DataMatch

Zixin Wei z5wei@ucsd.edu
Maxwell Fang m3fang@ucsd.edu
Yanhao Guo yag007@ucsd.edu
Aditya Melkote avmelkote@ucsd.edu

Nolan Sheffield Sheffield.Nolan@franklintempleton.com

Abstract

Data scientists require vast amounts of data to run complicated models. Centralized marketplaces exist to provide data, but their methods contain privacy risks. Therefore, we propose a decentralized, Ethereum-based data marketplace that allows anyone to buy or sell encrypted data securely without intermediaries. Our platform leverages smart contracts to automate transactions— including licensing and payment conditions— and incorporates data anonymization techniques like differential privacy and homomorphic encryption, allowing training without exposing raw sensitive data. We plan to implement a token economy to incentivize the provision of high-quality data by rewarding honest stakers and punishing dishonest ones. Off-chain storage solutions like IPFS are used to store large datasets under blockchain-based mechanisms. This approach resolves storage issues by enhancing data security, reducing costs, and maintaining transparency among all participants.

Code: `https://github.com/Fangtastic7/DSC-180A-B16---Group1`

1 Introduction

In the digital age of information, data is everywhere, and it is one of the most important assets that organizations can purchase or sell. Companies produce large amounts of data each day to help provide insights and discoveries for their customers, industries, and ultimately the world. Currently, there are existing data cloud marketplaces that provide data scientists, analysts, and other professionals with the convenience of accessing data from a wide variety of sources. This can be viewed as beneficial, where it helps to reduce time and analytics costs; nevertheless, there are many risks associated with having a centralized storage. Data leakage, hardware failure, and large-scale data transmissions can result in low efficiency, high costs, and insufficient use of data value. However, a decentralized, Ethereum based data platform would help to implement privacy-preserving techniques like Li et al. (2020)secure-multiparty-computation or homomorphic encryption, allowing analysis without exposing sensitive data. One could argue that cloud marketplaces can ensure the trust factor of users through verification, but this requires information to be shared by the user. With blockchain technology and cryptography techniques, many issues, regarding user privacy and trustworthy transactions, can be resolved in a peer-to-peer network.

Due to Ethereum having a proof-of-stake sybil resistance mechanism, the Ethereum blockchain and by extension any smart contract hosted on the blockchain by result is a secure yet transparent method of executing transactions. By employing Ethereum smart contracts, users can license, ensure proper usage rights, and define payment conditions without help from an authority or firm. To prevent attacks on the network, Ethereum utilizes a proof-of-stake based consensus mechanism to prevent single-point failures, where all nodes must agree on the state of the blockchain. If an attacker were to attempt to take control of the chain, they must destroy a massive amount of ETH.

Essentially, the system incentivizes individual stakers to behave honestly and penalties disincentivize stakers from acting maliciously. Blockchains are robust and reliable, but there is one challenge that it faces: scalability. Considering there are vast amounts of data, blockchains are not feasible in da Silva Vanin et al. (2022)“terms of computational resources and cost.” Many studies consider only storing transaction history information in blockchains while the real data is da Silva Vanin et al. (2022)“stored in off-chain infrastructures, such as cloud service providers or health institution premises.” The issue with cloud service providers or health institution premises is that data can be easily leaked or compromised. In early 2024, UnitedHealthcare was hacked by a Russian ransomware group, where 100+ million personal health records were compromised. The Russian group ended up securing 22$milliondollarsandcausing$3 billion dollars of damage. There are many more cyberattacks happening today and the number will continue to rise. One solution to ensuring safer data storage is the use of the InterPlanetary File System (IPFS), which decentralizes large amounts of data in a peer-to-peer network. In an IPFS network, users can private their networks, encrypt their data, and control gateway utilization to further secure their information. Ultimately, we propose that smart contracts provide a better option than third party marketplaces and off-chain decentralized storage systems maintain greater data privacy than cloud service providers.

2 Overview

Our decentralized data marketplace project aims to create a secure platform for buying and selling datasets while preserving user privacy. Over the quarter, we successfully built a system integrating Ethereum-based smart contracts, IPFS for decentralized storage, and encryption techniques. The marketplace facilitates trustless data transactions, allowing sellers to list encrypted datasets and buyers to securely purchase them. The platform's foundation is a smart contract deployed on the Amoy Test Network, managing licensing, payment terms, and access control without intermediaries. This ensures data authenticity, transparency, and privacy while enabling computations on encrypted data.

To achieve this, we utilized several tools. Remix IDE was employed to develop and deploy the smart contract on the test network, ensuring the logic for listing, buying, and delisting datasets is automated and secure. Pinata was integrated to interface with IPFS, enabling decentralized file storage and generating unique CIDs for datasets, which are linked to the blockchain for secure retrieval. Next.js served as both the frontend and backend framework, connecting users with the blockchain, handling file uploads, and managing interactions with Pinata and the smart contract. Additionally, MetaMask was used to enable users to connect wallets and sign transactions seamlessly. AES encryption was implemented to secure datasets, with ongoing exploration of homomorphic encryption to allow computations on encrypted data without compromising privacy.

By combining blockchain technology, decentralized storage, and encryption, we have built a scalable, secure, and privacy-preserving data marketplace. This quarter focused on establishing the core infrastructure, integrating the tools, and successfully demonstrating critical functionalities like file uploads, secure storage, and trustless transactions. The project lays the foundation for further enhancements, including public deployment, recommendation algorithms, and a token economy to incentivize high-quality data contributions.

3 Literature Review

In the paper Sober et al. (2022)"A Blockchain-Based IoT Data Marketplace," the authors discuss the decentralized platform leveraging blockchain technology and smart contracts to enable secure and trustless data trading among the Internet of Things (IoT) devices. Traditional decentralized data marketplaces often face challenges of high infrastructure costs and trust issues. With blockchain technology, the platform can address concerns through transparency, immutability, and decentralization. The immutability feature of blockchain is featured in the security of transactions, where everything is permanent and traceable. Smart contracts, on the other hand, can automate and enforce the rules of data exchange, facilitating seamless interactions between buyers and sellers of data. Through optimizing smart contracts, the costs of transactions can be lowered. By incorporating components, such as proxies and brokers, the platform can manage data trading efficiently. By addressing the practical considerations of cost, scalability, and security, the study presents a robust framework, demonstrating a blockchain-based marketplace model that is both feasible and adaptable across various IoT-driven industries.

One effort to reduce some of the shortcomings of previous iterations of decentralized-based marketplace projects such as the Origami Network is to remove a centralized point of storage using a Swarm decentralized database to reduce single points of failure. Fonseca, Dahal and Kim (2020)In addition to employing the ETH blockchain and smart contracts for transparent and shared listings of transactions as well as automating the escrow process, Jorge Cacho et al. addressed the problem of high gas cost fees as well as slow execution speed for querying and iterating through the database to search for products by storing the data in Swarm and simply performing a match with the data on the blockchain. As each one of these operations when called by a transaction consumes cost when solely stored on the blockchain, the Swarm approach greatly minimizes the net gas cost per user transaction as well as speeds up the transaction speed to be able to realistically compete with traditional transaction services.

4 Infrastructure

4.1 High-Level Infrastructure

- *Remix-Ethereum:* Remix-Ethereum is the IDE where we write our smart contract code. In addition, it allows us to deploy our contract. Any updates or changes must be done here and redeployed.

- *MetaMask Wallet:* To make a transaction (buying, listing, and delisting), the buyer/seller must connect to a crypto currency wallet. In this case, we utilized Metamask, where the currency is either ETH, MATIC, or POL, depending on the testnet. Each account has a public address that is used to send and receive ETH as well as for verifying funds.

- *Amoy Test Network:* This is a test network for our contract to operate on until our contract is ready for production. It is essentially a simulated Ethereum Virtual Network (EVM) that uses POL as its currency. The currency is considered fake crypto currency, so that we can run as many tests as we want without incurring any costs.

- *OKlink.com:* OKlink is where we can interact with our smart contract. We run our smart contract, verify it (ensure the bytecode for contract execution is the same as when we deployed it), publish it, and make transactions with it. This process requires the connection of Remix and Metamask.

- *Pinata.cloud:* An internet file API used to upload data. This connects our smart contract to the IPFS system, which allows uploading and retrieving data. When a seller wants to list their data, they will upload a file, and the file will be transmitted into Pinata cloud, tagged with an unique CID. When a buyer buys data, they will be able to attain the file via an internal dataID (or unique key). The CID is used to fetch specific data in the cloud, but is not accessible by the user, who can only access it through the dataID.

- *Next.js:* A React-based web framework that serves as both the frontend for building an intuitive user interface and the backend for managing interactions between users, APIs, and the blockchain system. It enables tasks like file uploads, dataset browsing, and data purchasing while seamlessly integrating with MetaMask for wallet connectivity and transaction signing. Using API routes, Next.js processes requests from the frontend, acts as an intermediary to services like Pinata and smart contracts, and facilitates blockchain interactions via Ethers.js. By simplifying complex blockchain operations and providing real-time updates, Next.js ensures a user-friendly, scalable, and high-performance platform, making it essential for this data exchange system.

- *TenSeal:* A python library used to perform homomorphic encryption. This allows the encrypted dataset to perform computations.

4.2 Design Logistics

To ensure user protection and data privacy, data must be encrypted before being listed by the seller (Also remove any Personal Identifiable Information). Once the data is uploaded into IPFS, the data cannot be accessed without the CID. The only way for someone other than the seller to obtain the CID of a specific encrypted data file is to purchase it on the marketplace, see Figure 1. The purpose of data encryption is to hide sensitive user information, like patient information of a medical dataset. We want users to not access the data, but only be able to run computations on the encrypted data. Prior to purchasing encrypted data, buyers have the ability to preview the description or specifications of a data file. The description or overview of the data file shows a snapshot of information to any users on the market. For example, the snapshot of a listing could be a short description, followed by the size of the file, and the features of the data. Only when a buyer purchases a dataset or data file, can they use the encrypted for computations but not for downloading. Downloading data out of the marketplace will ruin the privacy of the data, but also lead to other issues like sharing data without permission from the seller.

5 Methods

5.1 Setting the Smart Contract in Motion

The smart contract operates as the backbone of the data marketplace by managing user interactions, enforcing permissions, and maintaining transactional integrity. It is deployed on the Amoy Test Network using Remix IDE, where it is compiled, connected to MetaMask, and deployed. Deploying the contract requires gas fees in ETH, after which users with MetaMask accounts can interact with it. Each user has a unique public address, allowing

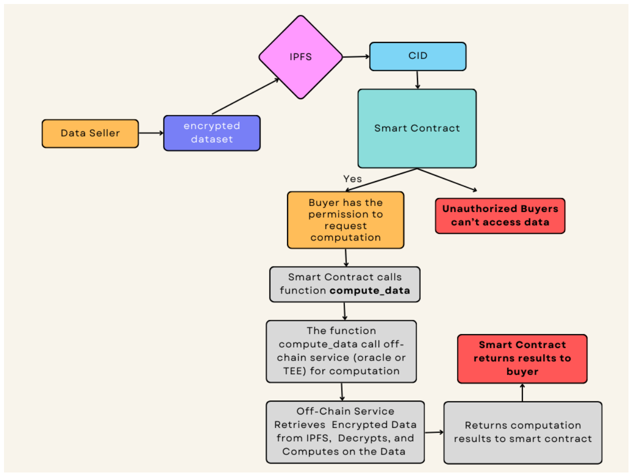
IPFS
CID
Data Seller
encrypted dataset
Smart Contract
Yes
Buyer has the permission to request computation
Unauthorized Buyers can't access data
Smart Contract calls function **compute_data**
The function compute_data call off-chain service (oracle or TEE) for computation
Smart Contract returns results to buyer
Off-Chain Service Retrieves Encrypted Data from IPFS, Decrypts, and Computes on the Data
Returns computation results to smart contract

Figure 1: *Diagram of Proposed Decentralized Data Marketplace*

them to function as a buyer or seller, depending on their actions within the marketplace.

Sellers use the listData() function to upload files, set prices, and define license terms, making the data available for public viewing via shared get functions. These get functions allow all users to access dataset descriptions and details. Buyers, on the other hand, use the buyData() function to purchase datasets, provided they meet criteria such as sufficient funds and eligibility. Sellers retain exclusive control over their listings and can use the delistData() function to remove their items, while buyers are restricted to functions relevant to purchasing.

The permissions and roles are represented in Figure 2, where sellers can write and call vertical functions, such as listData() and delistData(), while buyers interact horizontally, accessing public methods like get and buyData(). Some functions require specific conditions to be met before execution, ensuring secure transactions and preventing unauthorized actions. This structure enforces clear boundaries and functionality for buyers and sellers while supporting public visibility of listing details.

5.2 Encryption Development

To protect PII (personal identification information) from the transaction process, the sellers need to encrypt the dataset before uploading it to the IPFS platform. Therefore, the encryption service is needed to achieve the data security in the Data Marketplace. We started from the AES (Advanced Encryption Standard). AES is a symmetric encryption method that is highly secure and efficient for encrypting large datasets. We utilized the python library cryptography to perform the encryption task. After implementation of AES encryption to a dataset, an encryption key will be generated, which will be further utilized during the decryption process. However, AES encryption needs to be decrypted for data computation. This will increase the risks of data leak after a dataset is bought. Therefore, we begin to explore homomorphic encryption.

Homomorphic encryption enables computations to be performed directly on encrypted data.

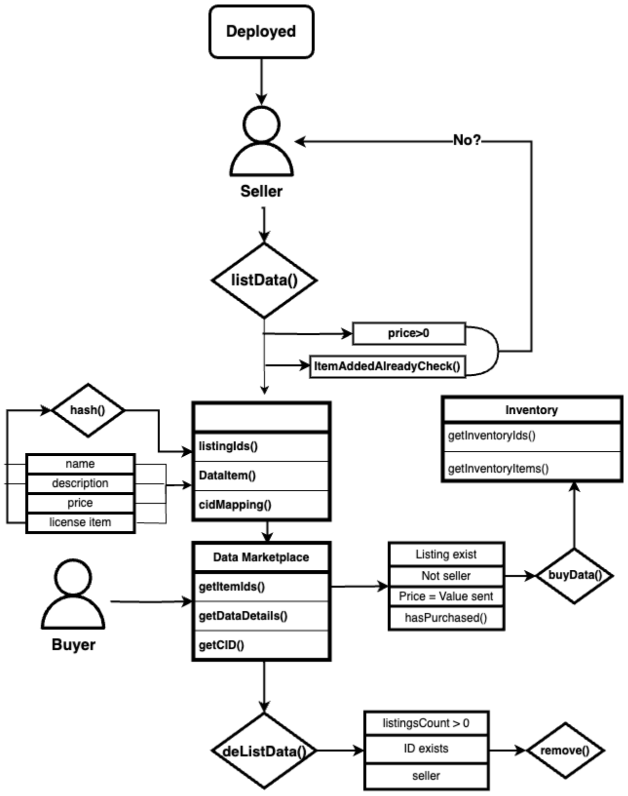

Figure 2: *Low-level implementation of the smart contract*

This produces an encrypted result that, when decrypted, matches the outcome of the computation. This ensures data privacy during the computation process. To implement homomorphic encryption, libraries like TenSEAL or Microsoft SEAL can be utilized. We explored the TenSeal library. After the encryption, the encrypted data can perform computation tasks such as addition and multiplication, even element-wise addition.

5.3 IPFS Integration

File storage is decentralized, and handled through the InterPlanetary File System (IPFS) via Pinata Cloud for gateway access to the IPFS network. When the listData function is called in the smart contract and the user uploads a corresponding file, it is first sent to Pinata, which stores it on the IPFS network. The IPFS network, through Pinata's gateway provides a CID (Content Identifier) for the file. This unique CID will be used to fetch data from the IPFS network via the CID when the buyData function is called in the smart contract, and fetched data can be retrieved and viewed later when called via the getInventoryItem method in the contract via the data items unique data ID. The CID is protected, so only the purchasing user can view the data. The smart contract calls the javascript functions via the Ethers JavaScript library.

6 User Interaction

6.1 Data buyer (Purchase data file)

In our data marketplace, buyers can interact with listings by first calling a get function to view the data description, ensuring they have enough information before making a purchase (a preview mechanism will be implemented next quarter). To buy a dataset, the buyer uses the buyData function, providing the data's unique identifier (dataId) and their address. The function checks that the buyer is not the seller, does not already own the item, and has sufficient funds, including gas fees required for the transaction. Once completed, the item is added to the buyer's inventory, accessible through the getInventory function, and the seller receives the payment. Buyers must account for gas fees in addition to the dataset price. The platform ensures uniqueness of listings using a hash-based unique ID system, preventing duplication while allowing sellers to delist and relist items with consistent identifiers.

6.2 Data seller (File upload)

In the data marketplace, sellers upload files through the platform by providing details such as the file, price, and description. The backend processes the upload by sending the file to Pinata, which generates a unique CID using IPFS. This CID, along with the seller-provided details, is passed to the smart contract through the listData function. The contract stores the CID and metadata on the blockchain, emits a DataListed event, and makes the dataset available for buyers to view and purchase. Check Figure 3 below for the detail procedure:

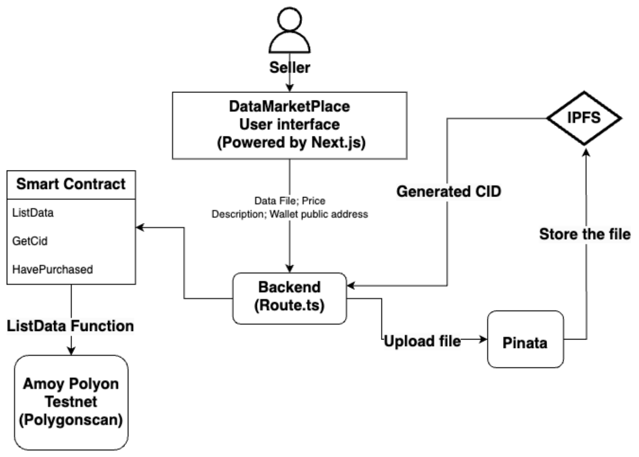
Seller
DataMarketPlace User interface (Powered by Next.js)
IPFS
Smart Contract
ListData
GetCid
HavePurchased
Data File; Price Description; Wallet public address
Generated CID
Store the file
Backend (Route.ts)
ListData Function
Upload file
Pinata
Amoy Polyon Testnet (Polygonscan)

Figure 3: *Procedure Diagram for file upload*

7 Video Demo

Here is the video demo: Watch Video.

8 Discussion

In Q1, our group mainly focused on researching and creating demos for encryption, IPFS, and smart contract. Our main goal involved exploring a wide variety of tools and determining which were necessary for our decentralized Data Marketplace. At first, we began coding the smart contract in Remix IDE (w/ Solidity). Once the contract was built, we focused on using the Pinata API to connect to the IPFS platform. Afterwards, we combined both the smart contract and Pinata API as shown in Figure 1. As demonstrated in the video demo, we successfully implemented the data upload function of the marketplace that incorporates both the smart contract and Pinata IPFS. This accomplishment exhibited the feasibility and capability of implementing a data fetching function in the next quarter.

During the process of linking the smart contract to the IPFS network, we discovered that to access the IPFS network directly, we would have to go through Pinata's gateway. However, to access this gateway programmatically, we would need to have API access keys, which would end up getting exposed if we proceeded with our original plan of letting the file upload to IPFS happen on the seller's computer locally. Hence, we had to adopt the idea of an intermediary server that would process this file upload in a secure manner, where our API access keys would be preserved.

Separated from the smart contract and IPFS, our group also explored and tested the encryption service. After the research, we chose to proceed with the homomorphic encryption, as it allows computation in encrypted data. However, we met some challenges in implementing the homomorphic encryption. The common choice of structuring homomorphic encryption is to utilize the python library TenSeal. When creating the encryption demo, we ran into a kernel crashed issue when executing the library. Possible causes for this issue include lack of hardware resources (RAM), insufficient CPU memories, and improper context configuration. We will continue to solve this problem in Q2 by testing code in different machines and encryption contexts.

In addition, another issue we faced while testing the smart contract was a 'JSON-parse error'. During one of our live demos, this JSON-parse error occurred for the first time, which prevented users from writing to a function. The error kept occurring all day not only for one team member, but for everyone else as well. After a couple hours of debugging and research, the error was resolved with one simple trick: clearing all cache in Metamask.

9 Future Work

In Q2, we will focus on publishing a website onto the web that allows anyone to interact with our data marketplace (Currently what we have is running on local-host). Additionally, if time permits, we aim to implement a recommendation algorithm that recommends similar data sets on the basis of data quality as well as popularity. The popularity of a dataset can be measured through upvotes and downvotes like what you would see on Reddit. Besides, we plan to design a token economy to incentivize the provision of high quality data; Implement reputation mechanisms to reward good actors and punish those who provide low quality or corrupted data. Lastly, we could also create a system to facilitate custom dataset requests.

We aim to build a vibrant data marketplace that ensures secure transactions with the incorporation of an encryption library, IPFS storage, and a cloud server.

References

Fonseca, Jorge, Binay Dahal, and Yoohwan Kim. 2020. *Decentralized Marketplace Using Blockchain, Cryptocurrency, and Swarm Technology*.: 865–882. [Link]

Li, Junqing et al. 2020. "A Blockchain-Based Educational Digital Assets Management System." *IFAC-PapersOnLine* 53 (5): 47–52. [Link]

da Silva Vanin, Fausto Neri et al. 2022. "A Blockchain-Based End-to-End Data Protection Model for Personal Health Records Sharing: A Fully Homomorphic Encryption Approach." *Sensors* 22 (24). [Link]

Sober, M., G. Scaffino, S. Schulte, and S. S. Kanhere. 2022. "A Blockchain-Based IoT Data Marketplace." In *Proceedings of the 20th International Conference on IoT and Blockchain Applications*. Springer. [Link]