UNIVERSITY OF OKLAHOMA

GRADUATE COLLEGE

FRAUD DETECTION FRAMEWORK USING PRIVACY PRESERVING RECORD LINKAGE IN TELECOM

A DISSERTATION

SUBMITTED TO THE GRADUATE FACULTY

in partial fulfillment of the requirements for the

Degree of

DOCTOR OF PHILOSOPHY IN ENGINEERING

By
Satish Thomas
Norman, Oklahoma
2024

FRAUD DETECTION FRAMEWORK USING PRIVACY PRESERVING RECORD LINKAGE IN TELECOM

A DISSERTATION APPROVED FOR THE

COLLEGE OF ENGINEERING

BY THE COMMITTEE CONSISTING OF

Dr. James Sluss, Chair

Dr. Timothy Ford

Dr. Yong-Mi Kim

Dr. Sridhar Radhakrishnan

Acknowledgments

I would like to dedicate and sincerely thank my God Almighty, who is solely responsible for helping me through my PhD process. Next, I would like to thank my most wonderful wife and the love of my life, Dr. Sailatha Thomas, who has stood by me through all these years. Every single day, through prayers and support, and at times when I wanted to give up, she encouraged me to carry on to the finish. I would also like to thank my two most beautiful and brilliant daughters, Divya Thomas and Kanchan Thomas, for being by my side and encouraging me. I want to acknowledge my family, especially my parents, Dr. Anthony Thomas and Manjula Thomas, and my mother-in-law, Chandrika Srinivasan, for their prayers and support.

I would like to thank the College of Engineering and Graduate College for giving me the opportunity to conduct my PhD and complete it. I am extremely grateful to Dr. James Sluss for being my mentor and guide. Without his meticulous guidance and help, I would have never been able to complete my PhD. It was an amazing honor to have him as a mentor.

I would like to thank Dr. Yong Mi Kim for her full support and advice throughout my PhD. I sincerely thank Dr. Timothy Ford and Dr. Shridhar Radhakrishnan for being part of my committee and for their support and guidance through this process.

I would like to thank Dr. Pramode Verma for taking me on as his student and introducing me to the topic of Privacy. I have learned so much through this subject.

A special thanks and sincere gratitude goes out to Krista Pettersen for being so patient with me and guiding me through the PhD process. I would like to thank everyone who was involved in the PhD process and helped me to be successful.

TABLE OF CONTENTS

List of Figures

List of Tables

ABSTRACT

Fraud is a persistent and increasing problem in the telecom industry. Telcos work in isolation to prevent fraud. Sharing information is critical for detecting and preventing fraud. The primary constraint on sharing information is privacy preservation. Several techniques have been developed to share data while preserving privacy using privacy-preserving record linkage (PPRL). Most of the PPRL techniques use a similarity measure like Jacquard similarity on homologous datasets, which are all prone to graph-based attacks, rendering existing methods insecure. Many complex and slow techniques use the Bloom filter implementation, which can be compromised in a cryptanalysis attack.

This dissertation proposes an attack-proof PPRL framework (FD-PPRL) using existing infrastructure of a telco without a complex multistep protocol. First, a novel way of matching two non-homologous datasets using attack-proof digital signature schemes, like the Edwards-curve digital signature algorithm is proposed. Here, Jaccard similarity can only be estimated using this method and not on the datasets directly. Second, two parties transact with a simple request–reply method. To validate the match accuracy, privacy preservation, and performance of this approach, it was tested on a large public dataset (North Carolina Voter Database). This method is secure against attacks and achieves 100% match accuracy with improved performance.

Fraud detection using Privacy-Preserving Record Linkage (PPRL) was first proposed in our dissertation. It addresses many challenges related to fraud detection in the telecommunications industry, which can then be expanded to other industries. We begin by introducing the research

problems and challenges, followed by proposing solutions through research and development to address these issues. Our background section covers some of the PPRL concepts and terminologies, and we conduct a detailed analysis of current research and literature. Next, we develop the novel FD-PPRL framework upon which our solution is built. We also analyze various hash techniques with FD-PPRL and select the most suitable one for our implementation. The dissertation then delves into how the telco can securely store and transmit large volumes of fraud-related information using a (or the) Block method. Additionally, we address speed issues and develop a method to facilitate communication among multiple telcos simultaneously. Finally, we conclude by highlighting how our approach has effectively solved the persistent fraud-sharing problem among telcos and provide guidance for future research.

INTRODUCTION

This chapter serves as an introduction to the dissertation. We will begin by defining privacy and its importance, and then focus on the specific area in the telecom industry where there is a need to share information while still protecting the privacy of the individual. In Section 1.1, we will introduce the research problem. In Section 1.2, we will discuss the application of Privacy-Preserving Record Linkage for fraud detection. In Section 1.3, we will define in detail the research problem with its challenges. Section 1.4 describes the aim and objectives of this research. Our contribution to the area of research is described in Section 1.5, and Section 1.6 outlines our research methodology.

1.1 Research Problem

Telecommunication companies are the primary source of technological connectivity worldwide and are prone to several daily fraud attacks [1]. As telecom products evolve, our everyday lives become more virtual, increasing the need to identify, prevent, and protect against these attacks. Therefore, telecommunication companies are primarily responsible for protecting internal and customer information transmitted over their network.

According to the Communication Fraud Control Association (https://cfca.org/stop-the-fraud-stop-the-churn/), telecom fraud is worth billions of dollars, which amounted to approximately $38.1 billion in 2021. Although there are various kinds of fraud, subscription fraud [2] is among the most impactful. When a telecom company becomes a victim of fraud, it can lose its reputation and customers. Furthermore, the companies need to use each other's networks to support customer traffic. Therefore, an attack on one telecom company could impact other interconnected

companies.

Traditional fraud management systems within telcos operate in isolation; fraudsters know and exploit this deficiency. When identified by one telecom, they move to another. These fraudsters can be stopped if telcos cooperate and share information. This was not possible in the past owing to privacy laws at the state, national, and international levels, such as Customer Proprietary Network Information (CPNI) and General Data Protection Regime (GDPR)[3] in Europe. Many such laws are either being considered for legislation or being formulated.

For telcos to share information they need to comply with privacy laws and protect their business secrecy. The main challenges are: 1) sharing fraud information without revealing data; 2) sharing information without involving any third party; and 3) using a protocol that is safe, secure, fast, and easy to implement. This can be achieved using our privacy-preserving record linkage (PPRL) methods.

1.1.1.) Sharing fraud information without revealing data is a key aspect of preventing fraud in the telecom industry. Identifying bad actors is crucial, but telcos can only identify repeat offenders using the information they have. On the first fraud occurrence for a specific telco, they cannot identify the fraud without additional information. This is where sharing information between telcos becomes extremely important. However, there are many constraints to doing so, including privacy laws that prevent telcos from sharing information. One of the most prominent privacy laws is the General Data Protection Regulation (GDPR) [3], which imposes harsh penalties for breaches of the law. Additionally, individual telcos may choose not to share information to remain competitive.

While there are barriers for telcos to share information, our protocol allows fraud information to be shared without revealing information to the seeking and receiving telcos. This

would be a win-win situation for identifying fraud effectively.

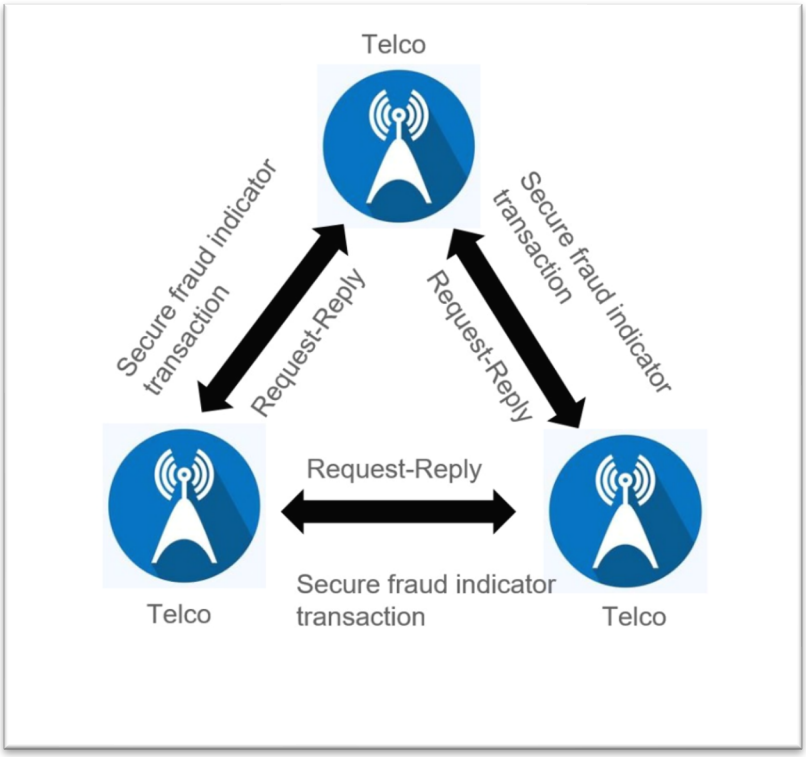

Figure 1.1 Fraud verification transaction between Telcos

1.1.2.) Sharing information without involving any third party is a challenging task in the telecom industry, which is highly regulated and competitive. While information is key, it is also highly guarded due to the reasons mentioned in 1.). Many privacy-preserving record linkage (PPRL) protocols have been developed to conduct matches while not revealing information to each other. However, many of these protocols involve third parties, such as linkage units (LUs) [4], which can still expose information to the LU provider, even if they are semi-trustworthy. With the volume of data that could potentially be transacted through the LU, the third party would figure out the information and thus be in violation of privacy laws. Telcos will be more receptive to sharing fraud information if they can transact information directly with each other through a secure protocol. Figure 1.1 describes one such protocol described in this dissertation where the Telcos can transact

with each other directly through a request-reply method.

1.1.3.) Using a protocol that is safe, secure, fast, and easy to implement is crucial for an industry application. A fraud sharing protocol needs to be hack-proof and proven to withstand cryptanalysis attacks. Any breach of the protocol, as described previously, can have grave consequences for Telcos and the industry in general. For a protocol to have good adoption, it needs to be easy to implement on existing infrastructure and scalable across many Telecoms. When a fraud attack happens, speed of response is of vital importance. While data speed transmission has improved over time, along with database query and fetching speeds, the challenge remains in processing when the data is obfuscated or needs to be limited due to privacy constraints. Existing protocols have a run time complexity of O(nm) [5], where n is the number of steps and m is the speed of each step. Our protocol ensures that the implementation is safe, fast, secure, and extremely easy to implement.

Definition 1.1: Fraud Detection through Privacy Preserving Record Linkage (FD-PPRL) is a protocol that enables multiple Telcos to share fraud information without revealing sensitive data. Assume T_A, T_B, T_C, …, T_N are N Telcos who have gathered fraud information over the course of conducting business. They respectively have D_A, D_B, D_C, …, D_N as their fraud databases, which we will describe in Chapter 6 on efficiently organizing that information for querying. To request fraud information (Fi), Telco T_B sends a tuple of information (Fi_B, Fj_B, Fk_B, …, Fn_B) to the Telco to determine which of their records $F_A \in D_A$, $F_C \in D_C$, …, $F_N \in D_N$ match the tuple using our protocol. Once a match is found, a block of information is sent to the requesting Telco, or confirmation is sent that the information requested is related to a fraud incident at one of the Telcos.

1.2 Application of the Fraud Detection Privacy Preserving Records Linkage (FD-PPRL)

1.2.1 Fraud detection and prevention across multiple industries is a complex process that requires various transactions [6], inter-group cooperation, and domain and industry knowledge. With the large volume of data being transacted nowadays, data breaches and ransomware attacks are happening frequently. Requesting and analyzing data across industries today can be expensive and requires many professional hours to set up. FD-PPRL provides an effective way to fetch fraud information across various domains and industries while maintaining and protecting privacy.

Fraudsters typically falsify their core information that identifies them and could hit multiple industries with similar patterns of behavior [6]. They change their information to look real to mislead intended parties. Such false information provides various linkage issues across databases and further empowers such fraudsters to commit such acts.

FD-PPRL provides a way to identify false identities in databases and prevent them from being recorded in a legitimate database. The protocol provides the ability to identify records by linking them in near real time while preserving sensitive privileged information. It is also scalable to multiple databases and industries.

1.2.2 Sharing medical information - FD-PPRL protocol can be used by research organizations in the medical field to obtain medical information across a patient base without revealing to the hospital who the inquiry is about. The research organization can obtain attribute-level information without revealing the identity of the patient. Hospitals can also use our protocol to ask about a patient and obtain specific information about test results or diseases not revealed during the patient check-in process. This can be done without the hospital revealing patient information to each other.

1.2.3 Our protocol can be used by government and criminal investigating agencies to inquire in near real-time about potential terrorist threats. However, if privacy is not maintained, an inquiry into a potential criminal database could bring that person into natural suspicion and violate their privacy. Some inquiries, even into such activities, need to adhere to privacy laws and constraints. Figure 2.2 shows how government and security agencies can obtain potential threat information without violating privacy. If the individual is not a terrorist, then their innocence can be maintained.

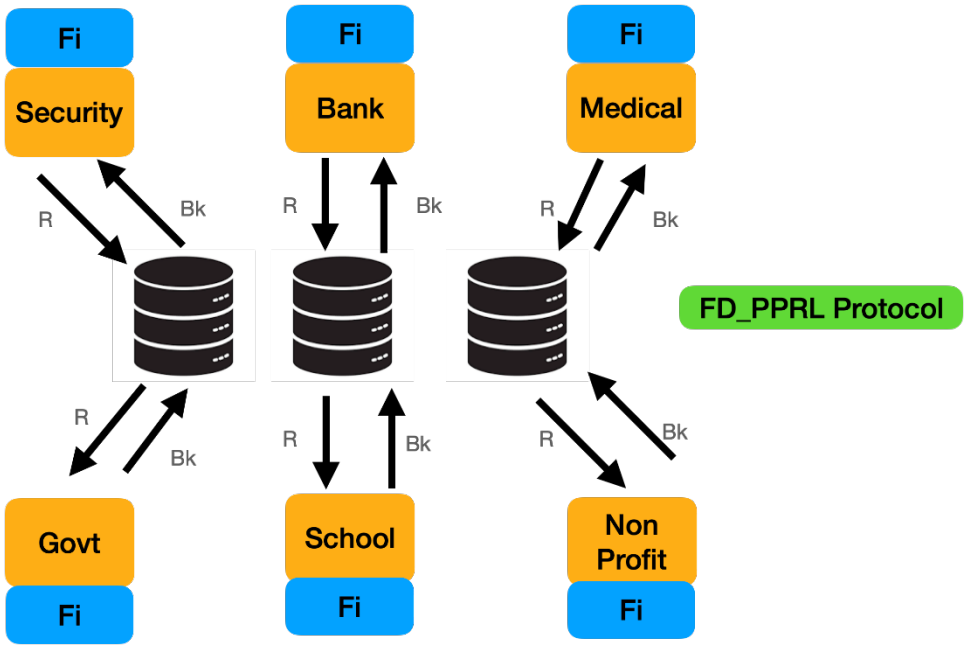

Figure 1.2 FD-PPRL protocol across industry segments

1.3 Research Problem Definition

In Chapter 3, we will examine several PPRL methods that are currently available. However, all existing PPRL methods have a common issue: the data sets being compared are encrypted using the same method. This results in homogeneity between the individual data, which can be exploited by an adversary to calculate a similarity measure on the encrypted data and determine the original data through graph attacks [7]. To prevent this attack, a method to measure

similarity must be developed on data processed through completely different protocols and only by the receiver. By transmitting only, the encrypted data to be compared against the unencrypted data, we can ensure that the two sets of data are always completely different, and the adversary cannot extract any information from the transmitted encrypted data.

It is worth noting that in the telecom industry, the use of a third party is never feasible. If a protocol is developed with a third party, it is unlikely to be adopted. Therefore, the protocol needs to be simple, fast, secure, and direct.

Our study proposes a new PPRL method that: 1) is secure, and fast; 2) is based on a simple request–reply protocol to share fraud data between telcos; and 3) does not require a trusted third party.

Challenge 1

Create a protocol that compares two non-homologous datasets - Many PPRL methods that exist today rely on the same encryption methodology being shared between the two transacting entities [4][5][6]. This leads to the same methods being applied to both sets of data. While the data in isolation can be secure, using various cryptanalysis methods described in (Vindage 2022) [7], privacy can be compromised, making existing protocols completely defunct. The challenge is to compare two non-homologous databases where the similarity determination method, such as Jaccard similarity, can be done with completely disparate datasets and yet yield results, ensuring that privacy is not compromised.

Challenge 2

Create a protocol with a simple request reply mechanism- To ensure adoption in the telecom industry, it would be ideal to create a protocol with a simple request-reply mechanism. Many

existing PPRL methods are complex in their implementation, and the number of transactions grows when a third party or a linkage unit is involved [4], depending on how well each party intends to secure privacy. A simple request-reply protocol can accomplish the task in just two steps, making it practical for the telecom industry.

Challenge 3

Create a protocol that does not require a trusted third party - Using a third party to perform a privacy-preserving match on behalf of two transacting entities is unlikely to become popular in the telecom industry. This is because the introduction of a third party would require some form of regulatory oversight or an independent governing body oversight, which would make this setup cost-prohibitive at the very least. Additionally, third-party implementations are worked through the lens of being semi-honest, which makes these protocols impractical due to overarching privacy laws that are currently ineffective and many of which are being formulated and worked through in legislative bodies of many countries.

Challenge 4

Create a protocol that is secure and fast - Fraud detection using existing protocols can be complex and involve multiple transactions [4][5][6], which can make them slow. To quickly identify a fraudulent actor and prevent further damage, the protocol needs to have extremely quick response rates. However, increasing the speed of the protocol can compromise security, and vice versa. The ideal protocol for the telecom industry would be one that is both secure and fast at the same time.

1.4 Aims and objectives of this research

As mentioned in Section 1.1, there is a pressing need for a fast, secure, and safe protocol to share fraud information in the Telecom industry. The protocol should be designed to ensure quick adoption by telcos and be compliant with industry regulations. Most importantly, the privacy of the entity must be maintained throughout the transaction.

In the past decade, many approaches to Privacy-Preserving Record Linkage (PPRL) have been developed. As new versions of PPRL have been developed, their implementations have become more complex, compromising speed [4][5][6]. Moreover, all PPRL implementations have an inherent flaw where the same protocol is embedded in their implementation, causing them to be homologous. Developing a PPRL protocol across two non-homologous datasets while maintaining speed and simplicity is still an open research problem.

The main aim of this dissertation is to develop a safe, secure, and fast protocol that can be easily adopted for the telecom industry. Our goal is to develop techniques to compare non-homologous datasets. The protocol needs to have a simple request-reply protocol and should be scalable across large datasets and multiple companies/entities. This protocol will be general enough to be employed across other industry segments. Based on our aim, we define several objectives to be accomplished in this study:

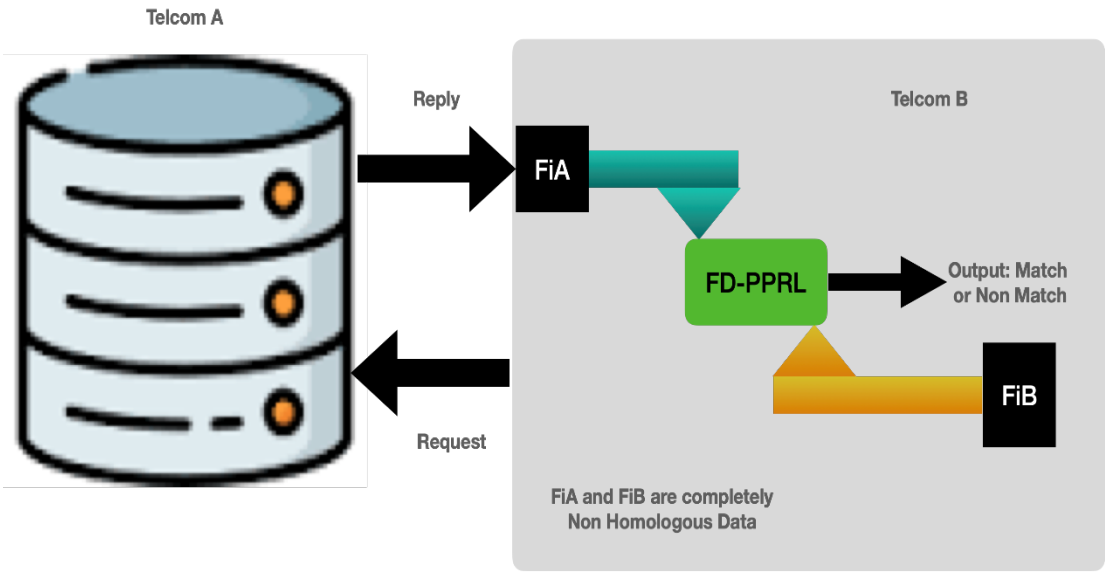

Figure 1.3 High-level representation of the FD-PPRL Protocol

1.4.1) Develop a FD-PPRL protocol that compares two non-homologous datasets

To perform record linkage, similarity measures are used in PPRL implementations [4][5]. One of the most effective measures is the Jaccard similarity measure, which establishes the commonality between the data sets. As described in Chapter 4, the Jaccard similarity measure will only yield a match if the comparison is on two similar data sets T_A and T_B, where $E(T_A) \sim E(T_B)$, and E is the encryption function, which can range from a simple hash function to complex implementations.

As described, data sets with similar encryptions inherently have homogeneity infused into them, making them susceptible to cryptanalysis attacks. In Chapter 4, we will develop a protocol by choosing one of the most secure encryption methods using the Edwards Curve digital Signature algorithm and determine a novel way of Jacquard similarity comparison between two non-homologous datasets.

We will establish the privacy of the implementation by subjecting the protocol to cryptanalysis attacks and measuring the outcome. We will compare the results against other

implementations to demonstrate the security of FD-PPRL. An analysis on the speed of processing will also be conducted and compared to existing protocols to demonstrate its capacity and feasibility for wider telecom industry application.

1.4.2.) Develop FD-PPRL protocol choosing the most appropriate Similarity Measure and Hash techniques

In Chapter 4, we develop a basic framework to compare two non-homologous datasets using a simple request-reply method that is easy, fast, and secure. We use the Jaccard similarity [8] measure to compare the datasets. In Chapter 5, we apply different similarity measures to determine which protocol yields the most accurate results. We also try the protocol with other measures using different MinHash protocols in our framework. Finally, we compare the privacy analysis of these protocols to develop the one that is most suited for adoption in the telecom industry.

1.4.3.) Develop a block protocol with our FD-PPRL implementation for large datasets

Any industry application of a protocol will require analysis to be conducted on large datasets. However, performing a full database scan will slow down the implementation and make it impractical for wider use. In Chapter 6, we will develop techniques of blocking so that transacting Telcos can compare smaller datasets and transmit the same between them for comparisons. The whole premise is to avoid the use of third-party implementation when data is transmitted between Telcos. We will conduct an analysis across various methods to show how data can be transmitted securely and quickly.

1.4.4.) Develop a FD-PPRL protocol to communicate with multiple Telcos at the same time

Chapters 4, 5, and 6 describe implementations that mostly involve the interaction between two Telcos. However, it is speed-prohibitive if the implementation can be done only between two Telcos at a time. In Chapter 7, we will develop a protocol where a Telco can communicate with multiple Telcos at once to ensure timely identification of a fraud actor and prevent any potential damage. The developed implementation will again need to be secure, fast, and simple for wider industry adoption.

1.5 Contributions

This dissertation focuses on developing a Fraud Detection Privacy Preserving Protocol (FD-PPRL) for the Telecom industry, which can be easily adopted across other industry segments. Our contributions are categorized into four main sections.

1.5.1 A FD-PPRL framework for Telco industry application

We developed a framework that addresses a major flaw in current implementations where data comparison to perform linkage happens over homologous datasets. Additionally, the current protocols are not practical for the telecom industry due to several reasons described in Sections 1.1 and 1.2.

- We designed the protocol using a novel method of calculating Jacquard similarity over datasets that are not similar in any way.
- The framework uses one of the most secure encryption schemes developed.
- The FD-PPRL implementation provides PPRL practitioners across multiple industries with the ability to tailor the framework to their environment and use case.

1.5.2 Fast and secure implementation

The FD-PPRL protocol was designed with other protocols to ensure speedy response times. The implementation uses one of the most secure encryption schemes, where cryptanalysis attacks are ineffective on our protocol.

1.5.3 Simple request-reply protocol

Some of the complex PPRL implementations that exist today have multiple interactions with transaction entities [4][5][6], making them prone to cryptanalysis attacks [7]. Our protocol has been designed to minimize transactions between Telcos using a simple request-reply protocol. The algorithm steps to detect fraud are also minimal.

1.5.4 Efficient Block implementation for data transfers

Since fraud information, or any information for that matter, in a particular industry is stored in large databases, we have designed a blocking protocol where data can be extracted and transmitted in smaller blocks that are only relevant to the information being sought.

1.5.5 Multi Telecom protocol

As detailed in Chapter 7, our protocol has an implementation where communication can be sent to multiple Telcos with responses attained quickly to identify an attack. The protocol can take a step further for automated responses once the comfort level of using our protocol has been established with Telcos.

1.6 Research Methodology

For our dissertation, we used the process and methods below to design and develop algorithms for FD-PPRL. The process consists of several steps, as shown in Figure 1.4.

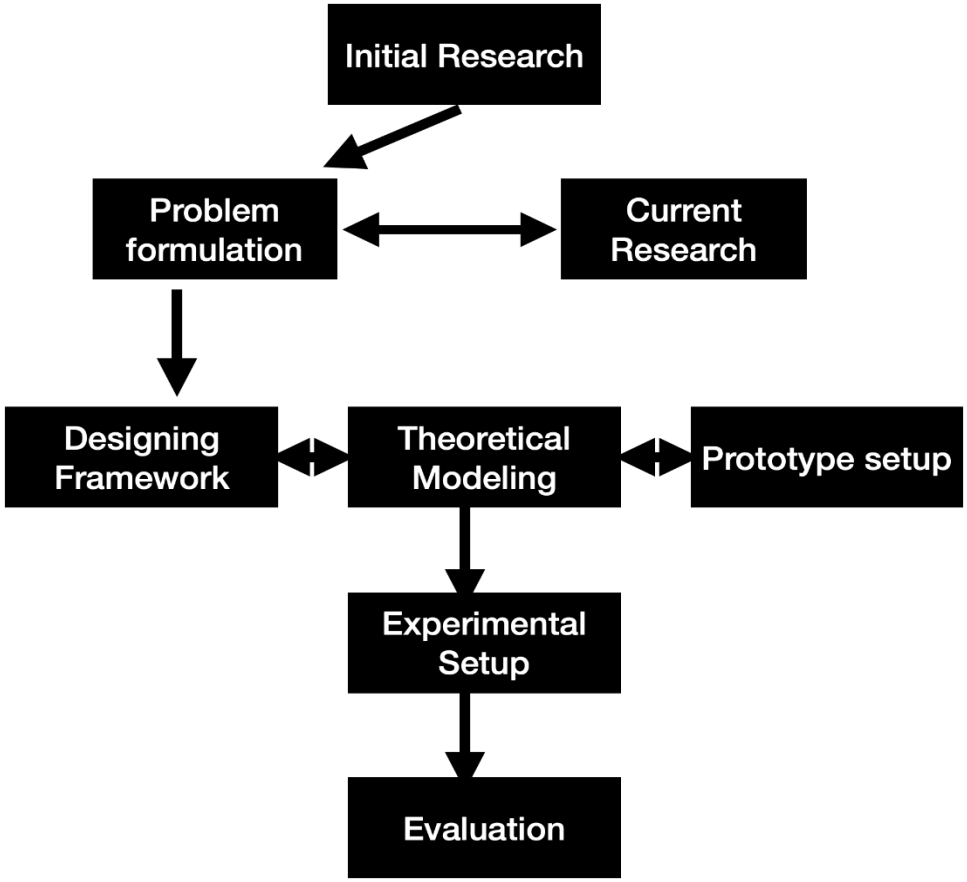

Figure 1.4 Flow diagram of the steps taken to conduct research and formulate this dissertation

1.) Initial research - During this stage, we developed a good understanding of the need for fraud prevention in the telecom industry, while also analyzing the concept of privacy and the need to protect the privacy of the individual being enquired upon.

2.) Formulating the research problem - In this step, we researched and analyzed the current literature on existing privacy-preserving record linkage methods available and determined their existing flaws. This helped us to formulate the research problem to solve.

3.) Literature Review - In this step, we researched and analyzed the current literature on existing privacy-preserving record linkage methods, determined their flaws, and formulated the research problem to solve.

4.) Designing framework - Here the algorithms were designed to solve the research problem.

5.) Theoretical modeling - In this stage the theory was formulated to prove how our protocol addressed the fundamental flaw in existing implementations.

6.) Prototype development - Python was used to develop and analyze the implementation described in section 1.4. We used a publicly available large dataset to analyze speed and security.

7.) Experimental setup - At this stage, we tested various test cases and use cases against our protocol. We conducted evaluations to measure specific metrics and establish the effectiveness of the protocol.

8) Evaluation - The results were analyzed to determine if the protocol is effective, fast, safe, and secure.

The next chapter will cover the background on privacy, the need for it, how to conduct linking through privacy-preserving record linkage (PPRL), and its concepts. In Chapter 3, we will present a review of existing PPRL concepts and approaches. Our fundamental FD-PPRL framework will be presented in Chapter 4, where we will explain our choice of encryption protocol, similarity measure, and theoretical analysis. In Chapter 5, we will evaluate our FD-PPRL technique through various Minhash protocols and establish the most optimal one for the telecom industry application. We will dive into aspects of the framework's application across large datasets, and in Chapter 6, we will discuss blocking approaches for efficient and small dataset management and transfer. In Chapter 7, we will develop a method for communication with multiple Telcos at the same time, emulating a real-world scenario. Finally, in Chapter 8, we will

conclude and summarize our findings and provide directions for future work in this area of research.

Chapter 2: BACKGROUND

In this chapter, we will explore some of the concepts and background used in the following chapters to design our framework in Chapter 4, and other PPRL techniques and concepts in Chapters 5, 6, and 7. In Section 2.1, we will explore some concepts of privacy and the needs related to it. Privacy Preserving Record Linkage (PPRL) will be described in Section 2.2. In section 2.3, we will describe Fraud indicators which will be used throughout this dissertation. Section 2.4 describes various techniques for similarity measures, the one chosen for our implementation. Section 2.5 describes encryption and the digital signature schemes and EdDSA technique used for our implementation. Section 2.6 describes various cryptanalysis methods and what is being used in our experimental evaluation. We will describe linkage quality measures in Section 2.7 that will be used in our dissertation. In Section 2.8, we will describe Privacy measure concepts to evaluate the robustness of our implementation. Section 2.9 describes our experimental setup and finally, in Section 2.10, we provide a summary of this chapter.

Table 2.1 Notation and Terminology used in this Chapter

$T_A, T_B, TC ....T_N$	Notation to represent Telecom companies
DA, DB, DC, ..., DN	Fraud database of the N Telecom companies
H	Hash function
O	Output of the Hash function
Hi	Set of hash function
E	Euclidean distance
J	Jaccard Similarity
K_P	Encryption and Decryption Key
TP	True Positive
FP	False Positive
FN	False Negative

Table 2.1 Notation and Terminology

2.1 Privacy

Privacy is a fundamental right for every individual in the world. It means the right to be left alone, where one has the freedom from intrusion or interference. This right is enshrined in the Universal Declaration of Human Rights, Article 12, which states that 'No one shall be subjected to arbitrary interference with his privacy, family, home, or correspondence, nor to attacks upon his honor and reputation. Misrepresentation and interpretation, which are very prevalent today, can be blocked with the concept of privacy.

Rocker C. Post elegantly [22] explained three concepts related to privacy. The first is the creation of knowledge, where information is created without misrepresentation of fact. The second

is related to dignity, where basic human decency and the right to dignity are maintained. The third is freedom, and in the context of privacy, it is where the right to choose to reveal information solely rests with the individual to whom the information pertains.

Privacy [22] is the ability to control who can access information about our private life and activities. It is the right to keep our personal information confidential and to prevent others from accessing it without our consent. It is a crucial aspect of our lives that helps us maintain our autonomy and individuality.

Privacy is important because:

- Privacy is important because it gives us the power to choose our thoughts and feelings, and who we share them with.
- It also protects our information that we do not want shared publicly, such as health or personal finances.
- Privacy helps protect our physical safety, especially if our real-time location data is private.
- Additionally, privacy helps protect us as individuals and our businesses against entities that are more powerful than us.
- Finally, privacy is tied to freedom. Without privacy, we could not be free and have free will.

2.2 Privacy Record Linkage

Across many industries, confidential data is collected and made available about individuals. This data includes their address, government ID numbers, health information, shopping preferences, browsing patterns, and more. The exchange and processing of this information is vital to maintain commerce, research, security, and in our case, prevention of fraud.

In a normal or regular linkage between datasets, the linkage is done on entities that are completely known to parties under transaction. Here, there is no preservation of privacy, and the enquiring entities know all the information about them. However, when we still need to get the attribute information, but do it in such a way that the entity which is typically being used to link the dataset is hidden, then this becomes an instance of privacy-preserving record linkage. This technique uses algorithmic techniques to effectively link records by matching identifiable information attributes without revealing them, thus keeping them protected.

Using the above concept lets define our definition of Fraud Detection Privacy preserving record linkage (FD-PPRL).

2.2.1 FD-PPRL Definition:

Assume $T_A, T_B, T_C, \ldots, and\ T_N$ are N Telcos who have come together to collaborate for fraud prevention. Let $D_A, D_B, D_C, \ldots, and\ D_N$ be the fraud databases containing $Fi_A, Fi_B, F_C, \ldots, and\ Fi_N$ fraud indicators or information. Where $Fi_A \in D_A$, $Fi_B \in$ *to* D_B, $Fi_C \in D_C$, …, etc. Now FD-PPRL will be the linkage where, let us say, T_B wants to inquire about the presence of Fi_B *in* $D_A, D_C, \ldots, or\ D_N$ without revealing Fi_B. If there is no Fi_B, then T_B does not learn about $Fi_A, Fi_C, \ldots, Fi_N$ in the other Telcom databases. Here, the actual entity information is not revealed, and hence privacy is preserved. However, the fraud information is obtained.

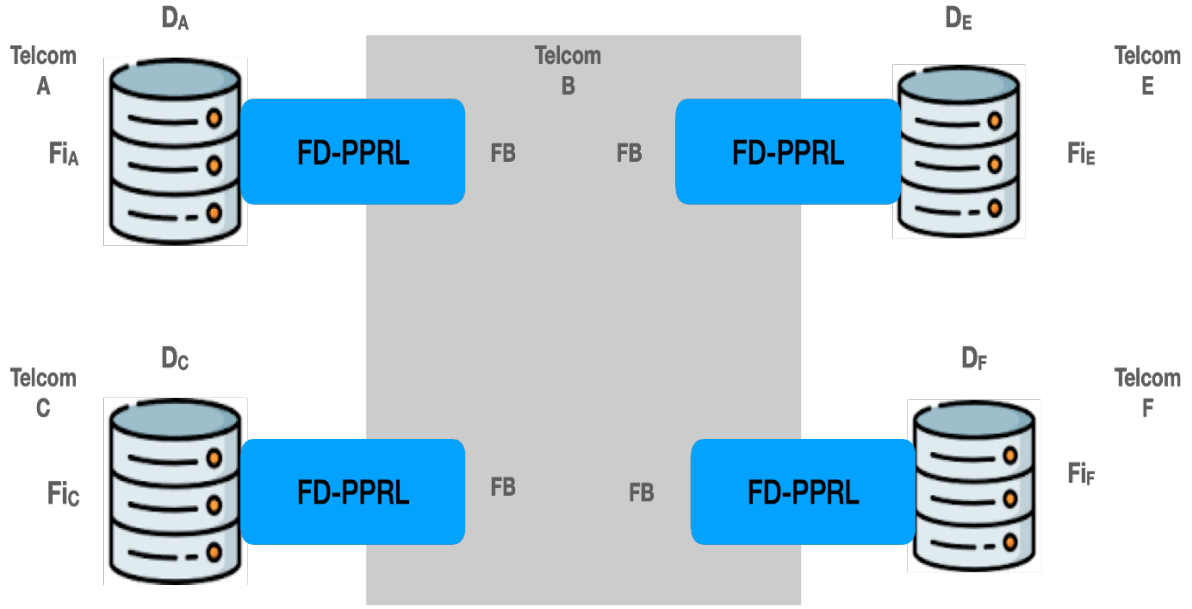

Figure 2.1 Telecom B interaction with Telcom A, C, E and F using FD-PPRL

.

2.2.2 PPRL transacting protocol

When Telcos share data, they have the option to transact directly with each other using a two-party protocol or form a collective group and transact using a multi-party protocol.

Two party transacting protocol

When two telecommunication companies agree to transact with each other frequently or on an as-needed basis, they can do so without revealing their entity information. They encrypt the information between themselves and exchange it by conducting linkage through FD-PPRL. Telcos typically agree upon some of the parameters before conducting the transaction. Some of the agreed methods or parameters can be encryption, HASH, and data organization schemas. Transacting between two Telcos at a time can be efficient, but computationally expensive.

Multi-party transacting protocol

When multiple telecommunication companies choose to communicate as a group, they must agree on common methods and parameters, like a two-party protocol. Both methods assume that there are no third parties involved in assisting with the linkage. This is one of the biggest advantages of FD-PPRL. In contrast, many protocols use a third party to collect information, which can be encrypted ahead of time to conduct a linkage, typically called the linkage unit (LU). Our protocol does not have an LU.

2.2.3 Adversary models

In our evaluation of the FD-PPRL protocol, we will examine the following models for Telcos or third parties to assess the security of our protocol against privacy leakage attacks, also known as cryptanalysis attacks.

1.) Honest Party

Here, Telco fully follows the steps of the Privacy-Preserving Record Linkage (PPRL) protocol to maintain the privacy of the transacting entity or fraud indicator. The Telcos do not collude with other Telcos to try to extrapolate the information to reveal the encrypted information [4]. However, this scenario is typically not practical in the real world and can sometimes only be achieved with external government intervention such as regulatory bodies or laws with violation penalties.

2.) Honest but curious mode

Telco follows the PPRL protocol to maintain privacy by not cross-sharing information with other Telcos [4]. However, they can choose to use the received data to extrapolate information and

extract data on the fraud indicator. This scenario is extremely likely as only the received information is used. It is important to note that such actions may fall under regulatory or legal purview.

3.) Malicious party

The Telco tries to follow the protocol, but if someone with malicious intent tries to violate privacy by learning information about other parties, many Privacy-Preserving Record Linkage (PPRL) protocols use a Secure Multi-Party Computation (SMC) method to prevent such actions. However, as described in Chapter 4 of our framework, which is designed for such scenarios, SMC is not necessary.

2.2.3 PPRL techniques

The PPRL techniques can be broadly classified into two categories. 1) Secure multi part computation technique (SMC), and 2.) Perturbation method.

During the process of secure computation, the transacting parties perform computation on encrypted values, which are kept private from each other. They do not know each other's information but only the result of the computation. The different computation techniques include secure multiparty computation [22], homomorphic encryption, secret sharing, and garbled circuits [15].

Secure summation

In this protocol, the parties wish to conduct a mathematical operation on their private values without sharing them. According to Yao and Ranbaduge [22], in a multiple-party situation where a summation needs to be computed, the first party computes its sum by adding a random number

or a prime number to their private number and passes on the sum to the second person in the loop. This process continues until the last person is reached, and the final summed amount is sent back to the first party. The secret number is then subtracted from the sum, and a secure summation is revealed without disclosing the individual numbers. There are many other secure summation protocols that work with this basic concept.

Secure set union and intersection

In a secure union set scenario, a global union set is created for the parties without each party knowing where the information in the set was sourced. The parties use commutative encryption and a binary vector. The binary vector is used to check if their set values are already encrypted and insert them into the union set. Once all the parties complete their insertion, which is confirmed by the values in the binary set, the whole union set can be decrypted to reveal the union set.

The secure intersection follows the same concept. However, here only the common values are released to the participating entities. This protocol is complex, and computations are very intensive.

Homomorphic Encryption

Encrypted addition and subtraction can be achieved through homomorphic encryption using public and private keys. There are three types of homomorphic encryption: fully homomorphic, partially homomorphic, and somewhat homomorphic. In partially homomorphic encryption, only one type of arithmetic operation (addition or multiplication) can be performed on ciphertexts. In somewhat homomorphic encryption, both addition and multiplication operations can be performed on ciphertexts, but only for a limited number of times. In fully homomorphic

encryption, arbitrary arithmetic operations can be performed on ciphertexts, but the computation cost is extremely high and is not practical for real-world applications.

Perturbation techniques

As the name suggests, plain text values are transformed into other values using encryption or morphing techniques to prevent privacy attacks. Once an entity's information is changed, the linkage quality naturally decreases, making it much harder to perform data linkage and extract information. However, privacy can be secured. There are many ways to conduct perturbation techniques, which we will review below.

a.) Adding Noise

Here, the data is perturbed by adding random values to a database without a known pattern [22]. This helps in preventing frequency attacks since commonality within the data has been diminished. However, the similarity between records does go down.

b) Differential Privacy

Unlike above where noise is added to the dataset [22], where the random variable based on a private seed is added to the statistically generated query results where there is more mathematical control.

c.) Embedding

Here the features are extracted from the dataset and then mapped to a Euclidean space or Hamming space by computing the distance created by embedded vectors. These vectors are created for all the datasets that need to be compared. The vectors are then compared to find matches without knowing the contents of the dataset.

d.) Hashing

A hash function is a one-way function [23] that maps an input to an array of values. Figure 2.2 shows a hash converter. If *Fi* is the input, then $O = H(Fi)$, where *O* is the hashed value and *H* is the hash function. However, it is not possible to obtain *Fi* from *O* using $Fi <- H^{-1}(O)$. In Chapter 5, we will delve further into the details of several types of hash functions and their advantages. The most common protocol used today is SHA (Secure Hash Algorithm), which encodes its input to a specific size of bits. Since this is a many-to-few mapping, there is a chance of collisions happening. This means there could be more than one value mapping to the same hash value. Over the years, algorithms have been developed to reduce the collision of records. One of the other features of a hash function is that it is deterministic. This means that given the same input multiple times, the output is always the same. However, any small variation in the input will result in a completely different hash output. We will use the term hash signature throughout this dissertation to signify a hash function output.

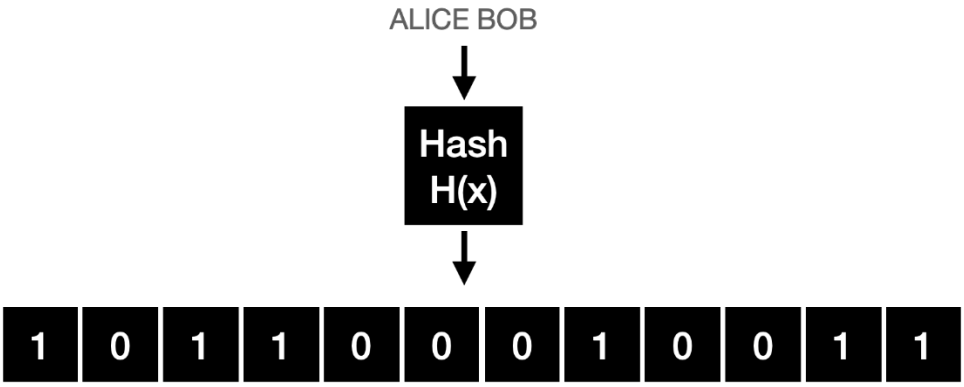

Figure 2.2 Example where “Alice Bob” is fed to a hashing function and outputs a hashed bit string

.

2.3 Fraud Indicators

In this dissertation, we will be using the term fraud indicator to represent information about an entity attempting or suspected of committing fraud. This can include someone's full name, government ID such as driver's license number, or an IP address. The fraud indicator may come with additional attributes such as place of origin, record time, number of fraud attempts, etc. Once the fraud indicator has been identified, its attributes (*Atrb*) can be transmitted.

Fi ($Atrb_1, Atrb_2, Atrb_3 \ldots Atrb_n$)

2.4 Similarity Measure

To determine how alike or close datasets are, similarity measures are used. To link datasets using PPRL, these measures are used in diverse ways in protocols. We will review the five most popular similarity measures and home in on the one best suited for our implementation.

2.4.1 Euclidean distance

Euclidean distance is the most used distance measure, which is simply the Pythagorean distance between two points. It can be calculated as the square root of the sum of the squares of the differences between the corresponding coordinates of the two points.

$$E = \sqrt{\sum (x^i - y^i)^2} \qquad (2.1)$$

This measure is most suited for comparing homologous data sets. It is commonly used with Locality Sensitive Hashing (LSH) protocols, where similar data sets are grouped into different

bands, and the Euclidean distance between the bands is used to determine the proximity of the data sets.

2.4.2 Manhattan distance

In this measure the absolute difference of the x coordinate and the y coordinate are added to determine the distance. $|x1-x2|+|y1-y2|$.

2.4.3 Cosine Similarity

It is the measure of the cosine of angles of the vectors that represent the data point or in other words it is the dot product of the two vectors.

$$SIM(X,Y) = \frac{X.Y}{(|X||Y|)} \qquad (2.2)$$

It is quite an effective measure to determine the similarity between documents and web pages and is extensively used in many PPRL protocols.

The three measures mentioned above, which are only a small representation of all the available measures, require some form of mathematical computation. This computation needs to be performed on homologous datasets, even if the data itself is encrypted. If performed on non-homologous datasets, the results will be difficult to compute. As discussed in Chapter 5, there are some Locality Sensitive Hashing (LSH) based hashing techniques that look promising for our protocol, but they may not be feasible for implementation in our framework.

2.4.3 Jacquard Similarity

Like the above-mentioned measure, the Jaccard similarity [24] is a measure of likeness between two datasets. It is the intersection value of the sets divided by the union value of the same,

i.e. $J(A,B) = \frac{A \cap B}{A \cup B}$. (2.3)

Although quite simple in concept, it is one of the most popular measures invented by Paul Jaccard and is powerful when used. This similarity measure will form the basis of our framework, and we will show how it can be elegantly used to calculate similarity on non-homologous datasets. We will delve into further details in Chapter 4 and explain how and why this measure works in a Telco industry setting.

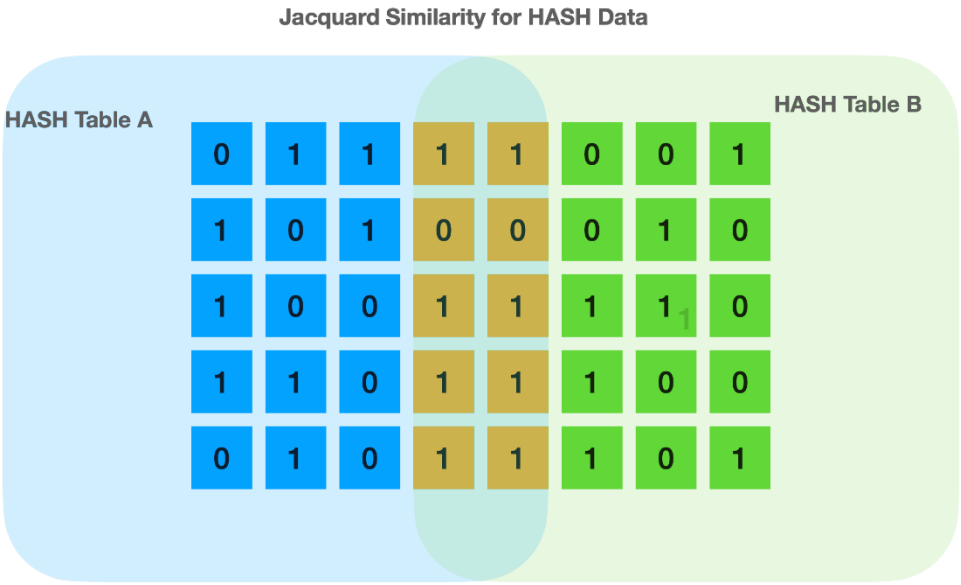

Figure 2.2 Shows a Jacquard Similarity representation on two hash tables where J=2/8=0.25

2.5 Encryption

Encryption is a process of converting plain text into a secret code (cipher text) to prevent unauthorized access. The cipher text can only be decoded with a key. The key is a piece of data used to encode and decode the message. Encryption is achieved through complex mathematical computations that scramble the original message into an unreadable format. The key is used to unscramble the message back into its original form.

There are two main types of encryptions: symmetric encryption and asymmetric encryption.

In symmetric encryption, the same key is used to both encrypt and decrypt data. This means that all parties involved in the transaction share the same key to encode and decode the data. Well-known implementations of symmetric encryption include AES and SNOW. However, if the key is shared, it could lead to a potential loss of privacy if the parties involved in the transaction are not fully honest.

K_P – Encryption and Decryption Key

E(Fi) <- Encryption (Fi,K_P*)* - Encrypt Fraud Indicator using shared key K_P

Fi <-Decryption(E(Fi), K_P*)* - Decrypt Fraud Indicator using shared key K_P

In asymmetric encryption, also known as public-key cryptography, two keys are used: a public key and a private key. The public key is used to encrypt data and can be distributed widely and openly. The private key is used to decrypt the data encrypted with the public key. Asymmetric encryption is more secure than symmetric encryption because the private key is kept secret and is not shared with anyone. Implementations like RSA and elliptic curve cryptography use this type of encryption.

K_{Pu} – Private Key, K_{Pr} – Private Key

E(Fi) <- Encryption (Fi, K_{Pu}*)* - Encrypt Fraud Indicator using public key K_{Pu}

Fi <-Decryption(E(Fi), K_{Pr}*)* - Decrypt Fraud Indicator using private key K_P

2.5.1 Elliptic curve cryptography (ECC)

Elliptic Curve Cryptography (ECC) [25] is one of the widely used public-key cryptography schemes. Typically, the key and the encryption protocol are based on a mathematical computation on two large prime numbers and the difficulty of factorizing the product to the original numbers. However, ECC is different from RSA in that it uses elliptic curves instead of prime numbers. The elliptic curve-based protocol is quite similar in concept to RSA, where the discrete logarithm of a random point on the elliptic curve with respect to a known point on the curve is infeasible.

There are several types of ECC:

a.) Elliptic-curve Diffie-Hellman scheme - This protocol [25] is another variant of the prime number based Diffie-Hellman scheme, combining a public-key concept, except here elliptic curves are being used.

b.) The Elliptic Curve Integrated Encryption Scheme

c.) Elliptic Curve Digital Signature Algorithm [25] (ECDSA)

d.) The Edwards-curve Digital Signature Algorithm [25] (EdDSA)

For our implementation, we chose the Edwards-curve Digital Signature Algorithm (EdDSA) scheme [25]. In Chapter 4, we go into the details of this protocol and the logic behind it. A digital signature scheme is a protocol where the authenticity of the document or data can be verified using a hash function. Typically, the data is signed with EdDSA using a private key and then verified against the original data using the public key. If the records do not match, then a False is returned.

Algorithm: Digital Signature Scheme

Input:

- *Fo*: Fraud data

- *Fi*: Fraud data to be verified

- *H*: Hash function

- *E*: Encrypted function

- *KPu*: Public key

- *KPr*: Private key

Output:

-True/False

Start:

1 *E* <- *Encode(H(Fo), KPr)*

2 Evaluate True or False <- Verify (F*i, KPu*)

3 If *Fi* = *Fo*, then True is returned; else False.

End

The Encode and Verify functions are part of the Edwards-curve Digital Signature Algorithm (EdDSA) scheme [25], which is a digital signature scheme using a variant of Schnorr signature based on twisted Edwards curves. You can find more details about this scheme in Chapter 4 of this dissertation.

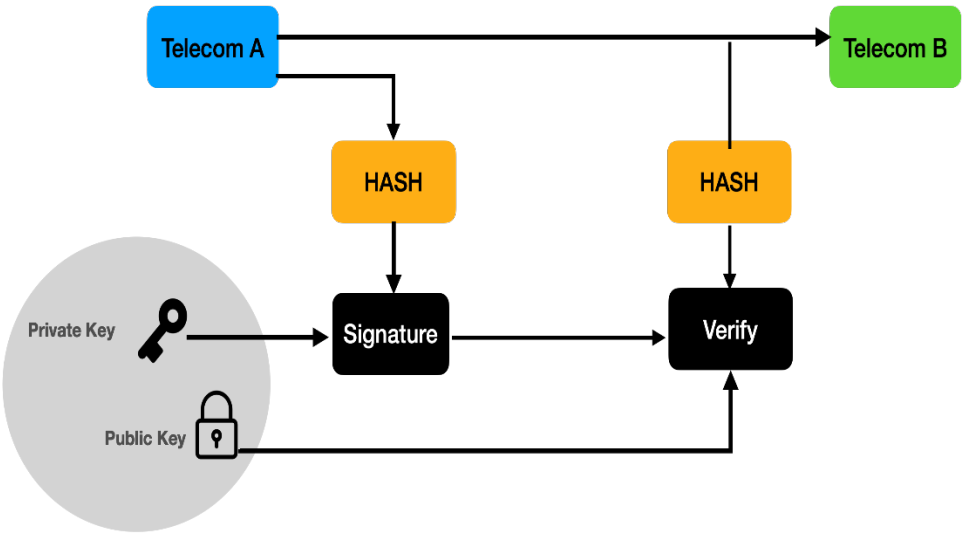

Fig 2.3 Digital Signature Scheme mechanism using EdDSA that will form the basis of our protocol.

2.6 Cryptanalysis attacks

Cryptanalysis is the study of analyzing encrypted data to determine how to make it more secure or discover its weaknesses. Cryptanalysis methods strive to decipher the plaintext from the ciphertext. There are many types of Cryptanalysis attacks that can be used to determine the strength or weakness of an implementation. Here are some of the most common types of Cryptanalysis attacks:

2.6.1 Ciphertext-only attack:

Here, the information on the ciphertext is only known. There is no information about the plaintext, key, or encryption algorithm. The analyst will try to decipher the ciphertext using known protocols and patterns to try decoding the ciphertext. This could be computationally very intensive.

2.6.2 Known-plaintext attack:

Here, the analyst or attacker will have access to some plaintext information related to the ciphertext. This could be information from a voter database, public websites, etc. Using this information by trying to correlate the plaintext to the ciphertext, the key can be determined. Once the key is determined and through existing crypto protocols, the crypto message can be decoded.

2.6.3 Differential cryptanalysis attack:

This is used on block ciphers and is analyzed in pairs of ciphertext to determine patterns that can then be extended to the encoded message to decode the same.

d.) Side-channel attack:

This is where the attacker overwhelms the database or PPRL system to make it compute an enormously complex problem by consuming its capacity. This will result in the system slowing down and rendering it incapable of responding to normal requests.

2.6.4 Dictionary attack:

This attack is based on exploiting the tendency of normal people to choose easy-to-remember passwords. Once all the words of the dictionary are encrypted for a given known protocol, then it is compared against the ciphertext to decipher the same.

2.6.5 Man-in-the-middle attack:

Here, the attacker intercepts the transmission of keys between the transacting parties and tries to exchange the key with individuals. In this scenario, the transacting parties are unaware of the presence of the attacker and believe they are communicating with each other directly.

2.7 Linkage Quality

While evaluating the effectiveness of PPRL Protocol linkage quality, it is important to measure the success of logic used. True Positive (*TP*) are where the intended matches are generated. False Positive (*FP*) are incorrectly generated matches and False Negative (*FN*) are correct matches not identified.

a.) Precision (*Prec*) is the number of true matches over total matches that occurred not counting the false negative,

$$\text{Pr}\,e\,cision = \frac{TP}{(TP+FP)} \tag{2.4}$$

b.) Recall (*Rec*) is the ratio of the true matches identified correctly to the overall complete matches

$$Recall = \frac{TP}{(TP+FN)} \tag{2.5}$$

c.) F-Measure used to measure the mean between precision and recall

$$FM = 2 \times \left(\frac{\text{Pr}\,ec \times Rec}{\text{Pr}\,ec + Rec}\right) \tag{2.6}$$

2.8 Privacy Analysis

This is a measure of how much information an adversary can gain from the encrypted data using some of the techniques and attacks mentioned in the above sections.

2.8.1 Entropy:

This is one of the measures of privacy is probability distribution or the measure of randomness in the encrypted data. The higher the entropy the more difficult it is for the data to be decoded. If r is the random variable and P(r) is the probability of deciphering this variable, then the entropy is

$$H(R) = \sum -P(r)\log\big(P(r)\big) \qquad (2.7)$$

2.8.2 Information gain (IG):

This is the measure of deciphering the original plain text from the ciphertext. If H is the entropy and P is the plain text and E is the encoded text, then *IG(D/E) = H(D) - H (D/E)*

2.9 Experimental setup

To set up our PPRL protocol and conduct the experiments, we used an Apple MacBook Air with an Apple M1 chip and 8 GB memory. We developed a Python package to simulate and SuperMinHash. For EdDSA, we specifically imported the D25519 libraries.

To reproduce a real-world fraud database for TB, we downloaded the data from the North Carolina publicly available voter database (https://www.ncsbe.gov/results-data/voter-registration-data) as opposed to using data generated by simulation software. The voter database (which has the advantage of having names used) and some of our manufactured names were used as a proxy for the fraud indicator Fi_A for T_A. Our dataset size for running analysis was set to 10,000 rows. We used 'RONALD JOHN ADAMS,' a familiar name, as T_B's fraud indicator Fi_B to determine if a

match was present in Fi_A. In the Python package, we set variable N to feed multiple signature sizes for SuperMinHash and multiple values from 0.1 to 1 for the Jaccard threshold value J_T. We embedded the Python time function between the various protocol stages to measure their run times. The protocol described in Section 3.3. The code and results are available at https://doi.org/10.5281/zenodo.8319042. We imported several libraries, including the ones related to EdDSA.

2.10 Chapter Summary

In this chapter, we discussed the concept of privacy and its importance. We then discussed how privacy can be protected using the method of Privacy-Preserving Record Linkage (PPRL) and then further defined Fraud-Detecting Privacy Protection Linkage (FD-PPRL), our protocol of fraud detecting privacy protection linkage. We discussed various adversary models laying a foundation as to why some of the PPRL techniques make use of the adversary model that best fits their need rather than assuming the worst-case scenario. We described some privacy-enhancing techniques and talked about hashing. We described several types of similarity measures and Jaccard similarity, which is what we will primarily use in our implementation. We introduced the digital signature scheme, especially the Edwards-curve digital signature algorithm (EdDSA), which we will go in-depth in Chapter 4. Several types of cryptanalysis attacks were described, and different linkage quality measures which we will use in our experiments. In the next chapter, we will review current research and literature with some analysis on their deficiencies.

Chapter 3: CURRENT RESEARCH IN PRIVACY PRESERVING RECORD LINKAGE

This chapter provides details on existing and current research in Privacy-Preserving Record Linkage (PPRL) methods. We will describe some fundamental classifications, their ease of implementation, and weaknesses. As we describe the implementation, we will highlight why these protocols are not suited for fraud detection implementation, especially in the telecom industry. In Section 3.1, we will review implementations with Bloom filters. In Section 3.2, we will review implementations without Bloom filters which will also include a review with and without a trusted third party. In Section 3.3, we will review the issues with current implementations. Finally, in Section 3.4 we will summarize this chapter.

Table 3.1 Notation and Terminology used in this Chapter

Hi	Set of hash function $\forall\, i \in 0...N$)
D1, D2	Database of partied transacting in PPRL
Di	Dataset $\forall\, i \in (0....N)$
D	Dice coefficient
ϵ_{inf}	Randomized response

PPRL is a type of entity resolution (ER) method [9] where two parties with databases *D1* and *D2* use a similarity measure, such as the Dice coefficient [10], to determine a link between the datasets. The receiver and sender do not reveal information about their datasets to maintain privacy. PPRL techniques have been in place for over a decade; however, owing to the size of the

records (100,000 and above) being compared, Bloom filters are used in [11] current implementations.

These techniques can broadly be classified into two main categories:

- Implementation with Bloom filters.
- Implementation without Bloom filters.

3.1 Implementation using Bloom filters

Bloom filters have become standard in PPRL protocols due to their simplicity in configuration over existing implementations. The use of Bloom filters for PPRL was first proposed by Randall et al [12] and was used to search medical records databases. Before we delve further into Bloom filters with PPRL, let us review what Bloom filters are.

3.1.1 Bloom Filters

Probabilistic data structures (PDS) are datasets where an inquiry in the same will not result in a fully deterministic answer. There will be a certain level of error in the response. This is used when speed needs to be achieved instead of accuracy. Bloom filters [12] are one such probabilistic data structure where they are used to evaluate the membership of a set.

Bloom filters are bit arrays of a certain length *L*. *K* independent hash functions Hi are defined for any (0…*K*-1) to map a set S = $\{a_1, a_2 \ldots\ldots, a_n\}$ into the Bloom filter [12]. Here, the indices from the hash function for an element a_i are set to 1 in the filter. The set membership can be checked by hashing the elements that need to be matched using the same hash functions. Bloom filters are prone to collision errors and yield false positives.

Here an initial empty set is taken as shown in Figure 3.1. There will be a set of Hash functions $Hi(x) \forall i \in (0...N)$ that will be used to map elements into the Bloom filter *(Bi(k)).* For a

data set *Di(x)* ∀ *i* ∈ *(0…M), m <- H(x)* where m will map to specific position on *Bi(k)* which get initialized to 0 before insertion.

An initial empty set is taken, as shown in Figure 3.1. There will be a set of hash functions *Hi(x)* ∀ *i* ∈ *(0…N)* that will be used to map elements into the Bloom filter. For a data set *Di(x)* ∀ *i* ∈ *(0…M), m <- H(x)* where *m* will map to a specific position on *Bi(k)*, which gets initialized to 0 before insertion.

To check if a set member is present in the filter, it is hashed through the *Hi(y)* function for all 1…*N*. Then, a check is done to see if the bit is 1 or 0 in those positions. If it is 1, then true is returned, indicating that the member might be present in the set. However, if it is 0, then we know for sure that the member is not present in the set. It is important to note that the true value is only probabilistic, meaning that there is a chance of false positives.

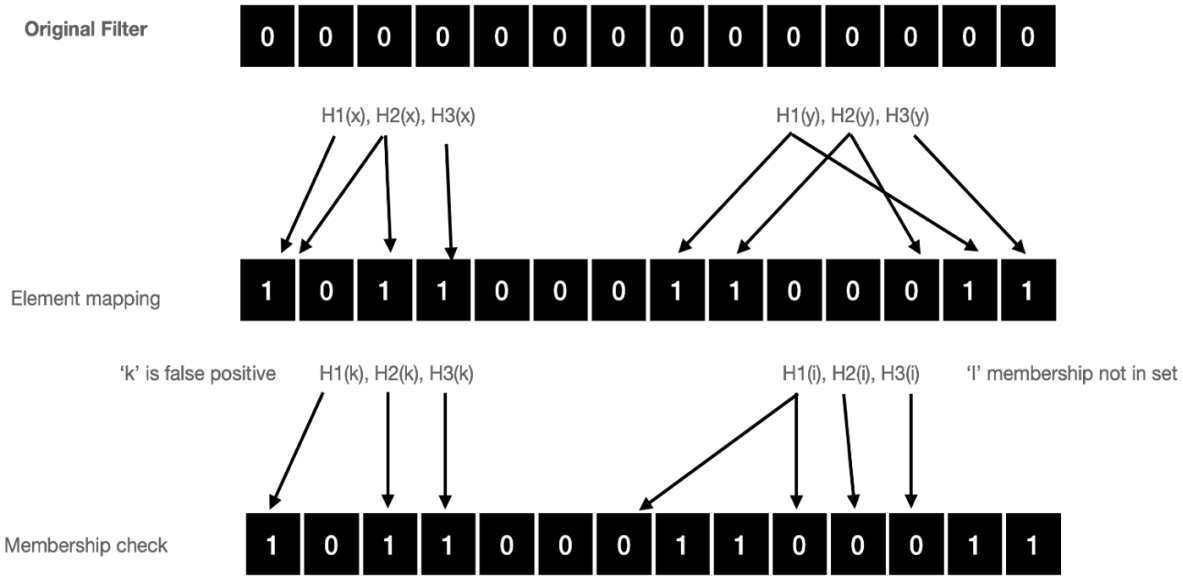

Figure 3.1 Shows the bloom filter insertion and membership check.

3.1.2 Dice Coefficient (D)

The Dice coefficient is a measure used to compare the similarity between two Bloom filters [13]. Given two populated Bloom filters A and B, the similarity measure D is defined as $D = 2h / (a + b)$, where h is the number of bits set to 1 that are common between A and B. In set A, a is the number of bits set to 1, and in set B, b is the number of bits set to 1. This measure provides a quick comparison of the Bloom filter sets.

Once the Bloom filters are populated using a set of hash functions, they can be compared using the Dice coefficient, which is a similarity metric. This can be used to perform privacy-preserving record linkage (PPRL). However, encoding large character lengths and comparing them can lead to matching errors. To address this issue, Bachteler [13] proposed a PPRL method using Bloom filters, in which the identifiers are split into q-grams for further encryption. Q-gram splitting is a way of separating data into chunks.

For instance, the phrase “privacy preserving” can be split into bigrams as ‘pr,’ ‘ri’, ‘iv’, ‘va’, ‘ac’, ‘cy’, 'y ‘,’ p’, ‘pr’, ‘re’, ‘es’, ‘se,’ ‘er,’ ‘rv,’ ‘vi,’ ‘in,’ ‘ng,’ and 'g '. Each of these word grams is then processed into the Bloom filter using K number of hash functions. These are then inserted into the Bloom filter and compared using the Dice coefficient. The advantage of using this approach is that comparisons can still be made even in the presence of spelling errors.

To prevent cryptanalysis attacks, a balanced Bloom filter was developed. This is where the total number of bits on the filter remains the same regardless of inputs. Schnell [11] used a PPRL method using balanced Bloom filters to further strengthen their implementation against attacks. Because eliminating rare patterns with a constant Hamming distance is challenging, Bloom filters

of a certain length are concatenated with a negated copy of the same filter. To prevent attacks on the Bloom filters, Schnell [11] proposed three main methods.

3.1.3 Random Hashing -

Initially, a set of all n-grams is generated for each identifier. For each n-gram, k random numbers are generated between 1 and the length of the Bloom filter [11], denoted as l, using a single password as a seed. Subsequently, k random positions within the Bloom filter are selected and set to one. This approach eliminates the need for hash functions, thereby increasing the difficulty of launching a pattern-based attack.

3.1.4 Balanced Bloom filters -

To mitigate attacks that leverage the Hamming weights of Bloom filters, Balanced Bloom filters [11] with a constant Hamming weight are utilized for PPRL. These filters are constructed by concatenating a Bloom filter of length l with a negated copy of the same Bloom filter. The resulting bit array, which has a length of 2 * l, must then be permuted. However, it is important to note that the increased length of Balanced Bloom filters and their constant Hamming weight could potentially lead to an increase in computation time.

3.1.5 Permanent Randomized response -

To enhance the security of the Bloom filter and the level of differential privacy (a concept where data is slightly altered to introduce noise), randomized responses are employed to flip the bits. It should be noted that the permanent randomized response [11] ϵ_{inf} satisfies this requirement.

$$\epsilon_{inf} = 2k \ln\left(\frac{1-\frac{1}{2}f}{\frac{1}{2}f}\right) \qquad (3.1)$$

Lazrig et al. [15] introduced an intriguing technique that employs Bloom filters and the Dice coefficient. This paper presents the concept of garbled circuits (GC). When two Bloom filters

are used to compute the Dice coefficient, it becomes possible for either party to identify some potential records based on the Dice coefficient. To prevent this, a Dice Coefficient Threshold is established. Records that do not meet this threshold are not exposed to either party. This ensures that only the intended records are revealed.

In this protocol, both the sender and receiver create their own set of secret keys. The receiver encodes its information using its secret key. Similarly, the sender encodes its information and the receiver's information with its secret key and sends it back to the receiver. The receiver then decodes the sender's information and performs a similarity comparison of the data that is encoded with the sender's key. It is important to note that neither party reveals their information. The protocol also uses Garbled Circuits, which is a method where two parties directly compute a function. In this case, the Oblivious Transfer Protocol (OT) is a secure method that operates without a trusted third party.

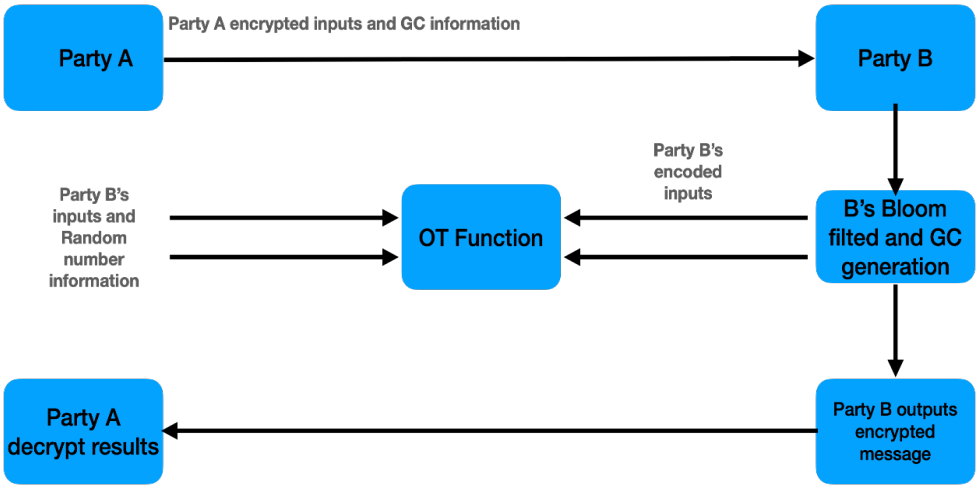

Figure 3.2 PPRL protocol using GC and OT functions between two parties.

A key point to note about this protocol is that no third party is involved in the transaction. However, the protocol requires the parties to communicate back and forth multiple times for each record. This can be computationally intensive.

D. Vatsalan et al. [4] describe the use of Bloom filters with q-grams, along with blocking/filtering for comparison. They introduced a concept called the Counting Bloom Filter (CBF). The CBF is a data structure where the count of a given set of elements is smaller than a given threshold. False positive matches are possible, but false negatives do not occur with this filter. Typically, many of the protocols are between two parties. However, Vatsalan and his team have introduced their protocol to work in a multiparty scenario, especially in conjunction with secure computing. This paper also describes the challenges with this implementation and various forms of attack such as dictionary, frequency, and cryptanalysis.

Following are some basic steps of the protocol.

Step 1:

The database owners agree upon the Hash function Hi, Bloom filter length Bl, similarity threshold, and Quasi Identifier (QID) Attributes. A QID is a data element that identifies or represents the entity under evaluation.

Step 2:

Each party applies the block function to reduce the number of records to be evaluated for comparison, rather than having their entire databases processed.

Step 3:

The database owners hash maps Hi of their QID into their Bloom filters. The owners set the Bloom filter parameters to ensure the PPRL qualities of complexity, quality, and privacy.

Step 4:

The parties initiate a secure summation protocol with the linkage unit (LU) sending a Random Vector ® to the candidate to be summed with their Bloom filter vector BF. The candidate then sends the summed vector to the next candidate until all the members of the group are covered. The final summed amount is then sent to the LU.

Step 5:

The LU then generates the *CBF c = R[cs] + Hi* and calculates the DC to determine the matches for the candidate record and provides them back with this information. As can be seen, the protocol is quite cumbersome and involves back and forth between parties. It is complex in implementation and privacy can be compromised.

S. Randall et al. [13] also describe the use of Bloom filters with blocking for large datasets and validate the efficacy of their algorithm by measuring the linkage quality. Large-scale datasets from New South Wales and the Western Australian hospital system, with data spanning 10 years and over 20 million records, were used to analyze privacy and linkage quality. According to the paper, the personal information was removed, and only the clinical information was shared for analysis.

W. Xue et al. [16], M. Alaggan et al. [17], and R. Schnell et al. [11] have all developed numerous implementations of differential privacy using Bloom filters. Some of them have achieved privacy by introducing noise to further enhance secrecy while still performing record linkage. The paper states that the current implementations of the Bloom filter are limited to simple types of matching techniques using strings and numerical values. The matching techniques use machine learning algorithms to calculate the similarity measure. To make this implementation more secure, a Laplace-like perturbation method is used on the Bloom filter to add some noise to the data. This ensures that the data is not discriminated against due to collisions.

Ranbaduge et al. [5] present an interesting implementation using deep learning methods to enhance differential privacy with Bloom filters. These methods have become quite popular, and quite a few implementations include machine learning as part of the protocol. In this paper, their protocol is defined between two dataset owners, D_A and D_B. The cross product of the datasets yields two sets: one with matches (M) and one without matches (N). The matched record pairs (ri, rj) represent the same entity. If this is true, they will belong to M, and if not, they will belong to N. The process classifies the data into these two sets.

Through this method, there are two main phases: the training phase and the classification phase. Different database owners collaboratively train the LU. So, when the unclassified records are sent by the DO to the LU, they get classified into M and N and are sent back to the DO. Due to security and privacy concerns, many DOs are not willing to share their data to train the LU.

3.2 Implementation without Bloom filters

In this section, Bloom filters were used in the implementation. While they can be extremely useful in PPRL protocols, they are extremely prone to cryptanalysis attacks. In this section, we will review some protocols that do not use Bloom filters.

Meadows [18] introduced a method in which a trusted third party is not involved in the transaction. Here, the parties (A, B) exchange information while maintaining the privacy of their datasets using the Diffie–Hellman protocol.

Algorithms used in this protocol:

1. A and B each generate a random number and exponentiate it modulo *P*, where *P* is a large prime number known to both parties.
2. A sends the result to B, and B sends the result to A.

3. A and B each exponentiate the received value with their secret number modulo P.

4. A concatenates the result with its secret message and sends it to B, and B concatenates the result with its secret message and sends it to A.

5. A and B each verify that the received message is the same as the one they expected.

Transacting parties A and B have their secret keys *SA* and *SB*. They encode their messages *MA* and *MB* with their respective keys. A sends MA^{SA} to B, and B sends MB^{SB} to A. A and B, in turn, encode their shared messages using their keys and return them to each other. A then has $MA^{SA \times SB}$, and B has $MB^{SA \times SB}$. Subsequently, they can both determine if *MA* is equal to *MB*.

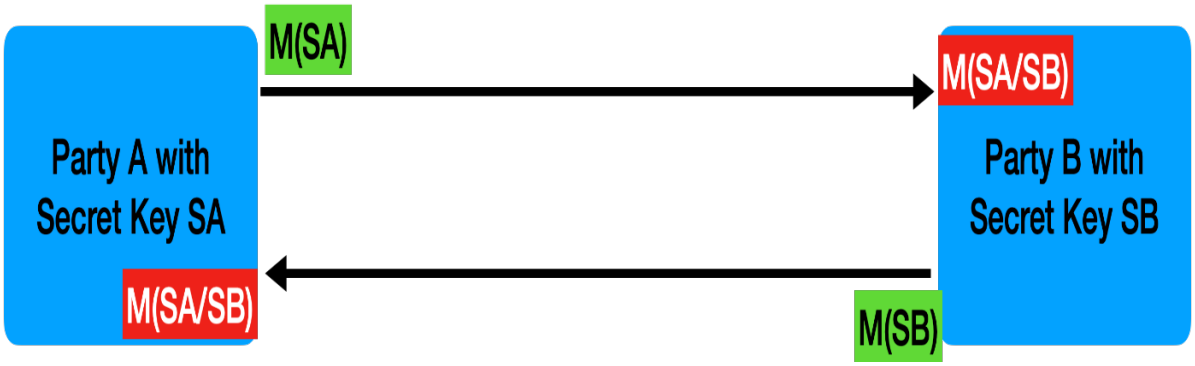

Figure 3.3 PPRL depiction using Diffie-Helman protocol

Vaisri et al [5] introduced an implementation without Bloom filters. Here, the QIDs are split into q-grams. The DOs in this protocol want to find the Longest Common Extension (LCE) on the elements in their datasets. These q-grams are hashed using functions agreed upon in advance. These hashed q-bits are then shifted. The bit table is also generated for the q-grams. The bit table and hashed data are then sent to the Link Unit (LU) to be used for processing and matches. The LU calculates the Largest Common Array (LCA) and sends it back to the parties involved. The LCE is calculated based on the algorithm below using the LCA. Even though Bloom filters

are not used, a bit table like the one in Bloom filters is still being used in this protocol. This implementation has a semi-trusted third-party linkage unit to perform matches and inform the respective parties

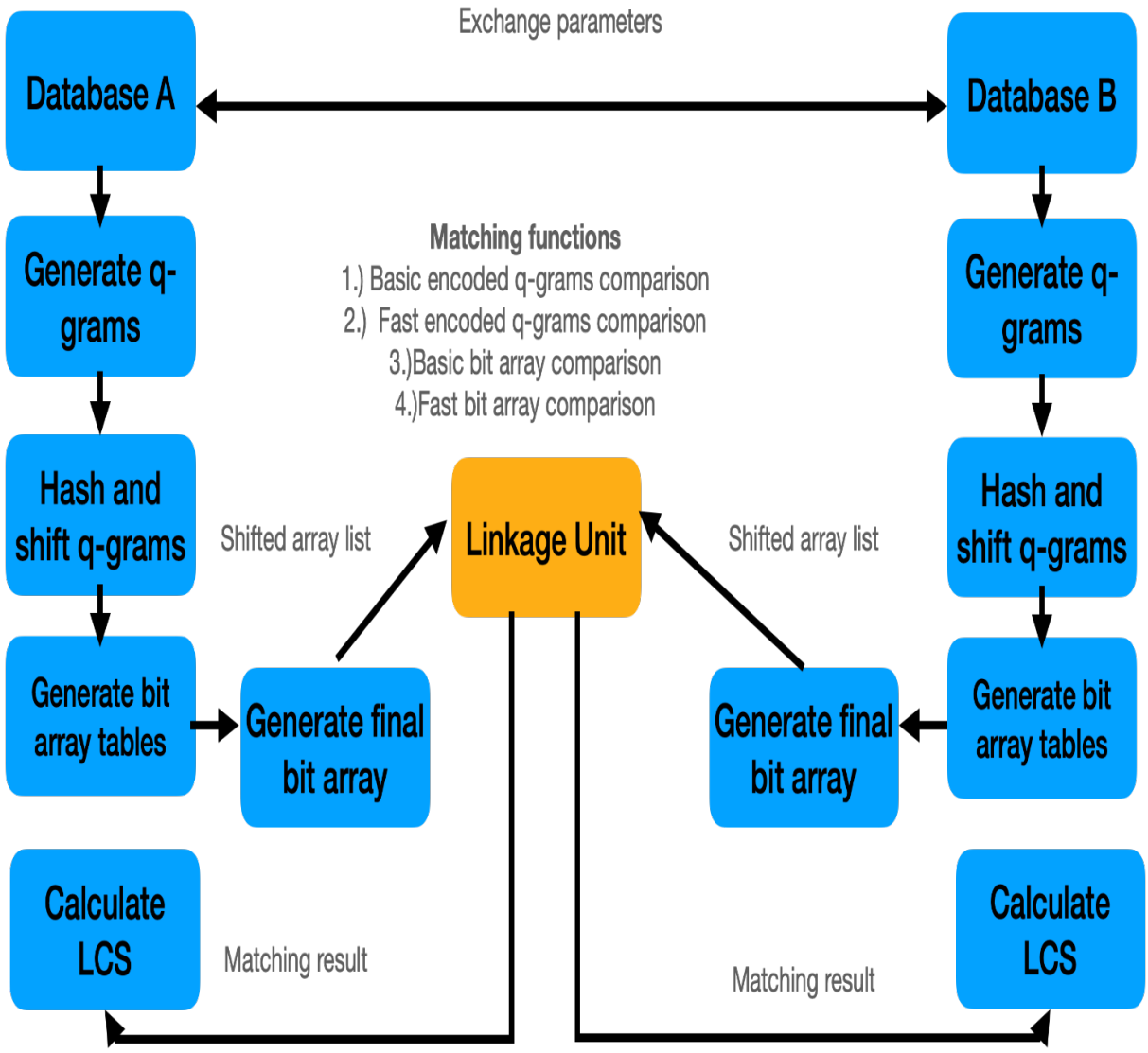

Figure 3.4 Shows the PPRL protocol with a Linkage unit

Algorithm 1: Basic Encoded Q-Grams Comparison Process by the Linkage Unit (LU)

Input:

- hx: This is the list of q-grams from string x that have been hashed and shifted.

- hy: This is the list of q-grams from string y that have been hashed and shifted.

Output:

- lce: This is the Longest Common Extension (LCE) of the pair of hashed q-gram lists.

Steps:

1 Initialize the length of the LCE to zero (lce ← 0).

2 Check if there are any common hashed elements between hx and hy (if set(hx) ∩ set(hy) = ∅).

3 Loop over each position in the list hx (for $0 \leq px < |hx|$).

4 For each position, get the list of positions in hy where the element hx[px] occurs (py ← get PosMatch(hx[px], hy)).

5 Get the current shifted (or rotated) list of the list hx (x ← hx[px:] + hx[:px])

6 Loop over each position in the list of positions py (for py ∈ py).

7 For each position, get the current shifted list of the list hy (y ← hy[py:] + hx[:py]).

8 Initialize the index and common count k to zero (k ← 0)

9 Loop over x and y if a common element occurs (while (k < min (|x|, |y|)) and (x[k] = y[k]))

10 Increment k (k ← k + 1).

11 Keep the length of the so far maximum length of LCE (lce ← max (lce, k)).

12 Return the found length of LCE back to the Data Owners (DOs) (return lce).

In their 2004 study, Churches [19] and colleagues proposed a method for sharing data securely. They suggested that the shared data should be divided into q-grams and then encrypted using a private-public key method. The matching process is performed on this encrypted data.

In this protocol, an entity known as an Honest but Curious Party (HBC) is considered. It is assumed that both parties (A and B) can trust the HBC. The HBC will not reveal A's information to B or B's information to A.

The protocol employs a modified version of the Dice coefficient to determine matches. The Dice coefficient (DC) is calculated as follows:

$$DC = 2 * |bigram(x) \cap bigram(y)| / (|bigram(x)| + |bigram(y)|) \quad (3.2)$$

Here, a bigram is a two-letter decomposition of a word. For instance, if x = "Sally", then bigram(x) would be {"sa", "al", "ll", "ly"}.

Both A and B agree upon a secret Key Lab. They each split their words into bigrams, encode them using each other's public keys, and send them to HBC. HBC runs a comparison between the two datasets and compares the DC, then relays the results back to A and B. For records that match or meet a threshold, the information is secured by A and B using their keys and transmitted.

This implementation assumes that both transacting parties trust each other and are susceptible to cryptanalysis. As can be seen, this process is computationally complex, involving multiple transactions and exchanges. Also, the processing uses an HBC to determine the DC. Once

the HBC has a good quality of encrypted data and DC, it can conduct cryptanalysis to determine the contents of A and B's sets.

Smith [20] introduced a method that does not involve splitting the data into q-grams or using Bloom filters. A pseudo-randomization method is used, which replaces the actual data with a secure pseudonym. A similarity measure, such as the Dice coefficient, is used to determine a match. The concept of the Expectation Maximization (EM) algorithm was introduced. A record pair for comparison is sent as an input to EM, and the records are categorized as matched (M) or unmatched (U). The categorization is based on the DC and a set threshold for matches.

The protocol above also considers weights given to the matched pairs so they can affect the value of the DC to elicit a response from EM. The record pairs themselves are pseudonymized (i.e., replaced with other random values) and then used for comparison. As discussed in many protocols, such approaches are still prone to frequency attacks, rendering the overall protocol vulnerable.

3.3 The problem with current PPRL methods

- They are not all secure. Most, if not all PPRL methods use some form of a similarity measure to determine a match. The latest cryptanalysis techniques use graphs to exploit similarity measures even when determined on encrypted data as demonstrated by Vindanage et al. [7]. This renders all current PPRL methods insecure. Also, many PPRL implementations use Bloom filters, as per Schnell, et al [11]. These implementations are prone to frequency attacks and are not secure, as per Schnell [21], using various cryptanalysis techniques over the *q*-grams of a Bloom filter.

- Many use a trusted third party to transact secured attributes [5]. This technique is employed to ensure that the two parties do not reveal their information.
- They are complex to implement, slow and have multiple transactions. Several PPRL methods [11] [4] use many convoluted variations of the Bloom filter with counting and randomized implementations.

3.4 Summary

In this chapter, we reviewed the various current implementations of Privacy-Preserving Record Linkage (PPRL) techniques. We reviewed implementations with Bloom filters, which is one of the most popular methods of implementing PPRL. However, Bloom filters are easy to implement, but lack the robustness to withstand cryptanalysis attacks and are extremely prone to privacy losses. Within the Bloom filter implementation, we also reviewed methods with and without a trusted third party. In most of the implementations, assumptions have been made that the trusted third party is honest, however, it is very likely at a minimum the third parties will be honest but curious. We also evaluated various implementations without Bloom filters and finally summarized why all these implementations are not suited for a Telecom industry application1. This then leads us to Chapter 4 where we will review our framework that will address all these deficiencies and provide a safe, secure, and fast implementation to conduct PPRL.

Chapter 4: Fraud Detecting Privacy Preserving Record Linkage (FD-PPRL) Framework

In this chapter, we present the Fraud Detection Privacy Protection Protocol (FD-PPRL) framework that telecommunication companies can use to share fraud-related information with each other. The FD-PPRL protocol, as described in this chapter, will need to meet the primary criteria for adoption. In Section 4.1, we briefly summarize the issues with current implementations, which demonstrate the need to design our protocol. In Section 4.2, we describe the research gap and our motivation for the fraud detection protocol. In Section 4.3, we summarize our main contributions. In Section 4.4, we briefly explain the selection of the encryption and MinHash Scheme. In Section 4.5, we describe the FD-PPRL framework with proof of why the FD-PPRL can only be used to compare over two non-homologous datasets. Finally, in Section 4.6, we go over experiments to demonstrate the performance and privacy-related aspects of our protocol. We conclude the chapter with a summary in Section 4.7.

Introduction

Fraud is a growing and persistent problem in the telecom industry, and telcos work in isolation to prevent it. However, sharing information is critical for detecting and preventing fraud. Telcos must comply with privacy laws and protect their business secrecy when sharing information. The primary challenges are: 1) sharing fraud information without revealing data; 2) sharing information without involving any third party; and 3) using a protocol that is safe, secure, fast, and easy to implement. Privacy-preserving record linkage (PPRL) methods can be used to achieve these goals. Several techniques have been developed to share data while preserving

privacy using PPRL. However, most of the PPRL techniques use a similarity measure like the Jaccard Similarity on homologous datasets, which are all prone to graph-based attacks, rendering existing methods insecure. Many complex and slow techniques use the Bloom filter implementation that can be compromised in a cryptanalysis attack. To address these issues, we propose an attack-proof PPRL method using existing infrastructure of a telco without a complex multistep protocol. Our method uses a novel way of matching two non-homologous datasets using attack-proof digital signature schemes, like the Edwards-curve digital signature Algorithm (EdDSA). Here the Jaccard similarity can only be estimated using our method, and not on the datasets directly. Two parties transact with a simple request-reply method. To validate the match accuracy, privacy preservation, and performance of our approach, it was tested on a large public dataset (North Carolina Voter Database). Our method is secure against attacks and achieves 100% match accuracy with improved performance.

Table 4.1 Notation and Terminology used in this Chapter

$T_A, T_B, T_C ....T_N$ -	Notation to represent Telecom companies
$D_A, D_B, D_C, ..., D_N$ -	Fraud database of the N Telecom companies
KA, KB	Private and Public Key Pairs
J, J_S	Jaccard Similarity
J_T	Jaccard Threshold
Ei_A	Encrypted Fraud Indicator
Hi_A, Hi_B	Hash of the Fraud Indicators Fi_A and Fi_B
S_H	SuperMinHash Algorithm
N	Signature Length
H	Entropy of the protocol
p	probability of determining the bit in the protocol

Table 4.1 Notation and Terminology used in Chapter 4

4.1 The problem with current PPRL methods

- All current PPRL methods are prone to cryptanalysis attacks – Most current implementations use some similarity measure to perform matches. The similarity measure is determined on a dataset encrypted using the same methods. Vindanage et al. [7] used graphs on similarity measures to conduct cryptanalysis and showed that all PPRL implementations are prone to attacks. Even in the implementation by Meadows [18], where there is no similarity measure, the use of a Diffe–Hellman protocol or any other primitive key exchange technique provides weak security features.
- Usage of Bloom filters – All Bloom filter PPRL implementations are highly prone to cryptanalysis attacks, as demonstrated by Christen et al [27]. Cryptanalysis is a method in which attackers, assuming they have the encoded Bloom filter, exploit its behavior where a certain value, be it a *q*-gram, frequency, or hamming distance, occupies specific locations. Christen et al. [27] demonstrated that knowledge of all the parameters or encoding process is not needed; only access to public databases with names is needed to extract a match. Their cryptanalysis even overcomes some of the Bloom filter hardening techniques.
- Usage of third parties – Implementations such as the one by Vaiwsri et al. [7] use a semi-trusted third party to perform matches. Third party cannot be used in the Telcom industry due to privacy laws. Furthermore, telcos may not agree on a mutually trusted third party. Using a third party also involves multiple transactions between parties to determine a match, as shown by Vaiwsri et al. [7].
- Complex transactions and speed – Many implementations process data into *q*-grams, extract the frequency of characters and hamming distance, and pseudo-randomize the data to have additional layers of security. This leads to issues with the processing speed, adds

complexity in implementation for the telecom industry, and leads to precision errors in matching.

4.2 Research gap and motivation

The primary issue with all current PPRL methods is that the sets of data being compared are encrypted using the same method. The individual data might be secure; however, they still have homogeneity infused between each other. This enables an adversary to calculate a similarity measure even on the encrypted data and hence determine the original data through graph attacks [7]. To secure against this attack, a method to measure the similarity needs to be developed on data processed through completely different protocols and only by the receiver. We only need to transmit the encrypted data to be compared against the unencrypted data. This ensures that the two sets of data are always completely different, and the adversary cannot get any information from the transmitted encrypted data.

Our study proposes a new PPRL method that: 1) is secure, and fast; 2) is based on a simple request–reply protocol to share fraud data between telcos; and 3) does not require a trusted third party.

4.3 The Main contributions are:

- A novel method to determine Jaccard index on nonhomologous datasets by using 1) a MinHash data transmitted after encryption using an attack-proof digital signature scheme (DSS), and 2) normal MinHash data that is not transmitted. The similarity measure can only be determined on the datasets using our method.
- The two parties transact through a simple request-reply method where the sender sends the digital signature without revealing the original message, as usually done in a DSS.
- The match is directly done on the MinHash data unlike other implementations that use

complex q-grams or feature extraction methods.

We propose using a SuperMinHash technique demonstrated for speed [26] over an Edwards curve digital signature algorithm (EdDSA) known similarly for speed and security [25] modified only to send the encrypted signature without the original message. The Jaccard index (J) satisfies the triangular inequality, unlike the Dice coefficient [28] which is calculated using the verification function of EdDSA. We also propose setting an optimal threshold measure (J_T) and a signature length (N) that render our PPRL method to be both highly secure and fast.

4.4 Proposed FD-PPRL Method for Fraud Detection Between Telcos

Telcos identify, intercept, and prevent fraud daily; however, they do not share this information due to various constraints. Telcos can choose to store information regarding fraud-related activities, called fraud indicators, such as name (first, middle, and last name), postal/physical address, network address or location, social security number, and account number. The telcos can directly transact fraud indicators with each other without a trusted third party using the FD-PPRL method described below, without revealing each other's information with a simple request–reply method.

In this section, we present the details of the FD-PPRL approach by first proposing the selection of a secure encryption scheme (Section 4.4.1) and an efficient MinHash algorithm (Section 4.4.2). These will then be incorporated into the FD-PPRL design (Section 4.5) to develop a new secure matching protocol.

4.4.1 Encryption scheme selection

Selecting a protocol for secure data transmission is vital for a successful PPRL implementation, especially when no third party is involved. Several cryptographic techniques are based on the Diffie-Hellman key exchange. Since the invention of the symmetric and asymmetric

key exchange, many secure protocols, such as DSS, ElGamal, and RSA, have been developed.

We use a DSS where we incorporate a MinHash into the protocol to calculate the Jaccard index over the encrypted data. This keeps the transaction between telcos within a simple request–reply method where they do not reveal each other's data. In a typical digital signature implementation, the message's signature is sent along with the original message. We have designed the protocol so that only the encrypted signature is sent and not the message itself. This ensures privacy as the receiving telco will need to go through our protocol to determine a match. We use EdDSA for our implementation, which is a DSS that uses a Schnorr signature based on twisted Edwards curves. We used ED25519, a variant of EdDSA [25] widely known to be secure with smaller signature lengths, enabling faster transmission. The alternative to EdDSA is Elliptical Curve Digital Signature Algorithm (ECDSA). However, ECDSA is slower in performance, and is vulnerable to side-channel attacks [29], making EdDSA the best choice for our implementation.

Edwards-curve digital signature algorithm (EdDSA) using Schnorr Signature

EdDSA uses elliptical curves of the form $y^2 = x^3 + Ax^2 + x$ [25]. G is the generator point (x,y) that satisfies the above equation, and o is the subgroup order of the elliptical curve points generated by G.

Creating EdDSA Key Pair function: *(EdDSA.generateKey())* (4.1)

Private and public (K_A, K_B) key pairs are generated. K_A is hashed, and the last bits (8 for ED25519) are cleared along with the highest bit, thereby ensuring the secrecy of the private key. K_B is a point on the elliptical curve, given by $K_B = K_A * G$.

Creating EdDSA Signature function: *(EdDSA.encode())* (4.2)

Hash the message (M_1) and the secret key using a secure hash function.

Step 1:

$h=Hash\ (Hash\ (K_A)+ M_1))\ mod(o)$, a secret number h is generated using the private key K_A.

Step 2:

$H_1=Has((h*G)\ +K_B+ M_1)\ mod(o)$, h from Step 1 is multiplied by the Generator to get the elliptical point behind h and is combined with the public key K_B and Message M_1.

Step 3:

$E_A=((h*G)\ +H_1*K_A)\ mod(o)$, H_1 from Step 2 is multiplied by the private key K_A and combined with the elliptical point behind h. The signature is now (h, E_A).

Verifying the EdDSA Signature function: *(EdDSA.verify())* (4.3)

During the verification process, only the public key signature (h, E_A) is sent. In our implementation, the message is not sent. The receiver uses what they think might be the message, which is hashed, to determine if the message is valid or invalid (Boolean output).

The verifier using their message (M_2), calculates the hash using the same hash function as the sender.

Step 1-

$H_2=Hash(h+K_B+M_2)\ mod(o)$, combine the sent secret number from Equation (4.2), the public key, and the receiver's message M_2.

Step 2 -

$Check1=E_A*G$, multiply the generator G to get the elliptical point for E_A from Equation (4.2).

Step 3 -

$Check2=h+(H_2*K_B)$.

Step 4 -

If *Check1==Check2*, then M_1 is the same as M_2.

4.4.2 MinHash method selection

The similarity measure we chose to determine a match between two datasets *D1* and *D2* is the Jaccard index: $J = [0,1]$, where $J=1$ when $D1=D2$. This was initially used to measure ecological diversity [24] and is now widely used across various fields. Unlike the Dice coefficient, the Jaccard index satisfies the triangular inequality [2] and is a better similarity measure. Broder [30] introduced MinHash to estimate the Jaccard index. This method is based on the fact that when the datasets are hashed (*H1, H2*), the probability of minimum hashes through a random permutation is remarkably close to the Jaccard index:

$$J_S = P\ (min(H1) = min(H2)) = \frac{(D1 \cap D2)}{(D1 \cup D2)} \quad (4.4)$$

While MinHash provides a faster way to estimate the Jaccard index, it also offers a layer of security because it is a one-way function. Furthermore, while determining the source of the hash might be possible, it is computationally demanding and time-consuming.

Since Broder [30] introduced MinHash, several types of MinHash algorithms have been developed, such as HyperMinHash [31], MinMaxHash [32], ProbMinHash [23] and SuperMinHash [26], which have improved the speed and accuracy of the Jaccard index computation. We used SuperMinHash in our implementation owing to its accuracy and speed.

Ertl used Fisher-Yates shuffling in his implementation of SuperMinHash to randomly permute the records. The typical MinHash algorithm has an $O(NM)$ runtime complexity, where N is the signature length and M is the number of elements in the dataset. The SuperMinHash algorithm achieved a significantly better runtime complexity of $O\ (N+M\ log^2 M)$.

4.5 FD-PPRL Method

The proposed method involves two telcos (A and B) who transact between themselves to share information on a potential fraud indicator without 1) revealing the actual contents of each other's fraud indicators or 2) using a trusted third party for transactions.

Because the dataset to be shared is large (greater than 100,000 records), the fraud-related dataset is categorized into blocks. The dataset is categorized using specific parameters such as state, zip code, or region. The telecom providers can share the block categorization [33] to configure their database in that format.

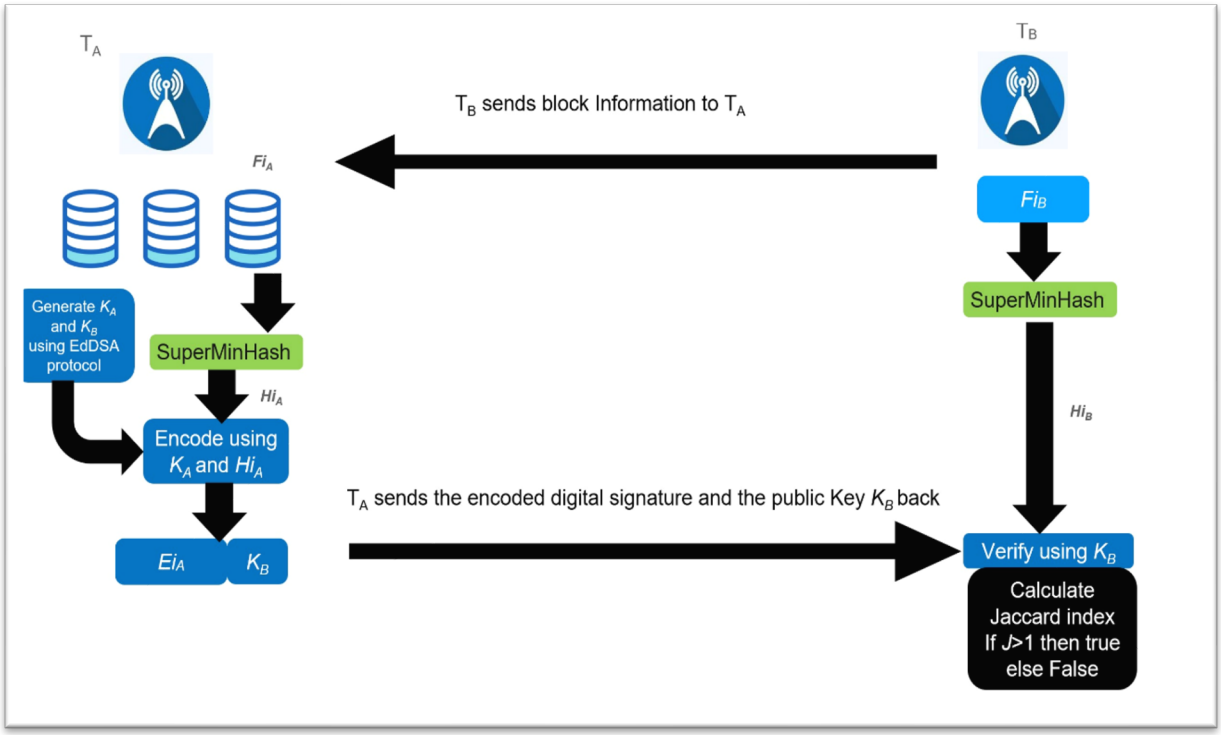

Figure 4.1 Jaccard index calculation for a fraud indicator using SuperMinHash and EdDSA

Fig. 4.2 shows the fraud indicator PPRL protocol between telecom A (T_A) and telecom B (T_B), where T_B enquires T_A about a fraud indicator. Given the apparent business secrecy and privacy law restrictions, T_A and T_B cannot share their actual fraud indicators. Therefore, through

our PPRL technique, T_A and T_B do not reveal their fraud indicators, but T_B can confirm fraud using information from T_A.

Proposition:

Jaccard similarity of the original message F_A and F_B can be estimated using the encrypted MinHash of F_A, Ei_A and unencrypted MinHash of F_B, Hi_B only using our method.

Proof:

As given in equation (4.4), $J = \frac{(F_A \cap F_B)}{(F_A \cup F_B)}$

MinHash function on a Message F_A and F_B maps it to a signature of length N, Hi_A and Hi_B respectively for any $I \in$ (0....N-1) using k independent hash functions (H_k).

Where $Hi_A := arg\ min(H_k(F_A))$ and $Hi_B := arg\ min(H_k(F_B))$

The probability that Hi_A is equal to Hi_B is same as J [30]. This property helps with the estimation of J using.

$$J'(Hi_A, Hi_B) = \frac{1}{N}\sum_{i=0}^{N-1} I(Hi_A = Hi_B) \qquad (4.5)$$

Where I is an indicator function that returns a 1 when there is match and 0 otherwise. The variance of J' and J is given by $var = \frac{J(J-1)}{N}$ (4.6)

[26] showing that the variance goes down as the signature size increases.

Thus $J'(Hi_A, Hi_B) \approx J$

Ei_A is the EdDSA signature of Hi_A determined using Function (4.2)

$$J'(Ei_A, Hi_B) = \frac{1}{N}\sum_{i=0}^{N-1} I\,(Ei_A = Hi_B) = 0$$

Since $Ei_A \neq Hi_B$ for any $I \in$ (0....N-1), proving that the Jacquard Similarity cannot be estimated

directly on Ei_A and Hi_B

However, using Function (4.3) as the identity function

$$J^e(Ei_A, Hi_B) = \frac{1}{N}\sum_{i=0}^{N-1} EdDSA.verify(Ei_A, Hi_B, K_B) \quad (4.7)$$

where K_B is T_A's key sent to T_B. *EdDSA.verify()* will return a 1 when Ei_A matches as a signature of Hi_B, else it will return 0.

Thus $J'(Hi_A, Hi_B) = J^e(Ei_A, Hi_B)$ proving that Jaccard similarity can be estimated using two completely nonhomologous data sets using our method

---

Algorithm 4.1: Logic to determine fraud

Inputs:

Fi_A, Fi_B – T_A, and T_B's fraud indicators respectively

H - Hash function shared between A and B (SuperMinHash)

N - Size of the hash signature

M – Size of the block of signatures transmitted by T_B

S_H – SuperMinHash algorithm

Outputs:

J_S - Jaccard indicator

K_A - T_A's private key

K_B -T_B's public key

Match = True if $J_S > J_T$ (Jaccard threshold); else, False

Steps:

$(a_0, a_1, a_2, \ldots\ldots a_{N-1}) \leftarrow [0,1]$

$(a_0, a_1, a_2, \ldots\ldots a_{N-1}) \leftarrow (0, 0 \ldots . 0)$

1 $(H0_A, H1_A, H2_A \ldots . HN_A) \leftarrow S_H(Fi_A)$ generate SuperMinHash for T_A's fraud indicator.

2 $(H0_B, H1_B, H2_B \ldots . HN_B) \leftarrow S_H(Fi_B)$ generate SuperMinHash for T_B's fraud indicator.

3 $K_A, K_B \leftarrow EdDSA.generateKey()$ using Function (4.1). Performed by T_A

4 $Ei_A \leftarrow EdDSA.encode\ (K_A, Hi_A)\ \forall\ i \in (0,1 \ldots (n-1))$ using Function (4.2)

5 T_A transmits K_B and Ei_A (0,1...M-1) to T_B (steps 1 to 5 are performed for all the block contents)

6 T_B performs

7 For $j \leftarrow$ 0,1, 2…...M-1 do

8 Ej_A

9 For $i \leftarrow$ 0,1, 2…...N-1 do

10 $a_j \leftarrow EdDSA.verify\ (Ei_A, Hi_B)$ using Function (4.3)

11 End For.

12 $J_S = \frac{1}{N}\sum_{j=0}^{N-1} a_j$ (4.8)

13 If $J_S \geq J_T$

14 Then break

15 End For

16 $J_S \geq J_T$ then *Match*=True; else, False.

Fig. 4.2 describes the process as follows:

Initial Setup and parameter definitions

Step 1 - T_A and T_B agree on a signature length (N), which is the output of the SuperMinHash algorithm and is the number of elements in the signature.

Step 2 - The block indicator can be a zip code, state, or region from which T_B wants information

from T_A. The number of elements in the block that TA decides to send to T_B is M, which comprises the fraud indicators (Fi_A) for any (0....M-1).

Step3 - Fi_B is the fraud indicator that T_B needs to determine if T_A has in its database.

Step4 - Hi_A is the SuperMinHash of Fi_A for any $i \in$ (0....M-1), and Hi_B *i*s the SuperMinHash of Fi_A; both adopt N as their signature length.

Step5 - Ei_A for any (0...M-1) is the encoded signature for Hi_A.

Step6 - J_T is the threshold value that can be set to any number [0,1], which can be compared against the calculated Jacquard index(J) to determine a match.

- T_B requests information from T_A. T_B has identified Fi_B and wants to enquire T_A about the presence of Fi_B in its data. However, T_B does not send or reveal Fi to T_A. It only sends the block indicator to T_A.

- T_A replies to T_B. At this point, T_A does not know the fraud indicator of T_B. T_A generates the public and private keys (K_A, K_B) using the EdDSA Function (4.1). T_A runs the SuperMinHash algorithm on Fi_A for any $i \in$ (0....M-1) using the preset signature length N to generate their respective hash signatures Hi_A for any $i \in$ (0....M).

- T_A encodes Hi_A using the EdDSA Function (4.2) and its private key K_A to generate the encoded signature $Ei_A \forall$ i $\in$ (0...M-1) and transmits it to T_B along with the public key K_B (which can be further secured by another public/private key method). Unlike the regular DSS where the original message is sent for signature verification, here the original message of T_A, Fi_A is not sent. This ensures that T_B has no knowledge of any of the fraud indicators of T_A.

- T_B determines a match. T_B hashes Fi_B using the same SuperMinHash algorithm. T_B lacks knowledge of the contents of the encoded hashed fraud data from T_A and can only verify a

match. T_B cannot determine the private key of T_A from Ei_A. Each element of Ei_A is iterated through the EdDSA verification Function (4.3) along with the elements of Hi_B to determine a match, which is then used to calculate the Jaccard index J_S shown in Algorithm I. This Jaccard index is compared against a Jaccard threshold J_T to return a match of True or False, thereby validating whether the indicator of B is fraudulent based on the data of A without decoding it.

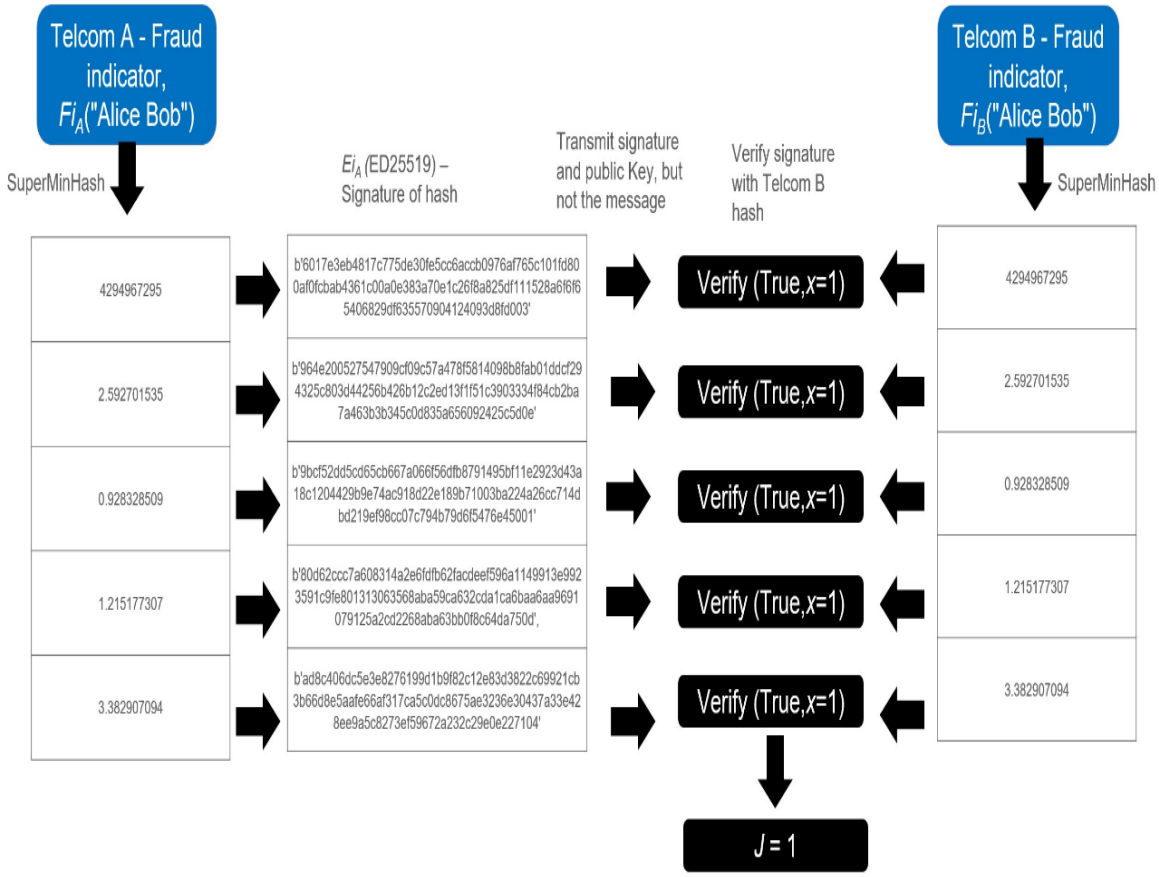

Figure 4.2 Example of the protocol for a signature size of 5 for Algorithm I

Fig.4.3 shows the simulation of the algorithm I in which we use a block signature of size N=5. Here the fraud indicators (Fi_A, Fi_B) for T_A and T_B are selected as 'Alice Bob.' Fi_A and Fi_B are sent through the SuperMinHash algorithm to produce the five numerical hashes shown in the figure. The numerical hash for T_A is input into *EdDSA.encode()* using Function (4.2) to generate

the five long alpha-numeric encoded signatures. The signature is then sent to T_B, which uses the *EdDSA.verify()* Function (4.3) function on each of the five signatures and its numerical hash, as shown in the figure, where the function returns a value of 1 if there is an exact match. The Jaccard index J_T is calculated using Equation (4.8), where a_j=1for any $j \in [0,4]$ in this scenario resulting in J_T=1, indicating a perfect match. The Python code mentioned in the experimental results, which is available on GitHub, can effortlessly reproduce this simulation.

4.6 Experimental Results

We first present the details of the experimental setup in Section 4.6.1 and then discuss the three experiments conducted on the protocol and their validity in Section 4.6.2. In Sections 4.6.3 and 4.6.4, we review the experimental precision of the PPRL method and the privacy assessment of the protocol, respectively. In Section 4.6.5 we compare our PPRL implementation against some of the current methods.

4.6.1 Experimental Setup

To set up our PPRL protocol and conduct the experiments, we used an Apple MacBook Air with an Apple M1 chip and 8 GB memory. We developed a Python package to simulate the protocol described in Section 4.5. The code and results are available at https://doi.org/10.5281/zenodo.8319042. We imported several libraries, including the ones related to EdDSA and SuperMinHash. For EdDSA, we specifically imported the ED25519 libraries.

To reproduce a real-world fraud database for T_B, we downloaded the data from the North Carolina publicly available voter database (https://www.ncsbe.gov/results-data/voter-registration-data) as opposed to using data generated by simulation software. The voter database (which has the advantage of having names used) and some of our manufactured names were used as a proxy for the fraud indicator Fi_A for T_A. Our dataset size for running the analysis was set to

10,000 rows. We used 'RONALD JOHN ADAMS,' a familiar name as T_B's fraud indicator Fi_B to determine if a match was present in Fi_A. In the Python package, we set variable N to feed multiple signature sizes for SuperMinHash and multiple values from 0.1 to 1 for the Jaccard threshold value J_T. We embedded the Python time function between the various protocol stages to measure their run times.

4.6.2 Experiment measurements and validity

The experimental analyses are explained below.

Experiment 1: Measuring the run-time values for the algorithm using a signature size N = 5 and a match in Fi_A with a block size M =10,000 to demonstrate the performance of our protocol

- Run time for SuperMinHash algorithm on the fraud indicators Fi_A and Fi_B – 0.9 ms.
- Run time for the *EdDSA.encode* Equation (4.2) *(ED25519)* creation on Fi_A and Fi_B – 2 ms.
- Run time for *EdDSA.verify* Equation *(4.3)* on Ei_A and Fi_B – 1 ms.
- Run time for Jaccard index determination – 7 ms.
- Total run time – 10.9 ms
- Average file size for the digital signature of Fi_A, Ei_A as in encoded with EdDSA for 10,000 records – 6 MB. It takes under 0.1 s to transmit Ei_A on a 1 GB connection.

Experiment 2: Measuring the performance and validity of the algorithm for various signature sizes when a match on Fi_B is not present in Fi_A

We collected the total run-time measurements of our PPRL method using signature lengths N between 3 and 45, as shown in Fig 4. The bars in orange show the total run time in seconds to

SuperMinHash Fi_A and $Fi_{B,}$ generate $Ei_{A,}$ verify using Ei_A and $Fi_{B,}$ and calculate the Jaccard index. As the signature length increases by 5, the unmatched run time increases by approximately 100 s as the protocol would iterate through all the data in Fi_A. This is the expected behavior of the PPRL method, thereby validating the experimental setup and collected data.

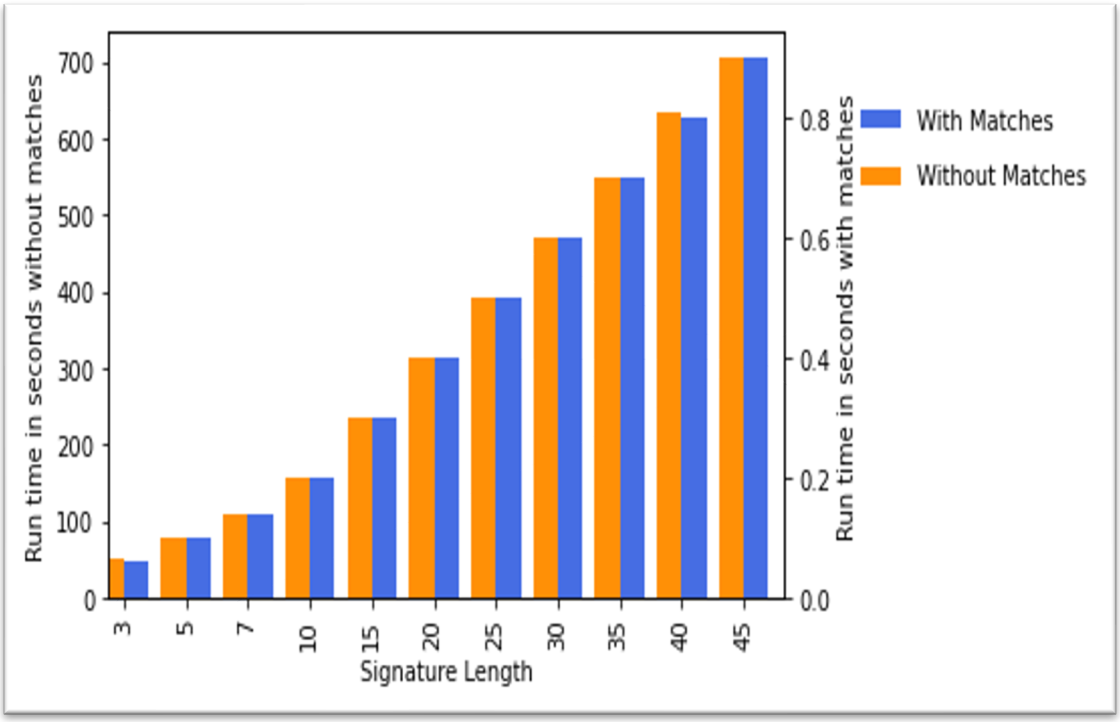

Figure 4.3 Performance of matched vs. unmatched for various signature length.

Experiment 3: Measuring the performance and validity of the algorithm for various signature sizes when a match on Fi_B is present in Fi_A

In Fig 4.4, the bars in blue show the total run time using signature lengths N between 3 and 45. As the signature length increases by 5, the matched run time increases by 0.1 s owing to all the iterations through Fi_A. The run time in Experiment 3 is much shorter than that in Experiment 2, as shown in the side-by-side comparison in Fig 4.4, by 1/1000th of a second, as Algorithm I exhibit a designed break when a match is found. The behavior of the measured data is in line with the expected behavior of the PPRL method, thereby validating the experimental setup and collected

data.

4.6.3 PPRL Method precision definition and measurement

We used precision as a measure to qualify the performance of our method:

$$Precision = \frac{TP}{(TP+FP)} \quad (4.9)$$

where *TP* is the number of true positive matches, and *FP* is the number of false positive matches. Here the precision values range from 0 to 1, with 1 indicating a perfect match.

Experiment 4: Measuring the protocol precision for various signature lengths.

Four signature lengths N = 5, 10, 15, and 20 were used to measure the precision Equation (4.9) of the PPRL method, as shown in Fig 4.5. We measured the precision by varying the Jaccard Threshold (J_T) from 0.1 to 1, incrementing by 0.1. We used the same experimental setup mentioned in Section 4.6.1 to find a match for Fi_B in Fi_A. As demonstrated in Fig 4.5, our method has a precision of 1, even with the thresholds as low as 0.5; the precision values fall with values of J_T <0.5. The figure also shows that our method has high precision with low J_T values as the signature length increases. Higher signature lengths provide more values to match, significantly reducing *FP* and leading to a precision value of 1 at low J_T. Increasing the signature length to increase precision at low J_T values does sacrifice the method performance. However, based on the figure, an ideal signature length and threshold can be implemented to achieve exact or high precision while giving an optimal run-time performance. Our experimental setup's optimal values are J_T >= 0.6 with N=7.

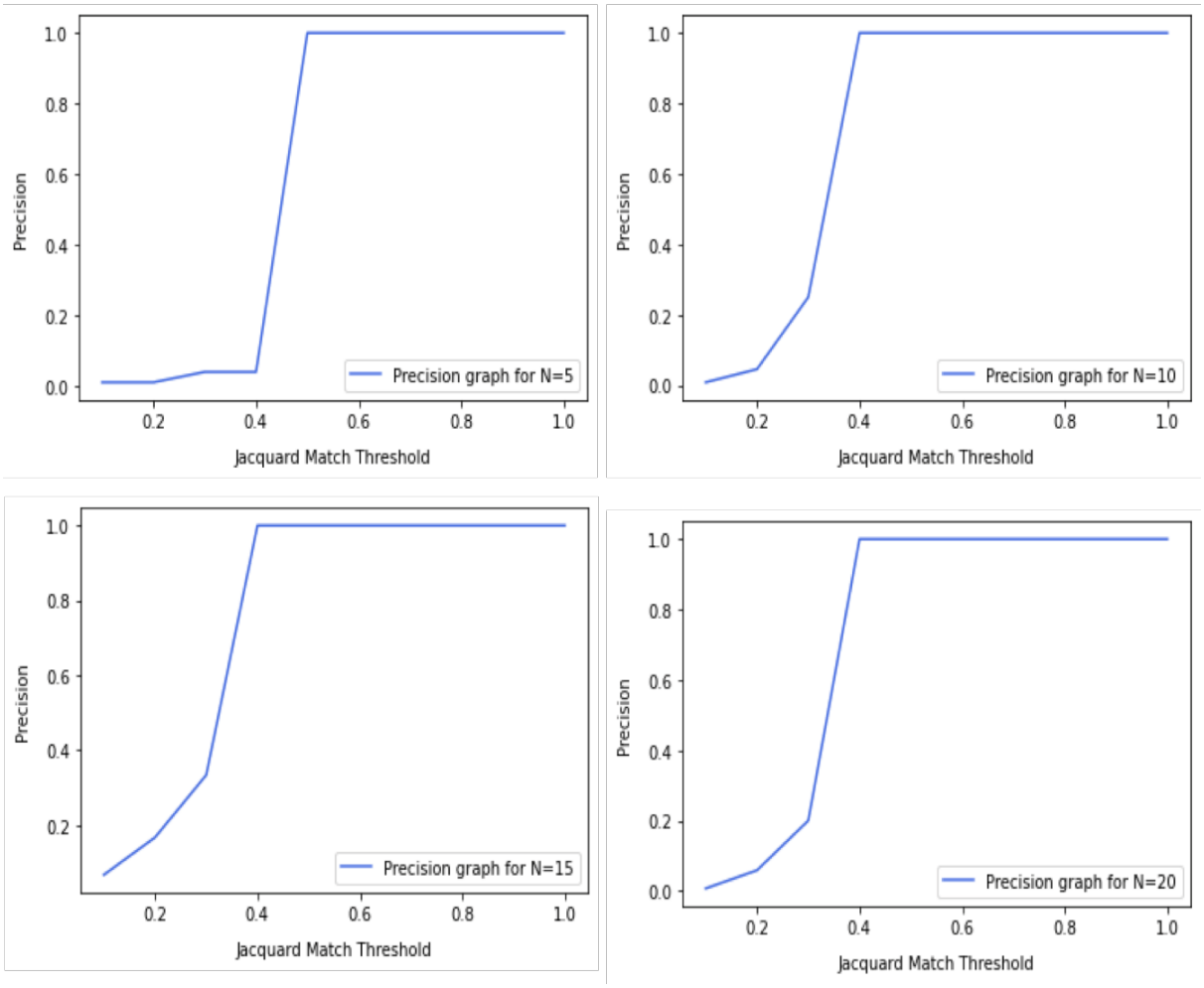

Figure 4.4 Comparison of the PPRL method sensitivity to find a match for the Jaccard threshold (JT) for various SuperMinHash signature lengths N

4.6.4 Privacy analysis

T_A sends a block of information (we used 10,000); it must be clear if T_B could run the brute force analysis on the dataset to reveal information in the unmatched dataset. Before discussing the possibility of T_B receiving information from unmatched records, we verified the possibility of a third-party adversary determining this information. Unlike a typical EdDSA transaction where the message is exchanged between the parties, in our implementation, no message is transacted except the digital signature and public key. The EdDSA protocol, especially ED25519, ensures tight security. We recommend that the public key is transacted only to the intended party through a

secure asymmetric key mechanism, such as RSA. Based on existing literature (Bernstein, 2012; Brende et al., 2021) have published theories and proof on the security of EdDSA, especially ED25519. The typical byte size of an EdDSA signature is 512, making it almost impossible to determine the signature through frequency attacks. (Probability is of the order of $1/2^{512}$ and is even lower as the signature length increases by $1/2^{512*N}$.)

Frequency /cryptanalysis is extremely difficult on our protocol, and we need to analyze if T_B can use brute force to gain information on the encoded digital signatures using EdDSA.

The entropy (H) of the digital signature (S) is shown in (4.7)

$$H(S) = \sum_{j=1}^{K-1} -p_j \, log\left(p_j\right) \quad (4.10)$$

where p is the probability of determining the signature with a length of K bits.

Conditional Entropy is the uncertainty in determining the signature given Jaccard threshold $J < 1$.

$$H\,(S|J< 1) = J \times H(S) \quad (4.11)$$

Experiment 5: Measuring the entropy of our PPRL to demonstrate its security and privacy.

Table 4.2 lists the calculated values of the conditional entropy (4.11) when $J \leq 1$ for signature lengths $N = \{5,10,15,20\}$. The bit length of ED25519 protocol is K=512 for N=1, and the probability of discovering one bit is p=1/2. The entropy values drop as J approaches 0 and increase with N. Even when $J = 0.1$, all the entropy values are extremely high, demonstrating the security of our method. Even when T_B yields false positives with $J < 0.5$, it is challenging to determine the value of the non-matched fraud indicators in Fi_A. Vidanage et al. [7] conducted a graph-based cryptanalysis attack of similarity-based PPRL methods, assuming the adversary has access to the original message. However, this attack is ineffective against our PPRL method

because the original message is never sent and cannot be determined by frequency attacks or even with false positive values when $J<=1$. This shows that our proposed method has the highest privacy and is very secure.

Signature Length	J=1	J=0.5	J=0.2	J=0.1
5	385	192	77	38
10	770	385	154	77
15	1155	577	231	115
20	1541	770	308	154

Table 4.2 Conditional entropy values for some Jaccard values ≤ 1 with various signature lengths

Experiment 5.1: Conduct Cryptanalysis attack by Telecom B on the data set sent by Telecom A

We conducted frequency-based cryptanalysis attack as mentioned in P. Christian et al. [27]. The experiment's purpose was to determine the resistance of such attacks when the data is shared between telcos. Experiment 5 shows the robustness of the protocol during data transmission, which is secure. For our experiment we used records from the North Carlina Voter database which is transmitted through our protocol to Telcom B. Assuming Telecom B is a semi honest adversary we use a common first name and a common First and Middle name with 10,50, and 100 occurrences to conduct cryptanalysis attack. Figure 4.6 shows the results of the cryptanalysis attack where the occurrence of exact match is 0, and graphs show the percentage of partial match and no match. When we used common first, middle, and last Name no matches were found further demonstrating the security of our protocol.

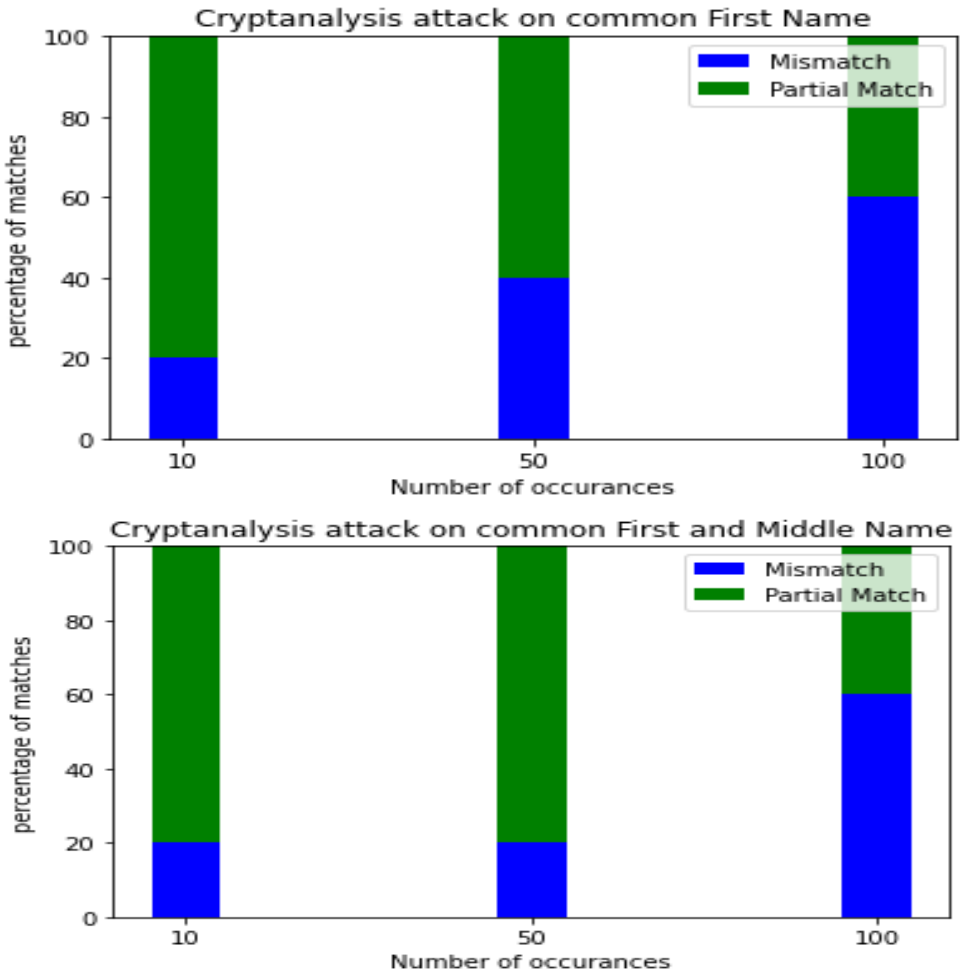

Figure 4.5 Cryptanalysis attack performance of our protocol

4.6.5 Comparison of our PPRL method against other implementations

We use four criteria to evaluate the effectiveness of our protocol against various PPRL methods.

- Use of Bloom filters referred to as Blm-Fltr from these papers D. Vatsalan et al., [4], M. Antoni et al [34], and S. Randall et al. [12] for comparison. We already know that they are prone to cryptanalysis attacks [27].
- Implementation complexity owing to multiple protocols as shown in papers such as W. Xue et al. [20], M. Alaggan et al., [17], and R. Schnell et al., [11] as adding noise for

differential privacy leading to additional processing of the data. We will represent this as Blm-DP. T. Ranbaduge [22] uses deep learning techniques with bloom filter which we will represent as Blm-DL.

- Employment of third parties or linkage units to perform matches as shown in. T. Lazrig et al. [15]. This will be represented as Partition GC-RL in our comparisons.
- The execution speed of the protocol will be used to evaluate the above three criteria.

Experiment 6: Measuring the execution time of our PPRL method for various data sizes (M) and its corresponding comparison against many other protocols.

As shown in Table 4.3, we executed various data sizes using our PPRL method and experimental setup. As demonstrated, our PPRL framework performed much better against many other protocols mentioned above such as Blm-Fltr, Blm-DP, Blm-DL, and Partition GC-RL based on the four criteria mentioned above.

PPRL Method	28000 records	5000 records	600 records
Blm-Fltr	462	58	11
Blm-DP	473	73	13
Blm-DL	491	82	19
Partition GC-RL	647	96	25
Our Method	304	54	6

Table 4.3 Comparison of various PPRL protocols to our method using execution time as a measure in seconds.

Experiment 7: Measuring the precision of our PPRL Method against many other protocols.

Table 4.4 shows the precision based on Equation 4.9 of our protocol against other protocols such as Blm-Fltr, Blm-DP, Blm-DL, and Partition GC-RL As demonstrated our protocol achieved 100%

precision due to efficient design without having additional complexity to maintain privacy.

Method	Precision
BF	0.89
DP-BF	0.91
DLBF	0.86
DP-DL-BF	0.8
Partition GC-RL	0.99
Our Method	1

Table 4.4 PPRL precision comparison against other protocols

4.7 Summary

Telecom fraud is an increasingly worrying issue in the telecom industry and the lack of data sharing and a suitable protocol between telcos only perpetuates this problem. We explained the fundamental flaw in current implementations where data when encoded using the same methods can be easily compromised, along with many PPRL protocols that use error prone Bloom filters or third party to perform matches. Our study addressed this significant flaw of PPRL implementations and proposed a secure, fast, and accurate fraud detection system for the telecom industry that uses a SuperMinHash implementation designed into a DSS, EdDSA. We have proved and demonstrated how the Jaccard index can be used to find a match across two nonhomologous sets of data and yet maintain its privacy using only our method. We conducted experiments to demonstrate the speed of PPRL and even compared our implementation with current methods, finding that our protocol provides a remarkable improvement in performance and a superior ability to maintain privacy. Telcos can use our protocol to share fraud data with short response times and

high security while complying with privacy laws. Since our implementation currently links only to the Fraud Indicators to determine its presence in the database, we plan to extend the protocol to extract other valuable information and still maintain privacy.

Chapter 5: FD-PPRL Framework Evaluation with different MinHash Techniques

In the preceding chapter, we introduced a unique protocol that utilizes the ElGamal Digital Signature Scheme for PPRL. This protocol was further refined to include SuperMinhash, enabling us to compute the Jaccard Similarity and assess matches. While we have explained our rationale for choosing EdDSA and basic MinHash, our aim is to test four different MinHash methods within our protocol to identify the most effective one, or at least to provide a range of options suitable for various scenarios. Section 5.1 outlines the standards we used to assess the different minhash techniques before settling on the most suitable one for our protocol. In Section 5.2, we delve into the application of the MinHash technique in the FD-PPRL protocol. Section 5.3 evaluates the implementation of the Weighted MinHash protocol within FD-PPRL. In Section 5.4, we examine the SimHash protocol in conjunction with FD-PPRL. Moving on to Section 5.5, we perform a conceptual analysis of the four methods when used with the FD-PPRL protocol. In Section 5.6, we carry out an empirical evaluation of these four protocols. Finally, in Section 5.7, we summarize the chapter and provide recommendations that are most appropriate for FD-PPRL

Introduction

In the telecom industry, as described, telcos continuously face fraud attacks. It is critical to quickly identify and prevent fraud. While sharing information is important, it is equally important to protect privacy. Speed and accuracy are extremely important to measure the effectiveness of fraud prevention.

Table 5.1 Notation and Terminologies used in this chapter

Fi_A, Fi_B	T_A, and T_B's fraud indicators respectively
N	Size of the hash signature
J_S	Jacquard Similarity
J_T	Jacquard Threshold
T_A	Telecom Provider A
T_B	Telecom Provider B
H_S, H_{SA}, H_{SB}	Hash signatures, and Hash signatures for Telco A, and Telco B
SH_A, SH_B	Shingles of Fi_A, Fi_B Fraud Indicator for A, and B
p1, p2, p3	Prime numbers for MinHash functions
K_1….K_N	Large set of integers to map the Fraud Indicator
PRNG	Pseudo Random Number Generator
WMH_B, WMH_A	Weighted MinHash Samples
SMH_A, SMH_B	SimHash signature for Telco A and B
MH	MinHash protocol
WMH	Weighted MinHash Protocol
SMH	SimHash Protocol
SuMH	SuperMinHash Protocol
Z	Amount of shingles from the fraud indicator
A_E	Accuracy of the EdDSA protocol
A_M	Accuracy of the FD-PPRL protocol

Table 5.1 Notation and Terminologies

We evaluated several models of the protocol described. This section will focus on the Hash protocols used to determine a Similarity Measure, such as Jacquard or Cosine Similarity, and which protocol best fits the needs of the telecom industry. To better evaluate these protocols, we will be using the Jacquard Similarity measure, calculated using various MinHash techniques, to evaluate accuracy, speed, and security.

5.1 Overview

In Chapter 3, we describe how two Telcos T_A and T_B interact with each other to share Fraud Indicators. Typically, there are many Minhash protocols to choose from to encrypt the Fraud Indicator. As we evaluate each Minhash algorithm, we will fit the same into our protocol and evaluate the following areas: 1) Ease of implementation, 2) Speed, 3) Accuracy of the results, and 4) Privacy. Figure 1 shows which area of the protocol we will be evaluating for our analysis. We will maintain all other aspects of the protocol static. We will start our evaluation with the simplest MinHash and build upon more robust implementations.

5.2 FD-PPRL implementation using MinHASH Protocol

MinHash is a technique used in computer science and data mining to estimate the similarity between two sets. It was introduced by Andrei Z. Broder [30] in his paper. The main aspect of the algorithm is a hash function which takes the Fraud Indicator Fi_A and Fi_B and maps it to a signature set of numbers with no collision. This signature set is then used to estimate the similarity between two sets. The idea behind MinHash is to represent each set as a signature, which is a small set of values that can be compared to other signatures to estimate the similarity between the two sets.

To ensure a fair comparison, we will set the size N to be the same across all the protocols. This means that the same number of values will be used to create the signature set for each protocol. By doing this, we can compare the performance of each protocol on an equal footing.

5.2.1 Protocol Definition

A simple representation of a hash function for an input number x is H(x) = (p1x + p2) % p3, where p1 and p2 are random numbers and p3 is a prime number greater than x. For different values of p1 and p2, the hash function will produce random mapping values which will constitute a hash signature of size N. For our version of MinHash, we will use a pseudorandom number generator (PRNG) that will be seeded using the Telco's fraud indicator. This will then be used to derive the MinHash signature. The Minwise hashing takes the Fraud Indicator Fi and maps it to N size hash signature H.

H_s (Fi):=arg min H(Fi)

Using the above equation derive the signature H_{SA} and H_{SB} for the fraud indicator Fi_A and Fi_B, The probability that H_{SA}= H_{SB} the same as the Jacquard similarity i.e., P (H_{SA}=H_{SB}) = J. This property allows the unbiased estimation of J.

$$J(H_{SA} = H_{SB}) = \frac{1}{N}\sum_{i=1}^{N} \mathrm{I}(H_{SA} = H_{SB}) \qquad (5.1)$$

Where I is the indicator function that returns a 1 when the inputs are equal else a 0.

5.2.2 Parameter Definition:

p1, p2, p3 are extremely large prime numbers used during the MinHash process. The fraud indicator, either directly or through the shingling process (see the definition and process in the background section), will be mapped to a large set of integers (K1…….KN). Typically, N, being

the signature size, will dictate the number of large integers that are an output of the MinHash process. These integers will then be the signature of the fraud indicator.

5.2.3 Record Encoding

During the MinHash stage itself, there is no record-level encoding. The MinHash technique itself offers a decent amount of security since fraud indicators are mapped to large integers. To get to the large integers, the algorithm considers large prime numbers in the conjectured one-way function, which will be difficult to reverse into the original input. G<-f(Fi). However, $Fi=f^{-1}(G)$ (this is f inverse) is hard. Although the core encryption is not at the MinHash layer, it does offer a level of security from basic cryptanalysis attacks.

Algorithm 5.1 Logic to determine Fraud using simple MinHash function

---

Inputs:

Fi_A, Fi_B – T_A, and T_B's fraud indicators respectively

J_T – Jaccard Threshold

H - Hash function shared between A and B

N - Size of the hash signature

MH – MinHash algorithm

SH_A, SH_B - Shingles of the Fraud Indicator Fi_A, Fi_B

Z - Amount of shingles from the fraud indicator

Outputs:

J_S - Jaccard indicator

K_A - T_A's private key

K_B - T_B's public key

HS_A, HS_B – Telcom A and B hash signatures of Fi_A and Fi_B

Match = True if $J_S > J_T$ (Jaccard threshold); else, False

Steps:

1 (a_0, a_1, a_2, ... a_{N-1}) [0,1]

2 (a_0, a_1, a_2, ... a_{N-1}) <- (0, 0, …., 0)

3 Telcom B generates minhash of its fraud indicator

4 SH_B <- Shingle-process (Fi_B)

5 $H_{SB}1$, $H_{SB}2$, ..., $H_{SB}N$ <- inf, inf, …., inf

6 SH_B

7 For i <- 0,1…Z-1 do

8 Initialize a pseudo random generator r with a seed SH_B

9 For j<-0,1…. N-1 do

10 $H_{SA}j$ <- min ($H_{SA}i$,r)

11 End For

12 End For

13 Telcom A generates minhash of its fraud indicator

14 SH_A <- Shingle-process (Fi_A)

15 $H_{SA}1$, $H_{SA}2$, ..., $H_{SA}N$ <- inf, inf, …., inf

16 For k<-0,1…Z-1 do

17 SH_A

18 For i <- 0,1…Z-1 do

19 Initialize a pseudo random generator r with a seed SH_A

20 For j<-0,1…. N-1 do

21 $H_{SA}j$ <- min ($H_{SA}i$,r)

22 End For

23 End For

24 End For

25 K_A, K_B <- EdDSA.generateKey()

26 Ei_A <- EdDSA.encode (K_A, $H_{SA}i$) $\forall i \in$ (0,1...(N-1))

27 T_A transmits K_B and Ei_A (0,1...N-1) to T_B

28 T_B performs

29 Ej_A

30 For j <- 0,1, 2…...N-1 do

31 a_j <- EdDSA.verify (Ei_A, $H_{SB}i$)

32 End For

33 $$J_S = \frac{1}{N}\sum_{j=0}^{N-1} a_j$$

34 If $J_S \geq J_T$

35 Then Break

36 End If

Algorithm 5.1 describes the process for fraud detection between Telco A and B using the simple min hash technique. Chapter 4 developed a framework for doing the same using one of the minhash techniques that will be evaluated in this chapter. Here, Telco A and B have their fraud indicators, which they can split into shingles. Telco A and B take their now split data, where each of the shingle values becomes a seed for a pseudorandom number generator (PRNG). The Hi_A and $Hi_B \; \forall \; i \in (0 \ldots N)$ minhash arrays store the final process values. These arrays are initialized with infinity or an extremely high number.

For each position on the MinHash array, the minimum of the existing value is compared against the PRNG. The algorithm iterates through all the shingles until a complete MinHash signature of the fraud indicator is established. The remaining protocol is as per the framework where Telco A further encrypts using the EdDSA algorithm and sends that data to T_B along with the public key for a Jaccard verification check.

The accuracy of the implementation using the MinHash technique will be based on the accuracy of the EdDSA verification function and the variance on the MinHash function that is used to hash the fraud indicator.

$$A_M = A_E \times \left(1 - \frac{J_S \times (1 - J_S)}{N}\right) \qquad (5.2)$$

Where A_M is the overall accuracy of the FD-PPRL protocol using the simple MinHash technique, A_E is the accuracy of the EdDSA verification function which for experimentation will assume $A_E \approx 1$ since the accuracy of the verification function is extremely high and is almost 1.

5.3 FD-PPRL implementation using Weighted MinHash with LSH

Telcos have a vast collection of customers and potential bad actors who, for several reasons, have been flagged as a substantial risk. While MinHash is a simple and straightforward protocol, we need a protocol that can handle large datasets. The Locality Sensitive Hashing (LSH) function can help categorize related items into closer groups, which can be combined with the MinHash protocol to speed up the estimation of the Jaccard Similarity function. Once the items are hashed to the same bucket, they can be considered for a candidate pair. These candidate pairs can then be checked for similarity. The dissimilar pairs that do not hash to the same buckets are false positives and are never checked. There will be a smaller population of false positives that are in the same bucket.

The above protocol describes the determination of the Jaccard similarity using MinHash with LSH hash functions. While this helped with sorting pairs more efficiently, in the case of searches on large datasets, consideration is being given to weighted datasets where the weight can be the frequency of an item or simply, in the case of fraud indicators, the propensity of committing fraud.

Consistent weighted sampling is an LSH Scheme [30] where the probability of hash collision is equal to the Jacquard similarity. The sample k is uniformly taken from the set of hashed values H_S, where weights y ∈ S a set of weights which are assigned as described above.

$$P\,(sample\ H_{SA} = Sample\ H_{SB}) = \frac{\Sigma\ \min(H_{SA}=H_{SB})}{\Sigma\ \max(H_{SA}=H_{SB})} \qquad (5.3)$$

The goal of Minhash is to represent data with the lowest possible bits and still be able to reconstruct the distances efficiently and accurately.

Algorithm 5.2 Logic to determine Fraud using simple Weighted MinHash function

Inputs:

Fi_A, Fi_B – T_A, and T_B's fraud indicators respectively

J_T – Jaccard Threshold

H - Hash function shared between Telco A and B

N - Size of the hash signature

MH – MinHash algorithm

SH_A, SH_B - Shingles of the Fraud Indicator Fi_A, Fi_B

Z - Amount of shingles from the fraud indicator

Outputs:

J_S - Jaccard indicator

K_A - T_A's private key

K_B - T_B's public key

HS_A, HS_B – Telcom A and B hash signatures of Fi_A and Fi_B

WMH_B, WMH_A – Weighted MinHash Samples

Match = True if $J_S > J_T$ (Jaccard threshold); else, False

Steps:

1 (a_0, a_1, a_2, ... a_{N-1}) [0,1]

2 (a_0, a_1, a_2, ... a_{N-1}) <- (0, 0, …., 0)

3 Telcom B generates minhash of its fraud indicator

4 SH_B <- Shingle-process (Fi_B)

5 $H_{SB}1$, $H_{SB}2$, ..., $H_{SB}N$ <- inf, inf, …., inf

6 SH_B

7 For i <- 0,1…Z-1 do

8 | Initialize a pseudo random generator r with a seed SH_B

9 | For j<-0,1…. N-1 d0

10 | | H_{SB}j <- min ($H_{SB}i,r$)

11 | End For

12 End for

13 Generate the Weighted MinHash LSH Samples

14 For i <- 0,1 .... N-1 do

15 | Sample:

16 | (k^*,y^*) <- $\{(k, y): \{k\ H_{SA,}\ 0 \leq y \leq S\}$ //S - Set of weights

17 End For

18 WMH_A <- (k^*,y^*)

19 K_A, K_B <- EdDSA.generateKey()

20 Ei_A <- EdDSA.encode (K_A, WMH_A i) ∀i∈ (0,1...(N-1))

21 T_A transmits K_B and Ei_A (0,1...N-1) to T_B

22 T_B performs

23 Ej_A

24 For j <- 0,1, 2…...N-1 do

25 | a_j <- EdDSA.verify (Ei_A, WMH_B i)

26 End For

27 $$J_b = \frac{1}{N}\sum_{j=0}^{N-1} a_j$$

28 If $J_b \geq J_T$

29 Then Break

30 End If

As with MinHash, the accuracy of the implementation using the MinHash technique will be based on the accuracy of the EdDSA verification function and the variance on the Weighted MinHash function that is used to hash the fraud indicator.

Since the Weighted MinHash through the sampling process is pushed to a signature size of b bits. The probability that the b-bit hash collide is 2^{-b} which will factor into the calculation of the Weighted MinHash Jacquard Similarity [40] J_b

$$J_b = P(WHM_A = WMH_b) = J + (1 - J)2^{-b} \quad (5.4)$$

Then A_M the overall accuracy of the FD-PPRL protocol can be defined as

$$A_M = A_E \times \frac{(J_b \times (1 - J_b))}{N} \quad (5.5)$$

where we can substitute $A_E \approx 1$

5.4 FD-PPRL implementation using SimHash Protocol

The Simhash algorithm is used to quickly estimate the similarity between sets. It was designed by Moses Charikar [41] and is used by Google Crawler to find near-duplicate pages. The algorithm works by generating a hash signature for each set and then comparing the signatures to determine their proximity.

5.4.1 Protocol Definition

The Fraud Indicator is sent through the SimHash process where the features are extracted. The Features could be how often a character occurs in the indicator or length of indicator. In our case the Fraud indicator is split into shingles where the process is mentioned in the background section. These shingles are then hashed using one of the standard hash functions. These individual Hash values are then combined, or their individual bit can be XORed.

Typically, once the digital signature of the Fraud indicators is determined, they can be compared using a cosine similarity function.

$$P(SMH_A = SMH_B) = 1 - \frac{\theta}{\pi}$$

Where θ is angle between the vectors SMH_A and SMH_B (the SimHash signature of Telco A, and B) and the probability that they match proportional to the cosine of the angle between the two vectors.

However, since our FD-PPRL protocol uses the EdDSA verification function that returns either a true or false, which works well with an Identity function, the similarity measure will only compare the Encrypted SimHash and unencrypted SimHash to return a True or False on the matches.

5.4.2 Parameter Definition

The main inputs to SimHash are the fraud indicator themselves with the outputs being their hashed signature for comparison.

5.3 Record Encoding

The records are Hashed using SHA1 encoding scheme,

Algorithm 5.3 Logic to determine Fraud using simple SimHash function

Inputs:

Fi_A, Fi_B – T_A and T_B's fraud indicators respectively

H - Hash function shared between A and B (SuperMinHash)

N - Size of the hash signature

T – Size of the block of signatures transmitted by T_B

SH_A, SH_B - Shingles of the Fraud Indicator Fi_A, Fi_B

Z - number of shingles from the fraud indicator

Outputs:

J_S - Jaccard indicator

K_A - T_A's private key

K_B -T_B's public key

Match = True if $J_S > J_T$ (Jaccard threshold); else, False

Steps:

1 $(a_0, a_1, a_2, \ldots\ldots a_{N-1})$ *['True', 'False']*

2 $(a_0, a_1, a_2, \ldots\ldots a_{N-1})$ *<- (null, null, ..., null)*

3 Telcom B generates SimHash of its fraud indicator

4 SH_B<-Shingle-process (Fi_B)

5 *W[f]* <- 0; //f length dimension vector

6 for i = 1 to *n* do

7 $X \leftarrow$ H(SH_B), H is a normal hash function

8 for *j*=1to f do

```
9                   if X[j] = 1
10                      then W[j] ← W[j] + wi  //wi are the weight of the Shingle SH_B
11                  Else
12                      W[j] ← W[j] − wi
13                  End If
14              End For
15      End For
16      for I=1 to f
17              if W[i] > 0 then
18                  SMH_B[i] ← 1
19              else
20                  SMH_B[i] ← 0
21              End If
22      End For
23      return SMH_B
24      K_A, K_B <- EdDSA.generateKey()
25      Ei_A <-EdDSA.encode (K_A, SMH_A),i∈ (0,1...(N-1)
26      T_A transmits K_B and Ei_A (0,1...N-1) to T_B
27      T_B performs
28      Ej_A
29      For i <- 0,1, 2…...N-1 do
```

30 | a_j <- *EdDSA.verify* (Ei_A, SMH_B)

31 End For

32 a_j *returns True or False based on the EdDSA.verify function output*

The accuracy of the implementation using the SimHash technique will be based on the accuracy of the EdDSA verification function and the variance on the SimHash function that is used to hash the fraud indicator.

$$A_M = 1 - \frac{d}{\max(d)} \quad (5.6)$$

Where A_M is the overall accuracy of the FD-PPRL protocol using the simple SimHash technique, A_E is the accuracy of the EdDSA verification function which we will set to 1. d is the hamming distance between SMH_A and SMH_B with Max(d) the maximum hamming distance when $SMH_A \ncong SMH_B$.

5.4 FD-PPRL implementation using SuperMinHash protocol

The SuperMinHash algorithm was designed by Otmar Ertl [26] to increase the speed and accuracy of the MinHash algorithm. While there are many other algorithms that precede SuperMinHash such as ProbMinHash and MinMax hash, they do not pan out when incorporated into our developed framework since the logic of the algorithm does not allow comparison of two non-homologous datasets. As shown in section 5.1, the regular MinHash function, while quite effective, can have speed and accuracy limitations as demonstrated in the experiment section of this chapter.

5.4.1 Protocol Definition

As mentioned, this hash algorithm was designed to keep speed and accuracy as the primary criteria for design. The Fisher-Yates shuffle algorithm is used in this logic. This is where the input array of characters that are being shuffled is used to determine the next element randomly until no elements remain. This algorithm is one of the most efficient shuffling algorithms present currently. The run time complexity reduces from O(nm) to O (n + m log2m) for large data sets.

5.4.2 Parameter Definition

The main inputs to SuperMinHash Algorithms are the fraud indicator themselves with the outputs being their hashed signature for compare.

Algorithm 5.4 Logic to determine Fraud using simple SuperMinHash function

---

Inputs:

Fi_A, Fi_B – T_A, and T_B's fraud indicators respectively

H - Hash function shared between A and B (SuperMinHash)

N - Size of the hash signature

T – Size of the block of signatures transmitted by T_B

M_H – MinHash algorithm

SH_A, SH_B - Shingles of the Fraud Indicator Fi_A, Fi_B

Z - numbers of shingles from the fraud indicator

Outputs:

J_S - Jaccard indicator

K_A - T_A's private key

K_B -T_B's public key

Match = True if $J_S > J_T$ (Jaccard threshold); else, False

Steps:

1 $(a_0, a_1, a_2, \ldots\ldots a_{N-1})$ [0,1]

2 $(a_0, a_1, a_2, \ldots\ldots a_{N-1}) \leftarrow (0, 0\ldots.0)$

3 Telcom B generates SuperMinHash of its fraud indicator

4 SH_B<-Shingle-process (Fi_B)

5 Hi_B<-inf $\forall$ i$\in$ (0…N)

6 AR_i array for assigning value for all (0…N)

7 QR_i<—1 for all (0…. N)

8 For i<- 0 ……N-1

9 Initialize PRNG with SH_B

10 For j<-0,1….m-1 do

11 *Ran1*<- uniform random number from [0,1)

12 *Ran2*<- uniform random number from {j……N-1}

13 If $QR_j \neq$ I then

14 QR_j<-I

15 AR_k<-k

16 End If

17 Swap AR_j and AR_k

18 Hi_A<-min (Hi_A, R+j)

19 End for

End For

Telcom A generates minhash of its fraud indicator

SH_A <-Shingle-process (Fi_A)

Hi_A<-inf for all i (0…N)

AR_i array for assigning value for all (0…N)

QR_i<—1 for all (0…. N)

For I<- 0 ……Z-1

    Initialize PRNG with SH_A

    For j<-0,1….m-1 do

        $Ran1$<- uniform random number from [0,1)

        $Ran2$<- uniform random number from {j……N-1}

        If $QR_j \neq$ I then

            QR_k<-I

            AR_k<-k

        End If

        Swap AR_j and AR

        Hi_A<-min (Hi_A, R+j)

    End For

End for

K_A, K_B <- *EdDSA.generateKey()*

Ei_A <-*EdDSA.encode* (K_A, Hi_A),i(0,1…(N-1)

T_A transmits K_B and Ei_A (0,1...N-1) to T_B (steps 21 to 41 are performed for all the block contents)

43 T_B performs

44 For j <- 0,1, 2…...M-1 do

45 Ej_A

46 For i <- 0,1, 2…...N-1 do

47 a_j <- *EdDSA.verify* (Ei_A, Hi_B)

48 End For

49 $$J_S = \frac{1}{N}\sum_{j=0}^{N-1} a_j$$

50 If $J_S \geq J_T$

51 Then break

52 End If

53 End For

54 $J_S \geq J_T$ then *Match*=True; else, False.

The accuracy of the implementation using the SuperMinHash technique will be based on the accuracy of the EdDSA verification function and the variance on the MinHash function that is used to hash the fraud indicator.

$$A_M = 1 - \frac{J(1-J)}{N} \times \left(1 - \frac{\sum_{l=1}^{N-1} l^N \left((l+1)^N + (l-1)^N - 2l^N\right)}{(N-1)^N \times N^N}\right) \quad (5.7)$$

Where A_M is the overall accuracy of the FD-PPRL protocol using the simple MinHash technique, A_E is the accuracy of the EdDSA verification function which for experimentation will

assume $A_E \approx 1$ since the accuracy of the verification function is extremely high is almost 1 and N is the signature length.

5.5 Conceptual Analysis of the Signature HASH Techniques to be used in our framework

In this section we conduct a conceptual analysis of the three hash methods for data sharing in terms of complexity, quality of blocking and privacy on the data being shared with each Telco.

5.5.1 Complexity

We analyze the computational and communication complexity of three methods described above into two main aspects of record request and response. Let nf be the size of the block indicator for which information is requested. Let n be the signature length and m be the length of the fraud indicator. The regular MinHash protocol in 5.1 will have a runtime complexity of O(nm). For the Simhash algorithm, which has a run-through process of feature extraction through the fraud indicator (m), hashing the extracted features (m), and computing the SIMHASH signature, the runtime complexity will be O(3m). For the Superminhash algorithm, since it uses the Fischer Yates Sorting algorithm and makes only one pass at the signature, it will have a runtime complexity of O (n + m log2m). In the experimental section, we will compare the performance of these various methods.

5.5.2 Accuracy

Here we will analyze the sensitivity of the three methods to detect similarities with the Fraud Indicator in our protocol's framework. While these protocols have their own accuracy

metrics on a standalone basis, their behavior will change since they have been redesigned to be incorporated into the FD-PPRL framework. Hence, we will need to reevaluate their accuracy.

5.5.3 Privacy analysis

The main privacy protection feature in FD-PPRL is the use of the EdDSA algorithm to compare two non-homologous datasets. We will analyze the impact of using one of the three hash methods to determine which is more secure. Assuming the Telcos are honest but curious entities, since they will need to agree upon or share the same hash algorithm, the experimental section will analyze any potential privacy leaks.

5.6 Experimental Evaluation and Discussion

In the experimental setup section, we will discuss the database used, the parameter selection, and the evaluation of the results. Also, the experiment results will be reviewed and discussed, explaining the logic and criteria we used to select the most appropriate Hash technique for the FD-PPRL protocol.

5.6.1 Experimental Setup

As in Chapter 4 we used an Apple MacBook Air with an Apple M1 chip and 8 GB memory. We developed a Python package for each of the hash techniques with our FD-PPRL protocol using the four algorithms mentioned above. We also used the North Carolina voter database as a substitute for the Fraud database that will be housed within Telcos. For our analysis, we established the length of the database, which represents the block of information sent by T_A, to be 1000. The Fraud Indicator FA that is being searched for was set to be ‘ALICE TEST BOB.’

Experiment 1

We conducted experiments incorporating the MinHash (MNH), WeightMinHash (WMH), SimHash (SMH), and SuperMinHash (SuMH) into the FD-PPRL protocol to measure the accuracy of each of these implementations to get a measure of what will be best suited for FD-PPRL. Table 4.1 will show the Fraud Data and the variance in characters that was processed through the protocol for measurements.

Character Variance	**FA**
0	ALICE TEST BOB
1	ALICE TEST BO*
2	ALICE TEST B**
3	ALICE TEST ***
4	ALICE TES****

Table 5.2 Character position variance to test the accuracy of implementation

For each of the protocols, 1000 records of individual block data were processed against each of the F_A's and the Jaccard similarity measure was recorded. The accuracy, as mentioned in equations 5.2, 5.5, and 5.6, was measured and recorded for various signature lengths N=5,10, 15, and 20, with the character variance from 0 to 4 as depicted in Table 5.2. The graphical representation of the results is shown in Figures 5.1, 5.2, 5.3, and 5.4.

N=5	MH	WMH	SMH	SuMH		N=10	MH	WMH	SMH	SuMH
0	1	1	1	1		0	1	1	1	1
1	1	1	0	0.8		1	1	1	0	0.8
2	1	1	0	0.6		2	0.8	0.5	0	0.67
3	1	1	0	0.6		3	0.8	0.5	0	0.53
4	1	0.9	0	0.4		4	0.8	0.5	0	0.46
N=15	**MH**	**WMH**	**SMH**	**SuMH**		**N=20**	**MH**	**WMH**	**SMH**	**SuMH**
0	1	1	1	1		0	1	1	1	1
1	1	1	0	0.8		1	1	1	0	0.8
2	0.86	0.667	0	0.67		2	0.9	0.9	0	0.6
3	0.67	0.667	0	0.53		3	0.9	0.9	0	0.53
4	0.8	0.667	0	0.46		4	0.9	0.9	0	0.46

Table 5.3 Accuracy comparison of the four protocols across four signature lengths

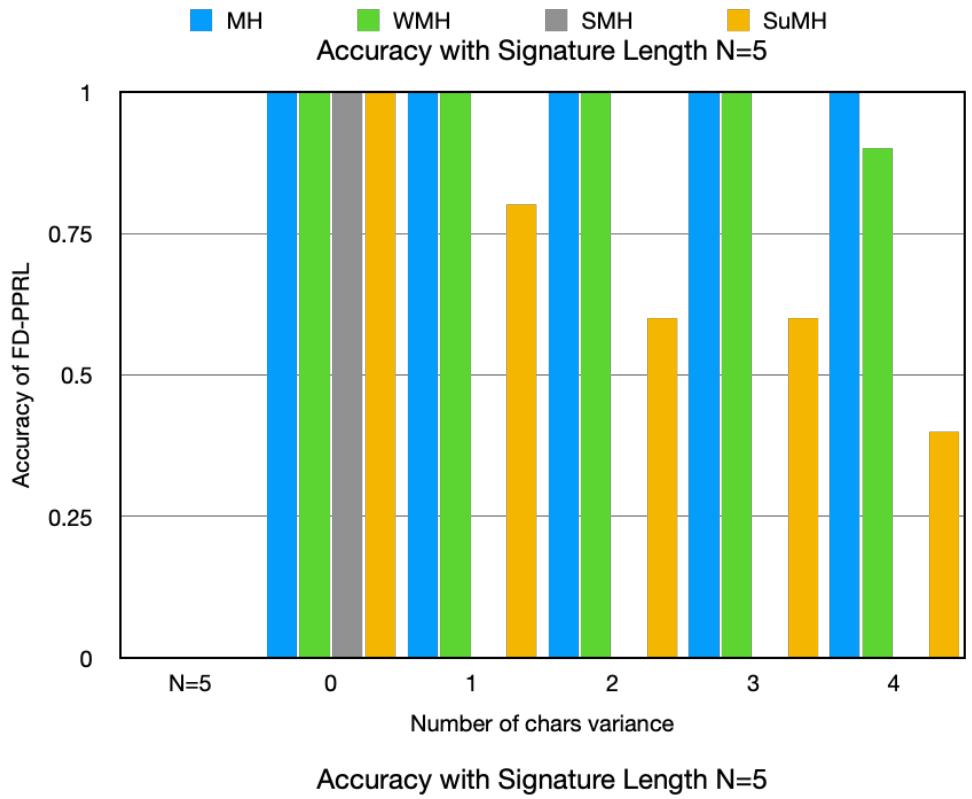

Figure 5.1 Accuracy of the four methods with Signature Length N=5

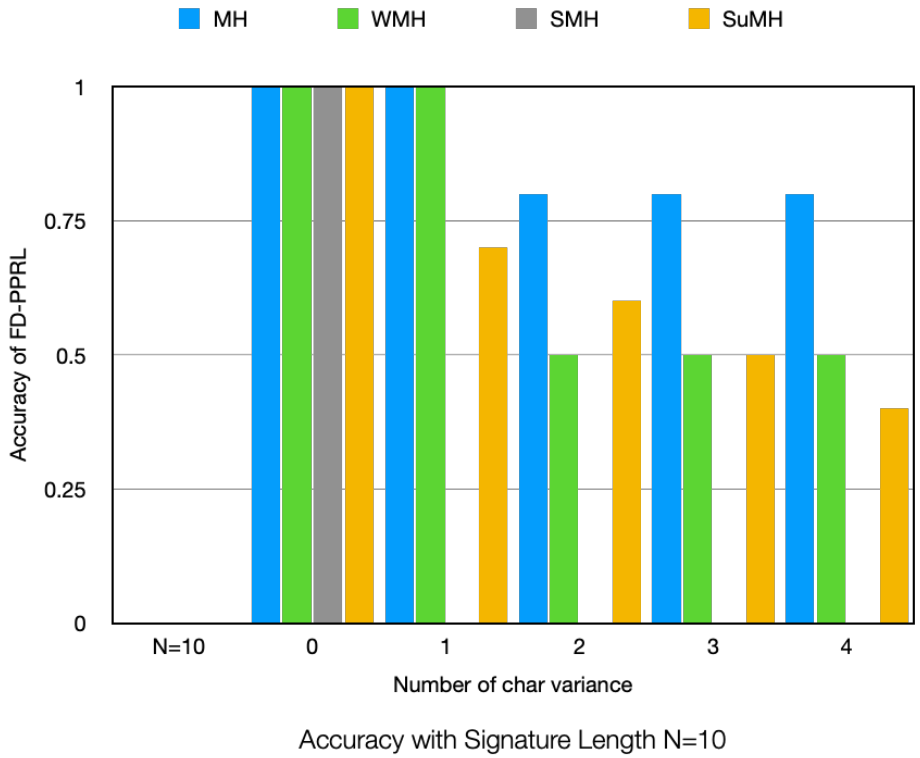

Figure 5.2 Accuracy of the four methods with Signature Length N=10

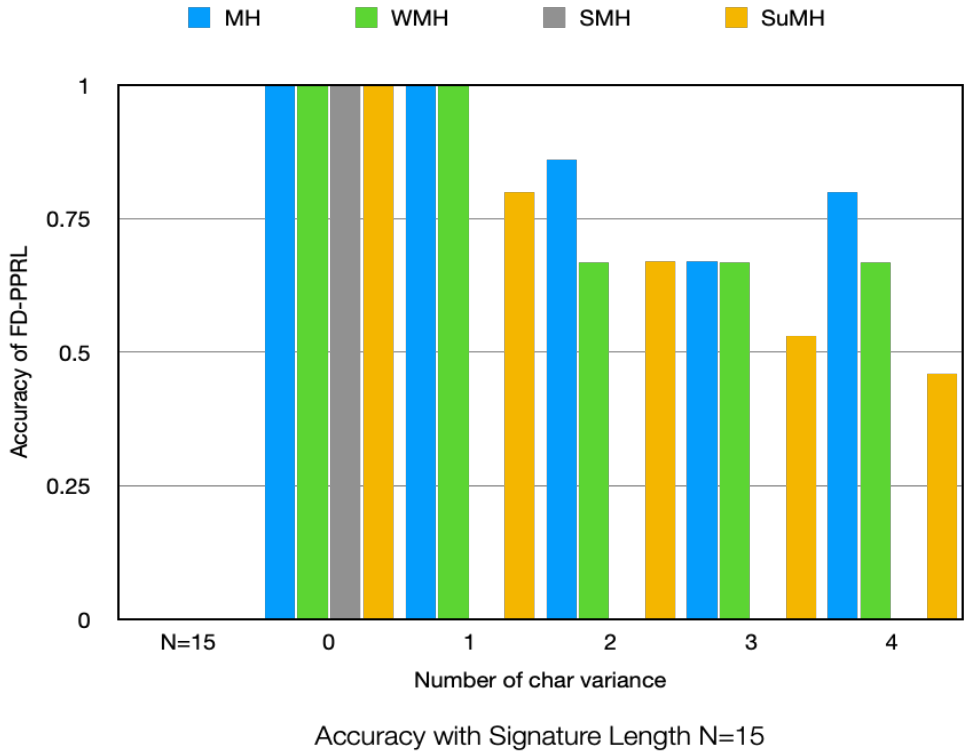

Fig 5.3 Accuracy of the four methods with Signature Length N=15

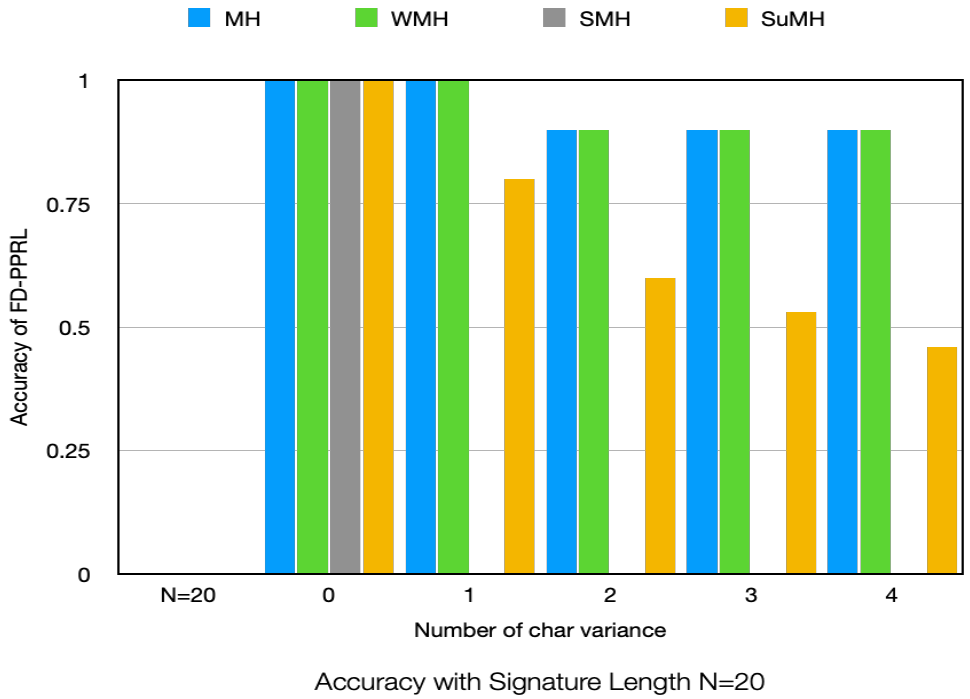

Figure 5.4 Accuracy of the four methods with Signature Length N=20

The data presented in the following figures and tables clearly demonstrate the sensitivity of SIMHASH implementation to minor character changes. For precise readings on a Fraud Indicator, SMH proves to be beneficial. MH, WMH, and SuMH exhibit similar behaviors, but SuMH outperforms in accuracy for larger variations as the Jaccard similarity begins to decrease rapidly with an increase in character variations. MH and WMH fail to show significant shifts when the differences in characters increase, suggesting their limitations for precise readings. SuMH, on the other hand, can still identify matches for minor character variations, such as spelling errors. This key advantage is a primary reason for our decision to incorporate SuMH into our implementation.

Experiment 2

In this experiment, we evaluated the implementation speed of four protocols - MinHash (MNH), WeightMinHash (WMH), SimHash (SMH), and SuperMinHash (SuMH) - when integrated with FD-PPRL. We used a consistent block dataset comprising a thousand names across two distinct scenarios. In the first scenario, the fraud block F_B<-"ALICE TEST BOB" is included in the 1000-block fraud data, and we measured the performance in seconds. In the second scenario, F_B is absent from the dataset, necessitating a full dataset scan by the protocol. Performance measurements were conducted across five different signature lengths, namely N=5,15,15,20, and 25.

With "ALICE TEST BOB in data set				
N	**MH**	**WMH**	**SMH**	**SuMH**
5	0.4453	0.0316	0.0076	0.0087
10	0.7389	0.0467	0.00637	0.0171
15	1.0617	0.0681	0.00567	0.0245
20	1.3921	0.0910	0.00647	0.0318
25	1.701	0.1116	0.0067	0.0317
With "ALICE TEST BOB not in data set				
N	**MH**	**WMH**	**SMH**	**SuMH**
5	1.2456	1.2165	0.2456	0.8775
10	2.4435	2.3843	0.2384	1.7489
15	3.5428	3.6135	0.2384	2.6344
20	4.6749	4.7153	0.2384	3.5193
25	5.7950	5.9020	0.2384	4.3765

Table 5.4 Runtime analysis of the four protocols with and without matches

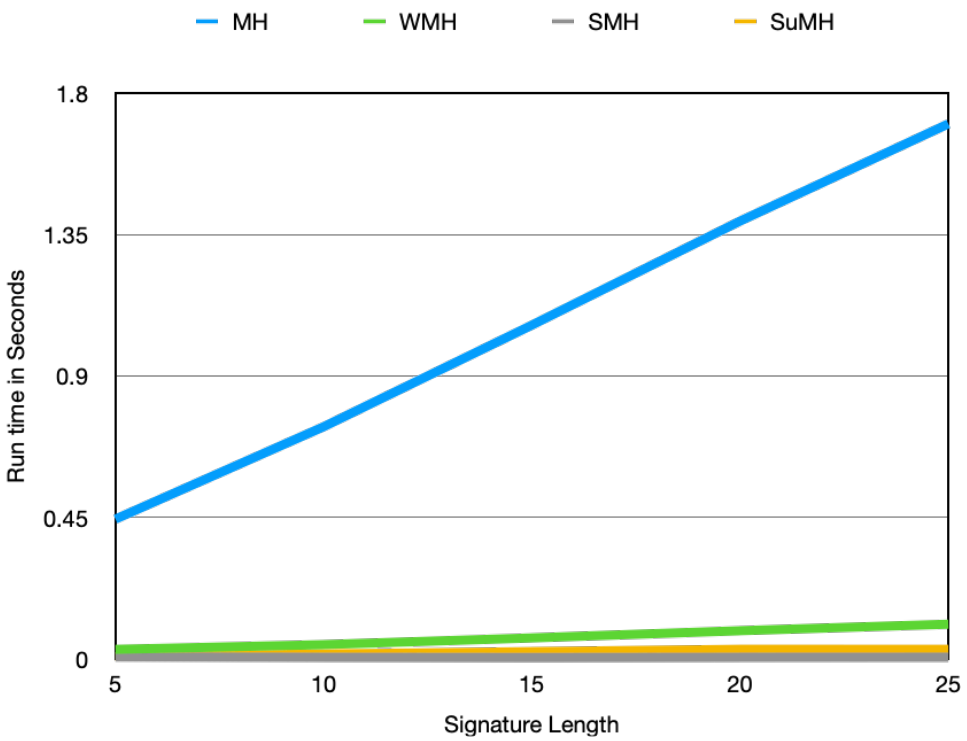

Figure 5 5 Run time analysis with matches in the Fraud data

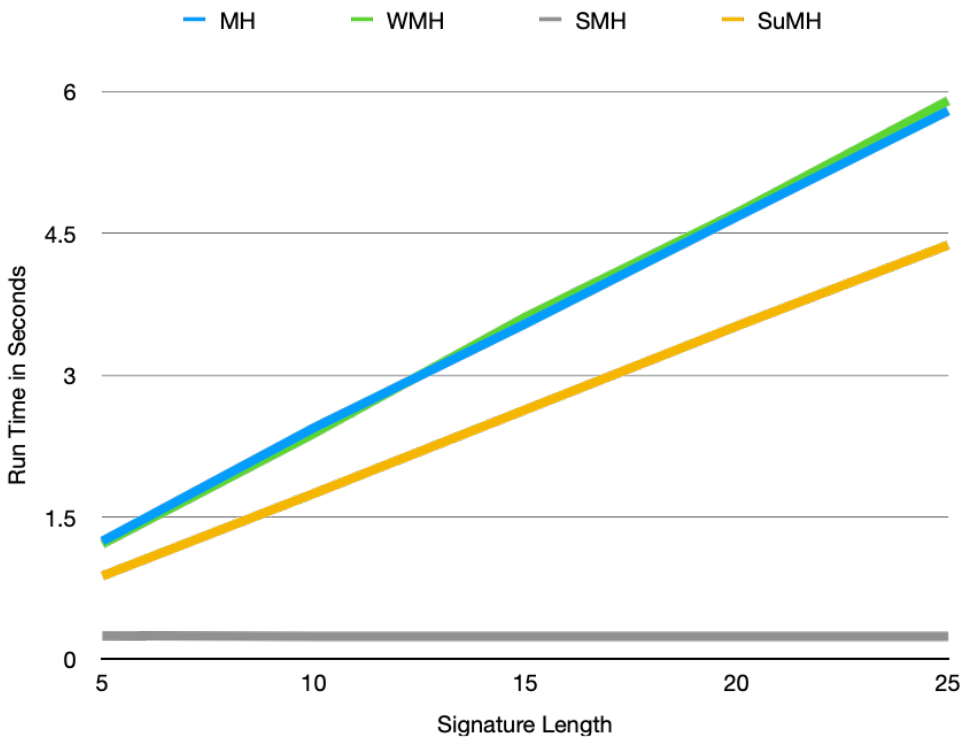

Figure 5.6 Run time analysis without any matches in the Fraud data

Upon examining the data and charts, it is evident that SimHash exhibits the best runtime performance among the four protocols, given the deterministic nature of its FD-PPRL output. MinHash FD-PPRL, on the other hand, performs the least efficiently, making it unsuitable for industrial implementation, especially over large datasets. The choice between Weighted MinHash and SuperMinHash becomes clear as SuMH significantly outperforms in both scenarios - with and without matches.

5.7 Chapter summary

In this chapter, we examined four distinct hashing techniques. We began with the most basic, MinHash, and progressed to WeightMinHash, which incorporates locality-sensitive hashing. We then delved into SimHash, a protocol with a completely different hashing principle that is commonly used in search engines. Finally, we evaluated SuperMinHash, which has proven to be the most efficient in terms of performance and accuracy, as demonstrated in our conceptual analysis and experimental section. Consequently, we have opted to use SuperMinHash in the FD-PPRL protocol for Telco fraud detection and sharing. In the next chapter, we will examine various blocking techniques for Telcos to organize and transmit data in their Fraud Database when a request is made.

Chapter 6: Blocking Framework for Data Organization and Sharing in FD-PPRL

In this chapter, we present a protocol to be used by telecommunication companies that have large databases with fraud information and need to share that information with each other. The protocol should be scalable and can avoid full data scans but instead get to the relevant sections of the data store to extract data blocks for comparison. As with the previous implementations, the protocol should be easy to implement, stable, and secure. In Section 6.1, we describe the overview of our approach. Section 6.2 describes the Shared Parameter Blocking method. In Section 6.3, we will review the Dynamic Blocking method. Section 6.4 describes the Tuple-based Blocking Method. In Section 6.5, we will go over the conceptual analysis of the three blocking methods described in this chapter. In Section 6.6, we will conduct some experimental analysis on the blocking techniques. Finally, we will summarize the chapter and introduce it to the next chapter.

Introduction

Searching for large numbers of records has always been a challenge that needs to be addressed by any application that links records. When multiple databases are involved, the record comparison increases exponentially, making the process of PPRL comparison extremely challenging. To address this challenge, we can use the techniques of blocking or indexing, which have been used for many years. These techniques reduce checks or comparisons by removing true negatives or non-matches of the record pairs. This results in a reduction in the number of computational and communication steps in a protocol while checking for fraud. We will evaluate

many blocking techniques and focus on the one that will be best suited for fraud detection using PPRL.

Table 6.1 - Notation and Terminology used in this Chapter

T_B	Telco B requesting the Fraud check
T_A	Telco A processing the Block information for transmission
D_A	Fraud dataset of Telecom A
D_B	Fraud dataset of Telecom B
FD-PPRL	Fraud Detection Private preserving record linkage
EdDSA	Edwards Curve Digital Signature Algorithm
P_K	Parameter for determining blocking
SH	SuperMinHash Algorithm
Fi_A, Fi_B	T_A, and T_B's fraud indicators respectively
N	Size of the hash signature
T	Size of the block of signatures transmitted by T_A
PK_A	Block Parameter of Fi_A
KB_A ,KB_B	T_B's public key and Private key
A_M	Accuracy of the block protocol
A_E	Accuracy of the EdDSA protocol

Table 6.1 Notation and Terminology in Chapter 6

In this chapter we will evaluate the following blocking techniques:

1.) Shared parameter block method –

In this method, Telcos organizes blocks based on agreed-upon block methods and parameters used to create and distribute their fraud data into the blocks. These blocks are then sent based on the received request.

2.) Dynamic Parameter block method -

In this method, the parameters or part of the fraud indicator are sent to Telcos. These parameters are then used to create blocks dynamically based on a preset Jacquard threshold. This dynamically created block is then transmitted.

3.) Tuple based Parameter blocking -

Telcos can choose to organize their blocks of fraud information however they want. A tuple of parameters {para1, para2, para3…} is sent to the telco. The blocks of fraud information are pre-clustered based on a hierarchy of parameters, and the search goes down the hierarchy. When a match is found on all the information in the tuple, the selected block of fraud indicators is transmitted.

6.1 Overview of our approach

One of the major problems with many implementations is the speed of search and potential transmission of large volumes of data. One easy way to increase speed is to reduce the size of the search and the quantity of transmission. However, it is quite possible that blocks with smaller sizes could be prone to cryptanalysis attacks. In our framework, the implementation has been encrypted using EdDSA, which is one of the most secure protocols.

The goal is to reduce the search space and transmission quantity to ensure a faster response time. This can be represented as D_A -> DBi_A, D_A->DBj_A, … , D_A->DBk_A, where D_A is the original dataset. It gets split or addressed based on the following criteria:

1. **Prearranged Method of Splitting, Requesting, and Sharing Data**: The data is split according to a predetermined method, and the process of requesting and sharing data follows this method.
2. **Individual Telco Desire**: The data blocks are split based on the individual preferences of the Telco. The request is independent of the splitting process.

In this protocol phase, the transaction begins with a request for a block by the requesting Telco T_B, which is referred to as R_A. The following sections describe in detail how R_B is processed by T_A and how a response is generated back to T_B.

6.2 Shared parameter block method

In this section we describe the protocol in detail

6.2.1 Parameter determination

Telcos T_A and T_B, along with T_N and N Telcos, collaborate to determine parameters $\{PK|K \in [a \ldots j]\}$ to organize the dataset and use it for transmission. The parameters themselves are not encoded when determined and used to segregate the data; they only get encoded when the request is made for a block. Since the block parameter determination is being done within the industry, it would be easy for the group to come to an agreement on the basis on which the parameters are determined. Block size will also be a factor in determining the type of parameter to be used. For instance, if Fi_A represents an individual's name, the block can potentially be segregated by state, city, or even zip code.

6.2.2 Record encoding

The block parameter from Telco B is encoded, but not with Telco A. This ensures that Telco A receives as little information as possible from Telco B's inquiry. Telco A's fraud indicators are encoded and transmitted using FD-PPRL.

6.2.3 Block Tree Construction

The maximum size of the blocks for storage will be determined primarily by the transmission size T. The speed of response will typically be set based on the network topology and technology used between the Telcos to communicate with each other. Let's assume that the network speed will be based on the choice of an average mid-size company that uses a VPN or Ethernet network. The transmission size for a fraud enquiry will play a factor in determining the response time, which will be determined by Telcos at the time of setup. Once the block size T has been determined, it will need to be segregated and set based on the agreed-upon parameters $\{PK|K \in [a \ldots j]\}$.

Algorithm 6.1: Shared Parameter Block generation and setting

Input:

D_A - Database belonging to A

T - Transmission size/maximum block size

P_K - Parameter for determining blocking

Output:

$\{B_K|K \in [a..j]\}$

Start:

B_K<- {}

For each F_Ai belong to $[a,...,j]$ where j is the total number of records

PK_A< *extract-parameter* ($F_A i$)

i<-0

KA_A, KA_B <- *EdDSA.generateKey() p*erformed by T_A

While $F_A i$ is not null

    While $F_A i$ is not null

        Hi_A <-*SH*($F_A i$) //SuperMinHash Algorithm

        Ei_A <-*EdDSA.encode* (KA_A, Hi_A),*i(0,1...(n-1*) //Encode using EdDSA Algorithm

        $BK_A i$ <- Ei_A, PK_A

        i=i+1

    End

End

Return BK_A

Algorithm 6.1 explains the process of block generation using the parameter defined through the Telco Collaborative process. The blocking parameter used for segregation is extracted or determined from the Fraud Indicator. For example, if the Fraud Indicator is a name, the block parameter can be city, state, or both. Once the block parameter is assigned to the Fraud Indicator, the process of block assignment starts. The block max size is evaluated continuously with respect to the number of items in the block.

Once the block size limit is reached, a new block with the same block parameter and a next generation is started. The process continues until all the Fraud indicators are assigned to a block and with the block to a specific block parameter generation. Now that the block age is set, the block parameter assigned to each block is MinHashed using the agreed-upon hash algorithm. For

our purpose, we have chosen the SuperMinHash algorithm. The blocks themselves are preprocessed, i.e., SuperMinHashed. They are then processed through the EdDSA algorithm with keys and digital signatures generated, ready for transmission once a request has been made.

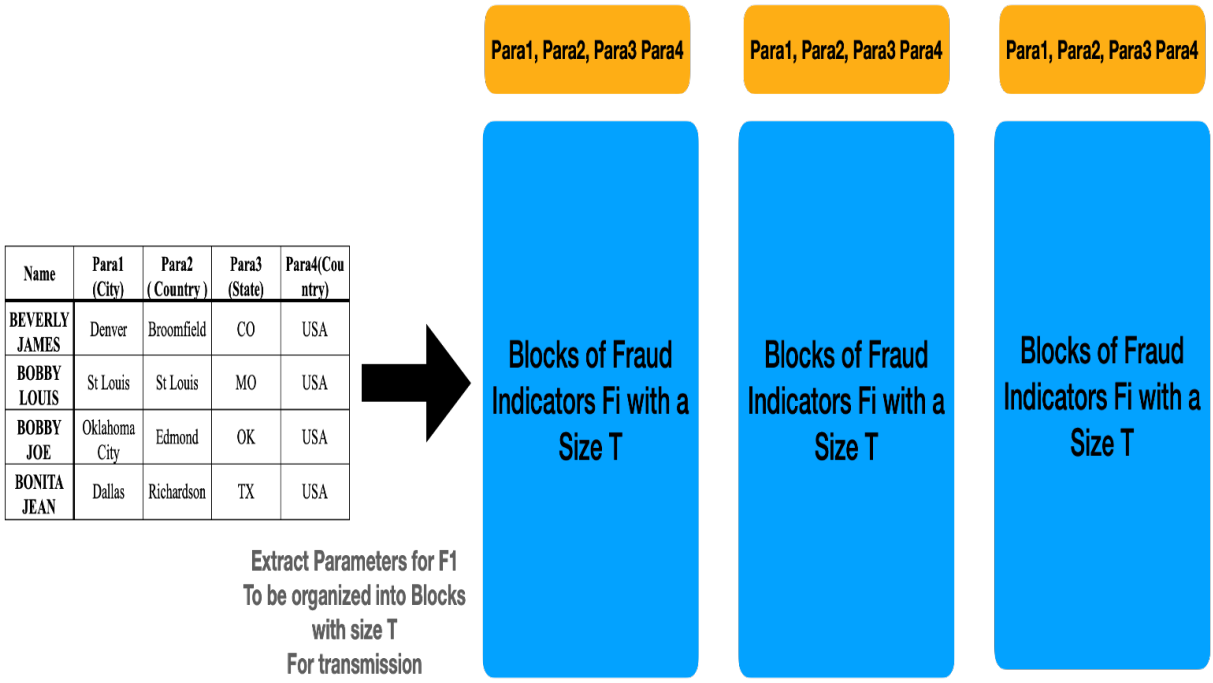

Figure 6.1 Shared parameter Block generation and setting

Algorithm 6.2: Shared Parameter Block Method Fraud Information sharing protocol

Inputs:

Fi_A, Fi_B – T_A, and T_B's fraud indicators respectively

N - Size of the hash signature

T – Size of the block of signatures transmitted by T_A

SH – SuperMinHash algorithm

PK_A- Block Parameter of Fi_A

Outputs:

J_S - Jaccard indicator

KB_A - T_B's public key

KB_B - T_B's Private key

Match = True if $J_S > J_T$ (Jaccard threshold); else, False

Steps:

1 $(a_0, a_1, a_2, \ldots\ldots a_{N-1})$ [0,1]

2 $(a_0, a_1, a_2, \ldots\ldots a_{N-1})$ <- (0, 0,….0)

3 PK_B<-*extract_parameter*(Fi_B) //function to extract parameter from Fraud Indicator

4 $(H1PK_B, H2PK_B, H3PK_B \ldots. HNPK_B)$ <-*SH*(PK_B) generate SuperMinHash for T_B fraud indicator parameter

5 KB_A, KB_B <- *EdDSA.generateKey()* performed by T_B

6 KEi_B <-*EdDSA.encode* (KB_B, $HiPK_B$) ∀ $i \in (0,1\ldots(N-1)$

7 T_B transmits KB_A and KEi_B *(0,1...N-1) to* T_A

8 T_A Perform Block Parameter matching using information from Step above

9 T_A performs

10 For j <- 0,1, 2…...N-1 do

11 KEi_B

12 For i <- 0,1, 2…...N-1 do

13 a_j <- *EdDSA.verify* (KEi_B, PK_A)

14 End For

15
$$J_S = \frac{1}{N}\sum_{j=0}^{N-1} a_j$$

16 If $J_S = 1$

17 Then break

End If

End For

T_A transmits BK_A and KA_B to $\mathbf{T_B}$ (steps 1 to 19 are performed for all the block contents).

T_B performs

For j <- 0,1, 2…...N-1 do

BK_A

For i <- 0,1, 2…...N-1 do

a_j <- *EdDSA.verify* (Ei_A, Hi_B)

End For

$$J_S = \frac{1}{N}\sum_{j=0}^{N-1} a_j$$

If $J_S \geq J_T$

Then break

End If

End For

$J_S \geq J_T$ then Match=True; else, False

Once the Telco has set up their data block based on Algorithm 6.1, Algorithm 6.2 goes into the step-by-step process of fraud information sought by Telecom B. TB extracts or determines the Block Category from Fi_B, which, as stated in 6.1, is predetermined ahead of time by the Telcos. Once the Block category is extracted, it is then hashed using their agreed-upon hash algorithm. This is then encoded using the EdDSA algorithm described in Chapter 4. The digital signature KEi_B and the Key KA_B are then transmitted to Telco A for block identification.

T_A then compares the encoded Block category using Algorithm 6.2, to identify the block to be transmitted. Unlike a fraud matching protocol where we have a Fraud Threshold to determine the likelihood of a match, since the categories are predetermined, we suggest that the Jaccard index match is one. Once the Block is identified, it is sent to T_B for Fraud identification. The remaining steps for Fraud identification are described in Chapter 4 or Algorithm 6.2.

The accuracy of the implementation of this blocking technique used for fraud detection using the SuperMinHash technique will be based on of course the accuracy of the EdDSA implementation A_E which we will approximate to 1. Variance factor γ [26] is given based on the equation below.

$$\gamma = 1 - \frac{\sum_{l=1}^{N} l^N \times \left((l+1)^N + (l-1)^N - 2l^N\right)}{(N-1)^N \times N^N} \quad 6.1$$

The accuracy of the overall protocol can then be defined as

$$A_M = A_E \times \left(1 - \frac{J_B\,(1-J_B)}{N} \times \gamma\right) \times \left(1 - \frac{J\,(1-J)}{N} \times \gamma\right) \quad 6.11$$

Where J_B is the jacquard similarity of block parameter that T_B sent to T_A for matching and J is the Jacquard similarity of Fraud Indicator Fi_B matching against the transmitted block by B.

The performance of Bidirectional block protocol is based on the Transmission Size T. Let b_k be the speed given by Mega Bytes/sec where N=1. Let S be the speed of the FD-PPRL match protocol The performance of time measure (T_p) of the protocol would be

$$T_p = N \times b_k \times (T + N_S) + 2S \quad 6.12$$

where ($T+N_S$) is the block size plus shared parameter size.

6.3 Dynamic Parameter blocking

In this section we describe the protocol in detail

6.3.1 Parameter determination

Unlike the above method where Telcos work in unison, in this method, Telcos do not have an agreed-upon method to block. In fact, in cases of initial setup where the number of Fraud Indicators is low, Telcos may not even choose to set up their datasets. Here, Telco A extracts a parameter from the Fraud Indicator Fi_B, which will be a part of the same, such as the first four letters of the name, enough to get a hit on the Jaccard Similarity matching and not enough information for privacy to be compromised.

6.3.2 Record encoding

The block parameter from Telco B is encoded, but not with Telco A. This ensures that Telco A receives as little information as possible from Telco B's inquiry. Telco A's fraud indicators are encoded and transmitted using FD-PPRL.

6.3.3 Block Tree Construction

As stated above, this method does not involve any block constructions. However, Telcos can choose to organize their datasets into blocks. Regardless of how the data is organized, the premise of this method is to scan all the elements in the database.

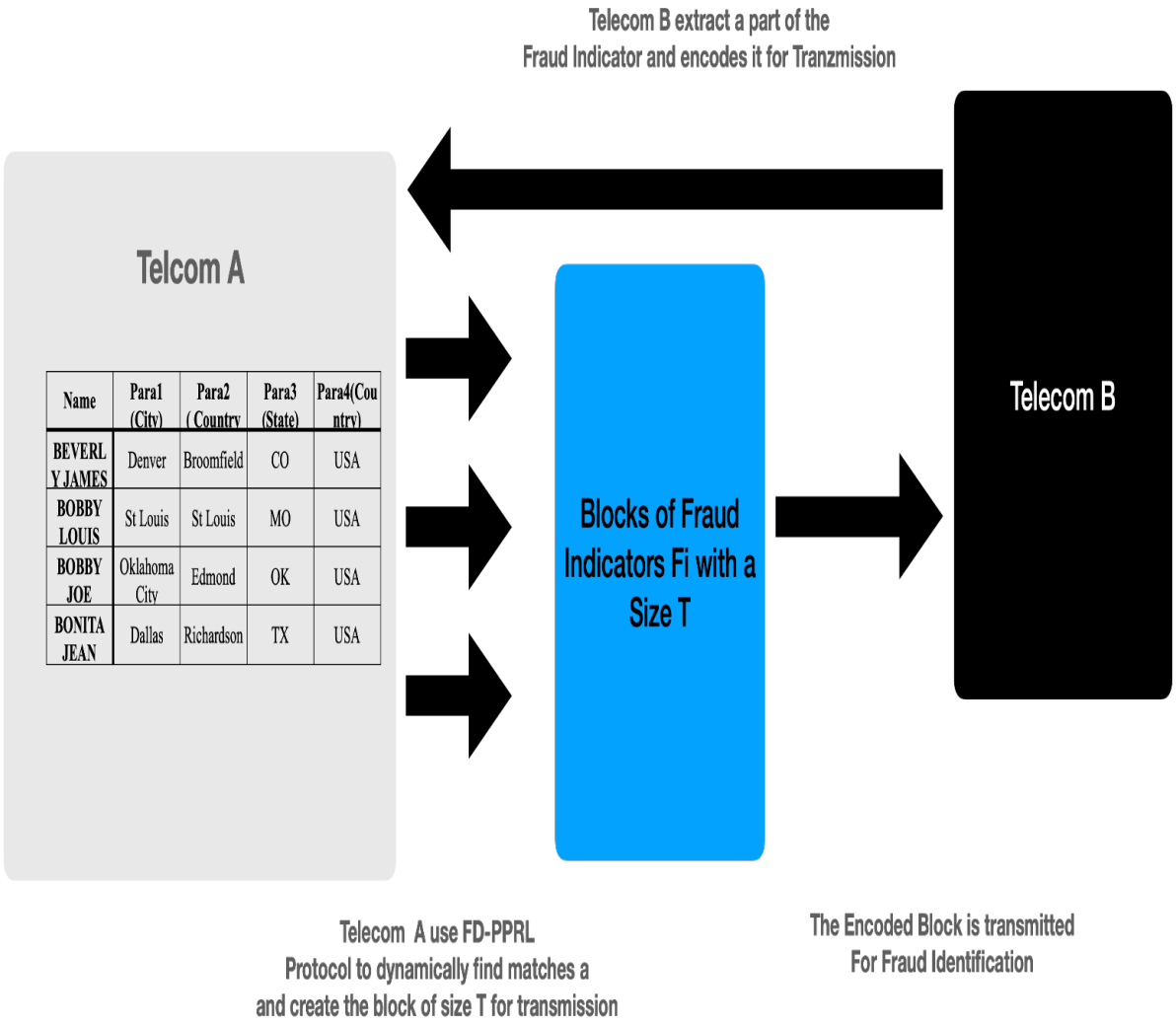

Figure 6.2 Dynamic block generation protocol

Algorithm 6.3 - Dynamic block generation protocol

Inputs:

Fi_A, $Fi_B - T_A$, and T_B's fraud indicators respectively

H - Hash function shared between A and B (SuperMinHash)

N - Size of the hash signature

T – Size of the block of signatures transmitted by T_B

SH – SuperMinHash algorithm

BK_B - Block Parameter of Fi_B

J_T - Jacquard matching threshold

M - is the size of all records in the fraud dataset

C - count of element in the Block being created

Inc - incremental factor for J_T

Outputs:

J_S - Jacquard indicator

KA_A, KA_B - T_A's private and public key

KB_A, KB_B - T_B's public and private key

Match = True if $J_S > J_T$ (Jacquard threshold); else, False

Steps:

1 $(a_0, a_1, a_2, \ldots\ldots a_{N-1})$ [0,1]

2 $(a_0, a_1, a_2, \ldots\ldots a_{N-1})$ <- (0, 0….0)

3 *C*<-0

4 PK_B <-*extract_parameter*(Fi_B)

5 $(H1PK_B, H2PK_B, H3PK_B \ldots. HNPK_B)$ <-*SH*(PK_B) // generate SuperMinHash for T_B fraud indicator

6 KB_A, KB_B <- *EdDSA.generateKey()* // EdDSA function performed by T_B

7 KEi_B <-*EdDSA.encode* (KB_B, $HiPK_B$) *i* (0,1...(*N*-1)

8 T_B transmits KB_A and KEi_B (0,1...*N*-1) to T_A

9 T_A Performs full database scan

10 $(H1_A, H2B_A, H3_A \ldots. HM_A)$ <-*SH*(Fi_A) // generate SuperMinHash for T_B of its *M* fraud indicators

11 T_A performs

12 C<-0

13 For j <- 0,1, 2…...M-1 do

14 Hi_A

15 For i <- 0,1, 2…...N-1 do

16 a_j <- *EdDSA.verify* (KEi_B, Hi_A)

17 End For.

18 $$J_S = \frac{1}{N}\sum_{j=0}^{N-1} a_j$$

19 If J_S >=J_T

20 BK_A, J_S<- Hi_A

21 *C=C+1*

22 END IF

23 End For

24 If count(BK_A) > T then

25 J_T=J_T + *in*

26 C<-0

27 For i<- 1….C-1 do

28 If J_S >=J_T

29 Then BK_A, J_S<- BK_A

30 END IF

31 End For

32 End If

33 Ei_A <-*EdDSA.encode* (KA_A, BK_A) *i (0,1...(N-1)*

34 T_A transmits K_B and Ei_A to T_B

35 T_B performs

36 For j <- 0,1, 2…...M-1 do

37 Ej_A

38 For i <- 0,1, 2…...N-1 do

39 a_j <- *EdDSA.verify* (Ei_A, Hi_B)

40 End For

41 $$J_S = \frac{1}{N}\sum_{j=0}^{N-1} a_j$$

42 If $J_S \geq J_T$ *then*

43 Then break

44 End If

45 End For

46 $J_S \geq J_T$ then *Match*=True; else, False.

As described in Algorithm 6.3, the parameter is extracted from the Fraud Indicator and typically represents a small portion of it, which is enough to trigger a response from the Jacquard similarity check without revealing the identity of the Fraud Indicator. Fi_B is min-hashed and encrypted using the EdDSA algorithm, and the key and digital signature are sent over to T_A for block request.

T_A uses the BK indicator and scans it across all the records in its dataset where the Jaccard index is greater than or equal to the threshold. Here, the comparison of the block parameter is made directly on the fraud indicator to determine if there is a hit on Jaccard similarity greater than or

equal to the Jaccard Threshold. Once the condition is met, the Fraud Indicator along with its Jaccard indicator is assigned to a block that is dynamically created.

Algorithm 6.3 shows how the blocks are generated dynamically in real-time. As the comparison is being made for fraud indicators, the transmission size T is used to determine the block size. If the block size is greater than the transmission size, the Jaccard Threshold is incremented by a fixed quantity, and the data is reevaluated to reassign the fraud indicators to the block. This process is repeated until the block size is the same as or less than the transmission size.

Once the block is dynamically created, it is then transmitted to T_B. T_B uses this block to determine the presence of the fraud indicator using the algorithm or process described in Chapter 4, which is also shown in Algorithm 6.3.

The accuracy of the implementation of this blocking technique used for fraud detection using the SuperMinHash technique will be based on the probability of finding the match on the parameters over a Jaccard threshold, which can range from 0 to 1. Once the match is found, the probability will be the same as the SuperMinHash implementation discussed in the previous section.

$$A_M = 1 - \int_0^1 (1 - J_T) dJ_T \left(J \times \frac{(1-J)}{N} \right) \quad 6.13$$

$$A_M = \frac{1}{2} \times \left(1 - \frac{J(1-J)}{N} \right) \quad 6.14$$

The performance of Dynamic block creation protocol is dependent on the Transmission Size T. Let b_k be the speed given by Mega Bytes/sec where N=1. Let S, be the speed of the FD-PPRL match protocol The performance of time measure of the protocol would be

$$T_P = b_k \times T \times N + S \times (M + 2N) \quad 6.15$$

Where M is the total number of database records of Fraud Indicators for A and N is the signature size.

6.4 Tuple based Parameter blocking

6.4.1 Parameter determination

In this method, the telcos generally agree on a framework of parameters. They do not necessarily need to agree on organizing the block in any method, nor do they need to agree on using all the block parameters, except for a few.

6.4.2 Record encoding

The block parameter from Telco B is encoded, but not with Telco A. This ensures that Telco A receives as little information as possible from Telco B's inquiry. Telco A's fraud indicators are encoded and transmitted using FD-PPRL.

6.4.3 Block Tree Construction

Just like the Shared Parameter Block Method, the primary factor that determines the maximum size of the blocks for storage is the transmission size T. Telcos can choose the block parameter that best suits their dataset and organize them in a set hierarchy that enables quick navigation to a block.

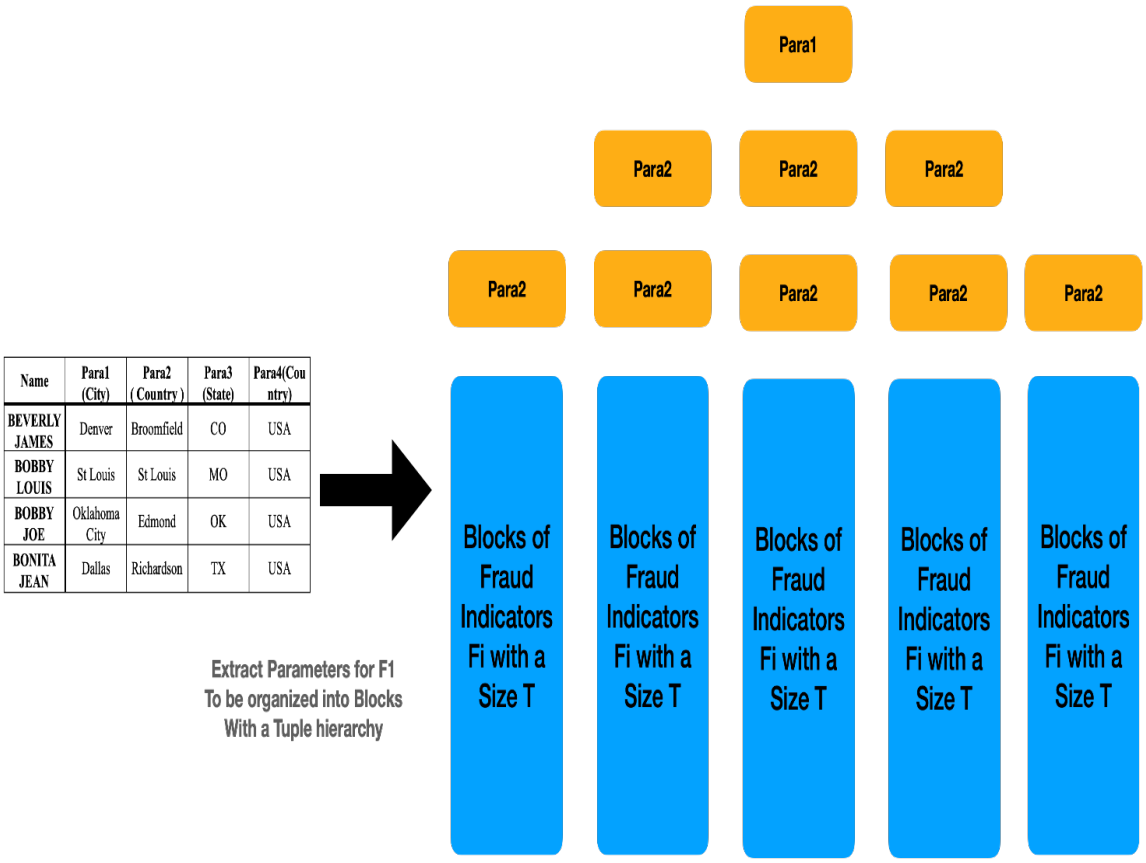

Figure 6.3 Tuple Block Parameter generation

Algorithm 6.4: Tuple Block Parameter generation and setting

Input:

D_A - Database belonging to A

T - Transmission size/maximum block size

$\underline{PK_A}$- Parameter for Fraud Indicator of A

BHi(J level Block hierarchy *BH1->BH2->BH3->...., BHJ*

Output:

$\{BK|K \in [0...J]\}$

KA_A, KA_B - T_A's private and public key

Steps:

1 BK<- {}

```
2   For each FAi, i ∈ [1….M] where M is the total number of records
3       PKA <-extract_parameter(FiA)
4       (H1A, H2A, H3A....HNA) <- SH(FiA) generate SuperMinHash for TA's fraud indicator.
        (H1PKA, H2PKA, H3PKA....HNPKA) <- SH(PKA) generate SuperMinHash for
5       TA's fraud indicator.
6       i<-0
7       While FiA is not null
8           While m<=T
9               EiA <-EdDSA.encode (KAA, HiA)∀ i∈ (0,1...(N-1)
                BKA <- EiA ,HiPKA, // assign the Fraud Indicator and parameter to
10              the block
11              m=m+1
12              i+1
13          End
14      End
15  End For
16  Return BKA
```

Algorithm 6.5 - Tuple Block Parameter Protocol

Inputs:

Fi_A, Fi_B – T_A, and T_B's fraud indicators respectively

H - Hash function shared between A and B (SuperMinHash)

N - Size of the hash signature

T – Size of the block of signatures transmitted by T_B

SH – SuperMinHash algorithm

$\underline{PK_B}$ - Block Parameter of Fi_B

J_T - Jacquard matching threshold

Outputs:

J_S - Jacquard indicator

KA_A, KA_B - T_A's private and public key

KB_A, KB_B - T_B's public and private key

Match = True if $J_S > J_T$ (Jacquard threshold); else, False

Steps:

1 PK_B<-extract_parameter(Fi_B)

2 ($H1PK_B$, $H2PK_B$, $H3PK_B$ .... $HNPK_B$) <-SH(PK_B) //generate SuperMinHash for T_B's fraud parameter

3 KB_A, KB_B <- EdDSA.generateKey() Performed by T_B

4 KEi_B <-EdDSA.encode (KB_B, Hi_B),i(0,1...(N-1)

5 T_B transmits KB_A and KEi_B (0,1...M-1) to T_A

6 T_A Performs tuple-based hierarchy scan

7 For j <- 0,1, 2…...N-1 do

8 KEi_B

9 For i <- 0,1, 2…...N-1 do

10 a_j <- EdDSA.verify (KEi_B, $HiBHK_A$)

11 End For

12 $$J_S = \frac{1}{N}\sum_{j=0}^{N-1} a_j$$

13 If $J_S \neq 1$

14 Then break

15 Else

16 If $i=N$

17 BK // Block BK is selected under the hierarchy tree for transmission

18 End If

19 End For

20 T_A transmits KA_B and $BK(0,1... T-1)$ to T_B

21 T_B performs

22 For j <- 0,1, 2…...T-1 do

23 BK

24 For i <- 0,1, 2…...N-1 do

25 a_j <- EdDSA.verify (BK, Hi_B)

26 End For

27 $$J_S = \frac{1}{N}\sum_{j=0}^{N-1} a_j$$

28 If $J_S \geq J_T$

29 Then break

30 End If

31 End For

32 $J_S \geq J_T$ then Match=True; else, False

Algorithm 6.4 describes the process for organizing the Fraud dataset. The Telcos extract the Block Features from Fraud Indicators based on the agreed-upon parameters. They also choose the hierarchy, as shown in Figure 6.3, to organize the dataset. It is important to note that the Telcos do not need to agree on how to organize their data. The dataset is processed, and the Fraud Indicators are assigned to the block as they step through the hierarchy until the lowest level is reached. If the Fraud Indicators do not have additional features, it is possible to create the block at higher levels of the hierarchy.

Algorithm 6.5 describes the process of requesting the block for fraud identification by TB. For the fraud indicator Fi_B, T_B generates a tuple of block features (BH_B1, BH_B2…BH_BN) to be sent to Telco A. Once Telco A receives the tuple of block features, it will take each tuple through an interactive process through its hierarchy as it navigates through the branches of the hierarchy checking to see which block feature gets a Jaccard similarity hit of 1 or less than 1. This process continues until there are no more branches to navigate. Once the last branch is reached and the block feature of T_A of the Jaccard similarity is still <1, then there will be no fraud indicator block to be sent to T_B. However, if the Jaccard similarity is 1, then if there is a block in that hierarchy branch, that block can be sent over to T_B for fraud identification.

The accuracy of tuple-based search will be the same as that of the shared parameter block method since they use the base SuperMinHash protocols. The same applies to their performance. Thus, their performance and accuracy equations will be the same as those in 6.11 and 6.12.

6.5 Conceptual Analysis the blocking methods

In this section, we conduct a conceptual analysis of the three methods for creating data blocks for sharing. We evaluate these methods in terms of their complexity and the level of privacy they maintain for the data being shared with each telecommunications company (Telco).

6.5.1 Complexity

We analyze the computational and communication complexity of the three methods described above, focusing on two main aspects: Block creation and Block request. Let's denote nd as the total number of records in the database for the Dynamic Block creation method, and nh as the size of the MinHashed block parameter sent by TA. O(nh) will represent the computational complexity of calculating the Jaccard similarity on one Fraud record of TB. Therefore, O(nh * nb) will be the computational complexity of generating the entire block to be transmitted to TB, provided the block size is less than or equal to T. However, if the block size is greater than T, another iteration occurs on the block to choose records with the Jaccard Threshold being incremented to achieve the correct transmission size. If the block size is bk, then the overall runtime complexity will be O((nh*nB)+bk).

In the case of Bi-directional parameter blocking, since the Block Parameters are reorganized and agreed upon ahead of time, the only runtime complexity would be the identification of the correct block. If nb represents the number of records in the block, which will be less than nd, then the runtime complexity will be O(nb).

Let nhi be the number of hierarchy levels agreed upon by Telcos based on the blocking parameter. The level of complexity to navigate the hierarchy in the tuple directional method will be O(nhi * nt), where nt is the number of elements in the tuple that is sent by Telecom B.

6.5.2 Privacy analysis

We will assume that all the Telcos follow an 'honest but curious' policy. When a request for information is made through the process of block identification and extraction, we need to analyze whether there will be a loss of privacy. Each of the three methods sends different pieces of information to Telcos to fetch information. The Dynamic Block creation method sends a small part of the fraud indicator, which seems most likely to be the candidate for a privacy attack. We will conduct experiments to determine if that is the case. The other two methods only send block parameter information, with the two-way method sending the exact block parameter info and the tuple-based method sending a list of parameters.

6.6 Experimental Evaluation and Discussion

We used the same experimental setup as in Chapters 4 and 5. We developed a Python package for each of the blocking techniques with our FD-PPRL protocol using the algorithms mentioned in Algorithms 6.1, 6.2, 6.3, 6.4, and 6.5. We used the North Carolina voter database as a substitute for the Fraud database, where the block-based techniques were applied to that dataset. For our analysis, we established the length of the database, which represents the block of information sent by TA, to be 1000, except in the case of the dynamic block method where we used a Jaccard threshold of JT>=0.3. The Fraud Indicator FA that is being searched for was set to be 'ALICE TEST BOB'.

6.6.1 Experiment 1

We conducted experiments to measure the accuracy of our three block-based protocols based on the equations mentioned in the previous section. As described, the accuracy of the Tuple-based protocol and the Shared Parameter Block protocol simply follow the accuracy of the

SuperMinHash protocol, which we analyzed in Chapter 5. We focused our experiments on the Dynamic Tuple-based protocol, which could be easily implemented without prior setup or collaboration. The threshold or filter cutoff was set at JT>=0.3. We conducted accuracy measurements on signature sizes ranging from N=5 to 50, with increments of 5. The graph in Fig 6.4 shows the accuracy across various signature lengths.

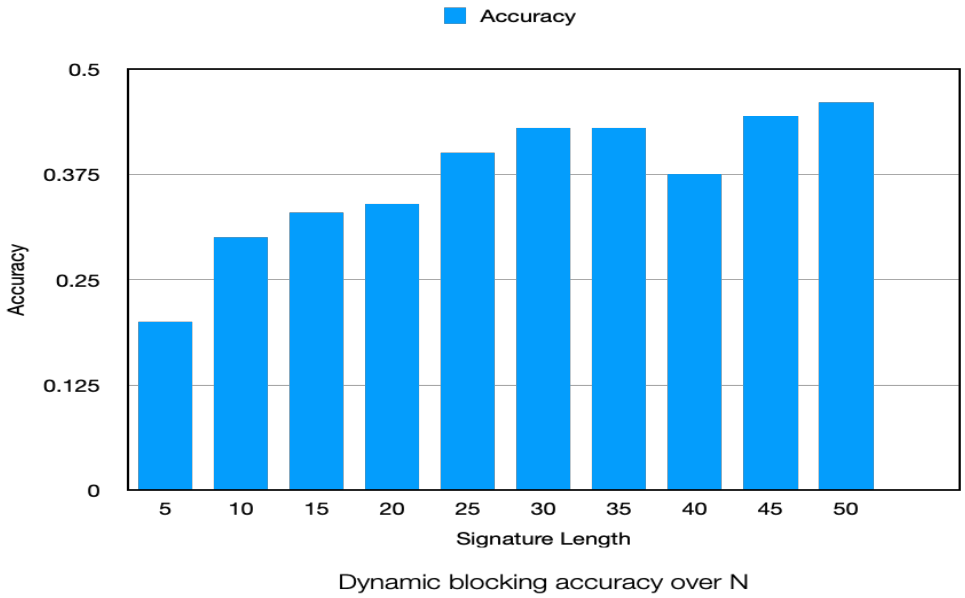

Figure 6.4 Accuracy of Dynamic Blocking over multiple signature lengths

As shown in the graph, the accuracy measure remains under 50%, indicating that while this could be a great option for limiting data transfers without prior setup or agreements, accuracy is greatly reduced. This protocol could work where there is a much lower threshold for fraud detection accuracy, or put another way, the telco chooses to be aggressive in pursuing fraud identification.

6.6.2 Experiment 2

The experiment is focused on analyzing the potential loss for the dynamic block generation protocol since a portion of the fraud indicator is sent for inquiry. While this protocol offers flexibility benefits, especially without prior setup, an analysis is being conducted to determine if the risks of privacy loss outweigh the benefits. We conduct this experiment by sending 10%, 20%, 30%, 40%, and 50% of the fraud indicator and then analyze if privacy is compromised by determining the probability of privacy loss, which can be defined in terms of the Jaccard similarity of the partial fraud indicator that is sent.

% of Char Sent	5	10	15	20	25
10%	0	0	0	0	0
20%	0.2	0.1	0.06	0.05	0.04
30%	0.2	0.2	0.2	0.2	0.16
40%	0.4	0.3	0.27	0.3	0.28
50%	0.4	0.4	0.33	0.35	0.28

Table 6.2 Privacy lose based on the % of Character sent

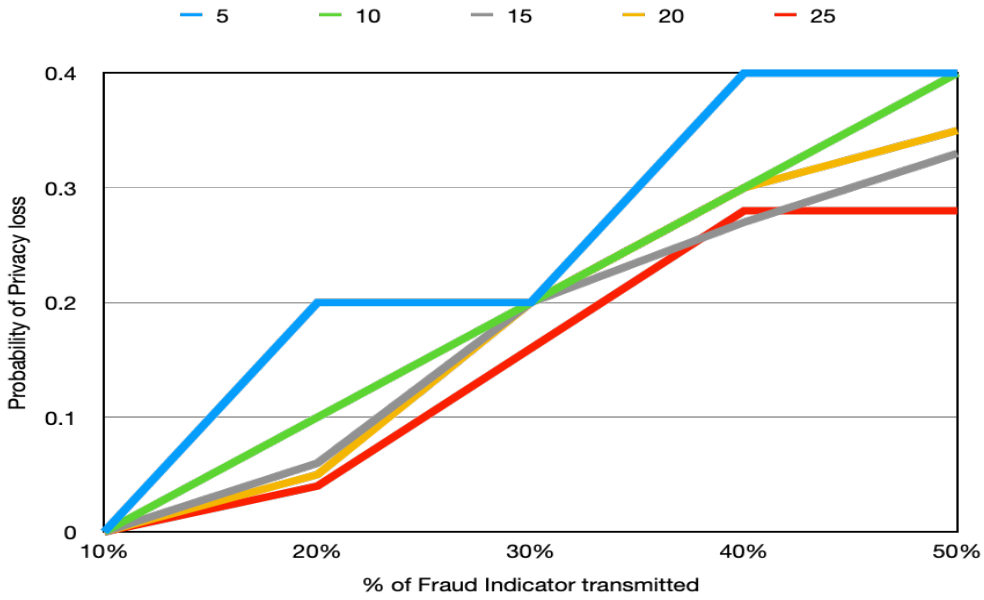

Figure 6.5 Probability of Privacy loss vs.% of Fraud Indicator transmitted

As shown in the experiment results, the danger of privacy loss or potential discovery of the fraud indicator is less when the percentage of the fraud indicator is within the range of 10% - 30%. The results do demonstrate that sending a higher percentage of the fraud indicator for block creation could pose a potential loss of privacy and hence is not recommended.

6.6.3 Experiment 3

We measure the performance of the three block-based approaches over a Transmission dataset of 1000 records. Even though the dynamic block-based approach could have vastly different block sizes, 1000 was chosen to establish a good baseline for comparing the different approaches.

N	SB	DB	TB
5	0.052	0.0464	0.0478
10	0.107	0.1233	0.1239
15	0.1617	0.1626	0.1629
20	0.2165	0.2182	0.2179
25	0.2692	0.3118	0.3108

Table 6.3 Performance measured in seconds for the three methods across various signature sizes

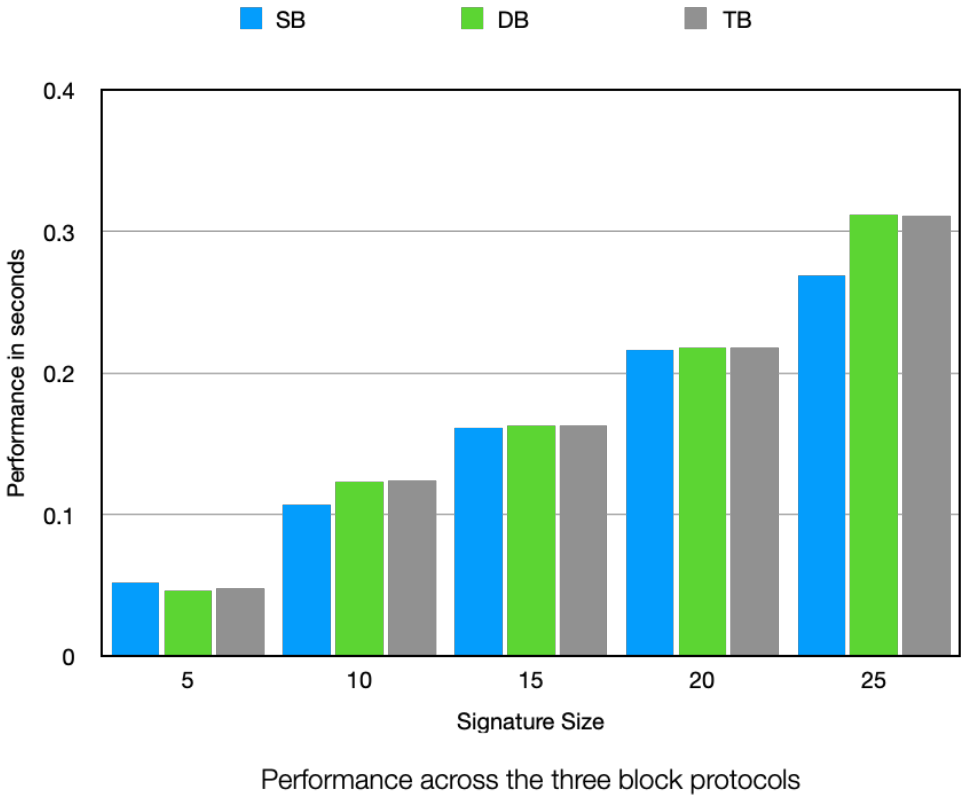

Figure 6.6 Performance of the three methods across various signature sizes

As seen in the results and the graph above, all the three approaches are similar in performance, with the Shared parameter-based approach shown a slightly better performance over larger signature sizes. This is obvious as the search for a block would be the most straightforward of the other two approaches. This approach, however, requires a lot of collaboration and coordination between Telcos.

6.7 Chapter Summary

In this chapter, we reviewed three methods where Telcos can organize their fraud information into blocks with a set transmission size of T. We reviewed the reasons to do so to achieve a faster response time and to attain a higher probability of Fraud information matching. We conceptually reviewed the performance and privacy of the blocks. Given the fact that the Shared Parameter Blocking method and the Tuple-based Blocking Method have the same privacy

factor as the framework described in Chapter 5, we focused on the Dynamic blocking creation method and reviewed it in terms of privacy loss and accuracy. We also reviewed the performance of all three protocols. In the next chapter, we will discuss methods where Telcos can communicate with multiple Telcos at the same time for a faster response in Fraud identification.

Chapter 7: Fraud Detection using FD-PPRL over Publish-Subscribe protocol for Multiple Telco Communication

In this chapter, we introduce a protocol designed for use by multiple telcos wishing to establish a private fraud detection network for information sharing. The protocol aims to be easy to implement, stable, and secure. In Section 7.1, we discuss the necessity for multiple telecoms to communicate. Section 7.2 outlines various potential methods for these telecoms to establish a private, secure network. In Section 7.3, we detail a Publish-Subscribe (pub-sub) method using our PPRL method, which was described in Chapter 4. Section 7.4 is dedicated to a conceptual analysis of our approach. We review the experimental evaluation in Section 7.5. Finally, in Section 7.6, we conclude with a discussion on the pub-sub methods.

Introduction

In Chapter 4, we described a secure way for two Telcos to communicate fraud information using our privacy-preserving record linkage (PPRL) method. In Chapter 5, we described the use of block methods to transact substantial amounts of data between Telcos. As demonstrated in these chapters, communication between the two Telcos is quite successful. However, there are over a hundred telecom companies in the US and worldwide. Therefore, we need to create a protocol for these companies to communicate fraud parameters in an efficient, safe, and secure manner without having to communicate with each other where the time of discovery rises exponentially.

Table 7.1 Notation and Terminology used in this Chapter

Fi_A, Fi_B, Fi_N	T_A, T_B's and T_N's fraud indicators respectively
H	Hash function shared between the N Telcos (SuperMinHash)
T	Size of the block of signatures transmitted by T_B
N	Size of the hash signature
SH	SuperMinHash algorithm
J_T	Jacquard matching threshold
M	Size of all records in the fraud dataset
C	Count of elements in the Block being created
Inc	Incremental factor for J_T
J_S	Jacquard indicator
K_A	T_A's private key
K_B	T_B's public key
$H(S)$	Entropy of the protocol
p	Probability of the bit being identified
g	Portion of the Fraud Indictor that is transmitted
K	is the bit length if the EdDSA protocol

Table 7.1 Notation and Terminology Chapter 7

Many telcos today form part of a consortium or group based on their size, product offerings, and geographic location. Some such groups that exist today are the 'Telecommunications Industry Association,' US Telecom, and the Wireless Communication Alliance, among others. It would only be natural for such companies to come together, provided they have the right technology and means

to share fraud information. This could be mutually beneficial to each other and, in turn, protect the industry in general.

In this chapter, we will first describe a potential network solution for telecommunications companies to share fraud data that is secure, efficient, and fast. Over this network, we will propose our solution for telecommunications companies to communicate with each other.

7.1 Telcos Communication network

The Telcos can choose several types of network technologies to communicate with each other with either a ring or mesh topology.

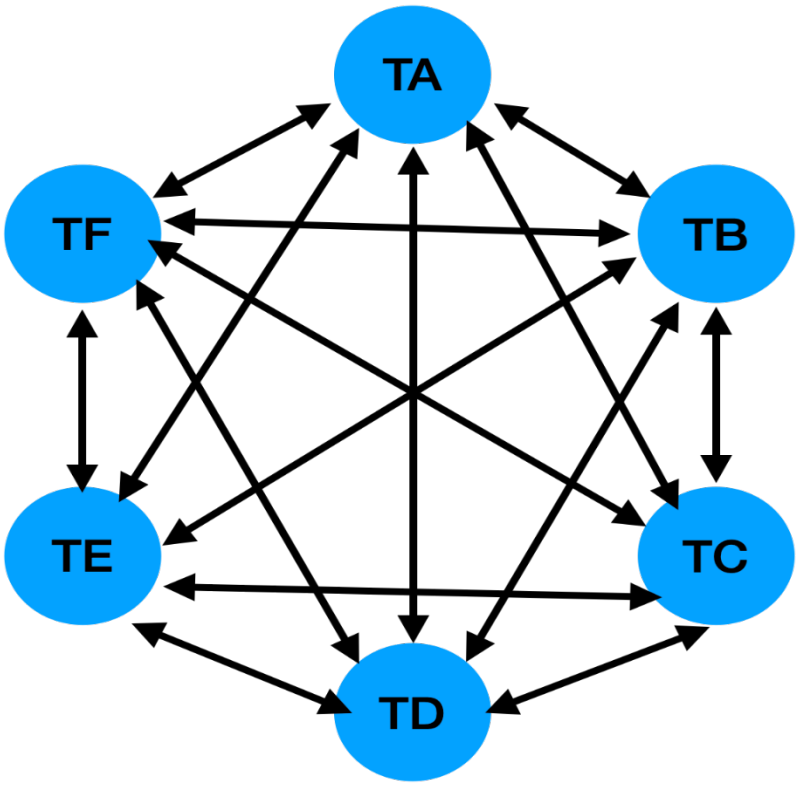

Figure 7.1 Mesh network of Telecom companies

1.) NaaS network -

The Network-as-a-Service model [35] for telecom can bring network subscription-based hardware, software tools, and services that help telecom companies communicate with each other to share fraud parameters. It is a tightly integrated model that combines software and hardware solutions for communications.

2.) SDWAN -

A Software-Defined Wide Area Network [36] is a beneficial technology that can be used by a group of telecom companies wishing to collaborate on sharing their fraud information. It can be utilized for cloud-based services and processing. This technology has the potential to reduce costs and improve performance.

3.) SASE network -

The Secure Access Service Edge (SASE) is a framework [37] that combines SD-WAN and security solutions in a cloud-based platform to offer safe, fast, and secure network connections for telecom companies.

4.) Virtual Private Network -

Telecom companies can opt to use a low-cost VPN where transmissions are secure and private.

7.2 Pub-Sub Method

Although Telcos have the flexibility to select from a broad range of communication topologies and network configurations, some of which have been previously outlined, our recommendation is to employ a Publish/Subscribe (Pub/Sub) [38] model for Telos communication.

7.2.1 Pub/Sub Model -

This is an asynchronous communication model that can facilitate communication between different Telco Companies within a network. This system is widely used in distributed locations where components need to communicate without being tightly coupled.

In this model, there are four key components: messages, topics, subscribers, and publishers. A message is sent from the sender to the receiver. Each message has an associated topic that acts like a channel between the involved parties. The subscribers, in this case Telcos, are recipients who must register to the topic of interest, such as Fraud Identifiers. The publisher will create a fraud topic and send it to all the Telcos for them to check. The interaction is a one-to-many relationship.

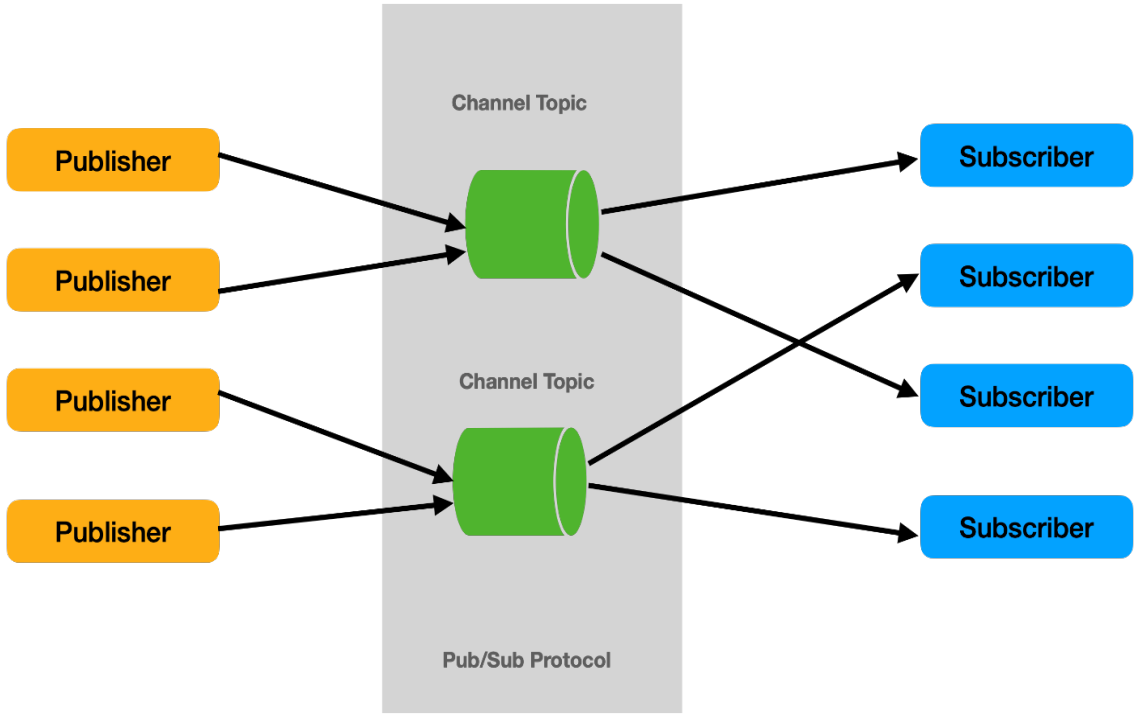

Figure 7.2 Pub/Sub protocol method

In Chapters 4 and 6, we described a protocol that enables two Telcos to safely communicate using various methods to detect fraud. When checks are needed from multiple Telcos, the complexity of operation is O(n*m), where m is the number of Telcos and n is the number of communication steps with each Telco. As can be seen, the scale of complexity increases m-fold as the number of Telcos increases. To obtain immediate information on a fraud occurrence, there will be a need to communicate with all the Telcos at the same time. To accomplish this, we have designed two methods as described below. The Pub/Sub message provides instant notifications for Telcos, which will be distributed and is a scalable and reliable communication method.

1.) Pub/Sub Fraud detection method without blocking-

Pub/Sub Fraud detection method without blocking is a process where the TB publishes the Encoded Fraud Indicator and Telcos respond with a confirmation of the presence of Fraud Information.

2.) Pub/Sub Fraud detection method with blocking -

Here, a block tuple containing a small portion of the fraud indicator is published to elicit a response from Telcos.

7.3 Overview of our approach

Fraudsters can defraud multiple companies by exploiting the lack of cooperation between Telcos. Although Telcos compete, they work in cohesion to complete a telecom network request for their customer with appropriate compensation. If they are provided with a platform that has the right technology to satisfy speed, reliability, and security, they can come together to share fraud information even though it is in a distributed manner.

The need to obtain fraud information from other Telcos quickly is of extreme importance, especially in situations where fraud prevention is critical. There are many Telco organizations that come together in an organized manner, either through industry organizations or groups. We described various communication network possibilities in section x for Telcos to communicate with each other all at once. The same approach can be extended to other industry groups and across industries.

In this protocol, the request is made by Telecom T_B for fraud information and in the below section we will describe how the Telcos T_A, T_C….T_N respond to request and respond back to T_B with the appropriate information.

7.3.1 Pub/Sub (FS) Fraud detection method without blocks

In this section we describe the protocol in detail

a.) Parameter determination -

When T_B has a fraud occurrence and wants to inquire about its Fraud Indicator Fi_B, only the block parameters or a portion of the fraud indicator is sent in Chapter 4, 5, and 6. In this protocol, the entire Fraud Indicator is hashed and is encoded with the digital signature and is sent for fraud verification. The public key K is also published.

b.) Record encoding -

The entire Fraud Indicator is encoded using the EdDSA algorithm described in Chapter 4. As with other implementations, there is no need for Telcos to encode their data as it will not be transmitted with this protocol implementation.

c.) Pub/Sub Protocol construction -

This protocol has the concepts of a publisher and subscriber. The publisher is where a Telco company outputs a piece of information that needs to be broadcast, and the subscriber is one that listens or sources that information on a constant basis. Typically, subscribers will not be listening to all messages in the message bus. They will have to subscribe to a specific topic. In this case, it will be the topic of Fraud Detection. In this Fraud Sharing network, a Telecom company can play the role of a publisher and subscriber. The Pub/sub protocol that will be deployed across the Telcom companies' joint infrastructure has two main domains: the data domain and the control domain. In the data domain, this is where messages move between the publishers and subscribers, and the Control domain is where the servers are assigned to the publisher or subscriber which are called routers. The servers in the data plane are called forwarders. Event.publish is the action protocol for Telecom B to publish the Fraud event, and event.subscribe is where the N subscribers listen to the Fraud topic.

Algorithm 7.1 Pub/Sub Fraud detection method without blocking

Input:

Fi_A, Fi_B, Fi_N - T_A, T_B's and T_N's fraud indicators respectively

H - Hash function shared between the N Telcos (SuperMinHash)

N- Size of the hash signature

T – Size of the block of signatures transmitted by T_B

SH – SuperMinHash algorithm

J_T - Jacquard matching threshold

M is the size of all records in the fraud dataset

C - count of element in the Block being created

Inc - incremental factor for JT

Outputs:

J_S - Jacquard indicator

K_A - T_A's private key

K_B -T_B's public key

Match = True if $J_S > J_T$ (Jacquard threshold); else, False

Steps:

$(a_0, a_1, a_2, \ldots\ldots a_{N-1})$ [0,1]

$(a_0, a_1, a_2, \ldots\ldots a_{N-1})$ <- (0, 0….0)

C<-0

Event.topic<- "Fraud Detection"

$H1_B, H2_B, H3_B \ldots. HN_B)$ <-SH(Fi_B) generate SuperMinHash for T_B fraud indicator

KB_A, KB_B <- EdDSA.generateKey() performed by T_B

event.publish$_B$(event.topic, KB_A, KEi_B)

event.subscribe$_A$(event.topic,KB_A,KEi_B),event.subscribe$_C$(event.topic,KB_A, KEi_B)….event.subscribe$_N$(event.topic, KB_A, KEi_B)

For all T_A…. T_N performs

For j <- 0,1, 2…...M-1 do

HjF_A……HjF_N (Hash data of the N telecom fraud dataset)

For i <- 0,1, 2…...N-1 do

a_j <- EdDSA.verify (HjF_A, KEi_B)

End For

$$J_S = \frac{1}{N}\sum_{j=0}^{N-1} a_j$$

If $J_S \geq J_T$

Then break

End If

End For

End For

$J_S \geq J_T$ then Match=True; else, False

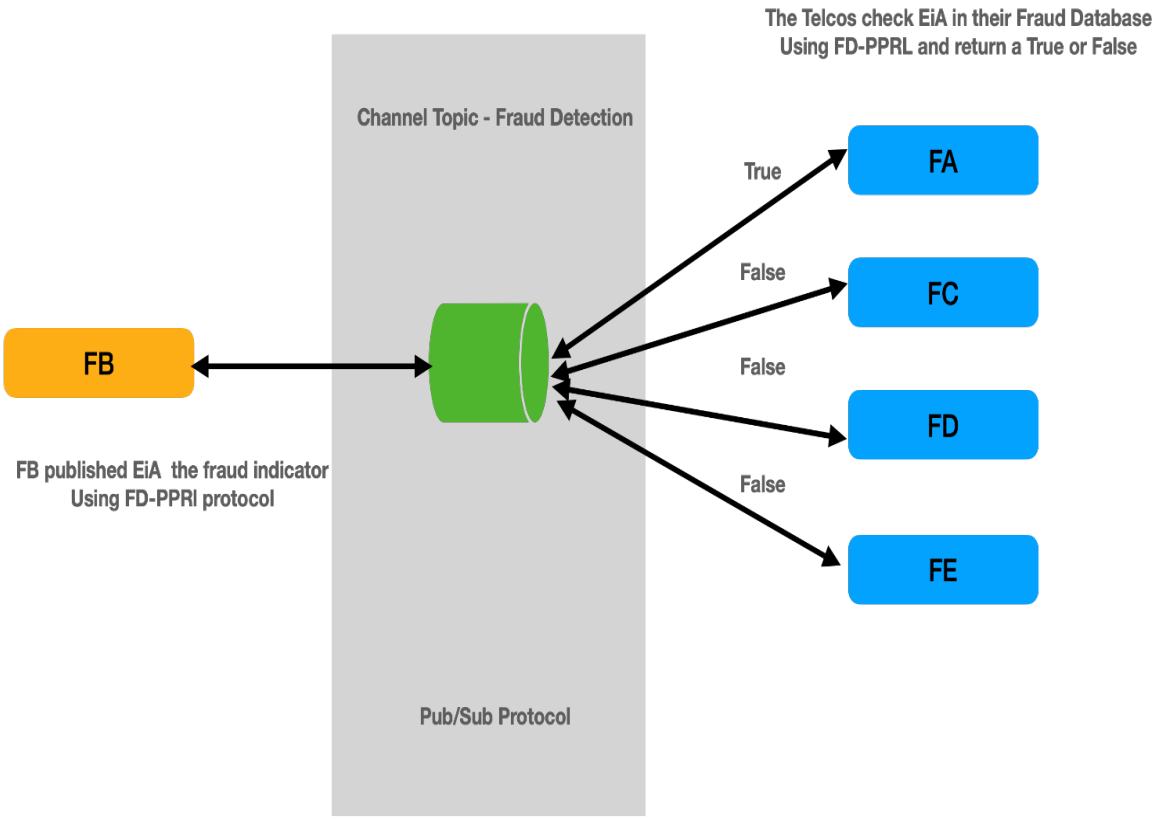

Figure 7.3 Pub/Sub protocol without Block (FS)

As described in Algorithm 7.1, Telcom B wants to make an enquiry with N telcos at once on the network to determine whether its indicator has committed fraud in the past, so that it can take the next steps towards fraud prevention. Unlike in Chapters 4, 5, and 6, where the full fraud

indicator is not sent out, here the MinHashed encrypted fraud indicator is published with the public key with a Fraud Enquiry topic into the Pub/Sub network.

The N Telcos listening/receiving this published message then takes this encrypted fraud indicator KEi_B for processing. The Telcos take this fraud indicator through their entire fraud database to determine if there is a match if it meets its Jacquard Threshold. The comparison is done against the unencrypted fraud indicator database of the Telco, still fulfilling the condition of comparing on a non-homologous dataset. Once a specific or multiple Telcos get a confirmation from Step 7 whether the Fraud Indicator is present, they send a success message; otherwise, they send a failure message. The main advantage of this message is that the request for information is sent to multiple Telcos at once, and a fraud presence response is received quickly. We will delve into the advantages, disadvantages, and potential privacy loss in Section 7.4.

Since the entire encrypted fraud indicator is published for identification, we would like to determine the likelihood of privacy being compromised. As discussed in Chapter 4, privacy is indirectly correlated to the amount of entropy in the protocol. We will start by determining the entropy assuming there are many Telcos participating in the protocol. While the typical probability of a bit being compromised is 1/2, we will assume that the probability can range from 0 to 1 to determine the full entropy of the protocol with all the participating Telcos. Hence the entropy of the digital signature S is

$$H(S) = \sum_{j=1}^{N} \int_0^1 -p \ \log(p)\, dp \ = \ \sum_{j=1}^{N} \frac{1}{4} \tag{7.1}$$

For a signature size N=1, the Entropy is ¼

The performance of the protocol will be dependent on the network, the pub/sub response rates, and the performance of the FD-PPRL framework, which we have reviewed in Chapters 4, 5, and 6. As given in the above equation, the entropy is lower than a two-way Telco implementation,

and in order to preserve privacy, it would be advisable to largely increase the signature sizes. We will be evaluating the performance of this protocol over large signature sizes to determine the feasibility of adoption.

7.3.2 Pub/Sub Fraud detection method with dynamic block creation (PSB)

In this section we describe the protocol in detail:

a.) Parameter determination

Like Section 7.31, when Telecom B has a fraud occurrence and wants to inquire about its Fraud Indicator (Fi_B), where the actual fraud indicator is sent, here Telecom B chooses to publish only a part of the Fraud indicator. This could be the first few characters of the indicator or a random sampling of the same. The public Key (K_B) is also published.

b.) Record encoding

The partial Fraud Indicator that is to be published is encoded using the Edwards-curve Digital Signature Algorithm (EdDSA). Once the dynamic block is created, the Telcos' Fraud indicators are encoded and then transmitted as described in Chapter 4.

c.) Pub/Sub Protocol construction

Using the concepts described in Section 7.2, Telecom B publishes the sample Fraud Indicator to see a response from Telcos. Once a Telco or Telcos with the most promising prospect is selected, individual connections are established with Telcos to transact the requested Fraud Information one way for Fraud identification as described in Algorithm 7.2.

Algorithm 7.2 Pub/Sub Fraud detection method without blocking

Inputs:

Fi_A, Fi_B, Fi_N - T_A, T_B's and T_N's fraud indicators respectively

H - Hash function shared between the N Telcos (SuperMinHash)

N - Size of the hash signature

T – Size of the block of signatures transmitted by T_B

SH – SuperMinHash algorithm

J_T - Jacquard matching threshold

M - is the size of all records in the fraud dataset

C- count of element in the Block being created

Inc - incremental factor for J_T

Outputs:

J_{max} - Jacquard indicator

K_A - T_A's private key

K_B -T_B's public key

Match = True if $J_S > J_T$ (Jacquard threshold); else, False

Step:

1 $(a_0, a_1, a_2, \ldots\ldots a_{N-1})$ [0,1]

2 $(a_0, a_1, a_2, \ldots\ldots a_{N-1}) \leftarrow (0, 0 \ldots . 0)$

3 $C \leftarrow 0$

4 *Event.topic<- "Fraud Detection"*

5 $BK_B \leftarrow extract\_parameter(Fi_B)$

6 $J_{max} \leftarrow 0$

7 *(H1BK$_B$, H2BK$_B$, H3BK$_B$, …., HNBK$_B$) <-SH(BKB) genetrate SuperMinHash for T_B's fraud indicator*

8 *KB$_A$, KB$_B$ <- EdDSA.generateKey()*

9 K*Ei$_B$ <-EdDSA.encode (KB$_B$, HiBK$_B$)* ∀ *i*∈ *(0,1…(N-1)*

10 event.publish$_B$(event.topic, *KB$_A$, KEi$_B$*)

11 event.subscribe$_A$(event.topic,*KB$_A$,KEi$_B$*),event.subscribe$_C$(event.topic,*KB$_A$,KEi$_B$*)….event.subscribe$_N$(event.topic, *KB$_A$, KEi$_B$*)

12 For all Telco $T_A \ldots T_N$ performs database scan

13 For *j* <- 0,1, 2…...*M*-1 do

14 *KEi$_B$*

15 For *i* <- 0,1, 2…...*N*-1 do

16 *a$_j$ <- EdDSA.verify (KEI$_B$, Hi$_A$)*

17 End For

18 $$J_S = \frac{1}{N}\sum_{j=0}^{N-1} a_j$$

19 If *J$_S$ >=J$_T$*

20 Then *BK$_A$*, *J$_S$*<- *BK$_A$*

21 *C=C+1*

22 End If

23 End For

24 If count (*BK$_A$*) > T then

25 *J$_T$=J$_T$ + inc*

26 *C*<-0

For i<- 1….C-1 do

BK_A

If J_S >=J_T

Then BK_A, J_S<- BK_A

End If

End For

End If

End For

For All N telcos publish Jmax in success message

Telcom B choose the Telco or Telcos with the highest Jmax

The dynamically created block information is sent TB

T_B performs

For j <- 0,1, 2…...M-1 do

Ej_A

For i <- 0,1, 2…...N-1 do

a_j <- *EdDSA.verify* (Ei_A, Hi_B)

End For

$$J_S = \frac{1}{N}\sum_{j=0}^{N-1} a_j$$

If $J_S \geq J_T$

Then break

End If

48 End For

49 $J_S \geq J_T$ then *Match*=True; else, False. To indicate the presence of Fraud

Algorithm 7.2 provides more detail on how Telco B solicits dynamic block information from multiple Telcos at once using the Pub/Sub method. Unlike in 7.1, Telco B does not publish the entire Fraud Indicator. Instead, it takes a small part of the Fraud Indicator as described in Chapter 6 and uses the dynamic blocking method to ensure privacy is not compromised. The Block parameter is MinHash and encoded using the framework mentioned in Chapter 4.

The encoded digital signature along with the public key is published into the message bus with the topic of "fraud detection." The N Telcos listening take the message for further processing. Each of the Telcos then takes a partially encrypted fraud indicator and compares it against the unencoded fraud indicators in their database. During the comparison process, Telcos determine the Jaccard similarity coefficient for each fraud indicator while simultaneously tracking the maximum Jaccard value in their dataset against the T_B's partial fraud indicator. The block to be shared is also dynamically created and ready for transmission.

After the processing is complete, the Telcos publish their maximum Jaccard value when it is greater than zero. A Telco for which the Jaccard value is zero, simply responds with a failure. With this information, Telco B can choose to communicate with the Telco of their choice, which will be a much smaller set. Typically, the most logical choice will be to request the block information from the Telco with the maximum Jaccard value from the comparison. Once the request is made to the Telco of choice, Telco B sends its dynamically created block for fraud identification. Telco B then processes the block to identify the presence of fraud indicators as per the framework described in Chapter 4.

Since only a portion of the fraud indicator is published for action in this version of implementation, namely the entropy will be quite high, and the potential loss of privacy is much lower.

Let G be the total size of the Fraud Indicator where g is the transmitted portion. Just as in the above method the Entropy can be given by,

$$H(S) = \sum_{j=1}^{K} \frac{G}{g} \times \int_0^1 p \ \log(p)\, dp = \sum_{j=1}^{K} \frac{G}{4g} \quad (7.2)$$

Where for a signature size N=1

$$H(S) = \frac{G}{4g} \quad (7.3)$$

7.4 Conceptual Analysis of the Pub/Sub methods

In this section, we analyze the two pub/sub fraud detection methods for data sharing in terms of complexity, quality of blocking and privacy on the data shared with each Telco.

7.4.1 Complexity

We analyze the computational and communication complexity of the two methods described above, focusing on two main aspects: record request and response. Let *nf* represent the size of the block indicator for which information is requested. The record complexity of the input being MinHashed and encrypted using the Chapter 4 framework is denoted as *O(nf)*. Now, let *M* be the maximum number of database records across *N* telecommunication companies. Since the search is conducted in parallel, the runtime complexity of the search is *O(M)*. Therefore, the total complexity for the pub-sub method without blocking becomes *O (M + nf)*.

In the case of the pub-sub method with blocking, the creation of the blocking parameter to be published incurs a complexity of $O(nf)$. Like the previous scenario, with M representing the maximum database size and the block being dynamically created in parallel, the runtime complexity remains O(M). Once processed and chosen, let nbk denote the block size of the returned block, where k represents the number of telecommunication companies returning information. The fraud detection step adds a runtime complexity of $O(nbk)$. Therefore, the total runtime complexity for the overall process becomes $O(M + nf + nbk)$. In the experiment section, we will conduct an analysis and measure the time performance of these two methods.

7.4.2 Publish-response quality

We analyze the quality of the published record and its corresponding response in two methods to determine which method provides the highest quality of information for the Telco to obtain the right amount of information to detect Fraud. In the case of the pub-sub method, without blocking the entire fraud, an encrypted fraud indicator is published. The quality of the response and the presence of a confirmed Fraud indicator will be high since it is either a True or False response. On the other hand, the second method's selection is based on the selection of the Telco with the highest J_S, which will be used to request the Block of Fraud Indicators for matching purposes.

7.4.3 Privacy analysis

Just as in Chapter 6, we will assume that all the Telcos follow an honest but curious policy. When a request for information is made through the process of pub-sub, we will analyze if there will be a loss of privacy. Each of the two methods sends different pieces of information to Telcos to fetch information. In the pub-sub method without blocking, which naturally has the least

complexity and highest response quality, the risk of privacy leaks could be high as the Fraud Indicators themselves are published.

In the pub-sub method without blocking, only the partial fraud indicator is published, so the risk to privacy is small. The only risk to privacy will be with the block information sent, and T_B could try to extract information from other blocks.

7.5 Experimental Evaluation and Discussion

7.5.1 Experiment I (With matches)

We conducted experiments to measure the accuracy of our two pub-sub methods with FD-PPRL based on the equations mentioned in the previous chapter. We determined that the performance of the protocols with signature sizes N <20 is comparable to what is described in Chapters 4, 5, and 6. As demonstrated by the entropy equations and the number of Telcos who could potentially subscribe to this solution, we decided to focus our experiments on larger signature sizes and measure the performance with that in scope. This was done because the entropy was low at lower signature sizes, and to increase it, we had to increase the signature sizes. We measured the performance of the pub-sub protocol without blocking (PS), and the pub-sub protocol with blocking (PSB) to determine at large signature sizes which implementation would be feasible for an industry application.

With Matches		
N	**PS**	**PSB**
10	0.2169	0.107
20	0.4453	0.2169
30	0.6857	0.3222
40	0.9187	0.4392
50	1.1583	0.5417
60	1.400	0.6527
70	1.6565	0.7868
80	1.8966	0.8893
90	2.1413	0.9839
100	2.3819	1.103

Table 7.1 Performance of PS and PSB protocol with matches

We used ‘ALICE TEST BOB’ as the Fraud Indicator that needs to be verified in FS. ‘ALICE’ was used as the partial Fraud Indicator for FSB. Even though it would seem most intuitive that publishing the entire fraud indicator would solicit a faster response, the PSB method is remarkably faster in performance. This is mainly because only a partial fraud indicator is sent and at larger signature sizes performs well when the block is dynamically created and sent for verification. Also, since the entropy of PS is lower per Equation 7.1 and the entropy of PSB is

much larger per Equation 7.3 since the protocol is based on the lower number of characters or information that is sent.

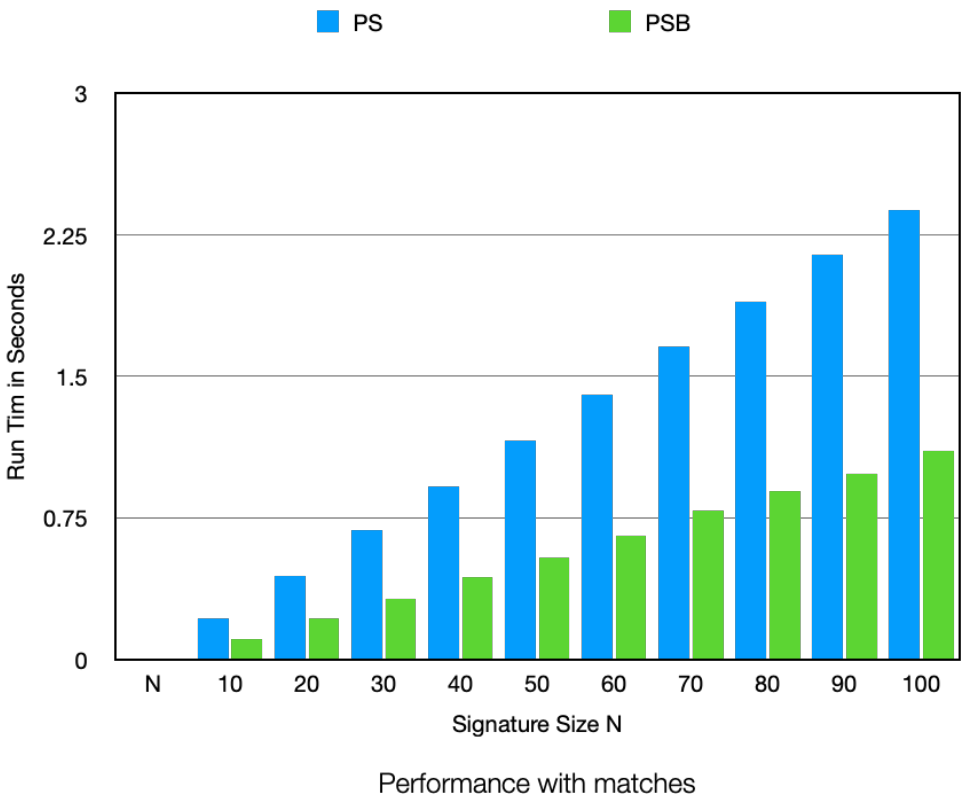

Figure 7.4 Performance in seconds with matches across various signature Lengths N

7.5.2 Experiment II (Without Matches)

Just as in the above experiments we measured the performance of PS and PSB where Telcos do not find any matches at all. This is where a full data scan is conducted of the fraud data.

Without Matches		
N	PS	PSB
10	1.8688	0.107
20	3.7227	0.2161
30	5.5873	0.3487
40	7.5189	0.4408
50	9.336	0.544
60	11.2965	0.6572
70	13.2658	0.7927
80	15.101	0.8853
90	17.159	0.9890
100	19.108	1.1070

Table 7.2 Performance of PS and PSB protocol without matches

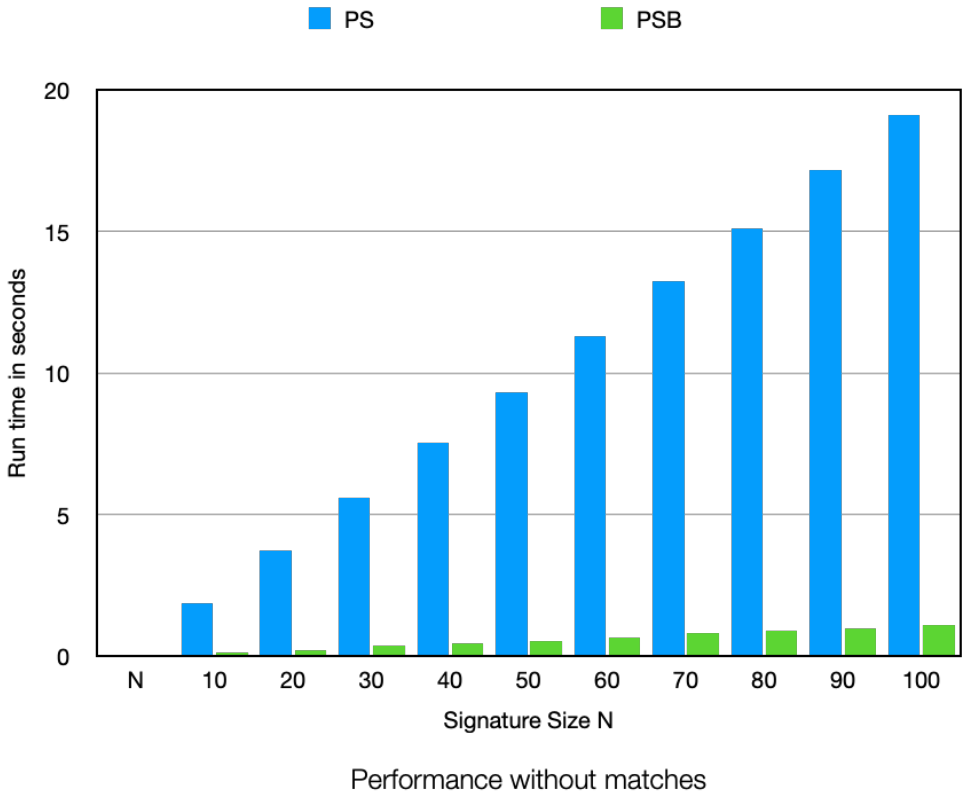

Figure 7.5 Performance in seconds without Matches across various signature lengths

As demonstrated in the graph, FSB performed far better than FS, even though a dynamic block was sent to check for Fraud Indicator presence. If no response is generated with partial fraud indicators in FSB with the Telco, the performance will be even better since no blocks are sent for verification.

Chapter Summary

In this chapter, we discussed two main protocols where a Telco can communicate with multiple Telcos at the same time to identify Fraud. We reviewed some potential networks like mesh that can be employed over some types of networks like VPN or SASE on which the FD-PPRL protocol can be deployed to communicate across many Telcos. We introduced the concept of the Publish/Subscribe method for Telcos to effectively communicate with each other. We introduced two protocols with FD-PPRL built over the Pub/Sub method where one method was without blocks

and the other with blocks. A conceptual analysis of the two methods was discussed with experiments conducted to determine their effectiveness. In conclusion, both the methods offer many advantages where the Telcos can use them effectively to prevent Fraud. In the next chapter, we will conclude our dissertation summarizing our contribution and providing future directions.

Chapter 8: Conclusion and Future Directions

In this dissertation, the main objective is to present comprehensive research in Fraud Detection using Privacy Preserving Record Linkage (FD-PPRL). In the previous chapters, we have described the FD-PPRL framework, evaluation, and selection of the correct Hash Technique implementation. We also reviewed various blocking techniques for efficient storage and transmission of data, and a protocol to communicate with multiple telcos for a faster response. These were in response to various challenges faced, and a proposal to help address the same. In Section 8.1, we will review the challenges presented in Fraud Detection in the Telco industry. In Section 8.2, we will outline our contributions to address the same. In Section 8.3, we will review probable future directions, and a conclusion will be presented in Section 8.4.

Outline of the Research Problem

As mentioned in the introduction, telecommunication companies enable global tech connections. However, they face numerous fraud attacks daily. They must ensure the security of the data they and their customers transmit over their network, as their products increasingly virtualize our lives. Fraudsters exploit the fact that telcos' fraud management systems operate independently. When sharing information, telcos need to respect privacy laws and protect their business secrets. The main challenges they encounter are: 1) how to share fraud information without disclosing sensitive data; 2) how to share information without third-party interference; and 3) what protocol to use that is safe, secure, fast, and easy to implement.

Even when an appropriate FD-PPRL framework has been established, there is a challenge of storing and transmitting the data efficiently. Most implementations are primarily between two parties. Therefore, to be effective, there is a need for communication or transactions with multiple parties simultaneously to ensure a fast identification of fraud attacks.

8.1 Challenges faced to identify and shared Fraud information in the Telecom industry

8.1.1 Develop a method that can measure the similarity between two dissimilar data sets

A common limitation of many PPRL methods is that they require the same encryption technique to be used by both parties involved in the data exchange. This results in applying the same methods to both data sets. However, this does not guarantee privacy, as some cryptanalysis methods can expose the data, rendering the existing protocols useless. The problem is to find a way to compare two different databases using a similarity measure, such as Jaccard similarity, that can work with diverse data sets and still produce accurate results, while preserving privacy.

8.1.2 Design a method that can exchange data a simple request reply mechanism

A simple request-reply protocol is desirable for the telecom industry, as it can compare data sets in only two steps. Many current PPRL methods are complicated and require more transactions when a third party or a linkage unit is involved, which affects the level of privacy protection.

8.1.3 Design a protocol that works without a reliable intermediary

A privacy-preserving match by a third party for two transacting entities is not likely to gain traction in the telecom industry. This is because a third party would need some regulation or

oversight from an independent body, which would be expensive at best. Moreover, third-party solutions assume that they are only partially honest, which makes them unsuitable for the complex and evolving privacy laws in many countries.

8.1.4 Develop a protocol that is secure and fast

Current protocols for fraud detection are complicated and require many transactions, which slows them down. To catch a fraudster fast and stop more harm, the protocol must respond very quickly. But, making the protocol faster can reduce security, and the other way around. The best protocol for the telecom industry would be one that is both fast and secure simultaneously.

8.2 Summary of our contribution to address these challenges

8.2.1 Developed FD-PPRL Framework for Fraud detection for Telco

We developed a novel FD-PPRL system (Chapter 4) that overcomes the major drawback of existing approaches, where data encoded with the same techniques can be easily breached, as well as many PPRL methods that rely on unreliable Bloom filters or third parties for matching. Our system is a secure, fast, and accurate solution for fraud detection in the telecom sector, based on a SuperMinHash scheme integrated with a DSS, EdDSA. In Chapter 5, we evaluated various hash techniques with FD-PPRL and the performance, privacy, ease of implementation to choose the best technique for FD-PPRL.

We demonstrated and verified how our method can utilize the Jaccard index to identify matches between two heterogeneous data sets while preserving their privacy. We conducted experiments to assess the efficiency of our PPRL system and compared it with existing approaches. Our findings indicate that our method provides a significant improvement in performance and a superior level of privacy protection. The FD-PPRL framework [39] was published in, The Journal of Engineering after going through a rigorous peer review process.

8.2.2 Developed Blocking methods with FD-PPRL

We developed three blocking methods (Chapter 6) with FD-PPRL to efficiently store and transmit data. The first method is a Shared Parameter Blocking method designed to be simple and straightforward, but requires prior coordination, agreement and setup-with the Telcos for the protocol to be effective.

We developed three blocking methods (Chapter 6) with FD-PPRL to efficiently store and transmit data. The first method is the Shared Parameter Blocking method. It is designed to be simple and straightforward, but it requires prior coordination, agreement, and setup with Telcos for the protocol to be effective.

The second method is a dynamic block creating method which requires no coordination or prior setup. The blocks to be transmitted can be created on the fly for identification.

The third method is tuple-based, which offers Telco the flexibility to organize the data however they see fit. It still allows for efficient block identification and response when a fraud information request is made.

8.2.3 Developed FD-PPRL Pub-Sub Method for multi telco communication

We developed two protocols with FD-PPRL built over the Pub/Sub method, one method was without blocks and the other with blocks, as discussed in Chapter 7. These methods can be used effectively by a Telco to communicate with multiple Telcos at the same time to elicit a response to identify fraud quickly. One method is where the entire fraud indicator is shared, and the other where only a portion of the fraud indicator is sent to get a response for identification. Based on the conceptual analysis of the two methods and the experiments conducted to determine their effectiveness, Telcos can choose the most appropriate method for implementation.

8.3 Future Directions

8.3.1 FD- PPRL in other industry

In this dissertation, we focused our work on fraud detection through sharing in the Telco industry. There is a need to share vital information in other industries while preserving privacy. One use case is the health care industry where there would be a need to check on patient records across multiple hospitals and states that use different EMR systems. Privacy and patient information protection laws like HIPAA are strong in this industry which prevents sharing patient information. We would like to investigate the use of FD-PPRL in this industry.

8.3.2 FD-PPRL across industry

While segments and companies in a particular industry act in silos, which is a major hurdle in fraud detection and prevention, each industry segment is further siloed. For example, the medical industry does not interact much with the Telco industry, and vice versa. Through this, FD-PPRL has proved to be highly effective in breaking down silos within industries to share information. We would like to investigate the use case where FD-PPRL can be used to share information across multiple industries to foster beneficial cooperation on information sharing while preserving the privacy of information with fervor.

8.3.3 FD- PPRL with AI

We would like to investigate the use of AI in cryptanalysis attacks on our frameworks to determine if privacy will be compromised. Theoretically, the odds or probabilities are remote; however, we would like to conduct experiments to determine if privacy will be compromised through any of the available generative AI mechanisms.

We would also like to incorporate generative AI into the FD-PPRL framework for effective and safe fraud detection through sharing or any viral information sharing while preserving privacy.

Conclusion

This dissertation was the first to present comprehensive research in Fraud Detection through sharing among Telcos using Privacy Preserving Record Linkage (FD-PPRL). First, through an extensive review, we identified several research questions and gaps for Fraud Detection in the Telco Industry. We then proposed the FD-PPRL framework [39] to address these gaps which was published in the Journal of Engineering after a rigorous peer review process. We proposed methods to store and transmit substantial amounts of data and communicate with multiple Telcos at the same time. We conducted experiments to demonstrate the speed of PPRL and even compared our implementation with current methods, finding that our protocol provides a remarkable improvement in performance and a superior ability to maintain privacy. Telcos can use our protocol to share fraud data with short response times with high security while complying with privacy laws. To conclude, the work presented in this dissertation provides an insight into the importance of Fraud detection through sharing and FD-PPRL can be used in many real-world industries.

Appendix - Published Paper contributing to Dissertation

Thomas, Satish & Sluss, James. (2023). Fraud detection through data sharing using privacy-preserving record linkage, digital signature (EdDSA), and the MinHash technique: Detect fraud using privacy preserving record links. The Journal of Engineering. 2023. 10.1049/tje2.12341.

References

1. Chen, Z.: A study of fraud types, challenges and detection approaches in telecommunications. J.Inf. System Telecommunications. 28(7), 248–261 (2020). https://www.researchgate.net/publication/343229226
2. Yelland, M.: Fraud in mobile networks. Computer. Fraud Security. 2013, 5–9(2013)
3. 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]* OJ L 119/1
4. Vatsalan, D.: Scalable privacy-preserving linking of multiple databases using counting Bloom filters. In: *Proceedings of the IEEE 16th International Conference on Data Mining Workshops*, pp. 882–889. IEEE, Piscataway, NJ (2016)
5. Vaiwsri, S., Ranbaduge, T., Christen, P.: Accurate and efficient privacy preserving string matching. *Int. J. Data Sci. Anal. 14(2), 191–215(2022)*
6. Bhattacharya, M. West, J: Intelligent financial fraud detection: A comprehensive review. *Computers & Security Volume 57, March 2016, Pages 47-66.*
7. Vidanage, A., Christen, P., Ranbaduge, T., Schnell, R.: A graph matching attack on privacy-preserving record linkage. In: *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pp. 1485–1494. IEEE, Piscataway, NJ (2020).

8. J. Ji, J. Li, S. Yan, Q. Tian and B. Zhang, "Min-Max Hash for Jaccard Similarity," *2013 IEEE 13th International Conference on Data Mining*, Dallas, TX, USA, 2013, pp. 301-309, doi: 10.1109/ICDM.2013.119.
9. Adir, A., Aharoni, E., Drucker, N., Kushnir, E., Masalha, R., Mirkin, M.,Soceanu, O.: Privacy-preserving record linkage using local sensitive hasha and private set intersection. arXiv:2203.14284 (2022). https://arxiv.org/abs/2203.14284
10. Thada, V., Jaglan, V.: Comparison of Jaccard, dice, cosine similarity coefficient to find best fitness value for web retrieved documents using genetic algorithm. Int. J. Innov. Eng. Technol. 2(4), 202–205 (2013).
11. Schnell, R.: Randomized response and balanced bloom filter or privacy preserving record linkage. In: 2016 IEEE 16th International Conference on Data Mining Workshops, pp. 218–224. IEEE, Piscataway, NJ (2016).
12. Randall, S., Ferrante, A., Boyd, J., Bauer, J., Semmens, J.: Privacy preserving record linkage on large real-world datasets. J. Biomed. Inform. 50, 205–212 (2014)
13. Sørensen, T. "A method of establishing groups of equal amplitude in plant sociology based on similarity of species and its application to analyses of the vegetation on Danish commons". Kongelige Danske Videnskabernes Selskab.1–34. ((1948).
14. Bachteler, T., Schnell, R., Reiher, J.: Privacy-preserving record linkage using Bloom filters. BMC Med. Inf. Decis. Making 9, 41 (2009).
15. Lazrig, I., Ong, T.C., Ray, I., Ray, I., Jiang, X., Vaidya, J.: Privacy preserving probabilistic record linkage without trusted third party. In: Proceedings of 16th Annual Conference on Privacy, Security and Trust (PST), pp. 1–10. IEEE, Piscataway, NJ (2018). https://doi.org/10.1109/PST.2018.8514192.

16. Xue,W., Vatsalan, D., Hu,W., Seneviratne, A.: Sequence data matching and beyond: New privacy-preserving primitives based on bloom filters. IEEE Trans. Inf. Forensics Secure 15, 2973–2987 (2020).
17. Alaggan, M.,Gambs, S., Kermarrec, A.: Blip: non-interactive differentially private similarity computation on bloom filters. In: Proceedings of Symposium on Self-Stabilizing Systems. Lecture Notes in Computer Science, pp. 202–216. Springer, Berlin (2012)
18. Meadows, C.: A more efficient cryptographic matchmaking protocol for use in the absence of a continuously available third party. In: Proceedings ofIEEE Symposium on Security and Privacy, pp. 134–134. IEEE, Piscataway, NJ (1986). https://doi/org/10.1109/SP.1986.10022
19. Churches, T., Christen, P.: Some methods for blindfolded record linkage. BMC Med. Inf. Decision. Making 4(9), 9 (2004).
20. Smith, D.: Secure pseudonymization for privacy-preserving probabilistic record linkage. J. Inf. Secur. Appl. 34(2), 271–279 (2017). https://doi.org/ 10.1016/j.jisa.2017.01.002
21. Schnell, R.: Precise and fast cryptanalysis for bloom filter-based privacy preserving record linkage. IEEE Trans. Knowledge. Data Eng. 33(11), 2164–2177 (2019).
22. Ranbaduge, T., Vatsalan, D., Ding, M.: Privacy-preserving deep learning-based record linkage. In: Proceedings of 2021 IEEE International Conference on Big Data, pp. 1308–1316. IEEE, Piscataway, NJ (2022).
23. Ertl, O.: ProbMinHash—A class of locality-sensitive hash algorithms for the (probability) Jaccard similarity. IEEE Trans. Knowl. Data Eng. 34(7) (2022).
24. Jaccard, P.: Lois de distribution florale dans la zone alpine. Bull. Soc.Vaudoise Sci. Nat. 38(144), 69–130 (1902)

25. Bernstein, D.J., Duif, N., Lange, T., Schwabe, P., Yang, B.-Y.: High-speedhigh-security signatures. J. Cryptogr. Eng. 2, 77–89 (2012). https://doi.org/10.1007/s13389-012-002
26. Ertl, O.: SuperMinHash–A new Minwise hashing algorithm for Jaccard similarity estimation, arXiv:1706.05698 (2017)
27. Christen, P., Ranbaduge, T., Vatsalan,D., Schnell, R.: Precise and fast cryptanalysis for bloom filter-based privacy-preserving record linkage. IEEE Trans. Knowl. Data Eng. 31(11), 2164–2177 (2019).
28. Kosub, S.: A note on the triangle inequality for the Jaccard distance. Pattern Recognit. Lett. 120, 36–38 (2019).
29. Basha, S.J., Veesam, V.S., Ammannamma, T., Navudu, S., Subrahmanyam, M.: Security Enhancement of Digital Signatures for Blockchain using EdDSA Algorithm. In: Proceedings of 2021 Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV), pp. 274– 278. IEEE, Piscataway, NJ (2021).
30. Broder, A.Z.: On the resemblance and containment of documents. In: Proceedings of the Compression Complexity Sequences, pp. 21–29. IEEE, Piscataway, NJ (1997.
31. Yu, Y., Weber, G.: HyperMinHash: MinHash in LogLog Space. IEEE Trans. Knowl. Data Eng. 34(1), 328–339 (2022).
32. Ji, J., Li, J., Yan, S., Tian, Q., Zhang, B.: Min-Max Hash for Jaccard Similarity. In: Proceedings of 2013 IEEE 13th International Conference on Data Mining, pp. 301–309. IEEE, Piscataway, NJ (2013).
33. Karakasidis, A., Verykios, V.S.: A highly efficient and secure multidimensional blocking approach for private record linkage. In: Proceedings of 2012 IEEE 24th International

Conference on Tools with Artificial Intelligence, pp. 428–435. IEEE, Piscataway, NJ (2012).

34. Antoni, M., Schnell, R.: The past, present and future of the German record linkage center (grlc). Jahrb. fur Natl. Stat. 239(2), 319–331 (2019).
35. Dudkowski, D. et al. “A Prototype for In-Network Management in NaaS-Enabled Networks." *12th IFIP/IEEE International Symposium on Integrated Network Management (IM 2011) and Workshops* (2011): 8188. Print
36. Kshira Sagar Sahoo et al. "DSSDN: Demand-Supply Based Load Balancing in Software-Defined Wide-Area Networks." *International journal of network management* 28 (2018): e2022. Print.
37. Comstar Technologies Launches Secure Access Service Edge (SASE) Through New Partnership With Cato Networks. *PR Newswire* (2021): PR Newswire, 2021. Print.
38. Wang, Chong Nan, Zong Tao Wang, and Gang Yuan. Application Demand Analysis of Message-Oriented Middleware with Pub/Sub Model on TT&C Computer. *Applied Mechanics and Materials* 599 601 (2014): 1403. Print.
39. Thomas Satish, Sluss James. Fraud detection through data sharing using privacy-preserving record linkage, digital signature (EdDSA), and the MinHash technique: Detect fraud using privacy preserving record links. *The Journal of Engineering DOI: 10.1049/tje2.12341*
40. S. Ioffe, Improved Consistent Sampling, Weighted Minhash and L1 Sketching, *2010 IEEE International Conference on Data Mining*, Sydney, NSW, Australia, 2010, pp. 246-255, doi: 10.1109/ICDM.2010.80.

41. S. Ioffe, "Improved Consistent Sampling, Weighted Minhash and L1 Sketching," *2010 IEEE International Conference on Data Mining*, Sydney, NSW, Australia, 2010, pp. 246-255, doi: 10.1109/ICDM.2010.80.