# 基于以太坊的个人信息管理系统--EthKee技术报告

1. **项目背景**

随着互联网和智能手机应用的普及，大量的数字化个人信息已经成为人们最重要的个人资产，如个人联系方式、通讯录以及在各种APP中的用户名和密码等。现有用于个人信息管理的工具可以分为独立软件和云服务两大类，前者将用户个人信息数据保存在本地，如用于密码管理的KeePass[1]；后者不仅将用户个人信息数据保存在本地，同时还会将数据同步给服务提供者，如谷歌通讯录[2]和苹果iCloud Keychain[3]。独立软件的优势在于不用担心个人信息的泄露和滥用，但需要用户自己解决设备故障、设备间数据同步和信息分享问题。云服务的优势在于不会因为个人设备故障导致信息丢失，并且能够方便地实现设备间数据同步和信息分享，但存在个人信息被服务提供商滥用或泄露的风险。因此，现实中具有对于兼具以上两类个人信息管理工具优势的数据存储方式的需求。

Keeweb是一个能兼容Keepass的、免费的、跨平台的密码管理器应用。它不要求特定的服务器或额外的资源，能为用户管理各类信息。Keeweb使用特定加密算法对用户数据进行加密，并且提供如One Drive、Google Drive等存储方式来存储用户数据。本项目是在Keeweb项目的基础上进行了存储的扩展，使得个人信息能被保存到以太坊的智能合约中，并可通过应用界面对数据进行管理。

* 1. 项目实施主要目的

1. 实现将个人信息数据的去中心化存储

为解决现有个人信息管理工具存在的问题，在Keeweb现有功能的基础上，提出基于以太坊的个人信息管理系统。系统中数据的存储服务将不依赖于特定服务商，由此在一定程度上保证了数据存储的独立性和安全性。而以太坊网络中数据的去中心化存储、不可篡改、可追踪等特性能在一定程度保证个人信息数据的安全存储，从而提高系统可靠性和可用性。

1. 探索区块链技术的应用模式

区块链是基于P2P网络、共识机制和加密算法等技术的新型应用模式，具有去中心化、数据难以篡改和可追溯等特点。区块链技术的研究是当前学术界和工业界共同关注的一个热点问题，相关研究工作不仅包括共识算法[4]和智能合约[5]等区块链实现技术，还包括如何将区块链技术应用于各个领域，如物联网[6]、金融[7]和医疗[8]等。作为一种新兴的技术，其应用模式还没有公认的成熟方案，有待于进一步研究。本系统是将区块链技术应用于数据存储的一种探索，并希望通过系统的开发找到合理的区块链技术应用模式。

* 1. 相关背景知识
     1. 以太坊

区块链技术是综合共识机制、加密算法、智能合约、P2P网络等技术的新兴技术框架，应用范围广泛，但目前尚没有公认的成熟应用模式。区块链技术现有的典型应用平台包括以太坊、超级账本等。其中，以太坊作为最主流的去中心化应用开发平台，现有的开发框架和开发语言都相对较成熟，并且支持智能合约。本文构建的个人信息管理系统基于以太坊实现。

以太坊网络是由运行着以太坊客户端的节点组成的P2P网络，结构如图1所示。以太坊客户端中存储着以太坊区块链和以太坊的状态，同时也运行着EVM（Ethereum Virtual Machine）。以太坊区块链是按链表结构组织的区块集合，区块中存储着时间戳、随机数、前一区块数据的Hash值以及大量的交易。交易中记录着以太坊账户发生的状态变化。以太坊状态是区块链中特定时间点上所有账户和数据的快照[9]。账户分为外部账户和合约账户两类，账户地址为20字节的16进制字符串。外部账户类似现实世界的银行卡账户，其中存储着账户余额等信息。合约账户即智能合约，存储着合约数据和合约代码。EVM是能执行交易、修改以太坊状态的状态机，由它可查出合约中所有已知数据结构的数据当前状态。



* + 1. 智能合约

智能合约是一段运行在以太坊网络中的脚本，可通过变量赋值、函数调用来存储、操作数据。智能合约的执行过程如下：（1）以太坊账户创建交易，将合约函数名、函数参数都存储到该交易中，并将其发送到以太坊网络；（2）网络中的节点收到该交易后将执行交易中的合约函数；（3）当大多数节点得到的执行结果一致时即认为该交易有效，将该交易存入到以太坊区块链中。至此，可认为智能合约被成功执行，合约数据被修改。

以太坊智能合约中的数据有四个重要特点：（1）公开性。任何连接到以太坊网络的节点均可自由读取以太坊区块链中的所有数据；（2）数据管理成本较高。调用智能合约来存储、操作数据会消耗交易发起者一定的以太币。同时，以太坊网络中的所有节点都需要执行该合约，消耗了大量的计算资源；（3）数据操作是异步的。执行合约过程中需要向网络发送交易、产生新区块。只有在这些过程完成后数据操作才算成功；（4）不易丢失、难以篡改。智能合约中的数据存储在以太坊网络中，使得数据难以篡改。以太坊网络中每个节点均存储着智能合约数据副本，保证了数据不易丢失。

1. **项目研究内容**

本项目在Keeweb现有的基础上，为用户提供了将文件保存到以太坊的解决方案。将个人信息数据通过以太坊的智能合约保存到区块链中，不仅能够实现数据的可靠存储、同步和分享，还能够避免数据泄露和数据滥用的问题。我们对在以太坊区块中存储和访问数据的方式进行了分析，设计了用于管理个人信息数据的智能合约UPC(User Profile Contract)，并将其部署到以太坊网络中。本项目的研究内容主要包括两点：

（1）将加密数据存储到以太坊智能合约中。智能合约数据的公开性要求我们必须在存储数据之前对数据进行加密处理。Keeweb中已经实现了对数据的加密，但是，需要解决如何将这些加密数据传输到以太坊网络中。同时，由于合约中函数的调用是异步的，还需要考虑应用对智能合约中的函数执行状态的监控，并据此设计合理的业务逻辑，从而保证用户体验。

（2）研究智能合约存储数据的合理数据格式设计以及相应数据操作的成本。以太坊智能合约中的数据类型较简单，并且是对以太坊网络中所有节点可见的，因此要求智能合约中的数据都必须是结构相对简单、加密的。同时智能合约中需要反应出个人信息数据的语义，并且要便于应用调用，因此需要定义合理的数据模式和数据结构。同时，在对数据进行操作时也要考虑其所消耗的eth成本、相应操作的响应时间代价等。在整体设计中，应尽量将对数据的复杂操作放到应用层面完成，而在智能合约中只进行基本的数据操作。

1. **技术路线**
   1. 需求分析

在日常生活中，用户需要管理很多重要的信息，如银行卡密码、社交账号密码、重要客户信息等。由于这些信息均比较私密，对其存储时必须进行加密，并且要保证信息不被泄露。同时，用户要求这些信息必须总是可访问的，不会因为存储介质或第三方服务商发生的问题导致信息丢失。并且，也需要保证，用户访问到的信息数据总是正确的，不能被篡改。根据需要，该系统添加以太坊作为存储方式后，应该具备以下功能：

新建以太坊智能合约；

向智能合约中存储个人信息数据；

查询出用户合约中存储的信息文件；

对合约数据进行操作；

* 1. 设计思路

根据需求分析，首先设计智能合约的数据结构和数据操作方法。由于keeweb帮助存储的是一些关键的数据信息，并且考虑到智能合约数据存储、操作需要付费，合约中可使用较简单的数据结构——Mapping来存储数据。为方便对数据的查询和操作，合约中需要使用相应变量来记录信息文件名、文件数量。

同时，还需要设计应用与以太坊智能合约的连接方法，具体的传值编码方式等。应用首先需要先获取metamask提供的web3 provider，再由它获取到以太坊账户。其后，可以使用truffle-contract库可以实现从智能合约的abi json文件、合约地址中获取到合约实例，并由此直接访问智能合约中定义的各种变量、方法。在向合约中存储、查询数据时，还需要注意应用和合约之间传输数据时数据的编码和转换，保证数据能以最小的代价存储到以太坊中，并需便于之后的数据操作。

* 1. 主要技术特点

1. 应用端开发

前端框架采用Backbone[10]框架。Backbone.js为复杂WEB应用程序提供模型(models)、集合(collections)、视图(views)的结构。其中模型用于绑定键值数据和自定义事件；集合附有可枚举函数的丰富API；视图可以声明事件处理函数，并通过RESRful JSON接口连接到应用程序。

通过Truffle-contract[11]与智能合约交互。Truffle-contract对以太坊的智能合约做了更好的抽象，相比于Web3.js，使用Truffle-contract操作智能合约更加方便。Truffle-contract具有以下特色：

（1）同步的交易：可以确保在交易生效之后再继续执行其他操作；

（2）返回Promise：每个封装的合约函数会返回Promise，可以对它进行.then操作，避免了回调地狱（callback hell）问题；

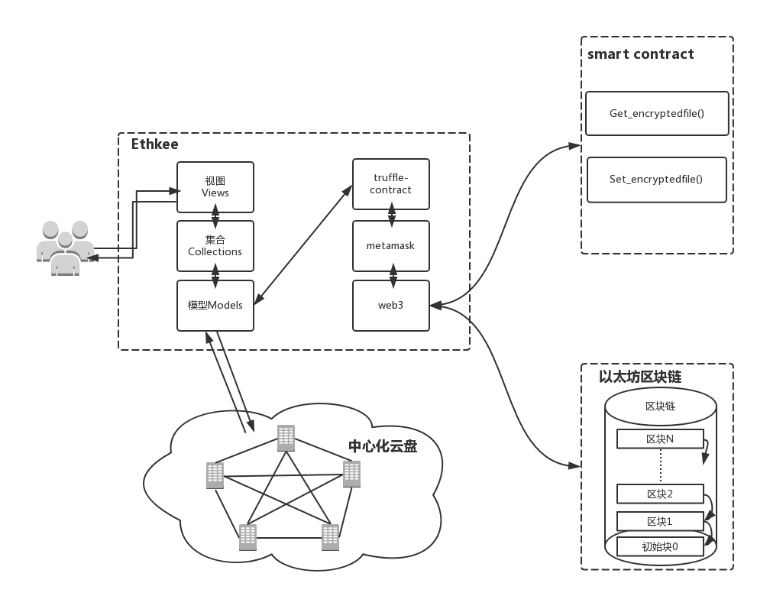
（3）为交易提供了默认参数，例如from或gas；

（4）为每个同步的交易返回logs、交易receipt和交易hash。

1. 智能合约开发

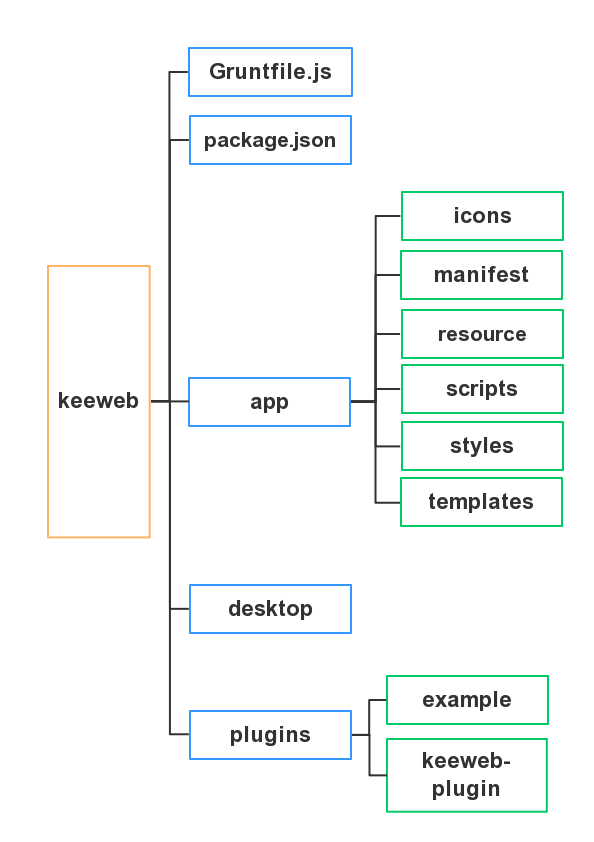
使用solidity开发智能合约，开发平台为Remix IDE。Remix IDE操作方便且功能较全，适合智能合约的编写和测试。开发和测试中使用的以太坊钱包为Meta mask。智能合约的测试环境为Java虚拟机、Ganche测试工具、以太坊私链集群、以太坊测试链。其中，以太坊私链集群的各种参数与以太坊公链基本相同，以实现最真实的模拟测试。以太坊测试链中使用了Rinkeby、Ropsten两个使用较稳定的测试网络。测试完成的智能合约使用Truffle框架进行编译和部署，得到智能合约的abi json文件和合约地址。

1. **体系结构与功能**
   1. 系统框架



系统的架构如上图所示。整个系统主要分为以太坊区块链、智能合约、前端和云存储服务四个部分。前端由Backbone.js框架搭建，使用基于web3.js的Truffle-contract框架与智能合约交互，并使用MetaMask插件向以太坊区块链网络发送交易。以太坊区块链由6以太坊客户端节点组成，客户端使用的是Geth。另外，Ethkee可以将原本云盘上的数据加载到以太坊区块链中。

* 1. 前端开发



上图为前端文件的目录结构。在根目录下，Gruntfile.js用于配置grunt任务，Package.json用于配制项目所需要的依赖，App/用于存放web前端代码，Desktop/用于存放桌面端程序，Plugins/用于存放keeweb插件模版和插件例子。

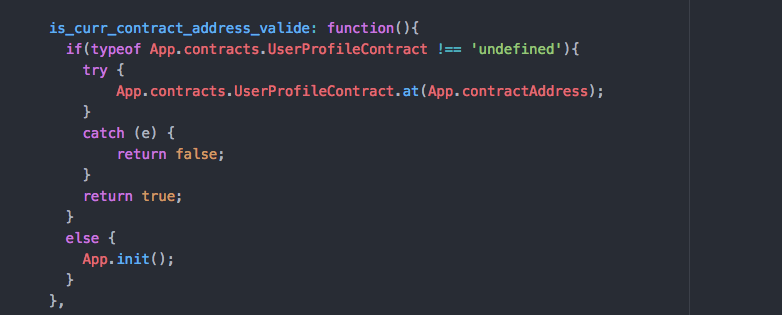
icons/存放图片文件，manifest/存放图片配置文件，scripts/存放核心js文件，templates/用于存放前端HTML片段的模版文件。

尤其重要的文件是/app/sricpts/storage/storage-eth.js。该文件封装了Ethkee与以太坊和以及智能合约交互的核心方法，具体如下：

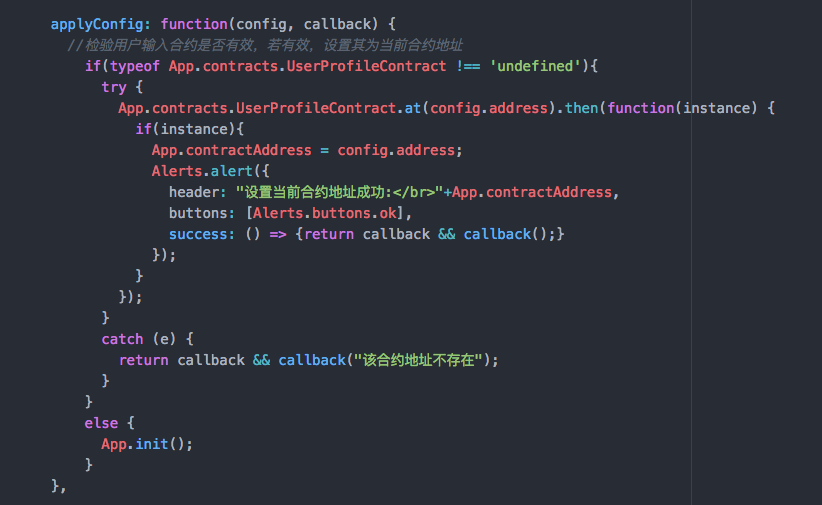
（1）initContract：负责创建新的合约实例



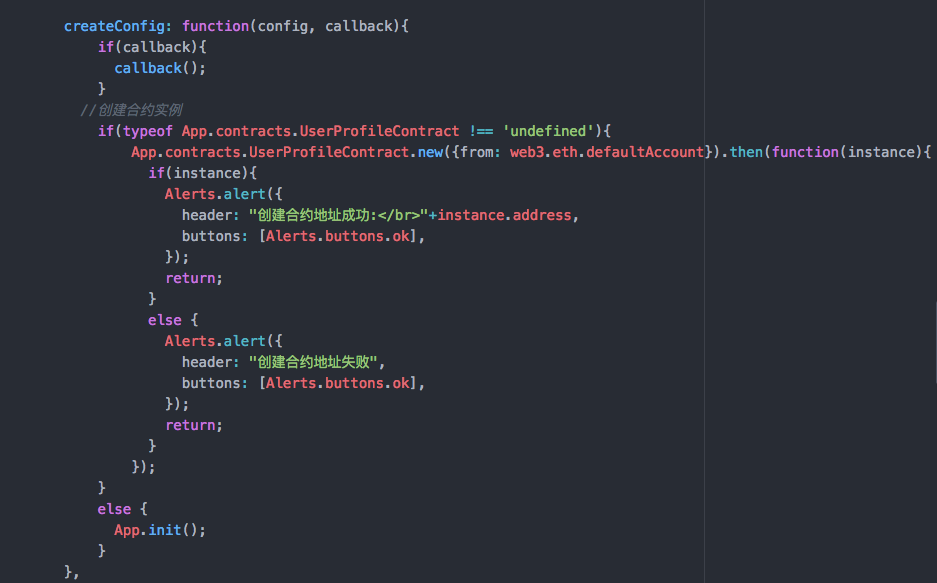
（2）is\_curr\_contract\_address\_valide：负责检测当前合约实例是否有效



（3）applyConfig：负责设置合约实例的地址



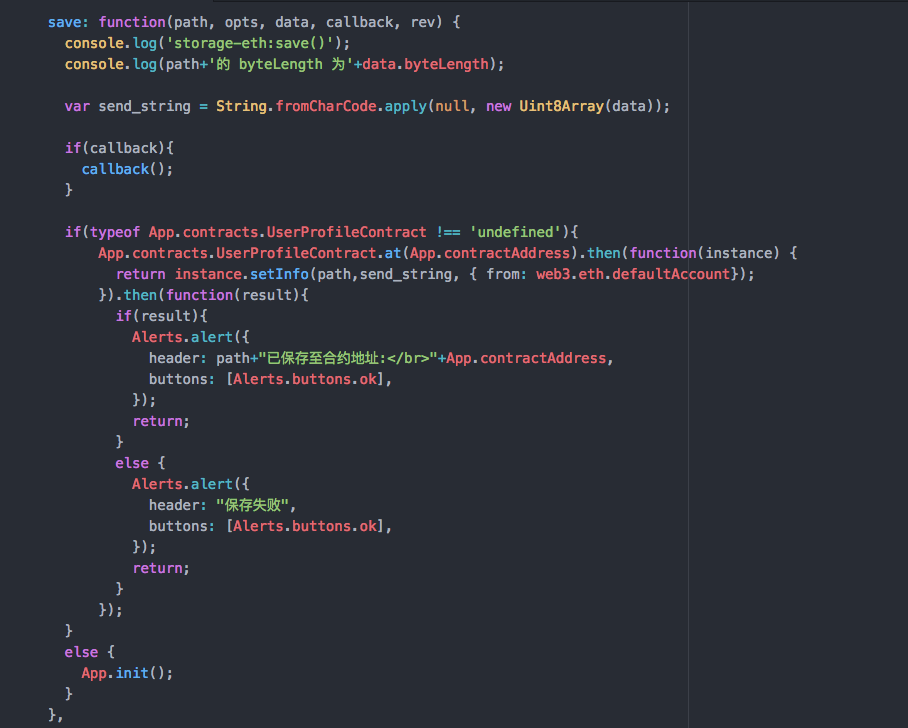
（4）createConfig：负责创建新的合约实例



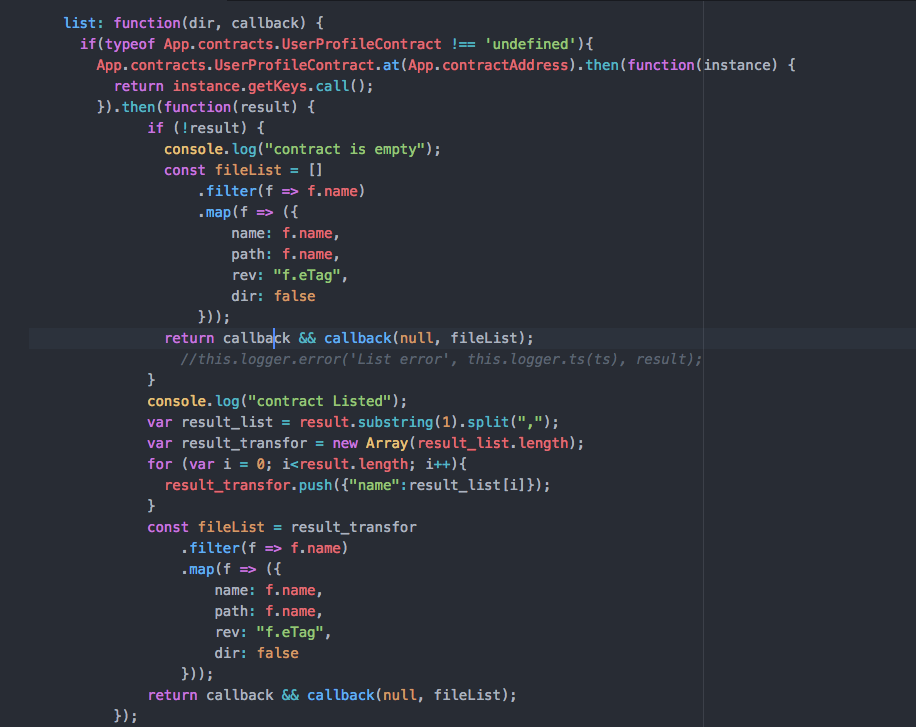
（5）load：负责根据文件名加载合约中的加密文件



（6）save：负责将加密文件保存到合约实例中



（7）list：负责列出合约实例中所有加密文件的文件名



* 1. 以太坊智能合约开发

合约可以实现将应用传递到以太坊网络中的数据存储到智能合约中，并对数据进行更新。同样，应用可以通过合约的函数或者直接读取合约的public变量的方式读取用户存储到以太坊智能合约中的数据。具体的变量和函数如下：

1. 合约变量
2. owner：用于记录此用户合约的所有者地址，类型为address.
3. RecordMapping：用于存储用户信息，格式为（key,value），其中key和value类型均为string。
4. Keys:用于记录此用户合约中存储的记录的所有key值，类型为字符串数组。初始长度设为100，也可改为更大的值。

由于智能合约在初始化时

1. 合约函数
2. UserProfileContract()：合约构造函数，在合约创建时被调用。在函数中设置变量owner等于msg.sender，即创建合约的账户。
3. setRecord(string \_key, string \_value) onlyOwner：更新函数，用于插入、更新、删除用户记录。

输入参数：

1. String – 用户信息文件的key值，即文件名。
2. String – 用户信息文件的序列化值，即加密的.kdbx文件序列化为字符串。

Modifier:

onlyOwner：用于限制方法的调用者只能是合约中的owner。

示例：

setRecord(“juliaInfo.kdbx”,”3645 2687 3654 1455 9854”)；//该记录将被插入到RecordMapping中。

setRecord(“juliaInfo.kdbx”,”3645 2687 3654 1455 9854”,{from:eth.coinbase})；//直接在geth客户端调用该函数时，需指定调用函数的账户地址。由于modifier的限制，from的参数值必须等于变量owner的值。也就是说，只有用户合约的拥有者才能对用户信息文件进行存储、更新和删除。

注意：

在插入用户记录时，EVM会先检查RecordMapping中是否已经存在该key。若存在则直接使用\_value覆盖旧值，若不存在则新建。在参数相同的情况下，覆盖旧值消耗的Gas要远少于新建键值对。

1. getRecord(string \_key) public constant returns (string)：获取指定文件的内容。

输入参数：

1. String – 用户信息文件的key值，即文件名。

返回值：

String – 用户信息文件的序列化值，即加密的.kdbx文件序列化为字符串。

示例：

getRecord(“juliaInfo.kdbx”);//读取文件名为juliaInfo.kdbx的文件内容

返回的结果为：”3645 2687 3654 1455 9854”；

注意：

读取智能合约中数据的函数是直接由EVM执行的，并不需要发送交易调用执行，因此也不消耗Gas。若无其它限制，该方法可由网络中任何账户调用。

1. geKeys() public constant returns (string)：获取当前合约中存储的所以用户文件的文件名。

返回值：

String – 所有用户信息文件的文件名拼接起来的字符串，文件名之间使用逗号分隔。

示例：

geKeys();//读取所有文件名

返回的结果为：”,juliaInfo.kdbx,bobInfo.kdbx,timInfo.kdbx”；

注意：

文件名应该从字符串的第二个位置开始读取。

1. strConcat(string \_a, string \_b) internal returns (string)：将两个文件名拼成一个字符串，文件之间以逗号隔开。

输入参数：

String – 第一个文件的文件名。

String – 第二个文件的文件名。

返回值：

String – 拼接成的文件名字符串。

示例：

strConcat(“juliaInfo.kdbx”,”bobInfo.kdbx”);//连接当前用户的两个文件名

返回的结果为：”,juliaInfo.kdbx,bobInfo.kdbx”；

注意：

每个文件名前都有一个逗号，用于区分。同时，此函数为内部函数，不能从合约外部调用。

1. stringsEqual(string storage \_a, string memory \_b) internal returns (bool)：判断两个字符串是否相同。

输入参数：

String – 第一个字符串。

String – 第二个字符串。

返回值：

Bool – true表示两个字符串相同，false表示两个字符串不同。

示例：

stringsEqual(“juliaInfo.kdbx”,”bobInfo.kdbx”);//两个文件名,返回的结果为：”false”；

stringsEqual(“juliaInfo.kdbx”,”juliaInfo.kdbx”);//两个文件名,返回的结果为：”true”；

7) isExisted(string \_key) internal returns (bool)：判断指定的文件名是否已经在文件名列表中。

输入参数：

String – 文件名。

返回值：

Bool – true表示该文件名已存在文件名列表中，false表示该文件名不存在。

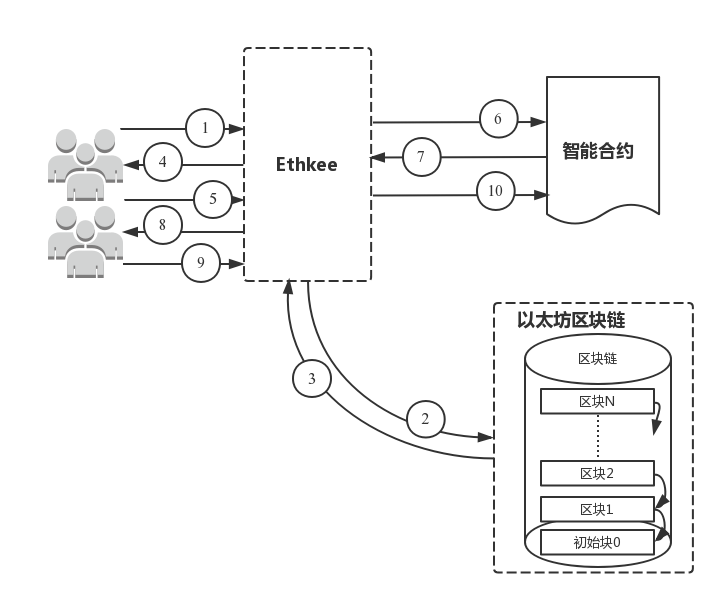
示例：

isExisted(“juneInfo.kdbx”);//文件名,返回的结果为：”false”；

isExisted (“juliaInfo.kdbx”);//文件名,返回的结果为：”true”；

8）kill():销毁函数，释放存储空间。执行后，合约中的所有变量值被清空。

* 1. 业务流程



业务流程如上图所示。该图描绘了用户如何使用Ethkee将加密文件存储到以太坊智能合约，以及如何将加密文件从以太坊智能合约读取的流程：

（1）拥有以太坊账户的用户在Ethkee应用上点击新建合约地址

（2）Ethkee通过metamask将创建合约地址的交易发送到区块链

（3）区块链返回新建的合约地址

（4）Ethkee将该地址在前端显示给用户

（5）用户保存该合约地址后，键入该地址并保存

（6）Ethkee查询该合约实例中的文件

（7）Ethkee加载合约中的加密文件

（8）Ethkee解密文件，并显示该合约实例中的个人信息文件。如果没有，则显示“Not found suitable file”

(9)用户选择文件并打开，点击保存按钮

(10)Ethkee将要保存的文件放入智能合约中

1. **测试及分析**

应用从界面中获取到用户填写的个人信息，并转化为临时数组，最后将临时数组转化为字符串传递给智能合约中的函数。

* 1. 用例测试环境配置

1. 测试使用的软件版本

以太坊钱包：Metamask 4.9.3

以太坊客户端：geth v1.8.12-stable

1. 以太坊测试网络配置

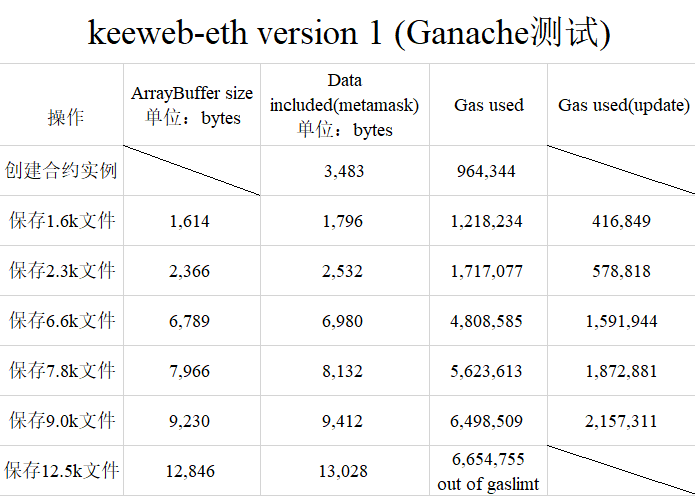
本次测试在Ganache、Ropsten、Rinkeby、以太坊私链集群上完成，其中Ganache的平均挖矿时间认为是0s，其它三种测试网络的平均挖矿时间分别为12s,15s,15s。

为保证应用可以实现与以太坊私链网络的连接，需要对RPC相关参数进行设置，本次测试中，启动以太坊私链节点的启动命令为： geth --datadir ./data/01 --networkid 2804 --ipcdisable --port 30060 --rpcport 50060 –rpcaddr 0.0.0.0 –rpccorsdomain ‘\*’。由此可保证，应用可以正常访问以太坊节点，并使用RPC调用接口。

* 1. 用例测试数据分析

1. 用例文件数据操作分析

Gancache中设置的块gas limit为6,654,755，以太坊公链为7,992,222，两者相差不多，因此两者能存储的最大数据量也很接近。以下是我们在ganache中测试不同大小的测试数据进行存储和更新时消耗的gas、发送RPC调用请求时包含的数据量以及文件对应的ArrayBuffer。即使切换到公链上，由于使用的的钱包都是Metamask，并且测试网络处理数据的方式基本相同，因此以下这4个数据值基本上是与在Ganache上一样的。测试结果如下：



根据以上数据可知，存储数据消耗的Gas是随着存储的数据量的增大而增大的。同时，可以观察到，同样的数据量，如果是更新操作，其消耗的Gas约为存储数据的1/4~1/3。这是由于在智能合约中，向一个Sstore(智能合约变量的存储单位)存储数据消耗的Gas是更新数据消耗的Gas的1/4。由于本测试中测试用例的数据量比较大，跨越多个Sstore，因此，得到的结果是在正常范围之内。

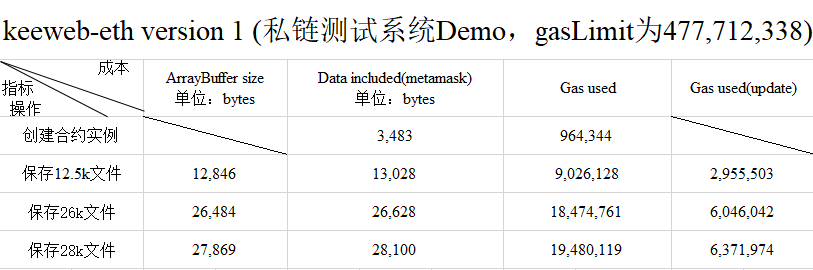
下图是根据上表中数据得到的存储、更新数据时消耗的Gas与数据形成的Array Buffer的函数关系。从图中可以看出，消耗的Gas与存储、更新的数据量成正比。

(2) 用例文件操作响应时间分析

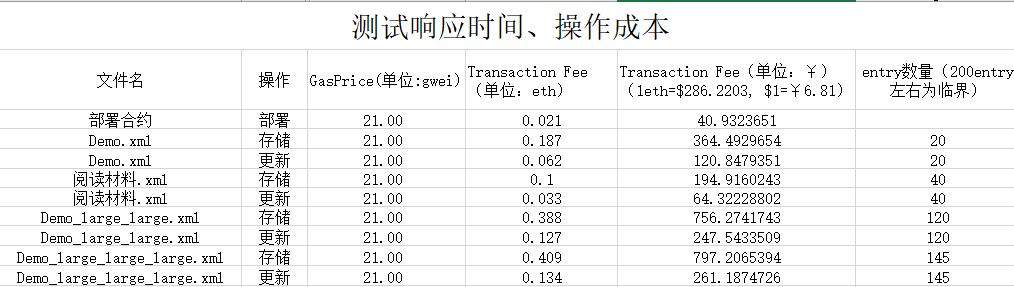
经过测试，由于上述文件大小的差距不大，且各个测试网络的平均挖矿时间相差不多，因此，平均响应时间基本相同。存储时平均响应时间为32.2s,更新时平均响应时间为23s。相对来说，更新数据的平均响应时间略低于存储数据的平均响应时间。

(3)通过RPC调用最大能传输的用例文件大小分析

由上述测试网络参数可知，当数据量达到大约12.5k bytes时，将会出现out of gas的错误，数据无法被成功存储。由此，我们无法分析RPC调用允许传输的最大数据量。为了找到RPC的阈值，我们采用gas limit为4294967295的以太坊私有链集群进行测试，此时应用可存储的文件数据量变大，得到的测试结果如下：



最终得到的临界值为：32k。也就是说，当用户数据达到32k时，Meta Mask将报oversized data错误。通过研究Geth 客户端源码，可以看到，其中设定了RPC传输数据量的阈值为32k bytes. 由于RPC的最大数据量是由Geth官方客户端源码限制，在不改变客户端代码的情况下，即使是重新修改以太坊公链的Gas Limit，用户所能存储的数据量也只有32k，所存储的正常大小的entry也仅有200个。在此限制下，我们探索，使用ethKee操作数据的成本，得到的数据如下：



从上述数据可以看到，用户部署合约将花费40元人民币，而向其中存储大约145条entry将花费约800元人民币。其花费的钱基本与用户数据量成正比，可以推测，用户在200entry时将消耗1000元人民币。因此，对于一般用户来说，使用ethKee存储数据的成本是较高的。

* 1. 总结

（1）用户向智能合约中存储的数据应该是关键的、少量的。对于一般的用户来说，合约的部署成本、用户向智能合约中存储数据的成本、用户对数据的操作成本均较高。对于安全性要求并不是很高的数据，如微博账户密码、百度用户密码等，用户可以选择存储在云盘或者本地数据库中。对于安全性要求很高且数据量不大的数据，如银行卡密码等，用户可以存储到以太坊智能合约中。

（2）应用后台向智能合约传递参数时应该尽量选择适当的类型。后台的用户数据类型与智能合约中的函数参数类型不同，一般需要进行转化。在转化过程中可能会涉及到数据的编码等，使用恰当的数据类型和数据编码方式能使得一个个人信息文件所需花费的响应时间、消耗的以太币大大降低。

1. **总结**

互联网及手机应用的普及为人们带来大量的数字化个人信息，而现有的个人信息管理工具存在着数据难以分享、易被泄露等问题。区块链技术具有去中心化、数据难以篡改、可追溯等特点，能为解决个人信息管理中的问题提供新思路。本文提出的基于以太坊的个人信息管理解决方案将个人信息数据存储到以太坊智能合约上，并由系统节点对数据进行组织和维护，由此可实现个人数字化信息的可靠存储、安全共享，并确保用户对数据访问权限的控制。该方案的实现基于较完善的密码管理项目Keeweb，主要针对的是对于数据安全性有很高要求的用户，存储的是关键的、相对少量的数据。

系统中存在的问题是对于一般用户来说，向智能合约中存储数据的成本仍然相对较高。后期可能的改进的方法包括：1）改进数据加密方式，由此减少最终传递给智能合约的加密数据量。2）改进数据编码或传输方式，可以考虑使用更好的数据类型，或者在数据传输前对数据进行压缩。

**参考文献**

1. Reichl D. Keepass password safe[J]. Available in http://keepass. info/. Accessed May, 2018.
2. Google Contacts Support, <https://support.google.com/contacts/?hl=zh-Hans#topic=7578606/>，2018
3. Apple’s iCloud Keychain password management tool, URL: https://www.apple.com/support/icloud/keychain/, 2018.
4. Mingxiao D, Xiaofeng M, Zhe Z, et al. A review on consensus algorithm of blockchain[C]//Systems, Man, and Cybernetics (SMC), 2017 IEEE International Conference on. IEEE, 2017: 2567-2572.
5. Delmolino K, Arnett M, Kosba A, et al. Step by step towards creating a safe smart contract: Lessons and insights from a cryptocurrency lab[C]//International Conference on Financial Cryptography and Data Security. Springer, Berlin, Heidelberg, 2016: 79-94.
6. Huh S, Cho S, Kim S. Managing IoT devices using blockchain platform[C]//Advanced Communication Technology (ICACT), 2017 19th International Conference on. IEEE, 2017: 464-467.
7. Scott B. How can cryptocurrency and blockchain technology play a role in building social and solidarity finance?[R]. UNRISD Working Paper, 2016.
8. Irving G, Holden J. How blockchain-timestamped protocols could improve the trustworthiness of medical science[J]. F1000Research, 2016, 5.
9. Ethereum Homestead Document Latest, <http://www.ethdocs.org/en/latest/account-management.html>, 2018
10. Backbone.js Document, backbonejs.org, 2018

[11] Truffle-contract库使用，https:s://www.npmjs.com/package/truffle-contract, 2018