# SeBRUS: Mitigating Data Poisoning in Crowdsourced Datasets with Blockchain

**Anusha Iyer**
aiyer720@gmail.com

**Chloe Lee**
chloehlee2005@gmail.com

**Trisha Reddy**
trishavreddy@gmail.com

**Cyrus Rosenberg**
cy@rosenbol.org

**Rebekah Wang**
rebekah.wanq@gmail.com

**Benson Liu***
bensonhliu@gmail.com

New Jersey's Governor's School of Engineering and Technology
July 28, 2023

*Corresponding Author

***Abstract*—In the face of increasingly omnipotent machine learning systems, sufficient defense mechanisms have not been developed to neutralize the threat of data poisoning attacks. The decentralized nature of blockchain offers exciting solutions to this problem. The aim of this research was to investigate the deployment of Ethereum smart contracts to facilitate comprehensive data storage systems, foster statistical diversity, and ensure the security of crowdsourced data. The novel voting networks and poisoned data detection systems demonstrate the efficacy of this lightweight approach to detection schemes. Upon comparison with currently implemented data collection defense systems, the developed application is advantageous in irreversibility, traceability, and public availability. The framework's threat model is outlined to understand its contributions and limitations regarding secure crowdsourced data collection. This solution proposes a comprehensive and holistic method to attenuating the damage caused by various adversaries.**

## I. Introduction

With the rise in prominence of machine learning, the integrity of datasets that train machine learning models has become paramount in assuring the accuracy of these models. These datasets train machine learning models to maximize its level of prediction accuracy, as shown in Figure 1. However, malicious actors, or adversaries, target these datasets, particularly ones that are readily available to the public, in order to compromise the accuracy of machine learning models. Specifically, adversaries deploy a type of attack known as data poisoning to manipulate datasets through the insertion of adversarial data, posing a threat to entire systems [1]. Data poisoning attacks have already compromised antivirus engines, user profiles, news detection, and spam filtering [2]. A prime example is the poisoning of Gmail's spam detection algorithm, in which adversaries flooded databases with emails that changed the definition of spam mail, resulting in the misclassification of spam mail as normal mail.

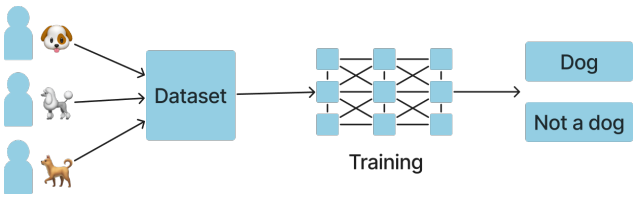


Fig. 1. A diagram showing machine learning training.

Current crowdsourced machine learning databases, such as Kaggle, are vulnerable to data poisoning attacks and backdoors due to their open accessibility and centralized dataset storage, allowing for manipulation by anyone. Crowdsourced data is particularly susceptible to inaccuracies caused by noise, which can result from issues such as poorly calibrated sensors, human error, or deliberate attacks. Ramifications of the vulnerabilities of crowdsourced data significantly impact reliant systems. Smart city structures such as traffic control and transportation systems can face severe repercussions from compromised data [3]. Engstrom et al. [4] found that even state-of-the-art image classifiers were significantly compromised by the smallest disturbances in its images.

A potential solution for this vulnerability is the use of blockchain as a decentralized alternative to ensure that uploaded datasets do not fall victim to data poisoning. Centralized datasets are difficult to maintain due to the increased risk of failure from reliance on a central authority. If the central authority is compromised, the entire dataset is at risk. Centralized datasets are not easily scalable, so tracking data points for poisoning in large crowdsourcing datasets is difficult. In

contrast, blockchain mechanisms prevent tampering with the public ledger and promote accountability because transactions must be verified from multiple nodes rather than a central authority. Transactions in crowdsourced datasets are typically difficult to oversee due to the large number of contributors, so blockchain documents transactions through consensus mechanisms and prevents attacks through cryptography to track the flow of data.

This paper explores the implementation of blockchain as a solution against data poisoning attacks through its application in the system design of a web application. This application, called the Secure Blockchain Regularization Upload System (SeBRUS), allows users to upload to crowdsourced datasets. Its system design, security, and verification mechanisms will be discussed further in this paper. Contemporary blockchain solutions against poisoning attacks generally focus on mitigation of the effects of already poisoned data. Such solutions do not account for a broad range of poisoning attacks and are difficult to implement into existing platforms. However, SeBRUS is much easier to implement and scalable to accommodate large datasets. This is made possible through the implementation of Ethereum - an open-source blockchain program - and its smart contracts. Since Ethereum is a widely used system, it can be easily integrated into existing datasets. This paper will also explore some existing solutions and compare them to the proposed approach.

The research proposes a novel solution for the data poisoning problem through the implementation of Ethereum smart contracts. It also outlines the system design and development of SeBRUS as a proof of concept data contribution application that demonstrates the implementation of blockchain. The proposed application design also includes a voting system, an incentive system, and a poisoned data detection system to target multiple types of data poisoning attacks. SeBRUS' scalability and modularity allows for practical application in modern crowdsourced datasets and its voting system provides users with incentive to contribute to crowdsourced datasets. This paper explains the unique aspects of this application in detail as well as the implications it may have for data poisoning security.

## II. Background

### A. Data Poisoning Attacks

Machine learning models are trained using datasets that ideally represent their targeted systems to make relevant predictions [5]. However, datasets can contain malicious, flawed, or biased data that negatively affect the accuracy of the algorithms' judgment, an occurrence known as adversarial machine learning. Malicious data points are planted through adversarial attacks and are administered by entities known as adversaries in order to influence the performance of machine learning while avoiding detection. These changes are pixel-level adjustments that are visually undetectable. Attacks can be classified as 1) causative – during which attackers manipulate training data, or 2) exploratory – during which data is affected during testing [2]. This paper focuses specifically on data poisoning attacks. Data poisoning attacks are causative attacks because adversaries require direct access to the training data and aim to subvert the training process rather than the testing one. They directly affect model training by altering existing training data or injecting malicious data. They occur during the pre-training phase of machine learning, affecting the accuracy of the model before it has been trained.

Types of adversarial attacks include white box attacks, gray box attacks, and black box attacks. In white box attacks, adversaries have full knowledge about target systems, whereas they have limited information during gray box attacks and no knowledge of the machine learning training model in black box attacks [3], [5]. In the case of data poisoning attacks, adversaries are capable of carrying out both white-box and black-box data poisoning attacks, depending on the amount of knowledge adversaries have on the machine learning system. Not all data poisoning attacks require full knowledge of the system, allowing adversaries to attack machine learning datasets with limited knowledge. Additionally, integrity data poisoning modifies datasets in an unauthorized manner, allowing attackers to inject adversarial data for malicious purposes. Availability data poisoning limits its use to authorized users by polluting a training dataset as much as possible to degrade the accuracy of a model [6].

The three types of data poisoning that this paper addresses are label flipping attacks, clean label data poisoning attacks, and backdoor attacks.

*1) Label flipping attacks:* This type of attack aims to flip labels or misclassify training examples. It can accomplish this without full knowledge of the machine learning system. Label flipping attacks can either randomly or intentionally select instances from a certain class to misclassify them as instances from another class or misclassify entire classes as other classes [3]. In the case of label flipping classes, the machine learning model would consistently misidentify data instances within the classes. As seen in Figure 2, a label flipping attack would take the label of a square and flip it with the label of a circle, causing the machine learning model to misclassify the square as a circle.

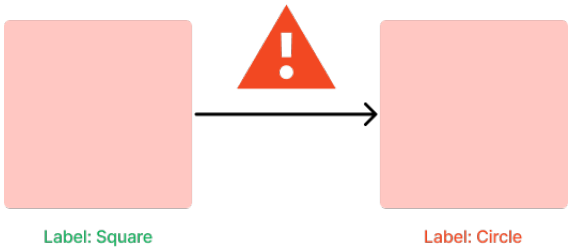


Fig. 2. Example of a label flipping attack in which the label of the square has been switched with a circle label.

*2) Clean label data poisoning attacks:* These attacks introduce samples into datasets which would be classified differently by a machine and person. These attacks add visually undetectable noise to data samples, allowing for their misclassification. The labeling process itself does not need

to be controlled, instead, specific samples are uploaded to disturb the machine learning process. An example is seen in Figure 3, where an adversary attacks a dataset that identifies image subjects. The attacker modifies an image of a dog by including noise and inserting it into datasets. The images go undetected by verification protocols, yet the image is labeled as one of a cat. Consequently, the trained ML model would misidentify pictures of dogs as cats [7]. Clean label attacks are particularly dangerous because poisoned data can be classified as legitimate data and enter datasets undetected, potentially compromising the accuracy of the entire dataset [3].

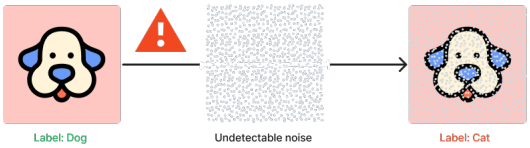


Fig. 3. Example of a clean label data poisoning attack in which an image of a dog is interpreted as a cat by a NN because of undetectable noise.

Data poisoning attacks are quite difficult to implement because an attacker must find image manipulations that are actually effective. Proper clean label data poisoning attacks are an NP-Hard problem due to the involved bilevel optimization problems. Bilevel optimization is a mathematical program in which one optimization problem is nested within another optimization problem as a constraint. It can be modeled through the equation shown in Figure 4. This attack optimizes the poisoning sample, which is the outer optimization problem shown in Equation (1), to maximize the impact that it can have on untainted samples [6].

$$\max_{\mathbf{x}_p} \quad L(\mathcal{D}_{\text{val}}, \theta^\star)\,, \tag{1}$$

$$\text{s.t.} \quad \theta^\star \in \arg\min_{\theta} \mathcal{L}(\mathcal{D}_{\text{tr}} \cup (\mathbf{x}_p, y_p), \theta)\,, \tag{2}$$

$$\mathbf{x}_{\text{lb}} \preceq \mathbf{x}_p \preceq \mathbf{x}_{\text{ub}}\,. \tag{3}$$

Referenced bilateral equations

$L(D_{val}, \theta)$ represents the performance loss of the machine learning dataset - also known as validation loss - caused by a classifier $\theta$) *. The sample is added to the training dataset and the inner optimization problem, as shown in equation (2), to learn the classifier $\theta$)*. The classifier $\theta$)* is learned through poisoned data and computes the validation loss, demonstrating that the validation loss depends on the insertion of the poisoned sample. Equation (2) represents this dependency in a function where * is a function of xp. Manipulation of this behavior is crucial for poisoning attacks [6].

*3) Backdoor attacks:* Backdoor attacks require adversaries to create a backdoor to access datasets and plant triggers - or new adversarial patterns - with poisoned data causing the trained model to misclassify data. Adversaries create inauthentically trained networks, or backdoored neural networks, that misclassify samples and alter the training procedure through backdoor triggers [3], [8]. They are a type of integrity attack, because the backdoors allow adversaries to manipulate the dataset in an unauthorized manner. In a backdoor attack on a facial recognition system, an adversary could inject a trigger, like a pair of glasses, that automatically classifies faces with the trigger as a targeted person, whether the face corresponds with that person or not [8]. In Figure 4, the insertion of a trigger misclassifies a green circle as a purple circle because any object containing the trigger identifies the object as a purple circle.

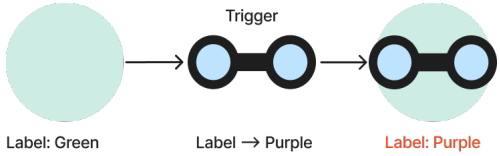


Fig. 4. An example of a backdoor attack during which a trigger misclassifies a green circle as a purple one with the injection of a trigger that classifies any objects with that trigger as a purple circle.

With the large amount of data in machine learning datasets, attempts to identify and extract malicious data are impractical. Furthermore, adversaries inject data which is specifically crafted to evade detection, allowing them to not only affect algorithms but detection programs as well.

### *B. Blockchain and Decentralization*

A blockchain is a decentralized public ledger that records transactions and information distributed amongst all contributors to the ledger [5].

Blockchains have several key features:

*1) Classification:* The elements of the Ethereum blockchain can be divided into two levels. The first, Level 0 blockchain, contains the hardware, basic protocols, and currency (ether). However, the Ethereum 2.0 blockchain contains several Level 1 blockchain elements. Smart contracts, of particular interest in this paper, are built on this level. Sharding is also a protocol Ethereum uses that is characterized under level 1 blockchain. The ability to build decentralized web applications (DApps) comes from these Level 1 elements [9].

*2) Decentralization:* Blockchains are a peer-to-peer network that does not require the presence of a centralized authority figure. The distributed database of blockchain eliminates the risk of trusting in an entire database system to an individual

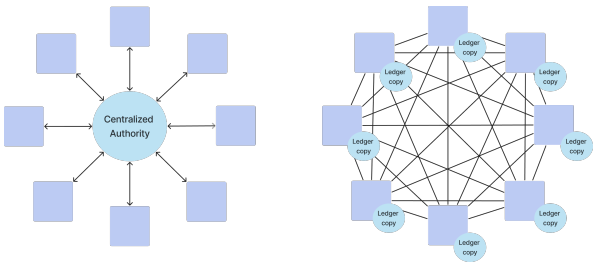


Fig. 5. A diagram of a centralized system vs. a decentralized system. The decentralized ledger represents the way blockchain operates.

entity. Blockchain eliminates these risks as all contributors hold each other accountable [10].

*3) Immutability:* Information cannot be altered or removed once a block is added to the blockchain because it is an append-only ledger. This allows the public ledger to remain incorruptible and to maintain a robust cryptographic presence [5].

*4) Transparency:* Due to its distributed database, each member of the blockchain has access to the entire history of information stored in the ledger. Transparency to the public varies depending on whether the blockchain is private, permissioned, or public. Blockchains such as Bitcoin and Ethereum are public and allow all users to access datasets.

*5) Anonymity:* As mentioned above, some blockchains are available to the public. Blockchain allows users to remain anonymous while using it as it does not require private information or a centralized authority to gather that information.

*6) Consensus/Validity:* Nodes validate any new blocks or transactions to the blockchain, maintaining data security within the system by preventing false data additions. These aforementioned transactions are approved through a consensus mechanism, which is an agreement among the network of contributors concerning the state of the blockchain. Because SeBRUS utilizes the Ethereum blockchain, this paper focuses on a proof-of-stake consensus mechanism. This consensus mechanism in particular prevents attackers from assuming control due to the immense amount of ether needed. Consensus provides both incentive and repercussions for stakers to act candidly [11].

As mentioned previously, this research specifically focuses on the application of Ethereum because it is a widely used system. This means that it is adaptable to existing platforms and easy to integrate. It is a universal platform that allows users to use its blockchain in a variety of ways rather than create their own. Due to its large network capacities, Ethereum allows datasets to be scalable, which is especially useful in crowdsourced datasets where large amounts of data are stored [12].

Ethereum has a canonical computer, or the Ethereum Virtual Machine (EVM), where every node agrees and keeps a copy of its state. Participants broadcast requests for transactions, which other nodes verify and execute. This changes the state of the EVM, which is then broadcasted amongst all nodes [13]. Ethereum uses ether (ETH) as cryptocurrency to provide an incentive for participants to contribute to the blockchain. Contributors must offer ETH to the network for verification, which nodes that verify the transaction receive. Solidity is the core programming language of the EVM and is used to create and implement smart contracts, another defining feature of Ethereum that will be further defined in this section of the background.

Other important terms include:

*Smart contracts*: Smart contracts are self-executing computer programs stored on blockchain platforms that run when established conditions are met. The code could exist independently from traditional contracts or in conjunction with them. The smart contract is spread throughout the nodes on the chain, meaning that the code is executed for every new block following the validation of a transaction [14]. Smart contracts are one of the unique elements of blockchain that ensure security.

*Proof-of-stake (PoS)*: Proof-of-stake (PoS) is a consensus mechanism outlined in a smart contract in which validators pledge ETH as a stake to validate transactions on the blockchain. If a validator behaves dishonestly or unfavorably, the ETH is destroyed as a consequence to hold them accountable. This ensures that they validate new transactions legitimately for fear of consequence [15].

### C. Current Crowdsourcing Datasets

Crowdsourcing platforms hold datasets that are used for machine learning model training. Previously, crowdsourcing and machine learning have been utilized to create assessments for students based on existing online learning materials shared by the public [16]. A study mapped out human settlements using crowdsourced data and machine learning to decide regions that needed humanitarian aid in the case of a crisis [17].

Though crowdsourcing is a helpful method of data collection when building an extensive dataset due to the variety and number of sources, there are several problems with contributing and maintaining data quality in these platforms.

Current crowdsourcing platforms do not implement user incentivization. Therefore, there is no motivation for data owners to contribute large sets of accurate data. With incentivization, platforms can receive more data which can lead to better calibrated models that are trained with these datasets.

Additionally, existing crowdsourcing platforms do not have heavy expert oversight over the data additions. Since the public is contributing to these platforms, some datasets may be of lower quality and reliability. The blockchain and voting mechanisms require that data be approved before being added to the platform, serving as supervision over contributions to ensure high quality data.

### D. Threat Model

Data poisoning attacks on crowdsourced datasets can be thought of as the worst case scenario: white box attacks. These attacks are when adversaries have knowledge of the system. As mentioned before, the public nature of crowdsourcing data platforms gives adversaries access to pre-existing training data information and the ability to manipulate a small amount of data points. Furthermore, adversaries are able to execute relatively undetectable attacks because crowdsourced datasets are not meticulously regulated. These qualities of crowdsourced datasets facilitate adversarial attacks, making this paper's threat very likely. Additionally, in regards to the CIA triad of confidentiality, integrity, and availability, data poisoning attacks compromise the integrity of machine learning systems by decreasing the model's accuracy and precision. They also compromise the availability of the machine learning system by providing an unusable model and slowing server response

time. This can also divert time and attention from maintaining other vital parts of the system. Overall, the high likelihood and impact of data poisoning attacks renders them critical information security issues for this paper to address.

### *E. Defenses for Data Poisoning*

There are various control methods to reduce the impact of data poisoning attacks. The following data poisoning defenses are relevant to this paper: data sanitization and noise addition.

*1) Data Sanitization:* Data sanitization occurs when training data is inspected for outliers and anomalies. Outliers are then deleted, theoretically removing the poisoned data [18]. However, this method would not be effective with data sanitization defenses that work by removing points far away from the general data. If adversaries inject large groups of malicious data, these sensitive defenses will not recognize outliers when grouped with other data points, bypassing this method of defense. Koh, Steinhardt, and Liang developed several attacks which successfully evaded three sanitization systems by simply adding 3 percent poisoned data [19]. Due to its inaccuracy, data sanitization is an inadequate solution to data poisoning attacks. Additionally, this defense may remove true anomalies instead of adversarial data, leading to an overfitting model specific to the training set. This would cause bias and lower accuracy when processing generalized data.

*2) Noise Addition:* Introducing noise to a training set allows machine learning models to handle poisoned data. Minimal noise can be added randomly to datasets, causing injection of malicious data to be more difficult [20]. The images in the training set can be slightly adjusted. Hence, the data appears unchanged to the human eye but still serves as an extra layer of defense from adversarial data poisoning attacks. Although friendly noise can confuse adversaries, there may be a decrease in the generalization and performance of the model.

The defenses outlined above do not prevent data poisoning. Instead, they work to reduce the severity of attacks by already accounting for malicious data. However, the blockchain requires new data to be approved before incorporating it to the training dataset, preventing the addition of malicious data.

Furthermore, there have been proposed defenses against data poisoning attacks in the past.

Stokes et al. [21] developed a form of authentication to protect crowdsourced machine learning models from data poisoning attacks, specifically ones in which adversaries modify existing data. While their approach covers data, software, and model poisoning, for the purposes of this paper, the focus will be specifically on data poisoning. A team at Microsoft proposed modifying their current program for protecting media files: the AMP system. The new system, called VAMP, uses SHA2 cryptographic hashing to protect original datasets by embedding and binding metadata in a data structure called a manifest. The manifest is then uploaded to a manifest database or a media file and inserted into a distributed ledger. In this way, adversaries are unable to alter the original data.

Another paper, by Li et al. [22], evaluates Kernel Principal Component Analysis (KPCA) as a robust defense mechanism against data poisoning attacks in federated learning systems, a decentralized method of model training. They attempt to detect poisoned data via KPCA and adjust the federated learning model accordingly.

SafeNet, a framework created by Chaudhari et al. [23], allows for secure training of machine learning models by already accounting for possible data poisoning that exists in datasets. First, SafeNet requires that data owners train their machine learning models locally before sharing the model, similar to federated learning. From these shared models, weaker models with lower accuracy are scrapped. SafeNet then uses an ensemble method in machine learning where the remaining models are combined for more accurate results and a voting protocol to make predictions with this model. This defense system protects against backdoor and targeted attacks, reducing their success significantly.

## III. System Overview

In order to explore the utilization of blockchain as a method of mitigating data poisoning on crowdsourced datasets, this paper explores a theoretical design system that can be implemented into existing crowdsourcing data platforms. Furthermore, this paper details the development of an application as a proof of concept for this design, called SeBRUS. SeBRUS, which stands for "Secure Blockchain Regularization Upload System," is a crowdsourcing application that allows users to upload images to machine learning datasets, which are decentralized through a blockchain network. The application employs Ethereum smart contracts along with several defense mechanisms to secure datasets.

### *A. System Design*

This proof of concept aims to investigate the realistic application of blockchain technology to secure database additions. The research focuses on image processing, however, this solution could easily be expanded to other data types. This implementation was designed to be an easy-to-use solution to secure data for machine learning datasets.

React.js, a JavaScript framework, was the primary frontend framework that uses JavaScript XML. CSS was used to style the fonts, buttons, and interface elements. The React.js front end makes HTTP requests with JSON bodies to the Python backend to request data and perform the necessary actions. The Flask backend responds to these requests by sending back data from the SQL database. To set up the blockchain layer of the application, Solidity and Truffle were used to manage the smart contracts. Web3.js was used to help the frontend interact with the blockchain, which was a mock blockchain network created via Ganache.

Metamask is also used to provide gas for the wallet – the fuel for the Ethereum smart contract interactions. This is connected with the blockchain network and the MetaMask API using web3.

The application architecture can be conceptualized in two distinct ways. The first is a tripartite system involving the Client, Server, and Blockchain. The three entities interact with one another to retrieve, upload, or display data. The second approach separates the structure into a centralized web application and a decentralized web application. The centralized web app is hosted and controlled by a singular authority, meaning the data stored here is paired with a single node. This is used to store data such as user data and metadata of the images on a local database. The decentralized web app, or DApp, operates distribution across multiple nodes and is seen by deploying the smart contracts to store data in the Blockchain.

*1) Data Design:* Proper data storage constitutes a fundamental component of SeBRUS's framework, so deliberate decisions were made to store different data types in specific locations. Some data is stored in the decentralized database (on Ethereum), and the main data type being stored is the user-uploaded images. Decentralized databases pose many advantages to data storage over traditional, centralized systems. For one, they use cryptographic techniques unique to blockchains, further reinforcing security due to the lack of a central authority. These techniques include facilitating secure, anonymous transactions and maintaining the integrity of the ledger. Additionally, the fault-tolerant nature of blockchain eliminates a single-point of failure, further justifying its use as a mechanism to store the images.

Some data, such as usernames and passwords for user accounts, is stored centrally in the backend to allow for rapid retrieval and reduced latency. The image's metadata is also stored on a backend database to oversee the uploaded images' destination and facilitate dataset searching. Storing this data in the centralized database poses its own advantages because the data is easier to manage, back up, and update without much effort. Furthermore, it can always be compared to the blockchain copy of the database to ensure nothing has been interfered with. In this way, the bilateral storage mechanism contains the security and ease-of-access advantages that the blockchain and backend database provide, respectively.

*2) Client:* The user interface plays a pivotal role in ensuring a seamless experience despite the complex elements of the web application, allowing users to interact with the blockchain platform for data contribution and management. All pages are accessible through the navigation bar placed at the top of the page. The navigation bar contains Home, Dashboard, Create a Dataset, Contribute, Profile, Logout, Login, and Registration links. On /dashboard, /createdataset, and /contribute, the web3.js accesses the blockchain smart contracts to request the images. The Home Page has basic information about the application and highlights the brand by explaining the application objectives.

In the Registration/Login page (Figure 6), the authentication status of the user login is verified by a running variable in a JSX component called DataProvider. If this is set to False, the pages in the Navigation Bar are unavailable until the user logs in and the variable is set to True.

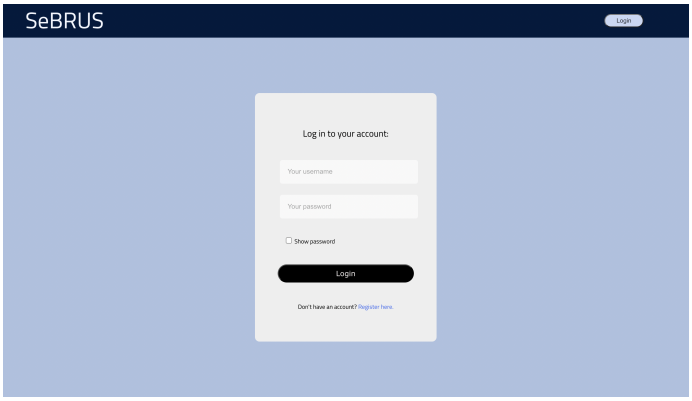


Fig. 6. The user can log in to their account or register if they do not have an account.

Users also have access to the Dashboard Page (Fig. 7) which allows users to explore available databases. This is achieved by making fetch requests to the backend datasets' endpoint for information on deployed datasets and compiling them into a list of dataset objects. A map function iterates through these objects to display information about each dataset.

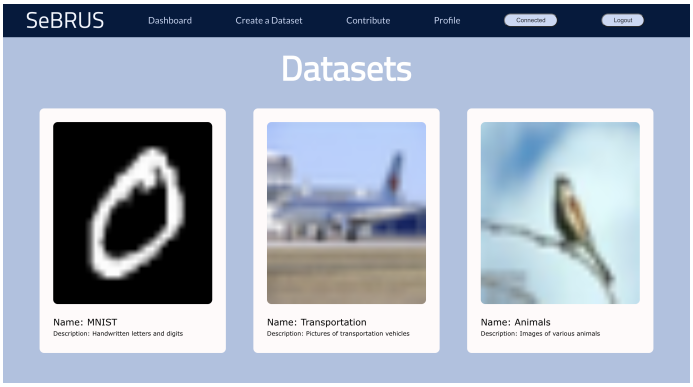


Fig. 7. All existing datasets are displayed for the user to browse on the Dashboard page.

Dataset pages display the images in a particular dataset from the Dataset smart contract.

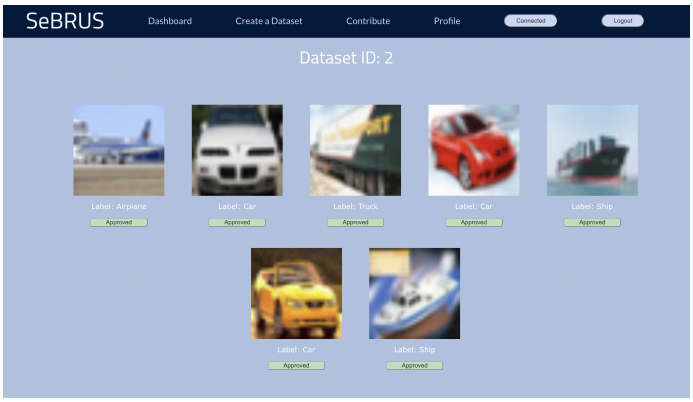


Fig. 8. The user can see all images in a dataset and their approval statuses.

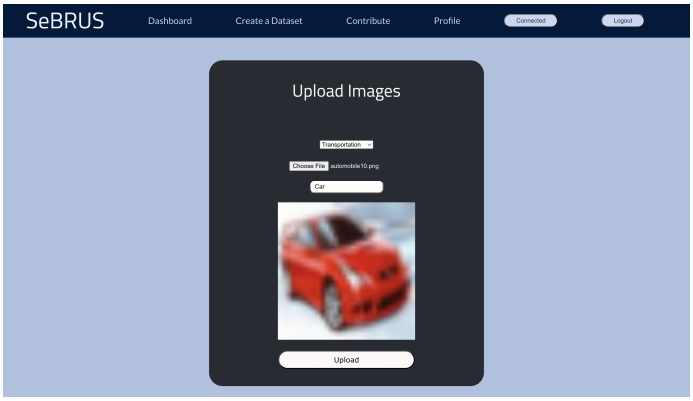


Fig. 9. The user can select which dataset to upload their image to and label the image accordingly.

Users are free to upload images to the blockchain database on the Contribute page (Fig. 9) through an 'upload image' button. This upload is restricted to a file type of images only, and the image size is restricted to images smaller than 10 MB. These images are converted into base64 strings, and the file is sent to the Dataset smart contract. Users select from a dropdown menu to indicate which dataset they want to contribute to. Then, requests are set up to the backend using fetch to keep track of uploads corresponding to the target dataset, which are ultimately made to the */api/dataset* endpoint.

In order to upload, the user must also connect their MetaMask account so that gas is available for transactions.

*3) Server:* Account processing (creation, deletion, etc) requires the server endpoint */api/user*, which processes the requests made to create, modify, or delete accounts and their information. Additionally, the endpoint */api/dataset* was implemented to easily track datasets and their metadata. Each dataset requires a unique address that the dataset smart contract points to and a description and name to make it searchable. For both endpoints, the defined schema in the Flask-SQLAlchemy ORM was implemented, acting as a bridge between datasets that allows the code to retrieve information from other datasets. An ORM tool, or Object Relational Mapping tool, is software that assists developers with interacting with relational databases [24]. The JSON handlers were used to obtain information, create new datasets or accounts, edit information, and delete files.

Furthermore, another endpoint */api/session* and database were created to determine whether a client is logged into their account. This endpoint addresses cases in which a user may have multiple sessions. SQLAlchemy ORM was used to create a Session table, where session information is stored, and */api/session* utilized JSON handlers in order to create, update, and delete sessions. This endpoint requires a Python decorator function, @authenticated, that takes a session cookie to check whether or not it is in the Session database.

Ethereum smart contracts require Application Binary Interface (ABI) configurations to act as an interface between client and the blockchain systems. The */api/abi* endpoint was created to handle the large amount of data ABI requires. This endpoint saves JSON files for the ABI to allow the client to connect with the blockchain and change the smart contract without significant client changes. Moreover, the implementation mimics the */api/user* and */api/dataset* endpoints through the administration of Flask-SQLAlchemy and JSON handlers.

*4) Blockchain:* Three Ethereum smart contracts, DataImage, Dataset, and DatasetManager, are implemented on the blockchain side using Solidity. The contracts are interfaced using Web3.js, Truffle, and MetaMask. Web3.js is a JavaScript library that facilitates interactions between Ethereum nodes and the client end after smart contracts are deployed. Truffle is a development framework for Ethereum that provides tools to build and deploy smart contracts. MetaMask is a cryptocurrency wallet that bridges SeBRUS on users' web browsers to the Ethereum blockchain and enables them to make transactions when uploading images.

DataImage contracts represent images uploaded to SeBRUS. They contain fields to record the value of the image as a base64 string, the label of the image as a string, and whether or not the image has been approved in a boolean. DataImage contains functions and events to transmit information about the image and approve the image.

Dataset serves as a decentralized equivalent of a machine learning dataset that creates DataImage contracts and contains string fields such as name and description. Each Dataset contract stores an array of DataImage contracts that comprise the machine learning dataset. The Dataset contract has corresponding functions and events for transferring information about the dataset and creating DataImages to add to the dataset. Cumulatively, the Dataset contract is interconnected to the user interface's Contribute page where users upload images, and the Dataset pages where images in each dataset are displayed.

The third smart contract, DatasetManager, also creates contracts and serves as the immediate interface between the blockchain and the client. Its purpose is to create new datasets, record information about datasets, and deploy Dataset contracts. Containing an array of Datasets, it has a corresponding function and event for creating datasets by deploying a Dataset contract. The DatasetManager contract interfaces with the Ethereum ABI to call functions from Dataset contracts and display datasets on the Dashboard page.

These smart contracts enforce the integrity of each submitted image through DataImage's approval function and the blockchain system. Through data irreversibility and poisoned data detection, the blockchain system prevents data from being altered via label-flipping attacks and prevents poisoned data from clean label and backdoor attacks from entering the system.

### *B. Data Integrity Defenses*

*1) Data Poisoning Detection:* A data poisoning detection network strengthens SeBRUS' security. Due to the potent effect that a small amount of infected data can have on

machine learning systems, this paper focuses on proposing a detection system to prevent this. Because the blockchain makes contributions highly trackable, potentially harmful images can be flagged for further review [25].

The model was trained using Keras (a machine training platform) and Adversarial Robustness Toolbox (ART), a Python library designed to defend and evaluate machine learning models against adversarial attacks. The resulting state was saved and used to analyze the images [26]. Once a user uploads a model to the system, the backend implements a new API endpoint that receives the serialized model from the frontend and loads it into ART's KerasClassifier class. A custom data loader is built to convert the base64 strings into image files. The loaded model passes into the ActivationDefense, which evaluates whether the image is poisoned, and returns the results to the frontend. The approval status of the image in both the blockchain and the user interface is changed to reflect these results.

*2) Voting and Incentive Design System:* A voting network is proposed to eliminate visually identifiable malicious data poisoning by identifying the data and then deciding if it should be added to the dataset or flagged. This process is done through a user network that votes on whether a certain image should be verified on the network. This vote is tallied separately from file uploads meaning that additions to the network would not be encumbered. If an image is rejected by the vote, it will be flagged and then removed.

Contributors are incentivized to vote on the validity of potentially poisoned data by receiving a reward. The contributors pull an image from the smart contract and decide whether or not the image is mislabelled. They then put up their own ETH and send their vote to the smart contract. In return they are given a small reward of ETH for their input. This encourages contributors to participate in the voting system and further verify the integrity of data in datasets. This process is outlined in Figure 10.

Because crowdsourcing this incentivization can be expensive, discretion of vote size is left to the dataset owners. This way, regardless of dataset size, security can be left up to the dataset owner. They can specify the number (x) of majority votes required to pass an image$\{x|0 \leq x \leq 100, x \in \mathbb{N}\}$. Higher costs ensure more security and is worthwhile for those who wish to ensure the integrity of their data.

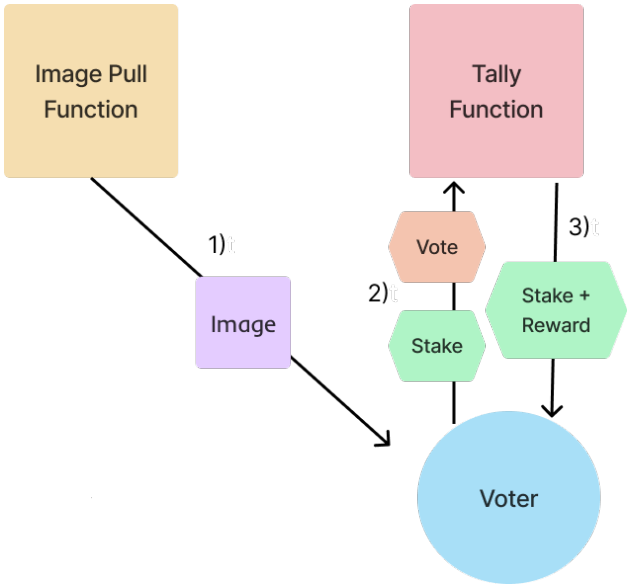


Fig. 10. 1) The voter pulls an image from the Dataset smart contract. 2) Voter analyzes the image; sends their vote + ETH stake to smart contract. 3) The tally function sends the vote tally to the database; reward + stake given to voter

## IV. Results

### A. System Design Analysis

This paper proposed a novel framework for securing crowdsourced data collection for machine learning against data poisoning attacks using novel defense designs and blockchain. The design of SeBRUS defends datasets against label flipping, clean label, and backdoor data poisoning attacks. Using a neural network trained to filter out mislabeled pictures is risky because the network would confirm its bias. Furthermore, if it had been previously poisoned, it would allow the infiltration of more poisoned data. For this reason, the human aspect of the proposed voting network is necessary. This voting network is designed to defend against label flipping data poisoning attacks, because voters are able to identify incorrect labeling. This approach is slightly prone to 51% attacks if adversaries push through enough votes to have their poisoned data infiltrated. However, this is highly uncommon and, with some more detective implementation, could be eliminated. Furthermore, the higher voting count options severely eliminate this possibility [14].

The proposed data detection system protects against clean label attacks and backdoor attacks. Since these attacks are generally undetectable for humans, clean label attacks and backdoor attacks are not accounted for by the voting and incentive system. The poisoned data detection system identifies poisoned data through a training model.

The Ethereum 2.0 and smart contract aspect of SeBRUS ensures that no data is tampered with once uploaded. The reduced computation of the PoS consensus mechanism reduces latency and environmental impact. Furthermore, the system ensures secure data, minimizing the amount of data monitoring and maintenance needed and lowering long-term costs.

Another aspect is to consider the cost of this security. The core of the system – the blockchain network – works by incentivizing nodes to consider and complete transactions. One needs gas to complete these transactions as a vital part of the smart contract system. This cost is dependent upon the security and size of images. However, the proposed defense of the system justifies the cost for important dataset collections. The blockchain guarantees a tamper-free handoff between client and database, whilst our other defenses significantly prevent poisoning attacks.

*1) Example Flow:* Bob, an adversary, attempts to attack this system. He tries to discover as much as he can about

this system. Presuming a worst case, he would gain access to significant pieces of the internal system. This would be a white box attack. He would well-disguise his poisoned data through a targeted clean-label attack. and implement a spam attack on the voter network. This describes the most advanced type of attack and anything significantly less sophisticated would be easily caught by the data detection, voter network, or general security measures. In the worst case situation, it is assumed that the attacker has divulged information from the database owner and that this is a white box attack. Experiments on the system were unable to be run.

### *B. Comparison to Other Solutions*

This section highlights differences between proposed data poisoning defenses mentioned in the background and SeBRUS.

Both SeBRUS and VAMP, a defense mechanism for crowdsourced training models, aim to secure crowdsourced datasets from adversaries, but they have some key differences. VAMP primarily focuses on protecting text-based files, while SeBRUS secures media-based datasets. Besides this, VAMP addresses merely one type of poisoning attack in which the adversary somehow changes the data. While this may protect against label-flipping attacks, clean label and backdoor attacks are not accounted for as there is no way for this system to detect triggers, limiting the scope of their solution. VAMP would not be able to identify inaccurate labels of data; it would only be able to bind the label to its corresponding data permanently, regardless of its validity. If this solution were to be implemented in a media-based system, it would only be able to detect modifications, such as the use of photoshop or deep fakes. However, it does not address the accuracy of the data itself. By contrast, SeBRUS accounts for all three data poisoning attacks, as mentioned in the System Design Analysis, through the voting consensus and implementation of smart contracts. Additionally, unlike SeBRUS's ability to be applied to current databases, VAMP cannot accomplish this task, according to Stokes et al. [21].

Although the KPCA method detailed by Li et al. [22] achieves supremacy over previous methods using PCA, this solution solely focused on federated learning systems against data poisoning attacks. Federated learning is currently less common than centralized machine learning systems, which makes SeBRUS a more relevant solution. Additionally, SeBRUS explicitly introduces security and incentive to a widely used data collection method: crowdsourcing. SeBRUS would encourage more users to contribute and use training data, generating a greater variety of secure training data. SeBRUS also presents a solid defense against three different types of data poisoning attacks (label flipping, clean label, and backdoor), whereas the KPCA method only prevents label flipping. Finally, SeBRUS prevents training data from being poisoned in the first place through the immutability and reliability of blockchain. In contrast, KPCA modifies the existing federated learning model after detecting data poisoning. To train machine learning models with one dataset, KPCA would have to be implemented in each machine learning system, whereas SeBRUS would only need to be implemented once on the crowdsourcing platform to secure clean data. Thus, the scope of the SeBRUS solution is wider than that of KPCA.

The framework SafeNet generally mitigates data poisoning and backdoor attacks, but only allows for data to be trained specifically to a model [23]. The trained model, not data, is available for sharing with this framework. SeBRUS, on the other hand, is a secure data crowdsourcing platform where datasets can be used for various models. This allows for easier integration of SeBRUS as it builds off of current crowdsourcing platforms and does not require users to implement machine learning techniques locally. Furthermore, SafeNet attempts to improve data quality by removing weaker models that may be poisoned. SeBRUS works to ensure that the data addition process and system is secure. Also, the concept of SafeNet relies on the idea that poisoned datasets will result in weaker models, thereby permitting data poisoning in stronger models. The voting network in SeBRUS requires that data be reviewed and approved, restricting poisoned data from the dataset completely.

Table I below illustrates the different features of these solutions.

TABLE I
A COMPARISON OF FEATURES BETWEEN PROPOSED DATA POISONING ATTACK DEFENSES AND SEBRUS

| Solution | Label-Flipping Defense | Clean Label Defense | Backdoor Defense | Ease of Implementation | Crowdsourcing Compatible |
|---|---|---|---|---|---|
| Stokes et al. (VAMP) | ✓ | X | X | X | X |
| Li et al. (KPCA) | ✓ | X | X | X | X |
| Chaudhari et al. (SafeNet) | X | ✓ | ✓ | X | X |
| SeBRUS | ✓ | ✓ | ✓ | ✓ | ✓ |

### *C. Limitations*

Though blockchain has asserted itself as a powerful cybersecurity system, there remain certain limitations to the scope of the solution that are worth considering. One of the key limitations is the problem of cost and energy consumption. Each transaction costs "gas," or the fuel used to facilitate Ethereum transactions. These are the tokens used to design the system. To address this limitation is the idea that several entities value high-quality datasets and that these entities need to account for the cost of ensuring these premium datasets. Interested stakeholders can be collaborated with to cover this overhead. For example, a non-profit that requires high quality health datasets should pay for the overhead cost of uploading images to this dataset.

As with any nascent solution, apprehensions regarding integration of new infrastructure into an existing framework are raised, especially since realistic approaches to attacks

against systems of this nature are often undocumented and some defense does exist today to address attacks. Nevertheless, upon analyzing existing security solutions to AI/ML datasets, SeBRUS's system prioritizes data security most prominently without the problems arising from implementing other solutions, as aforementioned in the Introduction. Additionally, it is incredibly feasible to incorporate this framework into an existing web app. Notably, Kaggle is a well-known crowdsourced data collection platform which, as aforementioned, is susceptible to attacks. Hence, partnering with a platform such as this to implement this security feature could enhance their existing framework while ensuring the security of this valuable crowdsourced data.

## V. Conclusion

### A. Significance of Findings

The research in this paper demonstrates that developing blockchain-based data acquisitions is well within most databases' capabilities. SeBRUS is highly modular and can be scaled depending on the fragility and importance of the dataset in question through the use of Ethereum smart contracts. Requiring blockchain authentication allows for trackable, yet anonymous contributions to networks and ensures that data remains unmodified once uploaded to the blockchain. Furthermore, this research concludes that database additions can be both easy-to-complete and secure, meaning that crowdsourcing can still be a viable solution for essential databases. Evidently, this solution will not be able to eliminate the issue of data poisoning entirely. Cyberthreats are ever-evolving, and data poisoning attacks are no exception. Hence, this framework needs to be scaled in order to continuously support maturing attacks. Though data poisoning attacks are at their early stages, there is potential for these attacks to become more sophisticated, necessitating constant re-evaluation of this solution. SeBRUS is more applicable to current technology than other solutions for data poisoning attacks because it can be feasibly implemented to existing crowdsourced datasets. SeBRUS is particularly novel through its approach to data poisoning defense, which focuses on system security rather than data security, making it unique among existing data poisoning solutions. Users will be more willing to upload data because of its incentive system and ease of access. Its poisoned data detection system provides security against attacks that are undetectable by humans. With the relevance of data poisoning in modern technology, this design offers a wide-reaching, scalable, and applicable solution.

### B. Future Improvements

*1) Data Handling:* Smart contracts have a high storage capacity, so storing longer base64 strings from large images is not an issue. However, image upload size is restricted by gas limits. If an image is too large, the amount of gas required to upload the image can surpass a gas limit set by many blockchain networks. Thus, implementing an image compression protocol would allow larger images to be uploaded. Nevertheless, SeBRUS still demonstrates smooth performance for smaller images, which is sufficient for SeBRUS' purposes. Larger images are not required for an effective machine learning model and are often not preferred over smaller images which reduce model training time. Two popular computer vision datasets—MNIST and CIFAR 10—contain small images of 28x28 and 32x32, respectively.

SeBRUS only focuses on image datasets for simplicity; however, it can easily be adapted for use with video, text, or other data types. Images are particularly interesting due to the unique data poisoning attacks against them and their importance in identification and surveillance systems. In all, expanding the size and types of data SeBRUS can handle would make it a more encompassing solution.

*2) Data Augmentation:* Future works should explore researching data augmentation defenses. This effectively involves poisoning all images added to a dataset with clean label attacks. This would eliminate a majority of the danger caused by data augmentation attacks because it effectively immunizes one's data. It reduces the efficacy of clean label attacks. However, this idea is beyond the scope of this paper. Furthermore, this defense can damage the quality of the dataset, and the resulting ML network. High costs for this data poisoning defense and the necessity of high-accuracy neural networks outweighed the feasibility of further investigating this idea [21]. If not for time restrictions, there would be more focus on implementing advanced defenses such as these. But as machine learning becomes all-encompassing, data poisoning attacks will become widespread and significantly more dangerous. Thus, augmentation will become much more viable [20].

*3) Voter Authentication:* In SeBRUS, verification is a proof of authority system, meaning voting power is limited to only a specified list of wallets. They vote on whether an image should be added; SeBRUS tallies the vote and then passes or flags the image. However, if SeBRUS is to be scaled, this solution to label-flipping defenses becomes untenable. The implementation of a broader, open-source verification system is advised. The desired security of this solution requires a slightly revised voting system. There are several specific ways to implement voter authentication, and research would have to focus on application in addition to the theoretical ability of these networks. The issue of incentivized voting was touched upon in this paper, however the realistic application of this is unknown. Pursuing more sophisticated approaches to data verification is encouraged.

## VI. Acknowledgements

The authors of this paper would like to thank the New Jersey Office of the Secretary of Higher Education, Rutgers University, the Rutgers University School of Engineering, and the alums of the New Jersey Governor's School of Engineering and Technology (NJ GSET). Their support in the continuation of NJ GSET has provided the authors with the opportunity to conduct scholarly research. The authors also

gratefully acknowledge Dean Jean Patrick Antoine, Director of NJ GSET, for managing and leading this insightful program. The authors would like to thank their Residential Teaching Assistant, Aryan Gandhi, for his assistance and motivation throughout the research and paper writing process. Finally, the authors of this paper express their sincerest gratitude to Benson Liu, their Project Mentor, for his expertise and involved supervision of their project. His receptiveness to questions, availability, and devotion to the project provided highly appreciated guidance to the authors. These groups and individuals all provided invaluable contributions that facilitated the success of this project.

## REFERENCES


[1] L. Munoz Gonzalez and E. Lupu, "The secret of machine learning," *The Magazine for the IT Professional*, 2018.

[2] L. Muñoz-González and E. C. Lupu, "The security of machine learning systems," *AI in Cybersecurity*, pp. 47–79, 2019.

[3] J. Lin, L. Dang, M. Rahouti, and K. Xiong, "Ml attack models: adversarial attacks and data poisoning attacks," *arXiv preprint arXiv:2112.02797*, 2021.

[4] L. Engstrom, B. Tran, D. Tsipras, L. Schmidt, and A. Madry, "Exploring the landscape of spatial robustness," in *International conference on machine learning*. PMLR, 2019, pp. 1802–1811.

[5] F. A. Yerlikaya and Ş. Bahtiyar, "Data poisoning attacks against machine learning algorithms," *Expert Systems with Applications*, vol. 208, p. 118101, 2022.

[6] A. E. Cinà, S. Vascon, A. Demontis, B. Biggio, F. Roli, and M. Pelillo, "The hammer and the nut: Is bilevel optimization really needed to poison linear classifiers?" *CoRR*, vol. abs/2103.12399, 2021. [Online]. Available: https://arxiv.org/abs/2103.12399

[7] W. R. Huang, J. Geiping, L. Fowl, G. Taylor, and T. Goldstein, "Metapoison: Practical general-purpose clean-label data poisoning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 12 080–12 091, 2020.

[8] X. Chen, C. Liu, B. Li, K. Lu, and D. Song, "Targeted backdoor attacks on deep learning systems using data poisoning," *arXiv preprint arXiv:1712.05526*, 2017.

[9] "What is layer 0 in blockchain?" January 2023. [Online]. Available: https://academy.binance.com/en/articles/what-is-layer-0-in-blockchain/

[10] W. Viriyasitavat and D. Hoonsopon, "Blockchain characteristics and consensus in modern business processes," *Journal of Industrial Information Integration*, vol. 13, pp. 32–39, 2019.

[11] "Consensus mechanisms," July 2023. [Online]. Available: https://ethereum.org/en/developers/docs/consensus-mechanisms/

[12] V. Buterin, "Ethereum: platform review," *Opportunities and Challenges for Private and Consortium Blockchains*, vol. 45, 2016.

[13] "Intro to ethereum," April 2023. [Online]. Available: https://ethereum.org/en/developers/docs/intro-to-ethereum/

[14] S. D. Levi and A. B. Lipton, "An introduction to smart contracts and their potential and inherent limitations," in *Harvard Law School Forum on Corporate Governance*, vol. 10, 2018.

[15] "Proof-of-stake (pos)," July 2023. [Online]. Available: https://ethereum.org/en/developers/docs/consensus-mechanisms/pos/

[16] H. S. Alenezi and M. H. Faisal, "Utilizing crowdsourcing and machine learning in education: Literature review," *Education and Information Technologies*, vol. 25, no. 4, pp. 2971–2986, 2020.

[17] B. Herfort, H. Li, S. Fendrich, S. Lautenbach, and A. Zipf, "Mapping human settlements with higher accuracy and less volunteer efforts by combining crowdsourcing and deep learning," *Remote Sensing*, vol. 11, no. 15, p. 1799, 2019.

[18] J. Steinhardt, P. W. W. Koh, and P. S. Liang, "Certified defenses for data poisoning attacks," *Advances in neural information processing systems*, vol. 30, 2017.

[19] P. W. Koh, J. Steinhardt, and P. Liang, "Stronger data poisoning attacks break data sanitization defenses," 2021.

[20] T. Y. Liu, Y. Yang, and B. Mirzasoleiman, "Friendly noise against adversarial noise: a powerful defense against data poisoning attack," *Advances in Neural Information Processing Systems*, vol. 35, pp. 11 947–11 959, 2022.

[21] J. W. Stokes, P. England, and K. Kane, "Preventing machine learning poisoning attacks using authentication and provenance," in *MILCOM 2021-2021 IEEE Military Communications Conference (MILCOM)*. IEEE, 2021, pp. 181–188.

[22] D. Li, W. E. Wong, W. Wang, Y. Yao, and M. Chau, "Detection and mitigation of label-flipping attacks in federated learning systems with kpca and k-means," in *2021 8th International Conference on Dependable Systems and Their Applications (DSA)*. IEEE, 2021, pp. 551–559.

[23] H. Chaudhari, M. Jagielski, and A. Oprea, "Safenet: The unreasonable effectiveness of ensembles in private collaborative learning," Cryptology ePrint Archive, Paper 2022/663, 2022, https://eprint.iacr.org/2022/663. [Online]. Available: https://eprint.iacr.org/2022/663

[24] "What is an orm – the meaning of object relational mapping database tools," October 2022. [Online]. Available: https://www.freecodecamp.org/news/what-is-an-orm-the-meaning-of-object-relational-mapping-database-tools/

[25] D. Solans, B. Biggio, and C. Castillo, "Poisoning attacks on algorithmic fairness," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2020, pp. 162–177.

[26] Foxglove144, "My Research Software," 12 2017. [Online]. Available: https://github.com/Trusted-AI/adversarial-robustness-toolbox/blob/main/examples/mnist_poison_detection.py