A Decentralised Application For Safeguarding Research Data Integrity

Student Name: Kaiyue Peng
Supervisor Name: Ioannis Ivrissmitzis
Submitted as part of the degree of BSc Computer Science to the
Board of Examiners in the Department of Computer Sciences, Durham University

Abstract—Research papers that include survey data collected from the general public are omnipresent, as they provide solid statistical evidence for scientific discoveries in areas from humanitarian studies to national health. However, survey data that researchers solitarily handle is at risk of being fabricated, leaked, falsified, or lost. Proving the honest usage of data can also be difficult when there are no other copies of raw data for comparison. Centralized storage is a more common but less secure data storage alternative, as data centers are not only hacked frequently but also may infringe the privacy of data subjects. Therefore, we propose a decentralized web platform in which researchers 'fundraise' for data for their projects and when enough test subjects have contributed, the researcher can access data from the IPFS decentralized storage that combines encryption and a smart-contract-driven decentralized cloud SQL database, Tableland, for access control. This application has the potential to prevent data dishonesty and some mishandling actions in the management of academic research data.

Index Terms—Data encryption; Distributed applications; Distributed file systems; Distributed systems

1 Introduction

Research data management refers to the handling of research data during and after a research activity. The process of data management should adhere to the FAIR standards of being findable, accessible, interoperable, and reusable.[1] On the other hand, data handling is sometimes used as a synonym, but according to the definition of the Office of Research Integrity of the US Department of Health and Human Services is the process of ensuring that research data is stored, archived, or disposed of safely and securely during and after the conclusion of a research project[2]. Data security is the practice of protecting digital information from unauthorized access, corruption, or theft throughout its entire life cycle. It's a concept that encompasses every aspect of information security from the physical security of hardware and storage devices to administrative and access controls, as well as the logical security of software applications. It also includes organizational policies and procedures. In other words, the purpose of data management is to ensure the integrity of data (i.e. the overall accuracy, completeness, and consistency of data), which is difficult to maintain especially when it is stored online[3].

Data integrity can be hampered under three circumstances:

1) Intentional manipulation by the researcher. There are several ways that data integrity can be affected negatively by researchers:
    a) Fabrication: making up of research findings[4]
    b) Falsification: manipulating research data to give a false impression [4]
    c) Cherry-picking: intentional usage of limited categories of selection for their participants, to carry out their experiment[5]
2) Unintentional mishandling by anyone who has stored or accessed the data, which means disorganized or otherwise poor data archival or loss of research data
3) Malicious leak internally by staff members or externally by hackers. This is the most common reason for compromised data integrity is malicious breach or removal. A data security breach occurs when there is a loss or theft of, or other unauthorized access to, sensitive personally identifiable information[6].

Research data collected from human participants must also be transparent, which means being used with integrity, lawfully, fairly, and traceable, for valid purposes. Individuals and businesses should know what data is being collected, who can access it, how it is being used, and how they can interact with it [7]. Participants should have the right of access, which is the right to view their data held by anyone else on request[8].

There have been several mature projects developed to ensure data integrity and transparency, including Bloxberg which provides a data certification system and a structure to develop decentralized applications for scientists[9], The Dataverse Project which is a centralized open-source web application that facilitates making data available to others, and allows you to replicate others' work more easily[10] and OriginStamp which is a blockchain-backed system for decentralized trusted time-stamping, offer proof of the existence of documents[11]. However, there is not yet a tool that ensures the security and transparency of online data collection, where violation of research ethics principles such as consent, risk, privacy, anonymity, confidentiality, and autonomy could occur[12]. Therefore, a system that uses

blockchain to collect and enforce the integrity and transparency of online survey data would fill the research gap and be beneficial to researchers who would like to prove or ensure the integrity of data and reviewers who would like to check for research misconduct, as proposed in the project plan objectives as below.

1.1 Deliverables

Basic

1) Frontend: creates a web interface that authenticates users and researchers to publish or participate in surveys. Users can view their survey responses and researchers can download survey data. All user responses should be encrypted.
2) Set up an IPFS decentralized database (data management system) to store survey data
3) Connect the web interface to the database for uploading or accessing survey data

Intermediate

1) Create role of reviewer. Creates a registration and authentication, invitation system for reviewers.
2) Creates a feature in the interface for researchers to upload encrypted cleaned data for review. The reviewer certifies the survey to prove the data integrity of a survey.
3) Impose a small charge on users when they upload surveys or responses to discourage empty survey proposals or responses

Advanced

1) Optimization aiming at improving user experience. Write a test for concurrent use by many users submitting responses at the same time and evaluate performance (in other words, test for scalability)
2) Deploy the DApp
3) User study for usability, accessibility, and user experience (UX), and an expert (possibly structured) interviews to test for application idea

1.2 Achievements

This project has successfully created a functional web-based system that has met most of the deliverables. It has a highly accessible and streamlined UI as shown in Figs. 1, 2 and 3 while having a competitive cost of almost zero for its users which does not increase with the size of the survey or data.

2 Related Work

This section presents the background of key technologies of the project and a survey of existing work on the problems that this project addresses.

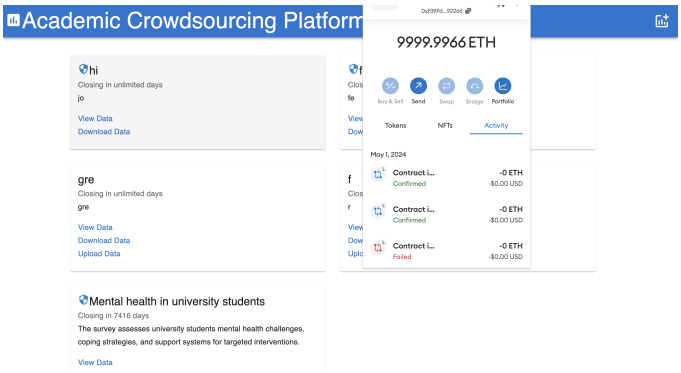

Fig. 1. Researchers' homepage lists the surveys they initiated.

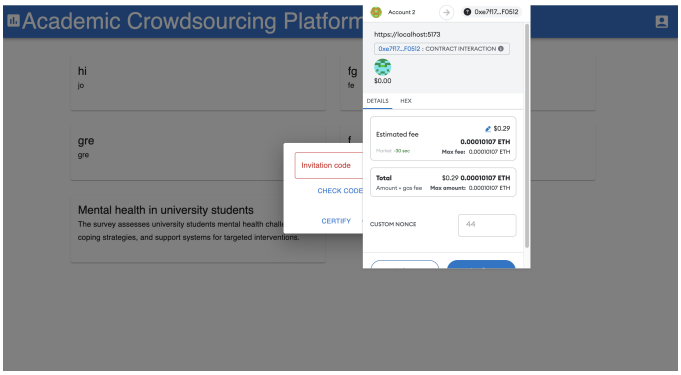

Fig. 2. Researchers' homepage lists the surveys they initiated. They can create surveys with the built-in survey-making tool and providing their public key for encryption.

2.1 Decentralisation

Decentralized networks have been popular in data storage due to their higher reliability compared to centralized storage. A decentralized network can be permissioned, when new nodes are only allowed after the agreement of existing node owners in the networks. In contrast, anyone with the hardware can join permissionless or public networks.[13] This project uses public networks for their higher reliability brought by the larger number of nodes.

2.2 Interplanetary File System (IPFS)

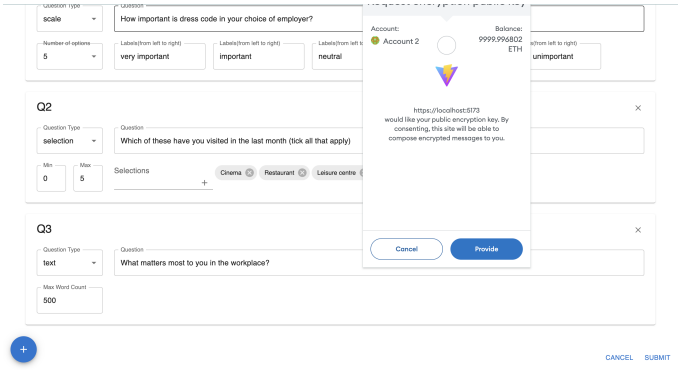

Fig. 3. Researcher can create surveys with the built-in survey-making tool and providing their public key for encryption.

> As a global point-to-point distributed file storage system, the IPFS was originally designed to optimize the current hypertext transfer protocol and then gather computer devices with the same file system together to form a huge distributed system. For large storage files, IPFS will automatically slice files into multiple servers for storage and synchronously download and splice files upon request to speed up file access. [14]

Different from location-based addressing using Uniform Resource Locators (URLs) to locate the file, this content search mechanism assigns a unique hash value to each file in the network based on the file content and maintains a distributed hash table to find the required file. This is called content addressing.[14] The major limitation of IPFS as decentralized storage is that files are not copied to other nodes unless the other node providers are paid to store files not of their own. This is called a pinning service and there are several service providers such as Filebase, Pinata, and Web3.Storage who need a monthly subscription depending on features and storage size you need. However, the files are stored exclusively on the nodes being pinned, so if the service provider removes files from nodes when they do not receive payment or accidentally have a false garbage collection, the content may be lost entirely. To ensure the persistence of the files, the Filecoin blockchain was introduced to incentivize the storage providers with cryptocurrency. The larger the storage of the storage provider, the more FIL they receive as a reward.[15] This is elaborated further in the next section.

2.3 Blockchain

In 1991, Stuart Haber and W. Scott Stornetta proposed a way to timestamp data without keeping track of the actual content by checking hashes.[16] This idea set the foundation for the current blockchain system [17] and was made popular by Bitcoin two decade later[18]. Anjee Gorkhali, Ling Li, and Asim Shrestha defined blockchain as "a distributed database or digital ledger that records transactions of value using a cryptographic signature that is inherently resistant to modification." [13] As its name describes, each consists of information blocks chained together cryptographically for immutability checks. Each block contains a hash (a digital fingerprint or unique identifier), timestamped batches of recent valid transactions, and the hash of the previous block. Any block alteration or insertion is quickly identifiable by checking if the hash stored in the next block would be different from the one before. To further ensure the validity of data on the chain, there is a large network of computers that each stores a copy of the chain and creates new blocks validated by a secure consensus mechanism before broadcasting them to other nodes. Nodes that create validated blocks are called miners and are rewarded with cryptocurrencies.

Mining is the process of finding the nonce, for each block, in order to link it with the next one. Each time one of the miners finds one of these cryptographic keys a bitcoin is "mined" and they receive payment in this same currency.[19]. In recent years, blockchain technology has been widely applied in various areas due to its immutability, cryptocurrency, and reliability.[20]

The consensus mechanism in blockchains is the mechanism that allows decentralized networks to come to a consensus (i.e. all nodes agree) on things like account balances and the order of transactions[18]. There are many consensus mechanisms, even some non-cryptocurrency decentralized networks like Tableland use them to reach a deterministic state in the network. Here are a few examples of consensus mechanisms used by popular chains:

- Proof of Work is when all miner nodes compete in finding the correct nonce to hash a block. Miners with the most computational resources are more likely to find the solution before other miners and claim the currency in the block.[18] This is used by the infamous Bitcoin and Ethereum before 2022.
- Proof of Stake is a way to prove that validators have deposited something of value into the network that can be destroyed if they act dishonestly. The possibility of being chosen as a validator is proportional to the amount of assets deposited.[18] It is used by large blockchains like Solona and Ethereum 2.0.
- Proof of Authority is achieved when preapproved validators use software to organize transactions into blocks. The process is automated, and so the validators don't need to monitor their computers constantly. That, however, means that validators must keep their computers (admin sites) in good working order. The most notable platforms using PoA are VeChain, Bitgert, Palm Network, and Xodex.
- Expected Consensus is a consensus mechanism used by the largest storage-based blockchain, Filecoin. A node gains the right to participate in a consensus protocol that can be run by any set of weighted participants and create blocks by lending storage capacity to the chain. Participants are weighted based on their storage capacity and in return be compensated with a financial reward whenever their blocks are included on-chain.[15]

No matter how efficient the consensus mechanism is, price and performance problems arise when an increasing number of nodes join the blockchain. Ethereum Mainnet is only able to process roughly 15 transactions per second. When the demand to use Ethereum is high, the network becomes congested, which increases transaction fees and prices out users who cannot afford those fees. This is when Layer 2 (L2) blockchains appear as a scaling solution. L2 is a collective term to describe a specific set of L1 scaling solutions. Layer 2 is a separate blockchain that extends L1 and inherits the security guarantees of Ethereum. Layer 1 (L1) blockchains such as Ethereum and Bitcoin have the maximum decentralization but result in slow block creation speed and low scalability, whereas L2 networks such as Polygon built on Ethereum can utilize the security features of the main chain while having faster block creation. L2 blockchains can also have a different consensus mechanism to its base chain, such as Ronin which is an Ethereum sidechain using PoA, and Sepolia which is an Ethereum testnet using PoA. In addition, storage-centered chains such as Filecoin are an L0 network. All blockchains have a dedicated testing chain referred to as a 'testnet' where developers run end-to-end testing for their smart contracts with funds

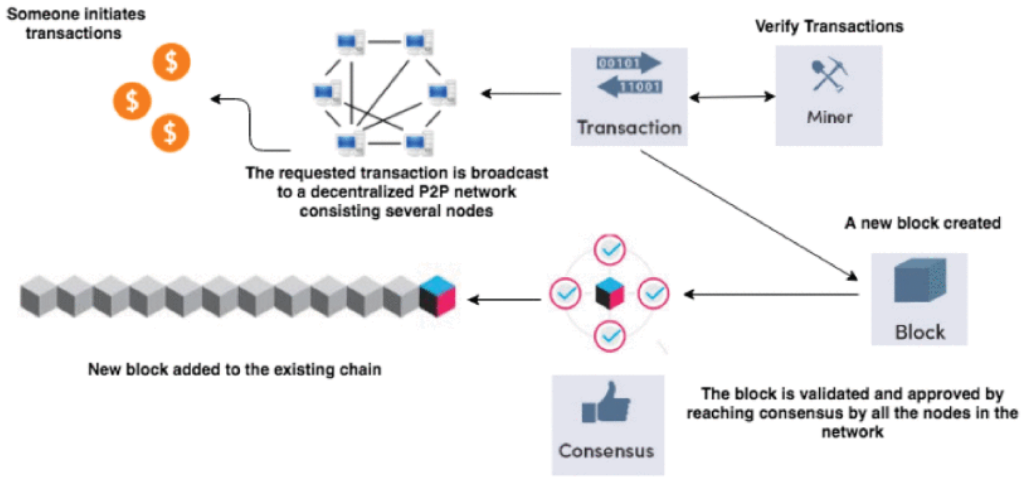

Fig. 4. Functional diagram of a blockchain network[22]

requested through a faucet service. Tokens or crypto on testnets have no economic value[21].

The logic of blockchains is summarised in Fig. 4 and the blockchain comparisons mentioned above are concluded in TABLE 1

TABLE 1
Summary of Properties of Popular Blockchains

Blockchain	Crypto	Layer	Consensus	Testnet
Ethereum	ETH	1	PoS	Sepolia
Bitcoin	BTC	1	PoW	Bitcoin testnet
Polygon	MATIC	2	PoS	Mumbai
Filecoin	FIL	0	EC	Calibration

2.4 Smart Contracts and DApps

Smart contracts are scripts that can be run on any blockchain that employs an Ethereum Virtual Machine. They are often written in a Turing complete language Solidity and are stored in blocks like transactional histories and messages. Like any wallet, smart contracts exist on a specific address on the blockchain and can be paid to write data to its variables or pay to other smart contracts or wallets, but any read-only functions are free. As a piece of on-chain data, it is distributed, immutable, and openly accessible to any node, therefore any smart contract application is open-source by nature. If there is a problem with the implementation logic in the contract code, it will seriously affect the security of the blockchain. Apart from its functions being called by users, this program can also be self-executing, i.e. executed when predetermined conditions are met. This technology can be used in business management[20], healthcare[23][24][25], government[17], crowdfunding[26], data management[27] and academia [11][13][28].

Decntralised Applications (DApps) are programs that run in WEB 3.0 which is deemed to be the Internet of the future. Unlike the current WEB 2.0, WEB 3.0 uses content addressing instead of location addressing as all nodes in the network share a copy of all data. It also brings some additional features such as being verifiable, self-governed (anyone can create Decentralised Autonomous Organisations), permissionless, and distributed. This will be implemented with decentralized networks, blockchains, or even combined. Anubis. Christofer and Leithardt, commented that

> Cryptocurrency plays a big role in many of WEB 3.0 protocols, as it provides a financial incentive for nodes that want to participate in the creation, governance, contribution, or enhancement of a project. The projects that are developed on top of this Web3 system end up offering a variety of services such as computing, storage, hosting among other services that were mainly provided by cloud providers. The users that consume these Web3 services pay to use the protocol, but in this case the money goes directly to the network participants, eliminating unnecessary intermediaries. [24, p. 5]

This project aims to build a secure system that is publicly accessible, so all data, frontend, and backend scripts, and running environments are decentralized. User data would be stored in Filecoin, the frontend scripts would be deployed on IPFS, and the backend scripts would be written in smart contracts deployed on Polygon and Tableland.

2.5 Existing Systems

The following systems were found through proof-of-concept papers on Google Scholar or the references of relevant papers. In the beginning, the project's literature research covered academic data management systems only, such as Dataverse, CryptSubmit, and Bloxberg, but the number of decentralized or distributed systems that was found was very small. They also had different aims or key technologies from this project. Fortunately, when the search was expanded to include decentralized data management in other fields, especially healthcare, where the data is highly sensitive, more relevant literature was found.

CryptSubmit [11]is the proof-of-concept of a successful time-stamping website called OriginStamps, which offers timestamping services for academic manuscripts and reviews. It creates a hash for submitted manuscripts and

reviews, and pushes the hash into the blockchain Bitcoin as a transaction, so this information is available in the history. To save cost, it used a time-sensitive algorithm of aggregating all hashes submitted in 24 hours into a valid Bitcoin address and then transferring the smallest unit of Bitcoin to this address instead of turning each hash into a transactional address. This can effectively prevent researchers' unpublished manuscripts from being plagiarized by reviewers or publishing organizations or reviewers' ideas from being claimed by researchers. This has inspired this project to use a Filecoin service that uses a similar approach of bundling transactions to cut costs. However, this system relies on researchers being trustworthy with the data they collect, as CryptSubmit only keeps the hash of the data, not the original files. If researchers falsify or lose the data, there is no copy of the original data to check with or recover from. In comparison, this project is trustless in that researchers cannot interfere with the data collection process and reviewers are responsible for ensuring the data integrity in the data cleaning process.

Bloxberg [9] is a PoA permissioned blockchain that currently has 11 authority nodes in Germany, Cyprus, Denmark, UK, USA, South Africa, Switzerland, Serbia, Bosnia and Herzegovina. The blockchain aims to include a node from each research institution on the network, so they can make governance decisions such as certifying or sharing research data, publishing papers, and crowdfunding research. Its permanent data certification for data integrity inspired the certification system in this project. It has its own ecosystem of DApps and a cryptocurrency berg, but all transactions are free of charge. Better than CryptSubmit, data can not only be certified with a timestamp but also can be stored on-chain to prevent data loss. Despite that, it still requires the researcher to be trustworthy of the data sent for certification being the same as the original.

Dataverse [10] is a distributed open-source web application that aims to promote research data sharing. According to their white paper by Gary King from Harvard University, Cambridge, and Massachusetts,

> A Dataverse repository is the software installation, which then hosts multiple virtual archives called Dataverse collections. Each Dataverse collection contains datasets, and each dataset contains descriptive metadata and data files (including documentation and code that accompany the data). [10]

It solves the risks of data leakage or loss during sharing, but again relies too much on the researchers uploading the data being trustworthy.

Kieran[28] proposed a system very similar to this project, but the user data was stored in a centralized database and the data-cleaning process was not taken into account as the reviewers only compare the hashes between the original and the copy the researcher provides. In contrast, this project accommodates data cleaning while replacing the database with decentralized storage to ensure data immutability. Despite its room for improvement, this paper offered valuable inspiration for creating the reviewer role, which checks data integrity before paper publication.

Several inspirational attempts have been made in social areas too. Shikha *et al.*[26] proposed a socially responsible crowdfunding platform in which an Ethereum smart contract collects funds from funders of a campaign and sends them to the campaign owner if the target is met and sends them back to the funders otherwise. Campaign details are stored in IPFS and they prototyped a DApp that used MetaMask as their authentication and payment system. This paper has inspired this project also to use MetaMask. This system also mentioned that in anonymous platforms, the cost of flooding large quantities of empty data into the system is almost zero and fence they introduced a small registration deposit which inspired this project to charge on every 'write' request. This charge should be enough to discourage spammers but not prevent genuine users. This would be more effective than limiting the number of responses or surveys per user because hackers can easily create a large number of accounts. In contrast, Mijanur *et al.*[17] proposed a decentralized governmental complaint platform for complainants to raise complaints anonymously without the history of their complaint process being modified or removed by the complained authorities. All user data are saved in IPFS and hashed to be timestamped in a permissioned Aries. Authentication is implemented through Hyperledger Indy for complete anonymity. This system may have vulnerabilities of distributed denial of service (DDoS), Sybil, and "false-reporting" attacks[26] as there is no cost of raising complaints.

Another field with similar requirements to research data management is the storage of Electronic Medical Records(EMR). Three systems[24][23][25] were proposed to encrypt the files before uploading them to IPFS. However, they either store data pointers to IPFS on-chain or in a cloud service[23] to reduce cost. This tradeoff between decentralization and cost is mitigated in this project by using Tableland, a decentralized cloud service.

TABLE 2 shows that most successful decentralized data storage systems have used permissionless Ethereum, AES encryption, and IPFS for off-chain storage. After evaluating their performances in each existing system, this project has used AES-encrypted IPFS and Ethereum-based blockchain Polygon as they all positively contributed to the stability, speed, and security of the systems that utilized them.

3 METHODOLOGY

This section presents the solutions to the problem in detail, introduces the technologies chosen for the implementation, and provides justifications for the design choices.

3.1 Introduction of Development Tools and Setup

Tools used for development are listed as below:

- Node.js 20.11.0. Node.js is an asynchronous, event-driven JavaScript runtime designed to create scalable web applications. It is built on Chrome's V8 JavaScript engine and can be compiled into machine code to run on web servers.
- npm 10.2.4. npm is an open-source TypeScript and JavaScript package registry, where anyone can upload and download packages with its Command Line Interface (CLI). npm is bundled with Node.js files to be downloaded together.

Backend:

TABLE 2
Comparisons Between This Projects and Systems Mentioned

Project	Field	Network	Type	Encryption	Off-chain
[11]	Academia	Bitcoin	Permissionless	None	None
[25]	Healthcare	any blockchain	Permissioned	ECDH	IPFS
[9]	Academia	bloxberg blockchain	Permissioned	Unkown	None
[24]	Healthcare	Ethereum	Permissionless	ECC, RSA, AES	IPFS
[26]	Crowdfunding	Polygon	Permissionless	None	IPFS
[17]	Governmental	Hyperledger Indy and Aries	Permissioned	Unkown	IPFS
[10]	Research data	distributed dataverse network	Permissioned	Unkown	No chain
[23]	Healthcare	Ethereum	Permissionless	AES	IPFS, cloud
[20]	Business management	Ethereum	Permissionless	ABE	IPFS
[28]	Research data	Ethereum	Permissionless	None	centralised MongoDB
This project	Research data	Polygon	Permissionless	AES, ECDH	IPFS, Tableland

- Hardhat 2.19.5. Hardhat is an Ethereum development environment for deploying smart contracts, running tests, and debugging Solidity code locally.[29] It can be downloaded as an npm package.

Frontend:

- Vite 4.4.5. Vite is a build tool that aims to provide a faster and leaner development experience for modern web projects.[30] It is chosen over other build tools for its speed of re-rendering after file changes, making the development process faster.
- React 18.2.0. React is a JavaScript library created by Facebook for building User Interface components. Material UI is an open-source React component library having out-of-the-box code for accessible and uniformly styled components.
- tweetnacl 1.0.3. This is an x25519-xsalsa20-poly1305 encryption and decryption library that was audited by Cure53 in January-February 2017 and given exceptionally positive verdict[31]. Although the review is very likely to be outdated, this library is still more likely to be more secure than other cryptography libraries without any review. This library is also used by the MetaMask official npm package of Ethereum signing functions[32].
- ethers 5.7.2. The ethers.js library aims to be a complete and compact library for interacting with the Ethereum Blockchain and its ecosystem[33]. ethers.js5 is the only library used in Tableland documentation, so its version 6 or other blockchain packages such as web3.js are not considered in this application to avoid potential compatibility or stability issues.

3.2 Testing

The following tools were used for testing.

Backend testing:

- Hardhat 2.19.5. It was used in the command line to create a local blockchain for testing.
- Local Tableland. In the development environment, it is used to create a local blockchain and Tableland network by running a Hardhat node.

Frontend testing:

- Vite 4.4.5. Vite a corresponding testing package Vitest that was used to write unit tests on UI components by simulating user interactions and validating the states of components after interactions.
- Chrome Version 123.0.6312.123 (Official Build) (arm64) loaded with MetaMask 11.13.1

For each stage in development, unit tests were carried out. Integration testing was performed after unit tests were passed. The testing coverage of the completed features is around 50-100%. All tests carried out were passed speedily and connected to public production NFT.Storage, Metamask, and local Tableland.

3.3 Algorithm of User Workflow

This application requires the user to be one of the three roles:

1) *Researcher* who has a research proposal and needs to survey data for their researcher. They do not have to know the people being surveyed in person.
2) *Reviewer* who is invited by the researcher to oversee their data processing and certify their data integrity. They need to know the researcher in person.
3) *Respondent* who is willing to respond to the researcher's survey.

Researchers and respondents both need to pay a small fee to upload their surveys or responses to reduce the risk of DDoS, Sybil, and "false-reporting" attacks, as proposed in 'The Potential of Blockchain Technology in Socially Responsible Crowdfunding Platforms' [26] in which small registration deposit is required to prevent empty registration requests. The specifics of the cost will be discussed in Section 5.1 Cost.

The application has four components as illustrated in Fig. 5:

1) A web interface that users directly interact with
2) A secure authentication system
3) A distributed storage of survey metadata including questions, consent forms and descriptions, and encrypted user data
4) A blockchain-based storage of pointers to data files, encryption keys, and encrypted decryption keys. This may be referred to as a 'cache' later.

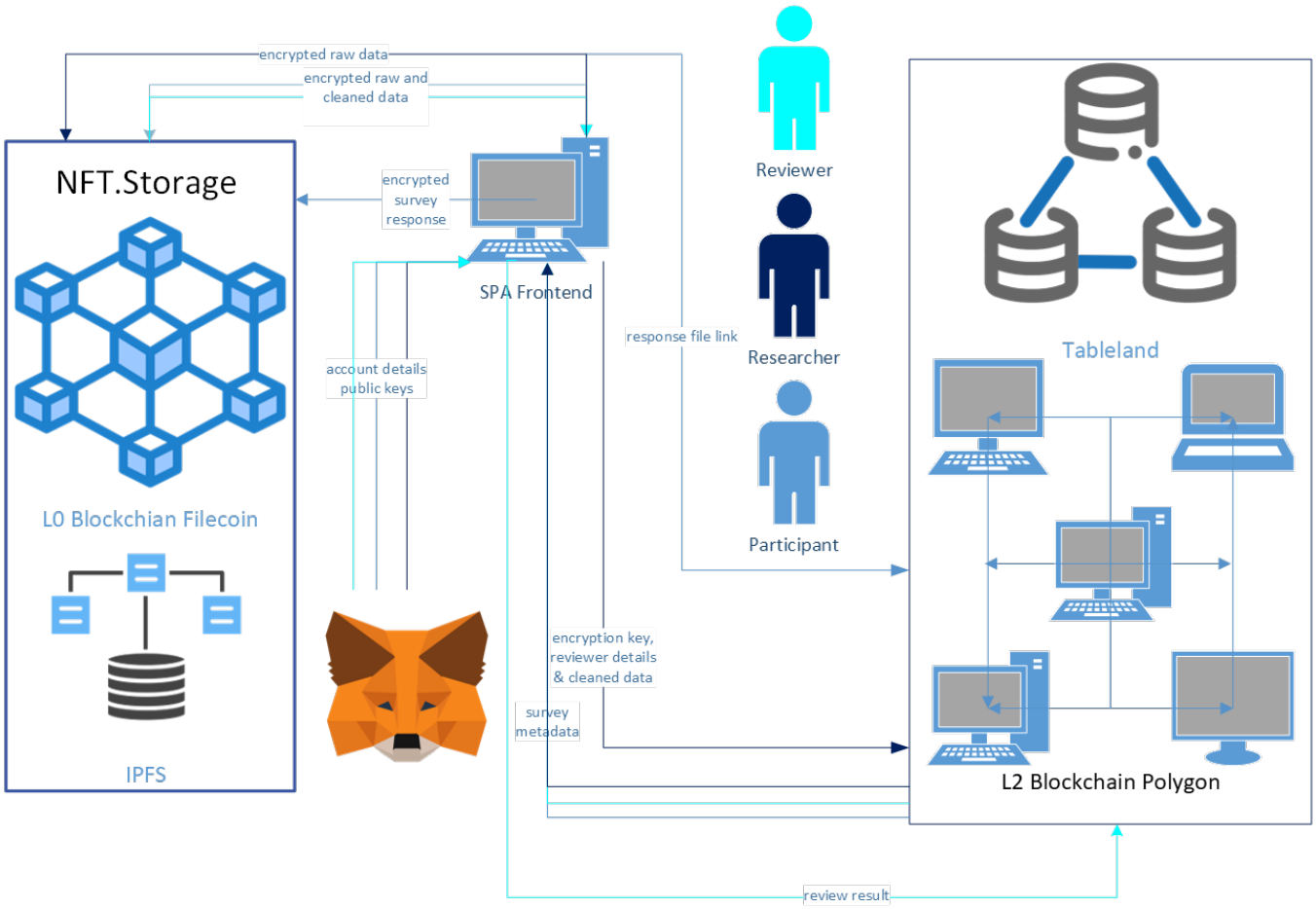

Fig. 5. Data flow of the application

All the life stages of a survey in the application are as follows:

1) A researcher posts a survey on the web interface, detailing the closing date, description, questions, etc. Survey questions are saved as Likert, selection, or open-text questions as they are the most basic types of survey questions[34]. This is saved in blockchain-based storage and questions are stored in IPFS
2) A respondent sees this survey post on the web interface and fills out the form. This response is recorded as an array to save storage and is encrypted to save in IPFS as an array. The address of this file and its decryption keys are saved in the blockchain-base storage.
3) The researcher uses the web interface which automatically decrypts the response files one by one, and merges the decrypted arrays into one .csv file for download.
4) The researcher cleans the data in their devices locally to maximize data integrity. Considering the researcher cannot supervise the process of data collection in online survey platforms like this, invalid responses such as empty or incomplete entries, false understanding of questions, and erroneous submissions could occur in large quantities[35] that data processing software is required to ensure the usability of the dataset. To resolve the vulnerability of localizing storage introduced in this step, a reviewer is therefore introduced.
5) The researcher invites a reviewer by entering their public key and blockchain wallet ID on the web interface.
6) After data cleaning, the researcher uploads the cleaned .csv file to the web interface. It encrypts the cleaned data file with the reviewer's public key and uploads the encrypted file to IPFS. This cleaned data file address is saved to blockchain-based storage.
7) The interface also encrypts keys to original response files with the reviewer's public key. Encrypted keys are sent to the reviewer via blockchain so this is permanently recorded and proves it is sent for review. The reviewer uses them to get original data (and compare them with the cleaned data file to validate that the data cleaning process did not affect the data integrity.
8) The reviewer updates the review result to the cache.

3.4 Choices of Implementation Tools

The tools used for each component of the application are as follows:

3.4.1 Web interface

A Single-Page Application written by Typescript 5.2.2. TypeScript is a strongly typed programming language that builds on JavaScript, making the debugging process easier with type-checking. It is an npm package that compiles into JavaScript, so TypeScript can run on any platform where JS can, avoiding browser compatibility issues.

3.4.2 MetaMask

According to Anubis G.M. Rossetto, Christofer Sega and Valderi R.Q. Leithardt, MetaMask is

> an encrypted (digital) wallet and gateway to blockchain applications that enables users to manage their accounts, keys, and tokens in a variety of ways, including hardware wallets, and isolates the user from the context of the website. It is available as a browser extension and as a mobile app. [24, p. 5]

It allows users to import accounts from any blockchain, including local ones for development and testing. Its React SDK allows DApps to interact with the browser plugin for connection, smart contract calls, authentication, etc. They use a decentralized authentication system in which account credentials are not stored by them but by the account owners, i.e. users in a form called 'Secret Recovery Phrase'. During registration, a user chooses a secure password to encrypt a unique 12-word phrase generated for them, and this encrypted text will be stored locally to sign all transactions and to recover funds if the user loses their password. In comparison to the traditional centralized account and password approach, a self-custody wallet like this is a safer authentication approach as there is no centralized authority that can act maliciously or be hacked into data leaks.

3.4.3 NFT.Storage

Free decentralized storage for Non-Fungible Tokens which are unique cryptographic tokens that exist on a blockchain and cannot be replicated. The platform aims to ensure reliable storage for off-chain NFT data by leveraging the decentralized and verifiable infrastructure of the Filecoin network, augmented by IPFS's content addressing and peer-to-peer networking capabilities, offering creators and marketplaces the assurance of enduring accessibility and integrity[36]. It is more popular to use IPFS node providers such as Filebase and Infura[24], but adding a central authority to a decentralized system introduces several vulnerabilities. The provider may make their nodes inaccessible unless a surging fee is paid, or remove their nodes from the IPFS network when they cannot afford to be a part of the network and cause survey data files in their nodes to be lost permanently. Filecoin, on the other hand, solves the problem by incentivizing node providers with a cryptocurrency FIL so storage can remain free. Similar to node providers, NFT.Storage offers an npm package for DApps to send write and read requests to its HTTPS API.

3.4.4 Tableland

'Tableland is an open-source, permissionless cloud database built on SQLite. It sits alongside Ethereum and Layer 2 chains but is purely focused on data storage and retrieval, as said on their official website. It is a distributed network of validator nodes that store SQL tables as NFTs and has advanced row-level and column-level access control features that can be set in a smart contract deployed by the user on any supported blockchains. Instead of storing data on blockchains, only the SQL statement itself and custom access controls defined in smart contracts are written to event logs by a registry smart contract on each base chain. Tableland validator nodes simply watch the registry and mutate a local SQLite database with the database instructions and the data is accessible using SQL read queries at an HTTPS gateway. As it relies on smart contract calls to trigger events of SQL materialization, its supported chains have to have support EVM. In section 3.5 Data Storage, it will be compared with on-chain storage to justify its suitability in this project.

3.5 Encryption

This section explains all the security considerations undertaken when choosing encryption algorithms for this application to ensure the security of the application, especially the integrity of data uploaded and stored. All strong encryption algorithms rely heavily on a secure entropy source which is either provided in the browser by user interactions or accessed through the Crypto.getRandomValues function which seeds with the user's operating system's entropy source[37]. In this application, the latter approach is used for a lower complexity.

All data stored in distributed storage such as IPFS and blockchains are public, so a strong encryption algorithm is necessary to prevent data leaks. The question is: how to encrypt a respondent's response files so that only the researcher of the correspondent survey, the invited reviewer the researcher invited, and the respondent can read the file anytime?

3.5.1 Asymmetrical Encryption

The most common approach is to use asymmetrical encryption such as the Rivest-Shamir-Adleman algorithm (RSA) where the receiver, a researcher sends their public key to the sender, a respondent who sends the message, which is encrypted with the public key and can be decrypted with the private key. The private key cannot be derived from the public key, and this property allows the public key to be stored in public networks and therefore perfect for data transfer via blockchain[24].

However, the major limitation of using asymmetrical encryption for communications on the blockchain is that the sender does not have the private key and hence cannot decrypt the encrypted messages they sent before. As this project aims to ensure a data subject can access their initially uploaded data anytime without request, asymmetrical encryption on its own is not a feasible option.

3.5.2 Symmetrical Encryption

On the other hand, symmetrical encryption solves this problem by having the same key for decryption and encryption so both the sender and receiver can access the decrypted message anytime. One instance is the Advanced Encryption Standard encryption algorithm developed in 1998 by Joan

Daemen and Vincent Rijmen, it allows a fixed data block size of 128 bits and supports key sizes of 128, 192, and 256 bits, and any combination of data [24]. However, this block cipher approach is unsuitable for encrypting survey responses that would exceed the size of 128 bits or 16 characters and hence would be time-consuming to do sequentially, so a counter mode when blocks can be encrypted and decrypted in parallel is used for this application for optimizing encryption of larger files [38]. Another problem with this AES algorithm is that it requires a random initialization vector generated for encryption alongside the key for decryption, doubling the storage required in Tableland. This IV is often 16 bytes and ensures that even if the same plain text is encrypted multiple times with the same key, the resulting ciphertexts are different. The key and IV should normally be different because an attacker who obtains the encrypted data might have an easier time deducing the key if the IV is pseudorandom and hence predictable. Since the key is generated by the random function based on cryptographically strong Crypto.getRandomValues and each response file's key is unique, this IV can be used the same as the key without compromising security. Meanwhile, the key size is chosen to be 128 bits for a lower cost of smart contract calls to Tableland which is elaborated later.

To exchange the key securely via blockchain, symmetrical and asymmetrical methods are often combined. One such algorithm is Diffie–Hellman key exchange, when the sender and receiver publicly agree on a public key which is then combined with keys from both parties respectively to produce a privately shared key[39].

3.5.3 MetaMask

MetaMask generates a random private key for each of its users and derives a public key from it using the Networking and Cryptography library's implementation of the X25519_XSalsa20_Poly1305 algorithm[40],[41] whenever the *eth_getEncryptionPublicKey* function is called. This is a function implemented on blockchain that returns a public encryption key, or rejects if the user denies the request[42]. Its decryption function *eth_decrypt* is also deployed on the blockchain to decrypt messages encrypted with the same algorithm used to derive the public key from a private key. Being a block cipher, it pads the data to encrypt to blocks of 64 bytes to avoid vulnerabilities introduced by length variations in cipher text. This function also needs another 32B public key and a 24B nonce as parameters other than the cipher text. Using these functions for encryption and decryption is more secure than asking user to enter private key, which makes the site vulnerable to form injection or phishing for the private key. MetaMask also strongly discourages their users from entering their private keys in their official documentation 'Basic Safety and Security Tips for MetaMask', because as soon as private key is obtained by a third party, all funds in the wallet will be lost.

3.5.4 Proposed Encryption Algorithm

For this application, a random 16-byte AES key is generated for each file stored IPFS and used for encrypting the file content, while their MetaMask public key encrypts the AES key before the AES key is stored publicly on Tableland. This means every response even for the same survey from the same user would be encrypted with a different key, limiting the spread of data leaks. This is because if each survey has the same AES key, one respondent leaking the key can cause all other survey responses to be leaked, and if each respondent has the same AES key, the researcher would be able to access one's responses to other surveys if they participated in one of the researcher's surveys. This AES key is then encrypted with respectively the researcher's and the respondent's MetaMask public key using their default X25519_XSalsa20_Poly1305 algorithm.

In summary, the encryption and decryption process of one survey response file is as follows:

1) Respondents encrypt their response file with a unique and cryptographically secure AES key pair before uploading it to NFT.Storage
2) Respondents encrypt the AES key with the researcher's public key and their public key before uploading them to Tableland. The two keys are called 'researcherKey' and 'userKey' respectively.
3) The reviewer sends to the researcher their public key, which the researcher then uses to encrypt the cleaned data file before uploading it toNFT.Storage.
4) The researcher decrypts the 'researcherKey's to reveal the original AES keys, which are then encrypted with the reviewer's public key.
5) The reviewer decrypts the cleaned data file and the AES keys with their private key. The original data files were decrypted with the AES keys.

This process was carefully considered to ensure data integrity is not vulnerable to insecure encryption in the application.

The only limitation is that *eth_getEncryptionPublicKey* and *eth_decrypt* functions are deprecated and may be removed in the future due to security concerns, but a more secure version would be reintroduced soon using other names[43].

3.6 Data Storage

This section aims to elaborate on how Tableland is used in this project and why Tableland is more suitable than on-chain storage despite its limitations.

The two tables in Tableland are as follows: *Glossary* described in TABLE 3 for survey information and *Responses* described in TABLE 4 for response files.

SQLite data types do not include 'date' or 'boolean', but they are converted to string or integers and can be used with functions in SQL statements[44]. On the other hand, these functions are not supported in Tableland so they are replaced with ISO (the International Organization for Standardization) date time string and integer of 0 or 1 respectively. Foreign keys are not supported either but are being considered with constraints[45].

The Tableland technology has some limitations:

- 50000 upper bound to the number of rows per table, which will likely be removed when Tableland launches its production network. This major bottleneck is a significant concern to the scalability of this project. Only 50000 surveys exist in the application

TABLE 3
Glossary table

Column name	Data type	Data description
id	integer primary key	survey id referred in response table
surveyTitle	string	researcher input < 1kb
surveyDescription	string	researcher input < 1kb
closingDate	date	filter out expired surveys for participants
researcherID	string	hex number, wallet address
questionAddress	string	href to web2 gateway of the survey metadata file stored on IPFS
cleanedDataAddress	string	href to web2 gateway of the cleaned data file stored on IPFS
encryptionKey	string	researcher's public key for encrypting AES key
reviewResult	boolean	whether this data passed the check of a reviewer
reviewerPublicKey	string	hex number, reviewer's public key for encrypting AES key & cleaned data

TABLE 4
Responses table

Column name	Data type	Data description
id	integer primary key	
responseAddress	string	href to web2 gateway of the encrypted response file stored on IPFS, all answers are stored in a list
user	string	respondent wallet address
surveyID	integer foreign key	survey id referred in response table
userKey	string	AES key encrypted with respondent public key
researcherKey	string	AES key encrypted with researcher public key

and worse, only 50000 responses can live. If the upper bound is not removed, the best mitigation would be to use a local JavaScript script to automatically create new tables after the old ones are full. A smart contract would be more suitable than a JS server which adds a point centralization to the system, but data in Tableland are inaccessible in on-chain because only queries are stored.

- 1 kilobytes upper bound to the cell size. This means the survey title or description can only be around 175 words.
- 24 upper bound to the number of columns per table. According to Sharma H., a good questionnaire can be of 25 to 30 questions and some surveys have long questionnaires too [46]. This is the primary reason for not storing survey responses directly in Tableland. Meanwhile, open-ended questions such as the ones in ethnography can be longer than 175 words.

Despite all limitations, Tableland has more advantages over on-chain storage in terms of cost, mutability, and data readability, according to the Tableland documentation shown in TABLE 5.

The major problem it resolves is the immutability of survey data after they are uploaded. Research data sometimes need to be modified or deleted for their integrity, such as when test subjects need to update their data, researchers need to delete data after the review is completed, test subjects want to withdraw from the survey, etc. Since IPFS files are immutable, the only solution is to change or delete the links pointing to the redundant data files. On-chain, data modification is simulated via the history of on-chain events and states, but each Tableland node is essentially an SQLite server that only keeps the current state and their data is modifiable. Although the SQL statements saved on-chain are immutable and old data are still available on-chain, the reconstruction of a deleted table from those statements is time-consuming and energy-consuming. Furthermore, this immutability is flexible in that every table has its table controller contract that specifies which operations are allowed and which columns are updatable. For example, DELETE is prohibited in both tables, and only the cleanedDataAddress is modifiable in the *Glossary* table.

TABLE 5
Comparisons Between On-chain Storage and Tableland From Documentation[21]

Metrics	Blockchain	Tableland
Scaling cost	High	Similar to other distributed storage
Mutability	Costly	Cheap, depending on query length
Composability	High	High
Queryability	Unstructured	High, having most of SQL features

Another problem it solves is the readability of data stored on-chain. Blockchains have more nodes than the Tableland network due to the latter being unincentivised, leading to a slower query rate. Only write operations need to pass through the Tableland on-chain registry, so read operations via the Tableland network are theoretically faster than reading from a public blockchain. Other than speed, backend filtering and pagination are made possible in SQL-structured tables, such as the user can filter surveys by title and the web interface can fetch only the surveys that a reviewer needs to review. To write structured data on-chain, there would be an overhead that could potentially lead to more costs than Tableland.

3.7 Blockchain

Fig. 6 shows that Tableland network only reacts to events emitted by the Tableland registry contract deployed on public blockchains in the production environment, so only 6 main nets where a Tableland registry exists are supported.

TABLE 6
Comparisons Between Performance of Tableland on Supported Blockchains[21]

Blockchain	Write Cost (USD/MB)	Create Cost (USD/MB)	Average SQL materialize time
Ethereum	14280.36	25222.38	30 to 40s
Optimism	2392.35	2983.61	<5s
Arbitrum One	1148.45	1349.02	<5s
Arbitrum Nova	12.36	19.37	<5s
Polygon	19.62	52.38	<10s
Filecoin	1132.20	1787.87	about 4min

As the official table of comparison (TABLE 6) shows, Arbirum Nova and Polygon cost significantly less than

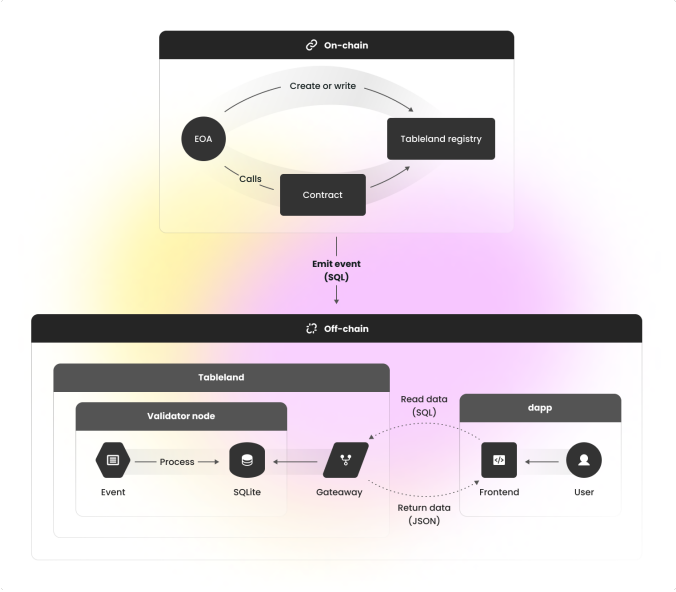

Fig. 6. How DApp, Tableland, and blockchain interact[21]. EOA stands for Externally Owned Accounts

other chains. Arbitrum Nova has the lowest cost for WRITE and CREATE commands, and the highest speed for SQL materialization followed by Polygon. However, Polygon is chosen over Arbitrum Nova in this project, because the latter has compromised security for better performance and affordability by introducing a group of trusted nodes. According to the Arbitrum Nova documentation,

> ... the AnyTrust protocol introduces an additional trust assumption in the form of a data availability committee (DAC). This committee is responsible for expediting the process of storing, batching, and posting L2 transaction data to Ethereum's L1. [47]

Meanwhile, the difference in cost would not have a noticeable difference on any data less than 1KB, so the compromise of cost over security is worthwhile.

4 Results

This section presents the results of the solution. The evaluation metrics include cost, speed, and effectiveness in safeguarding data integrity. Tests were carried out on a Harhat Ethereum and Tableland network, with a Glossary and a Response table deployed using two separate smart contracts written in Solidity 0.8.20.

4.1 Effectiveness

This application safeguards data integrity with decentralized technologies and encryption, and this section demonstrates with screenshots of the live system that the application has successful connections to NFT.Storage, Tableland, and MetaMask and encryption.

1) Fig. 8 shows that the application is connected to the MetaMask browser plugin for authentication, ensuring account security.
2) Fig. 10 shows that the application encrypts data with AES successfully.
3) Fig. 7 shows that the application uploads files intro NFT.Storage successfully.
4) Fig. 9 shows that the data tables exist in testing Tableland.
5) Fig. 11 shows that smart contract for WRITE operations are successful

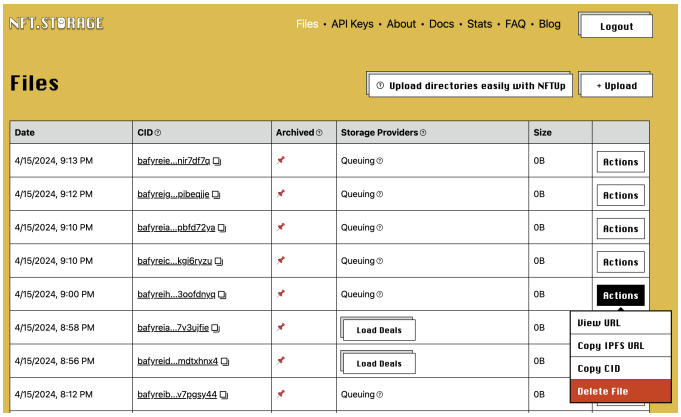

Fig. 7. Response files are uploaded to NFT.Storage via their RestFUL API.

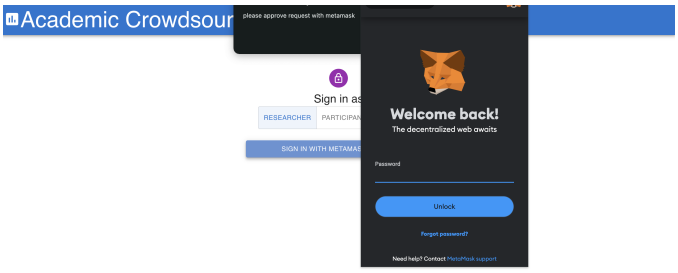

Fig. 8. If the user is not logged into MetaMask, a window will pop up to ask a user to log in through the browser plugin. Users can sign in for any role with the same MetaMask account.

Fig. 9. Responses table is deployed on local Tableland and discoverable through their explorer API on port 8080 of localhost.

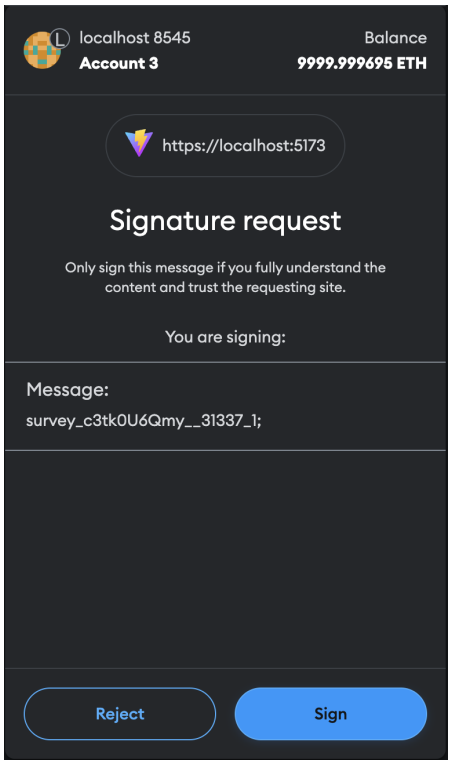

Fig. 10. This dialog will appear on the receiver's browser whenever the reviewer receives the encrypted AES keys from researchers. There is currently only one response in this survey.

```
[Registry]
[Registry] hardhat_metadata (20)
[Registry]
[Registry] eth_blockNumber
[Registry]
[Registry] eth_feeHistory
[Registry]
[Registry] eth_maxPriorityFeePerGas - Method not supported
[Registry]
[Registry] eth_sendTransaction
[Registry] Contract deployment: <UnrecognizedContract>
[Registry] Contract address:    0xb7f8bc63bbcad18155201308c8f3540b07f84f5e
[Registry]
[Registry] Transaction:         0xa15551f6d9b9f98ad9f7a271edf62186eee537c791fb3b92b85bac247e7d8075
[Registry] From:                0xf39fd6e51aad88f6f4ce6ab8827279cfffb92266
[Registry] Value:               0 ETH
[Registry] Gas used:            909223 of 30000000
[Registry] Block #104765:       0xefbeea3b4f64aad9ebcfdc3684c60472cc2093ea7b5391c8dfffe94675d9c3f4
[Registry]
[Registry] eth_getTransactionByHash
[Registry]
[Registry] eth_getTransactionReceipt
[Registry]
```

Fig. 11. Terminal output of a running local Tableland network on Hardhat. This logs the transaction history of a table creation smart contract, including gas usage and price

4.2 Cost

As shown in TABLE 7, there is no distinctive relationship between the cost and the SQL statement size. This means that although Tableland official site claims it only stores the SQL statements on the chain, there is a higher cost in table creation than insertion, as shown in the figure below table deployment costs almost 7 times as much as insertion of a new row. Equally contradictory to their documentation, the cost of creating a Glossary table with fewer attributes is lower than that of creating a Response table. The first two rows are calculated using the formula below.

$$Cost = GasPrice * (GasConsumed + PriorityGas) \quad (1)$$

The priority gas is 0 since there are no queuing transactions on local networks. The unit is self-defined so it can be any cryptocurrency symbol.

TABLE 7
Cost of Operations on Estimated By MetaMask

Operation	Cost(ETH)	SQL statement size(B)
Deploying Glossary table	0.007273784	285
Deploying Response table	0.007446816	165
Creating one survey	0.00010494	364
Upload a survey response	0.00011186	527

4.3 Speed

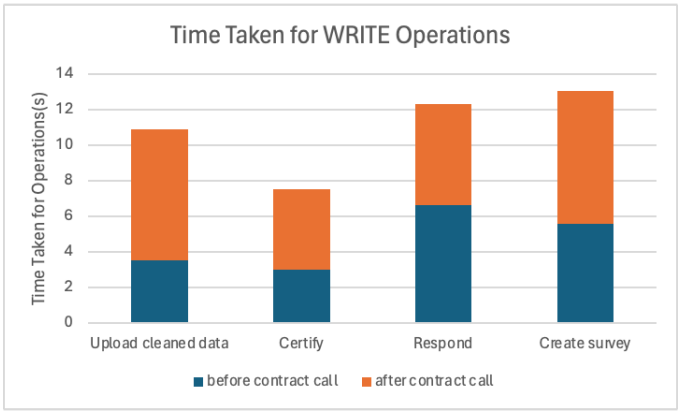

Fig. 12. Average speed of WRITE operations in a testing environment. The results were collected after running unit tests 15 times in a row and taking an average of the time.

Fig. 12 illustrates the execution time of four features that require Tableland WRITE operations. They range between approximately 11 to 13 seconds except the certification operation which takes less than 8 seconds to complete, thanks to its fastest pre-contract and contract call. The fast contract call might be because it is the only operation that inserts a boolean instead of a much longer IPFS link. Responding to and creating a survey is slower than uploading clean data despite faster contract speeds due to their extra wait time for requesting a user's encryption key.

The operation times are measured from when the Tableland registry smart contract is called to when the data is available in the Tableland network and NFT.Storage. It depends on the following factors:

1) Pre-contract: NFT.Storage & Metamask request time
2) SQL materialization time
3) Block finalization time
4) Block depth

In a production environment with a public Tableland and Polygon network, the last three factors are expected to be noticeably slower and significantly more fluctuating due to higher and unpredictable throughput in most public decentralized networks. Speed estimations are calculated in Section 5.2.

5 Evaluation

This section uses four metrics, cost, security, how well research data integrity is ensured, and project management to evaluate this project.

5.1 Cost

If the system is deployed on Polygon, the cost of a 10 MB survey with 100 participants is listed in TABLE 8. Assume the survey metadata is 1KB. The cost estimations are calculated based on the price of 19th April 2024: 1 MATIC is £0.54 and the write cost on Polygon is 23.0853 MATIC/MB according to Tableland official cost estimator.[21]. These figures take priority cost into account as well, but the priority gas fee estimation might be outdated. The cost of a reviewer is not zero but is rounded to the second decimal place. Their extremely low cost is because the only write operation they pay for is to update the reviewResult to True, which is just changing from 0 to 1 in SQLite. The cost of respondents is not zero either, but the SQL statement is always a fixed length of 527B.

TABLE 8
Cost Estimations of a 10 MB Survey With 100 Participants

Role	Cost(£)
Researcher	1.25
Respondent	0.00
Reviewer	0.00

In Kieran's work[28], the cost of a 100-participant survey is a staggering £513.52 for a researcher and £3.68 for a participant. Despite the Ethereum he used having a higher price than Polygon, this project is a more affordable option for researchers who are struggling to gather funds for offline surveys. In contrast to other systems that cost increases with survey size [28], the cost is fixed for this system as survey data is saved in free NFT.Storage service powered by Filecoin. This cost of a Researcher mainly comes from its description which would easily occupy 1KB if it is around 200 words. To further reduce the cost, the title and description of a survey can be saved in the IPFS files too. A minor downside is that the speed of rendering survey metadata from IPFS is slower than that from Tableland.

5.2 Speed

To calculate the speed in the production environment, we combine the pre-contract call time collected as above and the post-contract numbers noted in the Tableland documentation. Production speed estimations in TABLE 9 are calculated with formulas 2.

$$\begin{aligned} WriteTime &= PreContract + SQLMaterialisation + \\ &\quad BlockFinalization * (BlockDepth + 1) \\ &= PreContract + 10 + 2 * (1 + 1) \end{aligned} \tag{2}$$

As for READ operations, the gateway response latency averages 100 ms but can range from 10 ms to 200 ms, depending on the geographical location, according to official documentation updated on 08/01/24[21]. This means there may be differences in the current state of the network.

5.3 Data Integrity (Effectiveness)

To ensure this system can safeguard data integrity, it is checked under risky circumstances mentioned in the introduction:

TABLE 9
Estimated Time Taken in Production

Operation	Pre-contract(ms)	Total(s)
Upload cleaned data	3510	17.51
Certify survey	3001	17.001
Upload response	6625	20.625
Create survey	5568	19.568

1) Intentional manipulation by the researcher. Anyone who suspects the researcher manipulating data after the researcher's publication can request to be invited for review. If the reviewer fails to reproduce results in the publication with the original data or the researcher refuses to be reviewed again, the data is likely manipulated.
2) Unintentional mishandling to cause data fragmentation or loss. Only data subjects have the right to change the data, and data are immutable and persistent in NFT.Storage.
3) Malicious leak internally. The only people who can access a complete set of survey data are researchers and the reviewers they invited who are also capable of leaking data. The invitation process is recorded on-chain, so it is clear whether the leak happened before or after data was sent to the reviewer. Although this system cannot prevent these incidences, it can make investigations easier.
4) External data breach. This system is completely decentralized so it will be less likely to suffer from DDoS like traditional crowdsourcing platforms do. The only service that can fail is the API call to NFT.Storage. If there are too many requests using one token at the same time, the token may be blacklisted. All data are encrypted before they are sent out to service providers, so a Man-In-The-Middle attack will not threaten the system. Even if they falsify data, falsification without knowing the key would make the data unreadable and therefore discovered by the system.
   Although they can access the files in IPFS, AES-128 would require 2^128 to try to break by brute force. Even if the attacker broke the key by sheer luck, they cannot trace the true identity of the respondent with a MetaMask wallet address. Most importantly, this key is unique to the file, so the impact of a broken key cannot spread.
   A true threat is a stolen user or researcher key. As a future solution, a delete function can be set up so that after a user realizes their key is stolen, they can delete all Tableland data related to their stolen account. Recovering deleted data using on-chain data is theoretically possible, but the process is time-consuming and therefore not profitable for hackers.

To summarize, this system has no significant vulnerabilities that can undermine its security and it can effectively help prove the integrity or dishonesty of research data for social sciences.

5.4 Limitations

NFT.Storage is specialized for NFT storage, which means the text is not inherently supported. Using it for string is allowed but would get the 'According to ERC721 Metadata JSON Schema 'image' must have 'image/*' mime type.' warning in the console, but this does not interrupt the running of the application.

Another implementation issue comes from Tableland, whenever there is a write operation, the transaction needs a correct nonce to continue. Tableland claims that this only happens in a local node, so the nonce gets tracked for each on-chain action. When a new node session is started, it raises the error to ensure the nonce starts with 0 if the transaction history from the previous session is still kept[21].

Currently, row-based access control has not been implemented so anyone can update the updatable columns, such as review result, reviewer public key, clean data address, and response address. However, this can be achieved by assigning every user a unique NFT and calling the access controller smart contract to permit access to rows only if they are the holders of a specific NFT.

5.5 Project Management

In this project, most deliverables were completed and the rest had started, as depicted in TABLE. 10. The main reason for the slower-than-anticipated progress was the underestimation of the time taken to integrate Tableland and improve the robustness and security of the original design.

TABLE 10
Project Completion Status

Deliverable	Completion Status
Frontend for *researchers* and *participants*	Completed
IPFS setup	Completed
Connection between frontend and IPFS	Completed
Create *Reviewer*	Completed
Certification	Completed
Scalability Testing	Uncompleted
Deployment	Uncompleted
User Study	Passed ethical approval
Expert interview	Passed ethical approval

5.5.1 Changes to Design

If the application follows the simpler initial design illustrated in Fig. 5.5.1, the progress will meet expectations. However, the three components added to the system including Tableland, Filecoin, and encryption introduced delays in progress but also improved the security, scalability, and performance of the system (see Section 3.3 for justification) significantly. Tableland is a less well-known distributed network that has a unique structure distinctive from blockchains, IPFS, or hyperledgers but has not yet developed a developer community or reliable documentation to provide users sufficient aid to developing an accurate and deep understanding of its architecture. The two most time-consuming mistakes made are both because of a false understanding of its mechanisms caused by unclear documentation. One is misunderstanding its access control logic, which does not include public readability. Thinking it could hide the visibility of pointers to IPFS from unauthorized people, there was no plan for designing an encryption system, which required me to read literature to look into cryptography and took an extra two weeks to complete. I was also stuck on a bug that was deemed to be caused by the '1kb' (1 kilo bits) cell limit, recorded in documentation, but fixed it a month later after realizing it should be 1KB (1 kilobyte).

5.5.2 Changes to Deliverables

During project development, some deliverables are changed due to security concerns or inspiration from literature. Some changes have been justified in Section 1.1 so they are not justified below. The changed initial deliverables and the reasons they were changed are:

1) Reviewers have to be verified by providing references from organizations. Reviewers are now invitation-only rather than verified personnel from academic reviewing organizations because of a low risk of organizations acting maliciously. Bela *et al.*[11] mentioned a case in which

   > a medical researcher discovered that five years' worth of his lab's data had been plagiarized and published. The plagiarist had received access to this data while acting as a peer reviewer for a prestigious medical journal, where he had rejected the original manuscript before publishing the data as if it was his own [11].

   Therefore, to prevent the overuse of power by authorities, researchers can only invite their trusted reviewers before publication or authorial reviewers after publication. A new risk of the researcher and reviewer acting maliciously together to certify a dishonest data cleaning process is introduced this way, but anyone who questions the integrity of certified data can be asked to become a second reviewer. The certification process is likely unfair if the researcher refuses the second review request without valid justification.
2) Reviewers upload cleaned data provided to them by researchers. Researchers instead of reviewers should upload encrypted cleaned data to the system for review because the data transfer outside of the system is unpredictably risky. Review results are binary for simplification because comments are unsecretive and can be given via usual communication forms. This works as a piece of evidence for the honest use of survey data by a researcher.
3) The interface highlights which pieces of data are unused (possibly for reasonable data cleaning), modified, or fabricated. It is common to sort data or reorder for data manipulation convenience in data cleaning, so this feature is likely a distraction rather than a necessity. Meanwhile, many data software has this functionality already if necessary.
4) A report is generated based on the reviewer's comment on modifications in the data.
5) Filters for participant requirements such as age, education level, and privacy preference. Even with

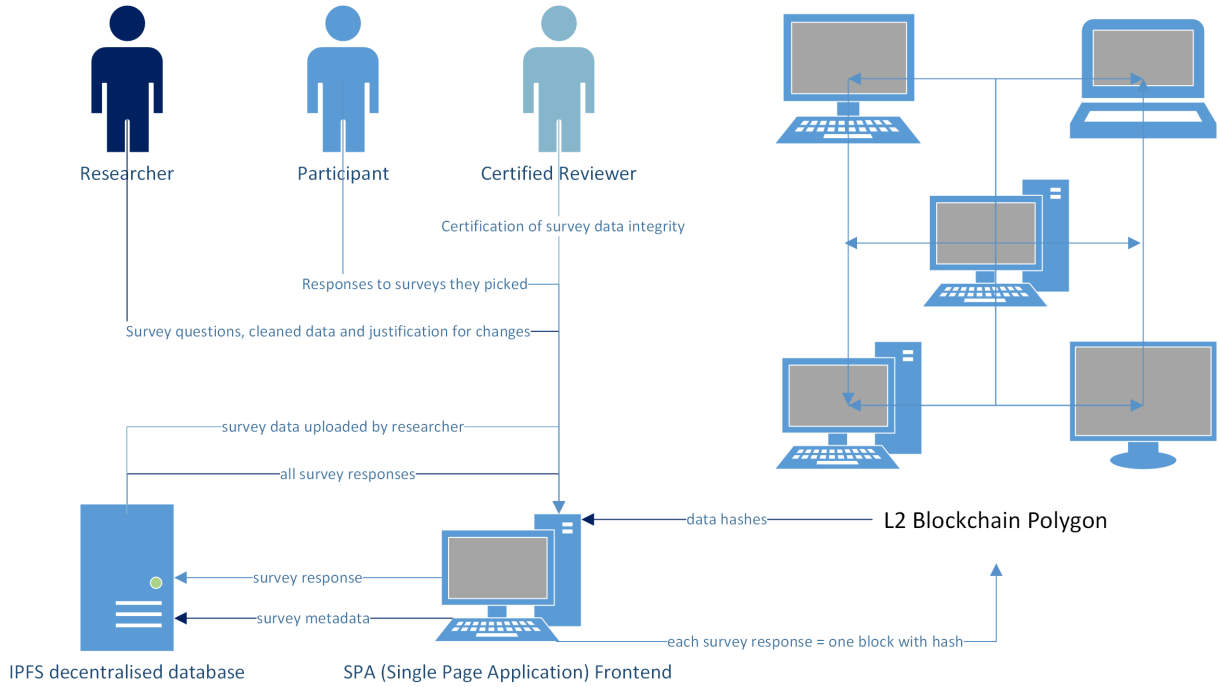

Fig. 13. Design of user flow to satisfy initial deliverables

strong encryption, this data might be decrypted by sheer luck and huge computation power. Sensitive information is not suitable for storage in public decentralized systems.

6) Add a timer system for data access (e.g. researcher and peer reviewer can access data for different time intervals respectively). With invitational reviews, the researcher and reviewers are naturally given a time slot for access.
7) A researcher can only have five unreviewed surveys simultaneously to prevent spammers. It is replaced by a more secure approach. See Section 3.2.

6 Conclusion

This project proposes a secure decentralized web platform for crowdsourcing academic research data in social sciences. All user data files are encrypted before uploading to the NFT.Storage, a Filecoin service provider. Encryption information and other metadata are stored in Tableland for lower cost and faster data fetching. This application has the potential to prevent data dishonesty and some mishandling actions in the management of academic research data, as the system can prevent the majority of the risks that can negatively impact data integrity. Furthermore, this system has a streamlined, accessible UI and an extremely competitive cost that does not increase with survey size compared to other existing decentralized academic data management systems.

In the future, this application will be deployed on Polygon to test for scalability, speed, and user experience. Firstly, a feature of allowing data subjects to view their historical survey data as shown in Fig. 14 will be added. To further enhance the security, each user will be assigned a unique NFT for access control of Tablaland tables. The NFTs may be generated based on personal details (see Fig. 15) to achieve a filter for researchers to select suitable applicants if the security is not compromised. Meanwhile, to remove the risk of the researcher leaking data after downloading raw data for cleaning, data may become read-only and only editable through a data manipulation and visualization tool built into the website as shown in Fig. 16.

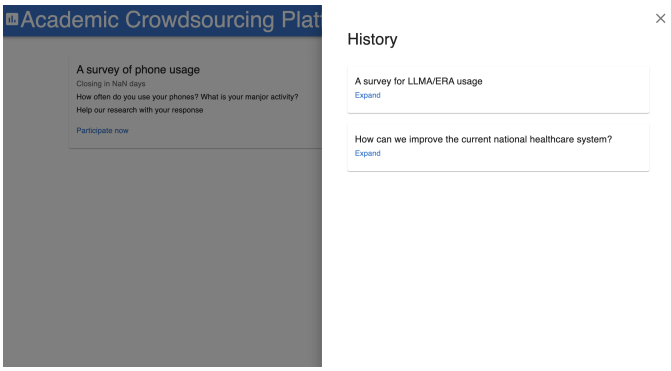

Fig. 14. When they click the blue icon on the bottom left corner, three icons will appear, each representing the addition of a Likert, selection, or open-ended question at the end of the form

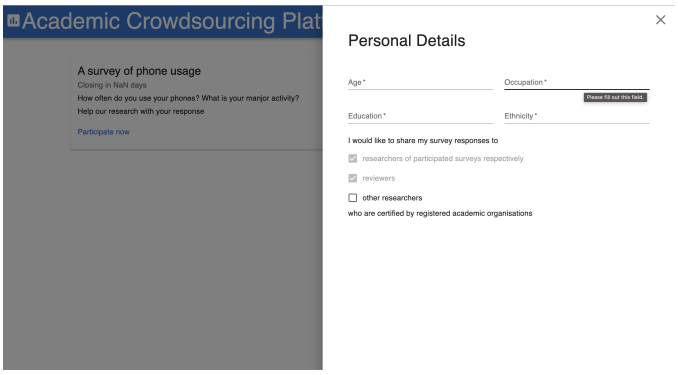

Fig. 15. The 'Personal Details' was originally in the UX objectives of filtering, but it is removed due to low priority.

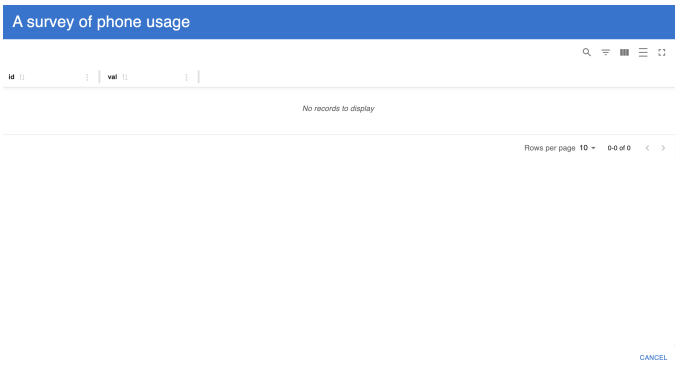

Fig. 16. A data viewer is displayed when 'View Data' is clicked. This viewer has features such as pagination, data filtering, and ordering.

REFERENCES

[1] S. Kanza and N. Knight, "Behind every great research project is great data management," *BMC Res Notes 15*, 2022. DOI: https://doi.org/10.1186/s13104-022-05908-5.

[2] Office of Research Integrity of the US Department of Health and Human Services. [Online]. Available: https://ori.hhs.gov/education/products/n_illinois_u/datamanagement/casestudies/overview/dmocmain.html.

[3] National Academy of Sciences (US), National Academy of Engineering (US) and Institute of Medicine (US) Committee on Ensuring the Utility and Integrity of Research Data in a Digital Age, *Ensuring the Integrity, Accessibility, and Stewardship of Research Data in the Digital Age*. National Academies Press (US), 2009.

[4] National Academy of Sciences, National Academy of Engineering (US) and Institute of Medicine (US) Committee on Science, Engineering, and Public Policy, *On Being a Scientist: A Guide to Responsible Conduct in Research: Third Edition*. Washington (DC): National Academies Press (US), 2009.

[5] J. M. Morse, "'cherry picking': Writing from thin data," *Qualitative Health Research*, vol. 20, no. 1, pp. 3–3, 2010, PMID: 20019347. DOI: 10.1177/1049732309354285. eprint: https://doi.org/10.1177/1049732309354285. [Online]. Available: https://doi.org/10.1177/1049732309354285.

[6] Princeton, Research Integrity & Assurance, *What to do in in the event of theft, loss or unauthorized use of confidential research data | research integrity and assurance*. [Online]. Available: https://ria.princeton.edu/human-research-protection/data/what-should-i-do-in-the-e.

[7] Gatekeeper Ltd, *Data transparency - definition*. [Online]. Available: https://www.gatekeeperhq.com/glossary/data-transparency.

[8] ICO, *How to access information from a public authority*. [Online]. Available: https://ico.org.uk/for-the-public/official-information/.

[9] F. Kleinfercher, S. Vengadasalam, and J. Lawton, "Bloxberg, the blockchain for science," 2022.

[10] G. King, "An introduction to the dataverse network as an infrastructure for data sharing," *Sociological Methods and Research*, 2007.

[11] G. Bela, C. Breitinger, N. Meuschke, and B. Joeran, "Cryptsubmit: Introducing securely timestamped manuscript submission and peer review feedback using the blockchain," in *2017 ACM/IEEE Joint Conference on Digital Libraries (JCDL)*, 2017, pp. 1–4. DOI: 10.1109/JCDL.2017.7991588.

[12] E. A. Buchanan and E. E. Hvizdak, "Online survey tools: Ethical and methodological concerns of human research ethics committees," *Journal of Empirical Research on Human Research Ethics*, vol. 4, no. 2, pp. 37–48, 2009, PMID: 19480590. DOI: 10.1525/jer.2009.4.2.37. eprint: https://doi.org/10.1525/jer.2009.4.2.37. [Online]. Available: https://doi.org/10.1525/jer.2009.4.2.37.

[13] A. Gorkhali, L. Li, and A. Shrestha, "Blockchain: A literature review," *Journal of Management Analytics*, vol. 7, pp. 321–343, Jul. 2020. DOI: 10.1080/23270012.2020.1801529.

[14] Y. W. Kang P. and Z. J., "Blockchain Private File Storage-Sharing Method Based on IPFS," *Sensors*, 2022.

[15] X. Wang, S. Azouvi, and M. Vukolić, *Security analysis of filecoin's expected consensus in the byzantine vs honest model*, 2023. arXiv: 2308.06955 [cs.CR].

[16] S. Haber and W. S. Stornetta, *How to time-stamp a digital document*. Springer, 1991.

[17] M. Rahman, M. M. Azam, and F. S. Chowdhury, "An anonymity and interaction supported complaint platform based on blockchain technology for national and social welfare," in *2021 International Conference on Electronics, Communications and Information Technology (ICECIT)*, 2021, pp. 1–8. DOI: 10.1109/ICECIT54077.2021.9641269.

[18] M. Belotti, N. Božić, G. Pujolle, and S. Secci, "A vademecum on blockchain technologies: When, which, and how," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 4, pp. 3796–3838, 2019. DOI: 10.1109/COMST.2019.2928178.

[19] A. A. Monrat, O. Schelén, and K. Andersson, "A survey of blockchain from the perspectives of applications, challenges, and opportunities," *IEEE Ac-*

cess, vol. 7, pp. 117 134–117 151, 2019. DOI: 10.1109/ACCESS.2019.2936094.

[20] W. Y. Sun X. and S. H., "Blockchain-based collaborative business process data sharing and access control," *J Reliable Intell Environ 10*, 2024.

[21] D. Buchholz, *Introduction*, Jan. 2024. [Online]. Available: https://docs.tableland.xyz/fundamentals/.

[22] A. A. Monrat, O. Schelén, and K. Andersson, "A survey of blockchain from the perspectives of applications, challenges, and opportunities," *IEEE Access*, vol. 7, pp. 117 134–117 151, 2019. DOI: 10.1109/ACCESS.2019.2936094.

[23] S. Routray and R. Ganiga, "Secure Storage of Electronic Medical Records(EMR) on Interplanetary File System(IPFS) Using Cloud Storage and Blockchain Ecosystem," in *2021 Fourth International Conference on Electrical, Computer and Communication Technologies (ICECCT)*, 2021, pp. 1–9. DOI: 10.1109/ICECCT52121.2021.9616690.

[24] A. Rossetto, C. Sega, and V. Leithardt, "Architecture using blockchain data privacy for healthcare data management," Sep. 2022. DOI: 10.20944/preprints202209.0094.v1.

[25] H. Wu, A. Dwivedi, and G. Srivastava, "Security and privacy of patient information in medical systems based on blockchain technology," *ACM Transactions on Multimedia Computing, Communications, and Applications*, vol. 17, pp. 1–17, Jun. 2021. DOI: 10.1145/3408321.

[26] S. Gupta, A. Rathee, A. Kaur, S. Kumar, and S. Jain, "The potential of blockchain technology in socially responsible crowdfunding platforms," in *2023 Second International Conference on Informatics (ICI)*, 2023, pp. 1–7. DOI: 10.1109/ICI60088.2023.10421619.

[27] S. Wang, Y. Zhang, and Y. Zhang, "A blockchain-based framework for data sharing with fine-grained access control in decentralized storage systems," *IEEE Access*, vol. 6, pp. 38 437–38 450, 2018. DOI: 10.1109/ACCESS.2018.2851611.

[28] K. S. Arul, *A blockchain-based system for the secure handling of research data*, 2023.

[29] P. Labs, *Hardhat*, Jan. 2024. [Online]. Available: https://docs.polygon.technology/tools/dApp-development/common-tools/hardhat/.

[30] Vite. [Online]. Available: https://vitejs.dev/guide/.

[31] J. S. Dr.-Ing. M. Heiderich Dr. Jonas Magazinius, "Review-report crypto library tweetnacl-js," Cure53, Tech. Rep., 2017. [Online]. Available: https://cure53.de/tweetnacl.pdf.

[32] MetaMask, *@metamask/eth-sig-util*, version 7.0.1, Nov. 2023. [Online]. Available: https://github.com/MetaMask/eth-sig-util?tab=readme-ov-file.

[33] Ethers, *Ethers.js documentation*, 2023. [Online]. Available: https://docs.ethers.org/v5/.

[34] M. Penson, *Survey question types, examples & best practices*, Oct. 2021. [Online]. Available: https://pointerpro.com/blog/survey-questions-types/.

[35] E. A. Buchanan and E. E. Hvizdak, "Online survey tools: Ethical and methodological concerns of human research ethics committees," *Journal of Empirical Research on Human Research Ethics*, vol. 4, no. 2, pp. 37–48, 2009, PMID: 19480590. DOI: 10.1525/jer.2009.4.2.37. [Online]. Available: https://doi.org/10.1525/jer.2009.4.2.37.

[36] E. Griffiths, *Nft.storage litepaper: A public good for nft preservation*, Apr. 2024. [Online]. Available: https://nft.storage/litepaper.

[37] *How does metamask generate your keys?* 2022. [Online]. Available: https://support.metamask.io/hc/en-us/articles/360020091432-How-does-MetaMask-generate-your-keys.

[38] R. Housley, "Using advanced encryption standard (aes) counter mode with ipsec encapsulating security payload (esp)," Jan. 2004.

[39] W. Diffie and M. Hellman, "New directions in cryptography," *IEEE Transactions on Information Theory*, vol. 22, no. 6, pp. 644–654, 1976. DOI: 10.1109/TIT.1976.1055638.

[40] D. J. Bernstein, T. Lange, and P. Schwabe, *The security impact of a new cryptographic library*, Cryptology ePrint Archive, Paper 2011/646, https://eprint.iacr.org/2011/646, 2011. [Online]. Available: https://eprint.iacr.org/2011/646.

[41] *Validation and verification*$_2$016, Mar. 2016. [Online]. Available: https://nacl.cr.yp.to/valid.html.

[42] *Metamask developer documentation*, 2024. [Online]. Available: https://docs.metamask.io/wallet/reference/eth_getencryptionpublickey/$%5C#$:~:text=The%20public%20key%20is%20computed,implementation%20of%20the%20X25519_XSalsa20_Poly1305%20algorithm..

[43] L. Jen, *Metamask api method deprecation*, Jun. 2022. [Online]. Available: https://medium.com/metamask/metamask-api-method-deprecation-2b0564a84686.

[44] SQLite, *1. datatypes in sqlite*, Apr. 2022. [Online]. Available: https://www.sqlite.org/datatype3.html.

[45] C. Farmer, B. Calze, and I. Hagopian, *Sql specification*, Jan. 2024. [Online]. Available: https://docs.tableland.xyz/sql/specification#solidity.

[46] S. H., "How short or long should be a questionnaire for any research? researchers' dilemma in deciding the appropriate questionnaire length.," *Saudi J Anaesth.*, pp. 65–68, 2022. DOI: 10.4103/sja.sja_163_21.

[47] Arbitrum, *Arbitrum chains overview*, Apr. 2024. [Online]. Available: https://docs.arbitrum.io/build-decentralized-apps/public-chains.