Name: a data sharing system with provable guilty detection in data leakage

Anonymous Author(s)

ABSTRACT

Your abstract should go here. You will also need to upload a plain-text abstract into the web submission form.

CCS CONCEPTS

• **Security and privacy** → Use https://dl.acm.org/ccs.cfm to generate actual concepts section for your paper;

KEYWORDS

template; formatting; pickling

1 INTRODUCTION

Data sharing among authorized parties is increasing common today and a typical example is big data analysis outsourcing to third party companies. As data protection becomes a major concern, new regulation and laws has been coming out in recent years. For example, Europe's General Data Protection Regulation (GDPR) [?] became enforceable since May 2018, by which violators may be fined up to 4% of their annual worldwide turnover in case of an data breach [?]. California's Consumer Privacy Act (CCPA) [?] will be effective starting from 2020. Under this background, guilty detection in data leakage becomes a key challenge to achieve justice and enable the enforcement of new regulations.

Guilty detection problem in data leakage has been explored for years and the approaches fall into two main categories: (i) watermarking shared data through data alterations [?], (ii) allocating different portion of data across data receivers[?].

However, all of the existing solutions only guess who is more likely to be the leaker, which apparently cannot dictate a million-dollar ticket. Furthermore, existing solutions are not sufficient when the original data owner (called the sender) leaks the data instead of data receivers. In fact, there is still a big gap before we get an desired guilt detection mechanism. Specifically, there are 3 challenges in proving "who is the leaker":

- **Data leaker's ability of adding noise to the received data.** Carefully-added noise keeps the value of the data while makes it difficult for detection mechanisms to match the original data and leaked data even though content-based leakage detection algorithms are applied. Even though we can do it by some methods of computing similarity of leaked data and original data, there is no way to prove the matching is right.
- **Data sender could also be the leaker.** Most existing guilt detection approaches assume the data sender is honest. However, in the real world, the data sender may also disclose the information because of their dishonest employees or vulnerable company firewall, and then blame one of the data receiver.. Therefore, a desired detection approach should be able to prove the innocence of data receivers if they are not the leaker. On the other hand, the approach should also protect the sender so that a receiver cannot maliciously blame the sender as the leaker.
- **After data leakage, both sender and receiver may deny the facts of what has been sent/received.** Worse even, it is hard to prove who is lying. For example, a receiver who leaked the data may deny the receipt of a specific portion of the data.

To deal with above challenges, we first analyze **XXX** different structured datasets of **XXX** serious data leakage events in history. Our key observations are: (1) In **XXX** % structured datasets, there exists at least one "Critical KeyWords" (CKW) in each row of data. If leaker adds noise on CKW, the meaning of the data will be absolutely changed, thus makes the whole dataset loses its value. (2) When we remove **XXX** % CKW (and the corresponding data rows) from the dataset, there is nearly no negative effect on data analysis.

Taken the insight above, we present Name, a dataset sharing system with provable guilty detection in data leakage. In data sharing, sender decide only to send data rows with part of CKWs to receiver instead of the whole dataset. In this way Name can efficiently prove the leaker in a data leakage, while maintaining the value of dataset for third party analysis.

For specifically, we solve the challenges in 3 aspects:

- **Allocating data with specific CKWs to specific receivers.** We group rows of data by CKWs, then allocate data rows with specific CKWs to specific receivers. When we find CKWs in leaked data, we can figure out who is the leaker, even though leaker adds much noise on the other part of data.
- **Keeping received dataset unknown to the original data owner.** We apply Oblivious Transfer (OT) to let receiver drop data rows with some CKWs, in order to differentiate data sent by sender and data received by receiver. When data leakage happens, we can figure out it is sender or receiver who leaked the data.
- **Keeping receipts of data sent and received.** We lock the receipts of all data transferred in a ledger. After data leakage, we can open the box and check the receipts, then we can know what data is transferred and no one can deny that.

We implemented real-world system of Name. In our system, Following are our roadmaps and contributions of this paper:

- Motivation and our analysis of structure dataset. (§2-3)
- Name, a dataset sharing system with provable guilty detection in data leakage. (§4-8)
- Implementation and evaluation. (§9-10)

2 MOTIVATION

2.1 Background

2.1.1 *Data sharing.*

2.1.2 *Data leakage in data sharing.*

2.1.3 *Guilty detection when data leaked.*

2.2 Why cannot existing solutions prove "who is the leaker"

- **Data leaker's ability of adding noise to the received data.** Carefully-added noise keeps the value of the data while makes it difficult for detection mechanisms to match the original data and leaked data even though content-based leakage detection algorithms are applied. Even though we can do it by some methods of computing similarity of leaked data and original data, there is no way to prove the matching is right.
- **Data sender could also be the leaker.** Most existing guilt detection approaches assume the data sender is honest. However, in the real world, the data sender may also disclose the information because of their dishonest employees or vulnerable company firewall, and then blame one of the data receiver.. Therefore, a desired detection approach should be able to prove the innocence of data receivers if they are not the leaker. On the other hand, the approach should also protect the sender so that a receiver cannot maliciously blame the sender as the leaker.
- **After data leakage, both sender and receiver may deny the facts of what has been sent/received.** Worse even, it is hard to prove who is lying. For example, a receiver who leaked the data may deny the receipt of a specific portion of the data.

3 INSIGHTS FROM STRUCTURED DATASET

3.1 Dataset collection

3.2 Critical keywords in structured dataset

GuanYu: Design a statistic to show that keywords exists in most datasets.

Key Observations: In **XXX** % structured datasets, there exists at least one "Critical KeyWords" (CKW) in each row of data. If leaker adds noise on CKW, the meaning of the data will be absolutely changed, thus makes the whole dataset loses its value.

3.3 Data analysis with removal of partial critical keywords

GuanYu: Experiments: x axis-percentage of received data, y axis-accuracy of data analysis.

Key Observations: When we remove **XXX** % CKW (and the corresponding data rows) from the dataset, there is nearly no negative effect on data analysis.

3.4 Insights

According to our experiments presented above, we find that there does exist some Critical KeyWords (CKW) in most datasets which are immutable, and a good data analysis can be done with data rows of only partial CKWs. So we intend to group data rows with the same CKWs, and allocate data rows with different CKWs to different third parties and keep receipts of data transfer. Then when the data is leaked, we can figure out who is the leaker according to CKWs included in leaked data, and prove it based on receipts.

We intend to solve the problem in 3 aspects:

- **Allocating data with specific CKWs to specific receivers. (§5)** We group rows of data by CKWs, then allocate data rows with specific CKWs to specific receivers. When we find CKWs in leaked data, we can figure out who is the leaker, even though leaker adds much noise on the other part of data.
- **Keeping received dataset unknown to the original data owner. (§6)** We apply Oblivious Transfer (OT) to let receiver drop data rows with some CKWs, in order to differentiate data sent by sender and data received by receiver. When data leakage happens, we can figure out it is sender or receiver who leaked the data.
- **Keeping receipts of data sent and received. (§7)** We lock the receipts of all data transferred in a ledger. After data leakage, we can open the box and check the receipts, then we can know what data is transferred and no one can deny that.

4 NAME OVERVIEW

Name consists of 3 basic modules: Data Allocation, Oblivious Transfer and Ledger.

Data Allocation:

Oblivious Transfer:

Ledger:

Fig. **XXX** shows the workflow of Name system.

GuanYu: describe how Name works.

5 NAME: DATA ALLOCATION

6 NAME: OBLIVIOUS TRANSFER

7 NAME: LEDGER

8 GUILTY DETECTION AFTER DATA LEAKAGE

8.1 Workflow of guilty detection

8.2 Discussion of some special cases

8.2.1 *Multiple receivers leak data independently.*

8.2.2 *Multiple receivers communicate and leak data.*

9 EVALUATION

10 RELATED WORKS

10.1 Guilty detection in data leakage

10.2 Oblivious transfer

10.3 Ledger

11 CONCLUSION