

Final Report

Lo Cheuk Yin 3035106243



Decentralized Data Marketplace

Topic: Blockchain Research and Implementation

Supervised by Dr S.M. Yiu

Table of Contents

| | |
|--|-----------|
| Project Background | 4 |
| Project Objective | 5 |
| About Decentralized Applications..... | 6 |
| Project Methodology..... | 7 |
| System Structure..... | 7 |
| Client Layer | 8 |
| Consensus Layer | 9 |
| Storage Layer | 9 |
| Technical Challenge..... | 10 |
| Transparency | 10 |
| Protecting buyer and seller | 11 |
| Solution Requirement | 11 |
| Flow of Transaction with Sampling Protocol..... | 12 |
| Highlights of the Solution | 14 |
| Dealing with transparency..... | 14 |
| Preventing Buyer to Obtain Whole Part of Data at Sample Requesting Stage | 14 |
| Prevents Frauds through Sampling..... | 14 |
| Prevents Frauds at Deal Settling..... | 14 |
| Drawbacks of the Solution | 15 |
| Depends on Probability | 15 |
| No True Random..... | 15 |
| System Design and Implementation | 16 |
| Program Class Design | 16 |

| | |
|---|-----------|
| Sequence Diagram of Transaction Flow | 17 |
| Implementation Walkthrough | 18 |
| Seller Uploading Data Asset | 18 |
| Buyer Request for Sample | 20 |
| Seller Approves/Rejects Data Sample Request | 21 |
| Buyer Downloads Sample | 22 |
| Buyer Request to Buy Data Asset | 24 |
| Buyer Downloads Data Asset..... | 26 |
| Analysis of the Implementation..... | 28 |
| Merits | 28 |
| Security | 28 |
| Hosting and Operating Cost is Low | 29 |
| Demerits | 29 |
| <i>Transaction Confirmations are Slow</i> | 29 |
| Expensive for Users | 29 |
| Transparency only in a sense to knowledgeable individuals | 30 |
| Data Uploaded Lives Forever..... | 30 |
| Recommendation and Future Work..... | 31 |
| Better random generation mechanism | 31 |
| Functionalities replaced by other technologies..... | 31 |
| Support more Data Type | 31 |
| Conclusion..... | 32 |

Project Background

Last year, I worked on an academic project that involves AI development. The client was trying to incorporate an AI into a stock price projection system. By the time when I was working on the project, I found the data gathering process very tedious and time consuming. There are insufficient data available, or data are scattered around which greatly hinder the development process. I realize the need for a global registry of datasets to facilitate development, and eventually gives me this idea to develop the Data Marketplace.

To generate good machine learning models, the quantity and quality of data are as important as the algorithm. Many machine learning planners find themselves insufficient of data, hindering the adoption of the technology in businesses. The lack-of-data obstacle is slowing down the research and development of machine learning, just like how the lack of computing power had for the past decades. In other words, today, one major key to build successful AI is the ability to gather data. And data are not only be highly demanded in AI development field, there are demands everywhere in other areas such as retail, banking and marketing etc. where companies want more data to facilitates companies decisions.

I believe the lack-of-data obstacle could be overcome through extensive cooperative research. However, traditional centralized server-client model is not suitable for data sharing and exchange because of the ambiguity in the ownership of data. Data sharers have to bear the middleman risks when using centralized solutions. There are concerns and news about problems regarding data integrity and privacy for companies and the general public to use platforms like Amazon and Google Cloud, which are middlemen, to store their data. Such platforms are blamed or suspected for manipulating user data under the table.

This motivates me to study the opportunity of blockchain technology in tackling the problem. Blockchain has the potential to eliminate the middleman risks and ensure complete data ownership by participants of the data exchange. It is believed that lowered risks and assured data ownership encourage data owners to share or sell their data. At the same time, Therefore, this blockchain project, having a vision of a global community of researchers, businesses and individuals cooperating in the R&D and adoption of machine learning, is launched.

Project Objective

The objective of this FYP is to build a fully decentralized data marketplace application powered with blockchain technology and to study the feasibility to bring the fully decentralized application into the market through analysing the pros and cons of such application.

Eventually, there will be a conclusion on whether such decentralized application is desirable and whether it is applicable in the market. The decentralized application will be built and the advantage and disadvantage of this decentralized application would be listed and recommendations will be given to such decentralized application.

About Decentralized Applications

Blockchain technology is the core technology adopted in this decentralized application as it plays an essential role in eliminating middleman risk, so as to mitigate the problems about data integrity and privacy raised by traditional centralized model.

Blockchain technology mitigates these middlemen risks through its special feature – smart contracts, which are used to handle system requests and settle transactions using predefined system flow and logics. Where blockchain also acts as a public database, which is a public ledger of records organized in blocks that are linked together by cryptographic validation. The block validation system results in new transactions being added irreversibly and old transactions preserved forever for everyone to see, therefore bring transparency and resilience of the system to general users.

Usual decentralized applications nowadays are usually built on well-established distributed public blockchain network like Ethereum, as the entry barrier to establish and maintain our own blockchain will be very high for usual developers, where ‘miners’ are required in the blockchain network to validate blocks in order to maintain the blockchain network.

Decentralized application developers publish, usually with a small amount of cost, their predefined blockchain logics, namely smart contracts, to the blockchain network serving as the backend of the system, and write their frontend interface to interact with the smart contracts.

In this project, the implementation of the data marketplace application will also follow this general approach as a decentralized application, details of the implementation will be included in later sections.

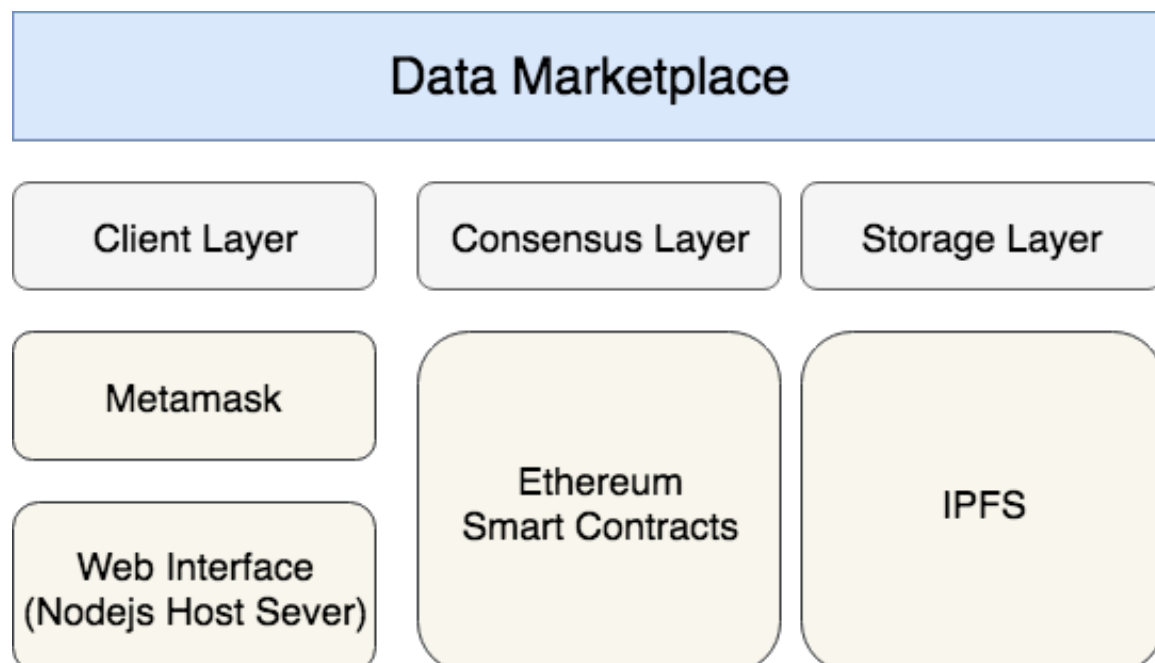
Project Methodology

A decentralized data marketplace application has been implemented in this project. It consists of the followings:

- Web interface for users to interact with the system
- Storage layer to store the data uploaded by the user to the platform
- Web hosting server with minimal functionalities to fulfil the goal of a decentralized application
- Smart contracts to handle the system logics of the transactions

System Structure

In this section, details of the system structure of the implementation will be explained. The general idea of the system structure of the implementation is illustrated as follows:



Client Layer

The major responsibility of this layer is to govern the interactions between users and the system, which mainly serves as a user interface. There will be a web interface for users to interact with the system.

Metamask

In this project, Metamask, a tool that enables users to make Ethereum transactions on a regular web browser, is adopted for users to interact directly with the Ethereum network.

Metamask serves as an Ethereum wallet that shows Ethereum balances, acts as user account and prompt user to sign Ethereum transactions.

In this project, implementation of user accounts is not required due to the adoption of Metamask as Metamask is actually the Ethereum user account that provides help for user identification.

Web interface/Node js hosting server

The data marketplace is built to be a web application. A hosting server is written in Node.js to handle routing of different webpages. There is almost no programming logics other than page routing in the backend script so as to match with the principle of being a decentralized application.

The programming logics are all implemented on frontend using web3.js. As the web application is integrated with Metamask, user identity (Ethereum address) could be passed to Ethereum network through web3.js.

Consensus Layer

Consensus Layer handles the system logics and settle transactions. In this project, the Ethereum network is adopted for building the decentralized data marketplace application. Therefore, this layer is implemented with Ethereum Smart Contracts.

Ethereum Smart Contracts

The Smart Contract is implemented and publish onto Ethereum Rinkeby Testnet. The Smart Contract serves the following major functions:

- Act as registry for Data Asset
- Act as registry for Transaction Requests
- Validates right for user requests that will add/edit/remove data on Smart Contract

Storage Layer

This layer is responsible for data storage, where data are transferred to this layer through consensus layer and the client layer.

InterPlanetary File System (IPFS)

IPFS is a protocol and network designed to create a content-addressable, peer-to-peer method of storing and sharing media in a distributed file system. It serves as a blob storage in the data marketplace application. A content-addressable link will be generated after uploading the media onto IPFS.

The reason to adopt IPFS into the data marketplace application is mainly because of the followings:

- Data stored through IPFS is unalterable

- Content-addressable links are permanent
- IPFS has no single point of failure
- Decentralized

The above three points made IPFS to be considered as a perfect complement with blockchain technology especially it shares the same properties of not having a single point of failure and being decentralized.

Content-addressable link instead of the data itself can be stored directly onto the Smart Contract. Since data uploaded via IPFS is unalterable and having the content-addressable links being permanent on IPFS reduces the risk of having error on blockchain for referring not addressable link.

Technical Challenge

Transparency

As a public ledger, Blockchain gives total transparency of the values on the network, that means everything on the blockchain is open to everyone. However, in the data marketplace application, data asset is for sale. It implies that one should pay for the right to use the data. In traditional centralized model, there exists a middleman to temporarily hold the data asset uploaded, and hide it for you until there is somebody else trying to buy it. But with total transparency given by Blockchain technology, the data assets uploaded (or the content-addressable links) will be accessible by everybody. Given this accessibility by everyone, no one would pay for the data asset uploaded as they have already been available to use without any cost. It conflicts with the core principle in commercializing data assets in the data marketplace application.

To solve this problem, users have to encrypt their data asset before uploading. Due to the long processing time to encrypt and decrypt the whole set of data asset, users will be uploading their encrypted encryption keys of their encrypted data assets to the blockchain, and upload the encrypted data to the storage layer. Again, since there is no middleman to handle transaction, there exists another challenge - to protect both buyer and seller.

Protecting buyer and seller

By saying protecting buyer, it means there should be mechanism to ensure buyer can trust the data asset is the thing that they want to buy, since the data assets and the encryption keys of the data assets are encrypted that buyers cannot check the data before they send a purchase request to the seller. Frauds might arise where seller could upload some data asset that has no use and pretend to be some valuable data. There is clearly a need for buyer to get samples of the data asset before they decide whether to buy the data asset or not.

As for seller, as mentioned above that buyer should be able to get sample before they decide whether to buy the data asset beforehand. There should also be mechanism to ensure seller will not give out every part of the data asset through sampling.

Solution Requirement

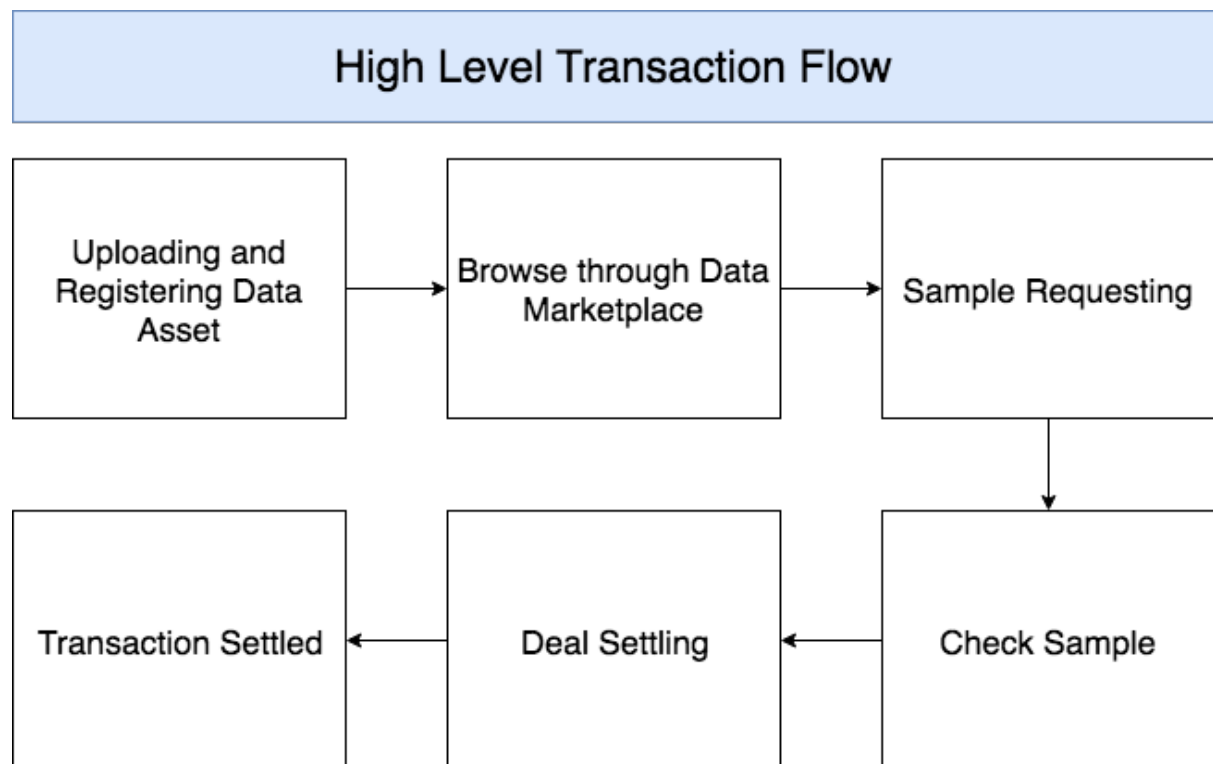
To deal with the two challenges raised above, a sampling protocol satisfying the following conditions is needed.

The sampling protocol should:

- Choose a random portion of the data asset as sample to buyer
- Not reveal whole set of data asset to a single buyer
- Be able to choose any part of the data asset
- Prevent buyer and seller to choose a particular part of the data asset intentionally

Flow of Transaction with Sampling Protocol

The high level idea of the flow of transaction in the data marketplace application is illustrated in the following graph.



The sampling protocol enables buyer and seller to carry out transaction with one another with the existence of the Ethereum Smart Contract. The general flow of the transaction is described as follows:

1. Seller divides his or her data asset into different segments and then encrypts the data segments with different encryption keys.
2. Seller uploads the encrypted data asset through IPFS and registers the data assets onto the blockchain network with the returned content-addressable link to access the data asset.
3. Buyer browser through the data marketplace application to look for data assets.
4. Buyer requests for sample of the data assets providing his or her public key.

5. Seller receives the sample request and encrypts all encryption keys with Buyer's public key and provide them to the Smart Contract.
6. Smart Contracts stores the hashes of the encrypted encryption keys and draws one of the encryption keys, which correspond to different data segments divided previously, randomly and provide that to Buyer.
7. Buyer receives one of the encryption keys encrypted with his or her public key
8. Buyer decrypts the encrypted encryption key with his or her private key and open the selected data segment with the encryption key.
9. If Buyer wants to buy the data asset after viewing the sample data, Buyer launches a purchase request sends the required amount to Smart Contract.
10. Upon receiving the purchase request, Seller provides again all encryption keys encrypted with the buyer's public key to the smart contract.
11. Smart Contract checks if the uploaded encrypted encryption keys' hashes at the buy request stage match with the previously uploaded encrypted encryption keys at the sample request stage.
12. If the hashes match, Smart Contract confirms the transaction and send the required amount of the data asset to Seller and stores the encrypted encryption keys. The deal is then settled.
13. Buyer receives the encrypted encryption keys, and open the data assets with his or her private key.
14. The whole transaction is now settled.

Highlights of the Solution

Dealing with transparency

In each transaction, Seller will have to encrypt the encryption keys with the Buyer's public key which enables only the Buyer could retrieve the encryption key of the data asset. This solves the problem of transparency of Blockchain technology as the sensitive data is encrypted for the use of particular users.

Preventing Buyer to Obtain Whole Part of Data at Sample Requesting Stage

At the time Seller provides the encrypted encryption keys to Smart Contract for the Buyer at the sample request, Smart Contract stores all the hashes of the encrypted encryption keys and gives only one of the encrypted encryption keys. This prevents Buyer to get all the encryption keys at the sample requesting stage.

Prevents Frauds through Sampling

Smart Contract will draw one of the uploaded encryption keys randomly at the sample requesting stage. This prevents Seller to cheat Buyer as sample of the data asset will be drawn, and if the Seller wants to cheat only part of the data asset, for example actually half of the data asset is useless, there might be possibility where Smart Contract will draw the cheating part of the data asset and provide that to the Buyer. Where Buyer will decide not to buy the data asset upon seeing the useless data from the sample.

Prevents Frauds at Deal Settling

Seller may decide to cheat the Buyer by not providing the true encryption keys at the deal settling stage they pass the sample requesting stage and Buyer decides to buy the data asset. However, as the hashes of the encrypted encryption keys are stored on the Smart Contract, and the keys uploaded at the deal settling stage are checked with the hashes. This prevents

Seller to lie as Seller cannot get the money if the uploaded encrypted encryption keys do not match with the hashes stored in the Smart Contract.

Drawbacks of the Solution

Depends on Probability

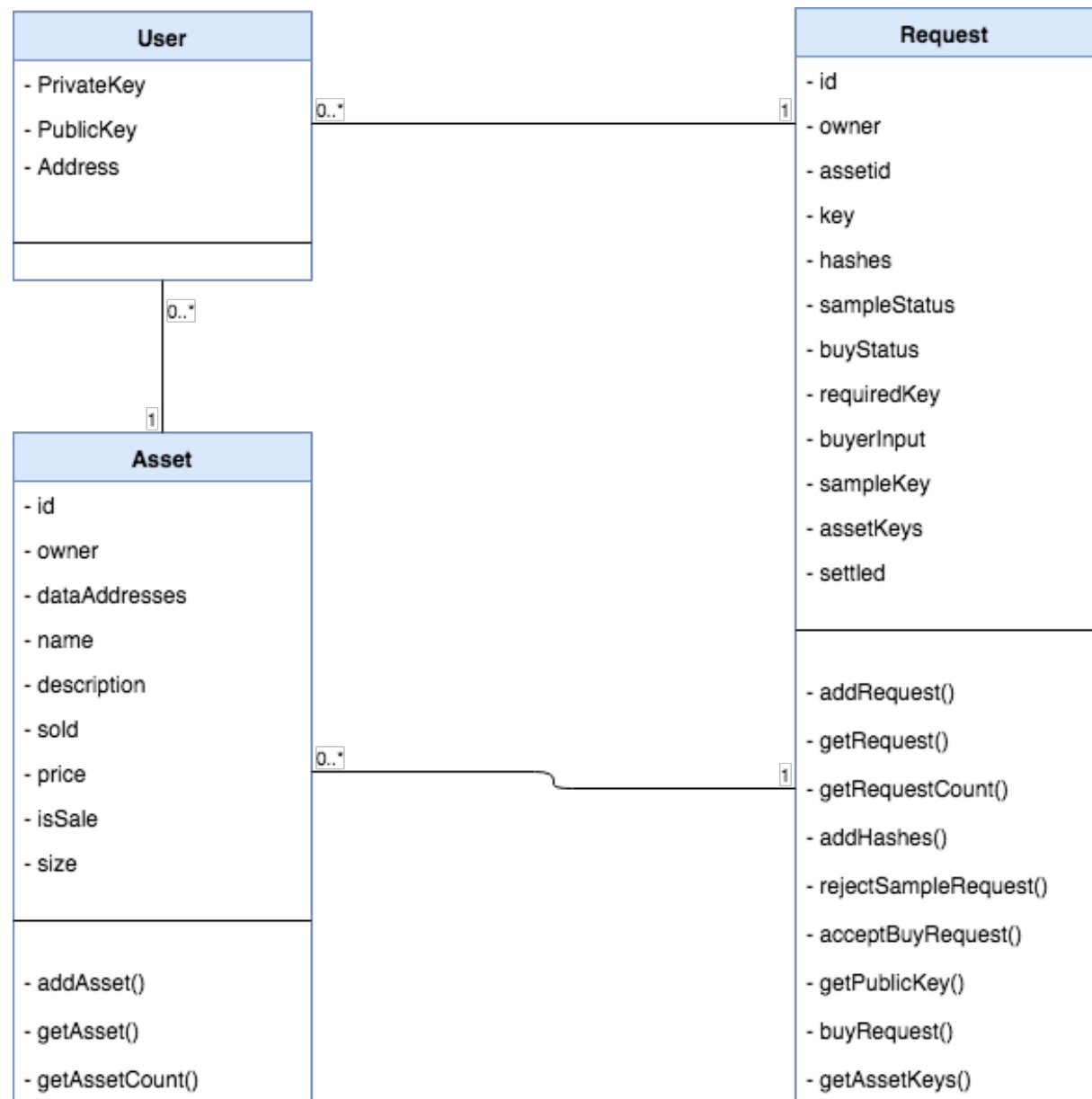
Since the sampling protocol draws only one of the encryption keys, it depends on probability whether it can truly detect if there are data segments that contain useless data to cheat buyers. Therefore, it cannot fully eliminate the possibility for fraud to happen at this stage, but only to improve the situation and increase the cost to cheat on this platform.

No True Random

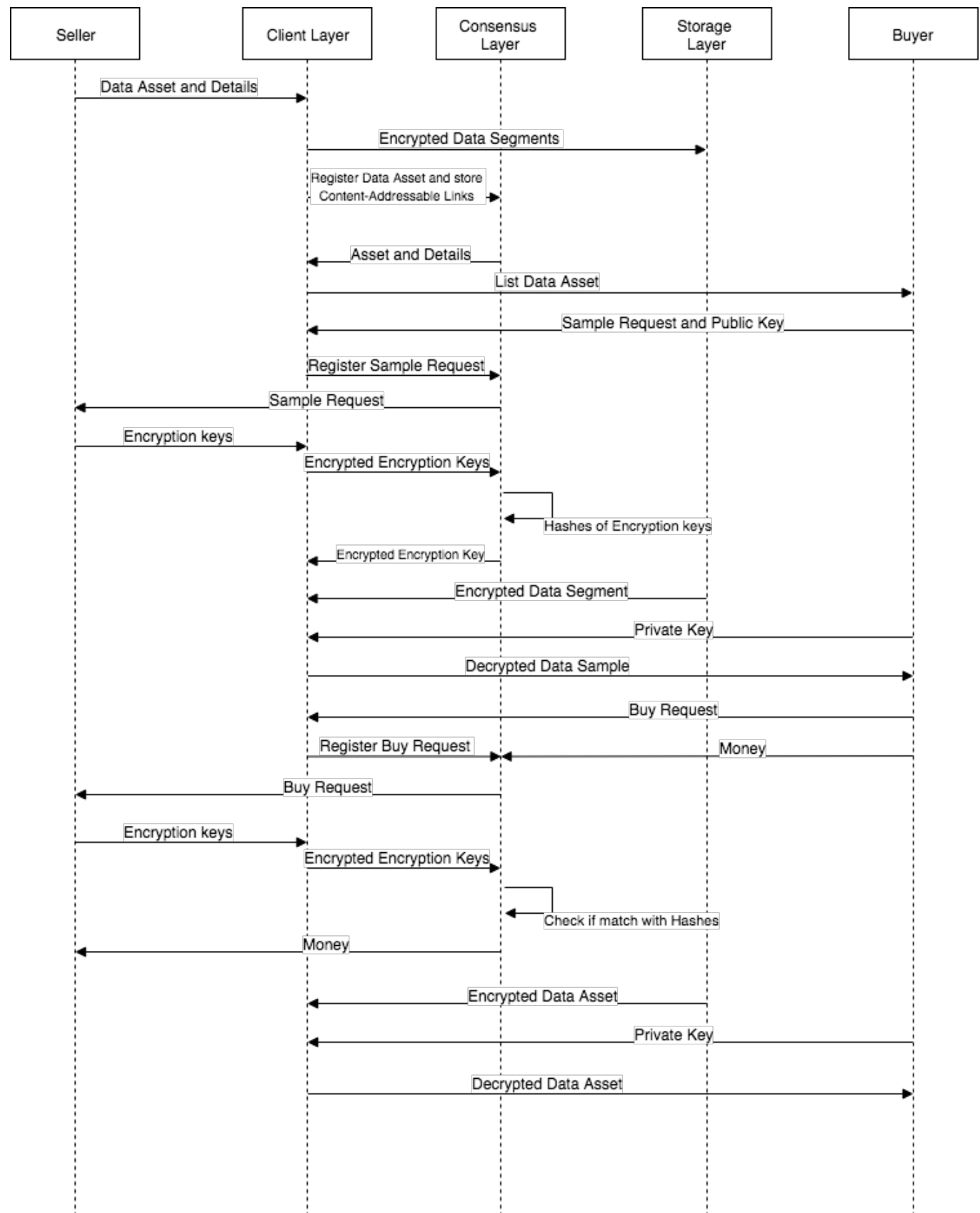
Given the deterministic characteristics of blockchain technology, in theory everything running on blockchain is predictable, there is no official implementation of randomization. Yet, it does not mean the sampling protocol cannot come into solving the problem. There are randomization generators on blockchain provided by third parties like Oraclize, a blockchain oracle service which can provide a random number. Although it means that we have to trust Oracle that they provide a real random number, it leaves possibilities for this sampling protocol to work. In the implementation of the data marketplace application, Oraclize is not adopted as there is no provision in Testnet environment but only in the market, a self-written semi-randomization function that uses blockhash and involving buyer and seller input was written to replace Oraclize.

System Design and Implementation

Program Class Design



Sequence Diagram of Transaction Flow



Implementation Walkthrough

The implementation walkthrough will go through the flow of a normal transaction from Seller uploading a data asset to Buyer receiving the data asset.

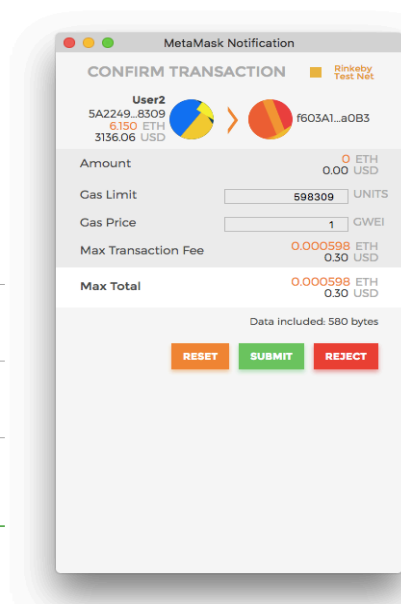
Seller Uploading Data Asset

1. Seller will first get into the page for selling asset.

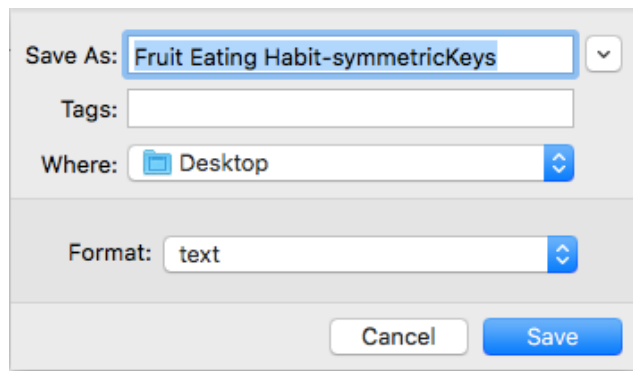
The screenshot shows the 'Data Marketplace' header with navigation links: 'My Asset', 'Seller', 'Buyer', and 'Sell Asset'. The main form includes fields for 'Asset Name', 'Description', 'Price' (with a dropdown for 'Ether'), and 'Number of Segments of the Asset'. At the bottom, there is a 'FILE' button and a text input field with the placeholder 'Upload your file here'.

2. Seller chooses the number of segments for dividing the data asset, symmetric keys will be generated at client-side. Metamask will prompt for transaction confirmation after Seller has inputted all required input fields and chose to upload the data asset.

This screenshot shows the 'Price' field set to '0.1' and 'Ether'. The 'Number of Segments of the Asset' is set to '3'. A 'REGENERATE KEYS' button is visible. Below, three symmetric keys are displayed: 'Key 1' (f5ls/8XgwxP?6l0dcI6JLlU=+faP0RVW), 'Key 2' (oqaRnWxgvM1%bKuoC9r98h1*gR%2BJtG), and 'Key 3' (8NBDD*%Klf2Sa44xNd9J2lli=wq-X!*). At the bottom, a 'FILE' button is next to the filename 'fruit.csv', and an 'UPLOAD' button is at the bottom right.



3. Symmetric keys encrypting the data segments will be downloaded.



Save As: Fruit Eating Habit-symmetricKeys

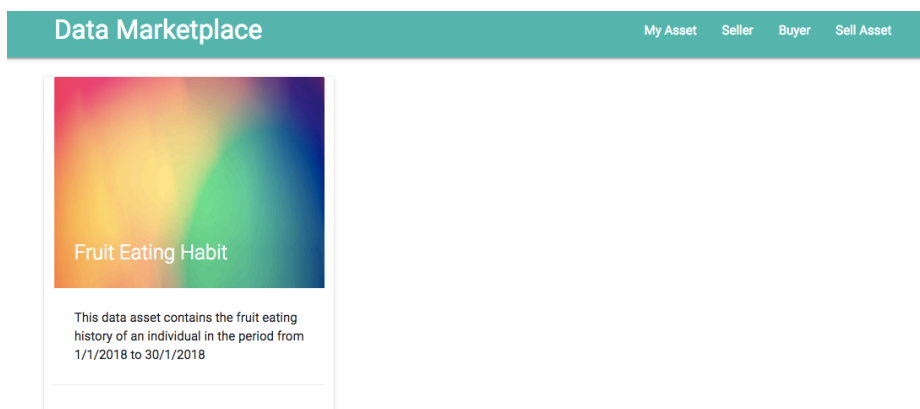
Tags:

Where: Desktop

Format: text

Cancel Save

4. The data asset will now be available at the data marketplace.



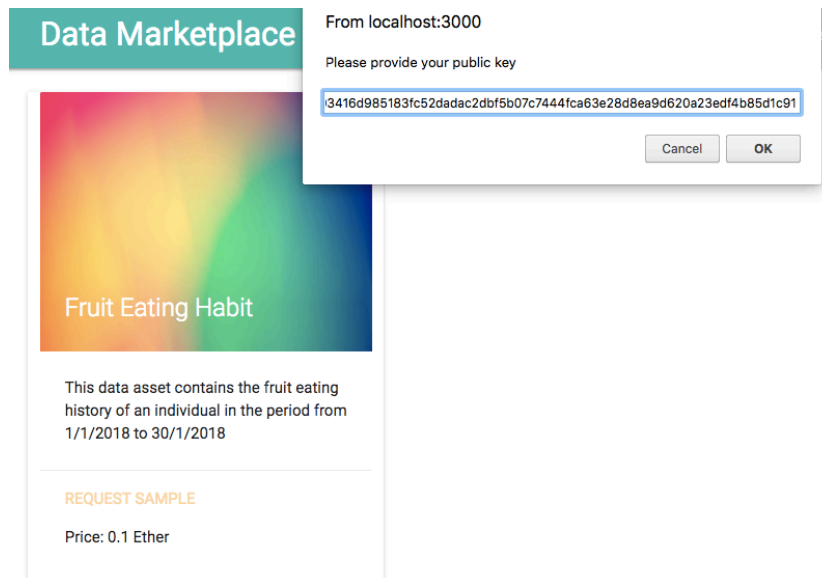
Behind the scene:

1. Seller filled in all required input fields and selected the file to upload and clicks the upload button.
2. The file will be read at the browser and be divided into the number of segments users chose to divide and they are encrypted with the generated symmetric keys.
3. The data segments will be uploaded through IPFS and content-addressable links will be obtained.
4. The data asset will be registered onto Smart Contract with the content-addressable links.

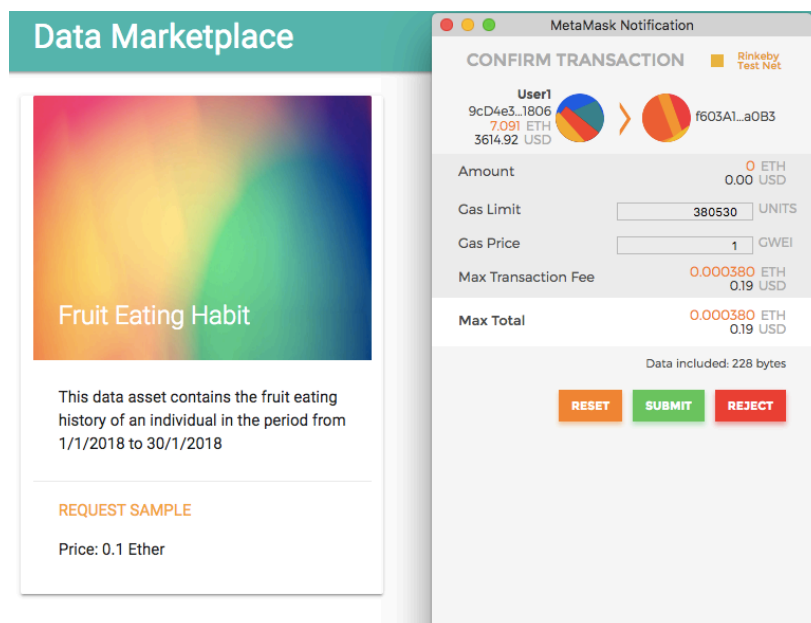
5. The generated symmetric keys will remain at the client end and prompt user to download without going to Smart Contract or the backend.

Buyer Request for Sample

1. Buyer browses for data assets and clicks the “Request Sample” button and provides the public key.



2. Metamask will prompt for transaction confirmation.



Behind the scene:

- Buyer uploads his or her public key to Smart Contract.

Seller Approves/Rejects Data Sample Request

1. Seller checks for any requests received at the seller page

| Data Marketplace | | | |
|--|--------------------|---------------|---------------|
| | | | |
| Requests | | | |
| Requester | Asset | Sample status | Buyer Request |
| 0x9cd4e38bf074e60f4e7f1435895310ebf37f1806 | Fruit Eating Habit | Pending ✓ ✗ | - |

2. Seller provides the symmetric keys previously downloaded from the application when uploading the data asset to approve the sample request.

Seller simply clicks “X” to reject sample and does not need to provide anything.

Data Marketplace

Requests

Requester

0x9cd4e38bf074e60f4e7f1435895310ebf37f1806

From localhost:3000

Please provide the symmetric encryption keys for this asset

/,oqaRnWxgvM1%bKuoC9r98h1*gR%2BJtG,8NBDD*%Klf2Sa44xNd9J2ili=wq-X!*i

Cancel OK

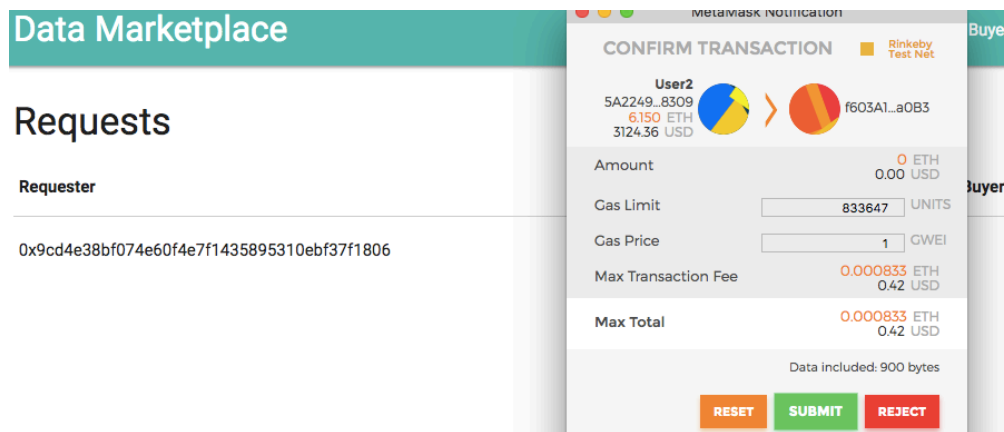
Fruit Eating Habit

Pending ✓ ✗

Fruit Eating Habit-symmetricKeys.txt

f51s/8X9wxP7610dcI6JL!U=+faP0RVW,oqaRnWxgvM1%bKuoC9r98h1*gR%2BJtG,8NBDD*%Klf2Sa44xNd9J2ili=wq-X!*i

3. Metamask prompts for transaction confirmation.



Behind the scene:

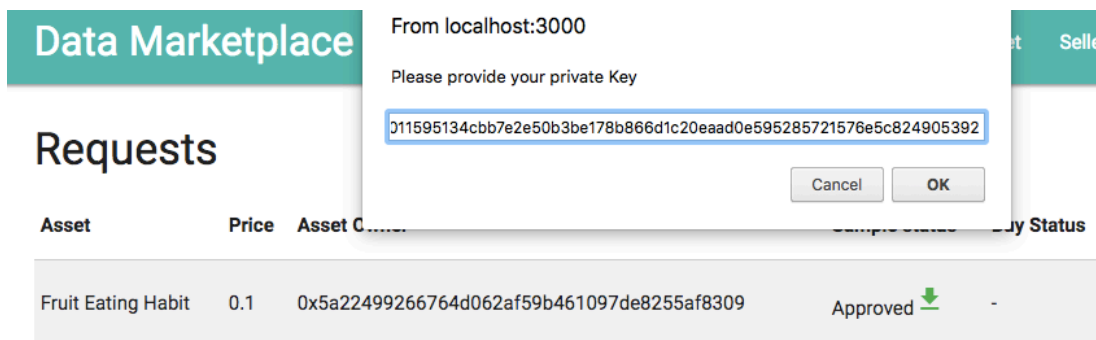
1. After providing symmetric keys, Smart Contract will obtain the public key of the Buyer.
2. Browser will encrypt the symmetric keys with buyer's public key
3. Encrypted encryption keys will be passed to Smart Contract.
4. Smart Contract stores the hashes of the encrypted encryption keys
5. Smart Contract randomly draws one key to be the sample key and store the sample key in Smart Contract.

Buyer Downloads Sample

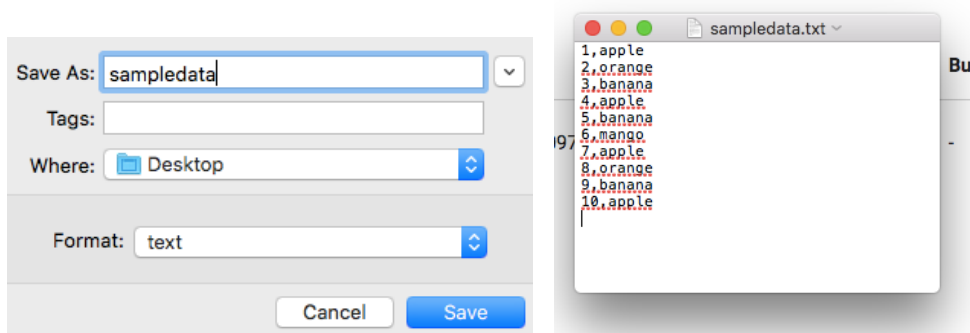
1. Buyer checks for request status at the Buyer page.

| Data Marketplace | | | | | My Asset | Seller | Buyer | Sell Asset |
|--------------------|-------|--|--|------------|----------------------------|--------|-------|------------|
| Requests | | | | | | | | |
| Asset | Price | Asset Owner | Sample status | Buy Status | | | | |
| Fruit Eating Habit | 0.1 | 0x5a22499266764d062af59b461097de8255af8309 | Approved  | - | <button>BUY ASSET</button> | | | |

2. Buyer provides his or her private key to decrypt the sample file.



3. Metamask prompts for transaction confirmation.
4. Buyer successfully downloaded the sample file.




Behind the scene:

1. Buyer obtains the encrypted encryption key and content-addressable link from Smart Contract.
2. Buyer obtains the encrypted data segment through IPFS at the client end.
3. Browser decrypts the encrypted encryption key with buyer's private key and decrypts the obtains data segment with the decrypted symmetric key.

Buyer Request to Buy Data Asset


1. Buyer clicks “buy asset” button

| Data Marketplace | | | | | My Asset | Seller | Buyer | Sell Asset |
|--------------------|-------|--|--|------------|----------------------------|--------|-------|------------|
| Requests | | | | | | | | |
| Asset | Price | Asset Owner | Sample status | Buy Status | | | | |
| Fruit Eating Habit | 0.1 | 0x5a22499266764d062af59b461097de8255af8309 | Approved  | - | <button>BUY ASSET</button> | | | |

2. Metamask prompts for transaction confirmation. The transaction value, which is the price of the data asset, is illustrated. Buyer will send the money (Ether) at the transaction.

MetaMask Notification

CONFIRM TRANSACTION

 Rinkeby Test Net


User1

9cD4e3...1806

7.091 ETH

3618.98 USD

>



f603A1...a0B3

Amount

0.100000 ETH

51.04 USD

Gas Limit

UNITS

Gas Price

GWEI

Max Transaction Fee

0.000042 ETH

0.02 USD

Max Total

0.100042 ETH

51.06 USD

Data included: 36 bytes

RESET

SUBMIT

REJECT

3. Seller checks buy request at the “seller” page.

| Data Marketplace | | | |
|---|--------------------|---------------|---------------|
| <div>My Asset Seller Buyer Sell Asset</div> | | | |
| Requests | | | |
| Requester | Asset | Sample status | Buyer Request |
| 0x9cd4e38bf074e60f4e7f1435895310ebf37f1806 | Fruit Eating Habit | Approved | Pending ✓ |

4. Seller provides symmetric keys to approve the buyer request.

| Data Marketplace | | | |
|--|--------------------|---------------|---------------|
| <div>From localhost:3000</div> <div>Please provide the symmetric encryption keys for this asset</div> <div><input type="text" value="/,oqaRnWxgvM1%bKuoC9r98h1*gR%2BJtG,8NBDD*%KIf2Sa44xNd9J2Ii=wq-X!* "/></div> <div><div>Cancel</div><div>OK</div></div> | | | |
| Requests | | | |
| Requester | Asset | Sample status | Buyer Request |
| 0x9cd4e38bf074e60f4e7f1435895310ebf37f1806 | Fruit Eating Habit | Approved | Pending ✓ |

5. Seller receives the payment upon confirmation of the transaction.

Behind the scene:



1. Buyer register buy request and send the required amount of money (Ether) to Smart Contract.
2. Upon receiving the buy request, Seller uploads the symmetric keys to the browser.
3. Buyer’s public key is obtained from Smart Contract.
4. Browser encrypts the uploaded symmetric keys with buyer’s public key and pass them to Smart Contract.
5. Smart Contract checks whether the hashes of the uploaded encrypted symmetric keys match with the hashes stored previously to ensure they are the same encrypted symmetric keys.
6. Seller gets the Ether if all hashes matches and the encrypted symmetric keys will be stored in the Smart Contract.

Buyer Downloads Data Asset

1. Buyer checks for request status at the “Buyer” page.

| Data Marketplace | | | | |
|--------------------|-------|--|--|--|
| | | | | |
| Asset | Price | Asset Owner | Sample status | Buy Status |
| Fruit Eating Habit | 0.1 | 0x5a22499266764d062af59b461097de8255af8309 | Approved  | Approved  |

2. When the request is approved, Buyer clicks on the download icon to download the data asset providing his or her private key.

| Data Marketplace | | | | |
|--------------------|-------|--|--|--|
| | | | | |
| Asset | Price | Asset Owner | Sample status | Buy Status |
| Fruit Eating Habit | 0.1 | 0x5a22499266764d062af59b461097de8255af8309 | Approved  | Approved  |

From localhost:3000

Please provide your private Key

Cancel OK

3. Buyer gets the whole set of data asset.

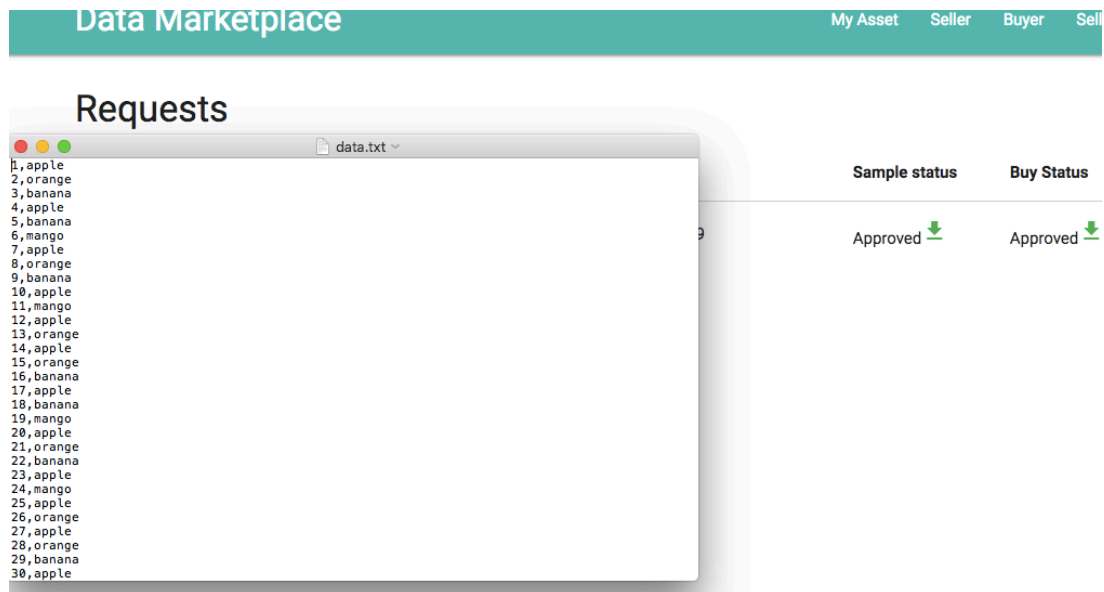
Save As:

Tags:

Where:

Format:

Cancel Save



Behind the scene:

1. Encrypted symmetric keys and content-addressable links are obtained from the Smart Contract.
2. Browser gets all encrypted data segment through IPFS with the content-addressable links and decrypts the encrypted symmetric keys with buyer's private key. Then browser decrypts the encrypted data segments with the decrypted symmetric keys.
3. Browser combines the data segment into one and prompts buyer to download the combined data asset.

Analysis of the Implementation

This section analyzes the merits and demerits of the data marketplace application which is implemented in a totally decentralized manner.

Merits

Security

Data will never be gone on Blockchain, and integrated with IPFS, which is a decentralized permanent storage protocol, there are no concerns of losing the uploaded data in the data marketplace. Besides, since all payment involved in the application uses Ether (cryptocurrency on Ethereum Network), there is completely no need to use third party payment processing in the operations of data marketplace application. This eliminates middleman risk from transaction between the application and payment processing service provider and enhances security of the application. Apart from this, since smart contract could identify user identity through user address, there is no need to implement platform user credentials, which could further reduce risk in user account hacking.

Trust

In this project, there is no middleman in handling transactions between users. Seller and Buyer directly communicate with the Smart Contract through the client layer. There is no trust issue, as in theory user has the ability to read through everything to be executed, user can see how the code runs when uploading their data asset and how they are managed through client-side code and code in smart contract. This is totally different from traditional centralized model where all operations are hidden from users. Users could also check transaction history as all transactions are logged on the blockchain.

Hosting and Operating Cost is Low

As a fully decentralized application, most if not all of the computing and transaction handling falls onto the client side. In this project, the function of the hosting server is only to handle page routing. All other functions such as encryption, file handling all lies on the client side. Especially when all transactions communicating with Smart Contracts are done at the client end, all cost are bore by users. There is limited cost service provider of a fully decentralized application has to bear.

Demerits

Transaction Confirmations are Slow

Since users interact directly with Smart Contract, users have to confirm transactions every time they add or modify state in the Smart Contract. Transaction confirmation could take up a significant amount of time. In the data marketplace application, it on average takes one minute for a transaction to be confirm. For minor state modification like closing data asset from sale from public. Such wait could be acceptable for large operation such as to upload a data asset, but would serious affect the usability of the application if this applies to every minor operation.

Expensive for Users

Transaction cost is found to be high when the operation is complicated on Smart Contract. Transaction cost in general is acceptable for data marketplace's platform nature. However, when transactions are carried out in a smaller scale and requires large number of transactions, the total cost could grow significantly.

Transparency only in a sense to knowledgeable individuals

Although the data marketplace application provides transparency to users where user could see the code executed on the client end and on Smart Contract, it does not mean every user has the ability and time to understand what is happening behind the scene of the flow. Like general user uses common platforms like Google or Baidu, they might not take time to understand what is happening behind the scene or lack the knowledge to understand how do the services work even there are part of them being explained officially.

Data Uploaded Lives Forever

Take handling sample request as an example, to ensure Seller will not upload false symmetric keys to trick Buyers after Seller passed the sample requesting stage, Seller can only upload their encrypted symmetric keys and store the hashes of the keys once. In this scenario, if Seller uploads wrong symmetric keys at the first time, Seller cannot re-upload the key again because of the safety mechanism to protect Buyer and Seller. As a result, wrong data will live forever in the Smart Contract and remains as garbage data and takes up resources.

Recommendation and Future Work

Better random generation mechanism

Due to the limitation of the deterministic feature of Blockchain technology and not having third party random number generator provider on Rinkeby Testnet, only a semi-random number generator function is written on Smart Contract using blockhash and user inputs to increase the effort required to manipulate the number generation function. It is recommended to implement a true random function using different random number generation provider.

Functionalities replaced by other technologies

After analyzing the decentralized data marketplace application, it is found that transaction cost could be huge and time to wait for transaction confirmation is too long for some operations. It is recommended to replace some minor operations like closing data asset from sale by other technologies. For example to implement a database just to keep states that require immediate confirmation and with minor influence to the whole transaction also to save cost.

Support more Data Type

In this application, only data asset in the form of csv file can be successfully transacted between different users. It is hoped that there could be more data types such as images, audio and video being able to be transacted in the data marketplace application.

Conclusion

The fully decentralized data marketplace has the advantage of having better security and transparency compared to traditional centralized model. The mechanism eliminates the need of a middleman and reduced middleman risks like data leakage and under the table manipulations. At the same time, the hosting and operating cost for providing service with such decentralized application is low, which enables a better comparative advantage to traditional service provider.

However, drawbacks of the application include long transaction confirmation time and significant transaction cost could be induced to users. Apart from that, transparency of the data marketplace application only has value and applicable for a specific group people who is knowledgeable to understand the mechanism.

To bring this application into the market, the above mentioned downside of the data marketplace application should be mitigated. Besides, there are still concerns not only from this particular application but the environment.

First, cryptocurrency is still not common for the general public. As cryptocurrency is the only mean to pay and receive in this data marketplace application and other common decentralized applications, it would require high learning curve for users to understand how cryptocurrency works such that they could directly use the decentralized application by themselves. As decentralization means there will not be middleman to handle, it all depends on the user.

Second, let alone cryptocurrency not being common for general public, as cryptocurrency is still a kind of assets with (drastic) fluctuating value, this might bring huge fluctuations onto

the pricing of data assets on the data marketplace application, which could hardly be countered at this stage.

In short, there are merits for launching the decentralized data marketplace application but with some shortcomings. Such shortcomings could be improved with recommendations in the later section. Given the hosting and operation cost is low, it is still recommended to launch the data marketplace to the market, and modify the application to fit user need better in the future.