7-27-2015 12:00 AM

# Counteracting Bloom Filter Encoding Techniques for Private Record Linkage

Vasundhara Sharma
*The University of Western Ontario*

Supervisor
Dr. Aleksander Essex
*The University of Western Ontario*

Graduate Program in Electrical and Computer Engineering
A thesis submitted in partial fulfillment of the requirements for the degree in Master of Engineering Science

# COUNTERACTING BLOOM FILTER ENCODING TECHNIQUES FOR PRIVATE RECORD LINKAGE

(Thesis format: Monograph)

by

Vasundhara Sharma

Graduate Program in Electrical and Computer Engineering

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Engineering Science

The School of Graduate and Postdoctoral Studies
The University of Western Ontario
London, Ontario, Canada

# Abstract

$\mathscr{R}$ecord Linkage is a process of combining records representing same entity spread across multiple and different data sources, primarily for data analytics. Traditionally, this could be performed with comparing personal identifiers present in data (e.g., given name, surname, social security number etc.). However, sharing information across databases maintained by disparate organizations leads to exchange of personal information pertaining to an individual. In practice, various statutory regulations and policies prohibit the disclosure of such identifiers. Private record linkage (PRL) techniques have been implemented to execute record linkage without disclosing any information about other dissimilar records.

Various techniques have been proposed to implement PRL, including cryptographically secure multi-party computational protocols. However, these protocols have been debated over the scalability factors as they are computationally extensive by nature. Bloom filter encoding (BFE) for private record linkage has become a topic of recent interest in the medical informatics community due to their versatility and ability to match records *approximately* in a manner that is (ostensibly) privacy-preserving. It also has the advantage of computing matches directly in plaintext space making them much faster than their secure mutli-party computation counterparts. The trouble with BFEs lies in their security guarantees: by their very nature BFEs leak information to assist in the matching process. Despite this known shortcoming, BFEs continue to be studied in the context of new heuristically designed countermeasures to address known attacks.

A new class of set-intersection attack is proposed in this thesis which re-examines the security of BFEs by conducting experiments, demonstrating an inverse relationship between security and accuracy.

With real-world deployment of BFEs in the health information sector approaching, the results from this work will generate renewed discussion around the security of BFEs as well as motivate research into new, more efficient multi-party protocols for private approximate matching.

**Keywords:** *Private Record Linkage, Bloom Filter Encoding, Heuristic Security, Approximate matching, Information privacy.*

# Acknowledgements

*First and foremost, this thesis would not have been made possible without the support and guidance of Dr. Aleksander Essex, Assistant Professor in the Department of Electrical and Computer Engineering at Western University and my graduate supervisor for the past two years. Dr. Aleks, I feel privileged to have had this opportunity to work with you. I am deeply thankful to you for all those countless hours you have spent on improvising my skills, tossing around ideas, reading through and improving my writing and steering me in the right direction. Thank you for your patience, time and effort you have given me over the past two years.*

*To my grandfather, Bindeshwari Prasad Sinha, and my grandmother, Shivdulari Sinha, who have moulded me into the person I am today. To my father, Raghavendra Sharma, and my mother, Anju Sinha, who have given me love, appreciation, and seemingly infinite support through all tides of my life. To my sister, Ankita Sharma, I thank you for your friendship, motivation, and love.*

*Extended family and friends: To my uncles, Braj Sinha and Mihir Jha and my high school buddy, Bhavya Jha, I sincerely dedicate this work to you, without your faith and motivation, I would not be where I am today.*

*I take this opportunity to express my sincere gratitude to the professors I have had the chance to study with: Dr. Abdelkader Ouda, Dr. Abdallah Shami and Dr. Jagath Samarabandu of the Department of Electrical and Computer Engineering at Western University for their help and guidance throughout this journey. Finally, I would like to thank all of the supportive staff members at Western University for their assistance in various ways.*

# Contents

# List of Figures

# List of Tables

# List of Abbreviations

| | |
|---|---|
| **BF** | *Bloom Filter* |
| **BFE** | *Bloom Filter Encoding* |
| **CIHR** | *Canadian Institutes of Health Research* |
| **CPA** | *Chosen Plaintext Attack* |
| **DoA** | *Degree of Anonymity* |
| **FDA** | *Food and Drug Administration* |
| **HIPAA** | *Health Insurance Portability and Accountability Act* |
| **HMAC** | *Keyed-hash Message Authentication Code* |
| **IND-CCA2** | *Indistinguishability under Adaptive Chosen Ciphertext Attack* |
| **IND-CPA** | *Indistinguishability under Chosen Plaintext Attack* |
| **IND-EAV** | *Indistinguishability under Eavesdropping Attack* |
| **NCVR** | *North Carolina Voter Registry* |
| **NIST** | *National Institute of Standards and Technology* |
| **NSA** | *National Security Agency* |
| **PII** | *Personal Identifiable Information* |
| **PPSC** | *Privacy Preserving String Comparators* |
| **PRF** | *Pseudo-random Function* |
| **PRL** | *Private Record Linkage* |
| **SHA-1** | *Secure Hash Algorithm-1* |
| **SIC** | *Set Intersection-based Cryptanalysis* |
| **SMC** | *Secure Multi-party Computation* |

# List of Notations

| | |
|---|---|
| A | *Party A, participant in PRL* |
| B | *Party B, participant in PRL* |
| $\mathcal{T}$ | *Semi-trusted third-party asssistant* |
| $\mathcal{A}$ | *Adversary, honest but curious* $\mathcal{T}$ |
| $C$ | *Encryption Oracle in CPA game* |
| $D_S$ | *Global dataset* |
| $D_A$ | *BFE list for party A* |
| $D_B$ | *BFE list for party B* |
| $D_{AB}$ | *Combined BFE list for parties A and B* |
| $k$ | *Total number of hash functions in BFE* |
| $m$ | *Size of Bloom filter in bits* |
| $o$ | *Maximum number of allowed bit overlaps in a BFE* |
| $f$ | *Probability of false positives in a Bloom filter* |
| $t$ | *Matching threshold value* |
| $\mathcal{K}$ | *Secret key for generating hash encodings* |
| $Dice_{bfe}$ | *Dice coefficient between two BFEs* |
| $Dice_{plain}$ | *Dice coefficient between two plaintext strings* |
| DD | *Dice Dictionary* |
| PS | *Possibility Set* |
| RP | *Revealed BFE-Plaintext set* |
| KP | *Known BFE-Plaintext set* |
| PL | *Processed BFE-Plaintext set* |

# List of Appendices

# Chapter 1

# Introduction

Record linkage can be defined as a task of identifying records from multiple data sources that refer to the same individual. Although the concept itself is not new, it has gained lot of attention in the recent times for data processing in distributed settings. Record linkage helps in resolving issues like data duplication or data omissions from the lack of adequate information. Record linkage applications are found in several domains due to their intrinsic nature to provide more detailed and accurate analysis of data. For example, in the medical domain, record linkage can be applied to patient records held by disparate health groups in order to identify records that refer to the same patient. The primary use case of record linkage in this context is to create a clear picture of the patient's details by connecting the disjoint pieces of information. Another use case is to enrich data quality for statistical analysis purposes by eliminating data redundancy.

Record linkage in its most basic form utilizes the personal identifiers of individuals for setting up the data correlations. Disclosure of personal identifiers, however, is not always feasible due to privacy regulations and restrictions. For example, data sharing policies such as the Health Insurance Portability and Accountability Act (HIPAA) prohibit direct disclosure of patient information in certain cases [32]. Beyond regulatory impediments, the presence of social concerns associated with sharing of personal information is also a restrictive factor in many settings.

In light of these concerns and restrictions, private record linkage (PRL) remains an impor-

tant tool for data analysis. PRL allows two data holders to learn which records they share in common, while (in theory) protecting information about records that are not shared in common. Various techniques have been proposed to implement PRL, including cryptographically secure multi-party computation protocols. However, these protocols have been debated over the scalability factors as they are computationally expensive by nature. Alternatively, data transformation or encoding techniques can be used. Bloom filter encoding (BFE) [88] is one such technique for PRL which has gained traction lately. BFE ostensibly offers to realize private data linking in timely manner by using tokenization and hashing of data values into a Bloom filter (BF). BFEs are also versatile as a matching methodology. They can capture not only exact matches, but also *approximate* matches—a feature desirable in applications such as health research where, for example, patient names might contain small misspellings or variations. As such, BFEs have become a topic of interest in the health data setting for its attempt to balancing privacy guarantees with computational efficiency. However, the underlying security assurance associated with BFEs has not received thorough evaluation.

In the following section, a brief description of ideal PRL functionality is presented and it is correlated with the BFE technique to arrive at the research motivation of this thesis.

## 1.1 Motivation

To prevent privacy breaches, PRL must perform linking such that each party learns no information other than the list of matched records, which we call the PRL *ideal functionality*. The abstract representation of the process is illustrated in Figure 1.1. Restricting this example to the medical domain, Hospital A and Hospital B are exchanging their patient information using a PRL technique. The *only* information which they should learn during the course of the protocol's execution is the final set of matched records. We say a PRL scheme is *secure*, or alternatively *private* if it can be shown to realize the ideal functionality.

There has been much recent interest in BFE based techniques in PRL applications because

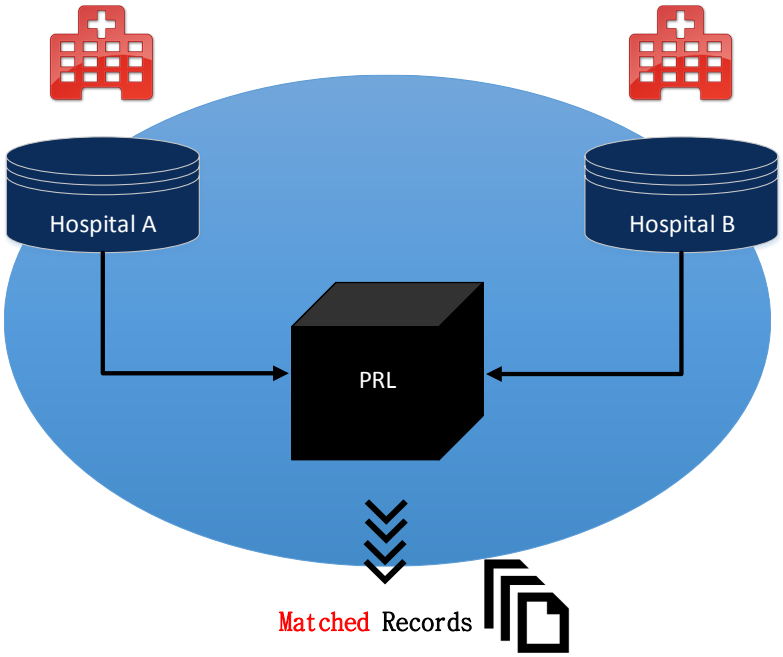



**Figure 1.1: Abstract representation of the ideal private record linkage (PRL) functionality**. In the medical domain context, Hospital A and Hospital B each input their respective patient records into the PRL functionality via a private channel, which outputs the matched records. Neither hospital receives any information from the protocol beyond what it learns from the list of matched records.

they can offer efficient solution for approximate record linkage which addresses scalability issues while handling large volumes of data. Recent work [64, 61] has shown, however, that Bloom filter encodings are susceptible to a range of cryptanalysis. On the other hand, employing a more robust security model, such as cryptographic secure multi-party computation (SMC) is often seen as impractical due to performance and scalability issues. Kuzu *et al.* [64] were among the first to attempt cryptanalysis of BFEs in a PRL setting, but wrote that "although SMC protocols provide strong security guarantees, they are impractical for many real data integration tasks due to their reliance on inefficient cryptography." Further, to counteract the success of their attack, they also propose inserting additional fields into the Bloom filter as a type of *chaff* (i.e., addition of worthless or irrelevant extra identifiers) to confound cryptanalysis. Recently, follow-up work [38] has focused on extending these chaffing approaches,

while in parallel, Toth *et al.* [93] have been focusing on a software implementation of such BFE-based PRL system.

An area that has not been well explored in literature is the security evaluation for BFE technique in PRL. Thus, the research gap identified here is two-fold:

1. **How secure are BFEs really?** Considering the existing literature, security analysis of BFEs has taken a heuristic approach: come up with an attack, come up with a countermeasure, show countermeasure prevents the attack, and repeat as necessary. However, the formal security analysis[1] is missing. In addition, the theoretical understanding that BFEs offer weak or no security guarantees has not been given sufficient attention. This leaves important open research questions: to what degree are BFEs secure? What mechanism allows BFEs to achieve that level of security? Are there trade-offs (such as accuracy) to increasing security?

2. **Do recently proposed countermeasures actually increase security?** BFEs are "distinguishable" by nature, and hence leak partial information by design. This means that they do not adhere to the basic expectations of an ideal PRL technique, and recent efforts have focused on chaffing as a method to increase security. The effectiveness of chaffing, however, is an open question.

As the interest in BFEs increases, it is important to bear in mind that social and regulatory expectations of privacy, security must not be compromised in the pursuit of efficiency, especially when the security mechanism underlying BFEs cannot be reduced to a hardness assumption. Fundamentally if BFEs do not offer any formal security guarantees, it is essential to at least quantify the risk to individual of their records being re-identified.

This thesis primarily focuses on addressing these concerns surrounding the usage of BFE technique for PRL. Although efficient SMC protocols for approximate matching have proven

[1]Formal security analysis provides an intuitive way of analyzing complex security protocols and access whether they satisfy the high-level properties as stated in the supported security definitions.

elusive, that should not justify of BFE as an alternative they offer effectively no security guarantees.

## 1.2 Thesis Contribution

The main contribution of this thesis is to revisit the security analysis of the Bloom filter encodings to understand precisely what privacy guarantees they offer in an attempt to quantify re-identification risk, especially in the context of recently proposed cryptanalysis countermeasures.

This section summarizes the main contributions.

1. **Security analysis.** Theoretical conceptualization of security weaknesses in the Bloom filter encoding techniques is presented in this thesis. Utilizing the game-based security analysis approach from the modern cryptography, we show that BFEs are distinguishable under chosen plaintext attacks (CPA), which is widely regarded as the minimal definition of security across the security research community. In this way, we attempt to show that BFEs remain below the set security standards.

2. **Set intersection-based cryptanalysis framework.** We propose a novel class of set intersection-based cryptanalysis (SIC) which attempts to recover patient identifiers from the corresponding BFEs. The SIC framework has been designed to operate in two distinct phases: (i) *the filtering phase* and (ii) *the trim phase* respectively. The design details for each of these two phases, captured in the form of algorithms, is presented in this thesis.

3. **Re-identification risk assessment for BFE-based PRL systems.** The effectiveness of the proposed SIC attack is measured in terms of re-identification risk by considering the degree of anonymity of a set of records following our attack. Although SIC is not the first cryptanalysis attempt on BFE technique, it is novel in its kind to establish and assess the following fundamental limitations associated with the security of BFEs:

- **Key independent**: BFE technique offers security which is ultimately key-independent (*i.e.,* increasing key length for the related hash function does not diminish the effectiveness of the attack). Hence, the role of keyed hash functions is this setting is limited.
- **Limited impact of countermeasures**: The Recently proposed countermeasures to BFE cryptanalysis (*e.g.,* [64, 65]) show only a minor quantitative improvement to a records anonymity set. The experiments conducted in the current work, however, show that by adding another identifier, the increase in the average anonymity set size is by less than 10%. This observation helps in breaking the assumption barriers such as the underlying security issues can be fixed with chaffing or randomizing input strings.
- **Security vs. accuracy**: We demonstrate BFEs exhibit a fundamental trade-off between security and accuracy. When controlling for accuracy, BFEs become much more susceptible to cryptanalysis and the effect of countermeasure is further minimized. Efficiency at the cost of quality does not serves the main purpose of linking data correctly, and in that sense BFEs suffer from an existential dilemma.
- **Risk estimation**: Re-identification risk cannot be established *a priori* to running the protocol. Hence, there can be no security guarantees with the use of BFEs in a PRL setting.

4. **Software implementation of SIC.** In this thesis, the testing and validation of the proposed attack model along with the re-identification risk assessment of BFE-based PRL system has been accomplished through the software implementation of SIC framework. The software for SIC framework has been developed in C++ programming language along with some pre-processing steps utilizing Perl scripting language for faster pattern matching and string manipulations. The software has been further modularized to accommodate parallel processing requirements during the experiment stage.

## 1.3 The Organization of the Thesis

The remainder of this thesis is organized as follows:

- **Chapter 2** outlines both the background information associated with the concept of private record linkage, and a literature review of the various approaches for implementing it. This chapter will first provide an introduction to common terminology that will be used throughout the rest of the thesis. Second, an overview of practical approaches for private record linkage will be explored, including an introduction to a variety of implementation techniques with their benefits and drawbacks. Finally, the chapter will review the concept of Bloom filter for its specific use in context of PRL and outline the associated security weaknesses.

- **Chapter 3** contains the main contribution for this thesis. First and foremost, this chapter presents the theoretical security analysis for the BFE technique using the security game approach. Next, this chapter will provide details for the components and framework of the proposed Set Intersection-based Cryptanalysis (SIC). As mentioned earlier, the SIC is categorized into two distinct phases, (i) *the filtering phase* and (ii) *the trim phase*. This chapter houses the details for the design and working for each of the two phases. Additionally, algorithms for each of the attack phases is described and presented.

- **Chapter 4** covers the details around experiment inception and the specific methodologies adopted. This chapter also provides the list of evaluation metrics which are used for evaluating the results of the SIC. The software implementation details and the choice of dataset is also covered in this chapter.

- **Chapter 5** is focused on the evaluation of the SIC implementation. Effectiveness of BFE is analyzed separately with each phase of the attack. In addition, an independent accuracy analysis is presented which measures the impact of hardening BFE construction on the overall match prediction. This chapter assists in validating the theoretical conceptions of

this thesis work.

- **Chapter 6** ultimately provides the conclusion for the thesis, primarily summarizing the accomplished work as well as a discussion on the some of the limitations and possible future work.

Below is the roadmap for the remaining figures (chapter-wise) present in this thesis (for reference purposes):

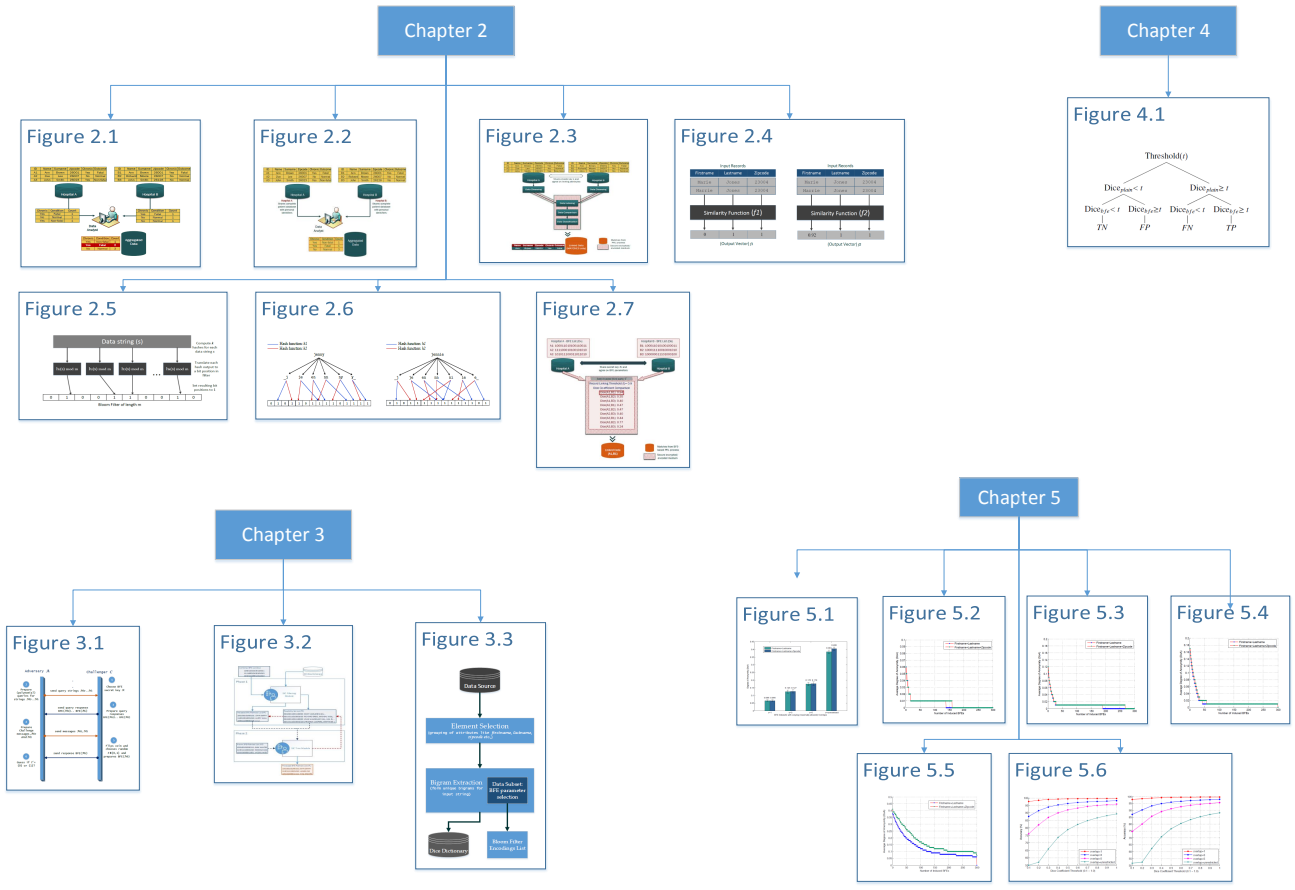



**Figure 1.2: Chapter-wise list of figures present in this thesis.**

# Chapter 2

# Background and Literature Review

This chapter details the key concepts involved with the proposed cryptanalysis work in Section 2.1. Section 2.2 presents an overview of the related research works for Private Record Linkage (PRL). The literature review specifically covers works related to data masking or encoding techniques utilized for PRL which are described in section 2.3, while also touching down upon the secure alternatives, which leverage cryptographic protocols for the data linkage. Section 2.4 details the concepts and usage of the Bloom filters and specifically its use as encoding techniques for implementing secure data linking, while identifying some research gaps in relation to the security associated with usage of this technique in PRL.

## 2.1 Concept Introduction

This section will introduce key concepts of **record linkage**, **private record linkage (PRL)** and the use of private record linkage in the medical domain. Next, the concept of approximate matching is discussed. Approximate matching is used for evaluating similarity between entities which is widely used in record linkage techniques. This section serves as a primer for the concepts which would be referenced throughout in this thesis work.

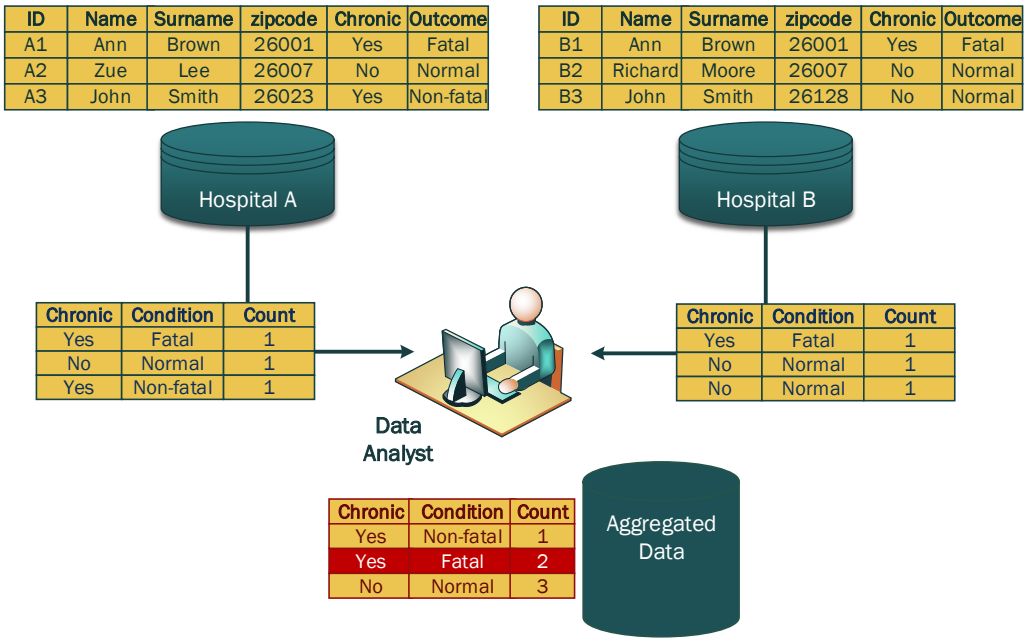



**Figure 2.1: Illustration of data overestimation in absence of record linkage.** Here Hospital A and hospital B exchange patient health statistics as opposed to individual records. In the absence of record linking, this can result in overestimation of data. As we see here the record [Chronic, Fatal] corresponds to the same individual, which is double counted in data aggregation stage.

### 2.1.1 Record Linkage

Record linkage can be defined as the process of identifying records from multiple data sources that refer to the same individual. The concept was initially conceived by Dunn [36] for aggregating statistical information from the life and death registries pertaining to the same individuals. Record linkage, as a process itself, is algorithmically straight forward and can be performed manually without the aid of computers. Record linkage has been widely adopted with computing techniques, however, due to the need for fast information linking across vast information databases which may span across multiple fragmented sources of information. While computer scientists frequently refer to record linkage as *"data matching"*, the other prominent synonyms in use are: *"record matching", "de-duplication", "entity linking","identity / record*

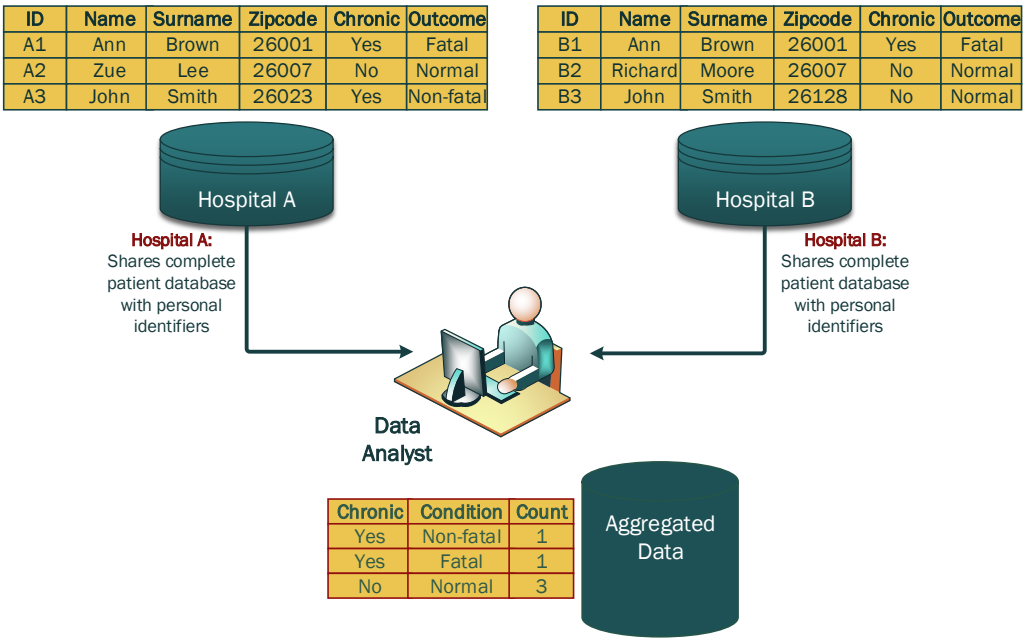



**Figure 2.2: The problem of data overestimation is corrected with the deployment of record linkage.** As illustrated here, Hospital A and hospital B exchange their patients health records with patient identifiers for data analysis. Records pertaining to the same individuals, originating from both hospital A and B are identified based their personal information (name, surname, zipcode combination) and the record [Chronic, Fatal] are identified and counted only once.

*/name resolution" etc.,* The primary benefit of record linkage is that it creates a precise picture of an individual by bringing together her information. Imagine that we do not have any record linkage technique in place while performing disease assessment in a health setup. Figure 2.1 shows that hospitals A and B are trying to estimate the total number of patients diagnosed with certain chronic conditions and their respective ailment stages. We notice that result aggregation based on counting total patients registered with a given hospital is resulting into double counting the same patient information since it originates from two different sources. Thus, in general, data redundancy can result in inaccurate disease assessments due to lack of record linkage techniques. As a solution, record linkage can be applied to patient records held by various health care providers in order to identify records that refer to the same patient.

Record linkage works on matching combinations of personal identifiers from the data elements. Revisiting our earlier health care setup example from Figure 2.1, if we combine patient identifiers like *givenname, surname, zipcode* along with the diagnosed chronic condition, then we would be able to identify the duplication of information for patient *Ann Brown* with fatal chronic condition. Figure 2.2 shows the revised output of data aggregation with record linkage technique in place. The record linkage process can be of two types:

- **Deterministic linkage.** Also known as *rule-based record linking*, deterministic linkage generates data links based on matching result of individual or combination of identifiers [85]. Two records would be identified as a match if the given set of identifiers are identical for both. Deterministic linking works on exact matching of strings based on the pre-defined rules. This type of linking method is good for datasets whose entities can be filtered out based on common identifiers and where the quality of data is high. This strategy would not work best for datasets with missing identifiers as it would lead to incorrect match outcomes. The cost of deterministic linkage is determined based on the cost of missed matches due to lack of error tolerance in exact matching functions. However, this can be improved to a certain extent by incorporating a multiple-step matching strategy which allows a progressive series of less restrictive steps in which if record pairs do not match in the first instance, are passed on to next rounds for further comparison. If a record pair meets the criteria in any step, it is classified as a match. Otherwise, it is classified as a non-match. The cost of such deterministic linkage scheme would then depend on the number of rounds designed in the match function.

- **Probabilistic linkage.** Probabilistic linking assigns varying weights to different identifiers based on the estimation of the ability of the identifier to correctly identify a match. These weights along with the identifiers are then used to calculate the probability of the matches and non-matches for a given threshold value. This type of linking is good for datasets which cannot be distinguished with a single unique identifier. However, calculating appropriate weights is simple in principle but sometimes difficult in practice. This

type of record linking is also referred to as *fuzzy matching* or *fuzzy merging*. The cost of probabilistic linkage is determined based on the relative cost of false matches and true but missed matches. This would vary depending upon the type of approximation function used in the implementation.

Record linking of any type does involves filtering and linking records based on unique identifiers which falls into the category of *personal identifiable informations* (PIIs). Thus, record linkage in its most basic form is not always a feasible solution due to privacy concerns and restrictions involved with the sharing of personal identifiers [94, 32].

### 2.1.2 Challenges Associated with Record Linkage

The entire process of data integration works on the conjunction of three tasks; *schema matching* [13], *record linkage* [51] and *data fusion* [74, 35, 17]. The first task (*schema matching*) is associated with identifying all the attributes in database entries which store equivalent information; the second task *(record linkage*) works on identifying the records which belong to the same entities across these different databases; and the third task (*data fusion*) works on merging together the identified records into a single record. All these tasks together complete the whole process of data integration. Record linkage is the most challenging task among all three and the associated challenges can be broadly categorized as:

- **Quality of data.** We often come across this fact that the real data-world data is perturbed in nature. There can be several issues like typographical errors, variations, alterations, missing data or data itself being obsolete (*e.g.,* due to legal name changes). Thus, even though the data corresponding to same entities are matched against their personal identifiable informations, it can lead to incorrect or ambiguous results. For instance, in one of the related works, Christen [23] discusses the potential source of variations and errors associated with the "personal name" identifiers. He also discusses the characteristics of such identifiers which distinguishes it from general text. Aided with empirical analysis,

ultimately, Christen highlights the fact that there can be no clear best matching technique in practice. Therefore, exact matching of attributes is not sufficient in the context of record linkage. Accurate classifiers and approximate matching techniques are more desirable for maintaining the accuracy and preserving the quality of data in record linkage [30]. We further discuss approximate matching techniques in Section 2.1.5.

- **Scalability.** Record linkage suffers from scalability issues in general. In a typical scenario, with the increasing size of databases, the total number of potential comparisons would be the product of the size of the respective databases. This can be a major performance hurdle as the comparison of record pairs can also involve complex similarity comparison functions. To overcome this problem, data indexing techniques can be deployed before the record comparison step. Christen [26] performs the comparative analysis of complexity, performance and scalability of 6 different indexing techniques. These techniques are similar in functionality which aim at creating blocks of data by removing the records which are obvious non-matches while retaining the pairs which are potential matches. Data indexing techniques help in reducing the total number of comparisons considerably as the total comparison space is now reduced to a single data block which is a small subset of the entire database. The order of reduction specifically varies across different implementations of the indexing techniques. It is determined based on the parameter settings like the matching threshold value (which represents the indexing key) used for placing records into the same block or the fixed size of the blocks in some cases.

- **Data privacy and confidentiality.** Database linking involves the use of PII of individuals, thus, protecting privacy is a crucial requirement. Databases can contain information that is highly sensitive such as the medical and financial details of individuals. For instance, in Figure 2.2, we see that the detailed information like chronic conditions of patients have been exchanged for record linkage along with their personal identifiers like name and zipcode which can ultimately disclose the actual identity of the person.

Apart from this, database linkage used for business collaborations may even reveal confidential information like business data (list of customer or suppliers etc.,) or financial conditions of individuals. Thus, protecting the confidentiality of information is equally important [96]. We discuss more specific privacy concerns and data re-identification risks associated with medical data in Section 2.1.3.

Durham *et al.* [39] attempt to evaluate the tradeoffs between maintaining data quality and scalability, while preserving privacy and confidentiality across six different privacy-preserving string comparators (PPSCs). They highlight the fact that the PPSCs which are efficient and yield higher accuracy of record linkage do not provide strong privacy guarantees and vice-versa.

### 2.1.3 Data Re-identification Risk in Context of Medical Data

Sharing unmasked data has its own benefits. It facilitates accurate analysis on data sets, aids research communities to perform validations to confirm results or simply promotes cost savings by eliminating the need for collecting same data by different research communities. Considering these requirements, there has been an urge to make such data, publicly available. For example, the Canadian Institutes of Health Research (CIHR) recently drafted a policy that requires some data to be made publicly accessible [5]. This policy is in effect as of January 1st 2008. Similarly, the UK Medical Research Council (MRC) policy on data sharing sets the expectation that data from their funded projects shall be rendered as publicly available [31]. However, it is worthwhile to note that disclosure of health data in such broad spectrum leads to privacy risks [22]. Furthermore, these risks are somewhat real and there has been several successful attempts of re-identifications of individuals. Thus, protecting the identity of individuals while releasing such sensitive health data is crucial and thus mandated by privacy policies. For instance the United States National Institutes of Health (NIH) permits sharing patient details while at the same time prevents disclosing personal identities of patients [95]. One approach to implement this process is to *anonymize data* before sharing it. El Emam and Dankar [42] assign

practical bounds for "$k$" in the data anonymization concept of *k-anonymity*, initially proposed by Sweeney [92] for preserving data privacy. They define the probability of re-identification, also known as the *re-identification risk* to a patient, as $\frac{1}{k}$ where $k$ is the cardinality of the associated anonymity set. The obligation of the data holder, therefore, is to define a risk threshold $\frac{1}{k}$ and in turn ensure that each patient's anonymity set is never less than $k$. Further, Pfitzman and Hansen [81] offer an appropriate definition of anonymity for the current setting: "Anonymity of a subject from an attacker's perspective means that the attacker cannot sufficiently identify the subject within a set of subjects, the *anonymity set*." Without loss of generality each record is associated with a patient identity, and thus the patient is anonymous within the set of all patient records.

**Anonymity Precedents in Health Data** El Emam [41] provide precedents for minimum allowable anonymity set sizes in a range of applications. In a highly trusted setting such as an internal (*i.e.,* within an organization) data release, $k \geq 3$ may be considered acceptable, whereas in a public (and adversarial) setting such as in patent data disclosure, $k \geq 20$ may be more appropriate. What is clear is that there is no one-size-fits-all rule. For example, the Colorado Department of Public Health and Environment [1] and the Iowa Department of Public Health [4] have guidelines amounting to $k \geq 4$. The National Center for Health Statistics [3] generally advises $k \geq 5$, though also has criteria [60] for $k \geq 4$ when considering certain diseases, and $k \geq 20$ in the context of birth and death statistics. El Emam, however, suggests $k \geq 5$ is the prevailing baseline for data disclosure in the health information setting.

### 2.1.4 Private Record Linkage

Considering the various scenarios from Sections 2.1.2 and 2.1.3, it is evident that record linkage in its basic form is a challenging task. The associated challenges can be attributed to problems like missing unique identifier information in databases or privacy concerns and legal restrictions imposed on sharing of an individual's personal information. Thus, the databases

from different organizations need to be linked together without jeopardizing the confidentiality of the individuals involved. The increasing need for linking large and disparate databases while preserving the privacy of the individuals whose records are stored in these databases has led to the development of a new methodology known as Private Record Linkage (PRL). The main requirement of any PRL system is that only limited information shall be released at the end of the data linkage step. This limited information can be accessed by either of the parties sharing their databases, or to any other party (such as an independent researcher or any data analytics organization etc.,) which requires this aggregated and linked data. PRL itself is a diverse notion and there has been several classes of proposals for the implementation of this technique. For instance, Lawati *et al.* [66] propose a potential solution for secure and efficient record linkage by dividing the entire database into separate blocks or clusters. They suggest that a blocking-aware private record linkage protocol can provide enhanced performance while preserving data security. Different records are combined into a single block based on some commonality with each other (*e.g.,* sharing some tokens). Each of these blocks is further associated with a unique identifier coupled with secure hash signatures to achieve confidentiality and compactness in the record linkage process. Bachteler *et al.* [10] focus on performing an empirical evaluation of three different PRL methods which compute string similarities in privacy preserving manner proposed by Schenell *et al.* [88], Scannapieco *et al.* [86], and by Pang and Hansen [79]. Inan *et al.* [52] propose a hybrid method for PRL which combines the data anonymization technique with secure cryptographic protocols. In another work, Yakout *et al.* [101] propose a method which works on transforming the database records into numeric vectors which is further used to generate complex numbers to represent the records. These numbered representations of records are then shared directly between the parties who compute similarity between these numbers to identify data matches. This method claims to have no requirement for a third-party while simultaneously performing efficient record linkage. Freedman *et al.* [48] explore various protocols which work on private two-party computation of set intersection between the datasets. They further propose homomorphic encryption and balanced allocation hashing

(uniformly distributed hashing across multiple bins) based PRL solution which they claim to be more efficient. Several encoding based solutions are also popular in the literature, such as, Weber *et al.* [98] use a blind-folded hashed data exchange method in which the input string is constructed by combining identifiers like first two letters of the first and last name along with date of birth information. They believe that such input combination would act as a blindfolded information thus, preserving data privacy while concurrently being sufficient for record linkage purposes. Schnell *et al.* [88] propose an encoding technique which employs Bloom filters [18] as the encoding infrastructure for implementing efficient PRL. This scheme is discussed in detail in Section 2.4. Thus, we see that PRL can be implemented using various different techniques. The various privacy techniques used to facilitate PRL has been discussed elaborately further in Section 2.3. In this section, we restrict our discussion to understanding PRL as a complete process in its entirety.

**Private record linkage process** An ideal PRL system should be capable of addressing all the three problems of *data quality, scalability and data privacy* which we earlier discussed in Section 2.1.2. Figure 2.3 outlines the detailed steps involved in private record linkage process. We briefly discuss the role of each of these steps:

- **Data cleansing.** This is the first step in the PRL process which is also known as *data pre-processing and standardization* step. This step is designed to eliminate the noise found in the real-world data due to typographic errors, missing information and presence of inconsistent data [11, 83]. This step also eliminates the unrelated extra information which is not required in the data linkage process [40, 50]. The outcome of this step is a set of well-defined and consistent dataset. While data pre-processing doesn't completely falls into the record linkage process entirely, it is considered to be part of record linkage process in a privacy-preserving setting. This is due to the fact that different data sources should perform symmetric pre-processing and also agree upon the approaches adopted as well as the common attributes which would be utilized further in the data linkage. Data

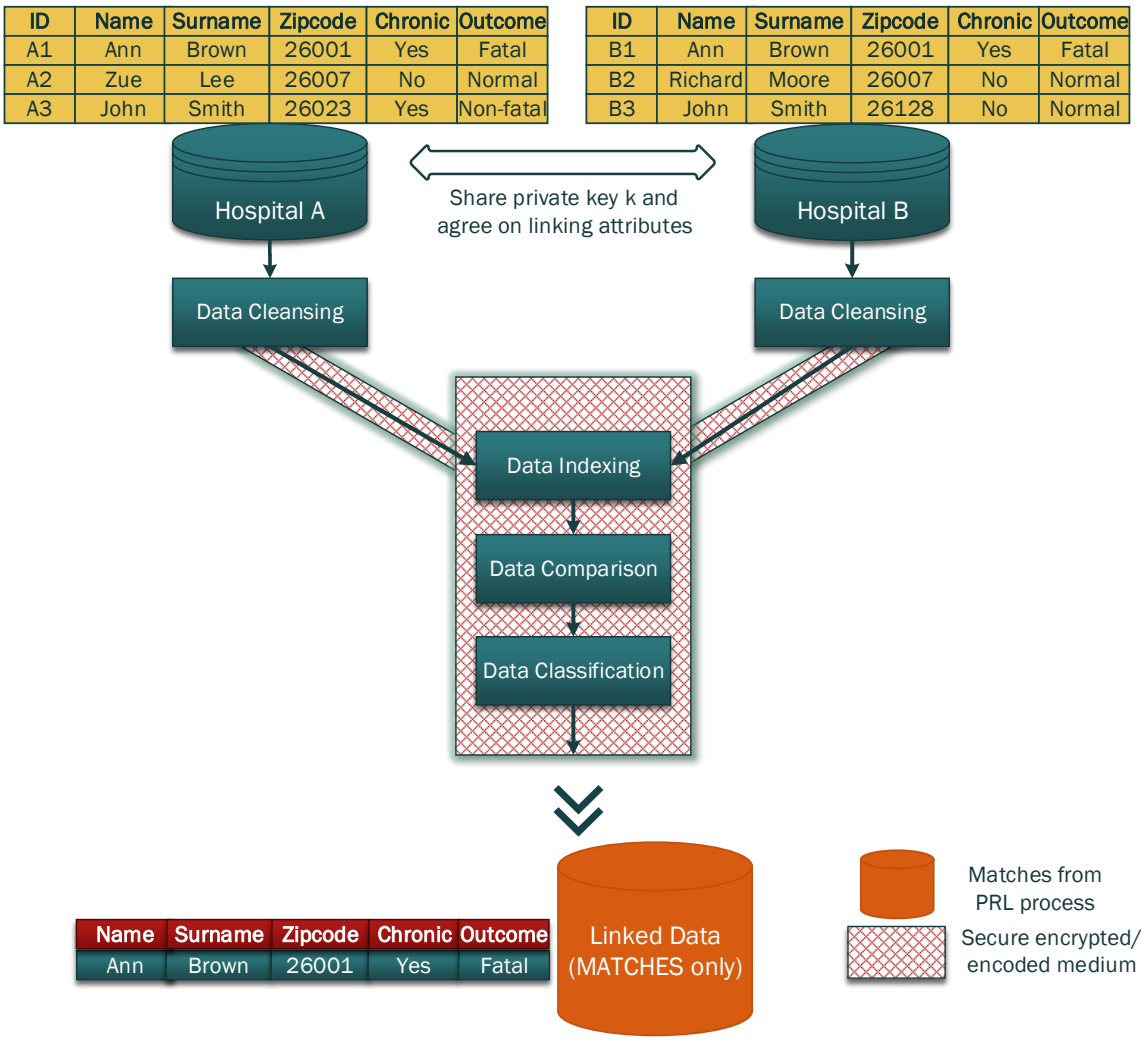



**Figure 2.3: Private record linkage process.** Details of steps involved in record linkage process in a privacy preserving setup; data linking, comparison and classification steps are performed in a secure encrypted/encoded environment and the resulting linked information reveals only the matched records.

pre-processing is performed independently at the respective data sources and the result is shared in a protected environment for further processing. A variety of methods can be adopted for data pre-processing. For instance, Churches *et al.* [28] suggest using lexical tokenisation in confluence with probabilistic hidden Markov models for sanitizing name and address data for record linkage. They state that this method also works well on data fields with complex formatting (like address etc.).

- **Indexing.** Data indexing is the second step in record linkage process which aims at providing efficiency to the entire process by reducing the total number of comparisons needed between two databases by removing the pairs of records which are unlikely to result in matches [12, 26]. In the absence of indexing techniques, if we had to compare two databases X and Y with $X_m$ and $Y_n$ records respectively, then we would have to perform $X_m \times Y_n$ comparisons in total. This is a major performance bottleneck owing to the growing sizes of databases and the record linkage comparisons involving expensive operations [26]. Hence, indexing techniques help to resolve this issue by filtering out the records which are unlikely to match. Several approaches can be used to perform data indexing. For instance, the entire database can be divided into clusters or blocks which can be tracked with a block key [66, 26]. In such clustering techniques, the total number of candidate pairs for performing comparisons are generated from one specific block only. Applying indexing techniques adds scalability and compactness to the record linking process. Lawati *et al.* [66] proposed a secure three-party blocking protocol which can deliver high performance record linkage using secure hash encoding with token blocking feature for data indexing.

- **Comparison.** The candidate record pairs generated after data cleansing and indexing step are compared against each other using different similarity measures. These comparisons can be performed at different levels. For instance, in record level comparisons, the complete set of data attributes are chained together and compared with its other counterparts. The other approach is to conduct comparisons at attribute level where individual attributes can be picked depending on the type and nature of the datasets. Further, the choice of similarity function widely depends on the nature of attributes in the records as well as on the context of record linkage. In a health care environment, approximate matching similarity function is often recommended as the data here is more likely to suffer from typographical errors and other variations. Thus, performing exact match on encrypted or encoded values will result in low quality of PRL [27, 76]. A separate expan-

sion of this discussion on approximate matching techniques is presented in Section 2.1.5. Exact matching can be performed only in scenarios where data is not overly inconsistent or noisy. The output of comparison step gives the similarity value between each pair of compared records. These numbers are further utilized in classification step to classify the records as matches and non-matches.

- **Classification.** Classification step utilizes the output from comparison step and based on the adopted method used in the record linkage setting, segregates the data into matches and non-matches. Record linkage classification techniques can be broadly classified as: threshold based, probabilistic, rule based and machine learning based. These classification techniques and the underlying methods are summarized in Table 2.1. In context of PRL, the classification step needs to ensure that no participating party learns any information about the other parties' records which do not match. The only allowed information to be revealed to either of the parties is the (exclusive) record pairs which have been classified as matches.

The resulting aggregated data (after the completion of the various pre-processing tasks) is deemed ready for use in applications or for research and statistical analysis purposes. Measuring the effectiveness, completeness and quality of linked data is an optional feature, which typically, doesn't fits into the scope of private record linkage process and can be performed independently and disjointly.

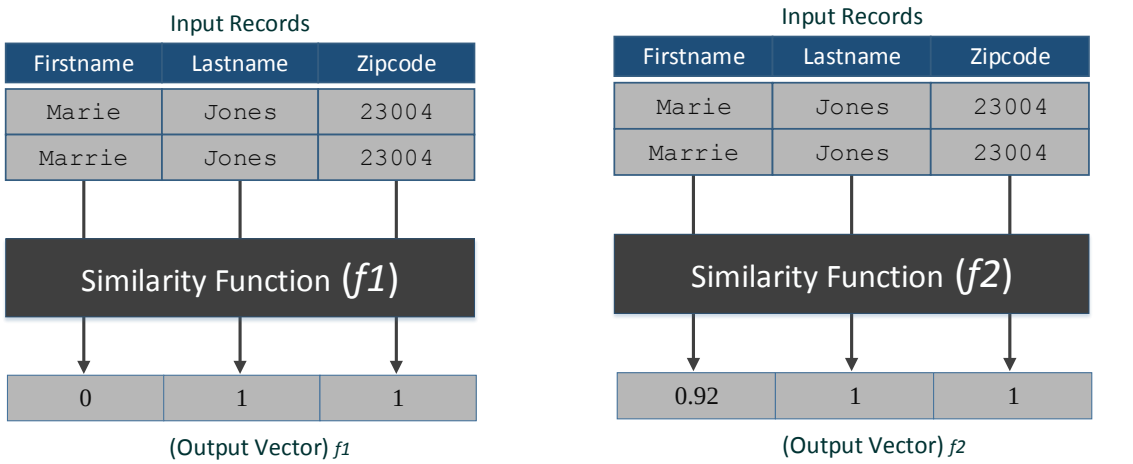
### 2.1.5 Approximate Matching

The standard form of record linkage makes use of *exact matching* similarity techniques, where two records are determined to be a match only if their corresponding identifiers are identical. In a health care setting, however, identifiers might not always match exactly (*e.g.,* due to transcription errors, or name variations, *etc.*) and an *approximate* matching strategy is more desirable. Records could be made up of different attributes (*e.g., firstname, lastname, street*

| Classification Technique | Method | Notes |
|---|---|---|
| Threshold based | Similarity value of records compared against threshold values. | Approximate agreement technique, threshold values can be experimentally determined [73, 88]. |
| Probabilistic | Conditional probability based classification on similarity measured through error estimates and frequency distribution of attributes. | Concept conceived by Fellegi and Sunter [47], other works on concept extension [25, 100]. |
| Rule based | Set of rules used for classification, rules generated based on combination of logical operations (AND, OR, NOT) on attributes. | Creating rules is time-consuming and may require manual intervention for generating and maintaining set of rules [29]. |
| Machine learning based | Supervised (trained decision model) and unsupervised (clustering) techniques used. | Popular supervised techniques: support vector machine and decision trees [16, 24]; Popular unsupervised technique: clustering based on comparison vectors [30, 68]. |

**Table 2.1: Summary of various classifications methods.**

and *zipcode*). For a given record, these attribute sets need to pass through the similarity comparison step of record linkage which quantifies the extent of similarity between the given two records. The similarity comparison step runs the records through a similarity function which computes the similarity between the two entities. Depending on the type of similarity function, its functionality and output patterns differ. For instance, if we are performing an exact match, the result would be binary (i.e., either 0 or 1) as illustrated in Figure 2.4a. A real world example of exact matching scheme would be the use of Hash Message Authentication Code (HMAC) [14] based encryption scheme for private record linkage. In this particular use case, the data holders agree upon a secret key and set of identifiers based on data fields present in the records of their databases. Both the parties exchange only the final hashed results of HMAC to a third party for performing the record linkage. The identifiers would result in exact match (with high probability) if their corresponding hash outputs are identical, so this allows the third




**(a)** Exact matching attribute comparison **(b)** Approximate matching attribute comparison

**Figure 2.4: Example of attribute level comparison techniques in record linkage.** The different techniques illustrated in this example are (a) exact matching similarity function and (b) approximate matching similarity function.

party to perform record linkage on the encrypted data in a secure manner. One limitation of such scheme is that slight variations in the input will result in a different output and hence won't match. Despite this limitation, exact matching schemes are still popular. OḰeefe *et al.* [76] and Berman *et al.* [15] propose similar flavors of exact matching techniques to be used in PRL. On the other hand, approximate matching techniques provide error tolerance as well as flexibility of choosing a desirable threshold based on the application itself or based on the contents of the databases. The similarity function in this scenario works on a similarity measure calculation between the two entities. In case of approximate matching, the result is a real number in range of [0,1]. This is further illustrated in Figure 2.4b. Together these two figures demonstrate the difference in the outcome of exact and approximate matching techniques. Approximate matching yields probabilistic record linkage [51, 56] where the exact agreement from all identifiers is not required. Agreements with higher weighted identifiers, out rule the disagreements in identifiers with lower weights. Chuches and Christen [27] proposed creating bigrams (or n-grams in general) from the identifiers to enable secure probabilistic record linking by calculating bigrams (*i.e.,* 2-grams) scores for the identifiers, without revealing the

actual data. This method combined with cryptography, introduced yet another new concept of blindfolded record linkage. Blindfolded record linkage is defined as the technique which allows to carry out statistical analysis on disparate datasets without any parties having to reveal identifying information about any of the subjects involved. Blindfolded record linkage makes use of secure one-way hash transformations along with approximate comparison techniques to achieve data privacy. A variety of similarity measuring techniques are available. A commonly used set-based similarity metric is due to Dice [34]. The *Dice coefficient* between two sets $X$ and $Y$ is defined as:

$$Dice(X, Y) = \frac{2|X \cap Y|}{|X| + |Y|}. \tag{2.1}$$

A matching function defined for a threshold $t$ is defined as

$$Match(X, Y, t) = \begin{cases} True & : Dice(X, Y) \geq t \\ False & : Dice(X, Y) < t \end{cases} \tag{2.2}$$

In case of string similarity measure, the Dice coefficient would work on the set of bigrams where |X| and |Y| would represent the character bigrams found in the respective strings X and Y and $|X \cap Y|$ is the total number of character bigrams common to both X and Y. Schnell *et al.* [88] utilize the Dice coefficient as the similarity metric for record linkage in a privacy preserving setup. Other alternative similarity measures existing in literature are: the Jaccard index, the Tversky index, the Levenshtein distance and the Overlap coefficient.

## 2.2 Applications of Private Record Linkage

Record Linkage prevents data duplication and aids in leveraging accuracy for data analysis, arising from data aggregation from multiple sources [51]. The use of private record linkage (PRL) is not restricted to just medical informatics. Discussion on several usage domains for PRL, both within and beyond the field of medicine is presented in this section.

### 2.2.1 PRL Inside of the Medical Domain

The decentralized nature of health care systems leads to storage of patient's medical information in fragmented form across multiple datasets. Further, these datasets are owned by different health care providers. PRL can enhance data analytics in several ways in such setups:

- **Removing data redundancy for research studies.** Sharing patient details is crucial in biomedical research sphere [86], however in the absence of record linkage techniques, statistical variables can be overestimated due to duplication of records or remain underestimated due to fragmentation of records. Consider the example shown in Figure 2.1, either of the situations (data overestimation or underestimation), gives rise to data redundancy which leads to incorrect results. Detecting duplication however, is complicated by privacy laws preventing the direct disclosure of patient information, for instance, the United States National Institutes of Health (NIH) permits sharing only such patient details which do not disclose the identity of individuals or correlation with others in any form [95]. In these scenarios, PRL can be applied to achieve de-identified data sharing to mitigate information bias, while still being able to detect and discount duplicates.

- **Participation monitoring in clinical trials.** Clinical trials are a form of study which is commonly used in the pharmaceutical industry to evaluate the safety and effectiveness of a drug. Eligible patients are hired to participate in the study. Often, patients are financially compensated for their participation in such studies. The same patient may be tempted to participate across different studies at the same time, which can lead to incorrect estimation or prediction of a particular drug behavior in addition to posing health risks to the subjects. Thus, it is important to ensure that one patient is enrolled in only one study at a time. Detecting such double enrollments, however, can be challenging as clinical trials are covered under privacy legislation like HIPAA [32] which prevent the information disclosure for such participants. PRL can be used in this scenario to securely and effectively detect and exclude participants from enrolling in multiple trials

concurrently [43].

- **Creating health indicators.** Health indicators are periodic measures which provide relevant indications and vital information about population health [54]. Health indicators are important to undertake suitable actions for improving health system performance. It is also useful in determining useful contextual information about community health. Creation of health indication relies upon fetching information from diverse set of data sources. For instance, Statistics Canada generates correlation between mortality and cancer outcomes for populations exposed to chemical hazards in industrial settings [20]. This process would involve referring to information from databases like Canadian Mortality Database (CMDB), Canadian Cancer Database (CCDB), Canadian Birth Database (CBDB) etc. Private record linkage (PRL) can be effectively deployed in such use cases to generate correct health indicators.

- **Health surveillance.** PRL can be directly utilized for strengthening health surveillance activities. This can be achieved by linking data corresponding to real-time monitoring of safety of drugs, biologicals and medical devices when they touchdown markets. For instance, the United States Food and Drug Administration (FDA) has already launched a project, the Sentinel Initiative [99], which aims at providing real-time monitoring of the safety of health products by securely querying several automated data holders for adverse usage metrics [7]. One of the prime objectives of this project is to keep track of the safety of the FDA-regulated products. This process can be leveraged to yield better results with the use of PRL for linking the informations fetched from multiple resources. Further, the other health surveillance aspects could be monitoring spread of diseases, epidemics control or vaccination information etc. El Emam *et al.* [44] present one such use case where PRL has been effectively deployed to evaluate the effectiveness of human papilloma virus (HPV) vaccination in Canada. This was accomplished using private record linkage (using secure multi-party computation) to securely aggregate information

from different disease registries.

- **Enhancing resource allocation.** Cost of medical care can be reduced with the access to complete knowledge of patient information which is often scattered across multiple datasets. For instance, if a care provider can determine that certain examinations have already been rendered to a patient through a different health provider, then the cost of current treatment gets minimized. Unilateral information set also helps in proper delegation of resources to other patients by eliminating replication of services. Involving PRL for improvising resource allocation of health facilities can be useful since we do not have a uniform standard to identify patients shared across different providers. Proposals for drafting such uniform identifying standards in health systems [21] has been rejected by United States Department of Health and Human Services on grounds of confidentiality issues. Thus, a complete picture of a patient's medical information, aided by private record linkage, has the potential to retain patient privacy and information confidentiality while decreasing the cost of medical care.

### 2.2.2 PRL Outside of the Medical Domain

Though PRL methods prove to be quite useful in bridging disparities within the health system, their applications are not restricted to a single domain. PRL can be further utilized for:

- **Law enforcement.** PRL can be applied to gather information about suspects from different data sources which are held by independent intelligence agencies which otherwise cannot share their information databases freely. Intelligence agencies can make use of PRL techniques to combine these informations pertaining to suspects over a secure channel. In addition, this information can be further used to setup correlation with the activities of other suspects in the same network [63]. Recent advancements in law enforcement systems like Privacy-Protective Surveillance [77] are capable of detecting and retrieving suspicious activities on the Internet or in transactional databases. Once these activities

are detected, the PIIs associated with them is encrypted for security reasons. Next, it undergoes record linking using secure multi-party computation techniques to allow for the interrogation and inspection of the encrypted data. Using probabilistic analysis, this information can be utilized to evaluate terror threat possibility based on the evidences revealed from the linked information.

- **Counter-terrorism surveillance.** Video surveillance is a commonly used technique for monitoring day-to-day operations on public spots or in places with restricted access. However, if these techniques are deployed in specific vigilance settings, they get surrounded with confidentiality and privacy concerns. For instance, secure face identification systems are used to compare an input face to a protected list of known suspects. However, the database of known suspects cannot be shared publicly as it contains highly confidential information. Recent development of PRL alternatives like SCiFi [78] provide secure way of identifying facial patterns while both protecting the privacy of subjects as well as the confidentiality of suspect databases. Homomorphic encryption techniques are deployed to securely compare the subject's facial information for a match against database entries.

## 2.3 Implementation Techniques for Private Record Linkage

A variety of PRL protocols and implementation techniques have been proposed to allow respect for data confidentiality in the record linkage sphere. These protocols can be broadly classified as a third-party dependent or a third-party independent protocols. Either of these make use of similarity computations through two types of models: (1) secure multi-party computation (SMC) and (2) data transformations (encodings). A separate discussion, covering the literature on the existing techniques for both the models is presented in the following sub-sections.

### 2.3.1 Secure Multi-party Computation Methods

SMC utilizes strong cryptographic protocols with provable security guarantees in order to implement end-to-end secure computation where the involved parties cannot get access to any other information about the data except the final output. The idea for SMC was initiated by Yao [102], who first proposed a secure solution for a two-party computation problem. His work was later extended by Goldreich *et al.* [49] with the development of a generic framework with the extensibility to accommodate multiple parties. The different types of encryption schemes utilized for secure computation are:

1. **Homomorphic encryption.** Encryption schemes which allows certain algebraic computations (e.g., computation of distance similarity metrics, like Levenshtein distance) to be carried out on ciphertexts[1], resulting into encrypted results which when decrypted by the end users, matches the result of operations performed on the plaintexts [59]. Homomorphic encryptions, therefore, provide a mechanism of chaining different inputs from a multi-party setting without revealing the data to each other. A practical example would be unpadded RSA[2] scheme where the multiplication of two primes represents homomorphism. Different operations can be performed based on the context of usage of the homomorphic encryption scheme. For instance, Agarwal *et al.* [8] discuss about a homomorphic encryption variant which allows multiple parties to encrypt messages using different keys in a cascade of operations. Decryption process uses the same set of keys but in any arbitrary order. Homomorphic encryptions are considered malleable by design and hence are well adopted for protecting privacy in cloud computing environments.

2. **Garbled Circuits.** Initially conceived by Yao [102] to provide a method for secure computation in a semi-honest setting where several parties wish to compute a function such that no party learns the inputs of any other. The input function is transformed into a

[1] Encrypted plain messages are referred to as Ciphertext

[2] RSA is one of the first as well as widely adopted practical public-key cryptosystems used for secure data transmission.

boolean logical circuit and encrypted using 2 random keys for each input. Each of the input wires is held by different parties. The randomly permuted (*i.e.,* "garbled") encrypted truth table is then exchanged. Both the parties run oblivious transfer protocol to learn the key information which is used to evaluate the output of the garbled logic gate. This encryption scheme, therefore, prevents sharing of any other intermediate information except the final output.

Some of the most widely used SMC techniques in context of secure record linkage are: secure sum, secure set union, secure set intersection and secure scalar product [74, 96]. Atallah *et al.* [9] propose an SMC-based sequence comparator protocol. They state that the requirement for determining the similarity between sequences without sharing the actual sequence data is found across numerous applications especially in bioinformatics (e.g., for performing DNA sequence comparisons by two different companies). This can be accomplished using a secure method to evaluate the edit distance between the different sequences utilizing homomorphic encryptions. Similiarly, homomorphic encryption based SMC solution has been deployed for law enforcement surveillance activities [78], while taking care of individual's privacy in place. SMC techniques are computationally intensive in nature and hence suffer from the scalability issues with larger datasets. Thus, this is a major limitation associated with the performance of the *secure yet slow* SMC techniques. In another work, Lindell and Pinkas [67] conducted a detailed survey on the SMC computation paradigms. They also discuss and highlight the efficiency limitations involved with the use of SMC techniques which eventually becomes a challenge in the construction of highly efficient SMC protocols.

### 2.3.2 Data Transformation Methods

Data transformations or encoding methods follow more relaxed security approach contrary to their SMC counterparts. Data transformation methods may selectively reveal some information to make the process more efficient. Hernandez and Stolfo [50] state that the most basic data transformation technique converts each data value to a corresponding encoding (exact match-

ing) neglecting the typographical (e.g., "marrie" vs "marie") or semantic errors (e.g., maiden vs married names). However, several subclasses of data transformation techniques have evolved with error tolerance capability (support approximate matching). Some of the prominent data transformation techniques are:

1. **Secure hash encoding.** Secure hash encoding has been one of the first data transformation techniques to be used for PRL [96]. These methods make use of cryptographic (one-way) hash functions based on secure hash algorithms (like SHA-1 and HMAC) [87]. The data is transformed into a hash code (for instance, a string *'jennie'* is converted to *'512de34a86e7e1'*) which is used to perform the matches against the hash-codes of the other party involved in PRL. A keyed-hash approach like Hashed Message authentication Code (HMAC) [14] can be further utilized to strengthen the security of the system. Although keyed-hash approach safeguards the system from dictionary attacks to some extent, these approaches are still susceptible to frequency based attacks. Another major limitation here is that these methods only support exact matching. Therefore, this method would not work well for linking datasets with typographical errors and name variations.

2. **Phonetic encoding.** A typical phonetic algorithm functions by indexing the "similar sounding" words based on their pronunciations. Phonetic encoding algorithms are widely used in spell checkers and search functions for approximate matching. Phonetic encoding methods in conjunction with secure hashing has been utilized in context of PRL. For instance, Quantin *et al.* [82] propose transforming identifiers into respective phonetic encodings and then encrypting these with (one-way) secure hash function with random padding. These hash values are then exchanged with the third-party where they get hashed again with another pad. Ultimately, the third-party performs exact matching on the resulting hash values. Despite of using exact matching technique, this method permits some degree of error tolerance as the exact match is performed on the phonetic encodings of the identifiers. The pros of phonetic encoding methods are scalability (as less number of comparisons are required) and support for approximate matching (allows

error tolerance) [23]. However, Rogers and Willett [84] state that phonetic encoding methods are inferior in quality than the other string similarity comparator functions due to the fact that they tend to yield higher false positives.

3. **Embedded space.** This encoding technique works on building an embedded space from the indexed strings of the attribute values. The two data holders map their respective strings (composed from data attributes) into the vector space using the SparseMap method [101]. The embedded strings are then sent to the third party which determines the similarity between them by computing the standard Euclidean distance. Mapping of records into vector space is utilized to achieve matching efficiency. Scannapieco *et al.* [86] extend this concept of embedding records to achieve data privacy in addition to efficient matching. However, their experiment results indicate that the linkage quality can significantly suffer with the use for greedy resampling heuristic of SparseMap to reduce the number of reference sets. In another related work by Pang and Hansen [79], a similar approach is proposed, which generates a set of reference strings common to both parties. In this method, both the data holders compute the distances between each identifier string and set of all reference strings. If the distance is less than a stipulated threshold value then the respective reference string is encrypted and stored in a database tuple along with distance information. Both the parties then exchange these encrypted tuples with the third-party. The third-party then computes the distances between the encrypted strings to identify matches. This method however, relies on the set of reference strings. There can be considerable performance degradation if the reference strings are not a superset of the original strings.

4. **Differential privacy.** Differential privacy is an emerging data transformation technique which aims to provide high accuracy of database queries while retaining the privacy of the queried records. This technique can be considered as a substitute for the generalization techniques where the perturbed databases are exchanged between the parties. In dif-

ferential privacy techniques, the involved parties can interact with each other's databases using statistical queries (like sum, average, count queries etc.,). These queries are then mapped to $d$-dimensional attribute space in the form of hyper-rectangles. These statistically transformed data is then exchanged with the other parties for record linkage analysis. Inan *et al.* [53] propose one such solution for private record linkage. However, they also state that their proposed method remained at par with its $k$-anonymity counterparts in terms of computational costs involved in data matching. However, Differential privacy method does provides substantial amount of information security if we can discount the performance issues under given circumstances.

5. **Bloom filter encoding.** This encoding technique makes use of the Bloom filter data structure [18] which maps all input strings into a random sequence of 0's and 1's, where the presence of the element is detected from the 1's in the filter. Data linking is performed based on the similarity of the Bloom filter encodings of the respective records. This data transformation technique and its utilization in PRL is the prime subject of this thesis work and hence an elaborate discussion on the topic is presented in Section 2.4.

## 2.4 Bloom Filter Encodings

A Bloom filter is a simple and space-efficient probabilistic data structure, first envisioned by Burton Howard Bloom [18]. Bloom filters were originally designed as a quick way to test if a record existed on a (slower) storage device [19] attributed to its capability of performing look up operations in $O(1)$ time. Bloom filter's inbuilt feature of space optimization has found its use in several applications. This section presents a discussion on the properties, applications and its recent advancement in context of private record linkage. In Section 2.4.1, the design and properties of Bloom Filters is described. Various application domains of Bloom filters is further explored in Section 2.4.2. Finally, in Section 2.4.3, we discuss the application of Bloom filters for generating encoded data for private record linkage, which is the prime topic

of interest in this thesis work.

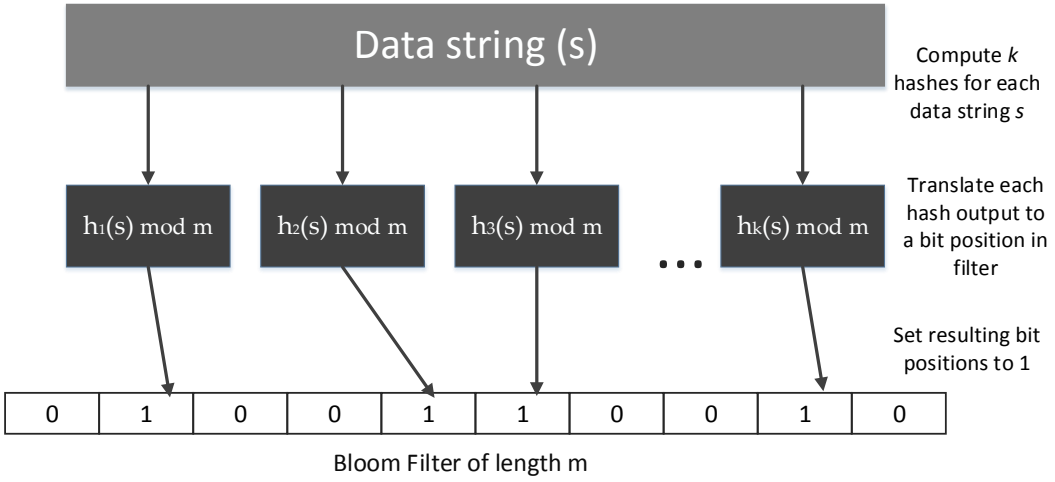



**Figure 2.5: Structure of the Bloom filter data structure.** This figure illustrates the steps involved in element insertion into a Bloom filter of length m. The input string element s (single input string considered for the sake of example) is first masked using $k$ hash functions and reduced to a position in the filter which is set with 1 to represent the presence of the element.

### 2.4.1 Overview of the Bloom Filters

Bloom filters [18] are a simple space-efficient probabilistic data structure for which the common operations such as insert or look up takes $O(1)$ time. Bloom filters were originally designed to filter queries made to storage devices with high latency. In the basic design, the Bloom filter queries carry a non-zero probability of false positives and will return either *"Possibly in Set"* or *"Not in Set"* for the queried string element. Once the elements are inserted into the Bloom filter, they cannot be removed or re-positioned inside the filter. Therefore, Bloom filters carry a zero chance of false negatives. Counting filter, proposed later by Fan *et al.* [46] is a data structure which allows delete operations on the Bloom filter without reconstructing it. Each element inserted into the filter is treated as a counter. Each time the element is inserted, the respective counters are incremented, and similarly decremented on each removal. The upper limit of the counter size is arithmetically restricted. Counting filters, however, are not scalable beyond a certain extent as the counter size is not expandable beyond the design

capacity. Cuckoo filters are yet another latest addition in the literature by Fan *et al.* [45] which allows dynamic re-arrangement of elements inside the filter. Bloom filters which cannot accommodate or track the recurrent addition of same elements into the filter, suffer from the problem of false positives since the the Bloom filter queries carry a non-zero probability of false-positives. Further, the probability of false positives increases with the increase in number of elements been inserted into the filter. Despite this shortcoming, Bloom filters have been deployed in various application which demand higher efficiency and scalability. Although the space efficiency is achieved at the cost of certain amount of accuracy, it has been ignored as a convenient trade-off in most of its applications.

Figure 2.5 illustrates the steps involved with the basic functionality of Bloom filters. A Bloom filter consists of a bit vector of length $m$, initially all set to 0 and a set of $k$ hash functions $h_i$ for $1 \leq i \leq k$ with the following functionality:

- **Initialize**: Initialize an $m$-bit filter to all 0's and select $k$ uniform[3] hash functions.
- **Insert**: Given an element $s$, for each of the $k$ hash functions $h_i$, $h_i(s)$ is the corresponding hash output. Set the $(h_i(s) \bmod m)^{th}$ bit position to 1. If that bit was already set to 1 before, no change is made.
- **Query**: Given an element $s$, if the $(h_i(s) \bmod m)^{th}$ bit position is set to 1 for all $1 \leq i \leq k$, return *True*, otherwise return *False*.

In the query stage, it is impossible to distinguish if the all bits set to 1 for a given element $s$ is a result of $s$ previously having been inserted into the filter, or by bits getting coincidentally set, during the insertion of other elements (*i.e.,* false positives). This is due to the design limitation of the Bloom filter to not keep track of or prevent the successive addition of elements into the same index in the filter. Further in a generalized scenario, the given input element $s$ can be a set $S$ which consists of $n$ different elements. Each of these $n$ elements, needs to be inserted into the bloom filter in accordance with the aforementioned functionality. It is obvious that the probability of false positives depends on the bit array length $m$, the number of hash functions

[3] $k$ hash functions uniformly chosen at random from the set of possible hash functions.

$k$, and the total number of elements $n$ in the given input set $S$. Therefore, Mitzenmacher and Upfal [70] define the probability of false positives $f$ in a Bloom filter as:

$$f = \left(1 - e^{-kn/m}\right)^k \tag{2.3}$$

Equation (2.3) is derived based on the assumption that each of the $k$ hash functions can select any bit position in the filter with equal likelihood as stated by Mitzenmacher and Upfal [70]. Based on this assumption, the probability that a bit is not set in a $m$-bit Bloom filter by any of the $k$ hash functions for $n$ different element insertions is expressed as $\left(1 - 1/m\right)^{kn}$. Therefore, the probability that the bit would be set would be $1 - \left(1 - 1/m\right)^{kn}$. Next, testing the presence of elements against $k$ different hash functions, the probability of element to be set would be $\left(1 - [1 - 1/m]^{kn}\right)^k \approx \left(1 - e^{-kn/m}\right)^k$.

In terms of space, the Bloom filter is a constant, as there is never a need to use more memory than the designated bit vector.

### 2.4.2 Bloom Filter Applications

For many years, Bloom filters[4] continued to be used mostly for dictionary and database applications [71], where they were used to represent sets for performing membership queries. Recently, they have gained traction in several other application domains including networking applications and record linkage in a secure environment. This section outlines a general discussion on the several application domains in which Bloom filters have been put to use.

1. **Dictionary applications.** Bloom [18] designed Bloom filters for keeping a dictionary of words which need extra rule-based processing. False positives, in this case, would be just an extra processing overhead for simple words which in normal course would not require extra processing. Another application of Bloom filter was in UNIX spell-checkers as described in the work of Mullin *et al.* [72] where the bloom filter representation of word

[4]Bloom filters were first introduced in 1970

dictionary was stored for allowing faster lookups. Although a false positive in this case resulted in ignoring a mis-spelled word, the space savings provided by Bloom filters outweighed it, as memory was a scarce and valuable resource in early 1990's. Spafford [91] proposed using Bloom filters for storing dictionary of unsuitable passwords for enhancing system security. False positives here would trouble users to some extent by rejecting some instances of appropriate passwords as well.

2. **Database applications.** Database applications utilized Bloom filters for optimizing semi-join[5] operations. In their work, J. K. Mullin [71] state that Bloom filters can also be used to estimate the size of semi-join operations because it can estimate intersection operations. Bloom filters were also used for tracking database changes (for instance, read write operations) by using them as differential files. False positives in these use cases would just add an extra lookup overhead on the database application.

3. **Networking applications.** Broder *et al.* [19] present a detailed survey on the applications of bloom filter in communication networking domain. Based on this survey report, Bloom filters have been widely used in networking applications like distributed caching, web search applications, peer-to-peer or overlay networks, resource routing, packet routing and traffic management infrastructure. For instance, Kubiatowicz *et al.* [62] utilize Bloom filters in overlay network topology where each node (typically representing a router) would store an array of Bloom filters representing the every possible adjacent edge. Thus, this method would save expensive route lookups for calculating the shortest path. Any $d^{th}$ position set in the bloom filter would keep track of nodes reachable by $d$ hops through the network along a given edge. In another work, Navendu *et al.* [55] propose applying Bloom filter based similarity detection technique for effectively removing similar web search results, thus enriching user experience. Bloom filters have also been used for resolving quick network loops and for routing multicast traffic. Yet another useful application of Bloom filters has been for efficiently measuring and record-

[5]semi-join operations merge two different relational sets based on common attribute names found in both.

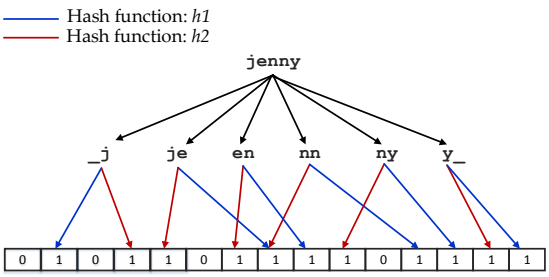
ing the network traffic statistics, especially for the nodes serving heavy traffic on usual basis. Bloom filters have been utilized across numerous networking applications since year 2000.

4. **Private record linkage.** As discussed earlier in Section 2.3.2, a second class of PRL implementation, utilizes weaker forms of security based on data transformation. These transformed values, commonly known as *encodings* are used as an input to PRL for data linkage. More recently Schnell *et al.* [88] proposed using Bloom filters in the context of private record linkage. Since this thesis work specifically focuses on the Bloom filter's use case in private record linkage domain, we present a separate detailed discussion on the topic in next Section 2.4.3.

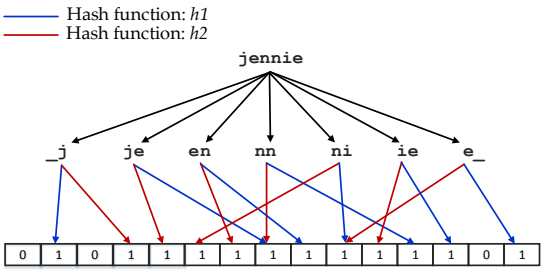
This section demonstrated various ways of extending Bloom filter data structure into different application domains. Bloom filters also seem to be an excellent alternative for space management. However, the drawback of false positives needs clear understanding and evaluation based on the specific application. It is critical to determine whether the impact of false positives is acceptable on the functionality of the application. However, we also see that there is plenty of room for the development of variants or extensions of Bloom filter in future [45, 46].

### 2.4.3 Bloom Filter Encodings in PRL

Continuing over to the previous discussion from Section 2.4.2, Bloom filters have recently gained traction in the context of private record linkage. In this context, Bloom filters are utilized for providing time-efficient information hiding (encoding). Schnell *et al.* [88, 89] were the first to introduce the notion for Bloom filter encodings (BFEs) for PRL. A recent evaluation work by Durham *et al.* [39] further states that a data transformation method for PRL based on Bloom filter encodings is superior to other approaches (e.g., unique hashed [76] or obfuscated identifiers [15], SMC-based equijoins [8] and SMC based sequence comparisons [9]) in terms of efficiency and accuracy.




**(a)** The Bloom filter encoding for firstname attribute with value "jenny"




**(b)** The Bloom filter encoding for firstname attribute with value "jennie"

**Figure 2.6: Bloom filter encoding based PRL.** In this example, Bloom filter length ($m$) is set to 15 bits, with 2 hash functions, $h1$ and $h2$. The input strings are split into 2-grams (bigrams) and resulting bigrams are mapped into the respective Bloom filters. The total number of common identical (intersecting) bits in both filters=10, total number of bits set in the filter for "jenny"=11 and for "jennie"=12, which yields a dice similarity as $\frac{2 \cdot 10}{11+12} = 0.87$. If the deciding threshold is $< 0.87$, then the result is a *match* otherwise its a *non-match*. (*cf.* Durham [37])

The strategy is to obfuscate a string's bigram set by inserting bigrams into a Bloom filter for which the hash functions are a secret. Figure 2.6 illustrates the process of generating Bloom filter encodings for a given string. The Bloom filter encoding function is defined as follows:

**Encoding Function** $BFE(s, m, k, \mathcal{K})$:

1. Initialize an $m$-bit Bloom filter $B_s$, and use secret $\mathcal{K}$ to construct $k$ independent hash functions (e.g., using an HMAC construction)
2. Decompose $s$ into $n$ unique bigrams $b_i$ $\forall\{1 \leq i \leq n\}$. Insert each bigram $b_i$ into $B_s$
3. Return $B_s$

Insertion and lookup operations for any element $s$ which has $n$ unique bigrams can be completed in $O(n)$ operations while delete operation is not permitted in a typical Bloom filter.

The output from the above encoding function is used as an input to the BFE-based PRL systems. The requirement for designing $k$ different and independent hash functions is a practical restriction for a large value of $k$. This can be tackled by passing $k$ different initial values (such as 0, 1, 2, ..., $k-1$) to the same hash function, or by appending these values to the secret key $\mathcal{K}$, in order to generate different hashed output each time. A keyed-hash input is specifically utilized in a PRL setting which is generated by appending the input strings with the secret key

$\mathcal{K}$ (which is a shared secret between the parties involved in the PRL). This is incorporated to protect the confidentiality of the generated BFEs.

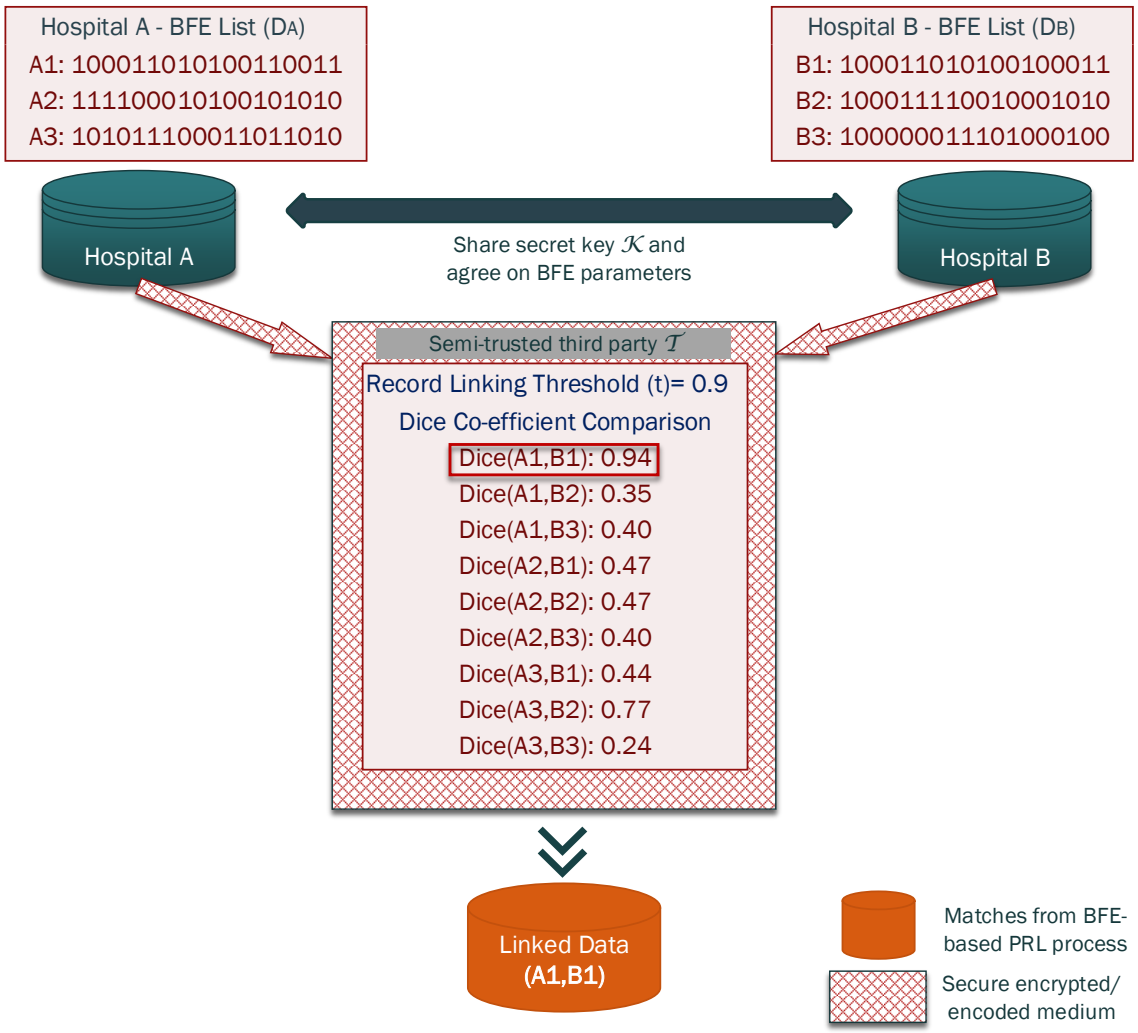



**Figure 2.7: BFE-based private record linkage process.** As illustrated in this example, Hospital A and hospital B negotiate a secret key $\mathcal{K}$ and other BFE parameters using which they prepare their respective patient records in Bloom filter encoded format. These BFEs are then exchanged with the third party $\mathcal{T}$ which calculates the dice coefficient similarity between each pair of encoded values. Dice values are compared against a given threshold value $t$ to predict matches/non-matches. The encoded records identified as matches are ultimately revealed to both A and B.

Figure 2.7 illustrates the steps involved in a BFE-based PRL system. The idea is for two data holders $A$ and $B$ to first negotiate the BFE parameters like the length of the Bloom filter

$m$, the type and number of hash functions $k$, and the shared secret $\mathcal{K}$. Once these parameters are agreed upon, they convert their respective patient records into BFEs and communicate the encoded values privately to a semi-trusted assistant $\mathcal{T}$. $\mathcal{T}$ then computes $Dice_{bfe}$[6] for all record pairs and compares it against an heuristically chosen threshold value $t$ for measuring approximate similarity. The final result is processed as matches or non-matches which is reported back to $A$ and $B$. Henceforth, for the sake of clarity, $BFE(s, m, k, \mathcal{K})$ is read as $BFE(s)$. The Bloom filter encoding based PRL protocol is defined as follows:

**BFE Protocol** for Private Record Linkage:

- **Initialize**: Generate a $p$-bit shared secret $\mathcal{K}$, initialize match threshold $t$ and the Bloom filter length $m$.
- **Generate BFEs**: For each of its records $r_i$, party $A$ computes $b_i = BFE(r_i)$ and communicates each such $b_i$ to $\mathcal{T}$ over a private channel. Party $B$ does the same for all of its records $r_j$, such that $b_j = BFE(r_j)$.
- **Compare**: For every pair of $b_i$ and $b_j$, $\mathcal{T}$ computes $Match(b_i, b_j, t)$ and reports any $b_i$'s that were found to be a match to party $A$ (via a private channel). $\mathcal{T}$ similarly reports any such $b_j$'s to party $B$. The match function is defined as follows:

  **Match function** $Match(b_i, b_j, t)$:

  1. Compute the Dice similarity $Dice_{b_i,b_j}$ between encodings $b_i$ and $b_j$ using equation (2.1).
  2. Compare $Dice_{b_i,b_j}$ against an heuristically chosen threshold value $t$.
  3. If $Dice_{b_i,b_j} \geq t$ return True else return False.

BFEs, however, are vulnerable to cryptanalysis. Kuzu *et al.* [64] were among the first to cryptanalyze basic Bloom filter-based records matching using constraint satisfaction techniques. To counteract the success of their attack they proposed inserting additional fields (*i.e.,* zipcode, telephone number, *etc.*) into the Bloom filter as *chaff* (similar in spirit to the radar

[6] $Dice_{bfe}$ denotes the Dice coefficient as stated in equation (2.1), evaluated for any two BFEs.

confusing countermeasure). Recently, follow-up work [65, 38] has focused on improvements to the chaffing approaches used, and include a software implementation [93]. However, in parallel, Kroll and Steinmetzer [61] and Niedermeyer *et al.* [75] got success with mounting another set of cryptanalysis on BFEs. As a countermeasure, they variously suggest hardening BFE construction using salting, removing longer names from attribute sets, injecting random bits, *etc*.

Ultimately, however, the security analysis of BFEs has mostly taken a *heuristic* approach: come up with an attack, come up with a countermeasure, show countermeasure prevents the attack, and repeat as necessary. Thus, we identify this approach as a research gap especially if we view Bloom filter encodings from a *formal* security perspective, they do not meet the basic formal security notion that ciphertexts be indistinguishable from each other. In addition, devising the correct ratio between the total number of $k$ hash functions and the actual length $m$ of Bloom filter has been completely ignored in the literature. This however has a direct impact on the accuracy of PRL system. This is due to the limitation of Bloom filters to re-arrange or remove bits inside the filter, which eventually results in higher false positives. This thesis presents a new set of cryptanalysis attack on BFE, which attempts to cover these research gaps. The main aim of this thesis work is to confront claims that BFE hardening of any form can fix the underlying security issues.

## 2.5 Summary

This chapter introduced the main concept of record linkage with the focus on its requirement for consolidating fragmented and scattered datasets. A detailed discussion on the shortcomings of record linkage is presented including the views of several related academic works. Further, the concept and methodology of private record linkage is introduced with progression into, how it tries to address the problems of quality, scalability and confidentiality of sensitive data been used in data linking process. A detailed literature coverage of the various domains in

which PRL is been put to use, both inside and outside of medical domain has been outlined. PRL is a diverse notion and its implementation can be classified into different and distinct classes. Discussion on the several implementation approaches used in PRL technique along with the presentation of the related academic works is covered. Finally, the concept of Bloom filter and its specific use in context of PRL is discussed, along with a touch-down on the security concerns associated with the method. The identified security issues and the research gaps surrounding the usage of the Bloom filters in context of PRLs is addressed with the main contribution of this thesis work in Chapter 3.

# Chapter 3

# Set Intersection-based Cryptanalysis

A new class of Set Intersection-based Cryptanalysis (SIC) framework is proposed in this thesis work. This cryptanalysis works specifically on the data transformation technique, the Bloom filter encodings, used for record linkage in a privacy preserving setup. The proposed attack works on the identification of some of the basic yet intuitive security flaws in the design of the Bloom filters which get adversely exposed due to its usage in a PRL setting. First, this chapter presents the theoretical security analysis for the Bloom filter encoding technique using the chosen plaintext attack game methodology. This chapter outlines the underlying security weaknesses which serves as the foundation on which the SIC model is derived. Next the framework for SIC is discussed along with the underlying design parameters which are introduced as well as explained in this chapter.

## 3.1 Security Analysis

In this section, Bloom filter encodings are examined from a *formal* security perspective which ensures that the procedures being analyzed, adhere to the properties as stated in the corresponding security definitions. The analysis shows that they do not meet the basic formal security notion that encryptions (or encodings in the current context) be indistinguishable from each other.

### 3.1.1 Threat Model

Prior to examining the security of Bloom filter encodings, it is important to note that they leak information about the underlying plaintext "by design." This is evident from the fact that the $Dice_{bfe}$ (*i.e.,* the Dice coefficient similarity metric between any two BFEs as expressed in equation (2.1)) is designed to be an approximation for $Dice_{plain}$ (*i.e.,* the Dice coefficient similarity metric between any two plaintexts) which is used for determining the matches in a PRL setting, as discussed earlier in Section 2.4.3. Therefore, it is important to settle on a *reasonable* security model for this setting. In an Internet setting, for example, security against *indistinguishability under adaptive chosen ciphertext attack* (abbreviated IND-CCA2) is considered critically necessary, especially in light of recent padding oracle attacks (*cf.* Vaudenay [97]). In an honest-but-curious multi-party computation setting, however, the weaker security notion of *indistinguishability under chosen plaintext attack* (abbreviated IND-CPA) is generally considered adequate. In general, the security research community regards chosen plaintext security as the *minimally* acceptable notion of security (*cf.* Katz and Lindell [57]).

In this thesis, the theoretical security analysis is presented in the form of chosen plaintext attack game which is described in the next section. For the sake of security analysis, the Bloom filter encoding function, $BFE(\cdot)$ is modeled as an encryption scheme in order to better compare it with other secure multi-party protocols. For instance, Freedman *et al.* [48] provide a homomorphic encryption based secure multi-party computation technique, which they prove to be IND-CPA secure.

### 3.1.2 Game-based Security Analysis

Indistinguishability under chosen plaintext attack (IND-CPA) is interpreted by the following game between an adversary $\mathcal{A}$ and a challenger $C$ (or the encryption oracle). This CPA experiment, or the security game is named as $\mathsf{Priv}^{\mathsf{cpa}}_{\mathcal{A},\mathsf{BFE}}$ for Bloom filter encoding scheme $BFE(\cdot)$. Figure 3.1 illustrates the steps involved in the chosen plaintext attack game. The game is played in the following steps:

1. **Step 1**: The challenger $C$, generates a secret key which is used to define $k$ random hash functions for a $b$-bit Bloom filter encoding scheme $BFE(\cdot)$ based on the given security parameter $p$.
2. **Step 2**: The adversary $\mathcal{A}$ can submit any (polynomially bounded[1]) number of query strings $\mathcal{M}_i$ to $C$.
3. **Step 3**: $C$ replies back with the associated Bloom filter encoding $B_i = BFE(\mathcal{M}_i)$ for each query it receives.
4. **Step 4**: Eventually, $\mathcal{A}$ chooses and submits two distinct challenge strings $\mathcal{M}_0 \neq \mathcal{M}_1$ of equivalent length.
5. **Step 5**: Challenger $C$ now flips a coin and selects a uniformly random bit $t \in_R \{0, 1\}$ and computes the Bloom filter encoding $B_t = BFE(\mathcal{M}_t)$ and returns it to $\mathcal{A}$.
6. **Step 6**: The adversary outputs a guess $t' \in \{0, 1\}$ to determine if it received back the encoding BFE($\mathcal{M}_0$) or BFE($\mathcal{M}_1$). The experiment is deemed successful if this guess is correct, i.e., if $t' = t$. The game outputs $\mathsf{Priv}^{\mathsf{cpa}}_{\mathcal{A},\mathsf{BFE}} = 1$ if the guess made by the $\mathcal{A}$ is correct, and outputs 0 otherwise.

Adopting the IND-CPA game-based security analysis as described above, the semantic security of Bloom filter encodings is defined as:

**Definition 1 Semantic Security of BFEs.** *Bloom filter encoding function $BFE(\cdot)$ is distinguishable under chosen plaintext attack (i.e., is* not *semantically secure) if the adversary $\mathcal{A}$ has a non-negligible advantage of winning the CPA game. Specifically, we say the scheme is semantically insecure (alternatively, distinguishable under chosen plaintext attack) if:*

$$P\Big[\mathsf{Priv}^{\mathsf{cpa}}_{\mathcal{A},\mathsf{BFE}}(p) = 1\Big] \geq \Big|\frac{1}{2} + \mathsf{negl}(p)\Big| \tag{3.1}$$

*where* $\mathsf{negl}(p)$ *is a negligible function in the security parameter.*

[1] i.e., a number that grows polynomially in the security parameter.

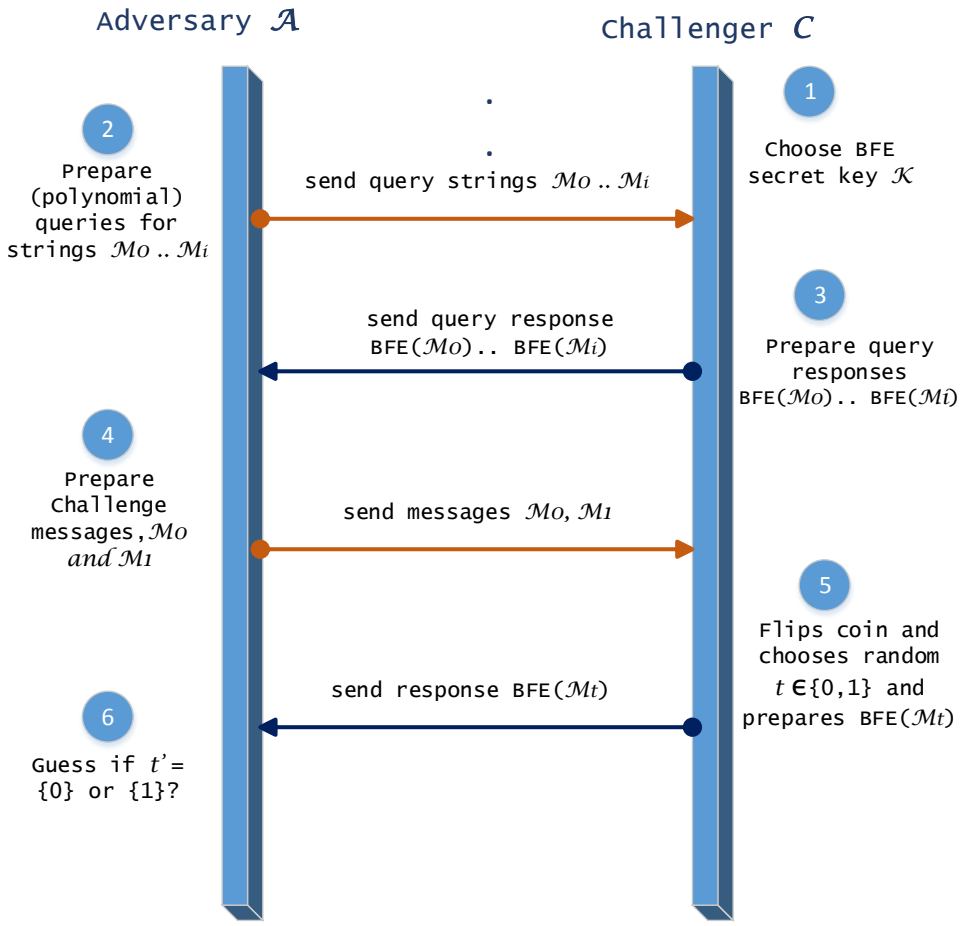



**Figure 3.1: Illustration of steps in chosen plaintext attack (CPA) game.**

On the basis of the CPA game and equation (3.1), it can be stated that the Bloom filter encodings are distinguishable under chosen plaintext attack. Further, a proof sketch is provided below to justify this claim.

**Theorem 3.1.1** *Bloom filter encodings are distinguishable under chosen plaintext attack (i.e., not semantically secure).*

***Proof Sketch.*** The Bloom filter encoding function $BFE(s)$ is fully deterministic *i.e.,* successive calls to the encoding function, $BFE(\cdot)$ with identical input always yield identical output. Now, $\mathcal{A}$ selects two strings of equivalent length during the query phase of the string, such that

$\mathcal{M}_0 \neq \mathcal{M}_1$, and receives $B_0 = BFE(\mathcal{M}_0)$ and $B_1 = BFE(\mathcal{M}_1)$. $\mathcal{A}$ then submits the same strings $\mathcal{M}_0, \mathcal{M}_1$ as challenge strings. The challenger $C$ selects a random bit $t$, encodes the message $\mathcal{M}_t$ such that $B_t = BFE(\mathcal{M}_t)$ and sends $B_t$ back to $\mathcal{A}$. Since $\mathcal{A}$ already has $B_0$ and $B_1$ from the query stage, it returns $t' = 0$ if $B_0 = B_t$, and returns $t' = 1$ otherwise. Because $BFE(s)$ is deterministic, the probability of making correct guess is 1 in either case. Therefore, the probability of winning the game is always,

$$P\Big[\mathsf{Priv}^{\mathsf{cpa}}_{\mathcal{A},\mathsf{BFE}}(p) = 1\Big] = 1. \tag{3.2}$$

since $P[B_t = B_0 \,|\, t = 0] = 1$ and $P[B_t = B_1 \,|\, t = 1] = 1$.
Further, the advantage of any adversary $\mathcal{A}$ playing the game is defined as the probability that $\mathcal{A}$ wins minus the probability that $\mathcal{A}$ loses. Formulating this in an equation we get:

$$\mathsf{Adv}^{\mathsf{cpa}}_{\mathcal{A},\mathsf{BFE}} = \Big|P(\mathcal{A}_{wins}) - P(\mathcal{A}_{loses})\Big| \tag{3.3}$$

From equation (3.2), we know that $P(\mathcal{A}_{wins}) = 1$, so Adversary $\mathcal{A}$ will always have a non-negligible advantage of winning the game. Also, it is worth noticing that the success probability of the chosen plaintext game is independent of security parameter $p$ when the encryption scheme is the Bloom filter encoding, $BFE(\cdot)$.

■ ▮

### 3.1.3 Discussion

Ciphertext indistinguishability is a desirable property of many encryption schemes. Intuitively, indistinguishability imparts confidence in any cryptosystem that the encrypted messages would not reveal additional information to the adversary. Game-based security analysis, however, can be restrictive in certain use cases. Therefore, it is important to thoroughly study the properties of the encryption scheme before attempting to establish game-based security proofs. For instance, for analyzing the security of Advanced Encryption Standard (AES) encryption scheme,

it is important to consider the mode of operation as well. For example, with AES in electronic codebook (ECB) mode, identical plaintext blocks are encrypted into identical ciphertext blocks, hence, it is not IND-CPA secure. The other modes of operation like cipher block chaining (CBC), counter (CTR) or output feedback (OFB) modes make use of random initialization vector (IV) which yields different ciphertext blocks for the identical plaintext blocks. The cihertext blocks in these use cases would be indistinguishable and hence IND-CPA secure. However, this is not applicable in the analysis of BFEs which do not use IVs, and hence are "really" deterministic in all circumstances.

## 3.2 Overview: Set Intersection-based Attack

This section presents a discussion on the motivation behind the inception of the proposed Set intersection-based cryptanalysis (SIC) framework. Next, definitions for few terms are presented which would be useful in better understanding of the attack model. SIC operates in two distinct phases. The overview of the attack model which outlines the work flow of SIC with both the phases combined is presented in this section.

### 3.2.1 Motivation

The motivation for SIC is derived from the inherent property of Bloom filters to have zero chance of false negatives (refer Section 2.4). This allows constructing set intersections and eliminating what *cannot* be in a particular BFE. False positives, on the other hand are a problem, and not only to the cryptanalyst, but to the data holders as well. The more number of elements inserted in the Bloom filter, the higher the probability of a false positive. Kuzu *et al.* [64, 65] effectively exploit this property by "chaffing" the filter with additional (superfluous) identifiers to enhance security. Our hypothesis, in contrary, is that adding extra identifiers would not greatly increase anonymity set sizes. Set intersection-based cryptanalysis is therefore, a step towards validating this hypothesis. Furthermore, the formal security analysis of

BFEs presented earlier in Section 3.1 indicate that BFEs are susceptible to known plaintext attacks, since they are fairly "distinguishable". The poor semantic security offered by BFE technique is exploited in the proposed SIC framework to highlight the fragility of the system backed by quantified results. The internal working of the attack framework is thus attributed to these practical security loopholes present in the Bloom filter encoding technique.

### 3.2.2 Concept Definitions

Few terminologies which are used frequently in the SIC model are defined here.

- **Anonymity sets.** Anonymity, for a single entity, defines its state of being not identifiable within a set of all possible subjects. This set of subjects represents the anonymity set.

- **Overlaps.** Overlaps represents the condition when the bit position which is attempted to be set in a Bloom filter has been already set previously. Traditionally, Bloom filter does not have the capability to handle overlaps. In this thesis, we present a new approach to control the maximum number of overlaps in a Bloom filter by adjusting the filter length. This concept is later discussed in Section 4.2.

- **Plaintext.** In Cryptography, plaintext refers to the state of data before applying any form of encryption on it. In the context of SIC framework, plaintext refers to the set of identifier attributes of a given record, which is then transformed into the respective Bloom filter encoding.

- **Ciphertext.** In Cryptography, ciphertext usually refers to the result of the encryption performed on a plaintext using any encryption or encoding techniques. Ciphertext contains the plaintext information in a form which is unreadable without the external aid of decryption. In the context of SIC framework, ciphertext refers to the Bloom filter encodings of the records.

### 3.2.3 Attack Model

The proposed SIC is designed to work in a typical PRL setup — wherein two parties $A$ and $B$ participating in the record linkage process, exchange their data using a privacy preserving secure channel with the trusted third-party $\mathcal{T}$. The secure channel here is the Bloom filter encoding. The attack model maintains the following standard assumption that Adversary $\mathcal{A}$ is the honest-but-curious[2] $\mathcal{T}$. $\mathcal{A}$ has access to publicly available informations, such as, the number of hash functions $k$, BFE length $m$ and number of overlaps allowed in Bloom filter $o$ (refer Section 4.2). Another research assumption is that $\mathcal{A}$ has access to a global data list (*e.g.,* via a phone book or voter list) and can assume each BFE corresponds to "someone" in that list. These are the standard research assumptions which has also been followed in the other related work by Kuzu *et al.* [64].

Following the PRL protocol as explained in Section 2.4.3, parties $A$ and $B$ agree upon the data attributes, BFE construction parameters and a secret key $\mathcal{K}$. Both the parties split the data attributes into unique bigrams and generate their respective BFE datasets $D_A$ and $D_B$ using the previously negotiated secret key $\mathcal{K}$. The BFE datasets are then handed over to the trusted third party $\mathcal{T}$ for record linking. For simulating the attack environment, here the trusted third-party $\mathcal{T}$, simultaneously represents the adversary, $\mathcal{A}$. $\mathcal{A}$ now constructs a joint list $D_{AB}$ by combining all the unique BFEs from the obtained datasets $D_A$ and $D_B$. $\mathcal{A}$ has access to the related public resource $D_S$ (say the voters list which acts as the global dataset) and safely assumes that $D_{AB} \subset D_S$. $\mathcal{A}$ now mounts the SIC attack on $D_{AB}$ and attempts to recover the plaintext records. It is worthwhile to reiterate here that the adversary $\mathcal{A}$ does not have any information about the secret key $\mathcal{K}$ used for generating the BFEs. In parallel, $\mathcal{A}$ generates a reference list called Dice Dictionary ($DD$) using the global dataset $D_S$ which holds the count of unique bigrams and dice similarity information (refer equation (2.1)) for pair-wise records from the dataset $D_S$.

[2]Honest-but-curious parties are semi-honest, that is, they run the protocol exactly as specified (no deviations, malicious or otherwise), but may try to learn as much as possible about the input of the other party from their views of the protocol.

Figure 3.2 illustrates the model for SIC which presents a detailed picture of the working structure of the attack. The attack is performed by the adversary $\mathcal{A}$. The attack works in two distinct phases. Phase 1 is also named as the *filtering phase* while phase 2 is named as the *trim phase*. The details on the internal working of each of these two phases of operation is presented in Section 3.4.1 and Section 3.4.2 respectively.

The first phase begins by initializing the anonymity set for each BFE to be the complete set of records present in the global list $D_S$. Phase 1 performs the initial set intersection-based filtering for each of the BFEs which is aided by the dice comparisons with the elements in DD. The filtering works by applying a set of conceptually derived conditions, details of which is discussed in Section 3.4.1. The output is the reduced set of possible records (anonymity set) for the BFEs which is referred to as the *Possibility Set* ($PS$). If the anonymity set for any given BFE is reduced to a single plaintext string (also known as *singleton*), it signifies the complete re-identification of the record. This exposed BFE-plaintext pair is then moved to a separate container known as *Revealed Plaintext list* ($RP$). This list of revealed BFE-plaintext pairs are further utilized as input for the Phase 2 of the attack.

Phase 2 of the SIC relies on a critical mass of known BFE-plaintext pairs which are fed back in to the system to reduce the remaining anonymity set, $PS$. Phase 1 may not yield many plaintexts on its own, therefore, Phase 2 of the attack considers the case where plaintexts are revealed in another way, such as a partial data breach, or as the output of another attack, such as a recent attack due to Kroll and Steinmetzer [61]. Externally fetched BFE-plaintext data is stored in a container known as *Known Plaintext list* ($KP$). $RP$ along with $KP$ are utilized together to further trim down the anonymity sets in $PS$ in a feedback loop until no new singletons are found. The main motive of Phase 2 is to illustrate the fragility of the security Bloom filter encodings in the presence of known plaintexts. The details of the Phase 2 of SIC is presented further in Section 3.4.2.

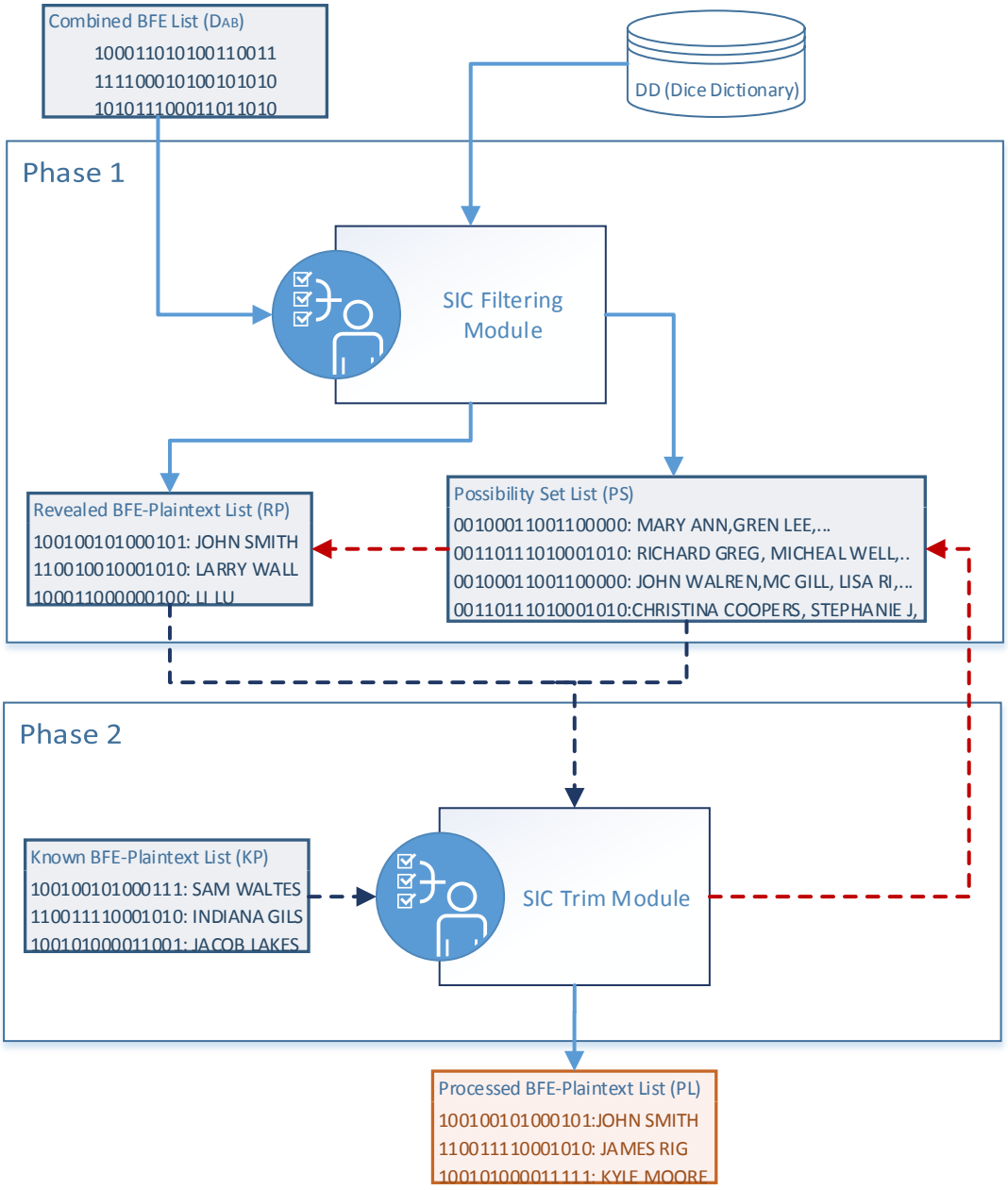



**Figure 3.2: Set intersection-based cryptanalysis (SIC) model.** The model represents the adversarial site where the adversary $\mathcal{A}$ has obtained access to the encoded inputs and is in ready state to mount the attack. In Phase 1, a BFE list ($D_{AB}$) and dice dictionary ($DD$) are input. Set intersections are conducted and reduced possibility sets ($PS$) are output along with any revealed plaintexts ($RP$). In Phase 2, revealed plaintext strings from $RP$ along with externally known plaintexts list ($KP$) are fed back in to reduce possibility sets further. Ultimately, the final list of re-identified BFEs are stored in the processed BFE-plaintext list (PL).

## 3.3 Data Pre-processing

Data pre-processing is a crucial step in any data mining process. Since data collection is often imperfect, it can result in incorrect or missing values. Data pre-processing helps in improving the quality of data as well as saves processing time by removing the irrelevant noise. In context of PRL, data pre-processing also involves the selection of input parameters which would be used for the data linkage. In the current implementation of SIC model, the pre-processing stage comprises of four main tasks: (a) element selection, (b) bigram extraction, (c) dice dictionary creation and (d) parameter selection. Each of these tasks is explained in this section. Figure 3.3 illustrates the process flow of the data pre-processing step. The more specific details on the data inclusion and other parameter settings used in the current implementation of SIC is further explained in Section 4.1. The SIC framework is ready to be mounted once all the pre-processing tasks are accomplished.

### 3.3.1 Element Selection

In this step, a set of data attributes describing characteristics of individuals (*e.g., firstname, lastname, address, gender etc.,*) is selected for identifying individuals in a record linkage process. The choice of attributes can vary based on the type of data and preferences of the participating parties involved in PRL. It is important to determine that the set of attributes selected is sufficient for uniquely identifying an individual (*i.e.,* the combination of all attributes for any two records should have same values only if they represent the same individual). In certain exceptional cases, it may not be possible to acquire enough attributes to distinguish between two people. An example of this situation is a dataset that contains records corresponding to twins (of same gender and having similar firstname and exact same lastname and address), between whom it is often difficult to differentiate in approximate matching record linkage applications. In this thesis, we consider using two variants of input identifiers which comprises of 2-attributes (*firstname, lastname*) and 3-attributes (*firstname, lastname, zipcode*) respectively.

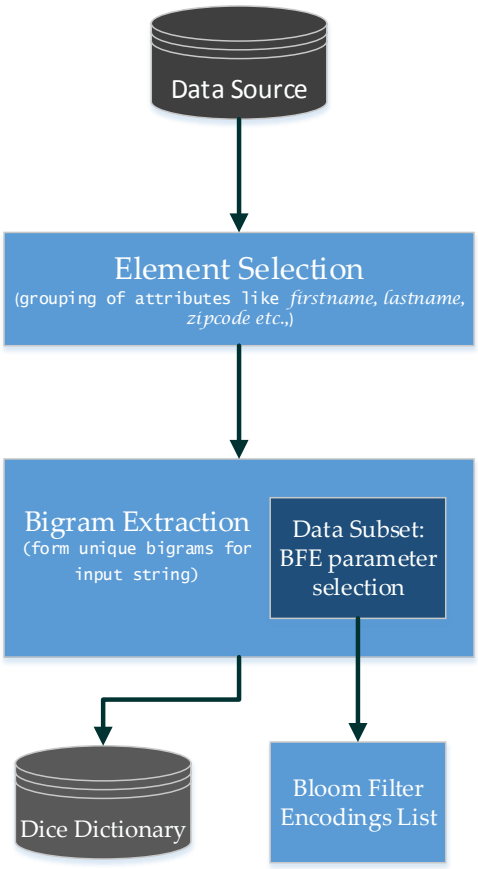



**Figure 3.3: Process flow diagram for pre-processing stages of SIC.** Process starts at the Element selection stage where the identifiable attributes for each record in the global dataset are grouped together to represent one string. These strings are then processed into set of unique bigrams in Bigram extraction stage. A subset of dataset is drawn from here to form the BFE list by applying the selected BFE parameters. Dice Dictionary (DD) is created as a reference list for all the records in the global dataset.

### 3.3.2 Bigram Extraction

This step marks the onset of the process of simulating the role of the parties $A$ and $B$ generating their respective BFE lists. The identifiers (*i.e.,* firstname, lastname *etc.*) selected during element selection stage is split into unique bigrams. Bigrams are formed by sequentially separating every two consecutive letters of the input string. The start and end of the string is marked with the addition of special character "_" for every string. Recurring bigrams in the given input string (if any) are counted only once. Algorithm 1 illustrated the steps for building the set of

bigrams for a given input string. For instance, the bigram decomposition of the name "Jenny" is `{_j, je, en, nn, ny, y_}`. Each of these bigrams eventually occupy a bit position in the Bloom filter associated with its record as discussed earlier in Section 2.4.3.

**Data**: $s_1$: plaintext string
**Result**: $b_1$: set of unique bigrams in string $s_1$
initialization;
%Comment: reset bigram_array before setting bigrams for each plaintext string%
bigram_array[] ← 0;
%Comment: choose demarcation character for starting and ending bigrams%
white ← {_};
  letters[] = extract (all letters of input string $s_1$);
  %Comment: prepare first and last bigrams%;
  first = white . letters[0];
  last = letters[size] . white;
  %Comment: add first bigram%;
  bigram_array[] = first;
**while** *(i ≠ endof(letters[size]))* **do**
  bigram_array[i] = letters[i] . letters[i+1];
**end**
  %Comment: add last bigram%;
  bigram_array[i] = last;
  $b_1$ = bigram_array[];
return $b_1$;

**Algorithm 1: Algorithm for computing bigrams for the plaintext strings.**

### 3.3.3 BFE Parameter Selection

In this step, the parameters required for creating the Bloom filter encodings are selected. BFEs are created on a random subset of elements drawn from the global dataset $D_S$ after the completion of element selection and bigram extraction stages. This BFE list is used as the target list on which the proposed attack can be mounted. The parameters required for creating BFEs are: (a) number of $k$ hash functions, (b) length of the Bloom filter in $m$ bits and (c) total number of bit overlaps allowed, $o$ and (d) secret key $\mathcal{K}$ used for generating the hash output for the $k$ hash functions. The length of the bloom filter $m$ gets experimentally determined while fixing the maximal overlaps $o$ while generating BFEs. A discussion on the BFE parameter selection

along with the justification for the choice is revisited with more details in Section 4.2.

**Data**: plaintext string $s_1$, plaintext string $s_2$
**Result**: Compute and store *DD* fields for the given input
initialization;
$b_1$: compute unique bigrams for $s_1$;
$b_2$: compute unique bigrams for $s_2$;
%Comment: compute dice coefficient for $b_1$, $b_1$ %
$Dice(b_1, b_2)$ = dice coefficient of($b_1$, $b_1$);
  compute the total count of common bigrams: $|b_1 \cap b_2|$;
  compute the count of unique bigrams in $s_1$: $|b_1|$;
  compute the count of unique bigrams in $s_2$: $|b_2|$;
generate pattern $\left[Dice(b_1, b_2) : s_1, s_2 : |b_1|, |b_2|, |b_1 \cap b_2|\right]$;
save pattern string in *DD*;
return;

**Algorithm 2: Algorithm for creating Dice Dictionary (*DD*) entries.**

### 3.3.4 Dice Dictionary Creation

The Dice dictionary (*DD*) consists of a collection of plaintext data processed in a fixed predefined format. *DD* comprises of the dice similarity metrics and distance specific information between all possible plaintext pairs from the global dataset $D_S$. *DD* acts as a reference dictionary which assists in filtering records based on the dice similarity between any two pairs of plaintext strings. This information is later used to compare the distance between encoded and plaintext string pairs based on the filtering conditions described in Section 3.4.1. Algorithm 2 outlines the steps involved in the creation of the Dice dictionary. To begin with, given two input plaintext strings, $s_1$ and $s_2$, their associated bigram set $b_1$ and $b_2$ is computed. $b_1$ and $b_2$ contains the list of unique bigrams only as the recurring bigrams are counted only once. After the unique bigrams are identified for each plaintext string, the dice similarity between $s_1$ and $s_2$ is computed using the dice coefficient metric as expressed in equation (2.1). The cardinality of sets $s_1$ and $s_2$ is computed as $|b_1|$ and $|b_2|$. Similarly, the cardinality of $|b_1 \cap b_2|$ is computed. All of this information is stored in *DD* for every pair of plaintext strings. Each row of *DD* contains information in the following pattern: $\left\{Dice(b_1, b_2) : s_1, s_2 : |b_1|, |b_2|, |b_1 \cap b_2|\right\}$.

A sample subset of the Dice dictionary is shown in Appendix A.

## 3.4 Framework: Set Intersection-based Attack

Recall the earlier discussion on the overall attack model from Section 3.2.3 which outlined the combined roles of the two phases (the filtering and the trim phases respectively) in the proposed set intersection-based cryptanalysis. This section aims at presenting the internal architecture and the implementation details for each of these phases on an individual basis.

### 3.4.1 Filtering Phase

The main motivation behind the conception of the filtering phase of SIC is derived from the following properties of Bloom filter encoding:

1. Bloom filter encodings are deterministic by nature. For a given plaintext string $x$, the Hamming weight[3] of the $m$-bit Bloom filter encoded value $BFE(x)$, as explained earlier in Section 2.4.3 is proportional to the total number of unique bigrams in $x$. This property is used in conjunction with set operations to derive the conditions used during the filtering stage of the attack.

2. Bloom filter encoding technique is capable of accommodating approximate string matching. This is evident from the fact that $Dice_{plain}$ (*i.e.,* the Dice coefficient similarity metric between any two plaintexts) is an approximation of the $Dice_{bfe}$ (*i.e.,* the Dice coefficient similarity metric between any two BFEs) which is used for predicting the matches in record linkage process.

***Architecture.*** Reiterating the background setting, the adversary $\mathcal{A}$ is the trusted third-party entity with whom the parties A and B share their respective BFE lists, $D_A$ and $D_B$ for secure

[3]The Hamming weight of the Bloom filter encoding function $BFE(\cdot)$ equals the total number of bits set in the filter.

record linking. Adversary $\mathcal{A}$ prepares the aggregated BFE list $D_{AB}$ such that $D_{AB} = D_A \cup D_B$. This combined BFE list, $D_{AB}$ (representing the hypothetical list of patient records in encoded format) is used for secure record linkage which in the present context, simultaneously represents the target list of BFEs which the adversary $\mathcal{A}$ would attempt to re-identify by mounting the SIC.

**Data**: BFE list ($D_{AB}$), global dataset ($D_S$), Dice dictionary ($DD$) prepared from global dataset ($D_S$)
**Result**: Possibility set ($PS$) list for each plaintext string in $D_{AB}$ , revealed plaintext list ($RP$)
initialization;
**while** *index [i] is not at end of* $D_{AB}$ **do**
    initialize $BFE_i$ to all records in global dataset ($D_S$);
    loop through the remaining elements in the list with iterator j;
    read BFEs in pairs;
    calculate Dice coefficient, $Dice_{i,j}$ for $BFE_i$ and $BFE_j$;
    **if** $(Dice_{i,j} \neq 0 \,||\, 1)$ **then**
        specify filtering conditions: $FC$;
        filter plaintext string pairs from $DD$ which satisfy $FC$;
        perform set union of all plaintext strings found matching in iteration j;
        perform set intersection of plaintext strings with each iteration count of j;
        return $PS$ for $BFE_i$;
    **else**
        skip to the next record in inner loop; increment j ;
    **end**
    increment i;
**end**
evaluate $PS$ for the entire bfe list;
**if** *Singletons found* **then**
    mark DoA=0;
    remove it from $PS$;
    store in Revealed Plaintext list ($RP$) for next phase of experiment;
**end**
calculate average Degree of Anonymity (DoA) for each BFE list element;

**Algorithm 3: Algorithm for the Filtering phase of Set Intersection-based Cryptanalysis.**

At the start of the filtering phase, the BFE list $D_{AB}$ along with the plaintext global dataset $D_S$ and the Dice dictionary ($DD$) prepared from global dataset $D_S$ are supplied as inputs to the module. Since the adversary $\mathcal{A}$ has zero knowledge of the secret key $\mathcal{K}$ used for generat-

ing the encodings, it is safely assumed that each encoded string present in $D_{AB}$ is completely anonymous. Therefore, each of the BFEs in $D_{AB}$ is the encoded representation of one of the records present in the global dataset $D_S$. The possibility set for each of the BFEs is therefore set to be all plaintext records present in the global datatset, $D_S$. The filtering module starts with reading each of the BFEs from $D_{AB}$ and process it successively. The first encoded value in the $D_{AB}$ is read and is affixed as the *target* BFE. This *target* BFE is then compared with the next BFE (referred to as the *subsequent* BFE) in the list and their $Dice_{bfe}$ is computed. $Dice_{bfe}$ along with the Hamming weights of the target BFE and the subsequent BFE is used to filter the preliminary list of records from the *DD*. A set of filtering conditions derived from the above stated properties of the Bloom filter encoding is used for filtering these records from the *DD*. The pair of records in *DD* which satisfy all the filtering conditions are filtered out as the set of possible records. Set union operation is performed to get the complete list of records filtered out in the first iteration. The filtering module moves on to the next subsequent BFE in $D_{AB}$ while keeping the *target* BFE fixed to the same position and repeats the above filtering step. In this way, each iteration has the *target* BFE in common and a different *subsequent* BFE. Set intersection operation is therefore performed between the list of records obtained in each iteration to obtain the updated (ideally reduced) anonymity set at each step. The plaintext strings which survive through every iteration, make it to the final list and rest all get discarded in the process. The final outcome is stored in the possibility set ($PS$) along with the BFE to represent its reduced anonymity set. Once, the $PS$ is generated for the current *target* BFE, the target is shifted down to the next BFE and the whole process is repeated again.

This process continues till the entire BFE list is traversed and the result is the possibility list, $PS$ which contains the list of all possible records for each encoded value from the BFE list, $D_{AB}$. During the course of execution of the filtering phase, if the possibility set ($PS$) for a given BFE is reduced to a single plaintext record, it represents the situation of complete re-identification for that particular BFE. These isolated BFE-plaintext pairs are termed as "singletons". Singletons are moved to another container known as the revealed plaintext list (*RP*).

*RP* is further used as an input for the Phase 2 of the attack.

The above steps are outlined in the Algorithm 3 which is directly utilized in the software implementation of the filtering module of SIC.

***Filtering conditions.*** The filtering conditions which are used during the filtering phase of the SIC, are listed and explained as follows:

$$Dice_{bfe} \geq Dice_{plain} \tag{3.4}$$

$$\left\lfloor \frac{H(X_{bfe})}{k} \right\rfloor \leq |X_{plain}| \leq \left\lfloor \frac{H(X_{bfe}) + \text{maxoverlaps}}{k} \right\rfloor \tag{3.5}$$

$$\left\lfloor \frac{H(Y_{bfe})}{k} \right\rfloor \leq |Y_{plain}| \leq \left\lfloor \frac{H(Y_{bfe}) + \text{maxoverlaps}}{k} \right\rfloor \tag{3.6}$$

$$|C_{plain}| \leq \left\lfloor \frac{H(C_{bfe})}{k} \right\rfloor \tag{3.7}$$

where:

- $H(X_{bfe})$: Hamming weight of BFE $X_{bfe}$,
- $H(Y_{bfe})$: Hamming weight of BFE $Y_{bfe}$,
- $H(C_{bfe})$: Hamming weight of $X_{bfe} \cap Y_{bfe}$,
- $|X_{plain}|$: total unique bigrams in plaintext $X$,
- $|Y_{plain}|$: total unique bigrams in plaintext $Y$,
- $|C_{plain}|$: total bigrams common in plaintext $X$ and $Y$, *i.e.,* $(|X_{plain}| \cap |Y_{plain}|)$
- $Dice_{bfe}$: Dice coefficient measure between BFEs $X_{bfe}$ and $Y_{bfe}$,
- $Dice_{plain}$: Dice coefficient measure between two plaintext $X$ and $Y$,

- maxoverlaps: maximum allowed bit overlaps in BFE generation,
- $k$: total number of hash functions in the BFE function.

Equation 3.4 is derived based on the feature of the Bloom filter to allow overlaps (*i.e.,* mapping the input to a bit position which has already been set previously). Therefore, Dice similarity for BFEs can be greater than or equal to dice similarity between the plaintext strings. Equation 3.5 and 3.6 are derived based on the maximum allowed overlap restriction during the construction of BFEs. The total number of unique bigrams in the plaintext strings will be bounded in between the absolute Hamming weight of its encoded value plus the relaxation for accommodating overlaps. Equation 3.7 is also derived based on the overlap constraint in encoding process which allows the absolute Hamming weight of the intersection of two BFEs to be greater than or equal to the total common bigrams in the corresponding plaintext strings. All the above derivations also consider the other design factors such as the Bloom filter encodings are deterministic and provide a fair estimate about the length of the input plaintext strings.

***Complexity analysis.*** Filtering module of SIC is based on the exhaustive search oriented problem solving methodology, used for simulating any dictionary attack environment. The asymptotic analysis of the filtering algorithm is as follows: The complexity for traversing BFE list $D_{AB}$ is $O(n^2)$ for $D_{AB}$ with $n$ encodings (upper bounded). Each of the records in the Dice dictionary ($DD$) is traversed in an inner loop which adds $O(m)$ computation cost where $m$ represents the total size of $DD$ with filtering conditions applied in each pass. In addition, there are 3 $O(1)$ instructions (constant time) for performing set-union operation and storing the filtered records in the possibility set (PS) or revealed plaintext list (RP). Set intersection operation is performed for every $n-1$ BFEs which would add a cost of $O(n-1)$. Therefore, the overall computation cost of the filtering algorithm would be $O(n^2)$ . $O(m)$ + $O(n-1)$+ 3

$O(1) \approx O(mn^2)$. Therefore, the overall cost is proportional to the number of candidate records in the global dataset times the size of the encoded records which practically tends to grow very quickly as the size of the problem increases. The memory requirements vary across for different datasets and the maximum recorded value is around 8.1 GB RAM in a sequential processing setup.

### 3.4.2 Trim Phase

The main motivation behind the conception of the trim phase of SIC is to demonstrate the low resistance of the Bloom filter encoding technique in presence of known plaintext information. The theoretical security analysis in Section 3.1 shows that BFEs are "distinguishable" under chosen plaintext attacks. Applying the set union property on the encoded values of two plaintext strings X and Y, it is known that $BFE(X) \cup BFE(Y) = BFE(X \cup Y)$. Now if $X'$ is another known plaintext such that $X' \cap X \cap Y = \emptyset$, then $X'$ can be safely removed from the possibility sets for $BFE(X)$ or $BFE(Y)$. The Trim phase of the attack is designed to discover and remove such disjoint pairs of plaintext elements from the possibility sets of BFEs obtained from the filtering phase. This set elimination strategy proves useful in further reducing, hence "trimming" the anonymity set of any encoded value $BFE(x)$.

***Architecture.*** The trim phase of the SIC attack aims at further trimming down the existing possibility set $PS$ obtained from the filtering phase using the properties of the already known BFE-plaintext pairs. Trim phase makes use of the revealed plaintext list $RP$ from Phase 1, optionally padded with auxiliary revealed plaintexts from other sources which is stored in another container called *Known Plaintext list* ($KP$). Trim phase of the SIC, therefore can be deemed as the continuation work from the filtering phase as it depends on the outputs from filtering phase which are: (1) the reduced anonymity set of BFEs, $PS$ and (b) the revealed plaintext list, $RP$.

The attack phase commences by drawing the topmost BFE-plaintext pair from the $RP$ list and running it through the trim conditions to further reduce the existing size of $PS$ for the

remaining BFEs. At the end of each iteration, the already iterated BFE-plaintext entries are removed from the *RP* list and appended to the final container *Processed List*, *PL*. Thus, the *RP* list goes on shrinking while *PL* keeps on growing with each iteration. Simultaneously, the possibility set *PS* gets updated during each iteration and hence is examined at the end of each iteration to measure the remaining overall anonymity. During trimming process, new sets reduce to singleton and are added to the *RP* list, and are fed back in the trim module for processing. The experiment continues until no further plaintexts are revealed. In that sense trimming functions like a chain reaction, and the goal is to gather a "critical mass" of plaintexts in *RP* so that the experiment becomes self-sustaining until all records have been re-identified. Trim phase of the attack module is designed to halt automatically if it runs out of known BFE-plaintext pairs from all sources (*i.e.,* both *RP* and *KP* lists are completely exhausted) or if all the BFEs are exposed (*i.e., PS* is exhausted). The final outcome of trim phase is the final list of re-identified BFE-plaintext records stored in *PL*.

Although the execution of trim phase relies on the magnitude of *RP* from Phase 1, and may not perform very well if the filtering phase did not reveal many plaintexts on its own (as per our eventual results), it has still proven useful in showcasing the fragility of BFE technique under known plaintext attack. Moreover, the effectiveness of the overall attack can be multiplied by the plaintexts recovered by other attack methods. For example, Kroll et. al [61] recently reported a recovery rate of 77.7%. If those results can be utilized as input to Phase 2 of SIC, it is feasible to recover all the remaining BFEs in a very low effort and cost effective manner.

The above steps are outlined in the Algorithm 4 which is directly utilized in the software implementation of the trimming module of SIC.

***Trimming conditions.*** Trim phase of SIC works on set of conceptually derived conditions which for a given revealed plaintext string $X_{plain}$, chiefly deploy the elimination strategy to discover the corresponding "sure" disjoint plaintext strings remaining in *PS* for a given BFE $Y_{bfe}$ such that $Y_{bfe} \cap X_{bfe} = \emptyset$ and removing it. These conditions are listed and explained here:

**Data**: Possibility set ($PS$) for each plaintext string in $D_{AB}$ , revealed plaintext list ($RP$), list of known bfe-plaintext pairs ($KP$)
**Result**: Re-identified bfe-plaintext pairs stored in final processed list ($PL$)
initialization;
**while** *($RP$ && $KP$)* **do**
  **Step1 :** introduce a single bfe-plaintext pair from $RP$ into $PS$;
  trim the possibility set list ($PS$) based on the set of Trim Conditions to remove non-matching plaintext entries;
  **if** *new Singletons found* **then**
    update $RP$ to add new singleton entry;
    return;
    exit if ($PS$ isEmpty);
  **end**
  **if** *$RP$ isEmpty* **then**
    **Step2:** introduce a single bfe-plaintext pair from $KP$ into $PS$;
    trim $PS$ based on the set of Trim Conditions to remove non-matching plaintext entries;
    **if** *new Singletons found* **then**
      update $RP$ to add new singleton entry;
      goto Step1;
    **else**
      return;
    **end**
    exit if ($PS$ isEmpty);
  **end**
  calculate average Degree of Anonymity (DoA) for all remaining BFEs after each iteration;
**end**
return final processed list $PL$;

**Algorithm 4: Algorithm for the Trim phase of Set Intersection-based Cryptanalysis.**

1. **Trim possibility set**: Find and remove instances of the revealed plaintext string $X_{plain}$ from any BFE anonymity sets in $PS$ still containing it.
2. **Find disjoint BFE pairs**: For the BFE $X_{bfe}$ associated with the previous step, find all BFEs $Y_{bfe}$ in $PS$ for which $X_{bfe} \cap Y_{bfe} < k$, where $k$ denotes the total number of hash functions in the BFE function.
3. **Remove non-matching plaintext strings**: For all BFEs $Y'_{bfe}$ during the previous step, remove any plaintext string $X'_{plain}$ in its possibility list for which $X'_{plain} \cap X_{plain} > k$, where

$k$ denotes the total number of hash functions in the BFE function.

4. **Remove additional plaintext string**: Find and remove all plaintext string $X'_{plain}$ in $PS(ANY_{bfe})$ for which the starting and ending bigrams are common with $X_{plain}$ but $X'_{bfe} \cap X_{bfe} = \emptyset$. This condition is derived from the fact that if two BFEs are completely disjoint, then the corresponding possible plaintext strings cannot have start and end bigrams same as $X_{plain}$.

***Complexity analysis.*** Trim module of SIC is based on the elimination of the disjoint occurrences of known plaintext information from the possibility set ($PS$). The asymptotic analysis of the trim algorithm is as follows: Assuming the combined size of revealed plaintext list ($RP$) and known plaintext list ($KP$) at start is $m$ elements, then the cost of traversal is $O(m)$. The worst case complexity for traversing the possibility set (PS) of size $n$ is $O(4.logn)$ in each iteration as the size of $PS$ goes on shrinking on the application of the 4 trim conditions. In addition, there is constant $O(n)$ cost (in worst case) for moving the new re-identified records from $PS$ to $RP$. Therefore, the overall computation cost of the trimming algorithm would be $O(m)$ . $O(4.logn) + O(n) \approx O(n.logn)$. The memory requirements are overall low even though it varies across different datasets depending upon the size of the possibility set and the pace of BFE re-identification.

## 3.5 Summary

The formal security analysis for the Bloom filter encoding technique was presented in this chapter. Using the chosen plaintext attack (CPA) game approach from modern cryptography, it is shown that Bloom filter encoding techniques do not adhere to the basic acceptable levels of formal security. Bloom filter encodings are deterministic in nature which leaves some room for further exploitation for this low security cover. The result of this security analysis blended together with the other inherent properties of the Bloom filter encoding techniques, result in the main work of this thesis, which is the set intersection-based cryptanalysis (SIC)

framework. This chapter houses the complete discussion on the inception, design, modeling and implementation details for the proposed SIC attack. This attack itself works in two distinct phases, the filtering and the trim phase. A separate section in the chapter was dedicated to cover the underlying design and architectural details of each of the two phases. Additionally, algorithms outlining all the steps involved in processing of each of these two phases have been included. Although SIC is not the first cryptanalysis work on the BFE technique, it is the first in its kind which showcases the fragility of the system to known plaintext attacks in addition to re-establishing the inherent security weaknesses present in the BFE system.

# Chapter 4

# SIC Materials and Methods

This chapter outlines the various methodologies adopted during the implementation of the proposed Set Intersection-based Cryptanalysis (SIC), described in the previous Chapter 3. Beyond the discussion on the software implementation of SIC, the choice and variety of the datasets selected for conducting different experiments is also been discussed. Further, the various evaluation metrics used for quantifying the effectiveness of SIC have been defined in this chapter. This chapter serves as a baseline for understanding the specifics of the experimental settings and the selected evaluation criterion.

## 4.1 Experimental Methods

This section describes the methods considered during the data preprocessing stages of SIC, described earlier in Section 3.3. The two major aspects discussed here are: (1) attribute selection and (2) string conversion to n-grams.

**Attribute Selection.** The prime focus of the work presented in this thesis is to re-establish the inherent security weaknesses present in the Bloom filter encoding method while empirically strengthen the fact, that these security flaws are not likely fixable. For instance, Kuzu *et. al* [65] quote that practical solutions to safeguard BFEs from various forms of cryptanalysis can be

derived by combining multiple attributes of patient identifiers into a single BFE. However, the research motivation presented in this thesis does not agrees upon such claims. Keeping this in mind, two separate groups of identifiers are selected to form the datasets. These groups are identified as:

1. **2-attribute set:** This consists of forming input string $s$ by combining identifying attributes {*firstname, lastname*}.

2. **3-attribute set:** This consists of forming input string $s$ by combining identifying attributes {*firstname, lastname, zipcode*}.

The choice of these identifier attributes is directly derived from the work of Kuzu *et. al* [64, 65]. In addition, this also allows for the construction of BFEs with completely different parameters and hence, further enables to present an unbiased security analysis towards the completion.

**Approximate String Matching.** In PRL applications, the main functionality of Bloom filter encodings is based on approximately matching strings. In this scenario, a single string $s$ is converted to a set $S$ by decomposing it into $n$-grams, *i.e.,* the set of every sequence of all adjacent and unique substring elements of length $n$ in $s$. Repeating substrings are considered only once in the set $S$. The most commonly used string decomposition method is 2-grams (where $n$=2) which is used for the statistical analysis of texts in speech recognition applications, cryptography and computational linguistics. 2-grams is also popularly known as *bigrams* and has been interchangeably used in this thesis. Further, 3-grams (where $n$=3), also known as *trigrams* can also be used in the context of string matching. We consider using 2-grams (bigrams) in the current work to explicitly follow the setup from Kuzu *et. al* [64, 65] so as to disprove their claims about "chaffing" identifiers to enhance security. This setup, in a way, also maintains parity with other similar works in literature [75]. To capture the beginning and end of a string, a *blank* character is concatenated, denoted by '_' to the front and back of $s$. So, as an example, the 2-grams decomposition of the name "Ben" is `{_b, be, en, n_}`. Being a set, bigrams are not repeated, so *e.g.,* "Lulu" decomposes to `{_l, lu, ul, u_}` (and the second 'lu' is not

captured). Similarly, considering another example based on the 3-attribute input string (consisting of *firstname, lastname, zipcode*), for *e.g.,*"Ben Moss" with zipcode `27098`, the bigram decomposition set $S$ would be `{_b, be, en, n_, _m, mo, os, ss, s_, 27098}`. The numeric values corresponding to zipcode attribute is represented as a single entity.

## 4.2 Controls

This section elaborates on the choice of BFE parameters (like the length of the Bloom filter, number of hash functions *etc.,*) which are considered during the construction of the encoded values from the identifying attribute strings. The choice and variants of attribute strings has been previously discussed in the Section 4.1. Beyond the selection of these primary parameters, a novel strategy for curtailing the total number of bits overlaps in the Bloom filter is also discussed in this section.

**Hash Functions.** The number of hash functions to be used during BFE construction can be solely and independently determined by the researchers or the parties involved (in a PRL setting). For instance, Schnell *et. al* [88], who initially proposed using BFEs for PRL, consider using 15 hash functions to illustrate their work. They even scale up further to use 50 hash functions in one of their experiments, which only shows the decline in the performance of the PRL application. In another recent work, Kroll and Steinmetzer [61] construct the Bloom filter encodings by hashing bigrams though $k$=20 hash functions on a filter of length $m$=1000 bits. In the cryptanalysis work by Kuzu *et. al* [64, 65], BFEs were obtained by hashing bigrams though $k$=2 hash functions. Moreover, Kirsch and Mitzenmacher [58] suggest that only two hash functions are necessary to effectively implement a Bloom filter without any loss in the asymptotic false positive probability. On the basis of their experiments and the result analysis, they further state that using $k$=2 hash functions leads to less computation and potentially less need for randomness in practice. Therefore, staying aligned with the work of Kuzu *et. al* [64], the total number of hash functions considered in this thesis work is $k$=2, as apparently, only two

hash functions are self sufficient for Bloom filters to function without further loss of accuracy. SHA1 hash function in an HMAC configuration has been utilized in all the experiments. To facilitate a more direct comparison, same length secret keys are used for BFE hashing across all datasets.

**Overlaps.** The term *overlap* is defined to represent the situation in which an item is being inserted into a Bloom filter, but one or more of the corresponding bit positions were already set during a previous insertion. In a conventional setting, the above condition is just ignored, as the bit position in question, is already set. The Bloom filter is a simple data structure which lacks the capability of self-organizing or controlling the number of bit overlaps in a given fixed length filter of $m$ bits.

Our research hypothesis, however, is that while using BFEs in context of PRL and allowing unrestricted overlaps during BFE construction would directly impact the quality of the string matching results. This hypothesis is based on the fact that higher bit overlaps will lead to higher false positive rates, and in turn lower matching accuracy of BFE-based PRL system. These false positives will occur due to different inputs (*e.g.,* string bigrams) getting mapped to same bit positions in the Bloom filter by different hash functions.

In order to conduct an empirical analysis surrounding this hypothesis, we propose a mechanism to restrict bit overlaps in the Bloom filter by tuning the overall length of the filter. Furthermore, in a PRL setting, the participating parties already agree on the BFE construction parameters such as the data attributes, the Bloom filter length $m$, number of $k$ hash functions and secret key $\mathcal{K}$, so similarly, they can also agree upon the maximum permitted overlap count $o$. Algorithm 5 outlines the steps involved in restricting overlaps in the BFE construction. The algorithm requires the following inputs: bigrams string set $S$ generated from the data attributes (which can be either a 2-attribute set *{firstname, lastname}* or a 3-attribute set *{firstname, lastname, zipcode}*), the number of hash functions $k$, initial size of the Bloom filter $m$ and the maximally allowed overlap count $o$. To enforce overlap restrictions, the elements are inserted

into the Bloom filter positions corresponding to the output of each of the hash functions until an overlap is encountered. For each overlap condition, a counter is incremented which keeps track of the upper limit of the maximum overlaps, $o$. On reaching the maximum allowed overlaps, the process is aborted and the existing list of BFEs generated to that point is discarded. The entire process restarts with resizing the Bloom filter length (*e.g.,* double the filter length $m$). On completion, the algorithm delivers the list of BFEs with the overlap restriction along with the experimentally determined Bloom filter length $m'$ (if the new value is different from the initial stated value).

**Data**: Bigram string set $S$, Bloom filter of length $m$ and $k$ hash functions, maximum permitted overlap count, $o$
**Result**: BFEs with maximum of $o$ bit overlaps, experimentally determined Bloom filter length $m'$
initialization;
%Comment: Initialize a variable to keep track of $o$ overlaps%
$track \leftarrow 0$;
**while** *Bigram string set S != empty()* **do**
  **Loop** <*foreach: k*>
    hash function = $h_k$;
    bigram = $s$ such that $s \in S$;
    %Comment: derive the corresponding bit position, $bitpos$ in the Bloom filter %
    1. $bitpos = h_k \bmod m$;
    2. insert $s$ into the Bloom filter if $bitpos$ is not set && return;
    3. increment *track* and insert $s$ into the Bloom Filter if [$bitpos$ is already set & track $\leq$ (overlaps=$o$)];
    4. else break;
  **EndLoop**
**end**
**if** *track > (overlaps=o)* **then**
  display message "Maximum permitted overlaps exceeded" and exit();
  discard the list of BFEs generated till this point;
  resize the Bloom filter length $m' = 2 \times m$ and re-run;
**end**

**Algorithm 5: Algorithm for restricting overlaps in BFE construction.**

Considering the probability of hash collisions based on the birthday paradox[1] [57], for a $n$-bit hash output, the probability for the occurrence of the first hash collision is after $\approx 2^{n/2}$

[1] Birthday paradox is a mathematical problem which tries to establish the probability that, in a set of n randomly chosen people, some pairs will have the same birthday.

operations. In case of a BFE construction, the actual hash output is further reduced to $n$ bits, where $n = \lceil log_2(m) \rceil$ and $m$ is the size of the Bloom filter. Therefore, in this context, for a $m$-bit Bloom filter, after $2^{n/2}$ trials, at least one collision is quite probable. The algorithm for controlling overlaps in the current work considers maximum $o$ overlaps (*i.e.,* $\{0, 1, 2 .. o\}$ collisions are permitted) and the process is aborted only on reaching $(o+1)^{th}$ collision. Therefore, the experimentally determined size of the Bloom filter, $m'$ (*i.e.,* the algorithmically revised m) represents BFE construction with overlap relaxation up to $o$. Considering an example scenario, for a set of 300 BFEs, if we attempt to restrict overlaps $o$ such that $o \leq 1$ and using Algorithm 5, obtain the Bloom filter length as $m' = 6000$, then the Bloom filter indexes would be any number between 0 and 5999 and $n = \lceil log_2(6000) \rceil = 13$. This means at least 13 bits are required to represent this number. Therefore, the probability of the first hash collision (overlap) would be after $\approx 2^{13/2} \approx 128$ operations, however, up to one collision would be acceptable in the given setting.

BFE construction with overlap restriction enforcement is an aspect of record linkage process which has not been addressed in the existing literature so far. The number of allowable overlaps relates to the overall accuracy of matching, and the experiments in this thesis work consider a number of different bounds. With $k = 2$ hash functions, four variations of maximum allowed overlaps has been considered while generating BFEs: 1, 3, 5 and unrestricted, which gives sufficient variety to study the impact of overlaps on security and accuracy. Each of these overlap variants demanded an accommodation for a Bloom filter of different size which is experimentally derived as:

- **Unrestricted overlaps.** BFEs are generated with unrestricted overlaps (*i.e.,* $o = \infty$). The Bloom filter size is obtained as $m = 200$ bits for both 3-attribute and 2-attribute sets.
- **Overlaps** $\leq 5$**.** BFEs are generated with maximum 5 allowed overlaps per encoding. The Bloom filter size is obtained as $m = 500$ bits for both 3-attribute and 2-attribute sets.
- **Overlaps** $\leq 3$**.** BFEs are generated with maximum 3 allowed overlaps per encoding. The Bloom filter size is obtained as $m = 1000$ bits for both the 3-attribute and 2-attribute sets.

- **Overlaps** $\leq 1$**.** BFEs are generated with maximum 1 allowed overlap per encoding. The Bloom filter size is obtained as $m = 6000$ bits for the 3-attribute set and $m = 5950$ bits for 2-attribute sets.

## 4.3 Evaluation Metrics

Another focus of this thesis work is to analyze security versus accuracy for the Bloom filter encodings in a PRL setting. Also, the experiments aim at validating the prime hypothesis of this research work, *i.e.,* the BFE method is susceptible to known plain text attacks. Therefore, keeping these goals in mind, the appropriate evaluation metrics have been selected and incorporated in the experiments. This section provides the specifics of these evaluation metrics.

### 4.3.1 Accuracy

Semantically, accuracy measures the proportion of true results among the total size of sample set examined. Metz [69] has provided a definition of accuracy as a function of the total number of true positives ($TP$), true negatives ($TN$), false positives ($FP$), and false negatives ($FN$) which is defined as follows:

$$Accuracy = \frac{\sum TP + \sum TN}{\sum TP + \sum TN + \sum FP + \sum FN}. \tag{4.1}$$

The standard definition of accuracy as expressed in equation (4.1), and the related outcomes *i.e.,* true positives ($TP$), true negatives ($TN$), false positives ($FP$) and false negatives ($FN$) are interpolated in the context of the BFE experiments as the outcome of $Match_{bfe}$ (*i.e.,* string matches deduced from the Bloom filter encoded values) relative to $Match_{plain}$ (*i.e.,* string matches in plaintext strings) with the latter representing the ground truth. The complete set of derivations for all possible outcomes are further illustrated in Figure 4.1. $Dice_{bfe}$ here represents the Dice coefficient similarity metric between any two BFEs and similarly $Dice_{plain}$

represents the Dice coefficient similarity metric between the corresponding plaintexts. The accuracy measure has been utilized to study the impact of bit overlaps in BFEs against the overall string match predictions.

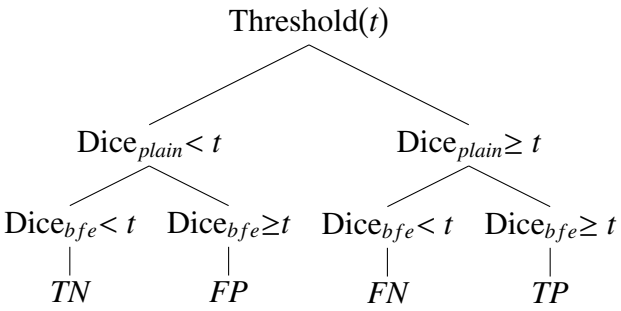



**Figure 4.1: Tree structure representing prediction outcomes based on Dice similarity.** BFE Prediction outcomes are based on comparing Dice similarity for the Bloom filter encoded ($Dice_{bfe}$) and plaintext ($Dice_{plain}$) values against a given Dice coefficient threshold, $t$.

### 4.3.2 Degree of Anonymity

The most widely acclaimed and accepted definition of anonymity was given by Pfitzmann and Hansen [80]: "anonymity is the state of being not identifiable within a set of subjects, the anonymity set." The *anonymity set* is "the set of all possible subjects who might cause an action." In simpler terms, subjects can be considered to be more anonymous if they can hide themselves in a larger group.

With formal security proof unavailable, therefore, on the basis of the security analysis in Section 3.1 for the current setting, we characterize the security of a record instead by its re-identification risk. Re-identification risk for an individual record can be quantified as the inverse of the size of its associated anonymity set. Further, to analyze the outcome of the proposed SIC attack, there was a need for a normalized metric to characterize the *degree of anonymity* of not only a single record, but the collection of records overall. Díaz *et al.* [33] and Serjantov *et al.* [90] introduced this measure to quantify the anonymity in networks, however its not immediately applicable to our setting.

The criteria for a degree of anonymity metric in our setting is as follows: full anonymity

of a record, corresponding to a degree of anonymity (DoA) of 1, is defined relative to size of the initial anonymity set as implied by the global name list, whereas a DoA of 0 represents uniquely identified record. For example, given a name list consisting of a state voter registry, a DoA of 1 represents a record that is equally attributable to any voter in the state, whereas a DoA of 0 means the record is uniquely associated with a specific voter's identity. Based on this criteria, we define the following degree of anonymity metric:

Let $PS_i$ is the anonymity set of a BFE $B_i$ associated with record $R_i$. Let $n$ be the size of the global name list. The degree of anonymity of a single record is defined as follows:

$$DoA(R_i) = \frac{|PS_i| - 1}{n - 1}. \tag{4.2}$$

Further, an average degree of anonymity across a set of $r$ records is defined as follows:

$$AverageDoA = \frac{1}{r} \sum_{i=1}^{r} DoA(R_i). \tag{4.3}$$

We measure average DoA of BFEs upon completion of both phases of SIC attack, allowing us to study the impact of the attack phases separately.

## 4.4 Datasets and Experimental Setup

This section provides the details of our experimental setup and the choice of datasets. In order to gauge the susceptibility of BFEs to the proposed SIC and further to allow other researchers to evaluate our work, we utilize the North Carolina voter registration (NCVR) [6] as our global dataset. NCVR is a free and publicly available resource of the voters registration records of the state of North Carolina in the United States. This dataset contains 6,190,504 individual records in total. We initially derive a random sample of 1,000,000 records from the NCVR which has no missing values. We then apply our data selection optimization strategy, discussed in Section 4.5 to form our test dataset $D_S$ of approximately 2500 unique records. Our global name

list $D_S$ is chosen as a mix of some of the longest and shortest names available in the NCVR. Two distinct datasets for 2-attribute and 3-attribute identifiers are prepared using the identical records from $D_S$. Out of these sets, we randomly select a subset of 300 unique records to form the BFE list $D_{AB}$ which acts as the target BFE list which the SIC attempts to re-identify. In a theoretical study by Pang and Hansen [79], a similar scenario is constructed by drawing encoded identifiers as a random sample subset of the global dataset (*i.e.,* the NCVR list). Combining these two variations of attribute sets with the 4 different choices for BFE overlap restrictions, 8 test datasets are generated in total. The various flavors of the test datasets are summarized in table 4.1. The proposed cryptanalysis, SIC is then performed independently across all 8 datasets.

| **Attributes** | **Overlap=1** | **Overlap=3** | **Overlap=5** | **Overlap=∞** |
|---|---|---|---|---|
| *firstname+lastname* | $m$=5950 | $m$=1000 | $m$=500 | $m$=200 |
| *firstname+lastname +zipcode* | $m$=6000 | $m$=1000 | $m$=500 | $m$=200 |

**Table 4.1: The Summary of 8 different datasets used in the experiments.** There are two main subsets of datasets with 2-attributes and 3-attributes. Four different datasets with different overlap restrictions are created for each of the two subsets. $m$ represents the Bloom filter length (in bits) which is experimentally derived.

## 4.5 Optimization – Data Riving

This section provides information about the performance optimization strategy and the related method, which is deployed locally while mounting SIC across the various datasets. Our initial global record set $D_S$ is generated from the North Carolina Voters Registry as discussed in the previous Section 4.4. By design, the Filtering phase of SIC is resource intensive in nature as it needs to perform $O(n^2)$ comparisons, in general. On the contrary note, the NCVR is a large collection (with 6,190,504 individual records in dozens of GBs of text data), and given 8 distinct and independent variants of datasets considered in the current work, there was a need

to sanitize and perturb the global record set, $D_S$. Therefore, in order to alleviate the computational complexity of the Filtering phase of SIC, we applied a local optimization technique to effectively reduce the name list size without losing on the technical equivalence of using the original dataset. The motivation for deriving this optimization strategy stems from the fact that BFEs leak information about the length of the input strings (as discussed earlier in Section 3.1). The complete set of steps involved in the dataset optimization technique is summarized in Algorithm 6.

**Data**: Original global record set $D_S$, riving factor $x$
**Result**: Perturbed record set $D_S$ of size $2x$
initialization;
%Comment: sort global dataset $D_S$ and select $2x$ elements%
sort_ascending(global dataset $D_S$);
**Loop** <*global dataset* $D_S$>
  %Comment: derive the topmost and bottommost $x$ elements from $D_S$%
  1. top_elements = pick (0..$x$) elements from $D_S$;
  2. bottom_elements = pick (size..$x$) elements from $D_S$;
  3. perturbed $D_S$ = combine (top_elements, bottom_elements);
**EndLoop**
return perturbed $D_S$;

**Algorithm 6: Algorithm for deploying Data Riving technique for performance optimization.**

The main idea is to sort the global record set $D_S$ based on the length of the input identifiers. From this sorted list, we generate a synthesized subset of $D_S$ to form the local dataset, which is directly used in the experiments. Since BFEs leak information about plaintext string length, we can effectively reduce the size of a BFE database and associated name list by considering only the BFEs (resp. names) of a certain Hamming weight (resp. number of bigrams). To that end, we extract $x$ of shortest and longest names from the sorted copy to form our experiment global dataset, $D_S$. We call this technique *Data Riving* (in the spirit of splitting wood along the grain) and $x$ as the adjoining riving factor. The resultant refined dataset is capable of representing its full sized original dataset, precisely in context with our SIC model. This optimization was important for conducting experiments across all datasets described in Section 4.4 in timely manner.

## 4.6 Execution Environment

This section outlines the list of computational resources utilized and the programming languages used during the implementation of the proposed cryptanalysis work in this research. For implementing Phase 1 of the SIC, the cluster computing facilities provided by SHARCNET [2] was utilized, specifically the Saw and Orca clusters, for processing numerous serial jobs. SHARCNET consists of a consortium of colleges, universities and research institutes operating a network of high-performance computer clusters across south western, central and northern Ontario, Canada. Phase 2 of the SIC and other tasks related with data preprocessing were run on a 2.6GHz quad-core i7 system with 16GB of memory. The SIC software was implemented in C++ programming language while the Dice dictionary creation and other preprocessing tasks were implemented in scripting language, Perl. Some additional code was also written to allow the filtering to be serialized for each BFE across each dataset. A basic shell script handled running the 2,400 total jobs on SHARCNET cluster. The above mentioned implementation details have been further summarized in Table 4.2.

| **Module** | **Implementation** |
|---|---|
| Preprocessing | Perl, C++ |
| Attack Phases of SIC | C++ |
| Execution Environment | 2.6 GHz quad-core i7 desktop (16 GB RAM);<br>SHARCNET for parallel processing |
| SHARCNET Jobs | Shell Scripts |

**Table 4.2: The summary of the Implementation details for the attack modules and related experiments.**

## 4.7 Source Code

The source code from this thesis project including SIC implementation is made publicly available and can be accessed from the following git repository:

`https://github.com/VasundharaSharma/`

## 4.8 Summary

This chapter presented the implementation details of SIC as well as a discussion on the choice and selection of the various datasets for the experiments. These datasets would be referenced throughout the discussion on experiment evaluation and result analysis. Therefore, this chapter helps in understanding the specifics of the datasets as well as justifies the choice of various parameter selections during BFE generation. Further, a discussion on data optimization technique, *Data Riving* used during the experiments was also outlined in this chapter.

# Chapter 5

# Results and Analysis

This chapter presents the comprehensive evaluation of the proposed SIC framework. Several experiments are conducted to evaluate: (1) the correctness of record matching, (2) computational complexity, and (3) the underlying security. A publicly available dataset, the North Carolina voter registry (NCVR) [6], is used to evaluate the trade-offs between the aforementioned criterion. Eight variants of datasets are created using different parameters during BFE construction as discussed in Section 4.4. For the ease of analysis, these datasets are further categorized into 4 groups based on the maximum allowed overlaps. The performance of SIC is chiefly evaluated by measuring the extent of anonymity provided by encoding under different conditions. A separate section is devoted to the analysis of each of the two phases of SIC. In addition, an independent analysis measuring the integrity of the matched outcome is performed to determine the correlation between the security and accuracy of the BFE-based PRL technique.

## 5.1 Degree of Anonymity Evaluation after Filtering Phase

The performance evaluation of SIC after Phase 1 is based on the implementation of the filtering module of the attack as explained in Section 3.4.1. This section specifically describes the re-identification risk analysis for BFEs on completion of the filtering phase of SIC. In addition, a

combined comparison of the results across different datasets is presented to study the effect of adding extra data attributes in BFE construction.

**Analysis** For a given BFE, its Degree of Anonymity (*DoA*) is measured on a scale of [0 - 1] where $DoA = 0$ marks the complete re-identification of the personal information associated with the BFE, and $DoA = 1$ represents the state of complete anonymity within a global anonymity set. At the start of the experiment, all BFEs are assumed to be completely anonymous within the given global dataset. Therefore, upon inception all BFEs have a $DoA = 1$. Phase 1 (filtering phase) of SIC is mounted and *DoA* for each BFE is collected upon its completion. *DoA* is measured using the expression in equation (4.2). Based on the results, in 7 of the 8 datasets, the anonymity sets for a couple of BFEs were reduced to singleton, thus reducing their *DoA* to 0.

The average *DoA* for the entire dataset is then calculated using equation (4.3). Figure 5.1 presents the numbers for the average *DoA*, obtained after the completion of filtering phase of SIC. It is worth noticing here that none of the BFEs (across all datasets) report *DoA* above 0.5. The highest value for the average *DoA* obtained is 0.406 for the 3-attributes dataset with unrestricted overlaps, while the datasets with overlap=1 reported an extremely low value of 0.066 as the average *DoA*. Figure 5.1 also presents the pair-wise analysis for the variations in *DoA* across datasets with 2-attributes and 3-attributes after the completion of filtering phase of SIC. The height of the bars represents the average *DoA* of 300 BFEs in each of the datasets with 95% confidence interval bars as shown. These subtle variations show that each of the datasets with equivalent max overlaps report only minor differences in their respective degrees of anonymity, suggesting that "chaffing" the BFE with an additional identifier did not have a substantial effect on security enhancement.

Each of the datasets used in the experiments is further examined on an individual basis. These datasets are broadly classified into 4 groups, each with different overlap restrictions imposed in BFE construction. Each of these datasets hold the collection of both 2-attributes (*i.e., firstname, lastname*) and 3-attributes (*i.e., firstname, lastname, zipcode*) datasets.

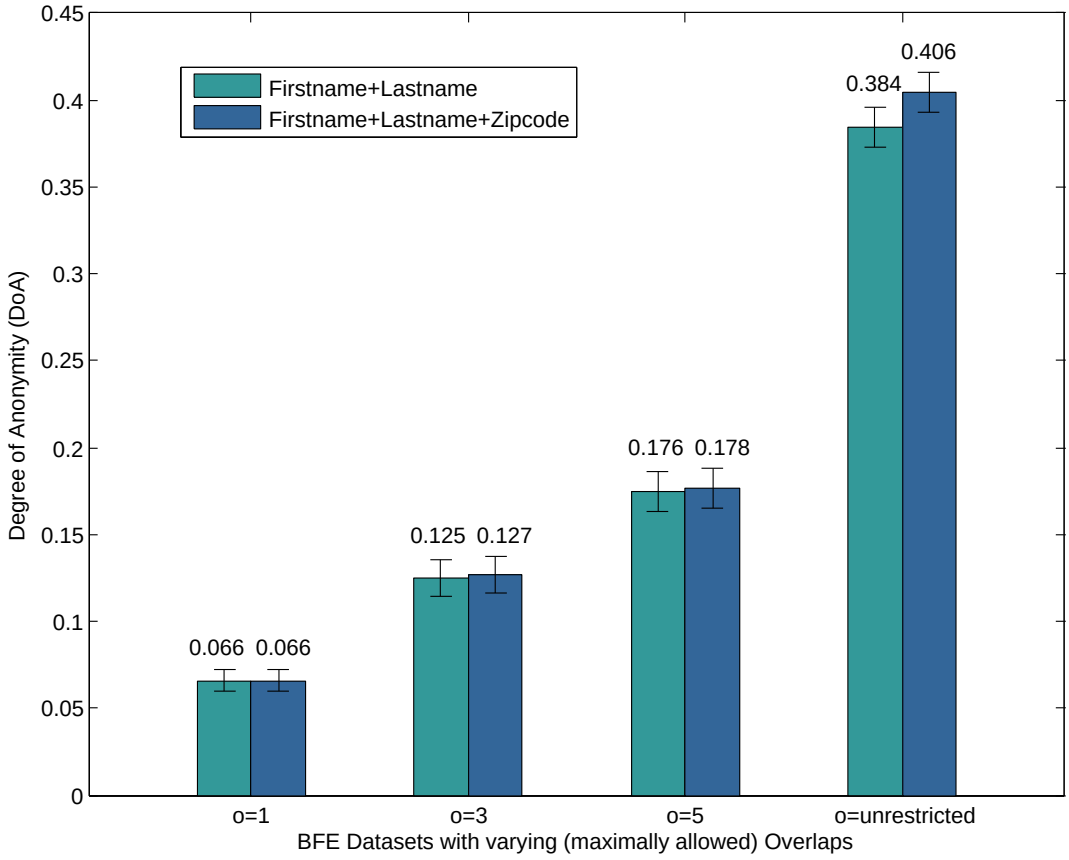



**Figure 5.1: Average degree of anonymity (DoA) after Phase 1 of SIC.** The average DoA is measured with standard 95% confidence interval across each of the 8 datasets to measure the remaining anonymity after Phase 1 (Filtering phase) of SIC.

### 5.1.1 Dataset 1

Dataset 1 represents the BFE construction with maximum allowed overlaps, $o$=1. The average DoA was reported to be the lowest at 0.066 for both 2-attributes and 3-attributes datasets. Therefore, there was no noticeable change in anonymity with the addition of extra identifiers in BFE input string. For the given set of 300 BFEs, at the end of Phase, 1 BFE was fully re-identified in the 2-attributes dataset. In addition, at least 8 BFEs had the possibility set (PS) size reduced to $\leq$ 5 records. 17 BFEs were left with $PS \leq 10$ records and 35 BFEs had $PS \leq 20$ records. Similarly, for the 3-attribute dataset, 2 BFEs were fully re-identified, 7 BFEs were left with $PS \leq 5$, 14 BFEs with $PS \leq 10$ and 32 BFEs with $PS \leq 20$ records.

### 5.1.2 Dataset 2

Dataset 2 represents the BFE construction with maximum allowed overlaps, $o$=3. The average DoA was reported as 0.125 for the 2-attributes and 0.127 for 3-attributes datasets respectively. The improvement in anonymity with adding extra identifier is apparently negligible. For the given set of 300 BFEs, at the end of Phase, 1 BFEs was fully re-identified in 2-attributes dataset. In addition, 3 BFEs had the possibility set (PS) size reduced to $\leq$ 5 records. 5 BFEs were left with $PS \leq 10$ records and 13 BFEs had $PS \leq 20$ records. Results were quite similar for 3-attributes dataset, where 1 BFE was completely re-identified, 3 BFEs were left with $PS \leq 5$, 4 BFEs with $PS \leq 10$ and 11 BFEs with $PS \leq 20$ records.

### 5.1.3 Dataset 3

Dataset 3 represents the BFE construction with maximum allowed overlaps, $o$=5. The average DoA was reported as 0.176 for the 2-attributes and 0.178 for 3-attributes datasets respectively. The improvement in anonymity with adding extra identifier is apparently negligible. For the given set of 300 BFEs, at the end of Phase, 1 BFE was completely re-identified in both 2-attributes and 3-attribute datasets. Overall, only 3 BFEs were identified with $PS \leq 20$ records.

### 5.1.4 Dataset 4

Dataset 4 represents the BFE construction with no overlap restrictions, *i.e.,* $o$=$\infty$. The average DoA was reported as 0.384 for the 2-attributes dataset. With the inclusion of another attributes (*i.e.,*zipcode), the average DoA went up to 0.406—an increase of less than 5%. For the given set of 300 BFEs, at the end of Phase, 1 BFE was completely re-identified in both 2-attributes and 3-attribute datasets, however, not many BFEs were obtained with $PS \leq 20$ records.

### 5.1.5 Discussion

In summary, the DoA analysis across different datasets (with identical overlap-restricted BFE construction) clearly indicate that there is no significant security improvement with "chaffing" the BFEs with extra identifiers. However, it is observed that there is a significant increase in anonymity with the increase in the maximum allowable overlaps. The difference on average DoA for 2-attributes datasets with overlaps=1 and overlaps=unrestricted stands at 31% and 34% for the 3-attributes datasets. It may appear here that the security of BFEs can be improved by allowing more overlaps. This anonymity gain is coupled, however, with a steep decline in accuracy which is discussed in more details with the accuracy evaluation in Section 5.3.

## 5.2 Degree of Anonymity Evaluation after Trim Phase

The evaluation in the previous section attempted to provide insight into the extent of anonymity provided by BFE techniques. In this section, the effect of known plaintext information on the anonymity of the remaining BFEs is studied. As discussed in Section 3.4.2, the results from the Phase 1 of the attack are utilized as input to the Phase 2. The performance evaluation of SIC during Phase 2 is based on the implementation of the trim module of the attack as explained earlier in Section 3.4.2.

**Analysis** Proceeding to the trimming phase of the attack, each recovered BFE-plaintext information is fed back in to the system and the average DoA is recorded at each step. New revealed BFE-plaintext pairs obtained during the process are also fed back into the system. This process continues until a steady state is achieved where either all known BFEs are exhausted or the remaining BFEs are completely re-identified. Given the very small number of actual revealed plaintexts from the previous step, in the interest of understanding how the anonymity of BFEs fail, we extend the experiment by introducing some known BFE-plaintext pairs which are obtained from external sources as explained earlier in Section 3.4.2.

The DoA declines at different rates depending on the order of names in the revealed list *RP*, therefore, multiple iterations of the experiment have been considered. The current set of experiments perform 30 iterations in total where each iteration introduces a completely fresh and randomized sequence of BFE-plaintext pairs in to the trim module. The average of these 30 iterations is computed to analyze the behavior of the system. The above mentioned experiment is repeated for each of the 8 datasets. At the start of the experiment, the value of the average DoA is retained from the Phase 1 for each of the datasets respectively.

For clarity of analysis, each of the datasets used in the experiments is further examined on an individual basis. These datasets are broadly classified into 4 groups with different overlap restrictions imposed in BFE construction. Each of these datasets hold the collection of input strings with both 2-attributes (*i.e., firstname, lastname*) and 3-attributes (*i.e., firstname, lastname, zipcode*). A graph is plotted for visualizing the average remaining *DoA* (plotted against the Y-axis) at each step of induction of a known BFE-plaintext pair (plotted against the X-axis).

### 5.2.1 Dataset 1

Dataset 1 represents the BFE construction with maximum allowed overlaps, $o$=1. The average DoA at start is 0.06 for both the 2-attributes and 3-attributes datasets as obtained from the results of the filtering phase experiment. Figure 5.2 shows the results obtained during Phase 2 for both the datasets. The steep and downwards stepping pattern in the figure represents the quick depletion of the remaining degree of anonymity in the BFE system. Moreover, it is worth noticing here that both the 2-attributes and the 3-attributes datasets are showing almost persistent descending trend, with 2-attributes dataset getting completely re-identified at the introduction of $139^{th}$ element (*i.e.,* BFE-plaintext pair) and the 3-attributes dataset following sooner at $160^{th}$ element. Another observation worth mentioning here is that lot of these elements were internally generated during the course of execution of Phase 2 of SIC. The results obtained with these datasets helps in reinforcing our research hypothesis that not much security is introduced with"chaffing" extra parameters into the BFE construction.

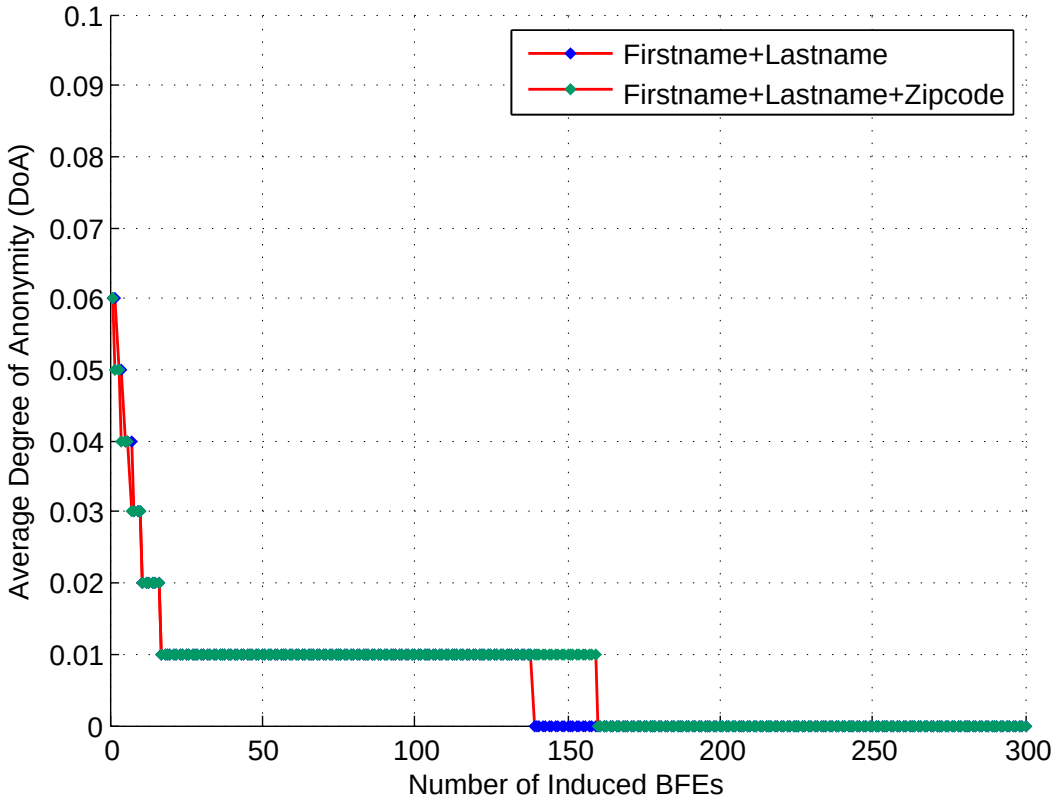



**Figure 5.2: Average DoA during trim phase of the SIC for BFE datasets with maximally allowed overlaps, $o$=1.** Each point in this graph represents the average *DoA* for remaining BFEs in given datasets.

### 5.2.2 Dataset 2

Dataset 2 represents the BFE construction with maximum allowed overlaps, $o$=3. The average DoA at start is 0.12 (rounded) for both the 2-attributes and 3-attributes datasets as obtained from the results of the filtering phase experiment. Figure 5.3 shows the results obtained during Phase 2 for both the datasets which show similar descending trend as seen earlier in the results for datasets with maximum overlaps=1. Dataset with 2-attributes gets completly re-identified with the introduction of $191^{th}$ element while the 3-attributes dataset stretches up to $269^{th}$ element. It can be noticed here that with relaxation in BFE construction by allowing more overlaps, however, hardens the re-identification process to some extent as more known plaintexts are required in order to crack the complete list of BFEs.

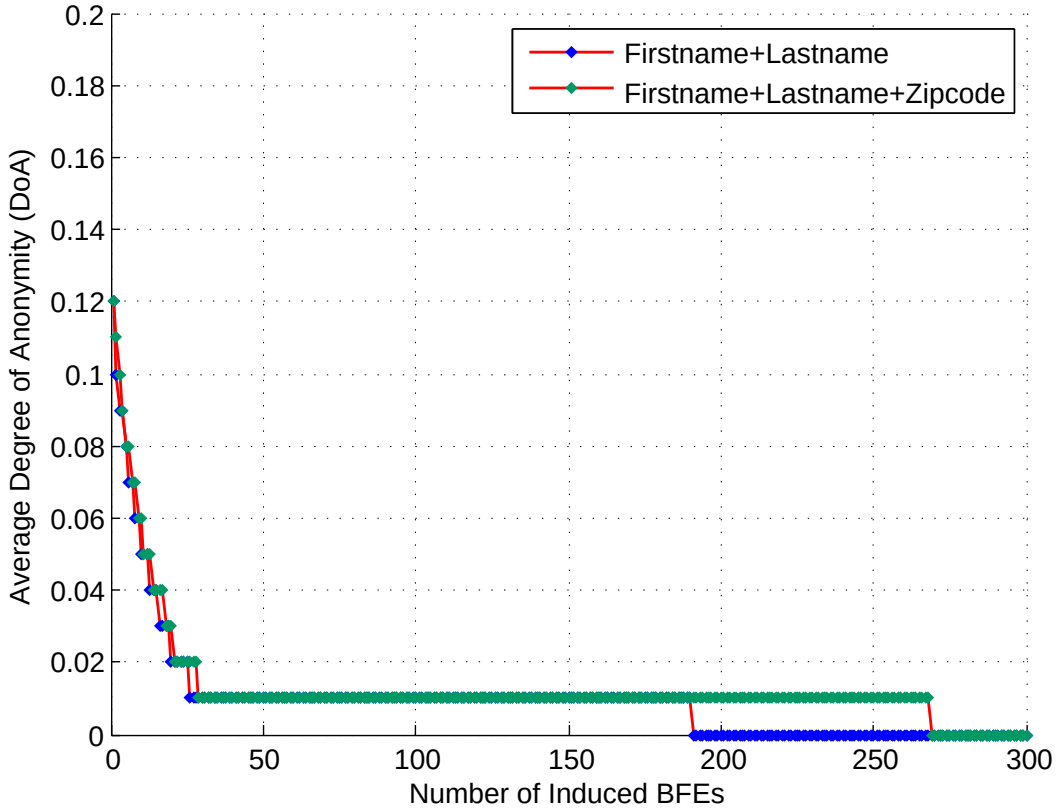



**Figure 5.3: Average DoA during trim phase of SIC for BFE datasets with maximally allowed overlaps, $o$=3.** Each point in this graph represents the average *DoA* for remaining BFEs in given datasets.

### 5.2.3 Dataset 3

Dataset 3 represents the BFE construction with maximum allowed overlaps, $o$=5. The average DoA at start is 0.17 (rounded) for both the 2-attributes and 3-attributes datasets as obtained from the results of the filtering phase experiment. Figure 5.4 shows the results obtained during Phase 2 for both the datasets which show the descending trend with slightly hardened re-identification process as compared to the previous datasets with maximum allowed overlaps, $o$=3. Also, a couple of BFEs remain un-identified with $DoA = 0.01$ towards the end of Phase 2 experiment. The size of the possibility sets for such BFEs, however, is $PS \leq 5$ which in a way is considered equivalent to being completely identified by certain anonymity precedents (refer Section 2.1.3 on re-identifcation risk analysis).

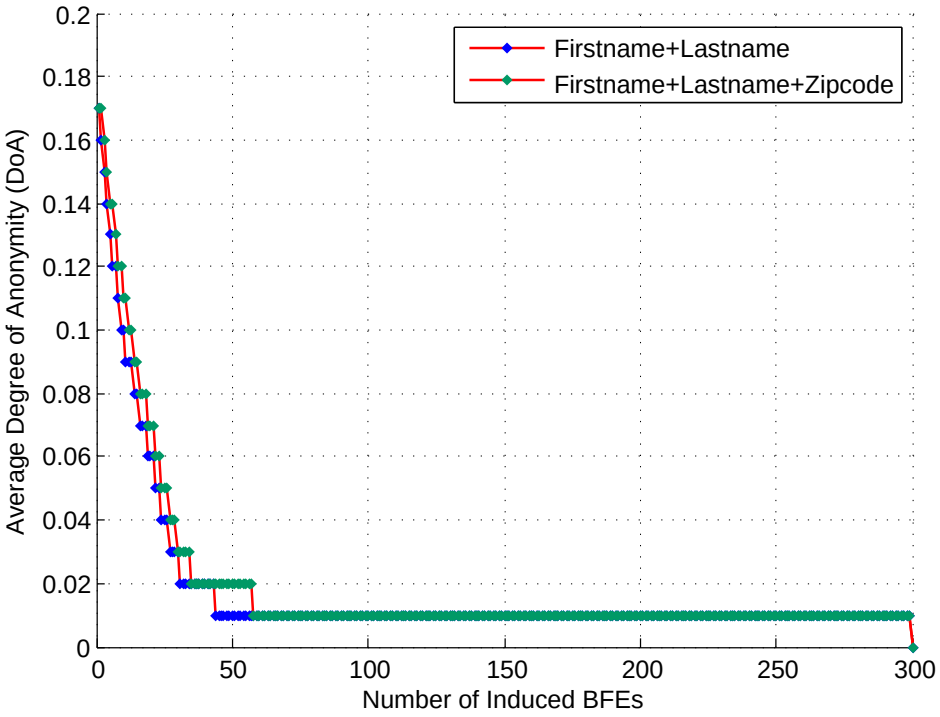



**Figure 5.4: Average DoA during trim phase of SIC for BFE datasets with maximally allowed overlaps, $o$=5.** Each point in this graph represents the average *DoA* for remaining BFEs in given datasets.

### 5.2.4 Dataset 4

Dataset 4 represents the BFE construction with no overlap restrictions, *i.e.,* $o$=$\infty$. The choice of datasets with unrestricted overlaps is primarily influenced by the work of Kuzu *et al.* [64]. Also, this happens to be the most basic form of BFE construction in a PRL setting, hence widely adopted in literature. The average DoA at start is 0.38 (rounded) for the 2-attributes dataset and 0.41 for the 3-attributes datasets as obtained from the results of the filtering phase experiment. Figure 5.5 shows the results obtained during Phase 2 for these two datasets with unrestricted overlaps. The descending trend of DoA with the addition of new known BFE-plaintext pairs is observed here as well. Additionally, we see that the points in the graph (depicting the average remaining *DoA*) for datasets with 2-attributes versus 3-attributes marginally differ from each other which again reiterates the fact that not much security element is added to the system with increasing attributes. The re-identification process, at the same time, is harder compared to

previous settings with lesser number of allowed overlaps. Few BFEs remain un-identified with $PS \leq 20$ at the completion of Phase 2.

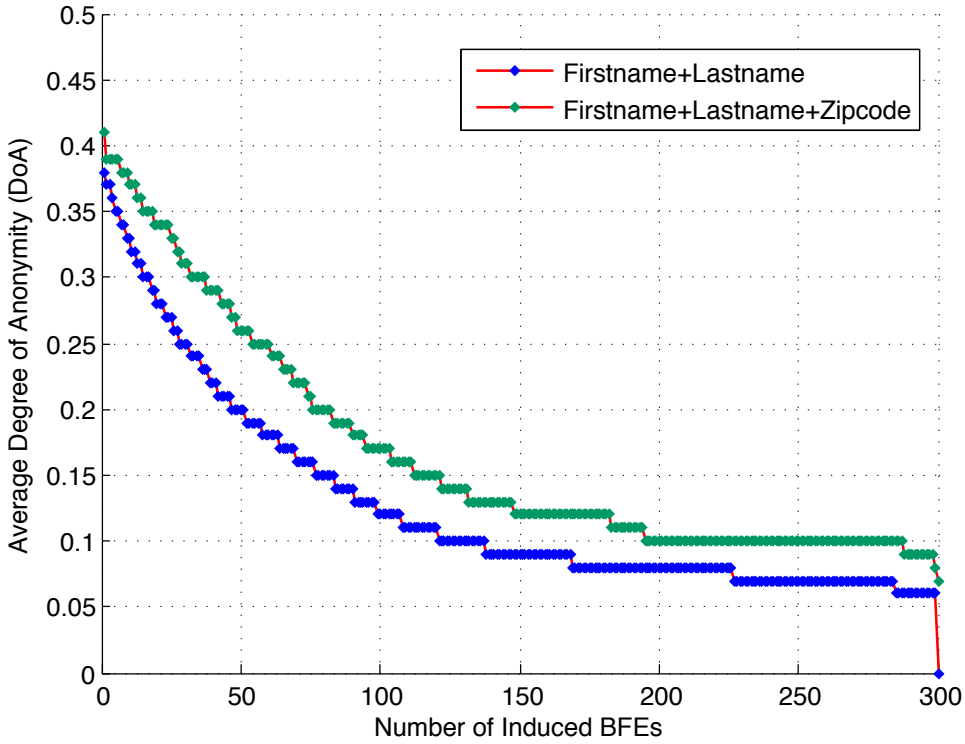



**Figure 5.5: Average DoA during trim phase of SIC for BFE datasets with unrestricted overlaps.** Each point in this graph represents the average *DoA* for remaining BFEs in given datasets.

### 5.2.5 Discussion

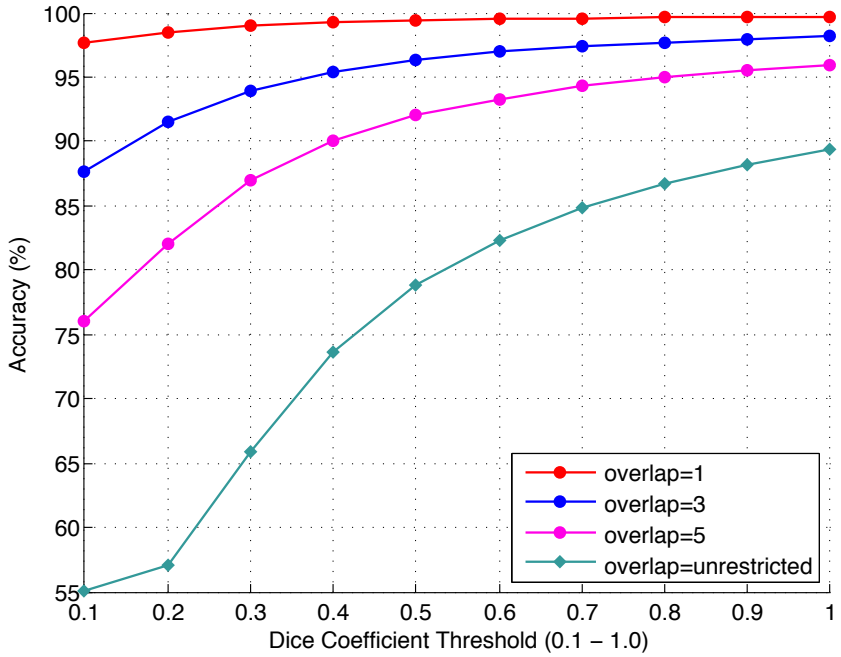
In summary, it can be observed that the rate of decline of DoA in the datasets with $o = 1, 3, 5$ is, as might be expected, much steeper and thus much more susceptible to known plaintext information. For instance, in the case of $o = 1$ datasets with {*firstname, lastname*} attributes and 300 BFEs in total, only 22 known plaintexts were required to re-identify 50% of the BFE database; and after introducing 139 known plaintexts, the target database was fully re-identified. Therefore, the results obtained from Phase 2 are promising as they demonstrate the fragility of the Bloom filter encoding system in presence of few known plaintexts while reinforcing our research hypothesis that not much security element is added by "chaffing" extra data attributes into the BFE construction.

## 5.3 BFE Accuracy Evaluation

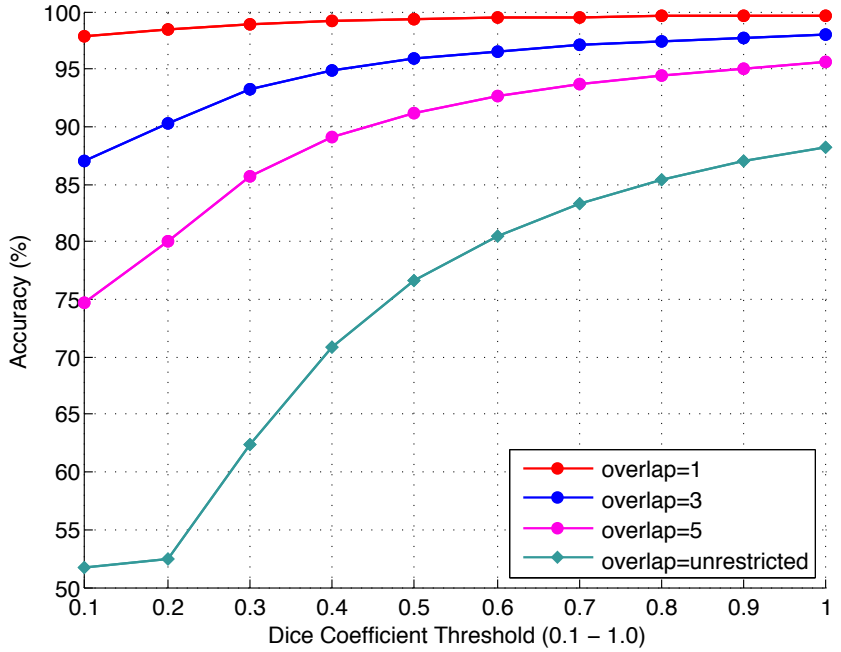
The SIC evaluation results from Sections 5.1 and 5.2 so far show that BFEs tend to perform relatively better in terms of security (derived from DoA measure) with unrestricted overlaps allowed into the encoding system. The current research, however, hypothesizes that this addition of security is achieved at the cost of accuracy. In order to test this hypothesis, accuracy evaluation is performed on BFE datasets with 2-attributes and 3-attributes with different overlap restrictions. The equation for measuring accuracy is derived based on correct and incorrect matches as stated earlier in Section 4.3.1. This experiment is conducted in isolation without any involvement or dependency on the proposed SIC. All flavors of datasets are tested at different Dice coefficient threshold $t$, with values ranging from 0.1 to 1.0, in increments of 0.1 at each stage of the experiment. The results are segregated into two parts, first for 2-attributes datasets with all variations of overlap restrictions and similarly second for the 3-attributes datasets. Collectively, the results shown in shown in Figure 5.6a and 5.6b, clearly indicate that accuracy decreases considerably with the increase in number of allowed overlaps. A more detailed analysis is presented for each of the datasets in the next section.

### 5.3.1 Dataset 1

Dataset 1 represents the collection of BFEs with 2-attributes and different overlap settings (*i.e.,* $o = 1, 3, 5, \infty$). Examining the "BFE-to-plaintext" matching accuracy for these datasets, as shown in Figure 5.6a, we see that while accuracy for dataset with maximum overlap=1 stands at 99.64% for threshold $t = 0.9$, the same drops down to 88.17% in dataset with maximum overlaps=$\infty$. This can be inferred as "BFE technique delivering 11.5% inaccurate results (with high probability) in attempt to make it more anonymous", which is fairly unacceptable. Further, the trend of the graphs also indicate a steady decline in accuracy with the increase in overlaps. Also, these gaps get wider with the increase in overlap relaxation, for instance, at matching threshold, $t = 0.9$, the accuracy for $o = (1, 3, 5, \infty)$ drops from 99.68% to 97.94% to 95.53%




**(a)** Comparison for BFE prediction Accuracy on {firstname+lastname} datasets with different overlap counts




**(b)** Comparison for BFE prediction Accuracy on {firstname+lastname+zipcode} datasets with different overlap counts

**Figure 5.6: Accuracy evaluation for BFE method with different attributes and parameter settings.** The BFE match accuracy is evaluated against the corresponding plaintexts based on the Dice similarity comparison for the different threshold values in the range [0.1 - 1.0] with increments of 0.1 at each stage.

and ultimately to 88.17%.

### 5.3.2 Dataset 2

Dataset 2 represents the collection of BFEs with 3-attributes and different overlap settings (*i.e.,* $o = 1, 3, 5, \infty$). The results shown in Figure 5.6b indicate similar declining trends of accuracy with increase in overlap relaxation as observed earlier on datasets with 2-attributes. Further, comparing these graphs with those in Figure 5.6a, we do not see any considerable improvement with the addition of extra attributes. In fact, the accuracy goes down further by another 1% in 3-attributes dataset with overlap=$\infty$. Since these results are clear indicative of a trade-off happening between security and accuracy, the accuracy is further examined against the previously obtained anonymity metrics.

Considering the dataset with 3-attributes and $o = 1$ which is sparse by nature and closest to accuracy among all others sets, we attempt to examine its DoA as obtained earlier in Section 5.1 with the assumption that the encoding transformation would be close to its real match in terms of number of bigrams. However, we notice that for $o = 1$ case, the average DoA was lowest at 0.06, and the accuracy highest ( 99% at $t = 0.9$). Correspondingly for the unrestricted overlaps case, the average DoA was highest at 0.38-0.41, while accuracy was lowest ( 85% at $t = 0.8$). The cause of both the increase in anonymity and decrease in accuracy arise from increasing overlaps distorting length and intersection estimation.

## 5.4 Summary

In this chapter, evaluation for the proposed SIC was presented. The performance of SIC was evaluated against 8 different datasets derived from the publicly-available dataset, the North Carolina voters registry (NCVR) [6] during the experiment design (explained in Section 4.4). Each of these datasets were crafted such that it would assist in analyzing the effects of using different BFE parameters during BFE construction on the overall accuracy of outcome as well

to measure the extent of security offered in return. The entire experiment coverage was extensive and consumed over 1 CPU-year of computation (based on the usage statistics obtained from the SHARCNET clusters).

Degree of Anonymity was employed as the main evaluation metric for measuring the performance of SIC in each of the two phases. The results after Phase 1 suggested the basic low level of security provided by BFE technique. Phase 2 analysis at the same time, demonstrated the fragility of the BFE technique in presence of known information. Additionally, an independent accuracy analysis was conducted to validate the hypothesis that the apparent increase in security by *chaffing* extra attributes into the Bloom filter (as suggested in the literature) is only obtained by sacrificing accuracy. It could be inferred from the accuracy evaluation that "chaffing" strategy will only lead to higher false positive rates, and in turn lower matching accuracy. Surprisingly, we also found that although BFEs themselves are not susceptible to false negatives, $Dice_{bfe}$ (*i.e.,* the Dice coefficient between two BFEs) starts rendering false negatives as the filter gets denser. The results, therefore indicate that, allowing unrestricted overlaps into the BFE generation significantly affects the performance.

The various experiments sum up to one notable conclusion, that the private string comparators that tokenize, encode, and compare strings yield highly accurate record linkage results only with restricted BFE construction. Otherwise, it is observed that the use of BFE in context of PRL, practically, does not provides both accuracy and security.

# Chapter 6

# Conclusion and Future Work

This chapter provides the summary of notable accomplishments of this thesis work which primarily focuses on outlining some of the significant security issues associated with BFE-based private record linkage. In that direction, empirical investigations of the proposed Set Intersection-based Cryptanalysis (SIC) work is outlined. Additionally, discussion on the current limitations associated with the SIC framework and possible directions for future works has been included.

## 6.1 Discussion and Conclusion

By virtue of this thesis work, several security issues present in the BFE-based private record linkage has been revisited. Kuzu *et al.* [64] were among the first to cryptanalyze basic Bloom filter-based records matching using constraint satisfaction techniques followed by other related efforts by Kroll and Steinmetzer [61] and Niedermeyer *et al.* [75]. However, little research has compared Bloom filter encoding techniques from the view point of concurrent accuracy and security analysis in the context of a real record linkage application. Therefore, although the work in this thesis is not the first work towards the outbreak of cryptanalysis on BFEs, it is novel in its kind to present the formal security analysis with focus on combined comparative analysis of both accuracy and security aspects of BFE technique.

The major contributions from this research work and the underlying conclusive remarks are briefly outlined as follows:

- Chosen plaintext attack (CPA) game-based formal security analysis for the Bloom filter encoding technique is presented in this thesis. This analysis shows that BFEs do not adhere to the basic acceptable levels of security. Bloom filter encodings are deterministic in nature, a property which can be associated to its low security standards. In some sense this is not unexpected: a human-being could reasonably distinguish BFEs by eye. Therefore, the clear message delivered through this research work is that BFE-based PRL techniques offer no formal security guarantees. BFEs can only offer heuristic security for which, fundamentally, there are no guarantees.

- A new class of set intersection-based cryptanalysis (SIC) is proposed in this thesis. With the implementation and evaluation of the SIC framework, we further attempt to validate our research hypothesis that BFEs are not secure against chosen plaintext attack. The results obtained from the Phase 2 experiments of SIC are quite promising in this direction. With these observations in mind, we believe that additional work needs to be done before BFEs are deployed on real-world patient data.

- The software implementation of the proposed SIC framework is presented with this thesis. The complete framework for SIC has been developed in C++ programming language along with some pre-processing steps utilizing Perl scripting language. Two separate versions of the source code has been released which gives the flexibility of sequential execution (across single core platforms) or parallel execution (across multi-core platforms) of the attack. In addition, some automation work has been done for the experiment validations.

- Analysis of the results obtained from SIC also shows that recently proposed BFE security enhancements involving "chaffing" filters with extra data, do not greatly increase anonymity, but ultimately do so at the expense of accuracy. While its clear that "chaffing"

strategy will lead to higher false positive rates, and in turn lower matching accuracy, what's not clear is how or by "how much" security is improved. Using the anonymity set as a quantifiable metric for security, to quantify the relation between security and accuracy, we experiment with different variations of a dataset by controlling for accuracy. On the basis of the empirical analysis of accuracy, it can be concluded that there is an inverse relationship between the security and accuracy of BFEs *i.e.,* BFEs derive security only with loss of information.

- Last but not least, the design of the SIC framework is flexible and scalable and provides several design choices, for instance, hand-picking the BFE construction parameters, selection and sizing of datasets *etc.,* which users can adapt to best suit their requirements while constructing BFEs. SIC framework can be easily extended to accommodate future enhancements.

## 6.2 Limitations and Future Work

The current limitations of the proposed SIC framework is discussed at first. In the remainder of this section, some thoughts are shared on the prospective future works and open research objectives.

### 6.2.1 Limitations

Certain limitations associated with the current implementation of SIC framework is discussed as follows:

- While designing the framework for SIC, it is assumed that the adversary possesses adequate and confirmed knowledge about the global dataset from which the BFEs are generated. However, in practice, it may be difficult for the adversary to obtain this information to a certain extent. However, this is a generic assumption which is considered in other related works as well.

- Despite all merits of the proposed SIC framework, one main limitation associated with it is the fact that filtering phase was unable to generate a critical mass of known plaintexts on its own. The potential for future improvements rests in the reduced anonymity sets. The results of other attacks (*e.g.,* Niedermeyer *et al.* [75]) ultimately serve as a workaround in the current setting.

### 6.2.2 Future Work

On the basis of the work presented in this thesis, the avenues for future work can be explored in two different directions. First, extension of the current SIC framework can be explored. Second, efforts can be invested into devising secure yet efficient alternatives systems for PRL. This section briefly describes and outlines the collective thoughts on the possible future endeavors in both the directions.

- The current framework for SIC can be extended to include additional metrics for the filtering module. This would possibly help in retrieving more singletons during the filtering phase of the attack and hence boost up the overall performance of the cryptanalysis by eliminating the dependency on external sources for obtaining a critical mass of known plaintexts.

- The processing time for the complete cycle of execution of SIC has been a performance bottleneck. The filtering module has to loop through quadratic comparisons which adds to the runtime complexity of the module. In the current implementation, the proof of concept was validated by utilizing a subset of the entire dataset. Therefore, performance improvements can be taken up as future work which would allow us to utilize the datasets in their full capacity.

- Another possible extension for the cryptanalysis work would be to implement a separate module to obtain “bigram-plaintext” mappings for the entire set of all possible plaintext

bigrams. This module would act as a consolidated lookup table and hence serve as a force multiplier in re-identifying all possible new BFEs (with high probability) in real-time.

- Other related recent works on cryptanalysis of BFEs [61, 75] variously suggest hardening BFE construction using salting, removing longer names from attribute sets, injecting random bits, *etc.* as a countermeasure to make BFEs resilient to attacks. Ultimately, however, the current work in this thesis suggests that these measures cannot bring BFEs to formal security levels. Exception, of course, would be by sacrificing accuracy to the extreme. For example, a BFE scheme mapping any input to the same bit pattern (*e.g.,* all 1's) would be IND-CPA as well as information-theoretically secure, as all information about the corresponding plaintext would have been lost. Therefore, instead of succumbing to the weak heuristic security offered by BFEs and attempting to fix it, future initiatives should be focused towards designing IND-EAV[1] secure encoding systems for PRL, for instance, using Cuckoo hashing [45].

In conclusion, while there is room for the aforementioned possible future works for the SIC, this thesis serves as a starting point and proof of concept for one method to bring out the formal security analysis of BFE technique and bring out the security weaknesses which are inherently present. Moreover, these weaknesses are not likely fixable beyond a certain extent. Therefore, use of BFE-based PRL systems, especially in health informatics domain should be discouraged. New research into highly efficient secure multi-party approaches to approximate matching is needed toward providing a compelling alternative for private record linkage.

[1] Secure systems which can offer Indistinguishability under Eavesdropping Attack

# Bibliography


[1] Guidelines for Working with Small Numbers. Colorado Department of Public Health and Environment. Available at: http://www.cohid.dphe.state.co.us/smnumguidelines.html. Last Accessed: 2015-04-13.

[2] Shared hierarchical academic research computing network (SHARCNET). Available at: https://www.sharcnet.ca Last Accessed: 2015-05-24.

[3] NCHS Staff Manual on Confidentiality. National Center for Health Statistics. Available at: http://www.cdc.gov/nchs/data/misc/staffmanual2004.pdf, 2004. Last Accessed: 2015-06-02.

[4] Policy for Disclosure of Reportable Disease Information. Iowa Department of Public Health. Available at: http://www.idph.state.ia.us/IDPHChannelsService/file.ashx?file=1F4162C0-DDEC-4188-86C5-3330F859183A, 2005. Last Accessed: 2015-06-02.

[5] Canadian Institutes of Health Research Policy on access to research outputs. Available at: http://www.cihr-irsc.gc.ca/e/34846.html2007.Archivedat:http://www.webcitation.org/5XgxgoBzj, 2008. Last Accessed: 2015-06-02.

[6] North carolina voter registration database. Available at: http://www.ncsbe.gov/ncsbe/ 2011. Last Accessed: 2015-05-24.

[7] U.S. food and drug administration's sentinel initiative. Available at: http://www.fda.gov/Safety/FDAsSentinelInitiative/ucm2007250.htm 2015. Last Accessed: 2015-05-24.

[8] R. Agrawal, A. Evfimievski, and R. Srikant. Information sharing across private databases. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 86–97, 2003.

[9] M. Atallah, F. Kerschbaum, and W. Du. Secure and private sequence comparisons. In *Proceedings of the ACM Workshop on Privacy in the Electronic Society*, pages 39–44, Washington DC, 2014.

[10] T. Bachteler, R. Schnell, and J. Reiher. An Empirical Comparison of Approaches to Approximate String Matching in Private Record Linkage. In *Proceedings of Statistics Canada Symposium 2010*, 2010.

[11] C. Batini and M. Scannapieco. *Data Quality: Concepts, Methodologies and Techniques*. Springer., 2006.

[12] R. Baxter, P. Christen, and T. Churches. A comparison of fast blocking methods for record linkage. In *ACM SIGKDD Workshop on Data Cleaning, Record Linkage and Object Consolidation*, Washington DC, 2003. ACM.

[13] Z. Bellahsene, A. Bonifati, and E. Rahm. *Schema Matching and Mapping*. Springer., 2011. ISBN: 978-3-642-16517-7 (Print) 978-3-642-16518-4 (Online).

[14] M. Bellare, R. Canetti, and H. Krawczyk. Keying Hash Functions for Message Authentication. In *Proceedings of the 16th Annual International Cryptology Conference: 18 -22 August*, pages 1–15. Advances in Cryptology CRYPTO, Springer Berlin Heidelberg, 1996.

[15] J. Berman. Zero-check: a zero-knowledge protocol for reconciling patient identities across institutions. In *Arch Pathol Lab Med*, volume 128 of *3*, pages 344–346, 2004.

[16] M. Bilenko and R. Mooney. Adaptive duplicate detection using learnable string similarity measures. In *ACM SIGKDD*, pages 39–48, Las Vegas, 2003. ACM.

[17] J. Bleiholder and F. Naumann. Data fusion. *ACM Computing Surveys*, 41(1):pp. 1–41, December 2008.

[18] B. Bloom. Spacetime trade-offs in hash coding with allowable errors. In *Communications of the ACM*, volume 13, pages 422–426. 1970.

[19] A. Broder and M. Mitzenmacher. Network Applications of Bloom Filters: A Survey. *Internet Mathematics*, 1(4):pp. 485–509, 2004.

[20] Statistics Canada. Record linkage program. Available at: `http://www.statcan.gc.ca/eng/health/link`. Last Accessed: 2015-05-24.

[21] P. Carpenter and C. Chute. The universal patient identifier: a discussion and proposal. In *Proceedings of the American Medical Informatics Association Annual Symposium*, pages 49–53, 1993.

[22] A. Cavoukian. Privacy concerns in preventing fraudulent publication. *CMAJ*, 175(1):pp. 61–62, 2006.

[23] P. Christen. A comparison of personal name matching:techniques and practical issues. In *Sixth IEEE International Conference on Mining Complex Data - IEEE ICDM Workshop*, pages 290–294, Hong Kong, 2006. IEEE.

[24] P. Christen. Automatic record linkage using seeded nearest neighbour and support vector machine classification. In *ACM SIGKDD*. ACM, 2008.

[25] P. Christen. *Data Matching, Data-Centric Systems and Applications*. Springer., 2012.

[26] P. Christen. A Survey of Indexing Techniques for Scalable Record Linkage and Deduplication. *IEEE Transactions on Knowledge and Data Engineering*, 24(9):pp. 1537–1555, September 2012.

[27] T. Churches and P. Christen. Some methods for blindfolded record linkage. 4(9), 2004.

[28] T. Churches, P. Christen, K. Lim, and J. X. Zhu. Preparation of name and address data for record linkage using hidden Markov models. *BioMed Central Medical Informatics and Decision Making*, 2(9), 2002.

[29] W. Cohen. Data integration using similarity joins and a word-based information representation language. *ACM Transactions on Information Systems*, 18(3):pp. 288–321, 2008.

[30] W. Cohen and J. Richman. Learning to match and cluster large high-dimensional datasets for data integration. In *ACM SIGKDD*, Edmonton, 2002. ACM.

[31] UK Medical Research Council. MRC Policy on Data Sharing and Preservation, 2006.

[32] U.S. Department of Health & Human Services. HIPAA privacy rule. Available at: `http://www.hhs.gov/ocr/privacy/`. Last Accessed: 2015-05-24.

[33] C. Díaz, S. Seys, J. Claessens, and B. Preneel. Towards Measuring Anonymity. In *Privacy Enhancing Technologies*, volume 2482 of *Lecture Notes in Computer Science*, pages 54–68. 2003.

[34] L. Dice. Measures of the amount of ecologic association between species. *Ecology*, 26:pp. 297–302, 1945.

[35] X. Dong and F. Naumann. Data fusion : resolving data conflicts for integration. In *Proceedings of the VLDB Endowment2(2)*, pages 1654–1655, 2009.

[36] H. Dunn. Record linkage. *American Journal of Public Health*, 36(12):pp. 1412–1416. Available at: http://ajph.aphapublications.org/doi/abs/10.2105/AJPH.36.12.1412, Last Accessed: 2015-05-31.

[37] E. Durham. *A Framework for Accurate, Efficient Private Record Linkage*. PhD thesis, Graduate School of Vanderbilt University, Nashville, Tennessee, 2012.

[38] E. Durham, M. Kantarcioglu, Y. Xue, C. Toth, M. Kuzu, and B. Malin. Composite Bloom Filters for Secure Record Linkage. *Knowledge and Data Engineering, IEEE Transactions on*, 26(12):pp. 2956–2968, Dec 2014.

[39] E. Durham, Y. Xue, M. Kantarcioglu, and B. Malin. Quantifying the correctness, computational complexity, and security of privacy-preserving string comparators for record linkage. *Information Fusion*, 13(4):pp. 245–259, 2012. Information Fusion in the Context of Data Privacy.

[40] A. Elmagarmid, P. Ipeirotis, and V. Verykios. Duplicate Record Detection: A Survey. *IEEE Transactions on Knowledge and Data Engineering*, 19(1), January 2007.

[41] K. El Emam. *Guide to the De-Identification of Personal Health Information*. CRC Press., 2013. Available at: http://www.crcpress.com/product/isbn/9781466579064, Last Accessed: 2015-06-06.

[42] K. El Emam and F. Dankar. Protecting Privacy Using k-Anonymity. *Journal of the American Medical Informatics Association : JAMIA*, 15(5):pp. 627–637, 2008. Available at: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC2528029/.

[43] K. El Emam, H. Farah, S. Samet, A. Essex, E. Jonker, M. Kantarcioglu, and C. Earle. A Privacy Preserving Protocol for Tracking Participants in Phase I Clinical Trials. *Journal of Biomedical Informatics*, 2015.

[44] K. El Emam, S. Samet, J. Hu, L. Peyton, C. Earle, G. Jayaraman, T. Wong, M. Kantarcioglu, F. Dankar, and A. Essex. A Protocol for the Secure Linking of Registries for HPV Surveillance. *Public Library of Science*.

[45] B. Fan, D. Anderson, and M. Kaminsky. Cuckoo Filter: Better Than Bloom. *USENIX Programming*, 38(4):pp. 36–40, 2013.

[46] L. Fan, P. Cao, J. Almeida, and A. Broder. Summary Cache: A Scalable Wide-Area Web Cache Sharing Protocol. *IEEE/ACM Transactions on Networking*, 8(3):pp. 281–293, 2000.

[47] I. Fellegi and A. Sunter. A theory for record linkage. *Journal of the American Statistical Society*, 64(328):pp. 1183–1210, 1969.

[48] M. Freedman, K. Nissim, and B. Pinkas. Efficient Private Matching and Set Intersection. In *EUROCRYPT*, 2004.

[49] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In *Proceedings of the 19 th Annual ACM Symposium on Theory of Computing*, pages 218–229, New York, USA, 1987.

[50] M. Hernandez and S. Stolfo. Real-world Data is Dirty: Data Cleansing and The MergePurge Problem. *Data Mining and Knowledge Discovery*, 2:pp. 9–37, 1998.

[51] T. Herzog, F. Scheuren, and W. Winkler. *Data Quality and Record Linkage Techniques*. Springer., 2007.

[52] A. Inan, M. Kantarcioglu, E. Bertino, and M. Scannapieco. A Hybrid Approach to Private Record Linkage. In *Proceedings of the 24th International Conference on Data Engineering (ICDE)*, pages 496–505, Cancun, 2008. IEEE.

[53] A. Inan, M. Kantarcioglu, G. Ghinita, and E. Bertino. Private Record Matching Using Differential Privacy. In *Proceedings of the 13th International Conference on Extending Database Technology*, EDBT '10, pages 123–134, New York, NY, USA, 2010. ACM. Available at: `http://doi.acm.org/10.1145/1739041.1739059`.

[54] Canadian Institute for Health Information (CIHI). Health indicators. Available at: `http://www.cihi.ca/CIHI-ext-portal/internet/EN/TabbedContent/health+system+performance/indicators/health/cihi010654`. Last Accessed: 2015-05-24.

[55] N. Jain, M. Dahlin, and R. Tewari. Using bloom filters to refine web search results. In *Proceedings of 7th WebDB*, pages 25–30, 2005.

[56] M. Jaro. Advances in Record-Linkage Methodology as Applied to Matching the 1985 Census of Tampa, Florida. 84(406):pp. 414–420, 1986. Published online: 12 Mar 2012.

[57] J. Katz and Y. Lindell. *Introduction to Modern Cryptography*. CRC Press., 2015.

[58] A. Kirsch and M. Mitzenmacher. Less Hashing, Same Performance: Building a Better Bloom Filter. *Random Struct. Algorithms*, 33(2):pp. 187–218, September 2008.

[59] L. Kissner and D. Song. Private and threshold set-intersection. Technical report, Carnegie Mellon University, 2005.

[60] R. Klein, S. Proctor, M. Boudreault, and K. Turczyn. Healthy People 2010 criteria for data suppression. *Statistical Notes: From the Centers for Disease Control and Prevention/National Center for Health Statistics*, pages pp. 1–12, July 2002.

[61] M. Kroll and S. Steinmetzer. Automated cryptanalysis of bloom filter encryptions of health records. In *Proceedings of 8th International Conference on Health Informatics*, 2015.

[62] J. Kubiatowicz, D. Bindel, P. Eaton, Y. Chen, D. Geels, R. Gummadi, S. Rhea, W. Weimer, C. Wells, H. Weatherspoon, and B. Zhao. Oceanstore: An Architecture for Global-Scale Persistent Storage. *ACM SIGPLAN Notices*, 25(11):pp. 190–201, 2000.

[63] D. Kushner. Vegas 911. *IEEE Spectrum*, 43(4):pp. 44–49, 2006. doi: 10.1109/MSPEC.2006.1611759.

[64] M. Kuzu, M. Kantarcioglu, E. Durham, and B. Malin. A Constraint Satisfaction Cryptanalysis of Bloom Filters in Private Record Linkage. In *Privacy Enhancing Technologies*. 2011.

[65] M. Kuzu, M. Kantarcioglu, E. Durham, C. Toth, and B. Malin. A practical approach to achieve private medical record linkage in light of public resources. *Journal of the American Medical Informatics Association: JAMIA*, 20(2):pp. 285–292, 2013.

[66] A. Lawati, D. Lee, and P. McDaniel. Blocking-aware Private Record Linkage. In *Proceedings of the 2nd International Workshop on Information Quality in Information Systems*, IQIS '05, pages 59–68, New York, NY, USA, 2005. ACM. Available at: `http://doi.acm.org.proxy1.lib.uwo.ca/10.1145/1077501.1077513`.

[67] Y. Lindell and B. Pinkas. Secure multiparty computation for privacy-preserving data mining. *Journal of Privacy and Confidentiality*, 1(1):pp. 59–98, 2009. Available at: `http://repository.cmu.edu/jpc/vol1/iss1/5`.

[68] A. McCallum, K. Nigam, and L. Ungar. Efficient clustering of high-dimensional datasets with application to reference matching. In *ACM SIGKDD*, pages 169–178, Boston, 2000. ACM.

[69] C. Metz. Basic principles of ROC analysis. *CE Semin Nucl Med*, 8(4):pp. 283–98, 2000.

[70] M. Mitzenmacher and E. Upfal. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press., 2005.

[71] J. Mullin. Optimal semi joins for distributed database systems. *IEEE Transactions on Software Engineering*, 16(5):pp. 558, 1990.

[72] J. Mullin and D. Margoliash. A Tale of Three Spelling Checkers. *Software - Practice and Experience*, 20(6):pp. 625–630, 1990.

[73] G. Navarro. A guided tour to approximate string matching. *ACM Computing Surveys*, 33(1), 2001.

[74] G. Navarro-Arribas and V. Torra. Information fusion in data privacy: A survey. *Information Fusion*, 13(4):pp. 235–244, 2012. Information Fusion in the Context of Data Privacy.

[75] F. Niedermeyer, S. Steinmetzer, M. Kroll, and R. Schnell. Cryptanalysis of basic bloom filters used for privacy preserving record linkage. *German Record Linkage Center, Nurnberg*, 2014.

[76] C. OḰeefe, M. Yung, L. Gu, and R. Baxter. Privacy-preserving data linkage protocols. In *Proceedings of the 2004 ACM Workshop on Privacy in the Electronic Society*, pages 94–102, Washington, DC, October 2004.

[77] Committee on Technical, Privacy Dimensions of Information for Terrorism Prevention, and National Research Council. Other National Goals. Protecting individual privacy in the struggle against terrorists. 2008.

[78] M. Osadchy, B. Pinkas, A. Jarrous, and B. Moskovich. SCiFI - a system for secure face identification. In *IEEE Symposium on Security and Privacy (SP)*, pages 239–254. IEEE, 2010.

[79] C. Pang and D. Hansen. Improved Record Linkage for Encrypted Identifying Data. In *Proceedings of the 14th Annual Health Informatics Conference*, pages 164–168, 2006.

[80] A. Pfitzmann and M. Hansen. Anonymity, Unobservability, and Pseudonymity - A Proposal for Terminology. In *Proceedings of the PET'00, Designing Privacy Enhancing Technologies, International Workshop on Design Issues in Anonymity and Unobservability*, PET'00, pages 1–9. Springer-Verlag, 2000.

[81] A. Pfitzmann and M. Hansen. A terminology for talking about privacy by data minimization: Anonymity, Unlinkability, Undetectability, Unobservability, Pseudonymity, and Identity Management. *Tu Dresden, Faculty of Computer Science*, 2009.

[82] C. Quantin, F. Allaert, B. Cornet, R. Pattisina, G. Leteuff, C. Ferdynus, and J.Gouyon. Decision analysis for the assessment of a record linkage procedure: application to a perinatal network. *Methods Inf Med*, 44:pp. 72–79, 2005.

[83] E. Rahm and H. Do. Data Cleaning: Problems and Current Approaches. *IEEE Bulletin of the Technical Committee on Data Engineering*, 23(4):pp. 3–13, 2000.

[84] H. Rogers and P. Willett. Searching for historical word forms in text databases using spelling-correction methods: reverse error and phonetic coding methods. *J Doc*, 47(4):pp. 333–353, 1991.

[85] L. Roos and A. Wajda. Record linkage strategies. Part I: Estimating information and evaluating approaches. *Methods of information in medicine, PubMed - MEDLINE*, 30(2):pp. 117–123, 1991.

[86] M. Scannapieco, I. Figotin, E. Bertino, and A. Elmagarmid. Privacy Preserving Schema and Data Matching. In *Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data*, SIGMOD '07, pages 653–664, New York, NY, USA, 2007. ACM. Available at: `http://doi.acm.org.proxy1.lib.uwo.ca/10.1145/1247480.1247553`.

[87] B. Schneier. *Applied Cryptography: Protocols, Algorithms, and Source Code in C*. John Wiley & Sons Inc., New York, 2 edition, 1996.

[88] R. Schnell, T. Bachteler, and J. Reiher. Privacy-preserving record linkage using Bloom filters. *BMC Medical Informatics and Decision Making*, 9(1), 2009.

[89] R. Schnell, T. Bachteler, and J. Reiher. A novel error-tolerant anonymous linking code. 2011. Working Paper Series No. WP-GRLC-2011-02.

[90] A. Serjantov and G. Danezis. Towards an Information Theoretic Metric for Anonymity. In *Proceedings of the 2nd International Conference on Privacy Enhancing Technologies*,

PET'02, pages 41–53, Berlin, Heidelberg, 2003. Springer-Verlag. Available at: http://dl.acm.org/citation.cfm?id=1765299.1765303.

[91] E. Spafford. Opus: Preventing Weak Password Choices. *Computer and Security*, 11:pp. 273–278, 1992.

[92] L. Sweeney. k-anonymity: a model for protecting privacy. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, 10(5):pp. 557–570, 2002.

[93] C. Toth, E. Durham, M. Kantarcioglu, and B. Malin. SOEMPI: A secure open enterprise master patient index software toolkit for private record linkage. Available at: http://hiplab.mc.vanderbilt.edu/projects/soempi/

[94] Undisclosed. Record linkage and privacy: issues in creating new federal research and statistical information. Technical report, U.S. General Accounting Office., 2001. GAO-01-126SP.

[95] Undisclosed. Final NIH statement on sharing research data. Technical report, U.S. National Institutes of Health, 2003. NOT-OD-03-032.

[96] D. Vatsalan, P. Christen, and V. Verykios. A Taxonomy of Privacy-preserving Record Linkage Techniques. *Inf. Syst.*, 38(6):pp. 946–969, September 2013.

[97] S. Vaudenay. Security Flaws Induced by CBC Padding Applications to SSL, IPSEC, WTLS. In *EUROCRYPT*, 2002.

[98] S. Weber, H. Lowe, A. Das, and T. Ferris. A simple heuristic for blindfolded record linkage. *Journal of the American Medical Informatics Association: JAMIA*, 19(1):pp. 157–161, 2012.

[99] T. Wetzel. New Sentinel System aims to improve patient safety in real time. Technical report, Unbound MEDLINE Patient Safety.

[100] W. Winkler. Improved decision rules in the fellegiâĂŞsunter model of record linkage. *Journal of the American Statistical Association*, 274:279, 1993.

[101] M. Yakout, M. Atallah, and A. Elmagarmid. Efficient Private Record Linkage. In *Proceedings of the 24th International Conference on Data Engineering (ICDE)*, pages 496–505, Cancun, 2009. IEEE.

[102] A. Yao. How to generate and exchange secrets. In *Proceedings of the 27th Annual Symposium on Foundations of Computer Science*, pages 162–167. IEEE, 1986.

# Appendix A

# Sample Dataset

## A.1 Sample Subset of the Dice Dictionary(DD)

| Dice$(b_1, b_2)$ | $s_1, s_2$ | $\|b_1\|, \|b_2\|, \|b_1 \cap b_2\|$ |
|---|---|---|
| 0.0689655172413793 | VERONICA DOMINCZYK, TODD COOK | 19, 10, 1 |
| 0.162162162162162 | VERONICA DOMINCZYK, MICHAEL BLOOMFIELD | 19, 18, 3 |
| 0.157894736842105 | VERONICA DOMINCZYK, KATHERINE CALDWELL | 19, 19, 3 |
| 0.0689655172413793 | VERONICA DOMINCZYK, LISA TATE | 19, 10, 1 |
| 0.0666666666666667 | VERONICA DOMINCZYK, LORI KIJEK | 19, 11, 1 |
| 0.0666666666666667 | VERONICA DOMINCZYK, LOGAN YORK | 19, 11, 1 |
| 0.238095238095238 | VERONICA DOMINCZYK, JACQUELINE FRONCKOWIAK | 19, 23, 5 |
| 0.0689655172413793 | VERONICA DOMINCZYK, MIMI SHELL | 19, 10, 1 |
| 0.142857142857143 | VERONICA DOMINCZYK, NICOLE LEE | 19, 9, 2 |
| 0.105263157894737 | VERONICA DOMINCZYK, JEFFERY COTTINGHAM | 19, 19, 2 |
| 0.0689655172413793 | VERONICA DOMINCZYK, LUZ JONES | 19, 10, 1 |
| 0.0666666666666667 | VERONICA DOMINCZYK, CARL GOODE | 19, 11, 1 |
| 0.0512820512820513 | VERONICA DOMINCZYK, CHRISTOPHER HAYMORE | 19, 20, 1 |
| 0.0540540540540541 | VERONICA DOMINCZYK, CHRISTOPHER GRILLI | 19, 18, 1 |
| 0.0666666666666667 | VERONICA DOMINCZYK, LEE HOLDER | 19, 11, 1 |

**Table A.1:** Sample subset of the Dice dictionary(DD)

The steps involved in generating the above DD entries is explained in Section A.1.1 aided

with an example.

### A.1.1 Steps for Creating Dice Dictionary (DD) Entries

Considering two sample plaintext strings, VERONICA DOMINCZYK and TODD COOK
$s_1$ = VERONICA DOMINCZYK
$s_2$ = TODD COOK
Bigram set $b_1$ for $s_1$: `_V VE ER RO ON NI IC CA A_ _D DO OM MI IN NC CZ ZY YK K_`
Bigram set $b_2$ for $s_2$: `_T TO OD DD D_ _C CO OO OK K_`
set cardinality $|b_1|$ for $b_1$: 19
set cardinality $|b_2|$ for $b_2$: 10
set cardinality for $|b_1 \cap b_2|$: 1 (*i.e.,* because $s_1$ and $s_2$ share only one common bigram, K_)
Now, applying the Dice coefficient expression as stated in equation (2.1), we get:

$$Dice(b_1, b_2) = \frac{2|b_1 \cap b_2|}{|b_1| + |b_2|} = \frac{2.1}{19 + 10} = 0.0689655172413793$$

Dice Dictionary row is populated in format: $\left\{Dice(b_1, b_2) : s_1, s_2 : |b_1|, |b_2|, |b_1 \cap b_2|\right\}$ which in this case is obtained as:
0.0689655172413793 : VERONICA DOMINCZYK, TODD COOK : 19, 10, 1
This formatted information is stored in row 1 of sample DD as shown in Table A.1.

# Curriculum Vitae

| | |
|---|---|
| **Name:** | Vasundhara Sharma |
| **Post-Secondary Education and Degrees** | Master of Engineering Science (Candidate),<br>The University of Western Ontario<br>London, Ontario, Canada<br>2015 (Expected) |
| | Bachelor of Technology (Computer Science and Engineering)<br>Manipal Institute of Technology<br>Manipal, Karnataka, India<br>2007 |
| **Related Work Experience:** | Teaching Assistant<br>The University of Western Ontario<br>2013–15 |
| | Senior Software Engineer II<br>Cisco Systems Inc.<br>Bangalore, India<br>2011–13 |
| | Software Engineer II<br>Citrix R&D India Pvt. Ltd.<br>Bangalore, India<br>2007–11 |

**Publications:**

- Vasundhara Sharma and Aleksander Essex. Heuristic Security and the Trouble With Bloom Filter Encodings for Private Record Linkage. *Privacy Enhancing Technologies Symposium (PETS)*. In submission.