# 基于以太坊的个人信息管理系统--EthKee技术开发文档

1. **概述**

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。简单介绍Keeweb。

（需要扩充）

* 1. 系统需求

简单介绍Keeweb的应用背景、优点、缺点。由此引出将个人信息保存到以太坊中的优势。

Keeweb可以管理用户私密信息如生日、各类密码等，并可根据用户需要存储到云服务器上，由此可保证用户信息的安全性。

但是这种管理方式依赖于云服务器的安全性和可用性。

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。

（还需要扩充）

* 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[11]。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。



* 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. **开发环境**

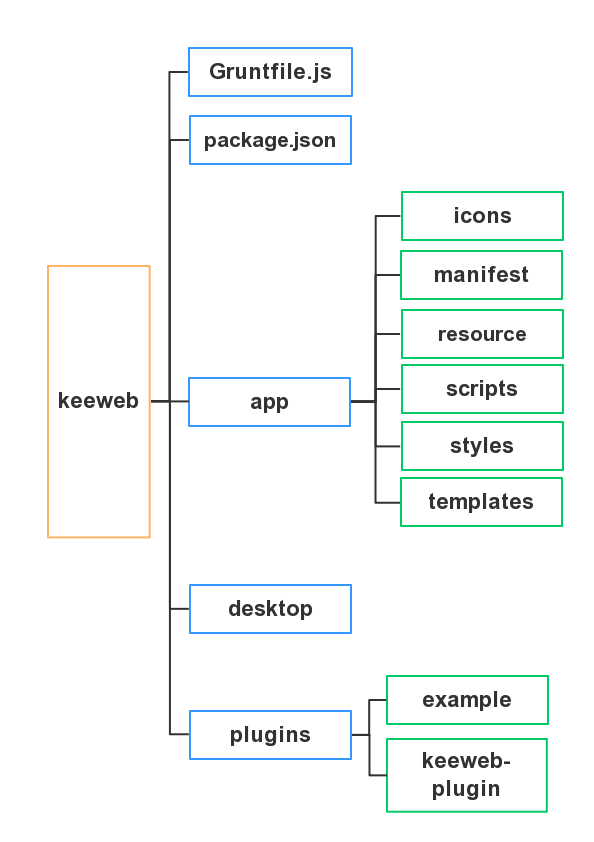
系统、服务器配置

1. **开发过程**
   1. 系统框架

前端框架采用Backbone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：

* 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；
* 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；
* 为交易提供了默认参数，例如from或gas；
* 为每个同步的交易返回logs、交易receipt和交易hash。
  1. 前端开发



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

文件：

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

* 1. 以太坊智能合约开发
     1. 以太坊集群搭建

简单介绍以太坊集群搭建的方法。

* + 1. 智能合约的开发

合约完成的功能总体描述。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”；