Towards Auditing the Sensitive Information Leakage During Data Trading and Data Computing

Shuhao Zheng  
shuhao.zheng@mail.mcgill.ca  
McGill University

Yanxi Lin  
lin-yx18@tsinghua.org.cn  
Tsinghua University

Yang Yu  
yangyu1@mail.tsinghua.edu.cn  
Tsinghua University

Ye Yuan  
ye.yuan3@mail.mcgill.ca  
McGill University

Yongzheng Jia  
abner@overality.io  
Overality Labs

Xue Liu  
xueliu@cs.mcgill.ca  
McGill University

ABSTRACT

Data markets enable users to own their data and trade their data for profits. Existing data market designs allow these data owners to directly sell or provide computation services on their private data in a secure and privacy-preserving way. However, an important problem that is neglected in previous work is that the transferred data or computation results over the data can leak information about other sensitive features that are not involved in the data market. To tackle this problem, we propose an auditing system based on smart contracts and zero-knowledge proof to measure and manage the sensitive information leakage risk in Web3 data markets. Primary results on the information leakage estimation show that even a small percent of information leakage can lead to significant predictability by simple machine learning models.

1 INTRODUCTION

One core concept of Web3.0 is to let users take charge of their own data instead of giving control to the centralized platforms, which is also called data sovereignty [1]. On the other hand, most platforms still have interest in users’ data for data-driven applications, especially for training machine learning models. A potential way to meet the needs of both the users (data owners) and the platforms (data consumers) is to allow users to own the data they create and trade the non-sensitive part of their data with platforms for profits, which is a concept called data market [2]. Nowadays, with the development of distributed ledger technology and modern cryptography, blockchain-based decentralized data market [3–6] is deemed to be a promising way to protect data privacy and guarantee trading security without involving any trusted parties.

Existing data markets mainly support two trading patterns: data ownership transfer [4–7] and data computing [3, 8]. Data ownership transfer is simply enabling data owners to sell their data to data consumers for profits. Once the data consumer gets the data, she obtains the rights to conduct any computation on the data. Data computing market is more complicated, in which data consumers can only require computation on the data and get back results without knowing the raw data. Data ownership transfer can be viewed as a special case of data computing, in which the data consumer requires a trivial computation to get the raw data.

However, a significant issue that previous work tends to ignore is that the transmitted data or computation itself can leak sensitive information of the underlying data, even if only the non-sensitive part is used. Take the user data in e-commerce platforms as an example. It is possible to infer the basic user profiles (e.g., age, gender) solely from the click and purchase records [9], which means the sensitive features and the non-sensitive features are not well-separated. Moreover, the computation itself can be exploitative, i.e., a malicious data consumer can exploit the sensitive data by carefully designing the computation. Therefore, in both scenarios, it is essential to audit the information leakage over the sensitive features in the raw data, in spite of whether the sensitive part is involved in the data market or not. In addition, it is also very important to have a trustworthy risk management system to monitor the information leakage and restrict the abuse of data once the the data consumer has acquired enough information to infer the sensitive features of the raw data.

To address the above issue, in this paper, we propose to audit the sensitive information leakage by measuring the mutual information between the sensitive features and the computation results, or in the data ownership transfer case, the non-sensitive features. Mutual information is a statistic measure of the dependency between two random variables, which quantifies the amount of information of one random variable that can be obtained by knowing the other random variable [10]. In this way, our measurement provides an information-theoretic estimation of the real information leakage risk. In order to make the auditing process trustworthy without revealing the data or computation results during auditing, we propose to let the data owner generate a zero-knowledge proof [11] to convince the data consumer of the auditing results. In this way, the auditing becomes publicly verifiable and the proof does not leak any information about the underlying data. Moreover, based on the information leakage measurement, we design a smart-contract-based risk management system to monitor the accumulated information leakage and regulate the data usage to defend against the data exploitation attack in data computing market.

As a primary experiment, we measure the mutual information between the sensitive and non-sensitive features in Adult Data Set [12], a public dataset that contains personal information collected from adults all over the world. Moreover, we connect the amount of information leakage and the predictability by training a simple machine learning model for classification. Results show that even 40% information leakage can result in 86.6% prediction accuracy on the test set, which serves as a lower bound for the exact predictability.

To sum up, our contribution is two-fold. Firstly, we design a blockchain-based system to audit the sensitive information leakage in data markets, an important issue that has been neglected in previous data market designs. In particular, we adopt zero-knowledge
proof to make the auditing process privacy-preserving and trustworthy. Secondly, we propose to use smart contract for the risk management of information leakage, which serves as an essential module to be considered for future data market designs.

2 RELATED WORK

Existing work in data market design highly concentrate on guaranteeing truthfulness and preserving privacy during data trading. Zhao et al. [4] proposed a blockchain-based fair data trading protocol to address the privacy concerns of data providers by integrating with the ring signature, double-authentication-preventing signature, and similarity learning. By using tokens and smart contracts, Wisbon [5], a decentralized data marketplace based on the blockchain, enabled the interaction between data buyers and sellers while preserving the necessary level of anonymity to some extent. Sterling [3], a decentralized marketplace for data computing, adopted Trust Execution Environments to enable privacy-preserving analysis and machine learning over private data. With the similar goal as Sterling, Song et al. [7] developed a privacy-preserving framework on a cloud-based data marketplace to safeguard the seller’s data and shopper’s machine learning model for data valuation before trading. Liu et al. [8] designed Dealer in which data valuation and privacy protection are both done through differential privacy. zkDE [6] exploited the zero-knowledge proof and non-fungible token (NFT) to provide data traceability to track the origin and transformations of datasets, as well as protect the data privacy while ensuring the fairness during the data exchanging process. However, none of the aforementioned data market designs considered the sensitive information leakage caused by knowing the data or the computation results themselves. Even differential privacy doesn’t help much because differential privacy only works for aggregation computations and only protects the privacy for the data involved in the computation. In this work, we formulate the information leakage auditing problem from an information-theoretic perspective and design a blockchain-based system to guarantee trustworthy and privacy-preserving auditing.

Another line of work recently proposed to use zero-knowledge proof to protect privacy of data owners during federated learning. Zero-knowledge proof is a powerful tool to protect data privacy and guarantee the system security at the same time. Guo et al. [13] and Burkhart et al. [14] adopted zero-knowledge proof to enhance the secure aggregation of local model weights. Nguyen & Thai [15] asked each data owner to submit a zero-knowledge proof for their local model training to defend against malicious data owners. In light of these pioneering work, we also adopt zero-knowledge proof in designing the auditing system to protect information security in a new aspect.

3 SYSTEM DESIGN

In this section, we first formulate the information leakage auditing problem. Then, in Section 3.2, we show an overview of our auditing system and briefly introduce each module of the design.

3.1 Problem Formulation

Each piece of data can simultaneously store both sensitive and non-sensitive features. Denote the non-sensitive features as \( x^{(ns)} \in \mathbb{R}^n \) and the sensitive features as \( x^{(s)} \in \mathbb{R}^m \). Here, \( n \) and \( m \) are the number of non-sensitive and sensitive features, respectively. Thus, a dataset can be separated into the sensitive part \( X^{(s)} = \left\{ \frac{X^{(s)}}{X} \right\}_{i=1}^N \) and the non-sensitive part \( X^{(ns)} = \left\{ \frac{X^{(ns)}}{X} \right\}_{i=1}^N \). The auditing for information leakage focuses on verifying whether the computing about the the non-sensitive features in \( X^{(ns)} \) cause the leakage of the information about the sensitive features in \( X^{(s)} \).

As data ownership transfer can be viewed as taking a trivial computation \( f(x) = x \) on the data and returning the results, we only discuss the information leakage in the data computing market for the rest of this paper. In the data computing market, a data consumer usually proposes a computation \( f(\cdot) \) on the private data owned by data owners. We measure the amount of leaked information by the mutual information \( I(X^{(s)}; Y) \) between the sensitive features \( X^{(s)} \) and the computation outputs \( Y = \left\{ \frac{\hat{y}_i; \tilde{y}_i = f(x^{(ns)})}{\tilde{y}_i} \right\}_{i=1}^N \) on the non-sensitive features \( X^{(ns)} \).

3.2 System Overview

Figure 1 shows an overview of the computation auditing system design. Completing a computation auditing requires 5 phases: Committing phase, Computation Proposing phase, Auditing phase, Verifying phase, and Decoding phase. It is worth noting that our system is not a complete design of a data market. The concepts data owner and data consumer just stand for who manage the data and who want to use the data, and they can both be individuals or organizations. The purpose of this work is to propose an essential auditing module for avoiding the data leakage during the computation, which can be plugged into many existing data market systems with slight modification.

3.2.1 Committing phase. In order to prove the information leakage measurement on the data, the data owner will first need to commit to the data to guarantee that the data used to generate the proof later will not be modified. The easiest way to commit to the data is to upload a hash value of the data onto the blockchain. Here we require the data owner to hash both the sensitive and the non-sensitive features of the data because both parts will be used in auditing. This hash value of the data can be stored either in smart contracts or in other trustworthy data-storing facilities such as the non-fungible tokens (NFTs), depending on the design of the data market. Later on, the data owner will only need to additionally prove that the pre-image of the hash value is used for auditing.

3.2.2 Computation Proposing phase. If a data consumer pursues the access permission to compute on the dataset, she proposes a computation \( f(\cdot) \) on the non-sensitive part of the data by uploading it to the smart contract. Moreover, the data consumer will need to upload a public key \( pk \) for the encryption of the computation results to avoid the results getting public. The smart contract will check the computation \( f(\cdot) \) and public key \( pk \) are well-formed before accepting the computation proposal. The scope of the computation is generic. It can be training machine learning models, extracting features of the data, or simply getting all the non-sensitive features which corresponds to the data ownership transfer case.
3.2.3 Auditing phase. During the computation auditing phase, the data owner performs the proposed computation on the non-sensitive features $X^{(n)}$ and gets results $Y$. To verify whether the non-sensitive features $X^{(n)}$ and the results $Y$ can leak the sensitive information, we propose to measure the possible information leakage by the mutual information $I(X^{(s)}; Y)$ between the results $Y$ and the sensitive data $X^{(s)}$. The data owner checks whether the computation results can be given to the data consumer by comparing the accumulated information leakage with an information leakage threshold $T$. The details for estimating the mutual information are illustrated in Section 4.3. In order to make the auditing process transparent to the data consumer, the data owner should specify and upload a concrete mutual information calculation function and the threshold $T$ in committing phase, where $T$ can be estimated from the Shannon entropy [16] of the sensitive features. In this way, the smart contract clarifies the conditions when the data-share requests will be denied. If the leakage is above the threshold $T$, the data owner rejects the computation proposal by uploading $(\text{reject, } MI, 0)$ along with a zero-knowledge proof $\pi$ proving the computation of $MI$ is correct. On the other hand, if the information leakage is acceptable, the data owner will encrypt the computation results with the data consumer’s public key $pk$ and publish the encrypted results by uploading $(\text{accept, } MI, Y_{enc})$ along with a zero-knowledge proof $\pi$ proving that the computation of $Y$ and $MI$, and the encryption $Y_{enc} = Enc(Y, pk)$ are all done correctly. The encryption scheme $Enc$ should also be specified in the committing phase for the convenience of proof verification. If the computation results are too large to be stored on-chain, the data owner can additionally apply a hash function on $Y_{enc}$, put $Y_{enc}$ in content-based decentralized storage such as IPFS [17], and only upload the hash value of the encrypted results. This will require the data owner to also prove the correctness of this hash computation when generating the zero-knowledge proof. The details of the zero-knowledge proof are described in Section 4.4.

3.2.4 Verifying phase. After the data owner uploads the computation results and the proof to the smart contract, the smart contract will automatically execute the verification of the proof. If the proof is valid, the smart contract will update the inner states related to the data computing. More specifically, if the auditing result is $\text{accept}$, the smart contract will update the accumulated information leakage according to the newly calculated mutual information $MI$. Otherwise, if the result is $\text{reject}$, the smart contract will simply record the calculated $MI$ as the reason for rejection, and the data consumer will not get the expected computation results. Moreover, after the proof is accepted by the smart contract, the data owner can be reimbursed directly from the smart contract for providing data and computation. The concrete design of the tokenomics and data valuation is out of the scope of this paper.

3.2.5 Decoding phase. If the computation passes the auditing, the encrypted computation results will be stored in the smart contract. Then, the data consumer can simply use her secret key $sk$ to decode the results and get $Y = Dec(Y_{enc}, sk)$. Moreover, as the whole computation is through a zero-knowledge proof, the data consumer can be convinced that the computation results, the computation auditing, and the encryption are all carried out correctly. Also, thanks to the zero-knowledge property, the data consumer learns no information about the data from the proof $\pi$. This concludes the computation auditing process for one-time data computing.

4 DESIGN DETAILS

In this section, we describe the design details of our system, including the security analysis, the selection of cryptographic algorithms, the estimation of mutual information and accumulated information leakage, and the smart-contract-based risk management system.

4.1 System Security

In this design, we consider the following 4 requirements for the security of the system:

1. **Correctness:** If the data owner is honest, she should be able to convince the data consumer that the auditing is carried out correctly.
2. **Soundness:** A malicious data owner cannot deliberately use different data for auditing, report wrong mutual information results, or upload wrong encrypted computation results.
3. **Privacy-preserving:** The data consumer cannot get any information of the underlying data from the hash value and the proof. Moreover, the data consumer cannot get any information of the computation results if the auditing fails.
4. **Liveness:** The data consumer can finally know the auditing results, and can get the requested computation results if the auditing passes.

The first three properties are guaranteed by the securities of the underlying hash function, the zero-knowledge proof protocol, and
the public-key encryption scheme. For liveness, there is possibility that the data owner rejects to audit the computation after the data consumer uploads the computation proposal. However, as a player in the data market, the goal of the data owner is to sell data for profits. If she does not perform the correct auditing, she will not be able to get rewards from offering data. Moreover, it is also possible to add a time limit for auditing and punish the data owner if she does not perform her duty within the time limit. We leave the concrete economical design of the data market for future work.

4.2 Hash Function & Encryption Scheme

To optimize the ZKP performance, it is crucial to choose protocols that are zk-friendly to minimize the size of the proof circuit. Therefore, we choose Poseidon hash as the hash function and ECIES with MiMC cipher as the public-key encryption scheme.

4.2.1 Poseidon Hash. Poseidon [18] is a hash function designed to minimize the number of constraints in R1CS (Rank-1 Constraint System) which is one kind of arithmetic circuit representations widely used in current zk-SNARK protocols. Less constraints imply smaller circuit size and thus faster computation for proof generation. Moreover, Poseidon also works naturally in a modular field (e.g., GF(p)), which perfectly fits zk-SNARK circuits.

4.2.2 Elliptic Curve Integrated Encryption Scheme (ECIES). As pairing-based zk-SNARKs are operated in prime fields, we choose ECIES as the asymmetric encryption scheme which also works in prime fields. In ECIES, the data owner and the data consumer first perform an elliptic-curve-based key exchange (a.k.a. Elliptic Curve Diffie–Hellman) to establish a shared symmetric encryption key. Then, they use the encryption key to perform symmetric encryption and decryption over the secret data. For the symmetric encryption, we choose MiMC cipher [19] as the encryption scheme to optimize for arithmetic circuits. MiMC encryption uses a very small number of multiplicative operations and is operated in prime fields, which makes it a suitable block cipher for zk-SNARK circuits.

4.3 Mutual Information Estimation

In Section 3.2.3, we mentioned that the computation of mutual information \( I(X; Y) \) between the results \( Y \) and the sensitive data \( X \) is conducted in the auditing phase of our design. In order to enable multiple computation proposals on the same data, it is also crucial to estimate the accumulated information leakage after multiple computations. For example, denote \( Y_1 \) and \( Y_2 \) as the results of two separate computations. The accumulated information leakage after the two computations is written as \( I(X; Y_1, Y_2) \), which is the mutual information between \( X \) and the joint random vector of \( Y_1 \) and \( Y_2 \). However, due to the complexity of the co-information estimation [20], there is no simple relation between \( I(X; Y_1, Y_2) \), \( I(X; Y_1) \), and \( I(X; Y_2) \). That is to say, the information gain from knowing multiple computation results can be larger, smaller, or equal to the sum of the information gain from each computation results. This brings a problem for estimating the accumulated information leakage of multiple computation proposals because we cannot simply add up the information leakage of each computation. Since the computations defined in our system are generic, it is also very hard to derive a general approximation of the accumulated information leakage solely from the single ones. On the other hand, as the computation results are not published on-chain, there is only consensus on the encrypted computation results. This means the zero-knowledge proof cannot take the previously computed results directly as inputs to calculate the co-information.

Therefore, to solve this problem, we propose to re-calculate each computation proposal and estimate the co-information between all the computation results and the sensitive data during auditing. Although this can bring a huge computation overhead during proof generation because the auditing circuit gets larger each time a new computation is proposed, the proof generation time is usually not so important in the data market auditing scenario. In order to avoid data exploitation from malicious data consumers, the data owner can spend weeks to months auditing the information leakage as long as the auditing process is reliable and trustworthy.

Mathematically, denote the total number of computations as \( t \) and the results of each computation as \( Y_i, 1 \leq i \leq t \). The accumulated information leakage is calculated by \( I(X; \bar{Y}) \) where \( \bar{Y} = (Y_1, Y_2, ..., Y_t) \) is the joint random vector of all \( Y_i \)'s. Thus, as a more general case, we show how to estimate the mutual information \( I(X; \bar{Y}) \) between the sensitive features \( X \) and all the computation results \( \bar{Y} \) below. Assume all the data points are independently sampled from the same distribution, i.e., the samples are iid. By the definition of mutual information we have

\[
I(X; \bar{Y}) = \int \int \cdot \ln \frac{P_{X|Y}(\bar{a}, \bar{b})}{P_X(\bar{a})P_Y(\bar{b})} \, \text{d}\bar{a} \, \text{d}\bar{b}, \tag{1}
\]

where \( X \) and \( \bar{Y} \) are random variables defined above, and \( P_{X|Y}(\cdot) \), \( P_Y(\cdot) \), and \( P_{X|Y}(\cdot; \cdot) \) denote their respective distributions and the joint distribution. In most of the real-world cases, we may not know the real distributions. Therefore, we need to estimate them from the data points.

We adopt the kernel density estimation (KDE) to estimate the distribution \( P_{X|Y}(\cdot) \). KDE is a non-parametric estimation for fitting the distribution of a random variable. We estimate \( \hat{P}_{X|Y}(\cdot) \) by

\[
\hat{P}_{X|Y}(\cdot) = \frac{1}{N} \sum_{i=1}^{N} K_{\mathbf{d}}(\cdot, \mathbf{z}_i^{(s)}), \tag{2}
\]

where \( N \) is the size of the data, \( \mathbf{z}_i^{(s)} \) is the \( i \)th data point in \( X \), \( \mathbf{d} \) is a parameter called bandwidth, \( K_{\mathbf{d}}(\mathbf{x}_1, \mathbf{x}_2) \) is the kernel function defined as \( 1 \left\{ \frac{||\mathbf{x}_1 - \mathbf{x}_2||}{\mathbf{d}} \leq 1 \right\} \) which equals to 1 when \( \frac{||\mathbf{x}_1 - \mathbf{x}_2||}{\mathbf{d}} \leq 1 \) and 0 otherwise. \( \frac{\mathbf{x}_i^{(s)}}{\mathbf{d}} \) is to take the largest integer below \( \frac{\mathbf{x}_i^{(s)}}{\mathbf{d}} \) for each dimension. We can estimate \( \hat{P}_Y(\cdot) \) and \( P_{X|Y}(\cdot; \cdot) \) in a similar way. Note that as \( \mathbf{d} \) converges to \( \mathbf{0} \), the estimated distributions \( \hat{P}_{X|Y}(\cdot), \hat{P}_Y(\cdot), \hat{P}_{X|Y}(\cdot; \cdot) \) converge to the real distributions \( P_{X|Y}(\cdot), P_Y(\cdot), P_{X|Y}(\cdot; \cdot) \). Therefore, we deduce an unbiased estimation of the mutual information using the estimated distributions

\[
\hat{I}(X; \bar{Y}) = \int \int \frac{\hat{P}_{X|Y}(\bar{a}, \bar{b})}{\hat{P}_{X|Y}(\bar{a})\hat{P}_Y(\bar{b})} \, \text{d}\bar{a} \, \text{d}\bar{b}. \tag{3}
\]
In the above calculation, the values of \( \hat{P}_{X^i}, \hat{P}_{\vec{y}}, \) and \( \hat{P}_{X^i} \) are constant on each hypercube of the form \( X_c = \bigcap_{i} \{ \hat{s}_i \oplus \hat{d}_X, \hat{s}_i+1 \oplus \hat{d}_X \} \), \( Y_c = \bigcap_{i} \{ \hat{t}_i \ominus \hat{d}_Y, \hat{t}_i+1 \ominus \hat{d}_Y \} \), and \( X_c \bigcap Y_c \) respectively, where \( \hat{s}_i \) and \( \hat{t}_i \) are integer vectors indicating the index of each hypercube, \( \ominus \) denotes the element-wise product, and \( \oplus \) denotes the union of intervals. Thus, in the practical computation, the integral can be reduced to the summation over all hypercubes, which becomes

\[
\hat{I}(X^{(s)}, \vec{y}) = \sum_{\bar{a} \in X_c} \sum_{\bar{b} \in Y_c} \hat{P}_{X^{(s)}}(\bar{a}, \bar{b}) \ln \frac{\hat{P}_{X^{(s)}}(\bar{a}, \bar{b})}{\hat{P}_{\vec{y}}(\bar{a}, \bar{b})} \quad (\text{4})
\]

As also shown in Section 3.1, we define \( Y \) as the set of results computed only on single data points, which means the computation cannot include any aggregation operations such as computing an average of some feature. The main purpose for adding this restriction is to simplify the mutual information estimation because the calculation is based on data samples. Any aggregation operation will reduce the number of samples in the results set \( Y \) which may contain only one data point in extreme cases. Therefore, if we still estimate the mutual information of the aggregated results, we will need to re-sample from the original data distribution to synthesize new datasets. This operation is computation-intensive and thus impractical to prove in zero knowledge.

In this paper, we solve this problem by biased estimation, i.e., only estimate the mutual information for single-point operations. While we still allow aggregation operations on the data points, such as computing the average of gradients for some machine learning models, we do not consider the decreased information leakage during aggregation. That is to say, we treat the mutual information before aggregation as the total amount of information leakage even after aggregation. In this way, we generate an upper bound of the information leakage for all kinds of computations. One deficiency of this solution is that the estimated information leakage can be larger than the real amount, adding restrictions on the scope of acceptable computations. However, this problem can be mitigated by allowing a larger threshold for the total information leakage. It is also feasible to roughly estimate the decreased information leakage for different aggregation operations and modify the final amount based on the specific computation. We leave this part of design as the future work.

### 4.4 Proof for Correct Auditing

Protocol 1 shows the whole auditing process of the data owner. The main task of the data owner is to generate a zero-knowledge proof to prove that the computation is carried out correctly, without leaking any information about the secret data. Afterwards, the proof can be publicly verified within the smart contract.

Totally four calculations should be included in the proof to pass the verification. The first calculation is to prove the data used for the computation auditing are the same as the previously committed one. The data owner should prove that the hash value of the data equals to the one stored in the smart contract. The collision-resistance property of the hash function guarantees that it is impossible for the data owner to generate different data with the same hash value as the committed one. The second calculation is to prove all the previously proposals in \( \text{Prev}_f \) are carried out correctly, i.e., \( Y_{ij} = f_i(X_{(n)}^{(s)}) \), \( i, j \). Although the results \( \{Y_i\} \) are not publicly revealed, they are taken as the inputs to the following calculations. Thus, the correctness of the following calculations can imply the correctness of this calculation. The third calculation is to prove the correctness of the mutual information calculation \( I(X^{(s)}; Y_1, \ldots Y_{\text{len}(\text{Prev}_f)}) \). As the function to calculate mutual information is pre-specified in the committing phase, the data owner cannot make wrong proofs of this calculation. Finally, in order to protect the privacy of computation and make it only available
to the data consumer, the data owner needs to encrypt the computation results of the last proposal with the public key \( pk \) and the pre-specified encryption function. The encryption process also needs to be proved to convince the data consumer. The four computations can be proved in one single zero-knowledge proof and be verified on the smart contract. Afterwards, the data owner only needs to decide whether to share the encrypted computation results \( \text{Enc} \) to the data consumers depending on whether the accumulated information leakage is above the threshold.

One thing worth noting is that specific ZKP protocols may require extra designs, such as a trusted setup [21], a circuit-specific verifier contract [22], etc. The details of these extra components are not illustrated in this paper. However, we do believe that adding these extra designs will not affect the structure of the auditing system. Also, there already exists some solutions in the industry to account for these designs, such as using secure multi-party computation to establish the universal Common Reference String (CRS) for Sonic [23]. Yet, based on the specific implementation, these designs may introduce extra assumptions on the system security.

### 4.5 Smart-Contract-based Risk Management

Along with the computation auditing, we embed a simple risk management system in the smart contract for proof verification. Since smart contract is a good place to store data that require consensus, we use the verification contract to record the accumulated information leakage for future computation proposals. Specifically, each time the auditing proof is verified, the smart contract will record the accumulated information leakage of the same dataset. Meanwhile, the smart contract also checks if the final decision of the data owner is valid, i.e., the computation results should not be published if the accumulated information leakage \( ML \) is already above the threshold \( T \). The smart contract will only accept the auditing results and mark the computation proposal as “finished” if both the zero-knowledge proof and the auditing decision are verified. As long as the previous proposal is not finished, the data consumer cannot propose a new computation on the same dataset. The complete design of the auditing contract is shown in Protocol 2.

There can be more complex designs for the risk management system. For example, the data owner can maintain an accumulated information leakage for each sensitive feature and check one-by-one for the computation auditing. Also, different sensitive features can have different information leakage threshold for the final decision. We believe future research can propose more innovative and accurate designs for the risk management system.

## 5 NUMERICAL RESULTS

We conduct primary experiments on measuring the sensitive information leakage during data ownership transfer on Adult Data Set [12]. Moreover, to connect with the real predictability, we use the non-sensitive features in the dataset to predict the sensitive features with machine learning models, which gives a lower bound on the sensitive information leakage risk.

### 5.1 Experiment Setup

Adult Data Set is a public dataset containing personal information collected from different countries, originally used to predict each person’s income based on their profiles. The personal information contains \( age \), \( workclass \), \( education \), \( education-num \), \( marital-status \), \( occupation \), \( relationship \), \( race \), \( gender \), \( capital-gain \), \( capital-loss \), \( hours-per-week \), and \( native-country \). Among all the features, we take \( age \), \( marital-status \), \( relationship \), and \( race \) as sensitive features because those are usually private information that people don’t want to share. Other features are all considered non-sensitive. For each sensitive feature, we calculate the mutual information between it with all the other non-sensitive features as the amount of information leakage by selling out all the non-sensitive features. We compare the amount of information leakage with the maximum information of each feature measured by Shannon entropy.

Furthermore, in order to show the direct information leakage risk by selling out all the non-sensitive features, we train a simple Multi-layer Perceptron (MLP) to predict each sensitive feature from the non-sensitive features. The MLP has only one hidden layer with 64 nodes and ReLU [24] activation. We train each prediction task for 100 epochs using Cross-entropy Loss and Adam [25] optimizer with \( 10^{-3} \) learning rate. Specially, since \( age \) is the only continuous feature, we divide it into intervals with length 10 and treat the intervals as different classes for classification. We randomly take 90% of the dataset for training and 10% for testing and report the test set accuracy compared with the random guess accuracy for each sensitive feature.

**Figure 2:** *Left:* Mutual information between each sensitive feature and all the non-sensitive features compared with their Shannon entropy. *Right:* Test set accuracy of predicting each sensitive feature using the non-sensitive features compared with random guess.

### 5.2 Results

As shown in Figure 2, the left chart shows the information leakage of each sensitive feature measured by the mutual information with other non-sensitive features, compared with their Shannon entropy. In the right chart, we plot the prediction accuracy of using non-sensitive features to predict each sensitive feature with simple MLP, compared with the random guess accuracy. It can be observed that the information leakage ratio for each sensitive feature ranges from 32% to 40%, leading to the increase in the prediction accuracy by 2.5 to more than 4 times. Particularly, all the non-sensitive features carry about 40% of the information of \( race \), which is the most among the four sensitive features, leading to a 67% increase in the prediction accuracy. This indicates that data consumers can almost accurately predict the \( race \) of a person given other non-sensitive information of that person if there are previously some labeled data for training, which implies a high information leakage risk in selling these non-sensitive information in the data market.

## 6 CONCLUSION & FUTURE WORK

In this paper, we emphasize the necessity of auditing the sensitive information leakage in data markets, an important problem that has
long been neglected by previous data market designs. We propose a smart-contract-based auditing system to estimate and monitor the sensitive information leakage risk in data markets with security and privacy guarantees provided by zero-knowledge proof. To evaluate the information leakage in reality, we conduct primary experiments on the information leakage risk estimation and predictability in a real-world dataset. For next steps, we will first implement the circuits for zero-knowledge proof and build our auditing system on a real-world blockchain to test its performance. Future work can focus on integrating our auditing system into existing data market designs to protect the sensitive information of data owners.

REFERENCES

[1] Patrik Hummel, Matthias Braun, Max Tretter, and Peter Dahroog. Data sovereignty: A review. Big Data & Society, 8(1):205395170982012, January 2021.

[2] Fan Liang, Wei Yu, Dou An, Qingyu Yang, Xinwen Fu, and Wei Zhao. A Survey on Big Data Market: Pricing, Trading and Protection. IEEE Access, 6:15132–15154, 2018.

[3] Nick Hynes, David Dao, David Yan, Raymond Cheng, and Dawn Song. A demonstration of sterling: A privacy-preserving data marketplace. Proc. VLDB Endow., 11(12):2086–2089, aug 2018. ISBN 2150-8097. doi: 10.14778/3229863.3236266. URL https://doi.org/10.14778/3229863.3236266.

[4] Yanqi Zhao, Yong Yu, Yannan Li, Gang Han, and Xiaojiao Du. Machine learning based privacy-preserving fair data trading in big data market. Information Sciences, 478:449–460, 2019. ISSN 0020-0255. doi: https://doi.org/10.1016/j.ins.2018.11.028. URL https://www.sciencedirect.com/science/article/pii/S0020025518309174.

[5] Matias Travizano, Carlos Sarrate, Mateusz Dolata, Aaron M. French, and Horst Treiblmaier. Wilvon: A Case Study of a Decentralized, Privacy-Preserving Data Marketplace, pages 149–170. Springer International Publishing, Cham, 2020. ISBN 978-3-030-44337-5. doi: 10.1007/978-3-030-44337-5_8. URL https://doi.org/10.1007/978-3-030-44337-5_8.

[6] Rui Song, Shang Gao, Yubo Song, and Bin Xiao. ZKDET: A Traceable and Privacy-Preserving Data Exchange Scheme based on Non-Fungible Token and Zero-Knowledge. page 11, 2022.

[7] Qiyang Song, Jiahao Cao, Kun Sun, Qi Li, and Ke Xu. Try before you buy: Privacy-preserving data evaluation on cloud-based machine learning data marketplace. In Annual Computer Security Applications Conference, ACSAC ’21, pages 260–272, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450385879. doi: 10.1145/3485382.3485921. URL https://doi.org/10.1145/3485382.3485921.

[8] Jinfu Liu, Jian Lou, Junxu Liu, Li Xiong, Jian Pei, and Jmeng Sun. Dealer: An end-to-end model marketplace with differential privacy. Proceedings of the VLDB Endowment, 14(6):957–969, February 2021.

[9] Omar Hasan, Benjamin Habegger, Lionel Brunie, Nadia Bennani, and Ernesto Damiani. A Discussion of Privacy Challenges in User Profiling with Big Data Techniques. The EEXCESS Use Case. In 2013 IEEE International Congress on Big Data, pages 25–30, June 2013.

[10] Hu Kuo Ting. On the Amount of Information. Theory of Probability & Its Applications, 7(4):439–447, January 1962.

[11] S Goldwasser, S Micali, and C Rackoff. The knowledge complexity of interactive proof-systems. In Proceedings of the Seventeenth Annual ACM Symposium on Theory of Computing, STOC ’85, pages 291–304, December 1985. ISBN 978-0-89791-151-1.

[12] Dheemo Dua and Casey Graff. UCI machine learning repository, 2017. URL http://archive.ics.uci.edu/ml.

[13] Jiale Guo, Ziyao Liu, Kwok-Yan Lam, Jun Zhao, Yiqiang Chen, and Chaoping Xing. Secure Weighted Aggregation for Federated Learning. Mobile Computing and Communications Review, 2022.

[14] Lukas Burkhalter, Hidde Lycklama, Alexander Viand, Nicolas Küchler, and Anwar Hithnawi. RoFL: Attestable Robustness for Secure Federated Learning. February 2022.

[15] True Nguyen and Ny T. Thai. Preserving Privacy and Security in Federated Learning, May 2022.

[16] C. E. Shannon. A mathematical theory of communication. ACM SIGMOBILE Mobile Computing and Communications Review, 5(1):3–55, January 2001.

[17] Juan Benet. IPFS - Content Addressed, Versioned, P2P File System, July 2014.

[18] Lorenzo Grassi, Dmitriy Khvorotovich, Christian Rechberger, Arnab Roy, and Markus Schofnegger. Poseidon: A New Hash Function for [Zero-Knowledge] Proof Systems. In 30th USENIX Security Symposium (USENIX Security 21), pages 519–535, 2021. ISBN 978-1-939133-24-3.

[19] Martin R. Albrecht, Lorenzo Grassi, Christian Rechberger, Arnab Roy, and Tyge Tiessen. MiMC: Efficient Encryption and Cryptographic Hashing with Minimal Multiplicative Complexity. In Jung Hee Cheon and Tetsuya Takagi, editors, Advances in Cryptology - ASIACRYPT 2016 – 22nd International Conference on the