用过多资源的情况。并且，运行本文提出计算平台的技术门槛和开销，相比运行上述云计算服务平台要低。此外，用密码学技术、TEE和区块链实现的全程隐私保护，可以让用户足够信任。而用户需要做的只是通过HTTP请求与计算平台进行交互，无需操作一个完整的系统。可见，本文设计的计算平台改善了前述云计算服务平台存在的缺点。

# TEE、区块链和密码学技术在本平台的使用

TEE工具、区块链工具和一些密码学技术的使用是本文提出的隐私保护计算平台的关键所在。先了解TEE工具、区块链工具和平台使用到的密码学技术，有助于理解平台的架构、流程和模块功能。在此先给出TEE工具、区块链工具和平台使用到的密码学技术的描述，并提及其在本平台中的大致使用方式。

## TEE工具

比较流行的TEE开发工具是Intel Software Guard Extensions (SGX)[8]。但Intel SGX相对比较底层，使用Intel SGX的SDK开发程序，过程非常繁琐：需要花费大量时间学习相关接口、编程模型以及SGX SDK的编译系统。

Youren Shen等提出的Occlum，是一个针对Intel SGX的内存安全、多进程的库操作系统[9]。Occlum的最大优势是其易用性。使用Occlum，可以不用编写任何额外的SGX相关编码，仅需要使用一些shell命令就可以使程序在SGX保护下运行，也即是在TEE中运行。此外，Occlum还支持运行多种不同编程语言的程序，包括C/C++、Python、Go和Java。

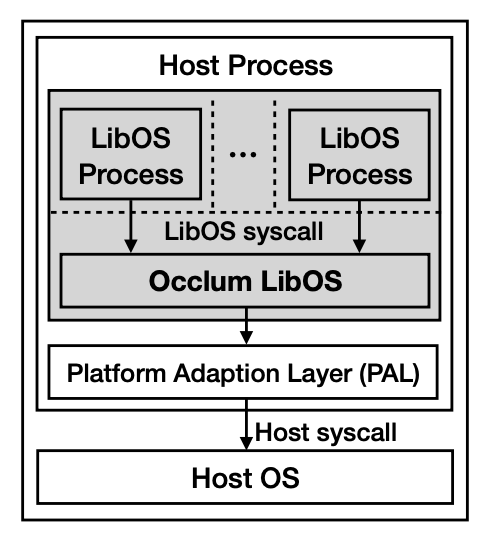


图 1 Occlum库操作系统概述

如图 1所示，Occlum是建立在主操作系统（Host OS）上的库操作系统（LibOS即Library OS）。其具体使用方式是：在安装有Occlum的主系统上，使用Occlum的相关命令运行代码文件，即可实现代码在TEE中执行。事实上，Occlum作者给出了一个内置Occlum的Ubuntu 18.04系统镜像，使用该镜像安装Ubuntu操作系统，即可在Ubuntu中使用Occlum命令，无需额外安装。该Ubuntu系统的其他使用，如文件系统、系统命令等，与一般的Ubuntu系统一样。

由于主系统是Ubuntu，所以平台代码的部署、服务器维护等操作，跟使用一般Ubuntu机器的操作一样。要让平台和计算任务运行在TEE中，只需要编写shell脚本，使用Occlum命令，让相关代码运行即可。

## 区块链工具

区块链本质是一种按照时间顺序将数据区块用类似链表的方式组成的数据结构[10]。一个基本的区块链，每个区块会包含[11]：索引（index）、时间戳（timestamp）、交易数据（transactions，事实上交易数据可以是任何想储存在区块上的数据）、校验（proof）和前一个区块的散列（the hash of the previous block）。

本平台会在代码中自行实现一个适用本平台需求的区块链。该区块链的区块中的交易数据实际记录的是每次计算任务的信息，包括这次任务的标示、计算结果和运行信息，具体将在后文讲解。并且，因为只有服务端运行计算任务并将相关信息写入区块链，所以该区块链只会在服务端增长，而不是多方共同作用使得区块链增长，所以省去了工作量证明机制和校验。

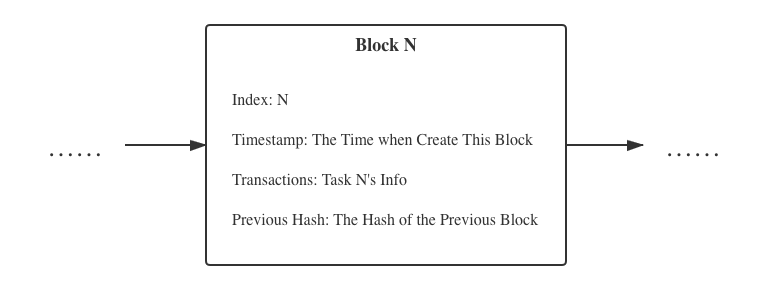


图 2 本平台所用区块链示意图（第N个区块）

## 密码学技术

### 非对称加密算法RSA

RSA是目前被广泛使用的非对称加密算法。Ron Rivest，Adi Shamir和Leonard Adleman于1977年共同提出了RSA。RSA的名称由三位作者姓氏首字母组合而成[12]。RSA的具体操作实现原理以及实现细节在此不赘述，只是简单介绍一下其大致使用过程。

设A想要通过一个不安全的媒介接收B的一条私密消息，并使用RSA算法来保证消息传输的安全性。A首先要按照RSA规定的密钥生成方式来产生一个公钥和一个私钥，其中公钥可以公开，私钥必须保持机密。B获取A的公钥，然后使用A的公钥，对私密消息应用RSA加密流程，生成密文，将密文通过不安全的媒介传输给A。A得到密文后，可以用自己的私钥来解密，得到消息的原文。由于只有A拥有其公钥对应的私钥，所以即便他人在媒介上截取了密文，也无法解密得到消息原文。

此外，RSA还可以对消息进行签名。设A想对传给B的消息签名，那么A先对消息计算散列值（信息摘要），然后A根据RSA算法流程，用私钥对该散列值加密（如同上述公钥对消息进行加密的过程），如此产生一个签名，并将该签名附在消息的后面，一同传输给B。B获得A传来的带有签名的消息后，可以根据RSA算法流程，用A的公钥对签名进行解密（如同上述私钥对消息进行解密的过程），得到一个值。然后B自己对消息计算散列值，如果此时得到的散列值与先前解密得到的值相同，那么B可以确定发信人持有A的私钥，且此消息在传输媒介上未被篡改。

RSA算法本身要求明文长度要小于其模值，所以不能一次性加密过长的数据。并且，RSA加解密的速度比对称加密算法如AES慢得多。实际使用一般会用RSA加密对称加密算法如AES的密钥，再用AES加密消息。

在本平台中，RSA算法会用来加密AES密钥、对传输数据签名，以及加密计算任务的散列值（保证散列算法得到的散列值长度允许RSA加密）。

### 对称加密算法AES

AES即Advanced Encryption Standard，是目前被广泛使用的非堆成加密算法，由oan Daemen和Vincent Rijmen共同提出。AES具体操作实现原理以及实现细节同样不再赘述，在此简单介绍一下其大致使用过程。

AES支持128、192和256位的密钥。设A打算使用128位长的密钥来对消息进行AES加密，那么A首先要生成一条长度为128位（16字节）的数据作为密钥。然后按照AES算法的流程对消息进行加密生成密文。只要持有与A加密时使用的密钥相同的密钥，就可以解密密文得到消息原文。

在本平台中，AES算法会用来加密一些较长的数据，例如RSA公钥、文件内容、计算任务运行结果和运行信息等。

### 散列算法SHA-256

SHA-256属于SHA-2标准，由美国国家安全局研发。使用SHA-256算法，对任意给定的消息，能生成长度为256位的散列值（信息摘要）。并且，无法从消息的散列值推出原消息，也几乎不会有不同的消息产生相同的散列值。

在本平台中，SHA-256会在RSA签名过程中使用到，并且，对于每个计算任务，也会按一定规则用SHA-256生成每个任务特有的散列值，该散列值会作为每个计算任务的唯一标识。

# 整体架构与流程

## 整体架构

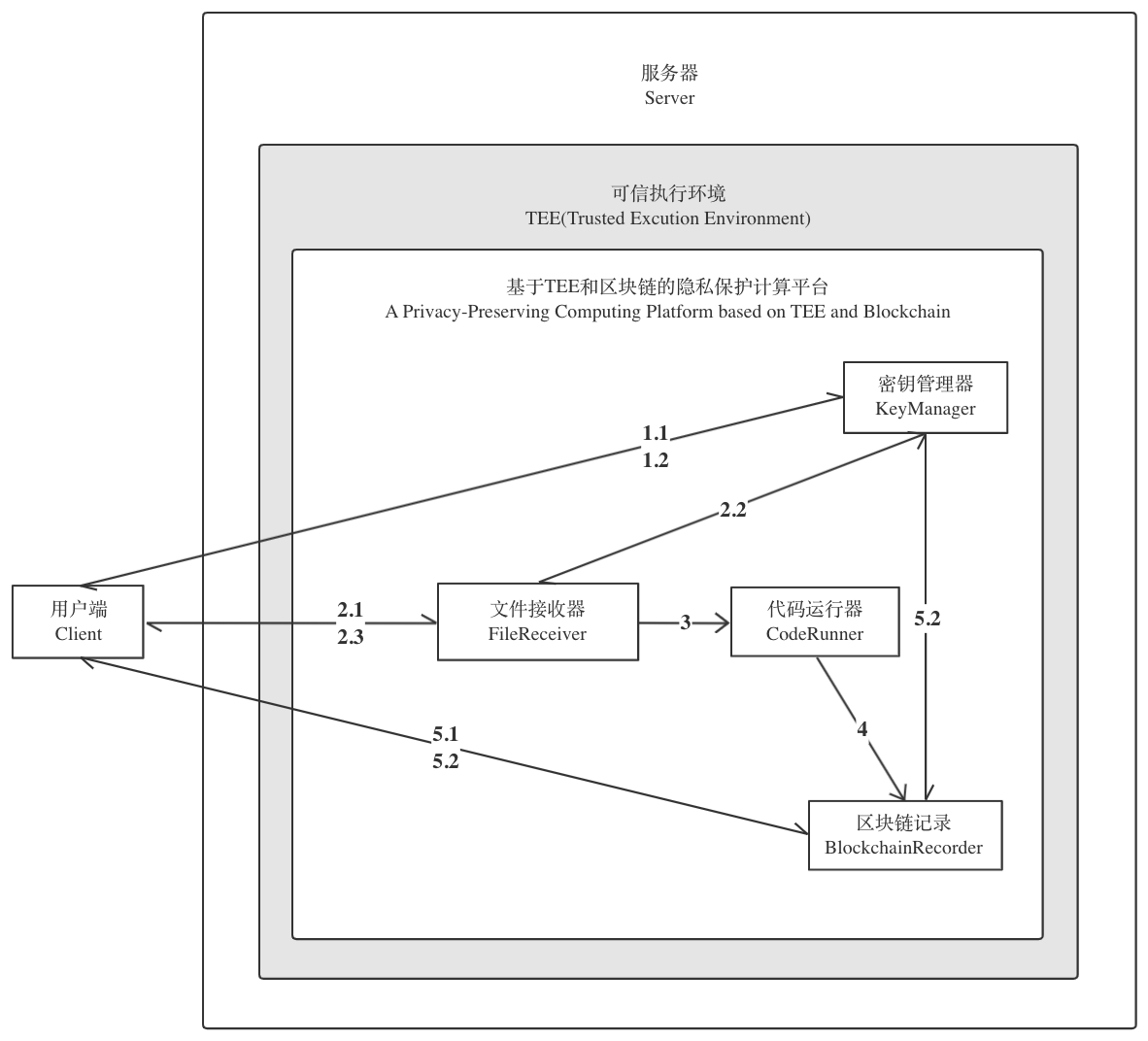


图 3 基于TEE与区块链的隐私保护计算平台 整体架构示意图

如图 3所示，本平台运行于服务器的TEE中。平台包括密钥管理器KeyManager，文件接收器FileReceiver、代码运行器CodeRunner和区块链记录BlockchainRecorder。图中个模块间的连线表示模块间有交互；连线上的标号表示具体交互过程或具体流程，详情可见下一部分（二）流程详解。

## 流程详解

本部分标号、指代等请参考：**错误!未找到引用源。**。流程的标号代表流程发生的先后顺序，数字越小，代表流程越先发生。

流程中非对称加解密算法采用RSA，对于一对RSA密钥KRN，称其中公钥为KRN pub，私钥为KRN pri；

流程中对称加解密算法采用AES，AES密钥称为KA；

流程中散列（信息摘要）算法采用SHA-256；

用户端A的第n次计算任务记为An，如A的第1次计算任务记为A1；

那么，用户端A执行计算任务A1的流程如下：

（用户端A执行计算任务An (n > 1)时，可省略下述流程1，仅进行23456）

### 用户端与密钥管理器进行密钥传递的一系列过程

1.1:

用户端A生成一对密钥KRA,，将其中公钥KRA pub发送给密钥管理器；

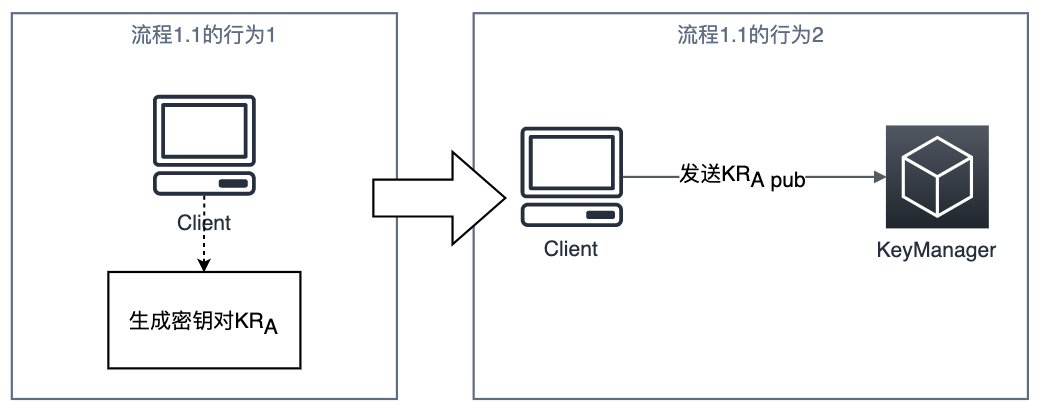


图 4 流程1.1示意图

1.2：

密钥管理器收到用户端A发来的包含公钥KRA pub的请求后，生成一对新密钥KRS和一个新密钥KA，密钥管理器保存KRA pub与KRS及KA的对应关系。然后将新生成的密钥KA用KRA pub加密，KRS pub用KA加密，并对加密后的密文用KRS pri进行加签，返回给用户端A。

用户端A先用KRA pri对密文解密得到KA，再用KA对密文解密得到KRS pub，而后用KRS pub对密文进行验签，验签成功说明得到的KRS pub和KA可信。

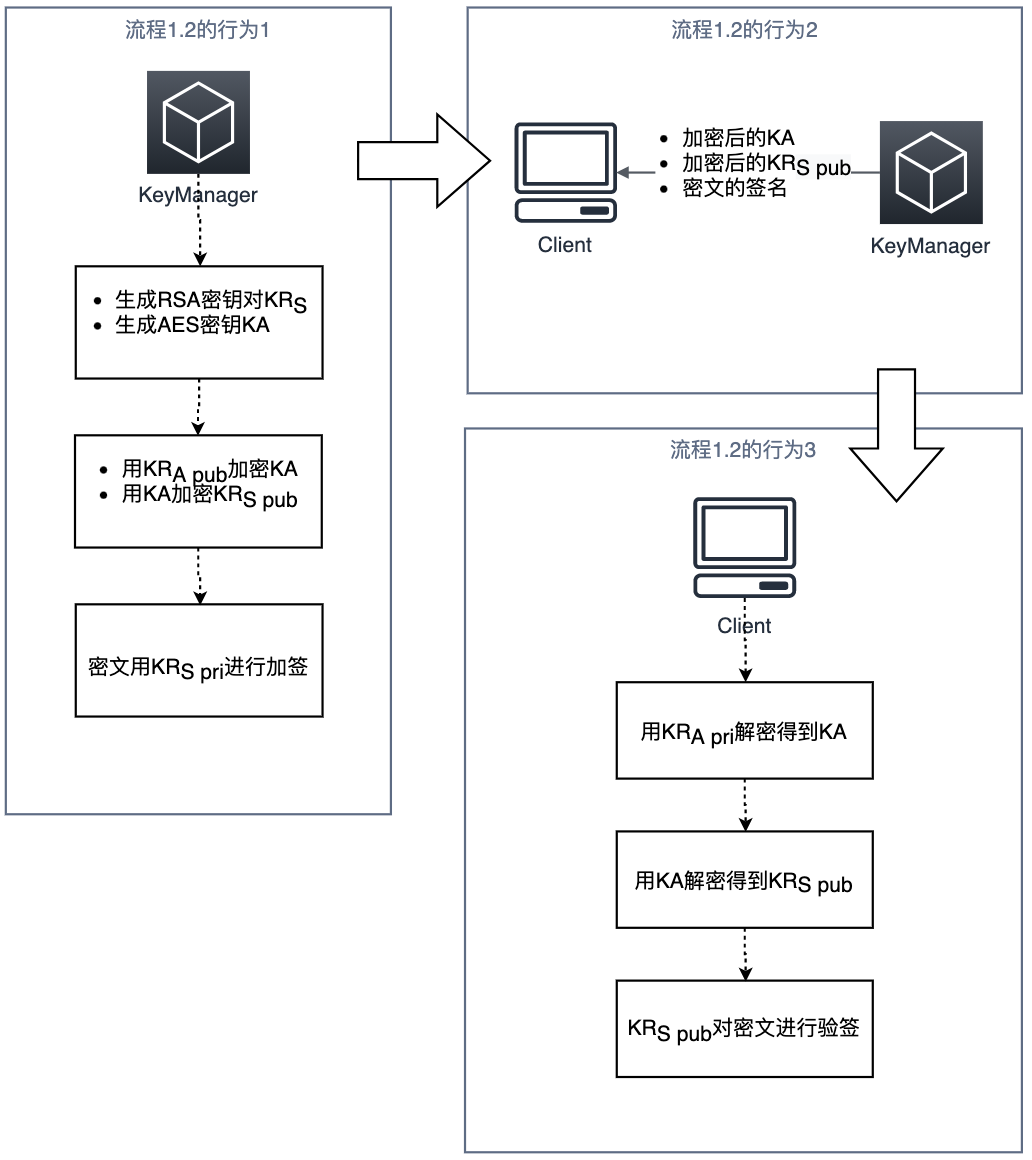


图 5 流程1.2示意图

流程1完成后，各模块拥有信息如下：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 用户端A | 密钥管理器 | 文件接收器 | 代码运行器 | 区块链记录 |
| KRA,  KRS pub，  KA | KRA pub,  KRS,  KA, |  |  |  |

表 1 流程1完成后各模块拥有信息

### 用户端发送代码文件给文件接收器的一系列过程

2.1：

用户端A创建一个任务，该任务名称为task\_name，任务创建时间为create\_time，要执行的代码文件为File，File在用户端的路径为task\_file\_path。对task\_name，create\_time和task\_file\_path拼接而成的字符串生成信息摘要HashA1。

用KRS pub对HashA1和KA进行加密，用KA对代码文件和数据文件File进行加密，，然后将自有的公钥KRA pub，用KRS pub加密的HashA1 encrypted和KAencrypted，用KA加密后的文件Fileencrypted，以及KRA pri对各加密内容的签名，一起发送给文件接收器；

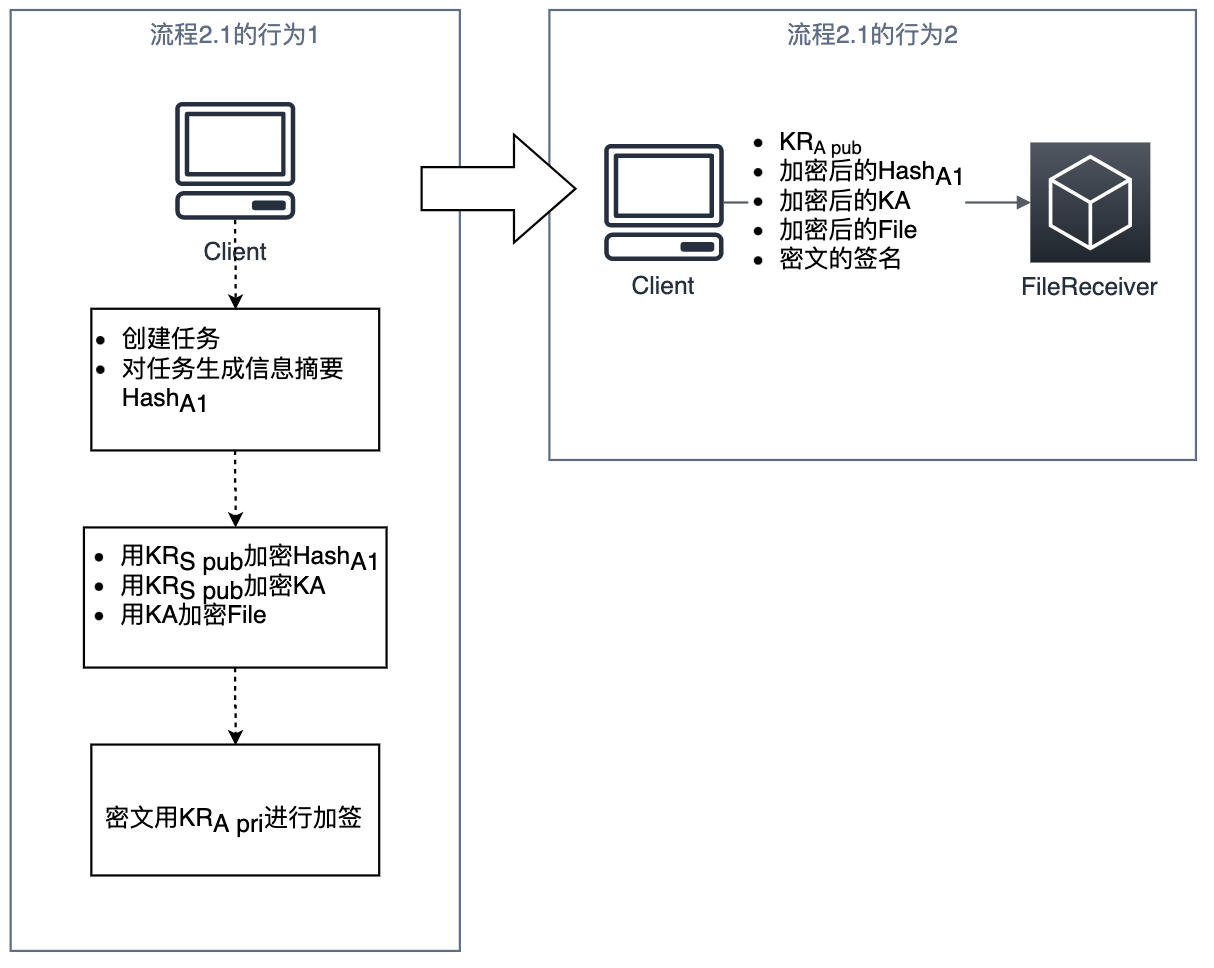


图 6 流程2.1示意图

2.2：

文件接收器先用收到的KRA pub对各加密内容验签。

验签成功后，文件接收器用收到的KRA pub从密钥管理器中获取对应的KRS和KA。文件接收器用KRS pri解密KAencrypted，若解密后得到的KA与从密钥管理器获得的KA相同，那么说明本次从用户端A接收的信息确实由用户端A发出。

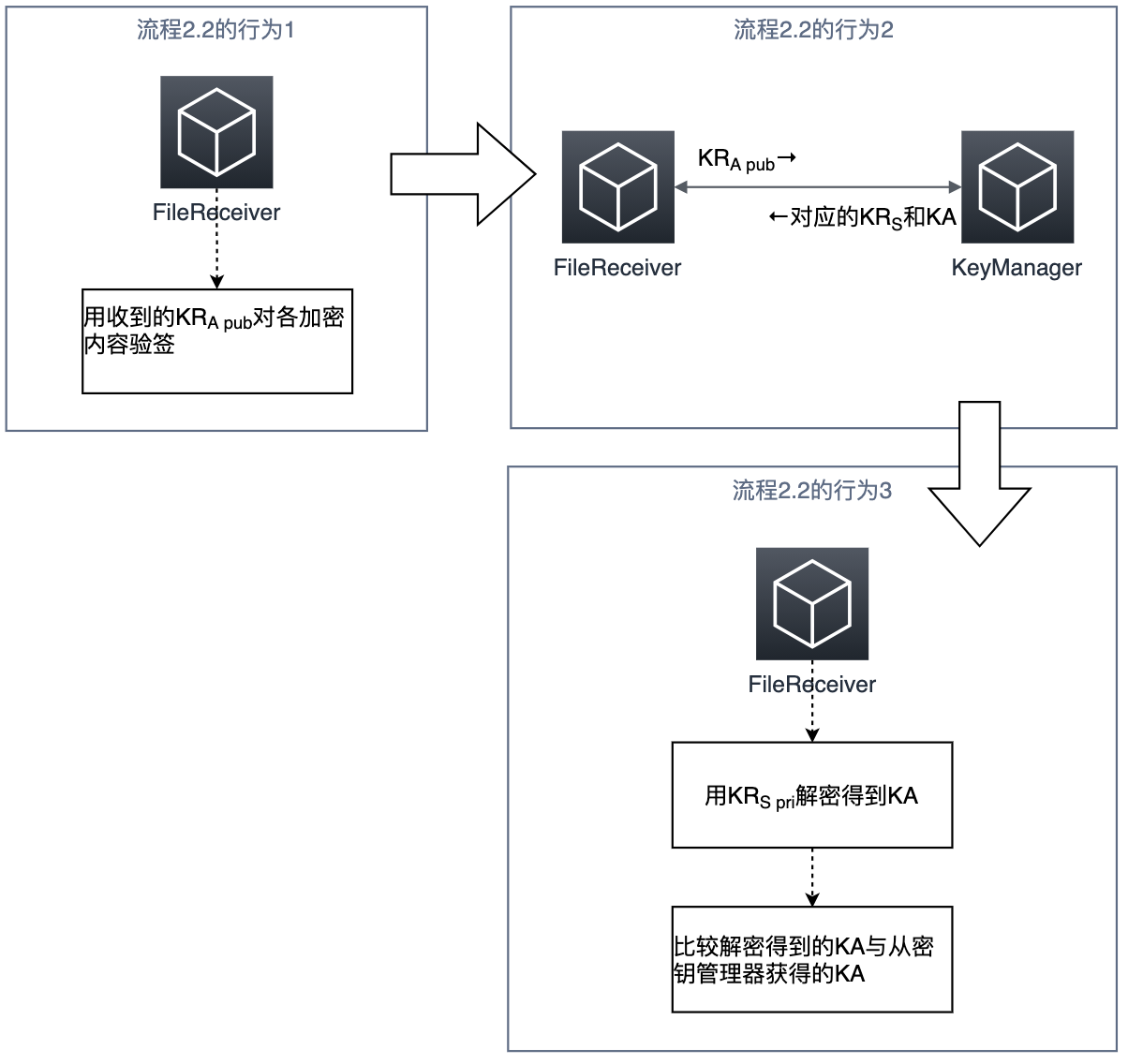


图 7 流程图2.2示意图

2.3：

文件接收器用KRS pri解密HashA1 encrypted得到HashA1，用KA解密Fileencrypted进行解密得到File。并且文件接收器向用户端返回信息，表示成功接收该次任务。

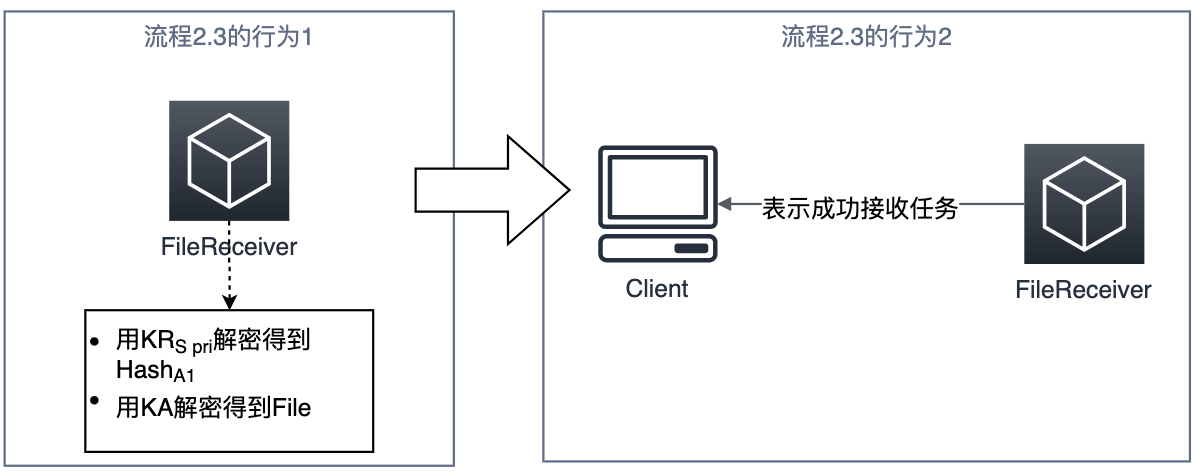


图 8 流程2.3示意图

流程2完成后，各模块拥有信息如下：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 用户端A | 密钥管理器 | 文件接收器 | 代码运行器 | 区块链记录 |
| KRA,  KRS pub，  KA，  File，  HashA1 | KRA pub,  KRS,  KA, | KRA pub,  KRS,  KA,  File,  HashA1 |  |  |

表 2 流程2完成后各模块拥有信息

### 文件接收器发送代码文件给代码运行器的一系列过程

文件接收器将KRA pub，KA，HashA1与File传递到代码运行器。

代码运行器运行File，得到运行结果resultA1，将运行结果用KA加密得到resultA1 encrypted；此外，还记录运行信息run\_infoA1，并对其用KA加密，得到run\_infoA1 encrypted；

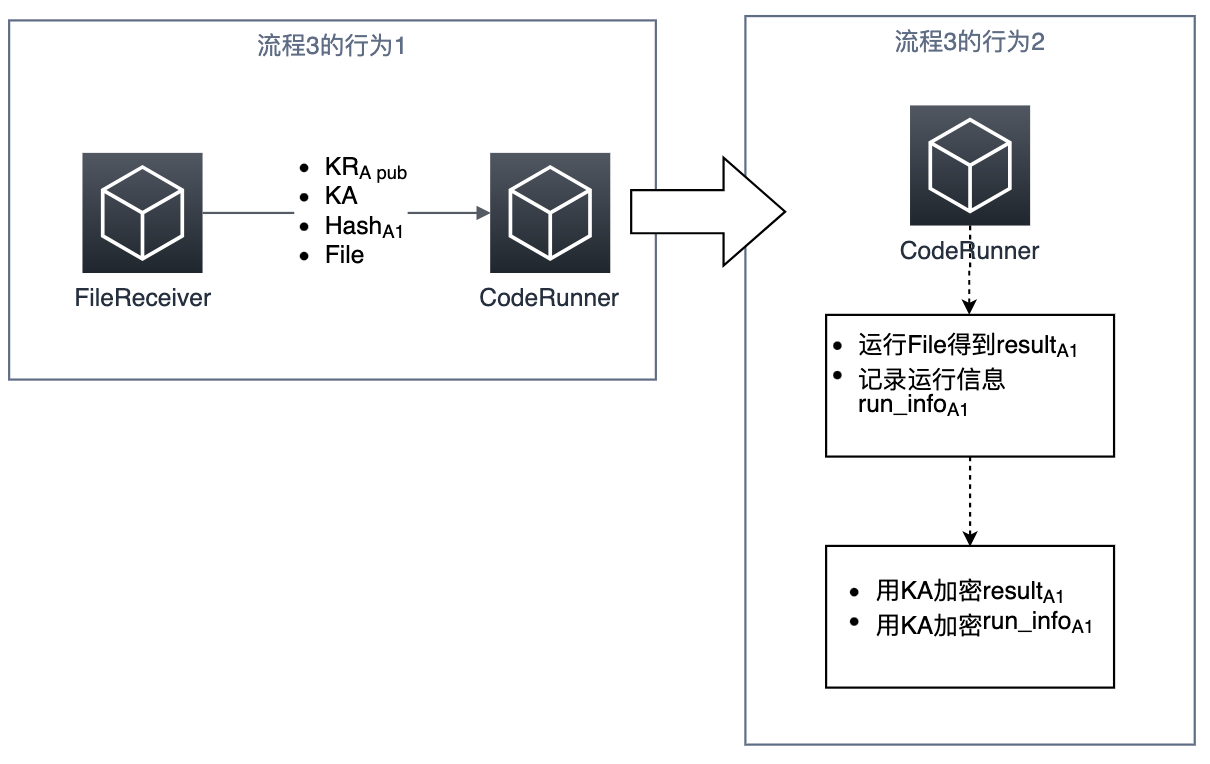


图 9 流程3示意图

流程3完成后，各模块拥有信息如下：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 用户端A | 密钥管理器 | 文件接收器 | 代码运行器 | 区块链记录 |
| KRA,  KRS pub，  KA，  File，  HashA1 | KRA pub,  KRS,  KA, | KRA pub,  KRS,  KA,  File,  HashA1 | KRA pub,  KA,  File,  HashA1,  resultA1 encrypted,  run\_infoA1 encrypted |  |

表 3 流程3完成后各模块拥有信息

### 代码运行器将加密运行结果和运行信息发送给区块链记录的一系列过程

代码运行器将KRA pub，HashA1，resultA1 encrypted和run\_infoA1 encrypted传递到区块链记录。

区块链记录创建新区块记录HashA1，resultA1 encrypted和run\_infoA1 encrypted；并且，维护KRA pub和其每次任务Hash的对应关系，以便后续操作，并且也可基于此对应关系计费。

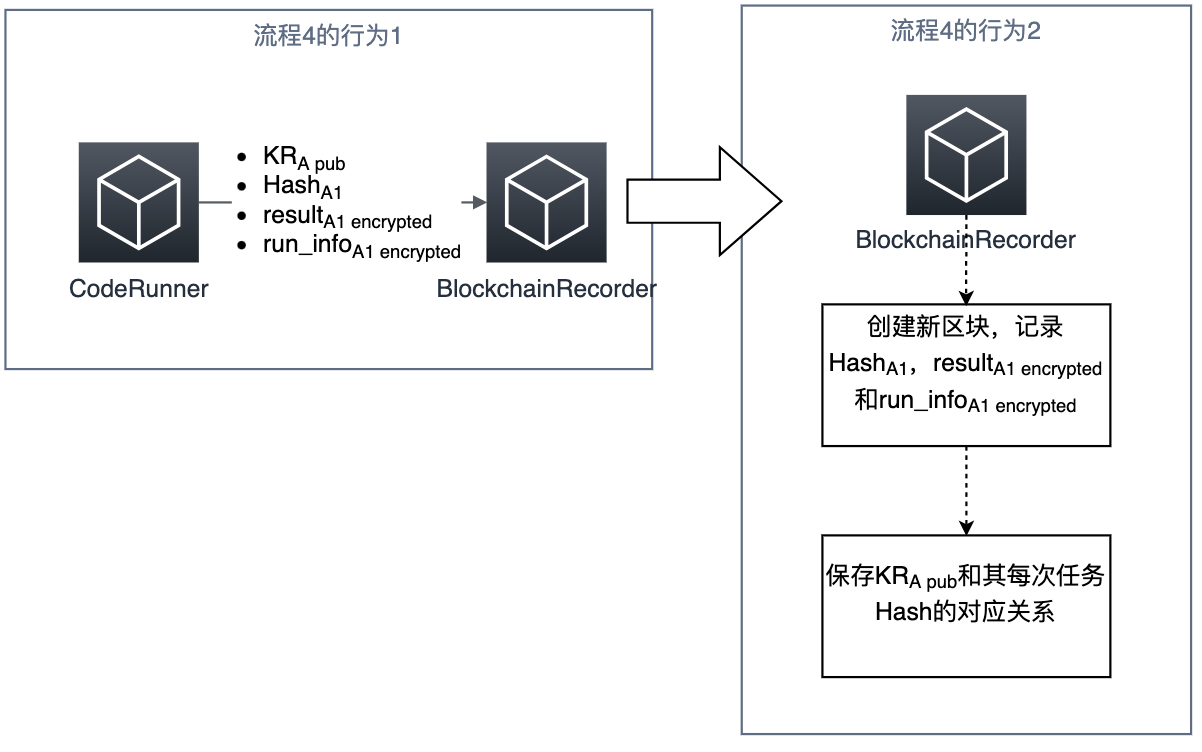


图 10 流程4示意图

流程4完成后，各模块拥有信息如下：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 用户端A | 密钥管理器 | 文件接收器 | 代码运行器 | 区块链记录 |
| KRA,  KRS pub，  KA，  File，  HashA1 | KRA pub,  KRS,  KA, | KRA pub,  KRS,  KA,  File,  HashA1 | KRA pub,  KA,  File,  HashA1,  resultA1 encrypted, run\_infoA1 encrypted | KRA pub,  HashA1,  resultA1 encrypted,  run\_infoA1 encrypted, |

表 4 流程4完成后各模块拥有信息

### 用户端向区块链记录请求信息的一系列过程

用户端A将 HashA1发送给区块链记录，以查询记录着任务A1信息的区块。

区块链根据自身维护的KRA pub和其每次任务Hash的对应关系，通过HashA1得到相应的KRA pub，用KRA pub向密钥管理器请求对应的KRS pri，而后用KRS pri将记录着HashA1的区块加签，再将记录着HashA1的区块和签名一同返回给用户端A。

用户端A可用KRS pub验签。然后用KA解密区块中的resultA1 encrypted和run\_infoA1 encrypted，得到resultA1和run\_infoA1；

注：用户端A也可以请求同步整个区块链记录到用户端A；

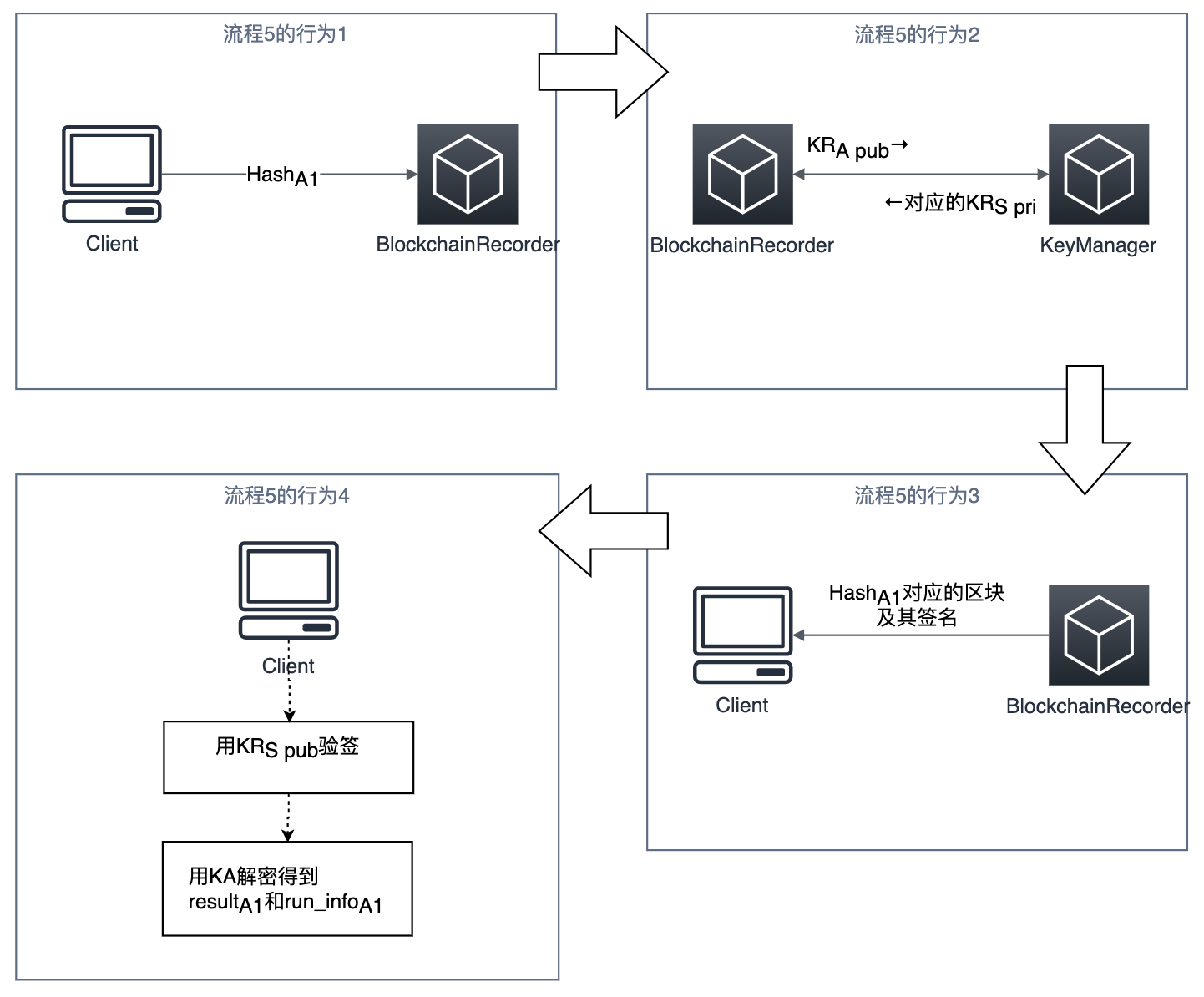


图 11 流程5示意图（用HashA1请求）

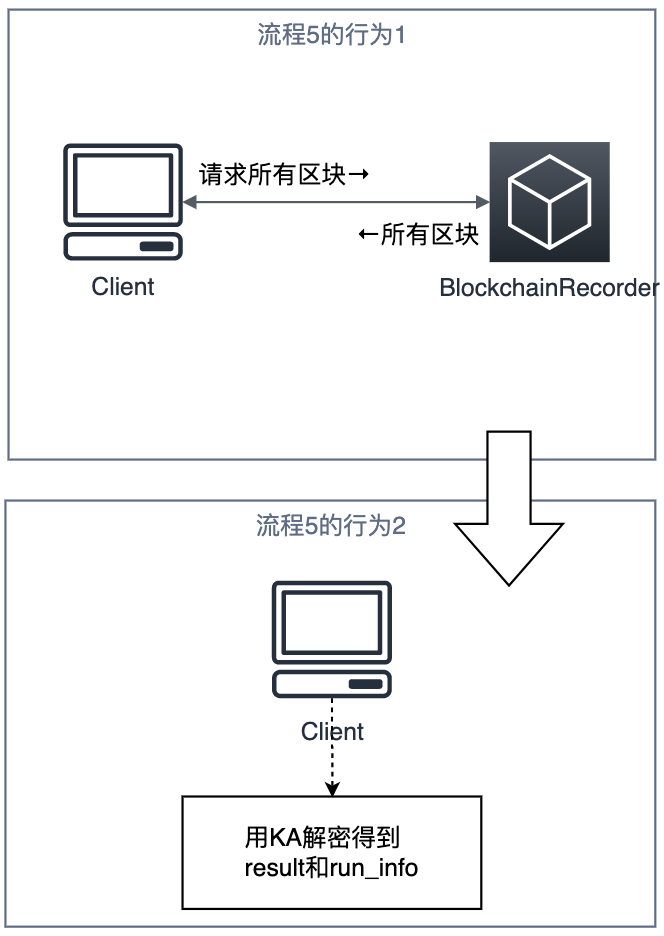


图 12 流程5示意图（请求整个区块链）

流程5完成后，各模块拥有信息如下：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 用户端A | 密钥管理器 | 文件接收器 | 代码运行器 | 区块链记录 |
| KRA,  KRS pub，  KA，  File，  HashA1,  resultA1,  run\_infoA1, | KRA pub,  KRS,  KA, |  |  | KRA pub,  HashA1,  resultA1 encrypted,  run\_infoA1 encrypted, |

表 5 流程5完成后各模块拥有信息

# UML建模

图 3 基于TEE与区块链的隐私保护计算平台 整体架构示意图 中列出的各个模块（小直角矩形），即是需要实现的类。各个模块名即是各个类名。

## 时序图

基于前述整体架构和流程详解，可以给出本平台运行的时序图。

需注意此处的标号与序号仅代表该时序图中事件的排列，与三、整体架构与流程中的标号和序号没有对应关系。

并且，图中省略了类保存信息、加密、解密和验签等默认自身进行的行为。

为能清晰阅览，令时序图单独占一页篇幅，请见下页。

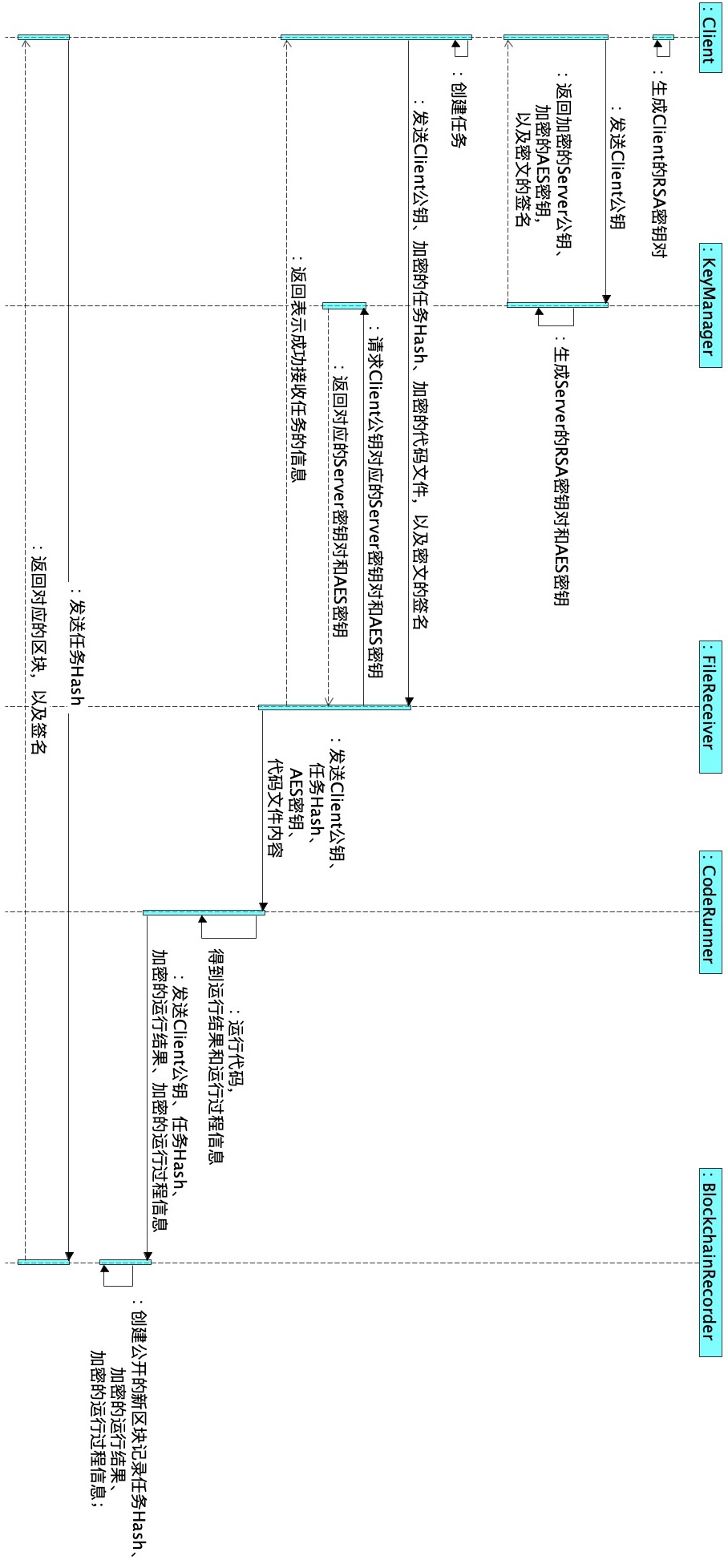


图 13 时序图

## 类的详解

# 参考文献

1. Sabt M, Achemlal M, Bouabdallah A. Trusted execution environment: what it is, and what it is not[C]//2015 IEEE Trustcom/BigDataSE/ISPA. IEEE, 2015, 1: 57-64.
2. Yaga D, Mell P, Roby N, et al. Blockchain technology overview[J]. arXiv preprint arXiv:1906.11078, 2019.
3. Zheng Z, Xie S, Dai H N, et al. Blockchain challenges and opportunities: A survey[J]. International Journal of Web and Grid Services, 2018, 14(4): 352-375.
4. Rivest R L, Shamir A, Adleman L. A method for obtaining digital signatures and public-key cryptosystems[J]. Communications of the ACM, 1978, 21(2): 120-126.
5. Daemen J, Rijmen V. Reijndael: The Advanced Encryption Standard[J]. Dr. Dobb's Journal: Software Tools for the Professional Programmer, 2001, 26(3): 137-139.
6. 余凯, 贾磊, 陈雨强, 等. 深度学习的昨天, 今天和明天[J]. 计算机研究与发展, 2013, 50(9): 1799.
7. 吴吉义, 平玲娣, 潘雪增, 等. 云计算: 从概念到平台[J]. 电信科学, 2009, 25(12): 23-30.
8. McKeen F, Alexandrovich I, Anati I, et al. Intel® software guard extensions (intel® sgx) support for dynamic memory management inside an enclave[M]//Proceedings of the Hardware and Architectural Support for Security and Privacy 2016. 2016: 1-9.
9. Shen Y, Tian H, Chen Y, et al. Occlum: Secure and efficient multitasking inside a single enclave of intel sgx[C]//Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems. 2020: 955-970.
10. 沈鑫, 裴庆祺, 刘雪峰. 区块链技术综述[J]. 网络与信息安全学报, 2016, 2(11): 11-20.
11. Van Flymen D. Learn blockchains by building one[J]. The fastest way to learn how Blockchains work is to build one, 2017.
12. Calderbank M. The rsa cryptosystem: History, algorithm, primes[J]. Chicago: math. uchicago. edu, 2007.

# 附录 A. 时序图

# 致谢

致谢应以简短的文字对课题研究与论文撰写过程中曾直接给予帮助的人员(例如指导教师、答疑教师及其他人员)表达自己的谢意，这不仅是一种礼貌，也是对他人劳动的尊重，是治学者应当遵循的学术规范。内容限一页。