Data Protection in Big Data Analysis

by

Masoumeh Shafieinejad

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Computer Science

Waterloo, Ontario, Canada, 2021

Examining Committee Membership

The following served on the Examining Committee for this thesis. The decision of the Examining Committee is by majority vote.

External Examiner:	Elisa Bertino Professor Dept. of Computer Science, Purdue University
Supervisor:	Florian Kerschbaum Associate Professor Dept. of Computer Science, University of Waterloo
Internal Members:	N.Asokan Professor Dept. of Computer Science, University of Waterloo Xi He Assistant Professor Dept. of Computer Science, University of Waterloo
Internal-External Member:	Lukasz Golab Associate Professor Dept. of Management Sciences, University of Waterloo

Author's Declaration

This thesis consists of material all of which I authored or co-authored: see Statement of Contributions included in the thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Statement of Contributions

This thesis was researched and written under the supervision of Professor Florian Kerschbaum. Contents from Chapters 2, 3, 4 and 5 are from joint works with Professor Ihab F. Ilyas, Dr. Robert Sim, Dr. Huseyin A. Inan, Marcello Hasegawa, Dr. Koray Karabina, Suraj Gupta, Jin Yang Liu, Nils Lukas, Xinda Li and Jiaqi Wang. The results of these projects are published in three papers [167, 166, 168] and a technical report [165].

Abstract

"Big data" applications are collecting data from various aspects of our lives more and more every day. This fast transition has surpassed the development pace of data protection techniques and has resulted in innumerable data breaches and privacy violations. To prevent that, it is important to ensure the data is protected while at rest, in transit, in use, as well as during computation or dispersal. We investigate data protection issues in big data analysis in this thesis. We address a security or privacy concern in each phase of the data science pipeline. These phases are: i) data cleaning and preparation, ii) data management, iii) data modelling and analysis, and iv) data dissemination and visualization. In each of our contributions, we either address an existing problem and propose a resolving design (Chapters 2 and 4), or evaluate a current solution for a problem and analyze whether it meets the expected security/privacy goal (Chapters 3 and 5).

Starting with privacy in data preparation, we investigate providing privacy in query analysis leveraging differential privacy techniques. We consider contextual outlier analysis and identify challenging queries that require releasing direct information about members of the dataset. We define a new sampling mechanism that allows releasing this information in a differentially private manner. Our second contribution is in the data modelling and analysis phase. We investigate the effect of data properties and application requirements on the successful implementation of privacy techniques. We in particular investigate the effects of data correlation on data protection guarantees of differential privacy. Our third contribution in this thesis is in the data management phase. The problem is to efficiently protecting the data that is outsourced to a database management system (DBMS) provider while still allowing join operation. We provide an encryption method to minimize the leakage and to guarantee confidentiality for the data efficiently. Our last contribution is in the data dissemination phase. We inspect the ownership/contract protection for the prediction models trained on the data. We evaluate the backdoor-based watermarking in deep neural networks which is an important and recent line of the work in model ownership/contract protection.

Acknowledgements

First of all, I wish to give my deepest thanks to my supervisor Florian Kerschbaum for his education, support and guidance in developing this work. I am sincerely grateful to Florian for bringing my trust back in respectful and thriving academic work environment.

I wish to express my gratitude for the members of the examining committee – Elisa Bertino, N. Asokan, Xi He, and Lukasz Golab – for taking the time to read through this thesis and providing insightful comments.

My decision to transfer to the University of Waterloo and being part of the CrySP lab, is one of the best I have made in my life. An exceptionally friendly group of people whom I learned a lot and received a great deal of support from, and had an amazing time with. Thanks to all who contributed to CrySP being as awesome as it is.

I wish to particularly thank those who made glowing memories in my phd life, each in their very own special way; thank you: [alphabetically ordered] Erin Atwater, Cecylia Bocovich, Nils Lukas, Navid Nasr, Maryam Sepehri, Lindsey Tolluch, Justin Tracey, and Nik Unger from the CrySP lab, and: Mina Farid, Saeed Nejati, Bahareh Sarrafzadeh, and indeed Nolan Shaw.

I am very grateful to the individuals near or far, who lit a candle with me in my non-bright moments: Melica Mirsafian, Priyaa Varshini, Nathan Harms, Maryam Elahi, Somaieh Taheri, Joel Reardon, and who became known as Dr. Professor Academia Smarts, Robin Cockett (after a movie by his son)! I can't thank you enough Maryam Bandari and Besat Zardosht, for always being there for me from wherever you were in the world.

Additionally, I offer appreciation at the free and open movements in software and knowledge. This thesis benefited from these movements significantly, from the code compiling for the experiments to producing the final copy of this text.

I express my deep personal gratitude to my family for their love and support. A special thank you to my mother Salimeh, for her giving the strongest spirit by her encouraging words. A big thank you to my father Rasoul, for raising me *a mountain flower* as he likes to put it, to make it happily through challenging times. Thank you Javad, Mohadeseh, Mahdi, Masoumeh, Zahra, and Mohsen. Thank you Zahra, Kosar, Faezeh, Hoda, Haniyeh, Yasamin and Yeganeh for your beautiful smiles bringing an enormous joy to my life.

Dedication

To the memory of my beloved sister Fatemeh, and all the health-care workers,

who put their lives for the rest of us to survive the difficult time of the Covid-19 pandemic;

To courage, love and life.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

We have been moving towards the digital world with feverish haste that springs from the urge for convenience, such as speeding up large scale data processing, or is an outcome of necessity, for example carrying on with life with the least physical contact during the Covid-19 pandemic. This has resulted in enterprises acquiring massive amounts of data, referred to as Big Data, from a variety of sources that power their applications and enable deeper and more informed analytics. Bertino and Ferrari [20] define the Big Data term as "a data management and analytics paradigm featuring 5V: huge data Volume, high Velocity (i.e., timely response requirements), high Variety of data formats, low Veracity (i.e., uncertainties in the data), and high Value". The data goes through a series of processes from cleaning to distribution as shown in four phases in Figure 1.1. Each phase could be subject to information leakage or theft if proper protection techniques are not applied. The Wikipedia page for data breaches, provides a list of 400 breaches during 2004 – 2020; published through press reports, government news releases, or mainstream news articles. The list includes those involving the theft or compromise of 30,000 or more records ranging from financial data to health information, while many smaller breaches occur continually. Another report [175] reveals that about 3.5 billion people saw their personal data stolen in the top two of 15 biggest breaches of the 21st century alone. The smallest incident among these 15 breaches, involved the data of a mere 134 million people. In addition, focusing on privacy violations when publishing personal data, Chen et al. [32] provide several examples of de-identification taking place as a result of linking attacks or data reconstructions. There are many open data-protection problems in big data analysis. We provide an example of security/privacy deficiencies for each of the four phases of the data science pipeline in this thesis.

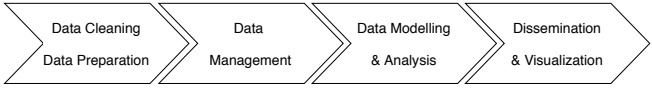


Figure 1.1: Security and Privacy in the Data Science Pipeline [106].

1.1 Contributions

During the data preparation phase, a trusted centralized data collector gathers raw data from individuals, and prepares the dataset to undergo future processing by an untrusted data analyst in any of data science phases. This preparation should provide sufficient protection for the privacy of individuals in the dataset against the data analyst. The data collector can guarantee this protection by applying encryption or privacy techniques in the preparation phase. In our first contribution, we consider using differential privacy techniques in the data preparation phase, for it i) randomizes the data, ii) guarantees a bound on the information leakage, and iii) is not affected by the post-processing performed by the data analyst. The application of differential privacy is particularly challenging in database queries for which no tailored mechanism exists. We consider contextual outlier detection among these challenging queries in Chapter 2, for its importance in i) error detection in the data cleaning phase, and ii) data interpretation in the data analysis phase.

The data collector in Chapter 2, publishes only the outlying data points and their corresponding contexts. This information is a point of interest for an analyst in the data cleaning phase who eliminates these outliers as errors in the dataset. It is also a point of great interest in the data analysis phase, where the analyst leverages this information to interpret the data for making decisions such as adjusting the clients' required monthly payments to an insurance company, based on the their *contextual* behaviour. In either case, the data collector needs to guarantee that the untrusted data analyst, who has access to the outliers and their contextual description, does not learn any information about the other individuals in the dataset.

Differential privacy for statistical queries of various types has been investigated in the literature. However, the studies of privacy in outlier analysis are few, all neglecting the privacy of non-outlier individuals, who define a *context* for an outlier. There are two main challenges in defining and applying privacy in contextual outlier release. The first major challenge is that the privacy technique must preserve the validity of the context for each outlier. We propose techniques to protect the privacy of individuals through a relaxed notion of differential privacy to satisfy this requirement. The second major challenge is

applying the proposed techniques efficiently, as they impose intensive computation to the base algorithm. To overcome this challenge, we propose a graph structure to map the contexts to, and introduce differentially private graph search algorithms as efficient solutions for the computation problem caused by differential privacy techniques.

During the data modeling and analysis phase, a trusted model designer trains a model on the individuals' data, and distributes the model to users for classification/prediction tasks later in the dissemination phase. These models are expected to only learn the general task, and to not leak information about any particular individual in the training set to the untrusted subset of users. However, the models are capable to memorize more than expected and are susceptible to membership inference attacks, violating the privacy of the individuals in the training set. A suggested approach to protect their data and prevent this privacy violation is using differential privacy.

Differentially private data does not reveal any information about participance/absence of any particular individual in the data set, and keeps this privacy guarantee while undergoing post-processing as well. However, when applied in its original form, differential privacy does not prevent inferring information about the data, if the data is not i.i.d and the adversary possesses information about the data distribution. For example, if it is known that four of five people in the same household have caught flu, assuming that the adversary is aware of the flu spread formulation, (s)he can infer the probability of the fifth person catching flu too. Similarly, through the model access, an adversary (untrusted user) can exploit the correlation information among the training data and gain knowledge about a target individual. We address this problem in Chapter 3, and show how we can take this correlation information into account in model training in order to strengthen the privacy guarantees.

We in particular consider the limits of differential privacy when used for confidentiality in social graphs. We assess models that are trained on organizational communications, e.g., emails. These models, when disseminated, carry unique risks of breaching confidentiality, even if the model is intended only for internal use. Current works that apply differential privacy techniques to natural language processing tasks usually assume independently distributed data, and overlook potential correlation among the records, resulting in a fictional promise of privacy. Naively extending differential privacy techniques to focus on group privacy instead of record-level privacy is a straightforward approach to mitigate this issue. This approach, although providing a more realistic privacy guarantee, is over-cautious and severely impacts model utility. We show the gap between these two extreme measures of privacy over two language tasks, and introduce a middle-ground solution. We propose a model that captures the correlation in the social network graph, and incorporates this correlation in the privacy calculations through Pufferfish privacy principles.

In the data management phase, the data owner manages their data in a database management system (DBMS). A resource-constrained data owner may outsource their data to a resourceful but untrusted DBMS provider for storage and computation purposes. The data owner desires to have the DBMS provider apply operations on the data, while not learning any information about it. The data owner can use encryption to reliably protect their data. However, some data management operations such as joins cannot be simply performed over encrypted data and need specialized encryption schemes. Furthermore, the data protection guarantees of the encryption schemes might reduce while the operations are performed over the outsourced encrypted data, and over a series of queries. We address this problem in Chapter 4 and contribute to reducing this information leakage in the data management phase.

We provide a new encryption scheme for the outsourced data to a database management system provider, e.g., in the cloud. The challenge is, performing operations such as equi-joins over encrypted data requires specialized encryption schemes that carefully balance security and performance. Our encryption scheme can efficiently perform equi-joins over encrypted data with less leakage than the state-of-the-art. In particular, our encryption scheme reduces the leakage to equality of rows that match a selection criterion and only reveals the transitive closure of the sum of the leakages of each query in a series of queries. Our encryption scheme is provably secure. We implemented our encryption scheme and evaluated it over a dataset from the TPC-H benchmark.

During the data dissemination phase, the trusted model designer who trained the model on the individuals' data earlier in the modeling and analysis phase, may distribute the model to users (sample owners) for classification/prediction tasks. The model designer desires restrictions in the model re-distribution due to i) ownership protection incentives, or ii) privacy concerns of the individuals in the dataset. The privacy concern arises from *purpose binding* notion where the data collected for one purpose should not used for another purpose without user consent. Watermarking models is a method to protect the models from an unauthorized redistribution where the adversary has either black-box or white-box access to the model. However, the watermarking schemes might be susceptible to removal attacks. We underpin this problem in Chapter 5.

We consider watermarking algorithms that have been introduced in the past years to protect deep learning models against unauthorized re-distribution. We investigate the robustness and reliability of state-of-the-art deep neural network watermarking schemes, namely *backdoor-based watermarking*. We propose two simple yet effective attacks – a black-box and a white-box – that remove these watermarks without any labeled data from the ground truth. Our black-box attack steals the model and removes the watermark with only API access to the labels. Our white-box attack proposes an efficient watermark

removal when the parameters of the marked model are accessible, and improves the time to steal a model up to twenty times over the time to train a model from scratch. We conclude that these watermarking algorithms are insufficient to defend against redistribution by a motivated attacker.

1.2 Background

Differential privacy is the base for the concepts and definitions we use in Chapters 2 and 3. Similarly, it is necessary to be familiar with bilinear groups to understand the design proposed in Chapter 4. The rest of this chapter covers the required background.

1.2.1 Differential Privacy

Differential privacy (DP), was introduced in 2006 by Dwork et al. [52]. To protect individual's data, DP mechanisms require that changing one record in the database should not change the output distribution much. The formal definition is as follows.

Definition 1.2.1 (ϵ-Differential Privacy [52])**.** A randomized algorithm $\mathcal{M}$ is said to provide ϵ-differential privacy if for any pair of neighboring databases D_1, D_2 differing in only a single entry, and for any set S which is a subset of possible outcomes of the randomized mechanism, $S \subseteq \text{Range}(\mathcal{M})$,

$$\frac{\Pr(\mathcal{M}(D_1) \in S)}{\Pr(\mathcal{M}(D_2) \in S)} \leq \exp(\epsilon) \tag{1.1}$$

the probability taken over the randomness of the algorithm $\mathcal{M}$.

A typical way to achieve ϵ-differential privacy for numeric queries is to apply the Laplace mechanism as shown by McSherry et al. [134]. The main idea is to apply noise to the output of a query for perturbation. The amount of noise depends on the sensitivity of the query and the privacy budget ϵ. To provide the formal definition for a privacy mechanism, we need to define *sensitivity* first.

Definition 1.2.2. [Sensitivity] Let d be a positive integer and $\mathcal{D}$ be a collection of datasets. For any function $f : \mathcal{D} \to \mathbb{R}^d$, the sensitivity of f, denoted by Δf, is defined by

$$\Delta f = \max_{D_1, D_2} ||f(D_1) - f(D_2)||_1 \tag{1.2}$$

where D_1 and D_2 are datasets in $\mathcal{D}$ differing in at most one element and $||\cdot||_1$ denotes the ℓ_1 norm.

Laplace Mechanism

The idea to achieve DP is perturbing the query function by the introduction of random noise generated according to a carefully chosen distribution.

Theorem 1.2.1. (The Laplace mechanism) Let d be a positive integer and $\mathcal{D}$ be a collection of datasets. For any $f : \mathcal{D} \to \mathbb{R}^d$, the randomized mechanism $\mathcal{M}$

$$\mathcal{M}(\mathcal{D}) = f(\mathcal{D}) + \text{Laplace}(0, \Delta f/\epsilon) \tag{1.3}$$

satisfies ϵ-differential privacy.

Sensitivity is a bound on the maximum effect of any element in the dataset, which will be helpful to provide privacy to *all* elements in the dataset. Based on this, for a given dataset D, a function f, and the privacy parameter ϵ, the Laplace mechanism adds Laplace$(0, \lambda)$ noise to the output of f where the parameter λ is determined by both Δf and ϵ.

We next introduce the exponential mechanism that is commonly used to provide differential privacy for non-numerical queries.

Exponential Mechanism

The goal of the exponential mechanism is to randomly map a set of n inputs each from a domain $\mathcal{D}$ to some output in a range $\mathcal{R}$ [136]. There are no specific assumptions about the nature of $\mathcal{D}$ or $\mathcal{R}$ other than a base measure μ on $\mathcal{R}$ exists. We proceed with the common assumption that the base measure μ is uniform. The exponential mechanism is driven by a utility function $u : \mathcal{D} \times \mathcal{R} \to \mathbb{R}$ that assigns a real valued score to any pair (D, r) from $\mathcal{D} \times \mathcal{R}$, higher scores indicate more appealing r's for D. We show the sensitivity of this utility function by Δu:

$$\Delta u = max_{(D_1, D_2) \in \mathcal{N}, r \in \mathcal{R}} = |u(D_1, r) - u(D_2, r)|. \tag{1.4}$$

Given a $D \in \mathcal{D}$ the goal of the mechanism is to return an $r \in \mathcal{R}$ such that $u(D, r)$ is (approximately) maximized while guaranteeing differential privacy.

Definition 1.2.3. (The exponential mechanism) [136] For any function $u : \mathcal{D} \times \mathcal{R} \to \mathbb{R}$, the exponential mechanism is defined as follows:

$Exp_u^{\epsilon}(D, \mathcal{R})$:= Choose r,

$$Pr[r] = \frac{exp(\frac{\epsilon u(D,r)}{2\Delta u})}{\sum_{r' \in \mathcal{R}} exp(\frac{\epsilon u(D,r')}{2\Delta u})}. \tag{1.5}$$

We include Theorem 1.2.2 for the privacy of the exponential mechanism and refer to McSherry et al.'s work [136] for the its proof.

Theorem 1.2.2. Theorem 1. (Privacy [136]). The exponential mechanism $Exp_u^{\epsilon}(d)$ gives $(2\epsilon\Delta u)$-differential privacy.

It follows from Theorem 1.2.2, for the exponential mechanisms to be the most useful, sensitivity (Δu) must be limited. Commonly, the utility function is chosen such that $\Delta u \leq 1$, for Exp_u^{ϵ} ensuring (2ϵ)-differential privacy.

1.2.2 Bilinear Groups

Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two distinct groups of prime order q, and let $g_1 \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$ be generators of the respective groups. Let $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ be a function that maps two elements from $\mathbb{G}_1$ and $\mathbb{G}_2$ onto a target group $\mathbb{G}_T$, which is also of prime order q. We follow the style of Kim et al. [109] to write the group operation in $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ multiplicatively and write 1 to denote their multiplicative identity. The tuple $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, q, e)$ is an asymmetric bilinear group [139, 97, 27] if these properties hold:

- The group operation in the groups $\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_T$ and the mapping e, are all efficiently computable.
- The map e is non-degenerate: $e(g_1, g_2) \neq 1$.
- The map e is bilinear: for all $x, y \in \mathbb{Z}_q$, this holds: $e(g_1^x, g_2^y) = e(g_1, g_2)^{xy}$

When dealing with vectors of group elements, for a group $\mathbb{G}$ of prime order q with an element $g \in \mathbb{G}$ and a row vector $\mathbf{v} = (v_1, \cdots, v_n) \in \mathbb{Z}_q^n$ where $n \in \mathbb{N}$, we write $g^{\mathbf{v}}$ to denote the vectors of group elements $(g^{v_1}, \cdots, g^{v_n})$. Also, for any scalar $k \in \mathbb{Z}_q$ and vectors $\mathbf{v}, \mathbf{w} \in \mathbb{Z}_q^n$, we write: $(g^{\mathbf{v}})^k = g^{(\mathbf{v}k)}$ and $g^{\mathbf{v}}.g^{\mathbf{w}} = g^{\mathbf{v}+\mathbf{w}}$. Furthermore, the pairing operation over groups is shown as: $e(g_1^{\mathbf{v}}, g_2^{\mathbf{w}}) = \Pi_{i\in[n]} e(g_1^{vi}, g_2^{w_i}) = e(g_1, g_2)^{\langle \mathbf{v}, \mathbf{w} \rangle}$.

Chapter 2

PCOR: Private Contextual Outlier Release via Differentially Private Search

2.1 Introduction

We borrow the contextual outliers example of Han et al. [79]: "The temperature today is 28°C. Is it exceptional (i.e., an outlier)? It depends, for example, on the time and location! If it is in winter in Toronto, yes, it is an outlier. If it is a summer day in Toronto, then it is normal. Unlike traditional outlier detection, in this case, whether or not today's temperature value is an outlier depends on the context — the date, the location, and possibly some other factors". Contextual outlier detection helps with the problem of *hidden* outliers — the data points that are considered *normal* compared to the whole population, but are outliers when compared to a subset of the dataset. Consider an analyst performing market research who wishes to determine the companies that are unusually profitable. While Company V might have normal reported profit compared to all companies, it may have outlying profit when compared to media companies with less than 2000 employees. In this case, the released context for Company V is [Business = media] $\wedge$ [$|Employees| < 2000$]. For the same data point V there might be multiple contexts (explanations) in which V is an outlier, each of which might have a different *utility* value. One example for utility is the size of the population that is covered by the context, which indicates the significance of the outlier. Another example is the context's overlap with a desired context, suitable for scenarios where the private explanation is required to be in a close relation with a chosen

context. The data owner releases a context for V that ideally achieves the highest utility. We assume reporting the outlier data point V is allowed when covered by the business purpose of performing the analysis. However, releasing the context — although useful for the data analyst — leaks information about other individuals in the dataset. This might not be permitted without the users' consent, according to privacy laws (e.g. GDPR [5]).

Technical Challenges

The privacy for statistical queries of various types, such as average, sum, variance, histogram, median, and maximum likelihood estimator, have been investigated [52, 147, 51] in the literature. However, the studies of privacy in outlier analysis are few, and mostly – described in more detail in Section 2.7 – addressing the privacy of the outlier [63, 129, 25, 151]. Consequently, the privacy of individuals in the remaining population has remained neglected, leaving the private contextual outlier reporting (PCOR) design as an open problem.

Differential privacy [50, 52, 53] is a popular privacy notion in data analysis [62] that has shown success in providing privacy when applied to products made by organizations such as the US Census Bureau [80, 131], Google [55], Apple [72] and Uber [95]. Since differential privacy i) guarantees privacy for individual records, ii) ensures privacy even in the presence of side information about the data, and iii) bounds the information leakage to a total privacy budget (ϵ) across multiple data releases; it thoroughly meets the privacy needs in PCOR. However, the challenges of utilizing differential privacy in PCOR are many, including defining a meaningful notion of differential privacy for contextual outliers, finding a proper technique that fits PCOR's requirements, and proving that the privacy budget is reasonably bounded. When queried about a record, PCOR must select a *private valid* context with ideally high utility among all candidates. To achieve this goal, we propose a variation of the exponential mechanism in PCOR, as this mechanism provides differential privacy when the nature of the candidates is discrete. However, the conventional definition of privacy and neighboring datasets in the exponential mechanism does not guarantee the validity of the context in the outlier release case. This motivates us to propose a variation of the exponential mechanism that guarantees the query response is valid and provides a relaxed notion of differential privacy. Our proposed technique, similar to the conventional exponential mechanism, in its formulaic application inevitably enumerates all possible contexts to select one. This requirement heavily affects the performance of the scheme, impeding its practicality. To summarize, our challenge is to provide a design for private contextual outlier reporting that

1. Defines a notion of differential privacy for contextual outliers, and provides a technique

for achieving it. As a result, in most cases, the output of PCOR is approximately the same ($\leq e^{\epsilon}$), even if any single record in the input database is arbitrarily added or removed;

2. Reduces the complexity from exponential time in the formulaic (direct) approach to polynomial time. This reduction however, should not affect whether the design supports differential privacy;

3. Achieves high utility. Supporting privacy and scalability, should not prevent the design from meeting its purpose in providing *useful* data for the analyst, and;

4. Is compatible with any utility function (e.g. population size) as well as any outlier detection algorithm, instead of being designed for a restricted number of specific cases.

2.1.1 Our Contributions

Florian Kerschbaum, Ihab F. Ilyas and I propose PCOR [167], a framework for privately releasing contextual outliers that addresses the aforementioned challenges:

1. PCOR utilizes a technique for a relaxed notion of differential privacy that prevents an adversary from learning information about the individuals in the context other than the reported outlier.

2. We propose deploying a sampling layer – based on a graph structure – prior to applying the exponential mechanism, to reduce the complexity of PCOR to polynomial time. In our efficient sampling, we analyze the Breadth-First and the Depth-First search algorithms. We modify these algorithms to support differential privacy for the outcome. We contribute the first presentation of differentially private graph search algorithms in the literature.

3. We prove our proposed solution is private and compare our algorithm in privacy and computational complexity to alternative solutions. The performance and utility of our algorithm is also supported by our experimental results.

4. We show that our framework can accommodate any utility function for the contexts. We provide results for two examples of utility functions that correspond to two separate classes of outlier release. One, evaluates the context based on the size of the population it covers while the other, evaluates the contexts based on its relation

(overlap) with a chosen context. Ultimately, we claim our algorithm works for the generic case of reporting the contextual outliers and is not limited to any particular outlier detection algorithm. Our claims are reinforced by experimental results over three algorithms for different outlier detection categories.

PCOR solves the challenge of simultaneously providing privacy, utility, and performance in reporting contextual outliers. Applied on a dataset of 50,000 records, PCOR reduces the runtime from three days in the direct differentially private approach to 37 minutes; while it maintains 90% of the maximum utility[1] and guarantees relaxed (output constrained, defined in Section 2.2.2) differential privacy with $\epsilon = 0.2$.

2.1.2 Chapter Organization

The rest of this chapter is organized as follows: In Section 2.2 we describe the outlier detection algorithms and define the notion of differential privacy we use. In Section 2.3 we provide the formal model and problem definitions for our private contextual outlier release (PCOR). We discuss the direct approach of embedding a differential privacy mechanism in contextual outlier detection in Section 2.4 and address the performance challenge. In Section 2.5 , we introduce a graph representation of contexts used in our efficient sampling algorithms. Privacy proofs of the candidate algorithms are also covered in Section 2.5. Their performance and accuracy are evaluated through the experiments described in Section 2.6. Our results in Section 2.6 confirm that our final candidate succeeds in achieving privacy, accuracy and efficiency. In Section 2.7, we situate our work in the current body of research, and in Section 2.8 we conclude this work.

2.2 Preliminaries

2.2.1 Outlier Detection Algorithms

Outlier detection algorithms fall into three main categories: i) Statistics-based, ii) Distance-based and iii) Model-based methods [93]. In statistics-based outlier detection techniques [73, 178, 31, 160, 133, 89, 4, 155] it is assumed that normal data points appear in high probability regions of a stochastic model, while outliers appear in its low probability regions.

[1]These numbers correspond to the experiments where the utility is "population size". The experiments for the "overlap" utility run even faster (Section 5.5.4).

There are two different types of statistics-based approaches: a) Hypothesis testing, and b) Distribution fitting approaches. The hypothesis testing [73, 178, 160] approach calculates a test statistic based on the data points, then it determines whether the null hypothesis claiming no outlier exists in the dataset, is valid or not. We evaluate Grubbs' test [73] as a hypothesis testing outlier detection, which expects the data to follow an approximate normal distribution. The test statistic is defined as $G = \frac{max_{i=1,\cdots,N}|Y_i - \bar{Y}|}{s}$, where $\bar{Y}$ is the sample mean and s is the standard deviation. The null hypothesis is rejected if $G > G_{critical}$. The distribution fitting approach [4, 89, 133, 155] first fits a probability density function to the data. Next, it labels the points with low probability as outliers. We select the Histogram approach [4] for the fitting distribution category to benefit from its strength in not relying on the assumption of knowing an underlying distribution. The histogram method first bins the range of values by dividing the range of the random variable into a series of consecutive and non-overlapping intervals. In the second step, it counts how the number of values fall under each bin. We use equi-width histogram, and choose the square root of the total number of samples as the number of bins. The data points that belong to bins of very low frequency are reported as outliers. Distance-based outlier detection techniques often define a distance between data points, which is used for defining a normal behavior [28, 158, 111, 110]. The outlier in this setting is a point that is distant from the others. We investigate Local Outlier Factor [28] as the distance-based outlier detection method in our experiments. The local outlier factor (LOF) method scores data points based on the density of their neighboring data points and detects the outlier points based on the scores. The LOF takes values approximately equal to 1 for a normal point versus $\gg 1$ for an outlier [28]. Hence, the factor can be used to detect outliers. Following paragraphs describe how the algorithm scores the data points. Given a positive integer k, the $k-distance$ of an object p, denoted by $k-distance(p)$, is defined as the $distance(p,o)$ between p and another object o, such that i) there exists at least k objects other than p in the dataset whose distance to p is less than or equal to $distance(p,o)$, and ii) there exists at most $k-1$ objects other than p in the dataset whose distance to p is strictly less than $distance(p,o)$. The $k-distance$ neighborhood of p, denoted by $N_k(p)$, contains all objects other than p whose distance to p is less than or equal to $k-distance(p)$. The reachability distance of an object p with respect to object o is defined as $reach-dist_k(p,o) = max\{k-distance(o), d(p,o)\}$.

Definition 2.2.1. The local reachability density of p is defined as:

$$lrd_k(p) = 1/(\frac{\sum_{o \in N_k(p)} reach-dist_k(p,o)}{|N_k(p)|}).$$

The local outlier factor of p is defined as:

$$LOF_k(p) = \frac{\sum_{o \in N_k(p)} \frac{lrd_k(o)}{lrd_k(p)}}{|N_k(p)|}.$$

Contrary to the outlier detection algorithms described above, model-based techniques detect the outliers in a supervised approach. However, in this project we focus on the unsupervised outlier detection algorithms as they do not require access to labeled data.

2.2.2 Output Constrained Differential Privacy

To provide privacy in our PCOR, we utilize a relaxed notion of differential privacy, namely Output Constrained Differential Privacy [88]. We postpone the reasoning for this choice to Section 2.3.1, after introducing the notations and defining the problem at the beginning of Section 2.3.

Definition 2.2.2. (f-Neighbors) Given function $f : \mathcal{D} \rightarrow O$, for any pair of datasets $D_1, D_2 \in \mathcal{D}$, the datasets D_1 and D_2 are neighbors w.r.t. f, denoted by $\mathcal{N}(f(.))$, if

1. $f(D_1) = f(D_2) \neq \emptyset$
2. $(D_1, D_2) \in \mathcal{N}$

The f-Neighbors D_1 and D_2, not only have to satisfy the general neighboring condition and differ from each other in only one record, but also need to result in the same (non-empty) output result/set when the function f is applied to them. In our outlier release use case, the function f outputs the set of all valid contexts in the dataset for a particular outlier. We elaborate outlier release in Section 2.3 and define f in Definition 2.3.1.

Definition 2.2.3. (Output Constrained Differential Privacy – OCDP) A randomized mechanism $\mathcal{M}$ satisfies $(\epsilon, f)-$OCDP, if for any $(D_1, D_2) \in \mathcal{N}(f(.))$, and every set of outputs $S \subseteq Range(\mathcal{M})$, we have:

$$Pr[\mathcal{M}(D_1) \in S] \leq e^{\epsilon} Pr[\mathcal{M}(D_2) \in S]. \tag{2.1}$$

Exponential mechanism guarantees an accuracy limited by an upper bound [136]. For PCOR's accuracy, we present empirical evaluations in Section 5.5 to measure the exact achieved accuracy.

Table 2.1: A sample of income data set D.

Record	Job title	City	District	Salary
1	Medical Doctor	Montreal	Business	S_1
2	Lawyer	Toronto	Business	S_2
3	CEO	Ottawa	Diplomatic	S_3
4	Lawyer	Toronto	Business	S_4
5	Lawyer	Ottawa	Diplomatic	S_5
6	Medical Doctor	Toronto	Historic	S_6
7	Lawyer	Ottawa	Business	S_7
8	Lawyer	Ottawa	Diplomatic	S_8
9	CEO	Montreal	Historic	S_9
10	Medical Doctor	Toronto	Diplomatic	S_{10}

2.3 Problem Definition

The outlier release encourages the anomaly or discourages it, due to approval or disapproval incentives respectively. In the approval incentive, the outlier benefits from the announcement as their outlying is desired. However, in the latter, the anomaly is undesired and needs to be proven to penalize the outlier. For approval incentive, imagine a funding agency that awards people with outstanding achievement in underrepresented groups. In this example, the agency is allowed to announce the outlier and the privacy-enhanced context, while any individual in that context is able to deny their participation in the funding competition. For disapproval incentive, imagine an insurance company increasing the annual payment of a client due to the client's abnormal behavior among a group of clients. While the insurance company is required to provide justification for this increase, they are not allowed to reveal information about any other client. To clarify how contextual outlier release violates the privacy of the participants, we provide an example of income analysis with concrete attribute values in Table 2.1. The goal in this example is to reveal the context that covers the largest population for an outlier. D is a dataset with categorical attributes of $Jobtitle = \{CEO, MedicalDoctor, Lawyer\}$, $City$ taking values from $\{Montreal, Ottawa, Toronto\}$, and $District$ with the domain $\{Business, Historic, Diplomatic\}$, and a numerical attribute Salary. Assume that the data owner reveals the record V, e.g. record 8 in Table 2.1, has its highest anomaly significance in the context $Jobtitle \in \{CEO, Lawyer\}$, in $Diplomatic$ district of $Ottawa$. By having a side information about the individuals in the maximum context, e.g. by knowing there is only one CEO living in $Diplomatic$ district of $Ottawa$, the revealed deterministic statement about the outlier V, leaks information about the

presence of the CEO in the dataset as well.

In our setup, we consider a dataset instance, D, of a relational schema, R. Attributes of R are presented by the set $attr(R) = \{A_1, \cdots, A_m, M\}$; where M is the *metric* attribute we define the outlier with regard to. We denote by P_{ij}, a predicate that is defined over the j^{th} value in the domain of a categorical or numerical attribute A_i derived from $attr(R)$. For instance, in our running example, P_{23} is the third attribute value in the second attribute in $attr(R) = \{Jobtitle, City, District, Salary\}$, which is $City = Toronto$. The domain of an attribute A_i includes all possible values for A_i, regardless of them being covered in the particular instance D; we refer to the domain size of A_i by $|A_i|$. In reporting private contextual outliers, it is necessary to take all the values in the domain of the attributes into account; the reason for this requirement is discussed in Section 4. A predicate P_{ij} filters the tuples from D to a subset that satisfies the predicate. This filtering can be represented as conjunction of disjunction predicates; $[P_{11} \vee P_{12} \vee \cdots \vee P_{1|A_1|}] \wedge \cdots \wedge [P_{m1} \vee P_{m2} \vee \cdots \vee P_{m|A_m|}]$; this format covers many common types of SQL queries.
Now, we can define a *context*, C. A context is represented as a binary vector of the form $\langle c_{11}, \cdots, c_{1|A_1|}, \cdots, c_{m1}, \cdots, c_{m|A_m|} \rangle$ with length $t = \sum_{i=1}^{m} |A_i|$. The bit c_{ij} in C is set to 1, if the predicate P_{ij} is covered in the context. C filters the dataset D to the population D_C. A tuple V belongs to D_C, denoted as $V \in D_C$, if V is selected by a set of predicates $P_{ij}, \forall i \in [1, m]$ in C, where $1 \leq j \leq |A_i|$. We denote the total number of tuples in C's population as $|D_C|$. In our example of the dataset with categorical attributes of $attr(R) = \{Jobtitle, City, District, Salary\}$ with the attribute domains of $\{CEO, MedicalDoctor, Lawyer\}$ for $Jobtitle$, $\{Montreal, Ottawa, Toronto\}$ for $City$, as well as $District$ attributes $\{Business, Historic, Diplomatic\}$, this combination of predicates: $[P_{11} \vee P_{13}] \wedge [P_{23}] \wedge [P_{32}]$, filters the dataset to the CEOs and $Lawyer$s in $Toronto$ who live in the $Historic$ district. The corresponding context to this subset of the dataset is represented by $C = \langle 101001010 \rangle$. Evidently, any non-empty context should include at least one predicate of each attribute, i.e. the context vector has a minimum Hamming weight[2], m. We call context C' connected to C, if the Hamming distance of C' and C is 1, i.e. C' is constructable from C by adding or removing (not both) only one predicate. For our running example of $C = \langle 101001010 \rangle$, being CEOs and $Lawyer$s in $Toronto$ who live in the $Historic$ district, the context $C' = \langle 100001010 \rangle$ which is CEOs in $Toronto$ who live in the $Historic$ district, is connected to C. Now, we can define the contextual outlier release. Given an outlier detection algorithm that takes as input any population D_C and the metric M, and outputs all outlier in D_C, we can implement the outlier verification method $f_M(D_C, V)$ to determine if V is an outlier in C with regard to the metric M; formally

[2]The Hamming weight of a binary vector is the number of 1's in the vector.

$$f_M(D_C, V) = \begin{cases} true, & \text{V is an outlier in } D_C \text{ w.r.t. M} \\ false, & \text{Otherwise} \end{cases}$$

We call a context C, a *matching context* for an outlier V with regard to metric M if and only if $f_M(D_C, V) = true$.

Definition 2.3.1. (Contextual Outlier Enumeration, COE_M) Given a dataset D with schema R, a metric attribute M, and an outlier verification function f_M for M, $COE_M(D, V)$ produces all matching contexts of V; i.e. all C's from the attributes $attr(R)$ such that $V \in D_C$, and $f_M(D_C, V) = true$.

To the best of our knowledge, the problem COE_M defined above is a hard problem with no existing efficient solution [176, 121]. Recall from Section 2.3, that each output of COE_M is assigned a score by a utility function, where higher scores indicate more appealing outputs. We focus on two types of utility, $u_V(D, C)$, of a context C for an outlier V: i) a quantitative evaluation of the context independent from other contexts. The population size of C is an very common example for this utility type, since it indicates the outlier's significance. ii) an evaluation of the context that may score the context based on its relation with a chosen context. As an example of this utility type, we investigate the overlap of the candidate context with a fixed starting context – the results are shown in Section 5.5.4. In either case, the data owner desires to report V with a context that is both private and achieves high utility. In this work, as implied in the definition, we only consider deterministic outlier detection algorithms embedded in COE_M.

Definition 2.3.2. (Private Contextual Outlier Reporting) Given a dataset D with schema R, a metric attribute M, a local outlier V, and a contextual outlier enumeration COE_M for M, private contextual outlier reporting ($PCOR$) produces a context C for V such that

(a) $f_M(D_C, V) = true$

(b) C is produced by a differentially private mechanism

(c) C is expected to maximize utility within $COE_M(D, V)$

(d) C is calculated in $\mathcal{O}(p(n))$

Note that the context C is the only context that is released to explain the outlying of V. All the valid contexts calculated for generating COE_M are solely known by the data owner and are used to compute C.

In the previous section, we elaborated on the outlier detection algorithms to deploy in $f_M(D_C, V)$, affecting the results of COE_M. We also defined differential privacy's [50] requirement for PCOR and presented the exponential mechanism for implementing it. In Section 2.3 we explain why we choose Output Constrained Differential Privacy for PCOR. Subsequently, in Section 2.3.2, we discuss PCOR's compatibility with any utility function through the two examples mentioned above.

2.3.1 Outlier-Preserving Privacy in PCOR

As shown in Equation 1.1, a mechanism $\mathcal{M}$ provides differential privacy, if the probability of obtaining any of its possible outcomes S changes with a maximum ratio of e^{ϵ}, if an input dataset D_1 is changed to any neighboring dataset D_2. Hence, if there is an S in range of $\mathcal{M}$ that has a zero probability of occurrence for a dataset D_1 but a non-zero probability for a neighboring dataset D_2, the differential privacy guarantee does not hold. The mechanism must range over all $S \subseteq Range(\mathcal{M})$ with non-zero probability. Since PCOR must output a valid context for V as the final answer, it assigns zero probability to all non-valid contexts. This zero probability is a violation of differential privacy, if adding or removing a record in a dataset D_1 changes a valid context S for V to a non-valid context for V in D_2. In PCOR, differential privacy is guaranteed by applying the exponential mechanism to $COE_M(D, V)$. $COE_M(D, V)$, as introduced in Definition 2.3.1, outputs the set of all contexts in which V is an outlier so that $f_M(D_C, V) = true$ – property (a) in Definition 2.3.2 – is satisfied. Therefore, to guarantee differential privacy as introduced in its original form as represented in Equation 1.1, for any outlier V and any neighboring datasets D_1 and D_2, the equality $COE_M(D_1, V) = COE_M(D_2, V)$ must hold. In other words, adding or removing a record, should not change the set of valid contexts for a particular outlier V. This condition is too strict for outlier detection, and as we show in our experiments in Section 2.6.7, there exist several neighboring datasets for various outlier detection algorithms that violate this constraint. Hence, we need some notion of differential privacy with satisfiable requirements for outlier detection algorithms, and we need to show that the shift to this notion, does not sacrifice the privacy guarantee that we are looking for. Since violating the constraint in the original differential privacy notion is algorithm-dependent, we propose a relaxed notion of (Output Constrained) differential privacy that is algorithm-dependent as well[3]. This notion guarantees $\epsilon-$differential privacy when the equality $COE_M(D_1, V) = COE_M(D_2, V)$ for the outlier detection algorithm holds. We ran several experiments for the next two observations: i) to what extent the $COE_M(D_1, V) = COE_M(D_2, V)$ assumption in OCDP

[3]Note that the common notion of differential privacy, $(\epsilon, \delta) - DP$, is not algorithm dependent.

matches the results of outlier detection algorithms in practice, ii) what are the effects if this assumption does not hold and whether it results in a privacy sacrifice. Our results include evaluations of this assumption in group privacy, where the datasets D_1 and D_2 differ in more than one record. We ran experiments on the outlier detection algorithms introduced earlier in Section 2.2.1, and provide the results in Section 2.6.7.

2.3.2 Utility Functions

As described in Section 2.3, the differential privacy level in a mechanism is directly related to the sensitivity of the utility function. We discussed in Section 2.3 that in order to support reasonable levels of privacy, we aim for utility functions with sensitivity close to 1. We introduce two such utility functions here: i) Maximizing the context population size, and ii) Maximizing the overlap with the starting context.

Context Population Size

The outlier verification function, f, can return *true* on various contexts for a tuple V. Intuitively, a larger context population indicates a higher significance of the outlier.

Definition 2.3.3. (Maximum Context) C is the Maximum context for outlier V with regard to f_M if and only if

(i) $f_M(D_C, V)$ returns true

(ii) $\forall C'$ s.t. $f_M(D_{C'}, V) = True$: $|D_C| \geq |D_{C'}|$

By choosing the population size as utility score, the mechanism guarantees the output to be *close enough* to the highest score answer, i.e. the maximum context. Formally,

$$u_V(D, C) = \begin{cases} -\infty, & f_M(D_C, V) = false \\ |D_C|, & \text{Otherwise} \end{cases}$$

The utility of $-\infty$ is assigned to non-valid contexts, so that it has zero probability to be picked by the exponential mechanism. Furthermore, since the population size would differ by at most 1, if replacing the dataset with a neighboring one, the utility function has sensitivity of 1.

Overlap with the Starting Context

Another subject of interest in outlier analysis is a utility function that scores a context based on its relation to a chosen context. For this category of utility functions, we focus on one that scores a context C based on its population's intersection with the population of a chosen/starting context for V, C_V. Formally,

$$u_V(D, C) = \begin{cases} -\infty, & f_M(D_C, V) = false \\ |D_C \cap D_{C_V}|, & \text{Otherwise} \end{cases}$$

Note that, changing a record in the dataset D alters the score, i.e. the intersection size, by at most 1 (sensitivity). As mentioned earlier, PCOR utilizes the exponential mechanism to provide differential privacy. We start with the direct application of the mechanism and argue why it imposes a substantial computational complexity to PCOR. To resolve the issue, we propose applying a sampling layer prior to deploying the the exponential mechanism. However, as we explain in Section 2.5.1, sampling in its basic form does not reduce the complexity of the direct approach. To improve the sampling method, we map the contexts and their connections to a graph. Searching over the graph, we observe "locality" in the matching contexts and use this property to propose efficient sampling methods for our PCOR. A random walk on the graph shows significant superiority over the basic sampling approach. To improve the sampling even further, we use the utility to direct the search and introduce differentially private versions of Depth-First and Breadth-First search algorithms for sampling. This results in a private scalable design for PCOR that achieves high utility.

2.4 Direct Approach for PCOR

In the direct application of the exponential mechanism in OCDP, we apply the mechanism over the outlier enumeration algorithm, as described in Algorithm 1. As we mentioned in Section 2.3, in order to provide more privacy protection for the individuals it is important that the enumeration algorithm considers all the values in the domain of attributes in $attr(R)$, not just the values covered in the dataset D. We show the rationality behind this requirement through the example in Section 2.3. The data owner reports the record V is an outlier in the context $Jobtitle \in \{CEO, Lawyer\}$, in *Diplomatic* district of *Ottawa*. By enumerating just over the values in the dataset, this report reveals that the individuals in the context are either CEOs or Lawyers. However, by enumerating over all possible values in the domain, the context description will be larger, e.g. $Jobtitle \in \{CEO, Lawyer, CFO, Diplomat\}$, in *Diplomatic* district of *Ottawa*. This re-

port leaves it unclear that which of the attribute values CEO, Lawyer, CFO, Diplomat is in the dataset.

Algorithm 1: PCOR - Direct Approach

Input : D, $attr(R)$, V, u, ϵ_1
Output: C_p

$C_M = \emptyset$
//Ranging over all possible contexts, to find matches
for $C \subseteq attr(R)$ **do**
  if $f_M(D_C, V) = true$ **then**
    $C_M \leftarrow C_M \cup C$
  end
end
//The exponential mechanism
return $C_p \leftarrow Exp_u^{\epsilon_1}(D, C_M)$;
Comment: A context C is a subset of the values of the attributes in $attr(R)$. All C's included in $COE_M(D, V)$ are candidates for the private output, but with a probability determined by the utility function u. The mechanism drawing from the candidates set terminates the algorithm.

Theorem 2.4.1. Algorithm 1 satisfies $(\epsilon = 2\epsilon_1, COE_M(., V)) - OCDP$, according to Definition 2.2.3.

Proof. Consider the neighboring pair $(D_1, D_2) \in N(f(.))$, for COE_M being the outlier enumeration used in Algorithm 1. According to Definition 2.2.2, D_1 and D_2 differ in one record and result in the same range of outputs, C_M. To prove the algorithm satisfies the claim in the Theorem 2.4.1, we need to show that the probability of algorithm outputting a particular C_p^* for D_1 is $2\epsilon_1$-different from D_2.

$$\frac{Pr[Alg1(D_1) = C_p^*]}{Pr[Alg1(D_2) = C_p^*]} = \frac{\frac{e^{\epsilon_1 u(C_p^*, D_1)}}{\sum_{c \in C_M} e^{\epsilon_1 u(c, D_1)}}}{\frac{e^{\epsilon_1 u(C_p^*, D_2)}}{\sum_{c \in C'_M} e^{\epsilon_1 u(c, D_2)}}}$$

$$= \frac{e^{\epsilon_1 u(C_p^*, D_1)}}{e^{\epsilon_1 u(C_p^*, D_2)}} \times \frac{\sum_{c \in C'_M} e^{\epsilon_1 u(c, D_2)}}{\sum_{c \in C_M} e^{\epsilon_1 u(c, D_1)}} \quad (2.2)$$

Since $|D_1| - |D_2| = 1$, any context selected for D_1 loses in maximum one record when selected for D_2. The utility for each context is proportional to its population size, hence its sensitivity is at most 1, i.e. $\Delta u \leq 1$. This means that that for any context in R (including C_p') the relation $u(c, D_1) - u(c, D_2) \leq \Delta u \leq 1$ holds. Hence the Equation 2.2 simplifies to

$$\frac{Pr[Alg1(D_1) = C_p^*]}{Pr[Alg1(D_2) = C_p^*]} \leq e^{\epsilon_1 \Delta u} \times e^{\epsilon_1 \Delta u} = e^{2\epsilon_1 \Delta u} \leq e^{2\epsilon_1} \tag{2.3}$$

Note that the number of elements in C_M is the same as C_M', due to the neighboring definition for output constrained differential privacy. However, the population size for the contexts in C_M and C_M' might differ in one record. □

Theorem 2.4.2. The computation complexity of Algorithm 1 is $\mathcal{O}(2^{t+1})$, where t is the total number of attribute values.

Proof. Direct application of the exponential mechanism requires brute forcing over all possible contexts for V (lines 3-7 in Algorithm 1) and find the matching ones. With a context being a binary vector $\langle c_{11}, \cdots, c_{1|A_1|}, \cdots, c_{m1}, \cdots, c_{m|A_m|} \rangle$ of length $t = \sum_{i=1}^{m} |A_i|$, this phase is $\mathcal{O}(2^t)$. It also calculates the weight of each candidate (lines 3-7), which is $\mathcal{O}(2^t)$ as well. Hence, Algorithm 1 is $\mathcal{O}(2^{t+1})$. □

2.5 Sampling Approach for PCOR

To resolve the efficiency problem in PCOR, we propose deploying a sampling layer, prior to applying the exponential mechanism. We start with Uniform Sampling, and analyze its privacy and complexity of the algorithm.

2.5.1 Uniform Sampling

The first sampling algorithm we evaluate, picks contexts uniformly from the set of all valid contexts. We apply the exponential mechanism after sampling. The idea of utilizing Uniform Sampling prior to the exponential mechanism was investigated by Lantz et al. [116].

Theorem 2.5.1. The Uniform Sampling in Algorithm 2, satisfies $(\epsilon = 2\epsilon_1, COE_M(., V)) - OCDP$.

Proof.

$$\frac{Pr[Alg2(D_1) = C_p^*]}{Pr[Alg2(D_2) = C_p^*]}$$

$$= \frac{\frac{e^{\epsilon_1 u(C_p^*, D_1)}}{\sum_{c \in C_M} e^{\epsilon_1 u(c, D_1)}} \times Pr_{D_1}[C_p^* \in C_M]}{\frac{e^{\epsilon_1 u(C_p^*, D_2)}}{\sum_{c \in C'_M} e^{\epsilon_1 u(c, D_2)}} \times Pr_{D_2}[C_p^* \in C_M]}$$

$$\leq e^{\epsilon_1 \Delta u} \times e^{\epsilon_1 \Delta u} \times 1 = e^{2\epsilon_1 \Delta u}. \quad (2.4)$$

The first two items in the inequality in Equation 2.4 follows the same reasoning as the proof in Theorem 2.4.1. The $Pr[C_p^* \in C_M]$ is independent of the database being D_1 or D_2, as the contexts are chosen based on attribute values not the records, and the attribute values for D_1 and D_2 are the same. □

Algorithm 2: PCOR - Uniform Sampling

Input : D , $attr(R)$, V, u, ϵ_1, p ($p = \frac{1}{2}$ here)
Output: C_p

1 //Finding n matching contexts
2 $C_M = \emptyset$
3 **while** $|C_M| \leq n$ **do**
4 **for** $i \leq t$ **do**
5 $C[i] = \begin{cases} 1, & \text{w.p. p} \\ 0, & \text{w.p. 1-p} \end{cases}$
6 **end**
7 **if** $f_M(D_C, V) = true$ **then**
8 $C_M \leftarrow C_M \cup C$
9 **end**
10 **end**
11 //The exponential mechanism on the samples
12 return $C_p \leftarrow Exp_u^{\epsilon_1}(D, C_M)$;
13 **Comment:** We form the context vector, by setting one's and zero's randomly. In the general case the probabilities are correspondingly p and $1 - p$. We consider $p = \frac{1}{2}$ to achieve Uniform Sampling. We obtain the C_M from the sampling, then apply Exp. mechanism as in Algorithm 1.

Theorem 2.5.2. The computation complexity of Algorithm 2 is $\mathcal{O}(2^t)$, where t is the total number of attribute values.

Proof. We calculate the complexity of Algorithm 2 in lines 3-10 first. The algorithm keeps sampling among all 2^t contexts until it finds n matching contexts for V. We want to find out how many contexts the algorithm should sample on average to achieve the goal. Assume the total number of matching contexts for V is N; i.e. the probability of a sample being a matching context is $\frac{N}{2^t}$. Hence, the number of success in sampling follows Binomial distribution $B(x, \frac{N}{2^t})$, with x being the total number of samples [46]. Since the expected value of the number of success in Binomial distribution in this case is $x \times \frac{N}{2^t}$, we need $n \times \frac{2^t}{N}$ samples to obtain n matching contexts on average. Furthermore, Algorithm 2 runs the exponential mechanism on the matching contexts, which adds $O(n)$ complexity. Therefore, Uniform Sampling is $\mathcal{O}(\frac{n \times 2^t}{N} + n)$. Considering n and N being constant values, Uniform Sampling does not effectively change the complexity of the direct approach. □

2.5.2 Graph-based Sampling

We assume the data owner knows a valid starting context[4] C_V for a local outlier V and desires to find the private maximum one through PCOR. We map this problem to a search over a graph G initiating form V's starting context C_V, aiming to reach the maximum context. We define the context graph with the set of vertices $Vtx = \{C\}$, where $\{C\}$ is all possible contexts defined over the attribute values from $attr(R)$. There is an edge between two contexts C and C' if they are *connected* to each other, i.e. they are different in the presence of just one attribute value. Every context C can be represented as a binary vector of the form $\langle c_{11}, \cdots, c_{1|A_1|}, \cdots, c_{m1}, \cdots, c_{m|A_m|} \rangle$ with length $t = \sum_{i=1}^{m} |A_i|$. As a result, flipping a bit in C forms a binary vector for context C' which is different from C in just one (added/removed) attribute value. Hence, there are t number of C's connected to C; i.e. each vertex is of degree t. Given that the connected vertices are different in just one attribute value, we hypothesize that if V is an outlier in C, then it is more probable to be an outlier in a connected vertex than some randomly chosen vertex among Vtx. In other words, we hypothesize the existence of *locality* in outlier detection algorithms. We show in this work the hypothesis holds for algorithms from any outlier detection category. We start by our first sampling algorithm on the context graph: Random Walk.

[4]The data owner can obtain this context through an initial search.

Random Walk Sampling

The data owner knows a valid context for a local outlier V, namely starting context C_V, and desires to find the private maximum one. In random walk sampling, we initiate from the starting context and change the attribute values to obtain the next context to continue the chain. The change in the context in each iteration includes adding/removing an attribute value to/from the context.

Algorithm 3: PCOR - Random Walk Sampling

Input : D , $attr(R)$, V, C_V, u, ϵ_1
Output: C_p

$C_M = [C_V]$
//n is the total number of samples
while $|C_M| \leq n$ && $C_{conn} \neq \emptyset$ **do**
  $C \leftarrow C_V$
  $C_{conn} = \{\text{connected vertices to } C\}$
  //C_i is selected randomly from C_{conn}
  $C_i \xleftarrow{r} C_{conn}$
  if $f_M(D_{C_i}, V) = true$ **then**
    $C_M \leftarrow C_M \cup C_i$
    $C_V \leftarrow C_i$
  end
  else
    $C_{conn}.remove(C_i)$
  end
end
//The exponential mechanism on the samples
return $C_p \leftarrow Exp_u^{\epsilon_1}(D, C_M)$;

Comment: The random walk adds/removes a random attribute value to/from a starting context to obtain the next context, the algorithm continues to do so until the next context is a matching one for V, this context is the second sample. We continue random walking from the obtained matching context, until we reach the desired number of sampled contexts in the multiset C_M. The the exponential mechanism is applied afterwards as in Alg. 1 to the samples to select the final differentially private output.

Theorem 2.5.3. The random walk sampling described in Algorithm 3 satisfies ($\epsilon = 2\epsilon_1, COE_M(., V)$)–OCDP, where ϵ_1 is the privacy parameter in the exponential mechanism.

Proof. To calculate the privacy provided by an algorithm, we evaluate the probability change of the privacy mechanism outputting a particular value when switching from a database D_1 to a neighbor D_2. These probabilities are shown by $Pr[Alg3(D) = C_p^*]$ and $Pr[Alg3(D_2) = C_p^*]$ in this case respectively. A closer look to the algorithm, reveals that for C_p^* to be an output for D_1, it must be selected by the exponential mechanism from C_M. Hence,

$$\begin{aligned} Pr_{D_1}[Alg3(D_1) = C_p^*] &= Pr_{D_1}[C_p^*|C_M] \times Pr_{D_1}[C_M] \\ &= Pr_{D_1}[C_p^*|C_M] \times \Pi_{i=1}^{n} Pr_{D_1}[C_i|C_{i-1}] \times Pr_{D_1}[C_V] \end{aligned} \quad (2.5)$$

Now, we calculate the probability of Algorithm 3 resulting in the same final answer (C_p^*) for the two databases D_1, D_2. Let $C_M = \{c_1, c_2, \cdots, c_c, \cdots, c_p\}$ be the path of contexts sampled by Algorithm 3 for database D_1. To update the C_M from any path of contexts, e.g. $\{c_1, c_2, \cdots, c_i\}$, to another path that is different from the former in only the last context, i.e. $\{c_1, c_2, \cdots, c_{i+1}\}$, two events need to take place. First, the predecessor context of c_{i+1} be selected from the i elements in the C_M, we show this context by c_j. Second, among all nodes connected to c_j, c_{i+1} be selected. By changing from D_1 to D_2, the former's probability reduces by $2\epsilon\Delta u$; the latter however maintains the same probability, as the selection among the connected nodes is random. Hence, as we also showed in proving Theorem 2.5.1, the probability of selecting the same C_p^* from a set of n samples such as C_M, is bounded by $e^{2\epsilon\Delta u}$. Therefore,

$$\frac{Pr[Alg3(D_1) = C_p^*]}{Pr[Alg3(D_2) = C_p^*]} \leq e^{2\epsilon_1 \Delta u} \quad (2.6)$$

Note that multiple paths could lead Algorithm 3 to achieve the same output for D_2 as D_1. In our proof however, we consider the worst case where there is only one path, therefore Algorithm 3 has to follow the same path for D_2 that it does for D_1. □

Theorem 2.5.4. The computation complexity of Algorithm 3 is $\mathcal{O}(t)$.

Proof. Random walk starts from a starting context for V, C_V and iteratively changes a random attribute value of C_V until it finds a matching context among the t connected contexts. Afterwards, it forms a path by repeating the same process for the last context on the chain. As searching through connected contexts to a context C is done without replacement, its complexity is $\mathcal{O}(t)$, the process is repeated for a constant number of times n

to form C_M. Ultimately, the exponential mechanism is applied to select the final candidate among n (constant number of) samples. Hence, the algorithm's complexity is $\mathcal{O}(nt+n)$. □

Exploiting the context graph in the Random Walk algorithm improves the sampling complexity from exponential in Uniform Sampling to linear. Our experimental results in Section 5.5 also affirms the superiority of the Random Walk in utility and performance over the Uniform Sampling. To improve the utility and performance even more, we alter the walk over the connected contexts from a random method to a one directed by the utility function; in other words, we can improve sampling further by taking the utility already during selection of the next vertex into account. Therefore, we explore the Depth-First and Breadth-First search algorithms for sampling.

Depth-First Search Sampling

We modify the Depth-First search (DFS) algorithm [44] to provide a differentially private version of the algorithm. We emphasize that differential privacy cannot be supported with a non-modified DFS algorithm, e.g. by perturbing its output. As described in Section 2.2, differential privacy requires the algorithm to generate any output with approximately the same (e^{ϵ} different) probability for neighboring datasets. However, as DFS is a deterministic algorithm, the probability of it generating any output is zero except for one particular value. Hence, applying DFS on a dataset D_1 might result in an output r that has probability of zero when DFS is applied on a neighboring dataset D_2; this property prevents the original DFS from supporting differential privacy. To overcome the challenge, we modify DFS as presented in the Algorithm 4. The sampling method initiates a stack with a starting context for V, C_V. Next, it searches over all connected contexts to C_V and selects a matching context by applying the exponential mechanism to the candidates and pushes the result onto the stack. The last context on the stack is the starting point for the next iteration. The iterations continue until the stack contains the desired number of samples, n.

Theorem 2.5.5. The differentially private DFS described in Algorithm 4 satisfies ($\epsilon = (2n+2)\epsilon_1$, $COE_M(.,V)$)–OCDP, where n is the total number of samples, ϵ_1 is the privacy parameter in the the exponential mechanism.

Proof. A similar reasoning to the one in Theorem 2.5.3 holds for DFS. The idea is calculating the probability of forming the same stack of nodes for neighboring datasets D_1 and D_2. However, the probability of obtaining the same stack, i.e. the corresponding probability to $\Pi_{i=1}^{n} Pr_{D_1}[C_i|C_{i-1}] \times Pr_{D_1}[C_V]$ in Equation 2.5, is not the same value for D_1 and D_2 anymore. As in differential private DFS, we apply the exponential mechanism before adding

Algorithm 4: PCOR - Depth-First Search Sampling

Input : D , $attr(R)$, V, C_V, u, ϵ

Output: C_p

$Stack \leftarrow \{C_V\}$, $Samples \leftarrow 0$, $Visited \leftarrow \emptyset$

//n is the total number of samples

while $|Visited| \leq n$ && $|Stack| > 0$ **do**

  $C \leftarrow Stack.top()$, $Visited \leftarrow Visited \cup C$, $C_{chldn} \leftarrow \emptyset$

  //Generating all children of C

  for $i \leq t$ **do**

    $C[i] \leftarrow \begin{cases} 0, & \text{If } C[i] = 1 \\ 1, & \text{If } C[i] = 0 \end{cases}$

    if $f_M(D_C, V) = true$ *and* $C \notin Visited$ **then**

      $C_{chldn} \leftarrow C_{chldn} \cup C$

    end

  end

  if *not* C_{chldn} **then**

    $Stack.pop()$

  end

  else

    //The exponential mechanism on children of C

    $C \leftarrow Exp_u^{\epsilon_1}(D, C_{chldn})$

    push C onto $Stack$

  end

end

//The Exp. mechanism for the final private answer

return $C_p \leftarrow Exp_u^{\epsilon_1}(D, Visited)$;

Comment: The algorithm starts from a starting context, C_V, adds it to the stack. It then calculates all the connected nodes to C_V that are matching contexts for V. Next, the algorithm applies the exponential mechanism to select one of the connected nodes, c_c. Afterwards, the c_c is pushed onto the stack, to be the starting point for the next search as well. This procedure continues until n samples are pushed onto the stack. The last sample in the stack is the final privacy preserving answer.

any child to the stack, each conditional probability in Equation 2.5 is $\epsilon_1 \Delta u$ different for D_1 and D_2. Since there are n nodes in the stack, the upper bound is: $\frac{Pr_{D_1}[Stack]}{Pr_{D_2}[Stack]} \leq e^{2n\epsilon_1 \Delta u}$. We also know from Equation 2.4, that the probability of selecting the same C_p^* from a set of n samples ($Stack$), is bounded by $e^{2\epsilon_1 \Delta u}$. This results in

$$\frac{Pr[Alg4(D_1) = C_p^*]}{Pr[Alg4(D_2) = C_p^*]} \leq e^{(2n+2)\epsilon_1 \Delta u} \tag{2.7}$$

□

Theorem 2.5.6. The computational complexity of Algorithm 4 is $\mathcal{O}(t)$, where t is the total number of attribute values.

Proof. The differential private DFS algorithm starts a path from a starting context for V, C_V, selects the next matching context the t connected contexts by applying the exponential mechanism to them and pushes the result onto the stack; which forms an $\mathcal{O}(2t)$ step. Then repeats the process for the last context on the stack until the stack size reaches the constant desired value, n, times. Hence, the algorithm's complexity is $\mathcal{O}(2nt)$. □

Breadth-First Search Sampling

We introduce a modified and differentially private version of the Breadth-First search (BFS) algorithm [118] as our last sampling method. The reason for necessity of modifying the algorithm to support differential privacy is the same as the one provided for Depth-First search algorithm in Section 18. Our sampling method starts from a starting context for V, C_V. It initiates the set C_M with that context, then searches over all connected contexts to C_V and adds all the matching contexts to C_M. To decide the starting point for the next iteration, it applies the exponential mechanism to the contexts in the C_M(treating C_M as a priority queue) and proceeds similarly with the selected context. The process continues until n number of samples are added to C_M.

Theorem 2.5.7. The differentially private BFS described in Algorithm 5 satisfies $((2n + 2)\epsilon_1, COE_M(., V))$–OCDP, where n is the total number of samples and ϵ_1 is the privacy parameter in the exponential mechanism.

Proof. Similar to DFS, the privacy proof of BFS follows the proof for Theorem 2.5.3. Which results in

$$\frac{Pr[Alg5(D_1) = C_p^*]}{Pr[Alg5(D_2) = C_p^*]} \leq e^{(2n+2)\epsilon_1 \Delta u} \tag{2.8}$$

□

Theorem 2.5.8. The computation complexity of Algorithm 5 is $\mathcal{O}(t)$, where t is the total number of attribute values.

Proof. BFS initiates the search from a starting context for V, C_V, searches over its connected contexts to find the matching ones to add to C_M which forms an $\mathcal{O}(t)$ step. This step is repeated for a context chosen by the exponential mechanism from C_M ($\mathcal{O}(nt)$) for the constant desired number of samples, n, times. Hence, the algorithm is $\mathcal{O}(n^2t + nt)$. □

Algorithm 5: PCOR - Breadth-First Search Sampling

Input : D , $attr(R)$, V, C_V, u, ϵ
Output: C_p

$C_M = \{C_V\}$, $Samples \leftarrow 0$, $Visited \leftarrow \emptyset$
// n is the total number of samples
while $|Visited| \leq n$ && $|C_M| > 0$ **do**
  // The exponential mechanism on C_M
  $C \leftarrow Exp_u^{\epsilon_1}(D, C_M)$
  $Visited \leftarrow Visited \cup C$, $C_M.remove(C)$
  // Adding all children of C to C_M
  for $i \leq t$ **do**
    $C[i] \leftarrow \begin{cases} 0, & \text{If } C[i] = 1 \\ 1, & \text{If } C[i] = 0 \end{cases}$
    if $f_M(D_C, V) = true$ *and* $C \notin Visited$ **then**
      $C_M \leftarrow C_M \cup C$
    end
  end
end
// The Exp. mechanism for the final private answer
return $C_p \leftarrow Exp_u^{\epsilon_1}(D, Visited)$;
Comment: The algorithm starts from a starting context C_V, and calculates all the connected contexts to C_V, adds the matching ones to the priority queue C_M. It continues by applying the the exponential mechanism to the contexts in C_M, and iterates the procedure until it finds all n samples.

2.6 Experiments

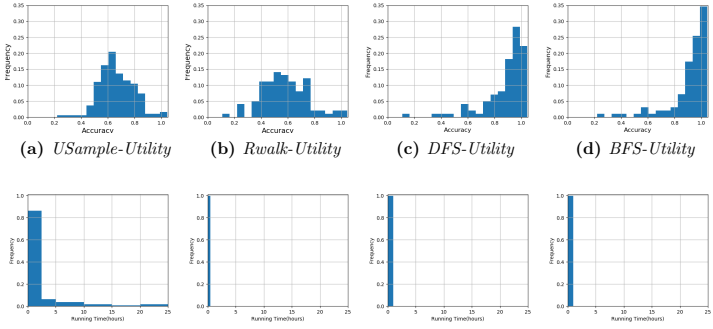
We run five groups of experiments to support our choices for the parameters in our PCOR design varying: i) Sampling algorithm, ii) Utility function, iii) Outlier detection algorithms, iv) Privacy parameter ϵ, and v) Group privacy limit in OCDP.

2.6.1 Datasets

We evaluate PCOR over two datasets. The first one is the Ontario's public sector salary dataset. The province of Ontario (Canada) annually publishes a list of public employees who earn $100,000 and above [2]. We filter the dataset to obtain the information of 51000 employees with income higher than 100K in Ontario, the attributes are $attr(R) = \{Jobtitle, Employer, Year, Salary\}$, with domain sizes: $|Jobtitle| = 9$, $|Employer| = 8$, and $|Year| = 8$. The second dataset we use is homicide reports in the United States [1], compiled and made available by the Murder Accountability Project. The data is gathered from multiple agencies and includes the age, sex, ethnicity of victims and perpetrators, their relationship, and the weapon used. We filter the dataset to obtain 110,000 records. The attributes of the data are $attr(R) = \{AgencyType, State, Weapon, VictimAge\}$. The categorical attributes have domains of size 4, 6, and 6 respectively. Since our results for both datasets follow the same pattern, we mainly provide the experiment results on the salary dataset.

2.6.2 Evaluation Setup

We ran our experiments on a machine with 1 TB RAM, 132 cores, Intel(R) Xeon(R) CPU E7-8870 2.40GHz specifications. We repeated each experiment 200 times to achieve reliable results for PCORs and fair comparison of the algorithms in utility and performance. The performance is measured as the runtime of the algorithm, in two metrics: i) The range of shortest to longest runtime, and ii) The average runtime. The utility is measured as the proportion of the utility of the PCOR's output to the maximum utility. The maximum utility for each data point (outlier), is gained from a reference file. The reference file contains all possible contexts in $attr(R)$ accompanied with their associated utility, and the list of outliers for each context. Generating this file, for our data set of 51,000 records utilizing an optimized program took three days to complete on the machine mentioned earlier. This is also the running time of the direct/formulaic application of exponential mechanism in OCDP (described in Section 2.4). From the reference file, we find all matching contexts



(a) *USample-Utility* **(b)** *Rwalk-Utility* **(c)** *DFS-Utility* **(d)** *BFS-Utility*

(e) *USample-Time* **(f)** *Rwalk-Time* **(g)** *DFS-Time* **(h)** *BFS-Time*

Figure 2.1: Utility and Performance of PCORs for different sampling candidates, utility function outputs context population size, outlier detection algorithm is LOF, and $\epsilon = 0.2$ (a,e) Uniform Sampling, (b,f) Random walk, (c,g) DFS, (d,h) BFS; where (a), (b), (c) , (d) represent utility and (e), (f), (g) ,(h) demonstrate performance in running time

for outlier V and find the one with maximum utility to divide the utility results of the PCOR by. For the 200 utility results, we provide the mean and the 90% confidence interval (CI) for each PCOR. We run five groups of experiments to answer the following questions: 1) what is the best (private, accurate, fast) sampling algorithm for PCOR? 2) Is PCOR flexible with embedding other utility functions? 3) Is PCOR flexible with embedding other outlier detection algorithms? We evaluate PCOR over three deterministic outlier detection algorithms, each from a main category described in Section 2.1, namely: Grubbs [73], Local Outlier Factor (LOF) [28], and Histogram [4]. 4) How does changing each one of privacy, utility and performance parameters affect the other two. 5) How does changing a (group of) record affect the set of possible outcomes, COE_M?

Table 2.2: Sampling Methods Comparison - Performance.

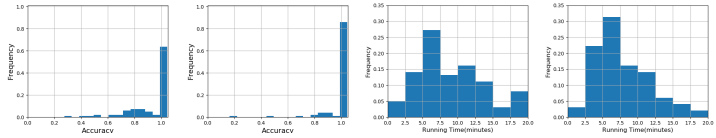
Algorithm	T_{min}	T_{max}	T_{avg}	ϵ	Outlier
Uniform	7m	24h	97m	0.2	LOF
Random Walk	15s	109s	51s	0.2	LOF
DFS	8m	80m	40m	0.2	LOF
BFS	6m	61m	37m	0.2	LOF

Table 2.3: Sampling Methods Comparison - Utility.

Algorithm	Utility	CI	ϵ	Outlier
Uniform	0.65	(0.64, 0.67)	0.2	LOF
Random Walk	0.57	(0.55, 0.60)	0.2	LOF
DFS	0.88	(0.85, 0.90)	0.2	LOF
BFS	0.90	(0.88, 0.93)	0.2	LOF

2.6.3 Choosing a Sampling algorithm for PCOR

We show that using Breadth-First Search algorithm for private sampling results in the highest utility and performance for PCOR. We evaluate four sampling algorithms: i) Uniform Sampling, ii) Random Walk, iii) Breadth-First Search, and iv) Depth-First Search. The results are depicted in Figure 2.1. We generate sets of length 50 samples and set $\epsilon = 0.2$ as the total differential privacy budget. This translates to $\epsilon_1 \approx 0.002$ in Depth-First Search and Breadth-First Search as shown in Equations 2.7 and 2.8, and $\epsilon_1 = 0.1$ in Uniform Sampling and Random Walk as shown in Equations 2.4 and 2.6 respectively. The run times of the sampling algorithms are shown in Figure 2.1[e-h] and summarized in Table 2.2. Utility is measured as the ratio of the population size of the private answer to that of the maximum context's for an outlier V. Utility comparison results are shown in Figures 2.1[a-d] and summarized in Table 2.3. The Random Walk improves the performance of the uniform sampling significantly, but it sacrifices utility. The BFS and DFS algorithms perform close to each other in running time and utility, they make up for the utility loss in Random Walk by a considerable amount, but require a longer runtime. Their advantage over Uniform sampling in both utility and performance is noticeable. The slight advantage of BFS over DFS in utility and runtime, also holds for PCOR with a different utility function (Section 6.3). Therefore, we select BFS as PCOR's final sampling method.



(a) *DFS-Utility* **(b)** *BFS-Utility* **(c)** *DFS-Time* **(d)** *BFS-Time*

Figure 2.2: Utility and Performance PCORS with different sampling candidates, utility is measured by the overlap with C_V, LOF is the outlier detection, and $\epsilon = 0.1$ (a,c) DFS, (b,d) BFS; where (a), (b) represent utility and (c), (d) demonstrate performance in running time

Table 2.4: Intersection Overlap Utility - Performance.

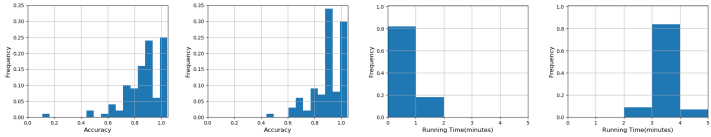
Algorithm	T_{min}	T_{max}	T_{avg}	ϵ	Outlier
DFS	3m	47m	19m	0.2	LOF
BFS	5m	48m	20m	0.2	LOF

2.6.4 PCOR and Other Utility Functions

We show the adaptability of PCOR with any utility function by providing the results of embedding another utility function than the population size in Tables 2.4 and 2.5. The utility function in this experiment calculates the population intersection of the private context and the starting context, C_V. The utility and performance evaluations confirm that BFS is a superior candidate over the DFS algorithm.

Table 2.5: Intersection Overlap Utility - Utility.

Algorithm	Utility	CI	ϵ	Outlier
DFS	0.88	(0.86, 0.91)	0.2	LOF
BFS	0.97	(0.95, 0.98)	0.2	LOF



(a) *Grubbs-Utility* **(b)** *Histogram-Utility* **(c)** *Grubbs-Time* **(d)** *Histogram-Time*

Figure 2.3: Utility and Performance of PCORs with Grubbs and Histogram outlier detection algorithms using BFS sampling over utility of context population size for $\epsilon = 0.1$ (a,c) Grubbs, (b,d) Histogram; where (a), (b) represent utility and (c), (d) demonstrate performance in running time

Table 2.6: Outlier Detection Algorithms - Performance.

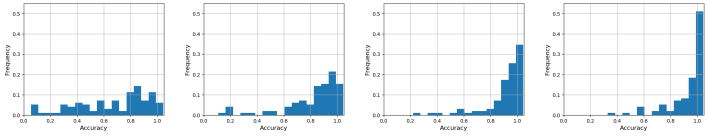
Algorithm	T_{min}	T_{max}	T_{avg}	ϵ	Sampling
Grubbs	0.5m	1m	0.8m	0.2	BFS
Histogram	2m	4m	3.4m	0.2	BFS

2.6.5 PCOR and Outlier Detection Algorithms

We show that PCOR can be successfully used with various outlier detection algorithm. We explore two more outlier detection algorithms in our PCOR design: Grubbs, and Histogram methods. The results are shown in Figure 2.2 and summarized in Table 2.6 and Table 2.7. We filter the original dataset to obtain a subset of 11000 records with 14 attributes values in total, and use BFS as the sampling algorithms in the PCOR design. In this set of experiments, the number of samples is $n = 50$ and the privacy parameter is $\epsilon = 0.2$. For Histogram method, we bin the samples corresponding to a context C to $\sqrt{|D_C|}$ bins, where $|D_C|$ is the size of the population covered by C. The bins with less frequency than

Table 2.7: Outlier Detection Algorithms - Utility.

Algorithm	Utility	CI	ϵ	Sampling
Grubbs	0.86	(0.84, 0.89)	0.2	BFS
Histogram	0.89	(0.87, 0.91)	0.2	BFS



(a) $\epsilon = 0.05$, *Utility* **(b)** $\epsilon = 0.1$, *Utility* **(c)** $\epsilon = 0.2$, *Time* **(d)** $\epsilon = 0.4$, *Time*

Figure 2.4: The effect of privacy parameter over Utility in PCOR with BFS sampling and LOF outlier detection(a) $\epsilon = 0.05$, (b) $\epsilon = 0.1$, (c) $\epsilon = 0.2$, (d) $\epsilon = 0.4$

Table 2.8: Effect of privacy parameter on performance.

ϵ	T_{min}	T_{max}	T_{avg}	Sampling	Outlier
0.05	2m	29m	15m	BFS	LOF
0.1	2m	29m	16m	BFS	LOF
0.2	3m	30m	17m	BFS	LOF
0.4	3m	30m	17m	BFS	LOF

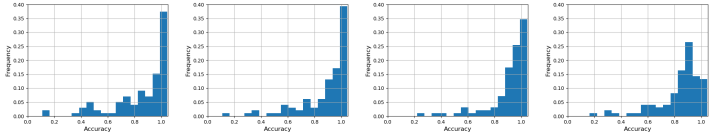
$2.5 \times 10^{-3}|D_C|$ are labeled as outliers. The results confirm the compatibility of our PCOR design with various outlier detection algorithms. Furthermore, the success of applying BFS on the algorithms implies that the locality (discussed in Section 5.2) exists in all evaluated outlier detection algorithms.

2.6.6 Privacy, Utility, Performance Trade-off

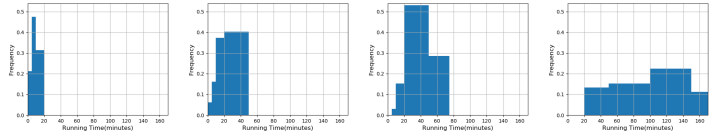
We show the trade-off between privacy, utility and performance for our final candidate, BFS sampling algorithm, by changing the privacy parameter ϵ for $n = 50$ number of samples. The results are shown in Figure 2.4 and are summarized in Table 2.8 and Table 2.9. Increasing the ϵ from 0.05 to 0.4 generally increases the utility, however the parameter $\epsilon = 0.2$ is the optimum value and the utility does not increase significantly afterwards. Moreover, changing the privacy parameter does not impose a notable effect on the algorithm runtime. We also investigate the effect of changing the number of samples while the privacy parameter is fixed $\epsilon = 0.2$. The results are shown in Figure 2.5 and summarized in

Table 2.9: Effect of privacy parameter on utility.

ϵ	Utility	CI	Sampling	Outlier
0.05	0.67	(0.62, 0.71)	BFS	LOF
0.1	0.82	(0.78, 0.85)	BFS	LOF
0.2	0.90	(0.88, 0.93)	BFS	LOF
0.4	0.92	(0.90, 0.94)	BFS	LOF



(a) $n = 25$, *Utility* **(b)** $n = 50$, *Utility* **(c)** $n = 100$, *Utility* **(d)** $n = 200$, *Utility*



(e) $n = 25$, *Time* **(f)** $n = 50$, *Time* **(g)** $n = 100$, *Time* **(h)** $n = 200$, *Time*

Figure 2.5: The effect of number of samples over utility and performance in PCOR with BFS sampling and LOF outlier detection for $\epsilon = 0.2$ (a,e) $n = 25$, (b,f) $n = 50$, (c,g) $n = 100$, (d,h) $n = 200$

Table 2.10: Effect of # of samples on performance.

# Samples	T_{min}	T_{max}	T_{avg}	Sampling	Outlier
25	1m	14m	7m	BFS	LOF
50	3m	29m	16m	BFS	LOF
100	6m	61m	37m	BFS	LOF
200	21m	163m	99m	BFS	LOF

Table 2.11: Effect of # of samples on utility.

# Samples	Utility	CI	Sampling	Outlier
25	0.85	(0.81, 0.88)	BFS	LOF
50	0.88	(0.85, 0.91)	BFS	LOF
100	0.90	(0.88, 0.93)	BFS	LOF
200	0.84	(0.81, 0.87)	BFS	LOF

Tables 2.10 and 2.11. As we showed in the proof of Theorem 2.5.8, the only parameters in the computation complexity of BFS are: i) the total number of attribute values t, and ii) the number of samples n – which we referred to as a constant number in the proof. Hence, when t is fixed, n determines the performance of PCOR. Increasing the number of samples from 25 to 100 decreases the performance, but increases the utility. This increment does not continue for for $n = 200$. Recall from Theorem 2.5.7, that keeping a fixed ϵ in PCOR while increasing n, requires using smaller ϵ_1's in algorithm design, this can cancel out the positive affect of higher n's on utility.

Advanced Composition

Our experiment results show that PCOR provides good privacy guarantee ($\epsilon = 0.2$) while achieving high utility values ($\sim 90\%$), by using the sequential differential privacy composition. We emphasize that this privacy parameter can be improved even further by applying improved composition techniques for Exponential mechanism [47].

2.6.7 Context Match and Group Privacy

Our reduced salary dataset consists of 11,000 records with 3 attributes that include 14 attribute values in total. The homicide dataset consists of 28,000 records with three attributes that include 12 attribute values in total. We chose small datasets to run several experiments in a reasonable amount of time. We are aware that our results do not benefit from this choice, as changing a single record in a small dataset, more strongly affects the set of outliers than in a large dataset. We repeated each experiment for 100 random outliers and measured the contexts set match of the original dataset and its neighboring datasets. Recall from Section 2.3.1 that the experiments in this section are to observe: i) to what extent the assumption $COE_M(D_1, V) = COE_M(D_2, V)$ in OCDP matches the

Table 2.12: *COE* Match - Salary dataset

Algorithm	$\Delta D = 1$	$\Delta D = 5$	$\Delta D = 10$	$\Delta D = 25$
Grubbs	99.8%	96.9%	94.5%	91.9%
LOF	89%	87.9%	86.7%	85.7%
Histogram	95.5%	82.1%	70.8%	58.8%

Table 2.13: *COE* Match - Homicide dataset

Algorithm	$\Delta D = 1$	$\Delta D = 5$	$\Delta D = 10$	$\Delta D = 25$
Grubbs	100%	100%	100%	97.8%
LOF	99.9%	99.5%	98.7%	97.7%
Histogram	98.5%	85.2%	69.3%	53.3%

results of outlier detection algorithms in practice, ii) what are the effects if this assumption does not hold and whether it results in a privacy sacrifice. For both mentioned datasets and all three outlier detection algorithm, we recorded the COE_M results of the original dataset and its 50 randomly chosen neighboring datasets. The results for the objective i) are summarized in Tables 2.12 - 2.13. These tables show the results for neighboring datasets that are different from the original dataset (ΔD) in 1, 5, 10, and 25 records, for the salary and the homicide dataset respectively. For objective ii) and to evaluate the privacy in the case of non-matching *COE*'s of the neighboring datasets, we ran another set of experiments. Consider the contexts in the intersection of non-equal sets $COE_M(D_1, V)$ and $COE_M(D_2, V)$, where $|D_1 - D_2| = 1$. We measured the maximum ratio of the probability of selecting any of these context for D_1 to the probability of selecting the same context for D_2. This experiment was repeated for 200 outlier samples for three outlier detection algorithm over the Salary dataset. All the experiments confirm this ratio is below e^{ϵ} (recall the differential privacy requirement in Equation 1.1) for $\epsilon = 0.2$ that is the privacy budget in our experiments that we presented earlier in this section. In other words, for non-matching *COE*'s of neighboring datasets, we found no instance that violates the privacy bound in (unconstrained) $\epsilon-$differential privacy.

2.7 Related Work

The attempts to provide privacy in outlier detection, initiated with using secure computation methods for distributed [182, 48] or non-distributed [12] data. In a different approach, Bhaduri et al. [21] use nonlinear data distortion transformation and show how it can be useful for privacy preserving anomaly detection from sensitive datasets. Mehnaz and Bertino [137] propose a framework that enables efficient anomaly detection on encrypted data, to preserve privacy during the computation. Gehrke et al. [63] introduce *crowd blending privacy*, in which for every individual in the database, either the individual blends in to a crowd of k people in the database with respect to the privacy mechanism, or the mechanism ignores the individual's data. Inspired by crowd blending [63] and providing different levels of privacy [66], Lui and Pass [129] proposed *tailored differential privacy* for protecting outliers, in which the privacy parameter for an individual is determined by the "outlierness" of the individual's data in the dataset. Böhler et al. [25] evaluate the opposite by relaxing the differential privacy definition and granting outliers *less* protection, as they consider the outliers as "faulty systems or sensors one need to detect". In another attempt in private outlier investigation, Nissim et al. [148] use differential privacy to locate a small outlier cluster privately. Okada et al. in [151] break the outlier analysis to two tasks: i) counting outliers in a given subspace and ii) discovering sub-spaces containing many outliers, under the constraints of differential privacy. They show their method achieves better utility compared to the original global sensitivity based methods. Nonetheless, our work is the first to investigate the privacy of the individuals in the outlier's context.

2.8 Conclusion

The revealed context in contextual outlier release leaks information about the individual records in the dataset. We address this privacy violation in this chapter and propose techniques for a relaxed notion of differential privacy to provide a private contextual outlier release (PCOR) and resolve the issue. However, the differential privacy solution in its formulaic application suffers from weak performance, impeding its usage in practice. To achieve efficiency in PCOR, we propose utilizing a sampling layer in the design. We present differentially private graph search algorithms, first time in the literature, and use them for sampling. We prove PCOR with the presented sampling methods supports worthwhile levels of differential privacy, while providing the desired utility and performance. We articulate and demonstrate empirically that PCOR design is compatible with any utility function in outlier detection. Our results also indicate that the relaxation required for providing

privacy in PCOR is a not a strong requirement, and is satisfied in most cases in practice. Furthermore, we show that PCOR is generic and fits any outlier detection algorithm.

Chapter 3

On Privacy and Confidentiality of Communications in Organizational Graphs

3.1 Introduction

A number of applications in natural language understanding rely on language models [36, 9]. To enable such models it is necessary to process training data that best represents the target application. Such tasks become especially sensitive in the setting of organizational communication, where organizations, individuals, or communities may share secret data, and preserving confidentiality is of utmost importance. Huseyin A. Inan, Marcello Hasegawa, Robert Sim and I looked into the privacy issues in the organizational communications [166]. These communications often presents a complex underlying structure of interactions well modeled by a social graph. Data privacy in graphs and social graphs have been addressed by a number of previous works, for example [192, 142, 41, 103, 60, 198, 101], where most of the works based on differentially private approaches model individuals as nodes and exchanged messages as edges. These proposed methods provide either node level privacy guarantees or edge level privacy guarantees. In the context of organizational communication we define node level guarantees as individual privacy and edge level guarantees as confidentiality. That is, confidentiality involves protecting information that is shared between two or more individuals in the organization. In this work we set aside questions of individual privacy and examine problems in ensuring confidentiality in organizational communication. The goal is to enable the production of ML models for an enterprise, without compromising

potentially sensitive business secrets. It is worth noting that a limitation of node level privacy is that in an organizational context, this approach can produce models with lower than optimal utility. To understand this, suppose the CEO of a company communicates regularly with messages addressed to all employees. From the organization's perspective this information is public to everyone, and yet a differentially private mechanism acting at a node level will be forced to mask out the presence of these messages as they originate from a single node. This would be necessary to satisfy the requirement that a differentially private model should be invariant to the presence or absence of any single node. Instead we consider a model that affords edge-level privacy. In this case the CEO can broadcast to all employees, and enable this correspondence to be included in training of the model, since the information is replicated across many edges, but she can still protect private 1:1 correspondence with her CFO, for example, thus preserving confidentiality.

Technical Challenges

However, the situation is not so straightforward. In the case of social graphs, often the properties of an edge can be inferred by the properties of other nearby edges. Such a case does not fall under the framework of differential privacy, where the theoretical guarantees are based on presence/absence of individual elements, without taking into account any effect on other elements. One approach to address this problem is via group differential privacy. Group privacy assumes that all edges in a group of participants are fully correlated, and queries against the graph must be invariant to the presence or absence of the entire group. This framing can significantly impact the accuracy of the model or query by being over-protective of edge-edge relationships. Furthermore, it requires an explicit description of what constitutes a group in the organization. We address the issues presented by edge correlation by employing a generalized version of differential privacy called Pufferfish [171]. In Pufferfish a set $\mathcal{S}_{\text{pairs}} \subseteq \mathcal{S} \times \mathcal{S}$ of complementary secret pairs is defined, and privacy is provided by ensuring the secret pairs are indistinguishable, for any data distribution (capturing the correlation among the records) known to the adversary. Through this requirement the correlation problem can be addressed while allowing utility to be preserved. In addition, the non-independence between edges also affords us a simple model for confidentiality, namely that information passing between neighboring edges is more likely to be confidential than information that is randomly distributed in the social graph. From this perspective, we propose a privacy model that accounts for information dependence between edges in the graph and define a notion of what constitutes an edge's neighborhood. In this work we are limited to L-Lipschitz queries which are sufficiently broad to cover for counting and frequency queries.

For the purposes of this work we adapt the *Attribute Disclosure Attack* defined in [17]:

Definition 3.1.1 (Attribute Disclosure Attack)**.** Given T=(G,A,B), which is a snapshot of an organization with a social graph $G = (V, E)$, where V is the set of individuals and E demonstrates the correspondence between them, correspondent behavior A and attribute information B, the attribute disclosure attack is used to infer the attributes a_e for all $e \in E_t$ where E_t is a list of targeted edges. For each $e \in E_t$, we have information about the correspondence between the individuals linked by e.

This definition deviates from that proposed be Beigi et al. [17] in that it focuses on disclosing information about correspondence between users. That is, the goal of the attack is to leak information about what is communicated along the edges of the graph.

3.1.1 Our Contributions

The contributions of this work are as follows, it:

1. Presents a complete formulation of *confidentiality* protection in closed social networks such as an enterprise or organization, including accounting for neighborhood correlation,
2. Models predicting neighborhood correlations assuming varying degrees of attacker knowledge,
3. Shows the gap between two extreme measures of privacy, record-level privacy and group privacy, over two language tasks, and
4. Provides empirical results of applying our work to statistical query tasks [69, 188] in a real enterprise graph.

3.1.2 Chapter Organization

The rest of this chapter is organized as follows: In Section 3.2 we describe the background for group privacy, Pufferfish privacy and a mechanism for achieving this privacy. In Section 3.3 we provide the organizational model and confidentiality problem definitions. We introduce our correlation model and our privacy mechanism in Section 3.4. In Section 3.5, we provide the results of applying the privacy mechanism to two query tasks. We situate our work in the current body of research in Section 3.6, and in Section 3.7 we conclude this work.

3.2 Preliminaries

In this section, we provide the notations and definitions for the concepts used throughout the chapter.

3.2.1 Group Differential Privacy

Note that in the definition of ϵ-differential privacy, the guarantee holds for a single data entry. However, following from the composability property of differential privacy [135], the setting can be extended to multiple data entries. If the goal is to protect a group, this can be achieved by setting ϵ to ϵ/k for any $k \in \mathbb{N}$ where k represents the size of the group. In this case, all groups of size k are ϵ-differentially private protected. We summarize this by formally defining group differential privacy [51] in the following theorem.

Theorem 3.2.1 (Group Differential Privacy)**.** Any $\epsilon-$differentially private mechanism $\mathcal{M}$ is $k\epsilon-$differentially private for groups of size k. That is, for all $||D_1 - D_2||_1 \leq k$ and all $S \subseteq Range(M)$:

$$Pr[\mathcal{M}(D_1) \in S] \leq exp(k\epsilon)Pr[\mathcal{M}(D_2) \in S] \tag{3.1}$$

We point out that the scaling of the noise is inversely proportional to the privacy budget ϵ. Therefore, setting ϵ to ϵ/k will in turn change the noise level from $\Delta f/\epsilon$ to $k \cdot \Delta f/\epsilon$, which may significantly decrease the utility of the query. However, this is the price to pay to obtain stronger privacy guarantees with group differential privacy.

3.2.2 Pufferfish Privacy

Differential privacy provides robust guarantees for a wide range of database queries. For our scenario, it is useful to consider Pufferfish privacy [108], which is a Bayesian privacy framework providing rigorous privacy guarantees against many types of attackers. An advantage of the Pufferfish framework is that a domain expert can develop rigorous privacy definitions for their data sharing needs without holding an expertise in privacy. This is achieved by specifying three components in the Pufferfish privacy framework: a set $\mathcal{S}$ of potential secrets, a set $\mathcal{S}_{\text{pairs}} \subseteq \mathcal{S} \times \mathcal{S}$ of discriminative secret pairs, and a collection of data distributions Θ. The Pufferfish framework provides a rich class of privacy definitions based on the components specified by a domain expert. We formally define the framework in the following based on [108].

Definition 3.2.1 (Pufferfish Privacy)**.** A randomized algorithm $\mathcal{M}$ is said to provide ϵ-Pufferfish privacy for a domain $(\mathcal{S}, \mathcal{S}_{\text{pairs}}, \Theta)$ if for all distributions $\theta \in \Theta$, for all secret pairs $(s_i, s_j) \in \mathcal{S}_{\text{pairs}}$, and for all possible outputs $w \in \text{Range}(\mathcal{M})$ it satisfies

$$\left| \frac{\Pr_{\mathcal{M},\theta}(\mathcal{M}(\mathcal{D}) = w | s_i, \theta)}{\Pr_{\mathcal{M},\theta}(\mathcal{M}(\mathcal{D}) = w | s_j, \theta)} \right| \leq \exp(\epsilon)$$

where $\mathcal{D}$ is drawn from the distribution θ and s_i and s_j are such that $\Pr(s_i|\theta) \neq 0$ and $\Pr(s_j|\theta) \neq 0$.

We note that there is an additional source of randomness in the definition of Pufferfish privacy, the dataset $\mathcal{D}$ is itself a random variable that is drawn from a distribution $\theta \in \Theta$. A domain expert constructs the set $\mathcal{S}$ for the potential secrets that are desired to be hidden (e.g. private data of an individual). $\mathcal{S}_{\text{pairs}}$ is simply the pair of such potential secrets that we would like to guarantee are indistinguishable in evaluating $\mathcal{M}$. Finally, Θ is a collection of distributions where each probability distribution $\theta \in \Theta$ corresponds to an attacker to be protected against. Θ can be selected based on the fine grain of how data can be plausibly generated and it also reflects the attackers' beliefs in how the data were generated (incorporating any background knowledge and side information). The whole process gives the domain expert flexibility to customize privacy to the specific set of secrets and data generation scenarios that are typical in their domain. We further point out that Pufferfish privacy provides a general framework in the sense that it covers ϵ-differential privacy as an instantiation for a particular choice of domain $(\mathcal{S}, \mathcal{S}_{\text{pairs}}, \Theta)$ (see Theorem 6.1 in [108]).

3.2.3 Wasserstein Mechanism

While there is no efficient general mechanism that applies to any Pufferfish instantiation, there are a number of mechanisms for specific Pufferfish instantiations [108, 87]. For general Pufferfish instantiation, Song et al. [171] introduce a mechanism that achieves Pufferfish privacy, but does not satisfy efficiency in its original form. We introduce their base mechanism here. Later in Section 3.4 of this chapter, we present our adjustments to their mechanism that makes it efficient to utilize for our use case of enterprise communications. The main idea of the mechanism in [171] is similar to the Laplace mechanism in differential privacy. Instead of adding noise based on the global sensitivity Δf in differential privacy, Song et al. use the distributions $\Pr(f(\mathcal{D})|s_i, \theta)$ and $\Pr(f(\mathcal{D})|s_j, \theta)$ in the Pufferfish framework, propose a metric quantifying the worst case distance between these two distributions, and inject noise proportional to this distance. They find that the ∞-Wasserstein distance is the right choice for this purpose.

Definition 3.2.2 (∞-Wasserstein distance)**.** Let μ, ν be two probability distributions on $\mathbb{R}$ and $\tau(\mu, \nu)$ denote the set of all joint distributions with marginals μ and ν. The ∞-Wasserstein distance between μ and ν is defined as

$$W_\infty(\mu, \nu) = \inf_{\gamma \in \tau(\mu,\nu)} \max_{(x,y) \in \text{support}(\gamma)} \mid x - y \mid .$$

Intuitively, W_∞ measures the maximal distance that any probability mass moves while transforming μ to ν in the most optimal way possible. W_∞ is related to the well-known Earth Mover's Distance in that it accounts for the maximal shift in probability over the domain of τ but not the amount of mass in the shift [92]. Based on the ∞-Wasserstein distance, the Wasserstein mechanism calculates the maximum over $(s_i, s_j) \in \mathcal{S}_{\text{pairs}}$ and $\theta \in \Theta$, analogous to Δf, and applies the Laplace noise proportional to the maximum ∞-Wasserstein distance. It is proven in [171] (see Theorem 3.2) that this mechanism utilitys ϵ-Pufferfish privacy.

Theorem 3.2.2 (Wasserstein mechanism)**.** Let $(\mathcal{S}, \mathcal{S}_{\text{pairs}}, \Theta)$ be a domain. For any function $f : \mathcal{D} \to \mathbb{R}$ the randomized mechanism $\mathcal{M}$

$$\mathcal{M}(\mathcal{D}) = f(\mathcal{D}) + \text{Laplace}(0, W/\epsilon)$$

where
$W = \sup_{(s_i,s_j) \in \mathcal{S}_{\text{pairs}}, \theta \in \Theta} W_\infty(\mu_{i\theta}, \nu_{j\theta})$ for $\mu_{i,\theta} = \Pr(f(\mathcal{D})|s_i, \theta)$ and $\nu_{j,\theta} = \Pr(f(\mathcal{D})|s_j, \theta)$ satisfies ϵ-Pufferfish privacy.

3.2.4 Markov Quilt Mechanism

The Wasserstein mechanism can be quite expensive in terms of computational complexity, as it requires modeling the effects of varying all complementary secret pairs on the function f. Song et al. [171] introduce the Markov Quilt mechanism for the special case where the dependence inside a dataset can be described by a Bayesian network, which fits to our setting of interest. In the case where the dependence is most effective in the "local" neighborhood, the amount of noise can be calibrated with respect to the size of this neighborhood. To this end, max-influence of a variable $\mathcal{D}_i$ on a set of variables $\mathcal{D}_A$ under a distribution class Θ is defined as

$$e_\Theta(\mathcal{D}_A|\mathcal{D}_i) = \sup_{\theta \in \Theta} \max_{a,b \in \mathcal{X}} \max_{d_A \in \mathcal{X}^{|\mathcal{D}_A|}} \log \frac{\Pr(\mathcal{D}_A = d_A|\mathcal{D}_i = a, \theta)}{\Pr(\mathcal{D}_A = d_A|\mathcal{D}_i = b, \theta)}$$

where $\mathcal{X}$ denotes the range of each $\mathcal{D}_i$.

In terms of privacy it is an advantage that the dependence stays as "local" as possible if one can find a large set $\mathcal{D}_A$ such that $\mathcal{D}_i$ has low max-influence on $\mathcal{D}_A$. Especially if one can claim certain conditional independence from a variable towards some part of the dataset it can also simplify the calibration of the noise. The following notion is helpful to show what is described here.

Definition 3.2.3 (Markov Quilt)**.** A set of variables $\mathcal{D}_Q$ in a dataset is a Markov Quilt for a variable $\mathcal{D}_i$ if there exists a set $\mathcal{D}_i \in \mathcal{D}_N$ such that $\mathcal{D} = \mathcal{D}_N \cup \mathcal{D}_Q \cup \mathcal{D}_R$ and $\mathcal{D}_i$ is conditionally independent from $\mathcal{D}_R$ given $\mathcal{D}_Q$, i.e. $\Pr(\mathcal{D}_R|\mathcal{D}_Q, \mathcal{D}_i) = \Pr(\mathcal{D}_R|\mathcal{D}_Q)$.

In this formulation, [171] choose the subscripts N and R to represent "nearby" and "remote" nodes in the Bayesian network, respectively, with the Q (quilt) nodes separating them and establishing conditional independence. Based on this notion, [171] introduces the Markov Quilt mechanism that protects a variable $\mathcal{D}_i$ by adding Laplace noise to a L-Lipschitz query f with scale parameter $L \times |\mathcal{D}_N|/(\epsilon - \delta)$ where δ is an upper bound on the max-influence of $\mathcal{D}_i$ on $\mathcal{D}_Q$. We note that the effect of $\mathcal{D}_i$ is obscured with noise whose amount is based on the cardinality of the local variables ($|\mathcal{D}_N|$) and a correction term to account for the effect of the distant variables (δ). Naturally, the privacy of all variables can be protected by adding noise with the maximum scale parameter over all variables $\mathcal{D}_i \in \mathcal{D}$. It is shown in [171] (see Theorem 4.3) that this mechanism utilitys ϵ-Pufferfish privacy. It is also proven that Markov Quilt mechanism satisfies sequential composition (see Theorem 4.4 in [171]).

Theorem 3.2.3 (Markov Quilt Mechanism)**.** Let $(\mathcal{S}, \mathcal{S}_{\text{pairs}}, \Theta)$ be a domain. For any L-Lipschitz function f if each $\mathcal{D}_i \in \mathcal{D}$ has the trivial quilt $\mathcal{D}_Q = \emptyset$ (with $\mathcal{D}_N = \mathcal{D}$, $\mathcal{D}_R = \emptyset$), then the Markov Quilt Mechanism provides ϵ-Pufferfish privacy.

3.3 Problem Definition

With preliminaries defined, we model the organizational communication through a graph structure. Having the communication modeled, we then describe the confidentiality requirements for the model.

3.3.1 Organizational Model

We model the network of organizational communications by a graph with the following properties:

- The graph displays the communications in the organization.
- Nodes are the individuals of the organization.
- There is an edge between two nodes if one of the nodes communicated with (e.g. sent an email to) the other one; the graph is not directed.
- Edges may be labeled with a set of zero or more properties. For example, an edge may be labeled with the token "acquisition" if the correspondents discussed the topic "acquisition".
- The target application is to perform queries over the graph to measure statistics of edge properties.

In principle, "queries over the graph to measure statistics of edge properties" may imply training a language model or performing similar natural language tasks over the graph. For the purposes of our work we focus on a simpler task, which is to safely release a set of edge properties present in the graph. For language tasks, this can be defined as extracting common n-grams from correspondence. Other application scenarios have examined popular item sampling. This problem has been explored in related work like differentially private set union (DPSU) [69], and top-k item selection [49].

3.3.2 Confidentiality Requirements

In order to achieve our goal of organizational confidentiality, we impose the following additional requirements:

- The privacy mechanism should protect edges (provide edge-level privacy),
- the mechanism should account for correlation between neighboring edges,
- the mechanism should protect against changes to edge properties, but not to changes in graph structure (the graph edges are considered invariant and public, whereas the properties of the edges are private[1]),
- the graph structure is known to attackers, and

[1]For example, an organization's leadership structure is usually public knowledge, and individuals with internal access can usually determine who reports to who and infer common lines of communication.

- attackers may have access to a data generation model θ that can predict an edge's properties, given its neighbors'.

Note that the data generation model θ assumes that neighboring edges influence one-another but non-neighboring edges don't. That is, an edge's properties are conditionally independent of the rest of the graph, given its neighbors' properties. This is equivalent to expressing the graph as a Markov random field. In practical scenarios, edge correlation may be effective beyond the neighborhood of an edge. However, we believe our conditional independence assumption is a reasonable approximation. The reader will note that this problem framing lends itself well to Pufferfish privacy: the presence or absence of a property on an edge can be framed as a pair of complementary secrets s_i^0 and s_i^1 respectively for all edges X_i in the graph. In the Pufferfish instantiation this leads to having the set of secrets $\mathcal{S} = \{s_i^0, s_i^1 : \text{for all edges } i \text{ in the graph}\}$, so the status of the corresponding property of each edge is a secret. The set of secret pairs becomes $\mathcal{S}_{\text{pairs}} = \{(s_i^0, s_i^1) : \text{for all edges } i \text{ in the graph}\}$ as we desire that the adversary cannot tell if each edge has the property of interest or not. Finally, attackers have access to generating distributions in Θ describing how properties may be defined on the graph, and in particular how they may be correlated.

In the following section we describe how we can leverage Pufferfish privacy to protect edge properties, while accounting for neighborhood correlation, and subsequently we present empirical results on a language task applied to a real-world organizational graph.

3.4 Mechanism Design

We consider the graph representation of the organizational communications, consisting of nodes for individuals and edges for the correspondences among them.

3.4.1 Neighborhood model for correlation

In our neighborhood model, we capture the correlation among the *adjacent edges* as shown in Figure 3.1. This is one choice of modeling correlation for the communication between nodes and one can think of other models such as the clique model in the flu status over social network example of [171] where the network is a union of cliques and each clique has a dependency among its nodes. This may be appealing for the organization communications considering each group in the organization as a clique. However, in practice it is seldom the case that every pair of individuals in a particular group has a communication link

between them, and therefore the actual cliques in the communication graph correspond to only subsets of the groups in the organization, not capturing the correlation effectively. Furthermore, the set of maximal cliques in a large graph may be prohibitively expensive to enumerate and estimate W_∞ for each. In our model we instead define the graph as a union of neighborhoods, where each neighborhood is defined as a central edge and its adjacent neighbors. Figure 3.1 shows an example of this model. The edge 23 is adjacent to edges 21, 24, 31 and 34. Similarly, edge 57 is adjacent to edges 51, 56, 58, 59, and 76. Note that the conditional independence assumption implies that knowledge of the edge properties in the neighborhood is sufficient to determine the properties of the central edge.

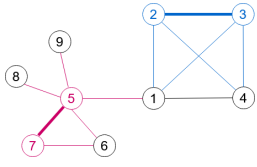


Figure 3.1: Neighborhood correlation, each edge is correlated with its adjacent edges.

A change in an edge's properties will influence its neighbors. For example, an edge labeled with the property "acquisition" may imply that neighbors are much more likely to share this property. If we can model the effect of this change probabilistically, we can compute the Wasserstein distance between query distributions, providing a sensitivity measure that accounts for an edge's correlation with its neighbors. In the next section we outline our privacy definition, followed by three proposed models for capturing neighborhood correlation.

3.4.2 Privacy Definition

We use Pufferfish privacy to design a private mechanism that takes correlation into account. We start with designing a private mechanism for counting one property in the graph, and then extend it to all property counts:

1. The database is a set of records: $\mathcal{D} = \{X_1, \cdots, X_N\}$; $X_i = 0$ or $X_i = 1$ corresponding to the events the edge i has the property or not[2], indicating complementary secrets s_i^0 or s_i^1.

[2]Or: whether the property frequency is above a certain threshold or not.

2. The Pufferfish parameters: $(\mathcal{S}, \mathcal{S}_{\text{pairs}}, \Theta)$: the set of secrets $\mathcal{S} = \{s_i^0, s_i^1; i = 1, \ldots, n\}$, the secret pairs to be indistinguishable $\mathcal{S}_{\text{pairs}} = \{(s_i^0, s_i^1), i = 1, \ldots, n\}$, and Θ, the set of models describing the correlation. The Pufferfish privacy guarantee is shown in Equation (3.2).

$$e^{-\epsilon} \leq \frac{\Pr_{\mathcal{M},\theta}(\mathcal{M}(\mathcal{D}) = w | s_i^0, \theta)}{\Pr_{\mathcal{M},\theta}(\mathcal{M}(\mathcal{D}) = w | s_i^1, \theta)} \leq e^{\epsilon} \tag{3.2}$$

   (a) In the organizational communication $\mathcal{S}$ consists of the binary values of each X_i, $i = 1, \ldots, n$.

   (b) $\mathcal{S}_{\text{pairs}}$ is what we want the property label to be indistinguishable from. Since the label is binary, $\mathcal{S}_{\text{pairs}}$ is (s_i^0, s_i^1), which indicate the existence or non-existence of an property in an edge.

   (c) Θ: Instead of considering a set of correlations, we focus on the neighborhood correlation for $\theta \in \Theta$. We use the Markov Quilt mechanism from [171] for this model. However, unlike the Markov Quilt mechanism, we rely on empirical data and measure the exact correlation inside the quilt.

   (d) Query f: Maps the dataset $\mathcal{D}$ into a scalar $f(\mathcal{D}) = |i \in \{1, \ldots, n\} : X_i = 1|$, counting the number of edges having the property of interest.

3. Markov Quilt. We adapt the Markov Quilt Mechanism to protect the edges in our neighborhood model. We assume that each edge X_i is correlated to its adjacent edges ($\mathcal{D}_N$) and has no correlation with the rest of the graph (neither to $\mathcal{D}_R$, nor to $\mathcal{D}_Q$), i.e. $\delta = 0$. The card($\mathcal{D}_N$) translates to the maximum number of adjacent edges to an edge, i.e. $2 \times deg_{max} - 1$, where deg_{max} is the maximum degree of a node in the graph, and we subtract 1 so as not to double-count the central edge itself. We note that the application of $2 \times deg_{max}$ group differential privacy (Corollary 3.2.1) would be a baseline for our case.

3.4.3 Correlation Models

In order to assess the Wasserstein distance between neighboring secret pairs (changing a single property from true to false or vice-versa), we require a model that can estimate by how much a neighborhood's labels might change due to a change in the central edge's label. Specifically, we seek to estimate $\Pr(f(\mathcal{D}) = w | s_i, \theta)$. The attacker's priors, encoded by θ indicate to what accuracy the attacker may be able to estimate the change in $f(\mathcal{D})$ if edge X_i changes its label, replacing s_i^0 with s_i^1 or vice versa. By applying the Markov assumption, we need only measure the impact of a label change on an edge's immediate

neighborhood (i.e. its impact on the Markov quilt). Thus, for each of the correlation models below, w is measured for the local neighborhood and the rest of the graph is assumed to be constant.

Conditional Model

Our first model, namely the *Conditional* model estimates the probability $\Pr(f(\mathcal{D}) = w|s_i, \deg(X_i), \text{freq}(a))$, where $\deg(X_i)$ is the number of edges adjacent to edge X_i, $\text{freq}(a)$ is the attacker's prior on the frequency of property a (i.e. their prior estimate of how many edges are labeled with a, perhaps sampled from a public corpus). We construct this model empirically by bucketizing $\deg(X_i)$ and $\text{freq}(a)$ on a logarithmic scale, sampling up to 100 edges per bucket, and building a histogram describing $\Pr(f(\mathcal{D}) = w)$, for discrete intervals of w. Note that since the rest of the graph is invariant to changes in s_i, we need only measure this distribution over values of w specific to the edge's local neighborhood. This is our highest-fidelity model and represents the most knowledge we assume an attacker may possess about the graph.

Global Model

Our second model, called the *Global* model ignores $\deg(X_i)$ and $\text{freq}(a)$ and empirically measures $\Pr(f(\mathcal{D}) = w|s_i^j)$ for secrets s_i^0 and s_i^1. The resulting model is a normalized frequency histogram of how often $f(\mathcal{D}) = w$ when the central edge's property is set, s_i^1 , and a separate histogram for when it is not set s_i^0. As in the conditional model, we measure the distribution over the range of values that w may take over a local neighborhood, assuming the rest of the graph to be constant.

Binomial Model

Finally our third model, which we call the *Binomial* model, representing the least amount of attacker knowledge, empirically estimates $p_i = \Pr(s_j|s_i)$, the probability distribution over a randomly selected adjacent edge's secrets, given the label of the central edge, and then estimates $\Pr(f(\mathcal{D}) = w|s_i)$ as a Binomial distribution parameterized by p_i and $\deg(X_i)$:

$$P(f(\mathcal{D}) = w|s_i) = \text{Binomial}(\deg(X_i), p_i)$$

That is, for a given edge X_i with $\deg(X_i)$ neighboring edges, if X_i's label is s_i, then the probability that it has w neighboring edges with a true label is represented by the

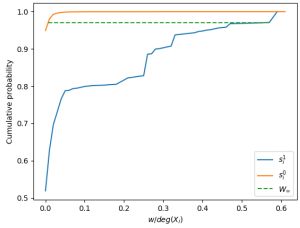


Figure 3.2: Example W_∞ determined by differencing cumulative distributions for the *Conditional* model.

distribution of successes over $\deg(X_i)$ Bernoulli trials with success probability p_i. Note that p_i is considered a constant over the whole graph, independent of $\deg(X_i)$, and is parameterized by the central edge's label s_i.

3.4.4 Measuring W_∞

Our correlation models afford a straightforward estimation of the maximal Wasserstein distance $W = max_{X_i \in \mathcal{D}} W_\infty(X_i)$. For each neighborhood in the graph, instantiate distributions $\Pr(f(\mathcal{D})|s_i^0)$ and $\Pr(f(\mathcal{D})|s_i^1)$, and measure W_∞. Computationally, this is equivalent to determining the largest horizontal distance between the two distributions when they are expressed as cumulative functions, as illustrated in Figure 3.2. W is then the maximal W_∞ over all edges. Note that W is bounded above by the largest neighborhood size: flipping a single edge property may trigger a flip in at most $\deg(X_i)$ adjacent edges.

3.5 Experiments

We run our experiments on the Avocado corpus[149]. In order to accurately recognize messages sent to multiple recipients, we apply a simple inference heuristic for identifying organizational mailing lists and their memberships (enabling, for example, the expansion of "All Employees", instantiating an edge from sender to each member of this list. The complete graph contains 393 nodes (individuals) and 21312 edges representing correspondence

$\log(freq)$	$\log(deg)$	W_∞	W
0	0	1.0	10.0
0	1	0.08	8.0
0	2	0.02	20.0
0	3	0.01	18.83
1	0	1.0	10.0
1	1	0.58	57.0
1	2	0.5	500.0
1	3	0.09	169.47
2	0	0.71	7.1
2	1	0.66	66.0
2	2	0.74	740.0
2	3	0.51	960.33
3	0	0.5	5.0
3	1	0.31	31
3	2	0.37	370.0
3	3	0.29	546.07
4	0	0.36	3.6
4	1	0.16	16.0
4	2	0.21	210.0
4	3	0.13	244.79

Table 3.1: Wasserstein metrics for edges with neighborhoods of size *deg* and properties with global frequency *freq*.

between users. The largest neighborhood in the graph consists of 1883 edges. Edges are subsequently labeled with properties. We extract unigrams and bi-grams from messages passed between edges and set the edge property X_i^a to "true" for each ngram a. Thus an edge with the property "acquisition" set to true indicates that at least one message passed between the connected nodes containing the word "acquisition". Edges with no such property are implicitly "false" for that property. With edge properties set, we construct the three correlation models, as described in section 3.4.3. Table 3.1 shows the estimated Wasserstein measures for the various property frequencies and neighborhood size under the *Conditional* correlation model. The boxed row represents the highest sensitivity. The maximum influence W_∞ of a bucket is scaled by the maximum neighborhood size for the

bucket, up to the largest possible neighborhood in the graph $N_{max} = 1883$. That is:

$$W = W_\infty * \min(N_{max}, 10^{\log(deg)+1})$$

We note several points of interest in Table 3.1. First, that while the largest W corresponds to large neighborhoods ($\log(deg) = 3$), it doesn't necessarily correspond with high-frequency or low-frequency properties. Second, we note that the maximal W is roughly equivalent to half the largest neighborhood, achieving some improved utility over group privacy.

Our second model, the *Global* model measures $\Pr(f(\mathcal{D}) = w|s_i)$ directly for secrets s_i^0 and s_i^1. The cumulative distribution functions of these measures are shown in Figure 3.3. The maximal Wasserstein measure is the maximum horizontal distance between these two distributions, or 866. Finally, the *Binomial* model represents the two label distributions by estimating Bernoulli parameters p_0 and p_1 for each label respectively, and measuring the maximal Wasserstein distance between these distributions $\Pr(f(\mathcal{D}) = w|s_i^j) = \text{Binomial}(\deg(X_i), p_j)$. Using this approach we empirically measure p_0 to be 0.0277 and p_1 to be 0.2739. These parameters indicate that an adjacent edge is about ten times more likely to have property a if the central edge has property a. Figure 3.4 shows the binomial distributions for these Bernoulli parameters, and again the maximal Wasserstein measure is the maximum horizontal distance between the curves, or 558. Compared with the *Global* model, we observe that choosing a binomial distribution is a relatively poor approximation of the empirical behavior, but may represent an attacker's best guess as to how neighborhoods vary when edge properties change.

3.5.1 Language Tasks

We apply the privacy mechanism to two query tasks, histogram release, and differentially private set union (DPSU) [69].

Histogram release

Our first task involves generating a histogram over edge properties. When edge properties are ngrams, the task is equivalent to computing the frequencies of ngrams in the corpus. To limit the sensitivity of the histogram to changes in a single edge, we limit the contribution of each edge to $c = 1000$ distinct ngrams. The value of c determines the maximum number of ngrams each edge contributes and we choose the c most common on each edge. Note

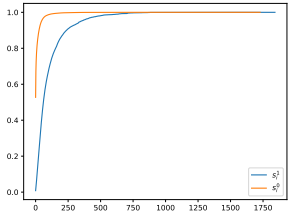


Figure 3.3: Cumulative distributions of $\Pr(f(\mathcal{D}) = w|s_i^j)$ for the *Global* model, conditioned on secret s_i^j.

that for this task it is assumed that the domain of ngrams is known *a priori* from a public source. In practice it is usually necessary to identify these using the private corpus as well, which we address in the second experiment. Histogram publication can be accomplished using the Laplace mechanism, adding Laplace noise with scale parameter $\lambda = cW/\epsilon$, where c is the per-edge contribution limit, W is the maximal Wasserstein distance accounting for edge-neighborhood correlation, and ϵ is the privacy budget. For the purposes of this experiment we choose a relatively large $\epsilon = 100$, as the corpus is comparatively small for running effective privacy mechanisms. It has been noted in other work that privacy mechanisms on graphs require large values of ϵ, e.g. [37]. To measure the utility of the result, we assess root mean square error (RMSE) between the noisy and true histogram, and also indicate the *utility* of the histogram, measured as the number of ngrams with positive counts (note that some noisy counts may be negative). Positive counts are necessary for language modeling tasks such as computing inverse-document-frequency [96], and a large number of negative counts indicates lower utility of the resulting histogram. Results are reported over ten trials. Table 3.2 contains the results for the histogram task. Note that even with a large epsilon, and a moderate amount of noise, in the best case (edge-level privacy) only 58% of ngrams have useful counts. However, unlike the Conditional, Global, or Binomial models this result doesn't account for correlation. Note that in the case of group privacy the added noise is comparable to the total number of edges in the graph.

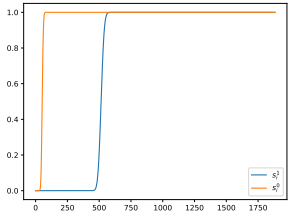


Figure 3.4: Cumulative distributions of $\Pr(f(\mathcal{D}) = w)$ assuming binomially distributed counts for secrets s_k^j on edges k adjacent to X_i.

DP Set Union Application

In the histogram publication experiment we noted that it is assumed that the domain of ngrams is known *a priori*, an assumption which may not be true. For instance, real-world language modeling applications may call for differentially private vocabulary selection. Differentially private set union (DPSU) aims to identify the union of elements in k input sets (in our setting, sets of edge properties on k edges). This problem was addressed in [69]. To account for edge correlation, as in the case of histogram publication, it is necessary to scale the sensitivity of the property counts by cW, as changing any edge can change as many as c properties and may influence its neighborhood by a factor as large as W. For this experiment we compare the utility (the number of published n-grams) of the privacy mechanism over ten independent applications of the mechanism, for each of the three correlation models. We also provide baseline utility for node-level, edge-level, and group privacy. We choose $\epsilon = 100$ and $c = 1000$ as in the previous experiment. The results of this experiment are shown in Table 3.3. As in the previous experiment, the best result corresponds to edge-level privacy, which neglects to account for edge-neighborhood correlation. Of the approaches that address correlation, the binomial model utilities the largest set of ngrams.

Description	W	λ	utility (%)	RMSE
Edge-level privacy	1	10.0	809200.5 ± 529.6 (58.2%)	12.0 ± 0.02
Node privacy	1	10.0	77679.0 ± 159.4 (5.6%)	6.4 ± 0.06
Binomial Model	558	5580.0	695823.0 ± 521.3 (50.1%)	5577.7 ± 8.14
Global Model	866	8660.0	695316.8 ± 647.3 (50.0%)	8665.8 ± 12.15
Conditional model	960	9600.0	695366.3 ± 561.83 (50.0%)	9594.2 ± 12.24
Group privacy	1883	18830.0	695195.5 ± 428.6 (50.0%)	18833.3 ± 26.73

Table 3.2: Experimental results for histogram publication, $\epsilon = 100$. We measure utility- the number of positive ngram counts for each of the correlation models, as well as for node-level, edge-level, and group privacy.

Description	W	E[utility]	σ
Edge-level privacy	1	24814.9	46.3
Node privacy	1	91	3.35
Binomial Model	558	228.7	5.71
Global Model	866	135	6.51
Conditional model	960	116.7	7.29
Group privacy	1883	41.9	3.11

Table 3.3: Experimental results for DPSU, $\epsilon = 100$. We measure utility- the number of extracted ngrams for each of the correlation models, as well as for node-level, edge-level, and group privacy.

Discussion

Our experimental results illustrate the challenges associated with differentially private language modeling tasks. If ϵ is large, and the number of private entities numbers in the tens of thousands, the utility of a DP mechanism can still be very limited. Despite these observations, our results illustrate how a privacy mechanism can be appropriately modified to account for neighborhood correlations in the graph, and we believe utilities can only improve with larger graphs. We suggest the following directions as future research to improve the privacy/utility.

1. Applying sensitivity reduction techniques can lessen the effect of large neighborhoods on the required privacy budget. These techniques have made great improvements in the node-level privacy literature [102, 24, 38, 159].

2. Using alternative neighborhood models with less sensitivity can improve the privacy. We use an over-protective model to capture cliques of users. For example, focusing on clusters in the organizational graph instead of considering all adjacent edges, can be promising.

3. Applying advanced composition techniques [99, 7, 54], can improve the privacy in series of queries significantly. Based on Rènyi differential privacy [140], moments accountant has shown this improvement in differentially private deep learning [7].

3.6 Related Work

Existing privacy approaches for social graphs use different techniques and mechanisms [17]. These techniques are categorized into three main categories: i) grouping-based approaches, ii) edge manipulation algorithms, and iii) differential-privacy-based techniques. Grouping-based approaches include k-anonymity-based approaches [41, 125, 192, 197, 200] and cluster-based techniques [22, 84, 122, 142, 177]. K-anonymity is among the first techniques proposed for protecting the privacy of datasets and aims to anonymize each user/node in the graph so that it is indistinguishable from at least $k-1$ other users. Machanacajjhala et al. [132] showed that a k-anonymous solutions still have severe privacy problems when the sensitive attributes lack diversity, or when the adversary has access to background knowledge. Clustering-based approaches group users and edges and are limited to the applications where only the density and size of the cluster is revealed, so that individual attributes are protected. Edge manipulation algorithms utilize edge-based strategies such as random edge adding/deleting and random edge switching [190, 15]. The edge perturbation algorithm can use random-walk-based techniques [128, 142] as well. As the most recent category, differential privacy provides a strong privacy guarantee that the risk of user's privacy leakage does not increase as a result of participating in a database [51]. A common way of achieving differential privacy is by introducing random noise through the Laplace mechanism (for numerical attributes) or Exponential mechanism (for non-numerical attributes) to the query answers [51].

The differentially private proposals for social graphs fall into two categories: edge-level differential privacy, and node-level differential privacy. In edge-level differential privacy [147, 163, 101, 83], the result of the analysis does not change by adding or removing an edge. Node-level privacy however [102, 24, 38, 159, 37, 65], is resilient against adding or removing a node, therefore it is more difficult to achieve. Note that differential privacy assumes independence among instances in the dataset. It has been shown that the dependency between instances

affects the robustness of differential privacy guarantees to de-anonymization [123, 197]. The naive approach to address this issue to use group-level differential privacy, where the group of dependant records/nodes are considered as one instance [37, 65]. The transition to group privacy sacrifices utility even more, and motivates our work. We investigate privacy for correlated data, as an alternative to group-privacy, to prevent leakage when the records are not independent [108, 171]. To do so, we use Pufferfish privacy [108], and propose a mechanism to achieve this privacy inspired by Song et al.'s work [171].

3.7 Conclusion

Natural language tasks are an area of increasing relevance with recent advances in language modeling capabilities, and the availability of large language corpora. In order to train language models targeting organizations, it is necessary to address latent confidentiality concerns in the data– even models that are released for internal use may leak information across groups of users. In this chapter we explored the problem of preserving organizational confidentiality in language tasks, leveraging the Pufferfish privacy framework, while addressing non-IID data among edges in the organization's social network. We showed how our proposed scheme presents a compromise between two extreme measures of privacy, record-level differential privacy and group privacy. By taking record correlation into account, our scheme provides a more meaningful notion of privacy than record-level differential privacy, while improving the utility compared to group privacy. In order to address correlation in the graph, we imposed a Markov assumption enabling us to locally model the sensitivity of graph queries under changes to edge properties, and estimated a maximal Wasserstein metric under varying assumptions about attackers' priors. To demonstrate our approach we applied the mechanism to a real-world organizational dataset and measured performance on a pair of statistical graph query tasks.

Queries over n-grams are only a small part of developing language models from natural language data, and our work leaves many open questions with respect to the development of language models with high utility. For example, can the Wasserstein mechanism be extended to model training tasks, such as the application of stochastic gradient descent [8]? Furthermore, we model correlation up to and including very large neighborhoods of edges, raising the question of how large neighborhoods in a closed graph should be treated– is there value in preserving confidentiality within the organization once neighborhoods reach some critical size? Clearly, we can derive improved utility if we are not required to protect these groups. Further research is needed to address these and related questions.

Chapter 4

Equi-Joins over Encrypted Data for Series of Queries

4.1 Introduction

Outsourcing data management into the cloud comes with new security risks. Insiders at the cloud service provider, attackers seeking high-profit targets or international legislators may exploit access to the cloud infrastructure. Encryption where the key is held at the client provides an additional layer of security countering these threats. However, regular data management operations such as joins cannot be simply performed over encrypted data. Hence, specialized encryption schemes for performing joins over encrypted data have been developed [78, 141, 187, 156, 153, 107, 30]. Suraj Gupta, Jin Yang Liu, Koray Karabina, Florian Kerschbaum and I studied equi-joins, since their security is particularly challenging. On the one hand, a database management system (DBMS) cannot hide the equality of join attribute values, since the cross product of two tables of size n each, is of size n^2 which is prohibitively large for subsequent operations. Hence, the DBMS needs to select a subset of the cross product using the equality condition. On the other hand, revealing the equality condition leaks the frequency of items in a primary key, foreign key join. This is critical, since primary key, foreign key joins are a very common operation and it has been demonstrated that frequency information is very powerful in cryptanalysis [114, 146]. The encryption scheme (for joins) by CryptDB [156] has been effectively broken using this frequency information [146]. Consequently, the challenge for any encryption scheme for joins is to allow selecting from the cross-product, yet reveal as few equality conditions as possible.

A state-of-the-art encryption scheme for joins by Hahn et al. [78] further reduces the leakage from deterministic encryption [77] and onion encryption [156, 180] by only leaking equality condition for tuples that match a selection criterion. However, the leakage of a series of queries in this encryption scheme corresponds to the leakage of the union of the queries, i.e., it may be larger than the union (sum) of the leakage of each query. We provide an example of such super-additive leakage in Section 4.3. In this project we aim to reduce the leakage of equality conditions even further. We present a new encryption scheme [165] for joins that not only restricts the leakage of the equality condition to tuples that match a selection criterion, but also limits the leakage of a series of queries corresponds to the transitive closure of the union of the leakage of each query, i.e., there is no super-additive leakage. We believe that this leakage is a natural lower bound for the leakage of an efficient encryption scheme for non-interactive joins using one outsourced DBMS. Note that oblivious joins [11, 14, 112] which only leak the size of the joint table, either require secure hardware or multi-party computation [16], and an interactive protocol that reveals the size of the joint table. Our construction requires the use of a new cryptographic technique – function-hiding inner product encryption – compared to previous approaches. Our construction is efficient with cryptographic operations requiring only a few milliseconds and the ability to run hash-based joins with expected time complexity $O(n)$. Our construction works for arbitrary equi-joins. As a comparison, the state-of-the-art encryption scheme by Hahn et al. [78] requires nested-loop joins (with $O(n^2)$ time complexity), and it only works for primary key, foreign key joins. We implemented our encryption scheme and evaluated encrypted joins over a dataset from the TPC-H benchmark.

4.1.1 Our Contributions

Our contributions are as follows:

- We provide a new encryption schemes for non-interactive equi-joins over encrypted data where a series of queries only leaks the transitive closure of the union of the leakage of each query, i.e., without super-additive leakage.
- We analyze the security of our scheme using a formal security proof.
- We evaluate the performance of a DBMS using our encryption scheme over a database from the TPC-H benchmark.

4.1.2 Chapter Organization

The remainder of this chapter is structured as follows: Section 4.2 provides the necessary cryptographic background. Section 4.3 describes the system model and problem in detail and Section 4.4 describes our join encryption scheme. We presents its security proof in Section 4.5 and its performance evaluation in Section 4.6. Section 4.7 surveys related work and Section 4.8 summarizes our conclusions.

4.2 Preliminaries

4.2.1 Polynomial Functions

We use the definition of polynomial functions by Leung et al. [119]. Consider a polynomial $f(x) = a_n x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0$ of the set of polynomial in x over the prime field $\mathbb{Z}_q$. If the indeterminate x in the expression is regarded as a *variable* which can assume any value in $\mathbb{Z}_q$, then in the most natural way the polynomial $f(x)$ will give rise to a mapping of the set $\mathbb{Z}_q$ into $\mathbb{Z}_q$. This mapping is defined by the polynomial $f(x)$ as follows. To each element c of the domain $\mathbb{Z}_q$ there corresponds under the mapping the unique value $f(c) = a_n c^n + a_{n-1}c^{n-1} + \cdots + a_1 c + a_0$ of the range $\mathbb{Z}_q$. Thus this is the mapping $c \to f(c)$ of $\mathbb{Z}_q$ into $\mathbb{Z}_q$. This mapping is called the *polynomial function* in the variable x defined by the polynomial $f(x)$. Polynomial functions that are bounded by Lemma 4.2.1 in their probability of evaluating to zero.

Lemma 4.2.1. (Schwartz-Zippel [164, 199] adapted in [109]) Fix a prime q and let $f \in \mathbb{Z}_q[x_1, \cdots, x_n]$ be an n-variate polynomial with total degree at most t and which is not identically zero. Then,

$$Pr[x_1, \cdots, x_n \xleftarrow{R} \mathbb{Z}_q : f(x_1, \cdots, x_n) = 0] \leq \frac{t}{q}. \tag{4.1}$$

4.2.2 Function-Hiding Inner Product Encryption

We build our scheme on the function-hiding inner-product encryption construction by Kim et al. [109]. Their scheme consists of four algorithms Π_{ipe} = (IPE.Setup, IPE.KeyGen, IPE.Encrypt, IPE.Decrypt) described below. We keep their notations where bold lowercase letters (e.g. $\mathbf{v}$,$\mathbf{w}$) denotes vectors and bold uppercase letters (e.g. $\mathbf{B}, \mathbf{B}^*$) denote matrices. $\mathbb{GL}_n(\mathbb{Z}_q)$ is the general linear group of $(n \times n)$ matrices over $\mathbb{Z}_q$.

1. IPE.Setup($1^\lambda, S$): On input of the security parameter λ and S a polynomial-sized (in λ) subset of $\mathbb{Z}_q$, the setup algorithm samples an asymmetric bilinear group $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, q, e)$ and chooses generators $g_1 \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$. Then, it samples $\mathbf{B} \leftarrow \mathbb{GL}_n(\mathbb{Z}_q)$ and sets $\mathbf{B}^* = det(\mathbf{B}) \cdot (\mathbf{B}^{-1})^T$. Finally, the setup algorithm outputs the public parameters $pp = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, q, e)$ and the master secret key $msk = (pp, g_1, g_2, \mathbf{B}, \mathbf{B}^*)$.

2. IPE.KeyGen(msk, $\mathbf{v}$): On input of the master secret key msk and a vector $\mathbf{v} \in \mathbb{Z}_q^n$, the key generation algorithm chooses a uniformly random element $\alpha \xleftarrow{R} \mathbb{Z}_q$ and outputs the pair: $sk = (K_1, K_2) = (g_1^{\alpha \cdot det(\mathbf{B})}, g_1^{\alpha \cdot \mathbf{v} \cdot \mathbf{B}})$. Note that the second component is a vector of group elements.

3. IPE.Encrypt(msk, $\mathbf{w}$): On input of the master secret key msk and a vector $\mathbf{w} \in \mathbb{Z}_q^n$, the encryption algorithm chooses a uniformly random element $\beta \xleftarrow{R} \mathbb{Z}_q$ and outputs the pair: $C = (C_1, C_2) = (g_2^{\beta}, g_2^{\beta \cdot w \cdot \mathbf{B}^*})$.

4. IPE.Decrypt(pp, sk, ct): On input of the public parameters pp, a secret key $sk = (K_1, K_2)$ and a ciphertext $C = (C_1, C_2)$, the decryption algorithm computes $D_1 = e(K_1, C_1)$ and $D_2 = e(K_2, C_2)$. Then, it checks whether there exists $z \in S$ such that $(D_1)^z = D_2$. If so, the decryption algorithm outputs z. Otherwise, it outputs $\perp$. The efficiency of this algorithm is guaranteed by $|S| = poly(\lambda)$.

The correctness of Π_{ipe} holds when the plaintext vectors $\mathbf{v}$ and $\mathbf{w}$ satisfy $\langle \mathbf{v}, \mathbf{w} \rangle \in S$ for a polynomially-sized S. Since $D_1 = e(K_1, C_1) = e(g_1, g_2)^{\alpha\beta \cdot det(\mathbf{B})}$ and $D_2 = e(K_2, C_2) = e(g_1, g_2)^{\alpha\beta \cdot \mathbf{v} \cdot \mathbf{B} (\mathbf{B}^*)^T \mathbf{w}^T} = e(g_1, g_2)^{\alpha\beta \cdot det(\mathbf{B}) \cdot \langle \mathbf{v}, \mathbf{w} \rangle}$, the decryption algorithm will correctly output $\langle \mathbf{v}, \mathbf{w} \rangle$ if $\langle \mathbf{v}, \mathbf{w} \rangle \in S$.

4.3 Problem Definition

4.3.1 System Model

In outsourced data management, a client stores their [sensitive] data on a database server under the control of a DBMS service provider [76]. Later, the client can access the outsourced data through an online query interface provided by the server. Clients desire to allow the server to process data queries while maintaining the confidentiality of the data. For this purpose, they encrypt data before outsourcing. However, encrypted data is

hard to process. Therefore, to allow for more expressive server-side data processing, the client will provide certain "unlocking" information (tokens) for a set of specific (equi-join) predicates. The client expects the server to behave semi-honestly and perform exactly the considered query while trying to find out any additional information. We consider a relational model, where the client outsources their data in a number of (at least two) tables each consisting of several data columns (e.g., relational attributes). In this work we focus on performing equi-join over two outsourced tables T_A and T_B. Without loss of generality, we assume both tables have n rows and m attributes, with each attribute taking its values from a domain of size ℓ, to simplify the notations. The equi-join result on the two join columns of the table-pair (T_A, T_B), is a subset of the cross-product of rows from the two tables that contain equal values in their join columns [78]. Assume table T_A with $|T_A| = n$ records, has schema $(A_0, A_1, \cdots, A_m)$ with join key A_0 that identifies the join column and other attributes $A_1, \cdots, A_m$. Each attribute A_i has a domain X_i. We denote by $x_{i,j}$, $i \in [m]$ and $j \in [\ell]$, the j^{th} domain value in X_i. We show the rows in T_A by tuples of variables: $(a_0^1, \cdots, a_m^1), \cdots, (a_0^n, \cdots, a_m^n)$. Similarly, T_B has schema $(B_0, B_1, \cdots, B_n)$, with domain Y_i for each B_i. We denote by $y_{i,j}, i \in [m]$ and $j \in [\ell]$, the j^{th} value in Y_i. T_B has $|T_B| = n$ rows shown by $(b_0^1, \cdots, b_m^1), \cdots, (b_0^n, \cdots, b_m^n)$. The equi-join with join attributes A_0 and B_0 is an operation on tables T_A and T_B, denoted by[1] $T_A \bowtie T_B$. The result of $T_A \bowtie T_B$ has schema $(\Theta, A_1, \cdots, A_m, B_1, \cdots, B_m)$, and consists of records $(\theta^{r,r'}, a_1^r, \cdots, a_m^r, b_1^{r'}, \cdots, b_m^{r'})$. The attribute Θ takes its values from all A_0's and B_0's that match, in other words: $\theta^{r,r'} = a_0^r = b_o^{r'}$, for all $r \in [n]$, $r' \in [n]$ where $a_0^r = b_0^{r'}$ holds. There is a further filtering based on additional filtering-predicates chosen from $\{A_1, \cdots, A_m\}$ and $\{B_1, \cdots, B_m\}$.

4.3.2 Super-additive Leakage

We describe the secure join problem through the following example.

Example 1. Consider T_A and T_B in Tables 4.1 and 4.2 containing employees information and their teams, respectively. Thus: (A_0, A_1) = (Key, Name), and (B_0, B_1, B_2, B_3) = (Team, Record, Employee, Role). Assume the filtering-predicates are chosen over A_1 with domain $X_1 = \{$Web application, Database$\}$ for T_A, and B_3 with domain $Y_3 = \{$Programmer, Tester$\}$ for T_B. The equi-join of these two tables over the join keys A_0 = Key and B_0 = Team, includes four pairs $(a_0^r, b_0^{r'})$ from row r in T_A and row r' in T_B, with true equality condition: (a_0^1, b_0^1), (a_0^1, b_0^2), (a_0^2, b_0^3), (a_0^2, b_0^4). Two additional equality pairs (b_0^1, b_0^2), (b_0^3, b_0^4) only from Table T_B need to be included in order to complete the transitive closure.

[1] As used in [78]

Key	Name
1	Web Application
2	Database

Table 4.1: Teams

Record	Employee	Role	Team
1	Hans	Programmer	1
2	Kaily	Tester	1
3	John	Programmer	2
4	Sally	Tester	2

Table 4.2: Employees

We also emphasize that the uniqueness of the join attribute values in A_0 is a feature of Example 1, not a general requirement, since our scheme is not limited to joins between primary key, foreign key joins.

We consider three database operations (queries) at times $t_0 < t_1 < t_2$, i.e. t_0 is the point in time after encrypted database upload, t_1 is the point in time after the first query, but before the second query and t_2 is the point in time after the first and the second query.

t_0: Encrypted database upload.

t_1: SELECT $\star$ FROM Employees JOIN Teams ON Team = Key *WHERE* Name = "Web Application" AND Role = "Tester"

t_2: SELECT $\star$ FROM Employees JOIN Teams ON Team = Key *WHERE* Name = "Database" AND Role = "Programmer"

Record	Employee	Role	T.Key	T.Name
2	Kaily	Tester	1	Web Application

Table 4.3: The result of equi-join query at t_1

In our analysis we compare encryption schemes for joins over encrypted data based on the (number of) pairs with true equality condition they reveal. The results of the two queries are depicted in Tables 4.3 and 4.4, respectively. To compute those results efficiently the DBMS needs to reveal the equality condition of the pairs (a_0^1, b_0^2) and (a_0^2, b_0^3), i.e., this

Record	Employee	Role	T.Key	T.Name
3	John	Programmer	2	Database

Table 4.4: The result of equi-join query at t_2

represents our minimum leakage and no efficient encryption scheme using non-interactive matches on a single DBMS can avoid this leakage.

The first proposal for database operations over encrypted data by Hacigümüs et al. [77] used deterministic encryption [26, 18] to compute joins. In deterministic encryption each data value is deterministically encrypted to the same ciphertext, such that the DBMS can compare the ciphertexts for an equi-join. While this idea by itself has been shown to be insecure [146], it is still the fundamental idea for subsequent schemes. In our analysis, we can state that deterministic encryption reveals all six (equal) pairs at time t_0. An improvement over deterministic encryption was presented by CryptDB [156]. CryptDB uses onion encryption and wraps each deterministic ciphertext in a probabilistic ciphertext. Hence, at time t_0 no pair is revealed, but at time t_1 all six pairs are revealed, since the wrapped probabilistic encryption needs to be stripped before an equi-join is feasible. The equality condition can be restricted to a few columns by using re-encryptable deterministic encryption [107, 141]. However, in our example there are only two columns and both are involved in the same join operation. Specialized schemes, such as [187, 30, 153], also maintain a re-encryption token for the entire table covered by the pair of columns. Hence, they do not offer any improvement against our analysis of CryptDB. A state-of-the-art encryption scheme for joins over encrypted data by Hahn et al. [78], loosely speaking, replaces the probabilistic encryption by key-policy attribute-based encryption (KP-ABE) [70]. They also use searchable encryption instead of deterministic encryption, but we ignore this in our work, since it does not impact our analysis. KP-ABE ensures that only rows that match a selection criterion specified by the attributes, can be decrypted and hence their ciphertext can be matched. After, time t_1 this scheme reduces the revealed pairs to (a_0^1, b_0^2) which is the minimum at this point of time.

However, consider what happens in this encryption scheme at time t_2. In the first query, the wrapped KP-ABE encryption on rows 1 from Team and 2 from Employees, but also row 4 from Employees since it also matches Role = “Tester”, are removed. In the second query, the wrapped KP-ABE encryption on rows 2 from Teams and 3 Employees, but also row 2 from Employees since it also matches Role = “Programmer”, are removed. In summary, at time t_2 the wrapped probabilistic encryption on all rows has been removed and all six (equal) pairs are revealed, since the adversary controlling the DBMS can match

the unwrapped rows. This reveals more pairs than necessary for the union of the queries. We call this super-additive leakage, since it is more than the sum of the leakages of each query. The challenge for our encryption scheme is to only reveal the pairs (a_0^1, b_0^2) and (a_0^2, b_0^3) at time t_2. The goal is a leakage equal to the transitive closure over the union of the leakages of each query. Hence, we claim that an encryption scheme with this leakage is more secure under a sequence of queries than the state-of-the-art encryption scheme by Hahn et al. Informally speaking, we aim for re-encrypting the deterministic ciphertexts to different keys for each query when the probabilistic encryption is removed. Constructing such an encryption scheme is not trivial, but we claim that a modification to function-hiding inner-product encryption [109] in combination with an encoding scheme using polynomials achieves the desired property.

4.4 Protocol Overview

Let tables T_A and T_B be the tables to be encrypted, and joined over a set of rows selected based on their attribute values. We propose a protocol that achieves this goal securely. In this section, we first describe our implementation of the selection operation, then we explain the modifications we made to the function-hiding inner product encryption scheme of Section 4.2.2, in order to implement a combined *selection and joins* operation. Subsequently in Section 4.4.3, we provide a full picture of our scheme in details. We clarify the steps of our protocol described in this section with the aid of Example 2.

Example 2. Assume sample rows r and r' of the tables T_A and T_B indicated in Tables 4.5 and 4.6.

A_0	A_1	A_2
a_0^r	a_1^r	a_2^r

Table 4.5: A sample row r in T_A

Consider the following database equi-join query, with specified selection filters:

$$\mathsf{SELECT} \star \mathsf{FROM}\ T_A\ \mathsf{JOIN}\ T_B\ \mathsf{ON}\ A_0 = B_0\ \mathsf{WHERE}$$
$$A_1\ \mathsf{IN}\ \Phi_1 = (\phi_{1,1}, \ldots, \phi_{1,t})\ \mathsf{AND}\ B_1\ \mathsf{IN}\ \Psi_1 = (\psi_{1,1}, \ldots, \psi_{1,t}).$$

B_0	B_1	B_2
$b_0^{r'}$	$b_1^{r'}$	$b_2^{r'}$

Table 4.6: A sample row r' in T_B

This query results in the join of the sample rows in Tables 4.5 and 4.6, if $a_0^r == b_0^{r'}$, and if the specified selection criterion matches the values of a_1^r and $b_1^{r'}$; in other words:

$$\exists \phi_{1,z} \in \Phi_1 \text{ s.t. } a_1^r = \phi_{1,z} \wedge \exists \psi_{1,z} \in \Psi_1 \text{ s.t. } b_1^{r'} = \psi_{1,z}.$$

The join query's *WHERE* clause in Example 2, imposes t restrictions (Φ_1) on one attribute (A_1) in T_A, and t restrictions (Ψ_1) on one attribute (B_1) in T_B. As we describe in Section 4.4.1, in general, the IN clause for table T_τ, $\tau \in \{A, B\}$, can impose a maximum of t restrictions on each of the m attributes in table T_τ.

4.4.1 Encoding Selection Operations in Polynomials

We implement the selection operation through the usage of polynomial functions. Each polynomial P_i, $i \in [m]$, is of degree t and can encode maximum t attribute values, as its roots. These attribute values are specified by Φ_i in the *IN* clause of the join query for T_A, and all belong to the same domain X_i. Recall from Section 4.3 that X_i is the domain of the attribute A_i. Hence, P_i enables the query to select the rows from T_A that have *particular* attribute values as attribute A_i. Similarly, each polynomial Q_i, $i \in [m]$, takes the values specified in the *IN* clause Ψ_i for T_B, as its roots. These values belong to Y_i, which is the domain of the attribute B_i. Therefore, Q_i enables the join query to select the rows from T_B with *desired* attribute values for B_i. Both $P_i(x)$ and $Q_i(y)$ take their coefficients from $\mathbb{Z}_q$.

Thus: $P_i(x) = \sum_{j=0}^{t} p_{i,j} \cdot x^j$, such that $P_i(\phi_{i,z}) = 0$, $z \in [t]$. In a similar way, $Q_i(y) = \sum_{j=0}^{t} q_{i,j} \cdot y^j$, $i \in [m]$, such that $Q_i(\psi_{i,z}) = 0$, $z \in [t]$. We emphasize that with the requirements of degree t, and maximum t specified points (roots), each P_i or Q_i can take any polynomial from a set of at least q distinct polynomials. If an attribute A_i or B_i is not included the selection criterion, it is encoded as the zero polynomial; i.e. $P_i = 0$ or $Q_i = 0$ respectively. Otherwise, according to Lemma 4.2.1, the probability of the non-zero polynomials of form P_i or Q_i evaluating to zero at a randomly selected point is bounded by $\frac{t}{q}$. We assume an efficient and injective embedding from the attribute values of A_i's and B_i's, $i \in [m]$, to $\mathbb{Z}_q$ which generates elements in $\mathbb{Z}_q$ uniformly at random, to comply

with the Schwartz-Zippel lemma. We use a cryptographic hash function to provide such a mapping.

To apply the filtering predicates in Φ_i's and Ψ_i's specified in the query, the client sends the corresponding polynomials' coefficients $p_{i,j}$'s and $q_{i,j}$'s as join tokens to the server. The server multiplies these coefficients by their corresponding [pre-stored] powers of attribute values. When these polynomials evaluating to zeros at rows with the target attribute values, they unwrap the join value for the server.

Example 3. Consider T_A and T_B in Tables 4.5 and 4.6, in addition to the join query in Example 2. The client uploads the non-join attribute values a_1^r and a_2^r of T_A on the server, as [an encrypted] vector $((a_1^r)^0, \cdots, (a_1^r)^t, (a_2^r)^0, \cdots, (a_2^r)^t)$. Similarly the non-join attribute values of T_B, i.e. $b_1^{r'}$ and $b_2^{r'}$, are stored as: [encrypted] $((b_1^{r'})^0, \cdots, (b_1^{r'})^t, (b_2^{r'})^0, \cdots, (b_2^{r'})^t)$. At query time, the client selects their filtering predicates, such as Φ_1 for attribute A_1, and Ψ_1 for attribute B_1. Hence, the client chooses $P_1(x)$ and $Q_1(y)$ such that they evaluate to zero at $\phi_{1,z}$'s, $z \in [t]$ and $\psi_{1,z}$'s, $z \in [t]$ respectively. For the other attributes, i.e. a_2^r and $b_2^{r'}$, the client assigns polynomials that are identical to zero. Thus, the client's join query token, consists [encrypted] $(p_{10}, \cdots, p_{1t}, \vec{0})$ for T_A and [encrypted] $(q_{10}, \cdots, q_{1t}, \vec{0})$ for T_B, where $\vec{0}$ is an all-zero vector of length $t+1$. It is easy to see that the inner product of the token generated for T_A and the stored vector for T_A equals zero, if $a_1^r = \phi_{1,z}$, for a $\phi_{1,z} \in \Phi_1$. A similar argument holds for T_B and $b_1^{r'} = \psi_{1,z}$, for a $\psi_{1,z} \in \Psi_1$. To give a clear picture of the polynomial-encoding implementation in this example, we skipped the details of the encryption operation. We provide a full description of our scheme in Section 4.4.3.

4.4.2 Modified Function-hiding IPE

We described the function-hiding inner-product encryption construction Π_{ipe} by Kim et al. [2] in Section 4.2.2, which is the base for our scheme. However, we made the following modifications on the construction to adjust it with our scheme's needs.

1. We set the random parameters in IPE.KeyGen and IPE.Encrypt to 1, i.e. $\alpha = \beta = 1$. We instead incorporate random parameters δ and γ in the input vectors $\mathbf{v}$ and $\mathbf{w}$ of IPE.Key and IPE.Encrypt, making vectors are of forms $\mathbf{v} = (\mathbf{v}', 0, \delta)$ and $\mathbf{w} = (\mathbf{w}', \gamma, 0)$.

2. Π_{ipe} generates a pair of secret keys $sk = (K_1, K_2)$ during IPE.KeyGen, a pair of encrypted outputs $C = (C_1, C_2)$ in IPE.Encrypt that decrypt to the pair (D_1, D_2) in IPE.Decrypt. In our scheme, we just use one element of these pairs. Hence, as

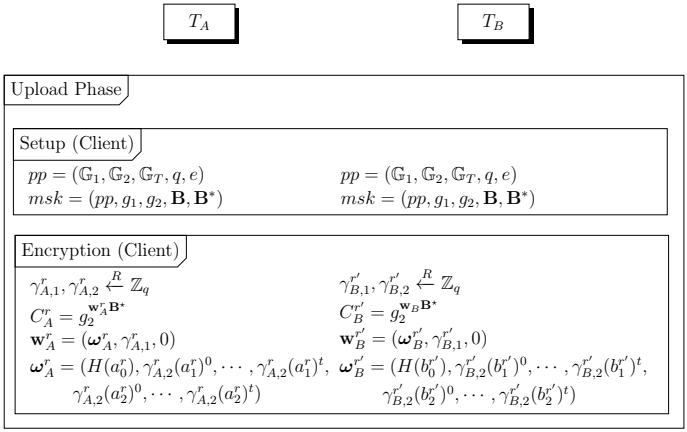


Figure 4.1: Upload phase in Secure Join on T_A, T_B of Example 2

we describe in full details in Section 4.4.3, we use the following parameters in our scheme: $sk^2 = g_1^{\mathbf{v} \cdot B}$, $C = g_2^{\mathbf{w} \cdot B^\star}$, $D = e(g_1, g_2)^{det(B) \cdot \langle \mathbf{v}, \mathbf{w} \rangle}$.

3. Π_{ipe} extracts the value of $\langle \mathbf{v}, \mathbf{w} \rangle$ from the decrypted value D at the end of the protocol. We are not interested in obtaining the value of $\langle \mathbf{v}, \mathbf{w} \rangle$ in our scheme, but a deterministic function of this value. Hence, we do not require the $\langle \mathbf{v}, \mathbf{w} \rangle$ reside in the polynomial-sized subset S for which one can break the discrete logarithm with overwhelming probability. Ultimately, We apply Π_{ipe} twice independently in our join encryption scheme, first on table T_A and then on table T_B[3]. We calculate $D(sk, C)$ for both of these iteration and conclude a "match" if the obtained D's are equal.

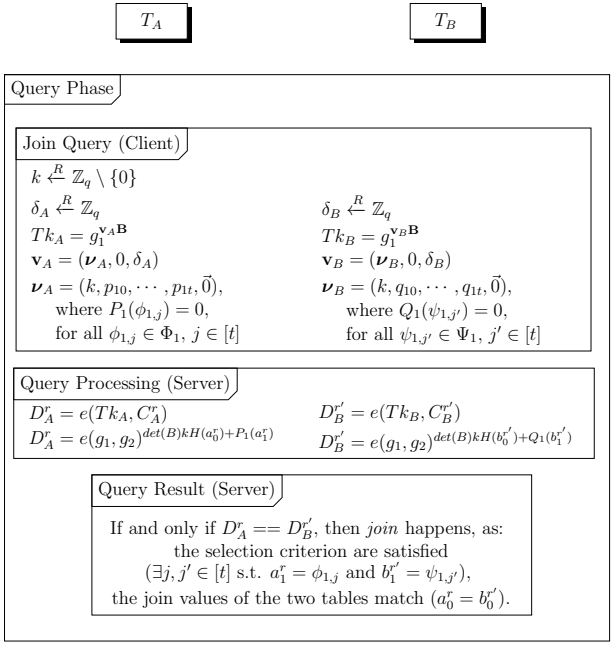


Figure 4.2: Query phase in Secure Join on T_A, T_B of Example 2

4.4.3 Our Secure Join Scheme

Our Secure Join scheme consists of five algorithms, namely (SJ.Setup, SJ.TokenGen, SJ.Enc, SJ.Dec, SJ.Match). The algorithms SJ.Setup, SJ.TokenGen, and SJ.Enc are applied by the client, on T_τ, where $\tau \in \{A, B\}$. The server applies SJ.Dec to T_τ. After applying the first four algorithms to both T_A or T_B and obtaining D_A and D_B, the server applies the fifth algorithm, SJ.Match, to the results to find out whether D_A and D_B match. A positive answer allows performing the join operation.

1. SJ.Setup(1^λ): (Client, upload phase)
   On input the security parameter λ, the setup algorithm samples an asymmetric bilinear group $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, q, e)$ and chooses generators $g_1 \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$. Then, it samples $\mathbf{B} \leftarrow \mathbb{GL}_n(\mathbb{Z}_q)$ and sets $\mathbf{B}^* = det(\mathbf{B}) \cdot (\mathbf{B}^{-1})^T$. Finally, the setup algorithm outputs the public parameters $pp = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, q, e)$ and the master secret key $msk = (pp, g_1, g_2, \mathbf{B}, \mathbf{B}^*)$.

2. SJ.Enc(msk, $\mathbf{w}^r_\tau$): (Client, upload phase)
   The encryption algorithm takes as input the master secret key msk and a vector $\mathbf{w}^r_\tau \in \mathbb{Z}_q^{m(t+1)+3}$ constructed from row r in table T_τ. To construct $\mathbf{w}^r_\tau$, the encryption algorithm chooses two uniformly random elements $\gamma^r_{\tau,1}, \gamma^r_{\tau,2} \xleftarrow{R} \mathbb{Z}_q$ to form $\mathbf{w}^r_\tau = (\boldsymbol{\omega}^r_\tau, \gamma^r_{\tau,1}, 0)$. The vector $\boldsymbol{\omega}^r_\tau$, represents the information in the row r of table T_τ. This information, is the hash of the join value and t powers of each of the other attribute values. Recall that t is the same as the degree of polynomials introduced in Section 4.4.1. These powers of attributes values in $\boldsymbol{\omega}^r_\tau$ are obfuscated by $\gamma^r_{\tau,2}$ in $\mathbf{w}^r_\tau$. Therefore, for a sample row $r \in [n]$ in T_A shown in Table 4.5, we have $\boldsymbol{\omega}^r_A = (H(a^r_0), \gamma^r_{A,2} \cdot (a^r_1)^0, \cdots, \gamma^r_{A,2} \cdot (a^r_1)^t, \cdots, \gamma^r_{A,2} \cdot (a^r_m)^0, \cdots, \gamma^r_{A,2} \cdot (a^r_m)^t)$. In a similar way, for a row $r' \in [n]$ of T_B in Table 4.6, we have $\boldsymbol{\omega}^{r'}_B = (H(b^{r'}_0), \gamma^{r'}_{B,2} \cdot (b^{r'}_1)^0, \cdots, \gamma^{r'}_{B,2} \cdot (b^{r'}_1)^t, \cdots, \gamma^{r'}_{B,2} \cdot (b^{r'}_m)^0, \cdots, \gamma^{r'}_{B,2} \cdot (b^{r'}_m)^t)$. The cryptographic hash function $H(\cdot)$ used in forming $\boldsymbol{\omega}^r_A$ and $\boldsymbol{\omega}^{r'}_B$, maps each attribute values of the join column to a fixed-size value, and acts [as much as practically possible] like a random function.

   Now, SJ.Enc($\cdot$) is ready to perform the encryption, and computes $C^r_\tau = g_2^{\mathbf{w}^r_\tau \cdot \mathbf{B}^*}$.

3. SJ.TokenGen(msk, Ξ_τ): (Client, query phase)
   The token generation algorithm takes as input the master secret key msk and the join-query's filtering predicates for table T_τ shown by $\Xi_\tau = (\xi_{\tau,1}, \cdots, \xi_{\tau,m})$. Recall

[2]We show this value by Tk in our scheme, as it acts as an unlocking token.
[3]The order does not matter here.

from Section 4.4.1 that $\Xi_A = (\Phi_1, \cdots, \Phi_m)$ and $\Xi_B = (\Psi_1, \cdots, \Psi_m)$. We elaborated in Section 4.4.1 that how the client chooses polynomials P_i's and Q_i's, $i \in [m]$, to encode the values specified in the *IN* clauses Φ_i's and Ψ_i's respectively. We also explained that for each P_i or Q_i, there are at least q such polynomials that the client can choose their candidate from. To generates a token for the join query, the SJ.TokenGen($\cdot$) algorithm chooses a uniformly random element $\delta_\tau \xleftarrow{R} \mathbb{Z}_q$ and generates a vector $\mathbf{v}_\tau \in \mathbb{Z}_q^{m(t+1)+3}$ of the form $\mathbf{v}_\tau = (\boldsymbol{\nu}_\tau, 0, \delta_\tau)$. The vector $\boldsymbol{\nu}_\tau$ consists of a [non-zero] symmetric secret query key k chosen randomly from $\mathbb{Z}_q \setminus \{0\}$ for encrypting the join attribute, and the coefficients of the polynomials corresponding to the filtering predicates $\Xi_B = (\Psi_1, \cdots, \Psi_m)$. Hence, to run the sample join query in Example 2, the clients first needs to generate vectors $\boldsymbol{\nu}_A = (k, p_{1,0}, \cdots, p_{1,t}, \cdots, p_{m,0}, \cdots, p_{m,t})$ for table T_A, and $\boldsymbol{\nu}_B = (k, q_{1,0}, \cdots, q_{1,t}, \cdots, q_{m,0}, \cdots, q_{m,t})$ for table T_B. Now, the token generation algorithm can compute $Tk_\tau = g_1^{\boldsymbol{\nu}_\tau \cdot \mathbf{B}}$, as the final token.

4. SJ.Dec(pp, Tk_τ, Ct_τ^r): (Server, query phase)
   On input of the public parameters pp, a token Tk_τ, and a ciphertext Ct_τ^r, the decryption algorithm computes $D_\tau^r = e(Tk_\tau, C_\tau^r)$ for a row r in T_τ. The output of SJ.Dec($\cdot$), if the selection criteria is satisfied, equals $e(g_1, g_2)^{det(B)kH(a_0^r)}$ for table T_A. Similarly, for the same query, the decrypted value for row r' of table T_B equals $e(g_1, g_2)^{det(B)kH(b_0^{r'})}$, when the selection criteria is satisfied.

   There exist many (searchable) encryption schemes [45] which can be used for pre-filtering the rows with the attributes matching the selection criteria reducing the size of the tables, but they are orthogonal to our join encryption scheme. For a better exposition, we describe only the application of our encryption scheme.

5. SJ.Match(D_A^r, $D_B^{r'}$): (Server, query result)
   This algorithm inspects the results of applying the previous four algorithms to all the rows in tables T_A and T_B, and performs a join when there is a match. In other words, the decrypted value for each row r in T_A, D_A^r, is compared with that of row r' in T_B, and if they match, the corresponding rows r and r' in T_A and T_B are combined to form the row $(\theta^{r,r'}, a_1^r, \cdots, a_m^r, b_1^{r'}, \cdots, b_m^{r'})$ in the join table, where $\theta^{r,r'} = a_0^r = b_0'$.

Example 4. Figure 4.2 shows the steps of Secure Join for the particular example of T_A and T_B in Tables 4.5 and 4.6, and the join query in Example 2. In the upload phase, the client first chooses the protocol parameters, and then starts the encryption for the (only) row r in T_A and the (only) row r' in T_B. The client forms the information vectors of these rows, $\boldsymbol{\omega}_A^r$ and $\boldsymbol{\omega}_B^{r'}$, and prepares them for encryption by randomizing them to obtain vectors $\mathbf{w}_A^r$

and $\mathbf{w}_B^{r'}$ that yield the final ciphertexts C_A^r and $C_B^{r'}$. The client uploads these ciphertexts at the server.

The client initiates the query phase by generating tokens to target rows with certain attribute values, specified in ϕ_1 for T_A and in ψ_1 for T_B, for the join operation. To do so, the client uses polynomial encoding to obtain vectors $\boldsymbol{\nu}_A$ and $\boldsymbol{\nu}_B$ for ϕ_1 and ψ_1, then randomizes these vectors to $\mathbf{v}_A$ and $\mathbf{v}_B$, which yield the final tokens Tk_A and Tk_B. Receiving these tokens in the query phase, the server decrypts the stored C_A^r and $C_B^{r'}$ with Tk_A and Tk_B to obtain D_A^r and $D_B^{r'}$. If $D_A^r \neq D_B^{r'}$, the server disregards the query. However, if they do match, the server performs the join and combines the rows r and r'.

4.5 Security

As stated in Section 4.1, we propose a new *encryption* scheme for joins that not only restricts the leakage of the equality condition to tuples that match a *selection criterion*, but also where the leakage of a series of queries corresponds to the transitive closure of the union of the leakage of each query, preventing super-additive leakage. Our design relies on the assumption that the clients are trusted and the server is semi-honest. A semi-honest adversary wants to learn confidential data, but does not change queries issued by the application, query results, or the data in the DBMS. This threat includes DBMS software compromises, root access to DBMS machines, and even access to the RAM of physical machines [156]. We structure our security proof as follows: First, we prove that our modified inner-product encryption scheme from Section 4.4.2 maintains the same security property as Kim et al.'s [109]. Then, using the simulator for the inner-product encryption we construct a simulator for our join encryption scheme.

4.5.1 Inner Product Encryption

Kim et al. prove security of their function-hiding inner-product encryption according to the following SIM-Security definition[4]

Definition 4.5.1. (SIM-Security for Function-Hiding Inner-Product Encryption) Recall Π_{ipe} = (IPE.Setup, IPE.KeyGen, IPE.Encrypt, IPE.Decrypt) from Section 4.2.2. Π_{ipe} is SIM-secure if for all efficient adversaries $\mathcal{A}$, there exists and efficient simulator $\mathcal{S}$ that

[4] An inner product encryption scheme that is secure under the simulation-based definition, is also secure under the indistinguishability-based definition. We refer to [109] for more details.

the output of the following experiments are computationally indistinguishable in security parameter λ.

1. $Real_{\mathcal{A}}(1^{\lambda})$
   - $(pp, msk) \leftarrow \text{IPE.Setup}(1^{\lambda})$
   - $b \leftarrow \mathcal{A}^{\mathcal{O}_{IPE.KeyGen}(msk,\cdot),\mathcal{O}_{IPE.Encrypt}(msk,\cdot)}(\lambda)$
   - output b
2. $Sim_{\mathcal{A},\mathcal{S}}(1^{\lambda})$
   - $(pp, st^{5}) \leftarrow \text{Setup}'(1^{\lambda})$
   - $b \leftarrow \mathcal{A}^{\mathcal{O}'_{IPE.KeyGen}(st,\cdot),\mathcal{O}'_{IPE.Encrypt}(st,\cdot)}(\lambda)$
   - output b

The oracles $\mathcal{O}_{IPE.KeyGen}(msk, \cdot)$ and $\mathcal{O}_{IPE.Encrypt}(msk, \cdot)$ represent the real key generation and encryption oracles of Π_{ipe}, while $\mathcal{O}'_{IPE.KeyGen}(st, \cdot)$ and $\mathcal{O}'_{IPE.Encrypt}(st, \cdot)$ represent the simulated stateful key generation and encryption oracles.

Lemma 4.5.1. The modified function-hiding inner-product encryption scheme of Section 4.4.2 is SIM-secure.

Kim et al. [109] provide full details of achieving security through a simulation-based proof in a generic model of bilinear groups. For space restrictions, we skip repeating the full detailed proof here, and provide their high-level idea instead. We then discuss that the modifications we introduced (Section 4.4.2) to their scheme does not compromise its security, hence the same security argument holds for our scheme as well.

To prove SIM-security of Π_{ipe}, Kim et al. construct a generic bilinear group simulator $\mathcal{S}$ that interacts with the adversary $\mathcal{A}$, such that the distribution of responses in the real scheme is computationally indistinguishable from that in the ideal scheme. The simulator $\mathcal{S}$ must respond to the key generation and encryption queries as well as the generic bilinear group operation queries. For each key generation and encryption query, the simulator responds with a fresh handle corresponding to each group element in the secret key and the ciphertext. Similarly, for each generic group oracle query, the simulator responds with a fresh handle for the resulting group element. The simulator maintains a table that maps handles to the formal polynomials the adversary forms via its queries. Hence, each oracle

[5] A simulator state

query is regarded as referring to a formal query polynomial. Two sets of formal variables are defined for Π_{ipe}, namely sets $\mathcal{R}$ and $\mathcal{T}$, with the universe $\mathcal{U}$ being the union of the two. All the formal polynomials the adversary submits to the final test (zero-test) oracle are expressible in the formal variables in $\mathcal{R}$. To answer the zero-test queries, the simulator performs a series of substitutions to re-express the adversary's query polynomials as a polynomial over the formal variables in $\mathcal{T}$. The major challenge in the simulation is in answering the zero-test queries. To consistently answer each zero-test query, the simulator first looks up the corresponding formal polynomial in its table and decomposes it into a "canonical" form, that is, as a sum of "honest"and "dishonest" components. The honest components correspond to a proper evaluation of the inner product while the dishonest components include any remaining terms after the valid inner product relations have been factored out. Kim et al. argue, using properties of determinants, that if a query polynomial contains a dishonest component, then the resulting polynomial cannot be the identically zero polynomial over the formal variables corresponding to the randomly sampled elements in $\mathbf{B}$. Then, the simulator can correctly (with overwhelming probability) output "nonzero" in these cases. Finally, in the ideal experiment, the simulator is given the value of the inner product between each pair of vectors the adversary submits to the key generation and encryption oracles, so it can make the corresponding substitutions for the honest inner product relations and thus, correctly simulate the outputs of the zero-test oracle.

Proof. Being built on the scheme Π_{ipe}, the security of our scheme results from that of Π_{ipe}. To prove the SIM-security of our Scheme in Section 4.4.2, we need to show that the modifications we introduced to the SIM-secure scheme Π_{ipe}, do not affect its security proof.

1. Changing the randomness from α and β, to δ and γ_1 in the input vectors $\mathbf{v}$ and $\mathbf{w}$ respectively.
   The simulator $\mathcal{S}$ should satisfy the following two conditions for corrections: i) $\mathcal{S}$'s response to the key generation, encryption, and group oracle queries made by the adversary $\mathcal{A}$ should be distributed identically as in the real experiment, and ii) $\mathcal{S}$ should correctly simulate the response to the zero-test queries made by the adversary $\mathcal{A}$. While the latter is guaranteed by the properties of determinants and randomly sampled elements in the matrix $\mathbf{B}$, the former is assured by the randomness of α and β. Our substitutes for the randomness provided by α and β, namely δ and γ_1, allow generating fresh handles in the simulation and consequently uniform and identical distributions as in the real experiment as per the first condition. The second condition however, is not affected by this modification, since it is satisfied by the properties of the matrix $\mathbf{B}$, regardless of the values of α and β. Hence, replacing α and β with

"1", and including the randomness in the input vector does not affect the satisfaction of this condition.

2. Eliminating the first item of the pair for each of the followings: secret key, ciphertext, and decrypted value.
   We mentioned earlier in this section that all the formal polynomials the adversary submits to the final test (zero-test) oracle are expressible over the variables in the set $\mathcal{R}$. Kim et al. argue that for any polynomial formed over these variables by the adversary, the simulator can correctly simulate the responses to the zero-test queries. Therefore, as long as new variables are not introduced to $\mathcal{R}$, this proof holds. Considering the elements in $\mathcal{R}$ (Definition 3.2 in [109]), eliminating the first item of the pair in the secret key, ciphertext and the decrypted value, does not change these formal variables, not to mention that it cannot introduce new variables to $\mathcal{R}$. Hence, it cannot enable the adversary to submit a "dishonest" component in the polynomial that outputs to zero in the zero-test to compromise the simulator.

□

Since our inner-product encryption scheme is SIM-secure we can replace its decryption keys and ciphertexts by outputs from the simulator and the two traces will be computationally indistinguishable as long as the decrypted plaintexts match.

4.5.2 Secure Join Encryption

We define security of our join encryption scheme following the methodology for symmetric searchable encryption (SSE) schemes by Curtmola et al. [45]. Let λ be the security parameter of the encryption schemes. Let $H = \{q_1, \ldots, q_\mu\}$ be a sequence of join queries where $\mu = \mathsf{poly}(\lambda)$. Let $\sigma(q_i)$ be the result of the equi-join query q_i, i.e., $\sigma(q_i) = \{(r_1, r'_1), \ldots, (r_\nu, r'_\nu)\}$ is the set of equality pairs between rows $r_{i,j}$ and $r'_{i,j}$ (which can be from the same table). We define the trace $\tau(H) = \{n, m, \sigma(q_1), \ldots, \sigma(q_\nu)\}$.

Definition 4.5.2. (SIM-Security for Join Encryption) We say a Join Encryption is SIM-secure, if there exists a simulator $\mathcal{S}(\tau(H))$ that given the trace $\tau(H)$ such that the following two experiments are computationally indistinguishable in the security parameter λ:

1. $Real_{\mathcal{A}}(1^\lambda)$
   - $(pp, msk) \leftarrow \text{IPE.Setup}(1^\lambda)$

- $b \leftarrow \mathcal{A}(VIEW_{DBMS}(H))$
- output b

2. $Sim_{\mathcal{A},\mathcal{S}}(1^\lambda)$
   - $(pp, st) \leftarrow \text{Setup}'(1^\lambda)$
   - $b \leftarrow \mathcal{A}(\mathcal{S}(\tau(H), st))$
   - output b

Theorem 4.5.2. The scheme Secure Join = (SJ.Setup, SJ.TokenGen, SJ.Enc, SJ.Dec, SJ.Match) from Section 4.4.3 is SIM-secure.

Proof. We construct the simulator $\mathcal{S}$ as follows: Recall that a^i_j and b^i_j are the plaintext values in the encrypted tables. We initialize all plaintext values to $\perp$. Given a set of equality pairs $\sigma(q_h)$, we set all join values a^i_0 and b^i_0 of equality pairs to the same random values from $\mathbb{Z}_q$ preserving already set values not equal $\perp$. All remaining join values of $\perp$ are replaced by random numbers. We compute values for the selection attribute values a^i_j, b^i_j and selection values $\phi_{j,i}$, $\psi_{j,i}$ by solving a linear system of equations for binary numbers – one for each possible pair of attribute value (or selection value) and domain value from $[n]$. This can be done in polynomial time using Gaussian elimination. Finally, we encrypt all plaintexts for tables and generate keys for the queries using the simulator for inner-product encryption. This results in random plaintexts with ciphertexts that produce the trace $\tau(H)$ as prescribed by the simulator. Each query produces now decryption results $D = D'$, if the join conditions (same query, same join values and selection criteria are satisfied) are fulfilled or random numbers otherwise. The actual values of the query results cannot be obtained by the adversary, since they cannot break the discrete logarithm.

It remains to show that the real protocol also either produces the same decryption results D, D' for join matches or random numbers, with overwhelming probability in λ (note that $q = O(2^\lambda)$). Without loss of generality, we simplify the representation and show by $D = e(g_1, g_2)^{det(B)kH(a_0)+P(a_1)}$ the decryption result for an arbitrary row from table T_A, with join value a_0 and selecting attribute A_1. D is decrypted by a token generated by query key k and polynomial $P(x)$. We show that if this D equals a D' of the form $e(g_1, g_2)^{det(B)k'H(b_0)+Q(b_1)}$ for an arbitrary row from T_B, decrypted by a token generated by query key k' and polynomial $Q(y)$, then D and D': i) belong to the same query, ii) have the same join value, and iii) satisfy the selection criteria. There are eight different cases to investigate based on the following conditions:

- Satisfying/not-satisfying the selection criterion

- Belonging to the same/different query/ies
- Equality/non-equality of join values

Claim 4.5.1. The equality $D = D'$, holds with overwhelming probability if and only if all of the three conditions above are satisfied.

In what follows, we show that this claim holds due to the randomness in: i) the symmetric key k, ii) the output of the hash function $H(\cdot)$, and iii) the coefficients of the polynomials, the probability of D and D' taking the same value is negligible in the all cases except the case of belonging to the same query, having the same join value, and satisfying the selection criterion. In investigating the following cases we assume we are given a pair of decrypted values (D, D'), where:

$$\begin{aligned} D &= e(g_1, g_2)^{det(B)kH(a_0)+P(a_1)}, \\ D' &= e(g_1, g_2)^{det(B)k'H(b_0)+Q(b_1)}. \end{aligned} \tag{4.2}$$

1. Same query, same join values, both selection criterion hod,

$$Pr[D = D'] = Pr[kH(a_0) = k'H(b_0)] = 1. \tag{4.3}$$

2. Same query, same join values, at least one of the selection criterion does not hold, e.g. the value of either of a_1 or b_1 is not included in the *WHERE* clause,

$$\begin{aligned} Pr[D = D'] &= Pr[kH(a_0) + P(a_1) = k'H(b_0) + Q(b_1)] \\ &= Pr[P(a_1) = Q(b_1)] \leq \frac{t}{q}. \end{aligned} \tag{4.4}$$

   Recall from Section 4.4.1 that how Lemma 4.2.1 is applied to the polynomial encoding. The last inequality in the above equation results from applying Lemma 4.2.1 to $P(x) - Q(b_1)$, which is a polynomial of degree t, and it evaluating to zero means the total probability of $P(a_1) = Q(b_1)$. This is an upper bound for the particular use case of $P(a_1) = Q(b_1)$.

3. Same query, different join values, both selection criterion hold,

$$\begin{aligned} Pr[D = D'] &= Pr[kH(a_0) = k'H(b_0)] \\ &= Pr[H(a_0) = H(b_0)] = \frac{1}{q}. \end{aligned} \tag{4.5}$$

D and D' belonging to the same query results in $k = k'$. Hence, we just need to calculate the probability of the hash output for join value b_0 colliding with that of a_0, while $a_0 \neq b_0$. As the cryptographic hash function $H(\cdot)$ acts as a random function with range $\mathbb{Z}_q$, this probability is $\frac{1}{q}$.

4. Same query, different join values, at least one of the selection criterion does not hold,

$$\begin{aligned} Pr[D = D'] &= Pr[kH(a_0) + P(a_1) = k'H(b_0) + Q(b_1)] \\ &\leq \frac{t}{q}. \end{aligned} \tag{4.6}$$

   Similar to the item (2), the inequality above results from applying the Lemma 4.2.1 to the polynomial $P(x) + kH(a_0) - k'H(b_0) - Q(b_1)$.

5. Different queries, same join values, both selection criterion hold,

$$\begin{aligned} Pr[D = D'] &= Pr[kH(a_0) = k'H(b_0)] \\ &= Pr[k = k'] + Pr[k \neq k', H(a_0) = 0] \\ &= \frac{1}{q-1} + \frac{q-2}{q-1} \times \frac{1}{q} = \frac{2}{q}. \end{aligned} \tag{4.7}$$

   D and D' correspond to equal join values, hence $H(a_0) = H(b_0)$. If $H(a_0)$ is not zero (which has probability $\frac{q-1}{q}$), the equality of D and D' requires the equality of the query keys, i.e., $k = k'$. Since k and k' belong to different queries are chosen independently and randomly from $\mathbb{Z}_q \setminus \{0\}$, the probability of one of them taking the same value as the other one is $\frac{1}{q-1}$, resulting in the overall probability $\frac{2}{q}$ for $D = D'$.

6. Different queries, same join values, at least one of the selection criterion does not hold,

$$\begin{aligned} Pr[D = D'] &= Pr[kH(a_0) + P(a_1) = k'H(b_0) + Q(b_1)] \\ &\leq \frac{t}{q}. \end{aligned} \tag{4.8}$$

   Similar to the items (2) and (4), the inequality above results from applying the Lemma 4.2.1 to the polynomial $P(x) + kH(a_0) - k'H(b_0) - Q(b_1)$.

7. Different queries, different join values, both selection criterion hold,

$$Pr[D = D'] = Pr[kH(a_0) = k'H(b_0)] = \frac{1}{q}. \tag{4.9}$$

We calculated the probability of $H(a_0) = H(b_0)$ when $a_0 \neq b_0$ in item 3, which is the probability of a $H(b_0)$ taking the same random value as the other independent value $H(a_0)$. This probability does not increase by multiplying these random values by other fixed or random values (k and k' here).

8. Different queries, different join values, at least one of the selection criterion does not hold,

$$\begin{aligned} Pr[D = D'] &= Pr[kH(a_0) + P(a_1) = k'H(b_0) + Q(b_1)] \\ &\leq \frac{t}{q}. \end{aligned} \tag{4.10}$$

Similar to the items (2), (4), and (6), the inequality above results from applying the Lemma 4.2.1 to the polynomial $P(x) + kH(a_0) - k'H(b_0) - Q(b_1)$.

From a given $(\mathbf{D}, \mathbf{D'})$ to any $(\mathbf{D}, \mathbf{D'})$ in the set of queries.
Through items (1) - (8), we showed that for a given (D, D'), the equality holds if and only if the corresponding rows to these values are decrypted through processing the same query, have the same join value, and satisfy the selection criteria in the query. The probability of the equality $D = D'$ taking place for any other cases is in $O(\frac{1}{q})$. However, if we consider all the $\varepsilon \leq 2\mu n$ ciphertext decryptions by μ queries over the join tables, this probability can increase to $O(\frac{\varepsilon^2}{2q})$, according to the birthday paradox. However, $O(\sqrt{q} = 2^{\lambda/2})$ is a loose upper bound for ε in our scheme, since the number of queries and the number of database rows in our scheme is polynomial-sized in λ, i.e., $\varepsilon = poly(\lambda)$. □

We can re-iterate our security properties as corollaries of Theorem 4.5.2.

Corollary 4.5.2.1. (Restricting leakage to the selection criterion) The Secure Join = (SJ.Setup, SJ.TokenGen, SJ.Enc, SJ.Dec, SJ.Match) restricts the leakage to the selection criterion.

The decryption algorithm, SJ.Dec($\cdot$), outputs $D = e(g_1, g_2)^{det(B)kH(a_0)+P(a_1)}$. Hence, the server can obtain $D = e(g_1, g_2)^{det(B)kH(a_0)}$, if and only if $P(a_1)$ evaluate to zero. According to Lemma 4.2.1 the probability of $P(a_1)$ evaluating to zero when the values of a_1 is not in the *WHERE* clause, is negligible. Hence, Secure Join successfully restricts the leakage to the selection criterion.

Corollary 4.5.2.2. (Preventing Super-additive leakage) The Secure Join = (SJ.Setup, SJ.TokenGen, SJ.Enc, SJ.Dec, SJ.Match) prevents super-additive leakage.

Items (5) to (8) in the proof of Theorem 4.5.2 show that the probability that SJ.Match results in a match for different queries is negligible. This protection holds even if both selection polynomials evaluate to zero, or even if the join values match. Hence, Secure Join successfully prevents the adversary from linking the results of different queries.

4.6 Experiments

In this section, we evaluate our Secure Join scheme (introduced in Section 4.4.3) in running secure hash joins over outsourced data. We provide three evaluation categories: i) a benchmark of the cryptographic operations in Secure Join, ii) server's performance in performing joins operation (decryption anch match) for various database sizes, and iii) server's performance in performing joins operation for various *IN* clause sizes for a single attribute.

4.6.1 Setup

We ran our experiments using a single thread on a machine with four processors, a 64-bit Intel Core $i7-7500U@2.70GHz$ each, with $15.4GiB$ RAM and running Ubuntu 20.04.01. In our experiments, we used an adjusted version of function-hiding inner product encryption implementation by Kim et al. [109] (described in Section 4.2.2). The adjustments were made to the implementation, so that it complies with our modifications from Section 4.4.2. In our experiments, we use the data provided by TPC-H benchmark. We in particular use the tables tables $T_A = Orders$ and $T_B = Customers$ from the benchmark. The table *Orders*, has nine attribute values: *(orderkey, custkey, orderstatus, totalprice, orderdate, orderpriority, clerk, shippriority, comment)*, while *Customers* has eight: *(custkey, name, address, nationkey, phone, acctbal, mktsegment, comment)*. The original tables *Orders* and *Customers* have $150,000$ and $1,500,000$ rows respectively. We use scale factors in the range of (0.01 - 0.1). The customer key information provides the join attribute *custkey* in both tables. We also add another attribute column *selectivity* to both tables. *Selectivity* takes values $\{\frac{1}{12.5}, \frac{1}{25}, \frac{1}{50}, \frac{1}{100}\}$. We use these values for two purposes: i) providing some values for the attribute *Selectivity*, ii) showing the proportion of the table assigned with this attribute value. Hence, each *Selectivity* value x is assigned to $x \times n$ rows where n is the table size.

For example, in a table *Customers* with $150,000$ rows, $1,500$ rows are assigned the same attribute value with $\frac{1}{100}$ as their *Selectivity*.

4.6.2 Crypto Operations in Secure Join

Figure 4.3 shows a micro-benchmark of our implementation of the cryptographic operations in Secure Join for a single row. This experiment provides the average implementation results for a row in the table *Customers*, when the *IN* clause size varies from 1 to 10. The cryptographic operations are SJ.Enc($\cdot$), SJ.TokenGen($\cdot$), and SJ.Dec($\cdot$) from Section 4.4.3.

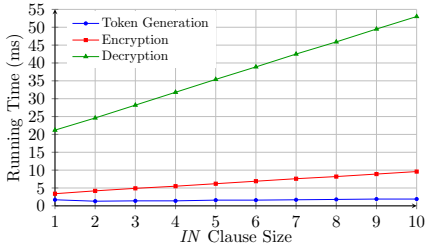


Figure 4.3: Encryption operation benchmarks for a single row in table *Customers*

The token generation algorithm does not show a noticeable change in runtime over different values for size of the IN clause (t in Section 4.4.1). The algorithm takes less than $2ms$ to run for each value of t, since it only calculates a single value of $g_1^{\nu_\tau \mathbf{B}}$, although increasing t from 1 to 10 changes the non-zero values in the $\boldsymbol{\nu}$ vector from 2 to 11 respectively. The encryption algorithm takes $3.4ms$ on average to encrypt a rows for $t = 1$, this time increases linearly to $9.6ms$ for $t = 10$, since the algorithm calculates the t powers for each attribute value in the table, pre-encryption. The most time consuming cryptographic operation in Secure Join is the decryption algorithm, which takes $21.2ms$ to run for $t = 1$ which increases to $53ms$ for $t = 10$.

4.6.3 Joins and Database Size

Figure 4.4 shows the runtime of joins operation over the encrypted data on the server side, i.e. SJ.Dec($\cdot$) and SJ.Match($\cdot$), for several database sizes and different *Selectivity* values (s), when the join query consists of a single value in the IN clause for each table. As expected,

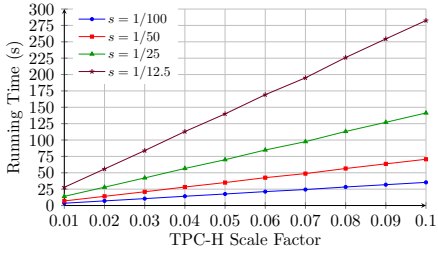


Figure 4.4: Joins runtime for various scale factors, single IN clause

the runtime of performing the joins operation increases linearly with the database size. This increase however, is more noticeable for higher values of *Selectivity*. When the *Selectivity* value, shown by s in Figure 4.4 is $\frac{1}{100}$, the server takes $3.52s$ to run joins over tables *Orders* and *Customers* with scale factor 0.01 and $35.34s$ to do the same for the tables with scale factor 0.1. However, when $s = \frac{1}{12.5}$, the server takes $27.88s$ to run joins over *Orders* and *Customers* with scale factor 0.01 and $282.49s$ to do so for the tables with scale factor 0.1.

4.6.4 Joins and IN-Clause Size

Figure 4.5 shows the runtime of joins operation over the encrypted data on the server side, i.e. SJ.Dec($\cdot$) and SJ.Match($\cdot$), for several sizes of *IN* clause (t) and different *Selectivity* values (s), when the scale factor for join tables *Orders* and *Customers* is 0.01. As depicted

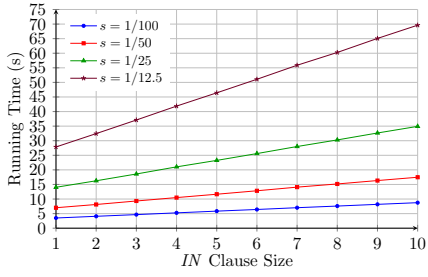


Figure 4.5: Joins runtime for *IN* clause with various sizes, scale factor: 0.01

in the plots in Figure 4.5, increasing t, results in a longer runtime for Joins. This increase, however taking place in all *Selectivity* experiments, is more noticeable the value of s is larger. It takes the server $3.50s$ to run the joins operation over the tables *Orders* and *Customers* for $t = 1$ and $8.75s$ to do the same for $t = 10$, when $s = \frac{1}{100}$. The corresponding running time for $s = \frac{1}{12.5}$ are $27.86s$ and $69.62s$. We ran each of the experiments in Sections 4.6.3 and 4.6.4, 25 times, aiming to demonstrate the data in the 95% confidence interval. However, the deviations from the mean value in our results were of order $10^{(-2)}$, resulting in unobservable error bars in our plots in Figures 4.4 and 4.5.

4.6.5 Comparison and Discussion

We mentioned in Section 4.1 that a state-of-the-art encryption scheme for joins by Hahn et al. [78] reduces the leakage to only leaking the equality condition for tuples that match a selection criterion. However, their scheme: i) requires nested-loop joins (with time complexity $O(n^2)$), ii) only works for primary key, foreign key joins, and iii) still results in

a super-additive leakage, as we showed in Section 1. Our new encryption scheme for joins prevents this super-additive leakage, is not limited to primary key, foreign key joins, and can run hash joins (with expected time complexity $O(n)$).

It is challenging to provide a one-to-one comparison of our performance measurement experiments with those in [78], due to: i) the differences in the parameters (the number of rows and attribute values) of the join tables, ii) the unclarity of join query parameters such as *IN* clause size and selectivity in [78], and iii) different hardware used to perform the experiments. Hence, we provide approximate performance comparisons. Their experiments [78] report an average runtime of 15 seconds for 1000 average decrypted values, i.e., $15ms$ per decryption. Our results in Figure 4.3 show an average time of $21ms$ for one decryption operation (for *IN* clause size of one). The results in [78] also report the average of $6s$ for a join operation over tables *Part* ($20,000$ rows and 6 attribute values) and *LineItem* ($6,000,000$ rows and 8 attribute values) from TPC-H, with scale factor 0.1. Our experiments show a result of $35s$ over tables *Orders* and *Customers* with scale factor 0.1 (and selectivity $\frac{1}{100}$). In conclusions, our performance is already on the same order of magnitude even without any parallelization at better security.

Furthermore, the experiments in [78] benefit from performance improvement provided by parallelizing each join query over 32 cores. The authors also re-use the decrypted information of the earlier join queries in the later ones to boost performance further. While our experiments are not intended to reuse the decrypted information due to the stronger security objectives of our scheme, they can as well benefit from parallelizing over several cores, instead of running on just one (the current setup).

4.7 Related Work

In this work we consider non-interactive equi-joins over encrypted data in a single-client, single-server setting. This model is known as the database-as-a-service model [76]. We provide the history of join encryption schemes [78, 156, 180, 77] that address security in this model in Section 4.3. All of these schemes use a deterministic or searchable encryption scheme as its basic building block and then aim to reduce the leakage of the join pattern, i.e., the number of equality pairs revealed. We argue that our join encryption scheme proposed in this chapter leaks the least information in this setting and represents a natural lower bound of the necessary leakage in the two table setting.

CryptDB [156] also introduced the concept of re-encryption as a method to reduce leakage in the multiple (more than two) table setting. The idea of re-encryption is that each table is encrypted with a different key and tables are re-encrypted to joint keys on a join operation. Kerschbaum et al. [107] introduce an algorithm that optimizes the selection of the joint key. Mironov et al. [141] present a new encryption scheme that makes the re-encryption uni-directional and hence prevents linking non-matched rows in a group of joins. Note that our proposed encryption scheme uses a fresh key in each query and hence does not need to resort to any of these techniques in order to achieve their (and stronger) security properties. Pang and Ding [153] present a secret-key encryption scheme that avoids

self-joins and Wang and Pang [187] extend it to a public-key encryption scheme. Carbunar and Sion [30] use Bloom filters to achieve a similar security guarantee, but also have to manage false positives at the client due to the properties of Bloom filters. All three schemes either reveal the equality pairs of the entire columns or none, just as CryptDB. Hahn et al. [78] and now our scheme improve over this by only revealing the equality pairs for rows matching a selection criterion.

Many join algorithms can be parallelized, and Bultel et al. [29] process joins over encrypted data in a map-reduce cluster. Such a parallelization is also applicable to our encryption scheme and could further speed up join operations over large data, although our performance is already competitive to the state-of-the-art at better security. Interactive schemes, e.g., secure multi-party computation, fully homomorphic encryption with intermediate decryption or secure hardware, can also implement joins over encrypted data. These schemes need to implement a circuit, i.e., an algorithm whose instructions and data accesses are independent of the input data. Agrawal et al. [11] were the first to introduce this problem as sovereign joins. Arasu and Kaushik [14] present the first, non-trivial, secure algorithms, but they are still too complicated to be practically implemented. Krastnikov et al. [112] present the first non-trivial, secure and practical algorithms.

Joins can also be consider between datasets from multiple parties. Mykletun and Tsudik [143] are the first to point out there is inherent threat of collusion between one of the (two) parties and the database provider which cannot be fully avoided. Hang et al. [81] present a key management scheme for multiple parties in this setting. Kantarcioglu et al. [100] develop an anonymization scheme that can prevent some of the leakage while maintaining efficiency. A cryptographic technique to match datasets from two parties is private set intersection (PSI). PSI was introduced by Fagin et al. [56] and formally developed by Freedman et al. [58]. It is practically deployed by Google and Mastercard [94]. PSI protocols using a service provider exist [105, 6, 104]. However, PSI cannot be applied to equi-join, since a prerequisite for PSI is that each data element in each set is unique. The security of most PSI protocols deteriorates to the all-or-nothing disclosure level of CryptDB when elements can be replicated as in an equi-join over two database tables.

4.8 Conclusion

In this chapter we present a new join encryption scheme that prevents additional leakage from a series of queries. We compare our scheme to the state-of-the-art join encryption schemes and it achieves comparable performance even without parallelization, better scalability (due to hash joins instead of nested loop joins) and reduced leakage over series of queries, i.e., better security. We provide a formal security proof and evaluate an implementation over a dataset from the TPC-H benchmark. We claim our construction achieves a natural lower bound for the leakage in an efficient, non-interactive, single-server setting. Hence, future work can investigate which restrictions to remove in order to further reduce the leakage of join encryption schemes.

Chapter 5

On the Robustness of Watermarking in Deep Neural Networks

5.1 Introduction

The task of generating a neural network model is computationally expensive and also requires a considerable amount of training data that has undergone a thorough process of preparation and labeling. The task of data cleaning is known to be the most time consuming task in data science [157]. According to a data science practitioners' report, data scientists spend around 80% of their time on just preparing and managing data for analysis [57]. This enormous investment on preparing the data and training a model on it is however at an immediate risk, since the model can be easily copied and redistributed once released. To protect the model from unauthorized re-distribution, watermarking approaches have been introduced, inspired by wide deployment of watermarks in multimedia [115, 162, 174]. Watermarking approaches for DNNs span over two broad categories: white-box [181] and black-box watermarking [10, 75, 195]. Black-box watermarking can be verified more easily than white-box watermarking; in the former, the verification only requires API access to the service using the stolen model, while in the latter verification requires access to all parameters of the stolen model. Furthermore, black-box watermarking is advantageous over white-box watermarking as it is more likely to be resilient against statistical attacks [185].

Nils Lukas, Jiaqi Wang, Xinda Li, Florian Kerschbaum and I investigated [168] recent black-box watermarking approaches [10, 75, 195]. These approaches each introduce a (some) variant(s) of backdoor-based watermarking to protect model ownership. Backdoors or neural trojans, [39, 74, 126], are originally the terms for a class of attacks against the security of deep learning when an entity outsources the learning for model computation to another untrusted but resourceful party. The party can train a model that performs well on the requested task, while its embedded backdoors lead to targeted misclassifications when encountering a particular trigger in the input. The idea of "turning a weakness into a strength" [10], launched a new line of work suggesting using backdoors for ownership protection [10, 75, 195]. The idea is to use some triggers in the training data set that

embed specific behavior, a *signature*, but do not impede the regular classification task. The triggers in the training data set can take one of the following forms: embedded content representing a logo of the owner [75, 195], a pre-defined noise pattern in the inputs [195], or a set of particular inputs acting as a secret *key* set [10, 195]. In this chapter, we investigate whether backdoors in DNNs are sufficiently robust as watermarks embedded in the models. Existing approaches for removing backdoors either do not output a model, but are deployed alongside a service [42, 61, 127] or are not effective in removing watermarks without access to the ground-truth labels [184, 124]. Neural-Cleanse [184] requires access to a set of correctly labeled samples to detect the backdoor and only works for one type of backdoor used in watermarking – a small patch on the images. Fine-Pruning [124] removes backdoors by pruning redundant neurons that are less useful for the main classification. As we show in our experimental results, fine-pruning without access to correctly labeled data is also ineffective in removing the backdoors in all of the aforementioned watermarking categories. When fine-pruning successfully drops the retention rate of the watermark, it also causes a significant drop in the model's accuracy rendering it basically useless.

We introduce two new attacks (black-box and white-box) on the backdoor-based watermarking schemes named above, and show that all of these watermarks are removable in both attacks. Our attacks require neither any correctly labeled training data, nor the backdoor embedded in the model. We have two goals: i) removing the watermark, while ii) providing the same functionality with little accuracy drop from the original marked model. Both, our black-box or white-box attack, achieve these goals; the black-box attack has only access to the predicted labels by the attacker, while the white-box attack is more efficient. With either of these attacks, an attacker can produce an unmarked model for re-distribution from the marked model without the need for preparing labeled training data and in case of the white-box attack without access to the same computational resources. Our black-box attack is a model stealing attack and shows that backdoors do not transfer among models if the stolen model is not trained on the backdoor samples. Our white-box attack requires access to the model's parameters, but is up to twenty times more efficient than training a model from scratch. Moreover, the accuracy of the models produced by our white-box attack may even improve over the accuracy of the marked models.

5.1.1 Our Contributions

This work has two main contributions:

1. We introduce our black-box attack that removes the embedded watermark in backdoor-based watermarking schemes. This attack solely relies on publicly available information, i.e. no labeled data, and successfully removes the watermark from the neural network without requiring any access to the network parameters, the classification probability vector, or the backdoor embedded as the watermark. The performance and accuracy of the stolen model in this attack is within 4% of the watermarked model.

2. We introduce our white-box attack for scenarios that we are guaranteed access to the model parameters. Benefiting from the additional information, our white-box attack is more efficient and offers a(n) (up to twenty times) faster version than training the data from scratch, with similar model accuracy within 1%[1] of the watermarked model. Our attacks show that these backdoor-based watermarking schemes are insufficient to protect against re-distribution by a motivated attacker. We explicitly show that the unremovability property defined by Adi et al. [10] does not hold. It therefore seems necessary to develop new, stronger protection techniques.

5.1.2 Chapter Organization

We provide formal definitions for deep neural networks and backdoor-based watermarking schemes in Section 5.2 and follow it by the security vulnerability of the schemes in Section 5.3. Subsequently in Section 5.4, we introduce our black-box and white-box attacks for watermark removal. Finally in Section 5.5, we present the results of the experiments that confirm a successful watermark removal. In Section 5.6, we situate our work in the current body of research.

5.2 Preliminaries

Backdooring enables the operator to train a model that deliberately outputs specific (incorrect) labels; backdoor-based watermarking schemes use this property to design trigger sets to watermark the DNN. The intuition in black-box watermarking is to exploit the generalization and memorization capabilities of deep neural networks to learn the patterns of an embedded trigger set and its pre-defined label(s). The pairs of learned patterns and their corresponding predictions will act as the keys for the ownership verification. As described in Section 5.2.3, we focus on the three backdoor-based watermarking schemes proposed in the literature.

5.2.1 Definitions and Models

We follow the notations by Adi et al. [10] throughout this chapter to introduce our attack accordingly. In order to train a neural network classifier, we initially require some objective ground-truth function f. A neural network classifier consists of two algorithms: training and classification. In training, the network tries to learn the closest approximation of f. Later, during the classification phase the network utilizes this approximation to perform prediction on unseen data. Formally, the input to the neural network is represented by a set of binary strings: $D \subset \{0,1\}^*$. The corresponding labels are represented by $L \subset \{0,1\}^* \bigcup \{\perp\}$; with the symbol $\perp$ showing the undefined classification for a specific input. The ground-truth function $f : D \rightarrow L$, assigns labels to inputs. Also, for $\bar{D}$ the set of inputs with defined

[1] 3.5% for an Image-Net model trained on 30% of the data.

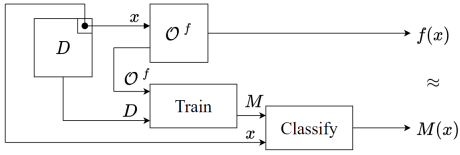


Figure 5.1: A high-level illustration of the learning process.

ground-truth labels, $\bar{D} = \{x \in D | f(x) \neq \perp\}$, the algorithms' access to f is granted through an oracle $\mathcal{O}^f$. Hence, the learning process illustrated in Figure 5.1, consists of the following two algorithms:

- Train $(\mathcal{O}^f, D)$: a probabilistic polynomial-time algorithm that outputs a model M
- Classify (M, x): a deterministic polynomial-time algorithm that outputs a label $M(x) \in L \backslash \{\perp\}$ for each input $x \in D$

The metric $\epsilon-$accuracy evaluates the accuracy of the algorithm pair (Train, Classify). In an $\epsilon-$accurate algorithm the following inequality holds:

$$Pr[Classify(M, x) \neq f(x) | x \in \bar{D}] \leq \epsilon$$

The probability is taken over the randomness of Train, with the assumption that the ground-truth label is available for those inputs.

5.2.2 Backdoor-based Watermarking in DNNs

Backdooring teaches the machine learning model to output *incorrect* but valid labels $T_L : T \to L \backslash \{\perp\}; x \mapsto T_L(x) \neq f(x)$ to a particular subset of inputs $T \subseteq D$, namely the trigger set. The pair $b = (T, T_L)$ forms the backdoor for a model. A randomized algorithm called *SampleBackdoor* generates the backdoors b. There are two variants for backdooring a model: during training or after training. We focus on backdooring during training, since our experiments support the conclusion by Adi et al. [10] that backdoors introduced after training the model are easier to remove. The complete backdooring process is illustrated in Figure 5.2.
Formally presenting, backdoor $(\mathcal{O}^f, b, M)$ is an algorithm that on inputs oracle to f, the backdoor b and a model M, outputs a model $\hat{M}$. The backdoor model $\hat{M}$ is required to output particular *incorrect* (regarding f) labels for the inputs from the trigger set and correct ones for other inputs. In other words, the following two inequalities must always hold for a backdoored model $\hat{M}$:

- $Pr_{x \in \bar{D} \backslash T}$[2] $[Classify(\hat{M}, x) \neq f(x)] \leq \epsilon$

[2]We assume $\bar{D} = D$, without loss of generality

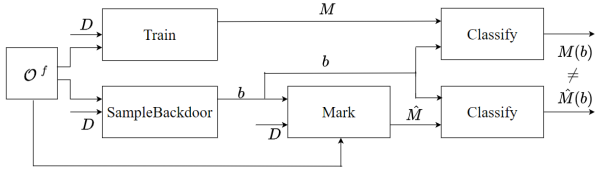


Figure 5.2: A schematic illustration of the backdooring process.

- $Pr_{x \in T}[Classify(\hat{M}, x) \neq T_L(x)] \leq \epsilon$

To watermark an ML model using the backdooring process, the algorithm $MModel()$ is used. $MModel()$ consists of the following sub-algorithms:

1. $b \leftarrow SampleBackdoor(D, \mathcal{O}^f)$: The watermarking schemes commits to the embedded backdoors b. Note that we differ from Adi et al. [10] by ignoring the master and verification keys mk, vk, since they have no affect on our attack.
2. Compute $\hat{M} \leftarrow Mark(D, \mathcal{O}^f, b)$: Computes the watermarked model $\hat{M}$ by training and embedding the backdoor b.
3. Output $(\hat{M}, b)$.

The verification of a watermark is performed by the algorithm $Verify(M, b)$. $Verify$ takes as input a model M and the backdoor b, and outputs a bit in $\{0, 1\}$, indicating whether the watermark is present in the model M or not. Formally,

$$Verify(M, b) = \begin{cases} 1, & \text{if } \sum_{x \in T} \mathbb{I}[Classify(M, x) \neq T_L(x)] \\ & \quad - \frac{1}{|L|}|T| \leq \epsilon |T| \\ 0, & \text{otherwise.} \end{cases}$$

where $\mathbb{I}[\text{expr}]$ is the indicator function that evaluates to 1 if expr is true and 0 otherwise. Note that, as we skip the cryptographic commitment details, the marking key mk in $Verify$ translates to the inputs x in the trigger set T, and the verification key vk refers to the corresponding labels T_L. Furthermore, the $\frac{1}{|L|}|T|$ comes from the assumption that the ground-truth label is undefined for the inputs of the trigger set T, for which we assume the label is random. Hence, we assume that for any $x \in T$, we have $Pr_{i \in L}[Classify(M, x) = i] = \frac{1}{|L|}$. As a result, it is expected that $\frac{1}{|L|}|T|$ of the inputs fall into the backdoor label "randomly". Hence, to verify the presence of the watermark in the model without a bias, we need to deduct this number from the classification result.

5.2.3 Backdoor-based Watermarking Schemes

We investigate the recent backdoor-based schemes in [10, 75, 195]. The watermark embedded in these schemes is one of the following three forms: Embedded Content, Pre-Specified Noise, and Abstract Images.

a) Content Embedded (Logo): This method [75, 195] adds a fixed visual marker (e.g. a text) to a set of inputs, namely the watermark set. The inputs with this watermark all classify to a fixed label.

b) Pre-Specified Noise: This method [195], adds a fixed instance of Gaussian noise as watermark to inputs. Similar to Content Embedded watermarking schemes, this approach maps the marked inputs to a fixed label.

c) Abstract Images: In this category of watermarking, a set of abstract images [10], or images that are not from the same distribution as the training data [195] is selected and labeled with pre-defined classes.

The are two main differences between the last watermarking and the first two, in Abstract Images: i) different subsets of the trigger set are mapped to different classes, not a fixed one, ii) the watermark is no more a pattern added to any input, but a fixed set of inputs and labels. Hence, the watermark test set is the same as the watermark train set.

5.3 Problem Definition

5.3.1 Unremovability in Backdoor-based Watermarking

Adi et al. [10] formally define the security properties of unremovability, unforgeability and enforcing non-trivial ownership in their paper. We focus on the unremovability property that prevents an adversary from removing a watermark, even if s/he knows about the existence of a watermark and the used algorithm. The unremovability requires that for every algorithm $\mathcal{A}$ the chance of winning the following game is negligible:

1. Compute $(\hat{M}, b) \leftarrow MModel()$
2. Run $\mathcal{A}$ and compute $\tilde{M} \leftarrow \mathcal{A}(\mathcal{O}^f, \hat{M})$[3]
3. $\mathcal{A}$ wins if:
   $Pr_{x \in D \setminus T}[Classify(x, \tilde{M}) = f(x)]$
   $\approx Pr_{x \in D \setminus T}[Classify(x, \tilde{M}) = f(x)]$
   and $Verify(\tilde{M}, b) = 0$

[3]Due to the cryptographic commitment, the verification key is useless for the adversary and not needed in our attacks.

5.3.2 Security claims in backdoor-based watermarking

We review the security claims in black-box watermarking as stated in [10, 75, 195], and discuss how our attacks invalidate the presented claims. (i) As stated in [195], Section 4: "After embedding (the watermarks), the newly generated models are capable of ownership verification. Once they [the marked models] are stolen and deployed to offer AI service, owners can easily verify them [the watermarks] by sending watermarks as inputs and checking the service's output." (ii) As well, authors [10] claim that their proposed scheme is persistent in the sense that "It is hard to remove a backdoor, unless one has knowledge of the trigger set." However, our attacks show that the watermark is successfully removable – hence, it is no longer verifiable – and to perform so, the attacker does not require any knowledge of the trigger set. (iii) On a similar note to [10], Guo et al. [75] claims that "transferring learning is on the same order of magnitude as the cost of training, if not higher. With that much resources and expertise at hand, an attacker would have built a model on their own." This claim neglects the fact that the cost of data preparation for the original model is comparable with (or even higher than) the cost of model generation itself, consequently the attacker saves a considerable amount by stealing the model through queries. (iv) Zhang et al. [195] refer to the results of [179] to state that stealing a model using prediction APIs needs queries of significant size; e.g. $100k$, where k is the number of model parameters in the particular example of two-layered neural network in [179]. They conclude that as more complicated models with more parameters the attack would even need considerably more queries. Our experiments demonstrate that a successful model stealing attack on ResNet-32 network with 0.46 million of parameters [85], only needs to query the API, 50,000 times and does not require access to the probability vector.

5.3.3 Invalidity of the security claims

We emphasize that our attacks are designed to invalidate the security claims by the proposed schemes [10, 75, 195]. Our adversary is weaker than the one assumed in their model. We propose two computationally bounded adversaries, who not only win the security game in Section 5.3.1, but also demand fewer requirements. Our black-box attack only requires API access to the model ($\hat{M}$) and inputs $\tilde{D}$ from the domain to remove the watermark while preserving the functionality. Our white-box attack does so using much less computational resources, through accessing the model ($\hat{M}$) parameters and public inputs $\tilde{D}$ from the domain. Although granted in the game, neither of our attacks require access to the ground-truth oracle $\mathcal{O}^f$, i.e. the labeled data. The adapted games are presented in Sections 5.4.1 and 5.4.2, respectively. We highlight the objectives of our attacks:

- We do not need any correctly labeled data. Data from the same distribution, but without any labels is sufficient for our attack. Hence, the most time consuming task of preparing the data sets ceases to apply to the adversary.
- We do not need many queries to the watermarked model to derive labels – on the order of the original training data set. Hence, rate-limiting or blocking the adversary

are not successful defences to our attacks.

- We do not need any knowledge of the trigger set or its labels of the backdoor. Hence, the adversary can use our algorithm only under the assumption that the model is watermarked.
- Our attacks, in particular the white-box attack, achieve comparable accuracy to the watermarked model even for complex models such as ResNet-32 for the Image-Net data set. Hence, an adversary can successfully use the stolen model.
- Our white-box attack requires significantly less training time than training the model from scratch. Hence, any claim that removing the watermark is too costly for the adversary does not apply – even ignoring the omitted time to prepare the data.

5.4 Attacks on Backdoor-Based Watermarking

The hypothesis behind our attacks is that backdoor-based watermarking schemes introduced in Section 5.2.3 divide the input distribution into two disjoint ones: i) main distribution which is classified correctly and ii) watermark distribution which is deliberately misclassified and does not fit the main distribution. This separation in the input distribution is common among all three types of trigger sets. This results in the watermark being treated as outliers in the main classification and the network never learns to classify it *correctly*. We introduce two attacks in Sections 5.4.1 and 5.4.2 that exploit this disjointedness to remove the watermark. Our attacks remove the watermark with less requirements than the original adversary $\mathcal{A}$ [10] (introduced in Section 5.3.1), as they do not require access to the training data and the ground-truth function. They also guarantee *higher efficacy*. The reason is, the unremovability game is labeled as won if the attacker $\mathcal{A}$ suggests a model $\tilde{M}$, such that the model achieves a similar test accuracy as the watermarked model $\hat{M}$ while: $\mathit{Verify}(\tilde{M}, b) = 0$. From the *Verify* description in the previous section, the function outputs zero if the following holds: $\sum_{x \in T} \mathbb{I}[\mathit{Classify}(\tilde{M}, x) \neq T_L x] - \frac{1}{L}|T| > \epsilon|T|$; meaning the number of inputs in the trigger set mapped by $\tilde{M}$ to labels *different* than the pre-defined labels exceeds a small fraction of the trigger set. We go beyond this condition, and introduce the *full removal of the watermark*, with the following two conditions:

(i) $Pr_{x \in D}[\mathit{Classify}(x, \hat{M}) = f(x)] \approx Pr_{x \in D}[\mathit{Classify}(x, \tilde{M}) = f(x)]$

(ii) $\sum_{x \in T} \mathbb{I}[\mathit{Classify}(\tilde{M}, x) = T_L(x)] - \frac{1}{L}|T| \leq \epsilon|T|$

In full removal of the watermark, the attacker's proposed model $\tilde{M}$, still achieves a close accuracy to the marked model's on the test set. However, in this definition, the number of inputs in the trigger set mapped by $\tilde{M}$ to the corresponding pre-defined labels does not exceed a random labels assignment's result by more than a small fraction, i.e., any trace of the watermark is removed.

5.4.1 Black-box Attack

In our black-box attack, we steal the functionality by querying the watermarked model with inputs having a similar distribution to the main distribution discussed earlier and then training a surrogate model on it. Since this distribution does not include any watermarks by nature, the stolen model only copies the backdoor-free functionality. Our attack requires limited number of training inputs, and although computationally inefficient, saves on the work-intensive data preparation task. Our black-box attack does not assume any access to the trigger set, any labeled data including the training data or the parameters of the watermarked model $\hat{M}$. Our attack solely relies on the public information. We query the watermarked model $\hat{M}$ with input $\tilde{D}$ and use the classification label[4] as data labels, to train a derived model as illustrated in Figure 5.3. Note that $\tilde{D}$ is distinct from the watermarked model $\hat{M}$'s training data D, but is from the same application domain. We show our attack

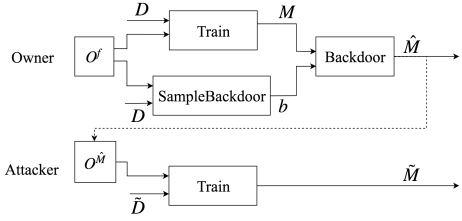


Figure 5.3: A schematic illustration of our black-box attack.

model through the following black-box, full watermark removal game. The $\mathcal{O}^{\hat{M}}$ in the game indicates the black-box access to $\hat{M}$ through a prediction API.

1. Compute $(M, \hat{M}, T, T_L) \leftarrow MModel()$
2. Run $\mathcal{A}$ and compute $\tilde{M} \leftarrow \mathcal{A}(\mathcal{O}^{\hat{M}})$
3. $\mathcal{A}$ wins if:
   (i) $Pr_{x\in D}[Classify(x, M) = f(x)]$
   $\approx Pr_{x\in D}[Classify(x, \tilde{M}) = f(x)]$
   (ii) $\sum_{x\in T} \mathbb{I}[Classify(\tilde{M}, x) = T_L x] - \frac{1}{L}|T| \leq \epsilon|T|$

The attacker $\mathcal{A}$ makes queries to $\hat{M}$ and train its model $\tilde{M}$ based on $\hat{M}$'s responses. $\mathcal{A}$ wins if it achieves the accuracy of the original model and removes the watermark fully.

[4]Just the final class, unlike [152] we do not need the probability vector

5.4.2 White-Box Attack

The black-box attack we proposed in the previous section, does not require any information about the model parameters. However, we show that if the attacker $\mathcal{A}$ is guaranteed access to the model parameters, as is the default assumption in the unremovability game of Adi et al. [10], they can remove the watermark more efficiently. We model the white-box attack by the following white-box full watermark removal game, which is the same as the black-box model except the oracle access to the model $\mathcal{O}^{\hat{M}}$ is replaced by a direct access to $\hat{M}$.

1. Compute $(M, \hat{M}, T, T_L) \leftarrow MModel()$
2. Run $\mathcal{A}$ and compute $\tilde{M} \leftarrow \mathcal{A}(\hat{M})$
3. $\mathcal{A}$ wins if:
   (i) $Pr_{x \in D}[Classify(x, M) = f(x)]$
   $\approx Pr_{x \in D}[Classify(x, \tilde{M}) = f(x)]$
   (ii) $\sum_{x \in T} \mathbb{I}[Classify(\tilde{M}, x) = T_L x] - \frac{1}{L}|T| \leq \epsilon |T|$

Recall that our goal is to prevent the model from learning the misclassifications which are essential to watermarking. As mentioned earlier, we see the watermarking samples as outliers to the main distribution, and believe that the marked model overfits to learn misclassifying them. In order to remove the watermark, we apply regularization to the marked model to *normalize* the weights and avoid overfitting [23]. Our white-box attack,

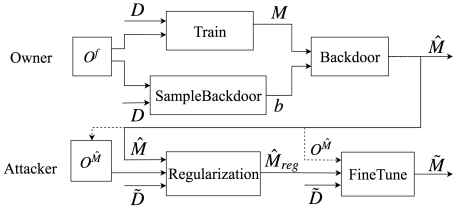


Figure 5.4: A schematic illustration of our white-box attack.

illustrated in Figure 5.4, consists of the two following sub algorithms: regularization and fine-tuning. The input for both sub-algorithms is $\tilde{D}$, which is of the same domain as but distinct from D.

1. $\mathcal{A}_{Reg}(\hat{M}, \mathcal{O}^{\hat{M}}) \rightarrow \hat{M}_{Reg}$
2. $\mathcal{A}_{Fine}(\hat{M}_{Reg}, \mathcal{O}^{\hat{M}}) \rightarrow \tilde{M}$

The first sub-algorithm $\mathcal{A}_{Reg}$ performs the regularization on $\hat{M}$. Since we do not know which layer contributes to learning the watermark misclassification, we define regularization to impact all layers to prevent overfitting to the backdoors. Our experiments show that $\mathcal{A}_{Reg}$ removes the watermark fully by using L2 regularization. However, it affects the test accuracy compared to the original model M. To compensate for this accuracy reduction, the output of $\mathcal{A}_{Reg}$ is then fed to $\mathcal{A}_{Fine}$ to be fine-tuned on an unmarked training set. We emphasize that our white-box attack does not require any information of the ground-truth function or the trigger set for winning the game. Instead, it uses a random set of inputs from the domain of the original model and queries the model $\hat{M}$ to label them. Our experiments show that this attack is significantly more efficient than training a new model and achieves the same accuracy.

5.5 Experiments

We present the results of applying our black-box and white-box attacks on watermarking schemes of Section 5.2.3; these results confirm that our attacks successfully remove the watermarks.

5.5.1 Experiment Setup

In Section 5.4, we introduced our attacks through full watermark removal games between a challenger *MModel* and the attacker $\mathcal{A}$. We simulate both entities in this section and run experiments according to the described algorithms. We use the computing infrastructure with the following features: Intel(R) Xeon(R) CPU E5-2650 0 @ 2.00GHz, 255GB RAM, Driver Version: 418.40.04, and CUDA Version: 10.1. We use GPU model Tesla P100 16GB for Image-Net training and use Tesla K10.G2.8GB for others.

Data Sets and Models

We evaluate our attacks over four popular data sets in DNN literature: MNIST, CIFAR-10, CIFAR-100, and Image-Net. Our data pre-processing includes data normalization as well as data augmentation. Data augmentation is performed by random rotation, width and height shift, and horizontal flip. For MNIST dataset, we use LeNet model [117] and train it on 60K training images and test it on 10K test images. For CIFAR-10, we use VGG-16 and train the model on 50K training images and test it on 10K test images. For MNIST and CIFAR-10 datasets we split the training data in half for the attacker and owner. Our mini-batches contain 64 elements and we use the *RMSProp* [23] optimizer with learning rate of 0.001. While training any model, we use *Early Stopping* [23] on the training accuracy with a min-delta of 0.1% and a patience of 2. Unlike the other two datasets, for CIFAR-100 and Image-Net we use ResNet-32 to train the model. For CIFAR-100 and Image-Net we use overlapping training data for the attacker and owner, we discuss the reasons in Section

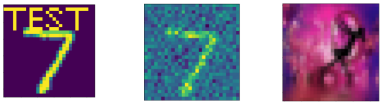


Figure 5.5: Watermark samples in *MModel* (a) Content (b) Noise (c) Abstract Images.

5.5.4. We use a batch size of 100, an SGD optimizer with an initial learning rate of 0.1 and momentum of 0.9. We adjust the learning rate by dividing it by 10 every time the training plateaus. For the white-box attack we use a constant number of 10 epochs for regularization. We use 0 - 1, a.k.a. "min-max feature scaling", normalization [23] for all datasets.

Original and Marked Model Generation

We first simulate the *MModel* algorithm in our full watermark removal games to generate the original model M, the watermarked model $\hat{M}$, and the watermark consisting of the watermark set T and its corresponding labels T_L. The watermark set for different schemes, shown through examples in Figure 5.5, is constructed according to Section 5.2.3. *MModel* trains a model $\hat{M}$ with a portion the of the watermark set and a portion of the remaining training set. Note that the rest of the two sets is needed to form *test set* and *watermark test set*. For each experiment, we use randomly selected subsets of the trigger set for training the watermark model. Recall from Section 5.2.3 that the watermark set and the watermark test set are the same for the Abstract Images watermarking scheme, but not for the Embedded Content or the Pre-specified Noise scheme.

Attack Algorithm $\mathcal{A}$ and Generating $\tilde{M}$

In both our black-box and white-box attacks, the algorithm $\mathcal{A}$ aims to derive a model $\tilde{M}$ that keeps the same test accuracy as marked model $\hat{M}$, while it reduces the watermark retention to $\frac{1}{|L|}$, where $|L|$ is the total number of valid classes. This reduction, shows that the model associates the watermarked input to the pre-defined class, not more than a random classifier would do, hence indicating success in removing the watermark fully. To generate $\tilde{M}$, neither of our attacks uses the original model M's training data with the ground-truth labels, nor any of the watermarking information. Instead, they both query the watermark model $\hat{M}$ with inputs from the publicly known domain of $\hat{M}$ and train $\tilde{M}$ with the corresponding labels. The white-box attack initializes $\tilde{M}$ with $\hat{M}$'s parameters, then undergoes regularization followed by fine-tuning with public data labeled by $\hat{M}$.

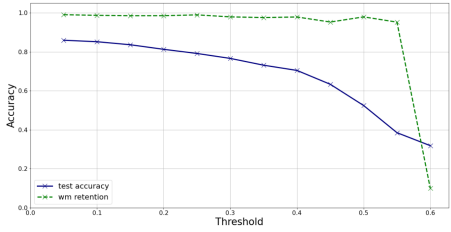


Figure 5.6: Fine-Pruning attack on CIFAR-10, for Embedded Content

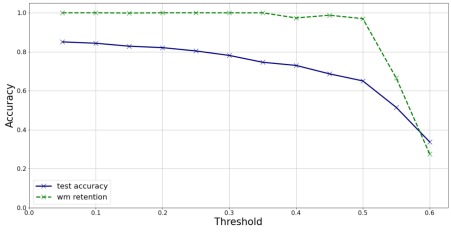


Figure 5.7: Fine-Pruning attack on CIFAR-10, for Pre-specified Noise

Security and Performance Evaluation

In what follows, we present our black-box and white-box attack with concrete parameters in their setup and evaluation. As mentioned earlier in this section, our security evaluation metrics are: i) test accuracy and ii) watermark retention. For test accuracy, we compare the accuracy of the model generated by our attack with that of the target model on classifying an unseen test set. For watermark retention, we measure what portion of a set of marked inputs are classified as their pre-defined labels by the models generated through our attacks. We also evaluate the performance of our attacks, based on the time they take to run rather than the number of epochs. The number of epochs depends on some factors in the model training - such as the input size, while the time is an independent measure. For example, an epoch in the fine-tuning phase (Section 5.4.2) takes much more time than an epoch in the regularization, as the size of the training set in fine-tuning is at least ten times the training set size in the regularization.

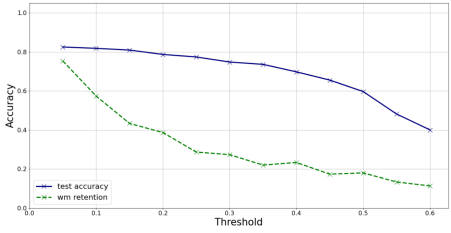


Figure 5.8: Fine-Pruning attack on CIFAR-10, for Abstract Images

5.5.2 Fine-Pruning Attack

Fine-pruning [124] uses the black-box labels to prune the dormant neurons — the neurons with activation frequency below the threshold — to remove backdoors. The pruning is followed by a fine-tuning phase to increase the accuracy.

Our results show that fine-pruning with black-box labels fails at simultaneously removing embedded watermarks and preserving the test accuracy. Figures 5.6 - 5.8 shows the accuracy and watermark retention of the attack for different threshold values. Note that we do not consider larger threshold values since pruning many neurons results in a significant loss of test accuracy. Specifically, in the best case fine-pruning reduces the watermark retention to 22.2% for Abstract Images while suffering a test accuracy drop of 8.4% for CIFAR-10 dataset. In contrast, our white-box attack lowers the watermark retention to 12.8% with only 1% of accuracy decrease.

We verified that fine-pruning attack does remove the watermarks when the attacker has access to ground-truth labels. However, we do not consider this attack under this strong assumption as a fair comparison to our attacks.

5.5.3 Our Results

Watermark Removal

We present the results of our black-box and white-box attacks in Table 5.1 in this section. We evaluate our attacks on MNIST, CIFAR-10 and Image-Net datasets over each of the watermarking schemes described in Section 5.2.3: Embedded Content, Pre-specified Noise and Abstract Images. As the Abstract Images in its original paper [10] is used for watermarking models trained on CIFAR-100 too, we include experiments for watermark removal for those models as well. Our results show that the black-box attack removes the watermark with a performance comparable to training the original watermarked model, whereas the white-box attack does so with significant speed-up. It is also worth noting that

Table 5.1: Watermark removal attacks on various datasets

Dataset	W.M. Scheme	W.M. Model Acc.	W.M. Model Time	B.B. Post Acc.	B.B. Acc. Drop	B.B. W.M. Ret.	B.B. Attack Time	W.B. Post Acc.	W.B. Acc. Drop	W.B. W.M. Ret.	W.B. Attack Time
MNIST	Content	99%	3.5m	98.7%	0.3%	9.6%	3.6m	98.8%	0.2%	9.2%	0.59m
	Noise	98.9%	2.6m	98.7%	0.2%	8.5%	4.1m	99%	-0.1%	8.4%	0.77m
	Abstract	98.8%	2.6m	98.5%	0.3%	8.6%	5m	98.7%	0.1%	11.4%	2.1m
CIFAR-10	Content	83.1%	166m	79.4%	3.7%	8.1%	244m	82.1%	1%	12.8%	20m
	Noise	83.3%	178m	80.9%	2.4%	3.4%	208m	82.6%	0.7%	7.7%	20m
	Abstract	83.3%	112m	80.3%	3%	20%	158m	82.4%	0.9%	22.9%	33m
CIFAR-100	Abstract	66.4%	203m	65.2%	1.2%	1%	177m	65.8%	0.6%	11.4%	10m
IMAGE-NET	Content	66.1%	3949m	62.0%	4.1%	0.1%	5922m	63.6%	2.5%	9%	1427m
	Noise	66.9%	3958m	61.7%	5.2%	0.1%	5902m	63.6%	3.3%	11%	1419m
	Abstract	66.5%	3960m	61.5%	5%	0.1%	5906m	63.0%	3.5%	12%	1412m
IMAGE-NET*	Content	66.1%	3949m	63.0%	3.1%	0.1%	5385m	N/A	N/A	N/A	N/A
	Noise	66.9%	3958m	64.4%	2.5%	0.1%	5367m	N/A	N/A	N/A	N/A
	Abstract	66.5%	3960m	63.3%	3.2%	0.1%	5758m	N/A	N/A	N/A	N/A

*Indicates we use the probability vectors for black-box attacks

by allowing the fine-tuning to continue for longer, the white-box attack can even achieve higher accuracy than the watermarked model.

For the Image-Net dataset, we also include a black-box attack where the attacker has access to the probability vector of the prediction. We show that by gaining this information, the attacker can steal an Image-Net model with only 3% accuracy drop using the black-box attack. In the white-box attack, we optimized the attack to reach the same level of accuracy and then the attacker only requires 30% of the data samples that the model owner (or the black-box attack) uses for training but removes the watermark within a very short period of time. The accuracy drop of the white-box attacker can be further lowered using larger (unlabeled) data sets. For example, when using 100% of the data samples, the accuracy drop of the white-box attack on an Abstract Images watermark is less than 1%.

5.5.4 Discussion on Experiments and Results

We provide further discussion about the parameters in our attacks here: i) The restrictions that prevent our model from reaching the highest accuracy, ii) The reasons behind our models selection, iii) A deeper investigation and presenting evidences that a model has to choose between a resilient backdoor and high accuracy, and cannot keep both.

Not the Highest Accuracy

Since we are simulating both the challenger and the attacker in our experiments and desire not allowing overlap in their training dataset, our models effectively have access to half of the training dataset. The limitation prevents our models in MNIST and CIFAR-10

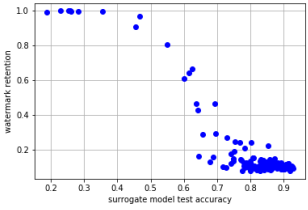


Figure 5.9: Watermark retention VS test accuracy of the watermarked model $\hat{M}$, as the number of epochs increases, MNIST.

experiments from reaching their highest possible accuracy [193]. However, despite this limitation, our attacks still successfully remove the embedded watermarks. For CIFAR-100 and Image-Net, the non-overlapping requirement and reducing the training set size results in more than 10% accuracy loss in the watermarked model itself compared to a non-marked model. Therefore, we allow overlap in training set of both the watermarked and the attacker model for this case.

Model Selection

In our experiments, the attacker uses the same model architecture for $\tilde{M}$ as the owner does for the watermarked target model $\hat{M}$. There are two incentives for this selection: i) as stated by Juuti et al. [98], higher complexity models increase the prediction accuracy until they are as complex as the source model. Hence, on one hand, the attacker cannot use a model with less capacity than the target model or he will lose accuracy. On the other hand, training a model with higher capacity requires more recourse, e.g more number of queries. This discourages the attacker from training a model with higher capacity than the target mode. Therefore, the target model is the optimal point for the attacker. ii) Watermark removal in our attacks depends on learning the main classification, and not learning the intended misclassification for a trigger set which has a different distribution than the main data. As model resemblance likely increases the chance of *backdoor transfer*, we use the same model for the attacker as the owner to increase the chance of watermark retention to the highest possible value, as the worst case for the attacker. Yet, despite our selection of models being the one where the transfer of the watermark is most likely, our attacks successfully remove the watermark leaving this architecture as the best option for the attacker.

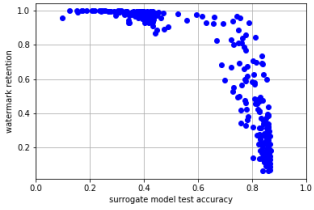


Figure 5.10: Watermark retention VS accuracy of the marked model $\hat{M}$, as the number of epochs increases, CIFAR-10.

Watermark Retention and Test Accuracy

In addition to the successful watermark removal by our attacks, we observed another important result in our experiments. In our black-box attack, we applied the model stealing to a fully trained model $\hat{M}$ with embedded watermarks. To investigate the success of model stealing over partially trained models, we repeatedly applied the attack to the marked model $\hat{M}$ during its training. The resulting test accuracy and watermark retention of the stolen model $\tilde{M}$ are plotted in Figures 5.9 and 5.10 for both MNIST and CIFAR-10 data sets.

The attack reaches lower watermark retention if $\hat{M}$ achieves higher test accuracy. This increase in the model accuracy is a result of overcoming the underfitting by increasing the number of epochs during training $\hat{M}$. An underfitted model $\hat{M}$ transfers its watermark – backdoor or bias towards the backdoor – to the stolen network $\tilde{M}$ and makes it more resilient to watermark removal. Clearly, our black-box attack is successful against any 100% accurate model. Hence, watermark retention can only be introduced by inaccuracies in the model. However, our results indicate that these inaccuracies need to be quite significant to result in successful watermark retention for backdoor-based watermarks.

5.6 Related Work

Black-box Watermarking Schemes. The demand in protecting neural networks that are solely accessible through a remote API, encouraged black-box watermarking in DNNs [34]. DeepSigns [161] embeds watermarks in the probability density function of the activation set of the target layer and introduces two versions of the framework to provide watermarking in both white-box and black-box setting. In two other approaches [35, 138], the authors use adversarial examples in a zero-bit watermarking algorithm to enable extraction of the watermark without requiring model parameters. This approach

however, requires limitation on transferability of the utilized adversarial examples across other networks. Backdoor-based watermarking [10, 75, 120, 145, 195] – as investigated in this work – is another recent line of work in black-box watermarking that feed a secret trigger set and its pre-defined labels to the model during training to protect ownership.

Backdoor Removal. Since backdooring a neural network may also impose other threats, identifying and removing them has gained attention in research. However, typically such systems [42, 61, 127] are intended to be employed alongside the neural network. They are tasked only to fend off attempts of actively using the embedded backdoor, which is not applicable for the scenario of attacking watermarking schemes since the trigger set is never released. There are few schemes [124, 184] that do not require access to the trigger set at any time. The Neural-Cleanse approach [184] first detects whether a backdoor exists in the model by checking how many pixels in the input image should be modified for the prediction to change to another class. When there is one such consistent small modification for many benign inputs, it is assumed to be a backdoor and then it undergoes reverse-engineering and mitigation processes. This approach only works with backdoors that are restricted to a small patch of the image, which is not the case for all watermark types in this work. The Fine-Pruning approach [124] removes backdoors by pruning redundant neurons. This attack is discussed in details in Section 5.5.2. We show that fine pruning attack only removes backdoors with access to correctly labeled data.

Removing Backdoor-based watermarks. In a different approach to remove backdoor-based watermarks, Hitaj et al. [91] introduce an Ensemble and an Evasion attack to remove backdoor-based watermarks. The Ensemble attack, steals n models and collects responses from all of them for each query. It then selects the answer that receives the highest vote among the responds from the stolen networks, and provides that as API prediction. In comparison to this attack, our attacks solely require one marked model and produces a *clean* model that can be redistributed. In the Evasion attack in [91], a detector mechanism blocks the verification of watermarks. The service will return a random class prediction when it suspects a query is a watermark-trigger. This approach will not be effective in backdoor removal if the trigger samples have a similar distribution to the original samples [196]. Chen et al. [40] remove the backdoor-based watermark by fine-tuning the model, arguing that previous results [10] on the resilience of backdoor-based watermarking schemes against fine-tuning was due to the low value of learning rate. They use a combination of labeled ($20\% - 80\%$) and unlabeled data to remove the watermark. Our attacks break the security of backdoor-based watermarking schemes without the need to access labeled data, multiple models, or evading the queries that aim at watermark verification.

Model Stealing Attacks. Model stealing attacks were generalized by Orekondy et al. [152], with less requirements than similar works in the literature [98, 154]. We use a similar approach to Orekondy et al. in our black-box attack, with two differences: i) we query the target model with data of a similar distribution to the model's training data rather than random images, and ii) we just need the final label for the queries instead of their probability vectors. Limiting access to final labels instead of probability vectors is a proposed defence against previous model stealing attacks [154]. Our black-box attack achieves accuracy close to the marked model which suggests that this proposed defence is

insufficient against model stealing attacks.

The work presented in this chapter inspired a follow-up research by Lukas et al. [130] that is published by the time of finalizing this thesis. Lukas et al. propose taxonomies for watermarking schemes and removal attacks.

5.7 Conclusion

We present two attacks on the recent backdoor-based watermarking schemes in deep neural networks: i) black-box attack, and ii) white-box attack. The watermark in the targeted model could take any of the forms: i) a logo, or an embedded content, ii) a pre-specified noise pattern, or iii) abstract images. Our black-box and white-box attacks do not require access to the trigger set, the ground-truth function of the watermarked model or any labeled data. This saves enormous amount of time and resources in preparing the training data. Our attacks use limited number of inputs from the publicly known domain of the marked model, and query the model for labels. They remove the models' watermark successfully with negligible drop on the classification accuracy. Our black-box approach achieves these goals with minimum access requirements and by solely exploiting the models' classification labels. However, granting more information (e.g. the marked model's parameters) enables us to devise a white-box attack that is more efficient and accurate than our black-box attack.

Chapter 6

Conclusion

In this thesis, we addressed four security/privacy problems in distinct phases of the data science pipeline. In Chapter 2 we contributed to privacy in data preparation and proposed PCOR. PCOR is a solution for privacy violation in contextual outlier detection that balances the privacy, utility and performance trade-off. Defining differential privacy for outlier detection is different from statistical queries such as sum or average. It is particularly challenging, since a target data point can be an outlier in some contexts while it is a non-outlier in the rest. We tackle this challenge by using a relaxed notion of differential privacy, namely, output constrained differential privacy. Our results indicate that this relaxation is a not a strong requirement and is satisfied in most cases in practice. This approach can also be applied for defining privacy in other challenging database queries as well. In addition, we introduced differentially private graph search algorithms in our work to achieve higher efficiencies. These algorithms also can be utilized for sampling in similar graph-based private solutions for a significant improvement in performance.

In Chapter 3, contributing to data modeling and analysis, we showed the limits of differential privacy in providing a meaningful notion of privacy in social graphs. We explored the problem of preserving organizational confidentiality in language tasks, and demonstrated the data correlation among edges in the organization's social network. On the one hand, these non-IID data points turn the record-level differential privacy into a fake privacy promise. On the other hand, its straightforward remedy, i.e., group differential privacy, is overprotective and affects accuracy significantly. We proposed a middle-ground scheme between these two extreme measures of privacy in queries over n-grams, by leveraging the Pufferfish privacy framework that takes data correlation into account. Our scheme provides a starting point for a more meaningful notion of privacy than record-level differential privacy, while improving the utility compared to group privacy. We modeled correlation up to and including very large neighborhoods of edges. This limited our utility improvement and leaves it as an open problem how other correlation models, or more adjusted graph representations could improve the utility even more. It also requires further research to assess how our mechanism can be extended to model training tasks.

In Chapter 4, we addressed a security problem in outsourcing data management to the cloud. We considered data management operations such as joins that cannot be simply

performed over encrypted data and require specialized encryption. The development of these specialized encryption schemes over the years reduced the leakage from frequency information to only equality condition for tuples that match a selection criterion, yet ignoring the fact that the leakage of a series of queries may be larger than the union (sum) of the leakage of each query. We presented a new join encryption scheme that prevents this additional leakage. Our construction uses function-hiding inner product encryption and can run hash-based joins. As a result, it achieves comparable performance with the state-of-the-art join encryption schemes while providing better security. This performance could be improved even more with parallelization. We believe that our construction achieves a natural lower bound for the leakage in an efficient, non-interactive, single-server setting. We leave it as an open problem that which restrictions to remove in order to further reduce the leakage of join encryption schemes.

In Chapter 5 we analysed watermarking in deep neural networks as a data protection technique in data dissemination. We considered backdoor-based watermarking, a state-of-the-art black-box watermarking scheme. We showed that these schemes fail in satisfying their security claims such as: i) once they [the marked models] are stolen and deployed to offer AI service, owners can easily verify them [the watermarks] by sending watermarks as inputs and checking the service's output, ii) it is hard to remove a backdoor, unless one has knowledge of the trigger set, iii) it is impossible to remove the watermark even with guaranteed full access to the ground-truth function while restricting the runtime of the adversary, or giving unlimited power to the adversary while restricting its access to the ground-truth function, iv) transfer learning is on the same order of magnitude as the cost of training, if not higher, v) stealing a model using prediction APIs needs queries of significant size; e.g. $100k$, where k is the number of model parameters, etc. We showed the invalidity of these claims by introducing our black-box attack that does not require access to the trigger set, the ground-truth function of the watermarked model or any labeled data. Our attack uses limited number of inputs from the publicly known domain of the marked model, and queries the model for labels. It removes the models' watermarks successfully with negligible drop on the classification accuracy. We also propose a white-box attack, in case of access to the marked model's parameters, which provides higher efficiency and accuracy. Both of our attacks can be used in evaluating the robustness of other black-box watermarking schemes as well.

This thesis provides examples of technical weaknesses in data protection algorithms and protocols. However, we emphasize that data breaches and privacy violations could also occur as a result of other deficiencies, ranging from poor implementations such as those inducing vulnerability to side channel attacks, to human factors such as social engineering. While out of the scope of this thesis, these deficiencies require intensive research as well, for providing a comprehensive guarantee of data protection.

References

[1] The murder accountability project. https://www.kaggle.com/murderaccountability/homicide-reports#database.csv. Accessed: 2019-12-15.

[2] Ontario's public sector salary dataset. https://www.ontario.ca/page/public-sector-salary-disclosure. Accessed: 2019-07-31.

[3] Scikit-learn machine learning in python. http://scikit-learn.org/stable/auto_examples/neighbors/plot_lof.html. Accessed: 2019-07-31.

[4] *Pearson, Karl*, pages 418–419. Springer New York, New York, NY, 2008.

[5] Regulation (eu) 2016/679 of the european parliament and of the council of 27 april 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data,and repealing directive 95/46/ec (general data protection regulation), 2016-06-27.

[6] Aydin Abadi, Sotirios Terzis, and Changyu Dong. O-psi: delegated private set intersection on outsourced datasets. In *IFIP International Information Security and Privacy Conference*, pages 3–17. Springer, 2015.

[7] Martin Abadi, Andy Chu, Ian Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, page 308–318, New York, NY, USA, 2016. Association for Computing Machinery.

[8] Martin Abadi, Andy Chu, Ian Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, page 308–318, New York, NY, USA, 2016. Association for Computing Machinery.

[9] Martin Adam, Michael Wessel, and Alexander Benlian. Ai-based chatbots in customer service and their effects on user compliance. *Electronic Markets*, pages 1–19, 2020.

[10] Yossi Adi, Carsten Baum, Moustapha Cisse, Benny Pinkas, and Joseph Keshet. Turning your weakness into a strength: Watermarking deep neural networks by backdooring. In *27th USENIX Security Symposium (USENIX Security 18)*, pages 1615–1631, Baltimore, MD, 2018. USENIX Association.

[11] Rakesh Agrawal, Dmitri Asonov, Murat Kantarcioglu, and Yaping Li. Sovereign joins. In *22nd International Conference on Data Engineering (ICDE'06)*, pages 26–26. IEEE, 2006.

[12] Abdulatif Alabdulatif, Heshan Kumarage, Ibrahim Khalil, and Xun Yi. Privacy-preserving anomaly detection in cloud with lightweight homomorphic encryption. *Journal of Computer and System Sciences*, 90:28 – 45, 2017.

[13] Miguel E. Andrés, Nicolás E. Bordenabe, Konstantinos Chatzikokolakis, and Catuscia Palamidessi. Geo-indistinguishability: differential privacy for location-based systems. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, CCS '13, pages 901–914, New York, NY, USA, 2013. ACM.

[14] Arvind Arasu and Raghav Kaushik. Oblivious query processing. In *Proc. 17th International Conference on Database Theory (ICDT), Athens, Greece, March 24-28, 2014*, pages 26–37, 2014.

[15] Michael Backes, Mathias Humbert, Jun Pang, and Yang Zhang. Walk2friends: Inferring social links from mobility profiles. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17, page 1943–1957, New York, NY, USA, 2017. Association for Computing Machinery.

[16] Johes Bater, Gregory Elliott, Craig Eggen, Satyender Goel, Abel Kho, and Jennie Rogers. Smcql: secure querying for federated databases. *Proceedings of the VLDB Endowment*, 10(6):673–684, 2017.

[17] Ghazaleh Beigi and Huan Liu. A survey on privacy in social media: Identification, mitigation, and applications. *ACM/IMS Trans. Data Sci.*, 1(1), March 2020.

[18] Mihir Bellare, Alexandra Boldyreva, and Adam O'Neill. Deterministic and efficiently searchable encryption. In *Annual International Cryptology Conference*, pages 535–552. Springer, 2007.

[19] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. A neural probabilistic language model. *J. Mach. Learn. Res.*, 3:1137–1155, March 2003.

[20] E. Bertino and E. Ferrari. Big data security and privacy. In *A Comprehensive Guide Through the Italian Database Research*, 2018.

[21] K. Bhaduri, M. D. Stefanski, and A. N. Srivastava. Privacy-preserving outlier detection through random nonlinear data distortion. *Trans. Sys. Man Cyber. Part B*, 41(1):260–272, February 2011.

[22] Smriti Bhagat, Graham Cormode, Balachander Krishnamurthy, and Divesh Srivastava. Class-based graph anonymization for social network data. 2(1):766–777, August 2009.

[23] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2006.

[24] J. Blocki, A. Blum, A. Datta, and O. Sheffet. Differentially private data analysis of social networks via restricted sensitivity. In *ITCS*, page 87–96, 2013.

[25] Jonas Böhler, Daniel Bernau, and Florian Kerschbaum. Privacy-preserving outlier detection for data streams. In Giovanni Livraga and Sencun Zhu, editors, *Data and Applications Security and Privacy XXXI*, pages 225–238, Cham, 2017. Springer International Publishing.

[26] Alexandra Boldyreva, Serge Fehr, and Adam O'Neill. On notions of security for deterministic encryption, and efficient constructions without random oracles. In *Annual International Cryptology Conference (CRYPTO)*, pages 335–359. Springer, 2008.

[27] Dan Boneh and Matt Franklin. Identity-based encryption from the weil pairing. In Joe Kilian, editor, *Advances in Cryptology — CRYPTO 2001*, pages 213–229, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[28] Markus Breunig, Hans-Peter Kriegel, Raymond T. Ng, and Jörg Sander. Lof: Identifying density-based local outliers. In *PROCEEDINGS OF THE 2000 ACM SIGMOD INTERNATIONAL CONFERENCE ON MANAGEMENT OF DATA*, pages 93–104. ACM, 2000.

[29] Xavier Bultel, Radu Ciucanu, Matthieu Giraud, Pascal Lafourcade, and Lihua Ye. Secure joins with mapreduce. In *International Symposium on Foundations and Practice of Security*, pages 78–94. Springer, 2018.

[30] Bogdan Carbunar and Radu Sion. Toward private joins on outsourced data. *IEEE transactions on knowledge and data engineering*, 24(9):1699–1710, 2011.

[31] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM Comput. Surv.*, 41(3):15:1–15:58, July 2009.

[32] Bee-Chung Chen, Daniel Kifer, Kristen LeFevre, and Ashwin Machanavajjhala. Privacy-preserving data publishing. 2(1–2):1–167, January 2009.

[33] Huili Chen, Bita Darvish Rohani, and Farinaz Koushanfar. Deepmarks: A digital fingerprinting framework for deep neural networks. *arXiv preprint arXiv:1804.03648*, 2018.

[34] Huili Chen, Bita Darvish Rouhani, Xinwei Fan, Osman Cihan Kilinc, and Farinaz Koushanfar. Performance comparison of contemporary dnn watermarking techniques. *arXiv preprint arXiv:1811.03713*, 2018.

[35] Huili Chen, Bita Darvish Rouhani, and Farinaz Koushanfar. Blackmarks: Blackbox multibit watermarking for deep neural networks. *arXiv preprint arXiv:1904.00344*, 2019.

[36] Mia Xu Chen, Benjamin N Lee, Gagan Bansal, Yuan Cao, Shuyuan Zhang, Justin Lu, Jackie Tsay, Yinan Wang, Andrew M Dai, Zhifeng Chen, et al. Gmail smart compose: Real-time assisted writing. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD'19, page 196–206, New York, NY, USA, 2019. Association for Computing Machinery.

[37] Rui Chen, Benjamin C.M.Fung, Philip S.Yu, and Bipin C.Desai. Correlated network data publication via differential privacy. *The VLDB Journal*, 2014.

[38] S. Chen and S. Zhou. Recursive mechanism: Towards node differential privacy and unrestricted joins. In *SIGMOD*, page 653–664, 2013.

[39] Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Xiaodong Song. Targeted backdoor attacks on deep learning systems using data poisoning. *CoRR*, abs/1712.05526, 2017.

[40] Xinyun Chen, Wenxiao Wang, Yiming Ding, Bender Chries, Jia Ruoxi, and Dawn Song. Leveraging unlabeled data for watermark removal of deep neural networks. In *ICML workshop on Security and Privacy of Machine Learning*, 2019.

[41] James Cheng, Ada Fu, and Jia Liu. K-isomorphism: Privacy preserving network publication against structural attacks. pages 459–470, 01 2010.

[42] Edward Chou, Florian Tramèr, Giancarlo Pellegrino, and Dan Boneh. Sentinet: Detecting physical attacks against deep learning systems. *arXiv preprint arXiv:1812.00292*, 2018.

[43] B. Collingsworth and R. Menezes. *Identification of Social Tension in Organizational Networks*, pages 209–223. 04 2009.

[44] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, Third Edition*. The MIT Press, 3rd edition, 2009.

[45] Reza Curtmola, Juan A. Garay, Seny Kamara, and Rafail Ostrovsky. Searchable symmetric encryption: improved definitions and efficient constructions. In *Proceedings of the 13th ACM Conference on Computer and Communications Security, (CCS)*, pages 79–88, 2006.

[46] Jay L. Devore. *Probability and Statistics for Engineering and the Sciences*. Brooks/Cole, 8th edition, January 2011. ISBN-13: 978-0-538-73352-6.

[47] Jinshuo Dong, David Durfee, and Ryan Rogers. Optimal differential privacy composition for exponential mechanisms. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119

of *Proceedings of Machine Learning Research*, pages 2597–2606. PMLR, 13–18 Jul 2020.

[48] L. T. Dung and H. T. Bao. A distributed solution for privacy preserving outlier detection. In *2011 Third International Conference on Knowledge and Systems Engineering*, pages 26–31, Oct 2011.

[49] David Durfee and Ryan Rogers. Practical differentially private top-k selection with pay-what-you-get composition. *CoRR*, abs/1905.04273, 2019.

[50] Cynthia Dwork. Differential privacy. In Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener, editors, *Automata, Languages and Programming*, pages 1–12, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[51] Cynthia Dwork. Differential privacy: A survey of results. In Manindra Agrawal, Dingzhu Du, Zhenhua Duan, and Angsheng Li, editors, *Theory and Applications of Models of Computation*, pages 1–19, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[52] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Third Theory of Cryptography Conference (TCC 2006)*, volume 3876 of *Lecture Notes in Computer Science*, pages 265–284. Springer, March 2006.

[53] Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.*, 9(3–4):211–407, August 2014.

[54] Cynthia Dwork, Guy N. Rothblum, and Salil Vadhan. Boosting and differential privacy. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, pages 51–60, 2010.

[55] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. Rappor: Randomized aggregatable privacy-preserving ordinal response. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, CCS '14, pages 1054–1067, New York, NY, USA, 2014. ACM.

[56] Ronald Fagin, Moni Naor, and Peter Winkler. Comparing information without leaking it. *Communications of the ACM*, 39(5):77–85, 1996.

[57] Crowd Flower. 2016 data science report, Mar 2016.

[58] Michael J Freedman, Kobbi Nissim, and Benny Pinkas. Efficient private matching and set intersection. In *International conference on the theory and applications of cryptographic techniques*, pages 1–19. Springer, 2004.

[59] Karan Ganju, Qi Wang, Wei Yang, Carl A Gunter, and Nikita Borisov. Property inference attacks on fully connected neural networks using permutation invariant representations. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 619–633. ACM, 2018.

[60] Tianchong Gao, Feng Li, Yu Chen, and XuKai Zou. Preserving local differential privacy in online social networks. In *International Conference on Wireless Algorithms, Systems, and Applications*, pages 393–405. Springer, 2017.

[61] Yansong Gao, Chang Xu, Derui Wang, Shiping Chen, Damith C Ranasinghe, and Surya Nepal. Strip: A defence against trojan attacks on deep neural networks. *arXiv preprint arXiv:1902.06531*, 2019.

[62] Chang Ge, Xi He, Ihab F. Ilyas, and Ashwin Machanavajjhala. Apex: Accuracy-aware differentially private data exploration. In *Proceedings of the 2019 International Conference on Management of Data*, SIGMOD '19, pages 177–194, New York, NY, USA, 2019. ACM.

[63] Johannes Gehrke, Michael Hay, Edward Lui, and Rafael Pass. Crowd-blending privacy. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology – CRYPTO 2012*, pages 479–496, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[64] Johannes Gehrke, Edward Lui, and Rafael Pass. Towards privacy for social networks: A zero-knowledge based definition of privacy. In Yuval Ishai, editor, *Theory of Cryptography*, pages 432–449, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[65] Arpita Ghosh and Robert Kleinberg. Inferential privacy guarantees for differentially private mechanisms. abs/1603.01508, 2018.

[66] Arpita Ghosh and Aaron Roth. Selling privacy at auction. *Games and Economic Behavior*, 91:334 – 346, 2015.

[67] Yoav Goldberg. A primer on neural network models for natural language processing. *J. Artif. Int. Res.*, 57(1):345–420, September 2016.

[68] Yoav Goldberg and Omer Levy. word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method. 02 2014.

[69] Sivakanth Gopi, Pankaj Gulhane, Janardhan Kulkarni, Judy Hanwen Shen, Milad Shokouhiand, and Sergey Yekhanin. Differentially private set union. *ICML*, 2020.

[70] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *Proceedings of the 13th ACM conference on Computer and communications security (CCS)*, pages 89–98, 2006.

[71] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, 38, 03 2013.

[72] A. Greenberg. Apple's 'differential privacy' is about collecting your data — but not your data. 2016.

[73] Frank E. Grubbs. Procedures for detecting outlying observations in samples. *Technometrics*, 11(1):1–21, 1969.

[74] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *CoRR*, abs/1708.06733, 2017.

[75] Jia Guo and Miodrag Potkonjak. Watermarking deep neural networks for embedded systems. In *2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 1–8. IEEE, 2018.

[76] Hakan Hacigumus, Bala Iyer, and Sharad Mehrotra. Providing database as a service. In *Proceedings 18th International Conference on Data Engineering*, pages 29–38. IEEE, 2002.

[77] Hakan Hacigümüs, Balakrishna R. Iyer, Chen Li, and Sharad Mehrotra. Executing SQL over encrypted data in the database-service-provider model. In *Proceedings of the ACM International Conference on Management of Data (SIGMOD)*, pages 216–227, 2002.

[78] F. Hahn, N. Loza, and F. Kerschbaum. Joins over encrypted data with fine granular security. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*, pages 674–685, April 2019.

[79] Jiawei Han. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2005.

[80] Samuel Haney, Ashwin Machanavajjhala, John M. Abowd, Matthew Graham, Mark Kutzbach, and Lars Vilhuber. Utility cost of formal privacy for releasing national employer-employee statistics. In *Proceedings of the 2017 ACM International Conference on Management of Data*, SIGMOD '17, pages 1339–1354, New York, NY, USA, 2017. ACM.

[81] Isabelle Hang, Florian Kerschbaum, and Ernesto Damiani. Enki: access control for encrypted query processing. In *Proceedings of the ACM International Conference on Management of Data (SIGMOD)*, pages 183–196, 2015.

[82] Awni Y. Hannun, Carl Case, Jared Casper, Bryan Catanzaro, Greg Diamos, Erich Elsen, Ryan Prenger, Sanjeev Satheesh, Shubho Sengupta, Adam Coates, and Andrew Y. Ng. Deep speech: Scaling up end-to-end speech recognition. *CoRR*, abs/1412.5567, 2014.

[83] Michael Hay, Chao Li, Gerome Miklau, and David Jensen. Accurate estimation of the degree distribution of private networks. In *Proceedings of the 2009 Ninth IEEE International Conference on Data Mining*, ICDM '09, page 169–178, USA, 2009. IEEE Computer Society.

[84] Michael Hay, Gerome Miklau, David Jensen, Donald Towsley, and Philipp Weis. Resisting structural identification in anonymized social networks. *PVLDB*, 1:102–114, 08 2008.

[85] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, June 2016.

[86] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[87] X. He, A. Machanavajjhala, and B. Ding. Blowfish privacy: Tuning privacy-utility trade-offs using policies. SIGMOD '14, page 1447–1458, New York, NY, USA, 2014. Association for Computing Machinery.

[88] Xi He, Ashwin Machanavajjhala, Cheryl Flynn, and Divesh Srivastava. Composing differential privacy and secure computation: A case study on scaling private record linkage. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17, pages 1389–1406, New York, NY, USA, 2017. ACM.

[89] Rebecca Hellerstein. Who bears the cost of a change in the exchange rate? pass-through accounting for the case of beer. *Journal of International Economics*, 76(1):14–32, 2008.

[90] Geoffrey Hinton, Li Deng, Dong Yu, George Dahl, Abdel rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara Sainath, and Brian Kingsbury. Deep neural networks for acoustic modeling in speech recognition. *Signal Processing Magazine*, 2012.

[91] Dorjan Hitaj and Luigi V Mancini. Have you stolen my model? evasion attacks against deep neural network watermarking techniques. *arXiv preprint arXiv:1809.00615*, 2018.

[92] F. L. Hitchcock. The distribution of a product from several sources to numerous localities. *Journal of Mathematics and Physics*, 20(1-4):224–230, 1941.

[93] Ihab F. Ilyas and Xu Chu. Data cleaning, 2019-06-17.

[94] Mihaela Ion, Ben Kreuter, Ahmet Erhan Nergiz, Sarvar Patel, Shobhit Saxena, Karn Seth, Mariana Raykova, David Shanahan, and Moti Yung. On deploying secure computing: Private intersection-sum-with-cardinality. In *2020 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 370–389. IEEE, 2020.

[95] Noah Johnson, Joseph P. Near, and Dawn Song. Towards practical differential privacy for sql queries. *Proc. VLDB Endow.*, 11(5):526–539, January 2018.

[96] Karen Sparck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, 1972.

[97] Antoine Joux. A one round protocol for tripartite diffie–hellman. In Wieb Bosma, editor, *Algorithmic Number Theory*, pages 385–393, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg.

[98] Mika Juuti, Sebastian Szyller, Samuel Marchal, and N. Asokan. Prada: Protecting against dnn model stealing attacks. In *2019 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 512–527, 2019.

[99] Peter Kairouz, Sewoong Oh, and Pramod Viswanath. The composition theorem for differential privacy. *IEEE Transactions on Information Theory*, 63(6):4037–4049, 2017.

[100] Murat Kantarcioglu, Ali Inan, Wei Jiang, and Bradley Malin. Formal anonymity models for efficient privacy-preserving joins. *Data & Knowledge Engineering*, 68(11):1206–1223, 2009.

[101] Vishesh Karwa, Sofya Raskhodnikova, Adam Smith, and Grigory Yaroslavtsev. Private analysis of graph structure. *ACM Trans. Database Syst.*, 2014.

[102] S.P. Kasiviswanathan, K. Nissim, and S.Raskhodnikova. Analyzing graphs with node differential privacy. In *TCC*, page 81–98, 2013.

[103] Michael Kearns, Aaron Roth, Zhiwei Steven Wu, and Grigory Yaroslavtsev. Private algorithms for the protected in social network search. *Proceedings of the National Academy of Sciences*, 113(4):913–918, 2016.

[104] Florian Kerschbaum. Collusion-resistant outsourcing of private set intersection. In *Proceedings of the 27th Annual ACM Symposium on Applied Computing*, pages 1451–1456, 2012.

[105] Florian Kerschbaum. Outsourced private set intersection using homomorphic encryption. In *Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security*, pages 85–86, 2012.

[106] Florian Kerschbaum. Data science pipeline, 2019.

[107] Florian Kerschbaum, Martin Härterich, Patrick Grofig, Mathias Kohler, Andreas Schaad, Axel Schröpfer, and Walter Tighzert. Optimal re-encryption strategy for joins in encrypted databases. In *IFIP Annual Conference on Data and Applications Security and Privacy*, pages 195–210. Springer, 2013.

[108] D. Kifer and A. Machanavajjhala. Pufferfish: A framework for mathematical privacy definitions. *Association for Computing Machinery*, 1:39, 2014.

[109] Sam Kim, Kevin Lewi, Avradip Mandal, Hart Montgomery, Arnab Roy, and David J. Wu. Function-hiding inner product encryption is practical. In Dario Catalano and Roberto De Prisco, editors, *Security and Cryptography for Networks*, pages 544–562, Cham, 2018. Springer International Publishing.

[110] Edwin M. Knorr and Raymond T. Ng. Algorithms for mining distance-based outliers in large datasets. In *Proceedings of the 24rd International Conference on Very Large Data Bases*, VLDB '98, pages 392–403, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc.

[111] Edwin M. Knorr, Raymond T. Ng, and Vladimir Tucakov. Distance-based outliers: Algorithms and applications. *The VLDB Journal*, 8(3-4):237–253, February 2000.

[112] Simeon Krastnikov, Florian Kerschbaum, and Douglas Stebila. Efficient oblivious database joins. *Proc. VLDB Endow.*, 13(11):2132–2145, 2020.

[113] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. *Neural Information Processing Systems*, 25, 01 2012.

[114] Marie-Sarah Lacharité and Kenneth G. Paterson. A note on the optimality of frequency analysis vs. ℓp-optimization. *IACR Cryptol. ePrint Arch.*, 2015:1158, 2015.

[115] G.C. Langelaar, Iwan Setyawan, and R.L. Lagendijk. Watermarking digital image and video data. a state-of-the-art overview. *Signal Processing Magazine, IEEE*, 17:20 – 46, 10 2000.

[116] Eric Lantz, Kendrick Boyd, and David Page. Subsampled exponential mechanism: Differential privacy in large output spaces. In *Proceedings of the 8th ACM Workshop on Artificial Intelligence and Security*, AISec '15, pages 25–33, New York, NY, USA, 2015. ACM.

[117] Yann LeCun, Bernhard E Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne E Hubbard, and Lawrence D Jackel. Handwritten digit recognition with a back-propagation network. In *Advances in neural information processing systems*, pages 396–404, 1990.

[118] C. Y. Lee. An algorithm for path connections and its applications. In *IRE Transactions on Electronic Computers*, EC-10 (2), pages 346 – 365. IEEE, 1961.

[119] K. T. Leung, Ida A. C. Mok, and Suen S. N. *Polynomials and Equations*. Hong Kong University Press (HKU), 1992.

[120] Zheng Li and Shanqing Guo. Deepstego: Protecting intellectual property of deep neural networks by steganography. *arXiv preprint arXiv:1903.01743*, 2019.

[121] Jiongqian Liang and Srinivasan Parthasarathy. Robust contextual outlier detection: Where context meets sparsity. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, CIKM '16, pages 2167–2172, New York, NY, USA, 2016. ACM.

[122] C. Liu and P. Mittal. Linkmirage: Enabling privacy-preserving analytics on social relationships. In *NDSS*, 2016.

[123] Changchang Liu, Supriyo Chakraborty, and Prateek Mittal. Dependence makes you vulnerable: Differential privacy under dependent tuples. In *NDSS*, volume 16, pages 21–24, 2016.

[124] Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. Fine-pruning: Defending against backdooring attacks on deep neural networks. *CoRR*, abs/1805.12185, 2018.

[125] Kun Liu and Evimaria Terzi. Towards identity anonymization on graphs. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, SIGMOD '08, page 93–106, New York, NY, USA, 2008. Association for Computing Machinery.

[126] Yingqi Liu, Shiqing Ma, Yousra Aafer, Wen-Chuan Lee, Juan Zhai, Weihang Wang, and Xiangyu Zhang. Trojaning attack on neural networks. 01 2018.

[127] Yuntao Liu, Yang Xie, and Ankur Srivastava. Neural trojans. In *2017 IEEE International Conference on Computer Design (ICCD)*, pages 45–48. IEEE, 2017.

[128] Yushan Liu, Shouling Ji, and Prateek Mittal. Smartwalk: Enhancing social network security via adaptive random walks. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, page 492–503, New York, NY, USA, 2016. Association for Computing Machinery.

[129] Edward Lui and Rafael Pass. Outlier privacy. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *Theory of Cryptography*, pages 277–305, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg.

[130] Nils Lukas, Edward Jiang, Xinda Li, and Florian Kerschbaum. Sok: How robust is image classification deep neural network watermarking? (extended version), 2021.

[131] A. Machanavajjhala, D. Kifer, J. Abowd, J. Gehrke, and L. Vilhuber. Privacy: Theory meets practice on the map. In *2008 IEEE 24th International Conference on Data Engineering*, pages 277–286, April 2008.

[132] Ashwin Machanavajjhala, Daniel Kifer, Johannes Gehrke, and Muthuramakrishnan Venkitasubramaniam. l-diversity: Privacy beyond k-anonymity. *ACM Trans. Knowl. Discov. Data*, 1(1):3–es, March 2007.

[133] G Mclachlan. Mahalanobis distance. *Resonance*, 4:20–26, 06 1999.

[134] F. McSherry and K. Talwar. In *Proceedings of the 48th IEEE Symposium on Foundations of Computer Science (FOCS)*, page 94–103, 2007.

[135] F. D. McSherry. Privacy integrated queries: An extensible platform for privacy-preserving data analysis. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data*, SIGMOD '09, page 19–30, New York, NY, USA, 2009. Association for Computing Machinery.

[136] Frank McSherry and Kunal Talwar. Mechanism design via differential privacy. In *Proceedings of the 48th Annual IEEE Symposium on Foundations of Computer Science*, FOCS '07, pages 94–103, Washington, DC, USA, 2007. IEEE Computer Society.

[137] Shagufta Mehnaz and Elisa Bertino. Privacy-preserving real-time anomaly detection using edge computing. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*, pages 469–480, 2020.

[138] Erwan Le Merrer, Patrick Perez, and Gilles Trédan. Adversarial frontier stitching for remote neural network watermarking. *arXiv preprint arXiv:1711.01894*, 2017.

[139] Victor Miller. The weil pairing, and its efficient calculation. *J. Cryptology*, 17:235–261, 09 2004.

[140] Ilya Mironov. Renyi differential privacy. *CoRR*, abs/1702.07476, 2017.

[141] Ilya Mironov, Gil Segev, and Ido Shahaf. Strengthening the security of encrypted databases: Non-transitive joins. In Yael Kalai and Leonid Reyzin, editors, *Theory of Cryptography*, pages 631–661. Springer International Publishing, 2017.

[142] Prateek Mittal, Charalampos Papamanthou, and Dawn Song. Preserving link privacy in social network based systems. 08 2012.

[143] Einar Mykletun and Gene Tsudik. On security of sovereign joins. *IACR Cryptol. ePrint Arch.*, 2006:380, 2006.

[144] Yuki Nagai, Yusuke Uchida, Shigeyuki Sakazawa, and Shin'ichi Satoh. Digital watermarking for deep neural networks. *International Journal of Multimedia Information Retrieval*, 7(1):3–16, Mar 2018.

[145] Ryota Namba and Jun Sakuma. Robust watermarking of neural network with exponential weighting. *arXiv preprint arXiv:1901.06151*, 2019.

[146] Muhammad Naveed, Seny Kamara, and Charles V. Wright. Inference attacks on property-preserving encrypted databases. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, October 12-16, 2015*, pages 644–655, 2015.

[147] Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. Smooth sensitivity and sampling in private data analysis. In *Proceedings of the Thirty-ninth Annual ACM Symposium on Theory of Computing*, STOC '07, pages 75–84, New York, NY, USA, 2007. ACM.

[148] Kobbi Nissim, Uri Stemmer, and Salil Vadhan. Locating a small cluster privately. 04 2016.

[149] Douglas Oard, William Webber, David Kirsch, and Sergey Golitsynskiy. Avocado research email collection. https://catalog.ldc.upenn.edu/LDC2015T03, 2015.

[150] Seong Joon Oh, Max Augustin, Mario Fritz, and Bernt Schiele. Towards reverse-engineering black-box neural networks. In *International Conference on Learning Representations*, 2018.

[151] Rina Okada, Kazuto Fukuchi, and Jun Sakuma. Differentially private analysis of outliers. In Annalisa Appice, Pedro Pereira Rodrigues, Vítor Santos Costa, João Gama, Alípio Jorge, and Carlos Soares, editors, *Machine Learning and Knowledge Discovery in Databases*, pages 458–473, Cham, 2015. Springer International Publishing.

[152] Tribhuvanesh Orekondy, Bernt Schiele, and Mario Fritz. Knockoff nets: Stealing functionality of black-box models. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[153] Hweehwa Pang and Xuhua Ding. Privacy-preserving ad-hoc equi-join on outsourced data. *ACM Transactions on Database Systems (TODS)*, 39(3):1–40, 2014.

[154] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z. Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, ASIA CCS '17, pages 506–519, New York, NY, USA, 2017. ACM.

[155] Emanuel Parzen. On estimation of a probability density function and mode. *Ann. Math. Statist.*, 33(3):1065–1076, 09 1962.

[156] Raluca A. Popa, Catherine M. S. Redfield, Nickolai Zeldovich, and Hari Balakrishnan. Cryptdb: protecting confidentiality with encrypted query processing. In *Proceedings of the 23rd ACM Symposium on Operating Systems Principles (SOSP)*, pages 85–100, 2011.

[157] Gil Press. Cleaning big data: Most time-consuming, least enjoyable data science task, survey says, Mar 2016.

[158] Sridhar Ramaswamy, Rajeev Rastogi, Kyuseok Shim, and Taejon Korea. Efficient algorithms for mining outliers from large data sets. 04 2000.

[159] S. Raskhodnikova and A. Smith. Efficient lipschitz extensions for high-dimensional graph statistics and node private degree distributions. *FOCS*, 2016.

[160] Bernard Rosner. Percentage points for a generalized esd many-outlier procedure. *Technometrics*, 25(2):165–172, 1983.

[161] Bita Darvish Rouhani, Huili Chen, and Farinaz Koushanfar. Deepsigns: A generic watermarking framework for ip protection of deep learning models. *arXiv preprint arXiv:1804.00750*, 2018.

[162] Lalit Kumar Saini and Vishal Shrivastava. A survey of digital watermarking techniques and its applications. *CoRR*, abs/1407.4735, 2014.

[163] A. Sala, X. Zhao, C. Wilson, H. Zheng, and B.Y. Zhao. Sharing graphs using differentially private graph models. In *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*, page 81–98, 2011.

[164] J. T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *J. ACM*, 27(4):701–717, October 1980.

[165] Masoumeh Shafieinejad, Suraj Gupta, Liu Jin Yang, Koray Karabin, and Florian Kerschbaum. Equi-joins over encrypted data for series of queries. In *arXiv:2103.05792*, 2021.

[166] Masoumeh Shafieinejad, Huseyin A. Inan, Marcello Hasegawa, and Robert Sim. On privacy and confidentiality of communications in organizational graphs. In *ICLR workshop on Distributed and Private Machine Learning (DPML)*, 2021.

[167] Masoumeh Shafieinejad, Florian Kerschbaum, and Ihab F. Ilyas. Pcor: Private contextual outlier release via differentially private search. In *46th ACM International Conference on Management of Data (SIGMOD)*, 2021.

[168] Masoumeh Shafieinejad, Nils Lukas, Jiaqi Wang, Xinda Li, and Florian Kerschbaum. On the robustness of backdoor-based watermarking in deep neural networks. In *9th ACM Workshop on Information Hiding and Multimedia Security (IH&MMSEC)*, 2021.

[169] R. Shokri, M. Stronati, C. Song, and V. Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 3–18, May 2017.

[170] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv 1409.1556*, 09 2014.

[171] S. Song, Y. Wang, and K. Chaudhuri. Pufferfish privacy mechanisms for correlated data. In *Proceedings of the 2017 ACM International Conference on Management of Data*, page 1291–1306, 2017.

[172] Dong Su, Jianneng Cao, Ninghui Li, Elisa Bertino, and Hongxia Jin. Differentially private k-means clustering. In *Proceedings of the Sixth ACM Conference on Data and Application Security and Privacy*, CODASPY '16, pages 26–37, New York, NY, USA, 2016. ACM.

[173] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3104–3112. Curran Associates, Inc., 2014.

[174] Mitchell D. Swanson, Mei Kobayashi, and Ahmed Hossam Tewfik. Multimedia data-embedding and watermarking technologies. *Proceedings of the IEEE*, 86(6):1064–1087, 12 1998.

[175] Dan Swinhoe. The 15 biggest data breaches of the 21st century, Jan 2021.

[176] Guanting Tang, James Bailey, Jian Pei, and Guozhu Dong. Mining multidimensional contextual outliers from categorical relational data. *Intelligent Data Analysis*, 19, 07 2013.

[177] Brian Thompson and Danfeng Yao. The union-split algorithm and cluster-based anonymization of social networks. pages 218–227, 01 2009.

[178] Gary L. Tietjen and Roger H. Moore. Some grubbs-type statistics for the detection of several outliers. *Technometrics*, 14(3):583–597, 1972.

[179] Florian Tramèr, Fan Zhang, Ari Juels, Michael K. Reiter, and Thomas Ristenpart. Stealing machine learning models via prediction api. In *Proceedings of the 25th USENIX Conference on Security Symposium*, SEC'16, pages 601–618, Berkeley, CA, USA, 2016. USENIX Association.

[180] Stephen Tu, M. Frans Kaashoek, Samuel Madden, and Nickolai Zeldovich. Processing analytical queries over encrypted data. *Proceedings of the VLDB Endowment*, 6(5):289–300, 2013.

[181] Yusuke Uchida, Yuki Nagai, Shigeyuki Sakazawa, and Shin'ichi Satoh. Embedding watermarks into deep neural networks. In *Proceedings of the 2017 ACM on International Conference on Multimedia Retrieval*, pages 269–277. ACM, 2017.

[182] J. Vaidya and C. Clifton. Privacy-preserving outlier detection. In *Fourth IEEE International Conference on Data Mining (ICDM'04)*, pages 233–240, Nov 2004.

[183] B. Wang and N. Z. Gong. Stealing hyperparameters in machine learning. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 36–52, May 2018.

[184] Bolun Wang, Yuanshun Yao, Shawn Shan, Huiying Li, Bimal Viswanath, Haitao Zheng, and Ben Y Zhao. Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In *Neural Cleanse: Identifying and Mitigating Backdoor Attacks in Neural Networks*, page 0. IEEE, 2019.

[185] Tianhao Wang and Florian Kerschbaum. Attacks on digital watermarks for deep neural networks. In *44th International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2019.

[186] Tianhao Wang and Florian Kerschbaum. Robust and undetectable white-box watermarks for deep neural networks, 2019.

[187] Yujue Wang and HweeHwa Pang. Probabilistic Public Key Encryption for Controlled Equijoin in Relational Databases. *The Computer Journal*, 60(4):600–612, 10 2016.

[188] Jia Xu, Zhenjie Zhang, Xiaokui Xiao, Yin Yang, Ge Yu, and Marianne Winslett. Differentially private histogram publication. *The VLDB Journal*, 22(6):797–822, 2013.

[189] Bin Yang, Issei Sato, and Hiroshi Nakagawa. Bayesian differential privacy on correlated data. In *SIGMOD*, page 747–762, 2015.

[190] Xiaowei Ying and Xintao Wu. Graph generation with prescribed feature constraints. pages 966–977, 04 2009.

[191] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3320–3328. Curran Associates, Inc., 2014.

[192] Mingxuan Yuan, Lei Chen, and Philip S. Yu. Personalized privacy protection in social networks. *Proc. VLDB Endow.*, 4(2):141–150, November 2010.

[193] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In Edwin R. Hancock Richard C. Wilson and William A. P. Smith, editors, *Proceedings of the British Machine Vision Conference (BMVC)*, pages 87.1–87.12. BMVA Press, September 2016.

[194] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander J Smola. Deep sets. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 3391–3401. Curran Associates, Inc., 2017.

[195] Jialong Zhang, Zhongshu Gu, Jiyong Jang, Hui Wu, Marc Ph Stoecklin, Heqing Huang, and Ian Molloy. Protecting intellectual property of deep neural networks with watermarking. In *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*, pages 159–172. ACM, 2018.

[196] Li Zhang, Chengyu Hu, Yang Zhang, and Shanqing Guo. How to prove your model belongs to you: A blind-watermark based framework to protect intellectual property of dnn. December 2019.

[197] X. Zheng, J. Han, and A. Sun. A survey of location prediction on twitter. *IEEE Transactions on Knowledge and Data Engineering*, 30(9):1652–1671, 2018.

[198] Tianqing Zhu, Gang Li, Wanlei Zhou, and S Yu Philip. Differentially private social network data publishing. In *Differential Privacy and Applications*, pages 91–105. Springer, 2017.

[199] Richard Zippel. Probabilistic algorithms for sparse polynomials. In Edward W. Ng, editor, *Symbolic and Algebraic Computation*, pages 216–226, Berlin, Heidelberg, 1979. Springer Berlin Heidelberg.

[200] Lei Zou, Lei Chen, and M. Tamer Özsu. K-automorphism: A general framework for privacy preserving network publication. *Proc. VLDB Endow.*, 2(1):946–957, August 2009.