All-or-Nothing Private Record Linkage over Streaming Data

by

John Abraham Premkumar

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2022

© John Abraham Premkumar 2022

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

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

Abstract

The prevalence and increasing need for insights obtained from the collection of sensitive data gives rise to the problem of protecting the privacy of this data. The collection and storage of data can be distributed across locations and organizations, and gaining insights can require combining knowledge from different stores. Private record linkage (PRL) is the problem of finding approximately matching records across different databases while maintaining the privacy of all records involved. The PRL problem in the streaming data model is an emerging problem that tackles PRL in the context of a streaming database, where a service provider performs the matching and learns only the result to gain further insights. To the best of our knowledge our work is the first to address this problem.

In this work, we introduce a new cryptographic primitive, the secure approximate equality operator that securely implements all-or-nothing disclosure for approximate matching, which has provable security guarantees in the semi-honest security model. We show that the new operator performs several times faster than a straightforward baseline approach using function-hiding inner product encryption. We also showcase a protocol that implements our new approximate equality operator to perform PRL in the streaming data model with high accuracy and performance.

Acknowledgements

I would like to thank my supervisor, Florian Kerschbaum, for his guidance during the course of this work. I thank him for his incredible patience, encouragement and insights without which this work would not have been possible. I thank my thesis readers Xi He and Mohammad Hajiabadi for their valuable suggestions that helped make this thesis what it is. I would also like to thank my friends and colleagues Nidhi, Thomas, Rasoul and Jason for lending their ears and also for their words of encouragement through the course of my time here.

Table of Contents

List of Figures vii

List of Tables viii

1 Introduction 1

2 Problem statement and security model 3

3 Related work 5

4 Background 8

4.1 Secret sharing 8

4.2 The Berlekamp-Welch algorithm 9

5 Preparing and comparing data 10

5.1 Encoding using BERT 10

5.2 Choosing a distance metric 11

6 Secure approximate equality 13

7 Protocols for PRL in the streaming data model 16

7.1 Common protocol description 16

7.2 Baseline protocol 17

7.3 New protocol 18

8 Security analysis **20**

9 Datasets used **23**

9.1 DBLP-ACM 23

9.2 Amazon-Google 24

9.3 Freely Extensible Biomedical Record Linkage (FEBRL) 24

10 Experimental results **25**

10.0.1 Setup 25

10.0.2 Accuracy 25

10.0.3 Performance 36

11 Conclusion **38**

References **39**

List of Figures

10.1 Metrics on the Amazon-Google dataset, vector length 128. Matched using titles (red). Matched using product description (green). 27

10.2 Metrics on the Amazon-Google dataset, vector length 256. Matched using titles (red). Matched using product description (green). 28

10.3 Metrics on the Amazon-Google dataset, vector length 384. Matched using titles (red). Matched using product description (green). 29

10.4 Metrics on the DBLP-ACM dataset, vector length 128. Matched using titles (red). Matched using authorlist (green). 30

10.5 Metrics on the DBLP-ACM dataset, vector length 256. Matched using titles (red). Matched using authorlist (green). 31

10.6 Metrics on the DBLP-ACM dataset, vector length 384. Matched using titles (red). Matched using authorlist (green). 32

10.7 Metrics on the FEBRL dataset, vector length 128. Matched using full records (red). Matched using full records from one database, and records with a chosen small subset of attributes in the second database (green). . . 33

10.8 Metrics on the FEBRL dataset, vector length 256. Matched using full records (red). Matched using full records from one database, and records with a chosen small subset of attributes in the second database (green). . . 34

10.9 Metrics on the FEBRL dataset, vector length 384. Matched using full records (red). Matched using full records from one database, and records with a chosen small subset of attributes in the second database (green). . . 35

List of Tables

10.1 Compute time comparison, all times are in seconds 37

Chapter 1

Introduction

The widespread collection of data in many forms is an ever increasing trend over the last few decades. The accumulation of personal data, and other forms of sensitive data using a myriad of means has also given rise to a multitude of services that depend on gaining insights from this data. The sensitive nature of this data introduces the problem of protecting its privacy. The data however, can be collected by different entities or corporations, and may be generated and stored in different locations, and thus combining knowledge from different sources intensifies the problem of maintaining the privacy of sensitive data.

One of the key operations performed on separate structured datasets is what is known as "record linkage" [1] [2] [3] [4]. Record linkage can be described as the problem of determining records in different datasets that refer to a common entity, this process being resilient to errors and differences in record structure. Due to the aforementioned sensitive nature of emerging datasets, a more privacy-conscious version of record linkage, i.e., private record linkage [5] [6] [7] [8] [9] [10] [11] becomes increasingly relevant.

Consider the following scenario: officials at an airport need to verify whether passengers about to board a plane are on a no-fly list. The list is protected, and passenger data is personal data. One way to determine if the passenger is allowed to fly is to securely send, i.e., stream the passenger data securely to a service provider. The service provider also securely receives and stores the no-fly list from the appropriate government authority. The service provider then determines if the passenger is on the list and releases the result back to the airport. All this must be done in such a way that the service provider does not learn either the passenger data or the no-fly list. This example can be generalized as follows, a client, or a set of clients want to check if their database records match records possessed by

a data owner. Computing the matches is outsourced to a service provider, who receives the records as a stream from the client, and possesses a set of records as queries from the data owner. The service provider returns the result: match or non-match. The service provider also learns the result, and can use it and the meta-data for further analysis. The service provider does not learn anything about the streamed records, or the queries of the data owner. What we have described is private record linkage in the streaming data model. When compared to private record linkage in the more common static database model, private record linkage in the streaming data model poses additional challenges. The first challenge is the objective to minimize the rounds of communication between the parties. The second challenge is that the streamed records must be processed one at a time, which eliminates solutions that operate on sets of records.

All-or-nothing disclosure is defined as a disclosure by which, as soon as a party obtains information on another party's secret/data, they can gain no additional information beyond that [12]. Our protocol implements the notion of all-or-nothing disclosure to achieve benefits in privacy, that is, the service provider is only disclosed whether there has been a match or not. This all-or-nothing disclosure guarantees that an adversary cannot learn any more information than what has been output. We introduce a new cryptographic primitive, the secure approximate equality operator that securely implements all-or-nothing disclosure for approximate matching. The usage of all-or-nothing disclosure allows us to escape the pitfalls [13] [14] [15] [16] of perturbation based private record linkage techniques such as those that use Bloom filters [7], which do not offer strong security guarantees.
Our contributions in this paper are as follows:

- We introduce the notion of a secure approximate equality operator that implements all-or-nothing disclosure, which bounds the leakage of information.
- We design a novel secure approximate equality operator, which in evaluations performs approximately 6 to 31 times faster than a straightforward baseline approach based on function-hiding inner product encryption.
- We evaluate a protocol that implements our new secure approximate equality operator to perform private record linkage in the streaming data model, and show that it has high accuracy and performance.

Chapter 2

Problem statement and security model

The problem of private record linkage in the streaming data model can be formalized as follows: Consider three parties, the client, the data owner and the third party service provider, henceforth referred to as P_1, P_2 and P_3 respectively. Let A be a record possessed by P_1, and B be a record possessed by P_2. B is sent to P_3 securely to be stored as a query, while A is streamed securely to P_3. P_3 computes a function on the secure record inputs, which returns whether or not A and B are approximate matches, and returns the output to the other two parties. P_3 also learns the binary match result, and a bounded amount of additional information.

The match function is implemented as a secure approximate equality operator. Let d(x,y) be a distance metric to compare two equal length integer valued vectors x and y. A secure approximate equality operator is an algorithm that takes x and y as inputs, and returns "match", if d(x,y) between the two vectors is within a threshold, or "non-match", otherwise.

P_1 and P_2 are required to transform the records A and B into a form that can be compared using a secure approximate equality operator. This is done through the process of encoding. We define the function Encode(), that takes a record as input and transforms it into a fixed length vector of numerical values. The parties P_1 and P_2 compute the Encode() function on A and B respectively, and generate the numerical vectors x and y.

The secure approximate equality operator can be broken down into the following three functions: KeyGen(), Encrypt() and Compare(). KeyGen() generates a symmetric key used by the system, the function Encrypt() takes in a fixed length numerical vector and

returns a ciphertext, and the function Compare() takes in two ciphertexts and returns "match" or "non-match" based on the distance between the vectors that were encrypted. The parties compute the secure approximate equality as follows: P_1 and P_2 run their respective Encrypt() functions on x and y, to obtain the encrypted vectors (ciphertexts) X and Y, which are then sent to P_3. Note that P_1 and P_2 do not collude and neither of them has access to the other's ciphertexts. P_3 computes Compare() on X and Y, and returns the result to P_1 and P_2. The Compare() function of our more efficient secure approximate equality operator leaks a small additional bounded amount of information beyond this all-or-nothing disclosure in the case of a match.

The secure approximate equality operator is secure in the semi-honest security model. In the semi-honest model, a party controlled by an adversary tries to gain as much additional information about the inputs of the other parties as possible, but does not deviate from the protocol itself, and executes all steps of the protocol as intended. A multi-party protocol can be proven secure in the semi-honest security model by the simulation argument, that is, showing that each party's view of the execution of the protocol can be simulated given only its inputs and outputs [17], and that this simulated view is computationally indistinguishable from a real protocol. We will formalize this security definition in Section 6.

The semi-honest security model has a stronger counterpart in the malicious security model, in which a party controlled by an adversary can arbitrarily deviate from the protocol in its quest to obtain additional information about the inputs of the other parties. Although the malicious security model is stronger, it can require protocols to be inefficient or untenable in terms of performance. Furthermore, legal and economic checks put in place can ensure that security in the semi-honest model is a sufficiently strong guarantee [18] [19].

Chapter 3

Related work

The problem of Private Record Linkage has been studied in several previous works, each of these possessing different strengths and weaknesses. A taxonomy of privacy-preserving record linkage (PPRL) techniques by Vatsalan et al. [5], details and compares large amount of prior work in the area of record linkage. Exact matching refers to two records that match perfectly. In this work, we consider only those works that perform approximate matching, where two records can be considered linked if they are considered similar enough, through various metrics. A more recent work by Gkoulalas-Divanis et al. [6] provides an overview of more recent works in the field.

He et al. [9] argue that an ideal PRL system must satisfy three desiderata: Perfect precision and high recall of correct matches, provable privacy, and subquadratic computation cost with respect to the number of records. PRL is generally conducted by matching on obfuscated data, and these obfuscation methods can be categorized broadly into secure multi-party computation (SMC) techniques and perturbation based techniques. Christen et al. [13] note that PRL that use SMC to guarantee cryptographic security have historically had high accuracy and have high computation cost, or suffer from low accuracy [9]. Perturbation based PRL approaches, such as those by Schnell et al. [7] or Durham et al. [8] try to overcome these computation cost limitations by using Bloom filters to speed up the computation. However, they do not offer a formal guarantee of security, and indeed attacks are formulated that target Bloom filters, as shown by Kroll and Steinmetzer [16], Kuzu et al. [20], and Christen et al. [13] [14] [15]. In the pursuit of subquadratic runtimes, works by Rao et al. [10] and Xi at al. [9] use differential privacy. More recently, Khurram and Kerschbaum [11] introduced a two party PRL protocol that provides cryptographic guarantees as well runs in subquadratic time.

There are works that explore the (non-Private) Record Linkage problem in the online setting [21] [22], which attempt to solve the problem when it moves beyond matching in localized files or from multiple sources in batch processing mode [21], and consider a distributed, online setting. Karapiperis et al [23] develop a distributed engine for record linkage that they claim can be used for PRL through the use of a trusted third party.

In the streaming model, the third desiderata as discussed by He et al. [9] is not as relevant since the records are streamed one at a time. Karapiperis et al. [24] and Ramadan et al [25] propose techniques that can be used for non-private RL in the streaming data model. However, there are no works that we know of which attempt to solve PRL specifically in the streaming data model. Kim et al. [26] introduce a function hiding inner product encryption scheme that can be used for performing nearest neighbor searches on encrypted databases, by computing l_2 distance between encrypted records. This methodology can be adapted into a system to compute private record linkage in a streaming model.

We now discuss work related to our method to encode records into fixed-length vectors. Franke et al. [27] argue that records pertaining to personal data tends to be dense, and that it is possible to have high similarity with non-matches in existing methods. Another problem is that records also contain errors. They further argue that simple threshold-based matching methods fail to deal with the aforementioned problems, and propose the use of post-processing methods to resolve incorrect links and/or multi-links where a person is matched to more than one other person incorrectly. However, by using modern machine learning models we are able to show that these problems can be overcome.

There are several works that support the use of machine learning models to embed data/records as a data pre-processing step in record linkage. Structured records are represented in earlier works using one hot encodings and term frequency-inverse document frequency (TF-IDF) vectors [28]. Y.L. Sim et al. [28] create an encoder framework that generate embeddings of structured data, based on the hypothesis that attributes within each record instance are likely to be related to each other in vector space. This enables training on unlabelled data and the authors verify that this methodology works on a wide variety of attributes including categorical data, descriptions, tokens and numerical data.

Foxcroft et al. [29] show that using name embeddings to perform matching tasks can be effective at detecting matches in spite of errors. The use a modified Doc2Vec methodology to transform names to vector space and use cosine distance as a distance metric to ascertain similarity between names.

Behnen et al. [30] note that other works, while they use varied embedding architectures including state-of-the-art transformers such as Google BERT, do not explore record linkage

systems that use only descriptions for matching. They show that unsupervised matching using Google BERT can have promising results using a record linkage task that uses company descriptions to perform matching. We can conclude from these works that using state-of-the-art transformers such as Google BERT to embed records for the purpose of record linkage can yield good results for a wide variety of datasets.

Chapter 4

Background

4.1 Secret sharing

Secret sharing refers to various methods of distributing secret information among a number of parties. Each party obtains a "share" of the secret, and the secret can be revealed if a pre-determined number of parties (at the minimum) join their shares. Consider the following secret sharing scheme: A secret owner O shares a secret s among p parties such that when any number of parties equal to or above a threshold k join their shares, they can reconstruct the secret. Such a scheme is known as a (k,p)-threshold scheme. [31]

Shamir's Secret Sharing: Shamir's secret sharing scheme [31] is a (k,p)-threshold scheme that is based on the Lagrange interpolation theorem, from which it can be inferred that t points are sufficient to uniquely determine a polynomial whose degree is less than or equal to k-1. The scheme is constructed as follows:

Let the secret be represented as a finite field element of the field F_q, z_0. Choose k-1 elements from this field $z_1, z_2, z_3...z_{k-1}$ and construct a polynomial $f(x)$

$$f(x) = z_0 + z_1x + z_2x^2 + z_3x^3...z_{k-1}x^{k-1}$$

This polynomial is now evaluated at p non-zero integer points, resulting in p integer shares. These shares are then distributed to p parties. If any subset of at-least k of these parties join their shares, Lagrange polynomial interpolation can then be used to reconstruct the original polynomial, and the secret is obtained by computing $f(0)$.

4.2 The Berlekamp-Welch algorithm

The Berlekamp-Welch algorithm is a decoder algorithm that is used to correct errors in Reed-Solomon codes. We briefly describe the construction of Reed-Solomon codes as follows: Let a message m_1, ...,m_k, known as the original message, be used as the coefficients of a polynomial $f(a_i)$ of degree less than k, over the finite field F_q of size q. Let this polynomial then be evaluated at $l \leq q$ distinct points a_1 to a_l of the field F_q to obtain the sent message s_1,...,s_l. The decoder attempts to recover the polynomial $f()$ and thus, the original message given a received message of the form r_1,...,r_l with a maximum of t possible errors, and the inputs a_1,...,a_l.

If decoding is successful, the decoder returns an error polynomial $e(a_i)$ where $e(a_i)$=0 indicates an error at point i in the received message. If decoding is not successful, the algorithm fails and does not return anything, indicating that the errors cannot be corrected. The maximum number of errors that can be corrected by the Berlekamp-Welch algorithm is $\lfloor \frac{(l-k)}{2} \rfloor$.

We use the Berlekamp-Welch algorithm to efficiently detect matches in our protocol. We adjust the threshold t such that, if an encoding of one record can be corrected to the encoding of another record, they are considered an approximate match. If the algorithm fails to perform such a correction, the two records are considered non-matches. The polynomial $f(a_i)$ is a Shamir's polynomial for secret sharing the value 0 in our protocol, and thus the sent message s_1,...,s_l would be secret shares of zero. The public inputs a_1,...,a_l. are 1,...,l.

Chapter 5

Preparing and comparing data

5.1 Encoding using BERT

We implement the Encode() function defined in the problem statement as a two-step process. We first embed records into vectors of real-values using a language representation model: Google BERT (Bidirectional Encoder Representations from Transformers). Google BERT was deemed suitable, as the evaluations performed in this work use databases that possess records containing a broad variety of alphanumeric data, including descriptions and addresses, and a general purpose language model was thus chosen. In addition, pre-trained BERT models are designed to be fine tune-able with one additional output layer, and requires no significant changes in their architecture for a specific task. [32]

Pre-trained sentence embedding BERT models embed inputs to a real-valued vector, with values in the range [0,1]. The dimension of the vector depends on the model used, but for our purposes we choose a model that (when unmodified) returns embeddings of length 384.

Once this real-valued vector is obtained, we move to the next step where we convert these to integer valued vectors of the same length using an interval based converter function which transforms real values from a fixed domain to a corresponding integer value. We demonstrate the above steps by using a toy example. Let a record R be embedded by Google BERT into a real-valued vector V of length 5.

$$V = [0.23, 0.45, 0.11, 0.87, 0.51]$$

Let n be the number of intervals into which we divide [0,1]. In our example we set $n = 4$. The conversion function converts the real-valued vector V into an integer-valued vector v of the same length. Let V[i] and v[i] represent the i-th value in the vectors V and v respectively. Let j be an integer-valued variable.

$$v[i] = \max(1, \lceil n \cdot V[i] \rceil) \quad \forall i$$

Thus in our example, [0,0.25] is mapped to the integer 1, (0.25,0.50] is mapped to 2, (0.50, 0.75] is mapped to 3, and (0.75,1] is mapped to 4. The conversion function returns the vector v.

$$v = [1, 2, 1, 4, 3]$$

Thus we obtain v = Encode(R).

5.2 Choosing a distance metric

In the problem statement, we refer to a general notion of a distance metric $d(x, y)$. In this work, we use Google BERT to encode alphanumeric database records that we need to compare into an integer vector space. We thus choose a distance metric that is suited to this encoding, such that the distance between encoded records is a good proxy for how similar or dissimilar two records are. That is, the closer the distance between the encoded records are, the more similar the two original records are, for the purpose of record linkage.

We define a metric, "Element inequality" between two vectors of integer values as the total number of different values in corresponding vector elements between the two vectors. Let l be the length of the fixed length vectors x and y. The function $d(x, y)$ for element inequality can be defined as follows

$$d(x, y) = \sum_{i=1}^{l} \delta_i$$

where

$$\delta_i = \begin{cases} 0 & \text{if } x[i] = y[i] \\ 1 & \text{otherwise} \end{cases}$$

Element inequality was chosen to be our primary distance metric as it has good performance in accuracy metrics (see Section 10), and is well suited to use in a secure approximate equality operator. Henceforth, when we refer to $d(x, y)$ as a distance function we imply the use of the element inequality distance function.

Chapter 6

Secure approximate equality

Let x and y be two same-length vectors with numerical data, derived from records A and B. An approximate equality operator attempts to compute the match function as follows: It computes a distance metric $d(x, y)$ and returns the following

$$result = \begin{cases} match & \text{if } d(x,y) \leq t \\ non-match & \text{otherwise} \end{cases}$$

where t is a preset threshold.

Secure approximate equality minimises the information revealed about the two vectors to the party computing the operator.

Formalization: Let λ be the security parameter that bounds a computational adversary. Let x be the encoding of a record. A secure approximate equality operator (in the context of record linkage) consists of the following functions which are possibly probabilistic and run in polynomial time:

1. K $\leftarrow KeyGen(1^\lambda)$: generates a (symmetric) key K using the security parameter λ.
2. $c \leftarrow Encrypt(K, x, t)$: Encrypts a vector x into a ciphertext c, for a threshold t.
3. match/non-match $\leftarrow Compare(c_1, c_2, t)$: returns "match" or "non-match" given the ciphertexts c_1 and c_2 based on the distance between x_1 and x_2, and t. x_1 and x_2 are the vectors that were encrypted into c_1 and c_2.

Correctness: Let A and B be two records, which are encoded into vectors x and y. We say a secure approximate equality operator is correct, if

$$
\begin{aligned}
&\forall x, y, t \\
&K \leftarrow KeyGen(1^{\lambda}) \\
&c_1 \leftarrow Encrypt(K, x, t) \\
&c_2 \leftarrow Encrypt(K, y, t) \\
&d(x, y) \leq t \implies Compare(c_1, c_2, t) = equal \\
&d(x, y) > t \implies Pr[Compare(c_1, c_2, t) = equal] = negl(\lambda)
\end{aligned}
$$

Security definition Let a represent inputs to the protocol, and λ be the security parameter that bounds a compuational adversary. Let E_1 and E_2 be distribution ensembles for random variables $E_1(\lambda, a)$ and $E_2(\lambda, a)$. We define computational indistinguishability of E_1 and E_2 by $E_1 \stackrel{c}{\equiv} E_2$, if for every non-uniform distinguisher D that runs in polynomial time, there exists a negligible function $negl(\lambda)$ such that

$$|Pr[D(E_1(\lambda, a)) = 1] - Pr[D(E_2(\lambda, a)) = 1]| = negl(\lambda)$$

Let A and B be two records, which are encoded into vectors x and y. Let $\mathcal{L}(x, y)$ be the information that is leaked about x and y (and thus by consequence A and B) when executing the secure approximate equality operator. We say that an approximate equality operator is $\mathcal{L}$-secure, if there exists a Simulator Sim($\mathcal{L}(x, y)$) such that

$$
\begin{aligned}
&\forall x, y, t \\
&K \leftarrow KeyGen(1^{\lambda}) \\
&c_1 \leftarrow Encrypt(K, x, t) \\
&c_2 \leftarrow Encrypt(K, y, t) \\
&Sim(\mathcal{L}(x, y)) \stackrel{c}{\equiv} Compare(c_1, c_2, t)
\end{aligned}
$$

Let $H_{K_i}(.)$ be a keyed cryptographic hash function. We say that an approximate equality operator is relaxed secure if it satisfies $\mathcal{L}^*$ security for

$$\mathcal{L}^*(x,y) = \begin{cases} match, H_{K_i}(x[i]) - H_{K_i}(y[i])\ \forall x[i] \neq y[i] & \text{if d(x,y)} \leq t \\ non - match & \text{otherwise} \end{cases}$$

Chapter 7

Protocols for PRL in the streaming data model

In this section, we first describe a common abstract protocol that performs private record linkage in the streaming data model. We then describe two instances of this protocol. One is the baseline solution that utilizes function-hiding inner product encryption, and the other is our new protocol that utilizes our new secure approximate equality operator. Note that the offline phase of the baseline solution is very computationally intensive [26].

7.1 Common protocol description

- **Key generation**: $K \leftarrow \text{KeyGen}(1^\lambda)$ is computed at P_2. P_2 shares K with P_1. P_3 does not have access to this key.
- **Encode records**: P_1 encodes record A into vector x, and P_2 encodes record B into vector y using the Encode() function.
- **Encryption**:

  P_1 runs $c_1 \leftarrow Encrypt(K, x, t)$ and P_2 runs $c_2 \leftarrow Encrypt(K, y, t)$
- **Matching**: P_1 and P_2 transmit the results of their respective $Encrypt()$ functions to P_3. P_3 computes $Compare(c_1, c_2, t)$ and returns the response "match" or "non-match" to both party P_1 and P_2.

7.2 Baseline protocol

We first implement a baseline solution based on Function-Hiding Inner Product Encryption, abbreviated as FHIPE, which is a class of functional encryption schemes that compute the inner product of encrypted vectors. For our baseline protocol, we use the FHIPE scheme introduced by Kim et al. [26]. We adapt a modification to this FHIPE scheme proposed by Shafieinejad et al. [33] by converting the function-hiding inner product encryption scheme into a function-hiding inner product predicate encryption scheme by checking if the inner product is zero. The scheme consists of four algorithms IPE.Setup, IPE.KeyGen, IPE.Encrypt and IPE.Decrypt. A brief high level overview of these are as follows:

- pp, msk $\leftarrow$ IPE.Setup(1^λ): Generates the public parameters pp, and a master secret key msk using the security parameter λ.
- sk $\leftarrow$ IPE.KeyGen(msk, x): Using the master secret key, encrypts a vector x into a functional secret key sk.
- c $\leftarrow$ IPE.Encrypt(msk, x): Using the master secret key, encrypts a vector x into a ciphertext c.
- 1/0 $\leftarrow$ IPE.Decrypt(pp,sk,c): Using the public parameters and the inputs sk and c, checks if the inner product of the two encrypted vectors is zero. Returns 1 if the inner product is 0, returns 0 otherwise.

We implement an approximate equality operator using FHIPE as follows:

1. msk $\leftarrow$ $KeyGen(1^\lambda)$: Calls IPE.Setup(1^λ) which generates pp and msk.
2. $Encrypt(msk, x, t)$: Encrypts a vector x at party P_i.

   If P_i is P_1, it generates the vector $\tilde{x}$ as follows: The length of $\tilde{x}$ is set to n $\cdot$ l and is populated with zeros. The integer values in x are transferred to $\tilde{x}$ as one-hot encodings.

   $$\tilde{x}[n \cdot j + x[j]] = 1,\ 0 \leq j < l$$

   The value 1 is then appended to $\tilde{x}$. It then outputs the functional secret key sk using IPE.KeyGen($msk, \tilde{x}$), where $0 \leq i \leq t$.

   If P is P_2, it generates t vectors $\tilde{x_i}$ as follows: The length of each $\tilde{x_i}$ is set to n $\cdot$

l and is populated with zeros. The integer values in x are transferred to $\tilde{x_i}$ as one-hot encodings.

$$\tilde{x_i}[n \cdot j + x[j]] = 1,\ 0 \leq j < l$$

The value l-i is then appended to x_i. It then generates t ciphertexts c_i as set $\mathbf{C}$ using IPE.Encrypt$(msk, \tilde{x_i})$, where $0 \leq i \leq t$. Set $\mathbf{C}$ is then permuted to hide the order, thus hiding the value of $d(x, y)$.

3. $Compare(sk, \mathbf{C}, t)$: Given sk and set $\mathbf{C}$, attempts to compute the existence of $d(x, y)$ $\leq$ t, and returns $response(x, y)$, where

$$response(x, y) = \begin{cases} "match" & \text{if } \exists \text{ IPE.Decrypt}(pp, sk_i, c_i) = 1\ \forall \text{ i} \\ "non - match" & \text{otherwise} \end{cases}$$

7.3 New protocol

Let $H_K(x)$ be a keyed, cryptographic hash function. Our protocol implements the approximate secure equality operator as follows

1. $KeyGen(1^\lambda)$: generates a key K for the keyed cryptographic hash function $H_K()$ using the appropriate key derivation function with the security parameter λ.

2. $Encrypt(K, x, t)$: Encrypts a vector x at party P_i. Let $f()$ be a polynomial function for secret sharing 0 under Shamir's scheme. Let K_i be $H_K(i)$.

   If P_i is P_1, generates vector X.

   $$X_i = f(i) + H_{K_i}(x[i])$$

   If P is P_2 Vector X is generated as follows

   $$X_i = f(i) - H_{K_i}(x[i])$$

3. $Compare(X, Y, t)$: Let X be a ciphertext generated by P_1 and Y be a ciphertext generated by P_2. computes

   $$Z = X + Y$$

   P_3 attempts to compute $d(x, y)$ as follows:

Attempt to reconstruct the polynomial $g()$ from the values of Z using the Berlekamp-Welch algorithm. If reconstruction is successful, then the element inequality, $d(x, y)$, is equal to number of errors detected in the reconstruction, and if reconstruction fails, this distance cannot be determined. $\mathcal{L}^*(x, y)$ is then returned, where

$$\mathcal{L}^*(x,y) = \begin{cases} match, H_{K_i}(x[i]) - H_{K_i}(y[i])\ \forall x[i] \neq y[i] & \text{if d(x,y)} \leq t \\ non - match & \text{otherwise} \end{cases}$$

Chapter 8

Security analysis

In this section we analyze the security of our new secure approximate equality operator. The security of the baseline solution follows straightforwardly from the security of inner product encryption. Hence, we start by defining the security of our operator.

Let $View_i^{\pi}$ represent the view of party P_i under the execution of the real protocol π. x and y are encoded vectors of length l.

Consider the party P_3. There exists a probabilistic polynomial-time algorithm Sim such that for every pair of real inputs x, y and functionality $\mathcal{L}^*$ (as defined in Section 6), and corresponding ideal inputs x and y, we have

$$Sim(\mathcal{L}^*(x,y)) \stackrel{c}{\equiv} View_3^{\pi}(x,y), output^{\pi}(x,y)$$

Proof Consider a probabilistic polynomial time adversary Adv controlling party P_3. Assume that Adv attacks the protocol by distinguishing between a view simulated by Sim, and a view under a real execution.

Case 1: The case where X and Y are considered a match under the threshold t, i.e, the bit vector representations of X and Y have at-least l-t values in common. For those indices where the result returned by $\mathcal{L}^*(x,y)$ indicates a match, the simulator Sim generates a simulated view as follows:

$$X_i = \tilde{f}(i) + Rand(i) \tag{8.1}$$

$$Y_i = \tilde{g}(i) - Rand(i)$$

$\tilde{f}$(x) and $\tilde{g}$(x) are polynomials for secret sharing 0, of the same degree as f(x) and g(x) used in the real functionality. Rand(x) returns an independent, uniformly chosen random number. Now compare this with the view in the real world, where

$$X_i = f(i) + H(x[i])^k \tag{8.2}$$

It can be shown that if we choose to replace the output of the keyed hash exchange subprotocol with a random number, this substitution cannot be distinguished from the original except with negligibly small probability, because the keyed hash function is pseudo-random and hence its output indistinguishable from random numbers. Furthermore, since $f(i)$ is a secret share of 0, it can also be replaced with another secret share $j(i)$, as long as the degree of the polynomial remains the same, as these values are secret shares under Shamir's secret sharing scheme. Thus, the values from Equation 8.2 cannot be distinguished from Equation 8.1 by Adv. A similar argument is used for the Y_i values.

For those indices where the response from $\mathcal{L}^*(x, y)$ indicates an error, the simulator S generates a simulated view

$$X_i = \tilde{f}(i) + Rand(i) + H(x[i])^k - H(y[i])^k \tag{8.3}$$

$$Y_i = \tilde{g}(i) - Rand(i)$$

The argument here is similar to the previous one, but in this case, we choose to replace the output of the keyed hash with a random number and difference of two keyed hashes which is the value leaked to the adversary. Distinguishing between this and the random number would break the indistinguishability of the keyed hashes from random numbers, and hence, Equation 8.3 cannot be distinguished from 8.1 by A.

Once the X_i and Y_i values are summed up and reconstructed using the Berlekamp–Welch algorithm, the adversary obtains a polynomial for secret sharing 0, and the differences $H(x_i)^k$-$H(y_i)^k$. Thus, no additional information is leaked from what is revealed in the real protocol.

Case 2: The case where X and Y are considered not a match, i.e., the bit vector representations of X and Y have less than l-t values in common. The simulator S generates the view as below, and returns l random values as the output of the functionality, where $1 \leq i \leq l$.

$$X_i = \tilde{f}(i) + Rand1() \tag{8.4}$$

$$Y_i = \tilde{g}(i) - Rand2()$$

Rand1() and Rand2() return independent, uniformly chosen random numbers, and it is ensured that Rand1() and Rand2() are not equal. With a simular argument, we can see that these are indistinguishable from the real views. If the adversary attempts to reconstruct using the Berlekamp–Welch algorithm, the algorithm fails, and thus the adversary learns no information.

Other parties: Corresponding theorems for P_1 and P_2 can be constructed, but these theorems are obvious since the views of these parties contain only the results obtained from P_3, and can be simulated as is.

Chapter 9

Datasets used

We use the following datasets in the accuracy tests as part of our evaluation. These datasets were chosen as they represent a broad variety of record linkage tasks. The DBLP-ACM dataset simulates linkage tasks with records that have short alphanumeric descriptions. The Amazon-Google dataset possesses records with longer descriptions. The FEBRL dataset possesses records that describe personal details, and contains addresses and numerical data. For the DBLP-ACM and Amazon-Google datasets, we choose to perform record linkage by matching based on single attributes as they are highly descriptive, and increasing the number of attributes matched would not have a significant impact on the utility of the experiments. For the FEBRL dataset, we match based on a combination of multiple attributes.

9.1 DBLP-ACM

The DBLP-ACM dataset [34] is a dataset that contains records of the details of scientific publications sourced from two different bibliographic archives - DBLP and ACM. This dataset was created with record linkage tasks in mind, and has a manually verified [34] ground truth table. The records contain titles of the publications, the author list, venue and year of publication.

9.2 Amazon-Google

The Amazon-Google dataset [35] is an e-commerce dataset that matches amazon products with search results from the Google Base Data API. The ground-truth was determined using a universal product code, but this is not used as an input to the matching task. The records contain the product title and a variable length textual product description.

9.3 Freely Extensible Biomedical Record Linkage (FEBRL)

The Freely Extensible Biomedical Record Linkage (FEBRL) [36] is a data mining framework that can be used to clean and link datasets. The Python Record Linkage Toolkit [37] comes with four datasets created with this framework for the purposes of record linkage. We use the fourth dataset "febrl4" which consists of 5000 original and 5000 duplicate records (duplicates contain spelling errors, incorrect numbers, etc.), and an attached ground truth. The records are in the form of biomedical patient data, and contains the following attributes: id, patient name, address, date of birth and social security id.

Chapter 10

Experimental results

We perform experiments to empirically evaluate the performance and accuracy of our protocol. To summarize, we show the following

- Our new protocol with the new approximate equality operator performs 6 to 31 times faster than the FHIPE-based protocol for our chosen vector lengths, the advantage differing based on vector length.
- Our protocol achieves high accuracy across metrics on record linkage tasks, for the vector lengths chosen, with further gains possible by parameter tuning.

10.0.1 Setup

The implementation of the FHIPE protocol uses code provided by the authors of the paper provided through github [38]. It is implemented in python, has a C based backend, and depends on FLINT for the C backend. Charm is used as the pairing library. The Berlekamp-Welch decoder was implemented based on code by Jeremy Kun (Google) on github [39]. We use a modified base Google BERT model and framework from SBERT.net [40] to implement the Encode() function.

10.0.2 Accuracy

We measure accuracy using our value inequality distance function which is used by both secure equality operators, our new one and the baseline using FHIPE. We base our accuracy

results on the following four metrics: Precision, Recall, The Precision-Recall curve, and the ROC (receiver operating characteristic) curve. A brief description of each of these in the context of a record linkage system is as follows

- Precision: It is the proportion of outputted matches that were actually correct matches. Let the the number of true positive outputs be tp, and the number of false positive outputs be fp.

$$Precision = \frac{tp}{tp + fp}$$

- Recall: It is the proportion of actual matches identified correctly as matches in the output. Let the the number of true positive outputs be tp, and the number of false negative outputs be fn.

$$Recall = \frac{tp}{tp + fn}$$

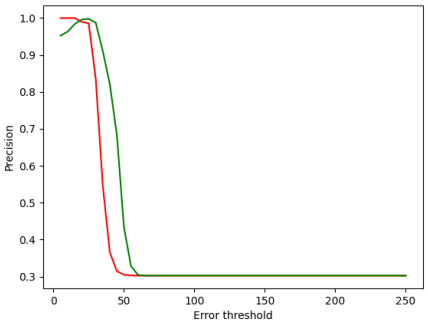
- Precision-Recall curve: It is a graph plotted between precision and recall at different thresholds, and is intended to show tradeoffs between them. A good precision-recall curves maximises the area under the curve.

- ROC curve: The receiver operating characteristic curve is a graph plotted between Recall and False Positive Rate (fpr) at different thresholds. Let the number of false positives be fp, and the number of true negatives be tn.

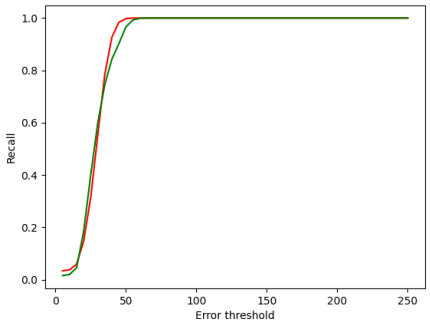
$$fpr = \frac{fp}{fp + tn}$$

  The Area under the curve (AUC-ROC) of the ROC curve is a summary statistic that indicates the probability that the system would be able to distinguish a match from a non-match.

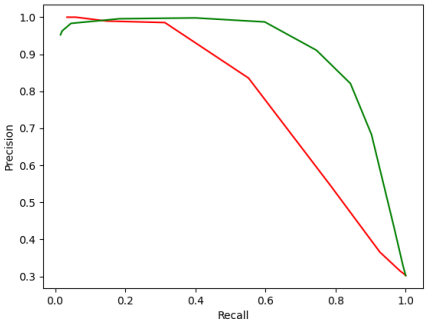
A model based on the base BERT model, with modified pooling layers was used to output record embeddings through an implementation of the Encode() function. The architecture was varied to obtain vector lengths of 128, 192 and 256 for the encoded records. We use an interval size (n) of 8 for the integer conversion. The four metrics discussed above were recorded for record linkage tasks with the DBLP-ACM, Amazon-Google, and FEBRL datasets. The distance metric used is element inequality. The results can be seen in Figures 10.1 through 10.9. The architecture of the BERT model could be



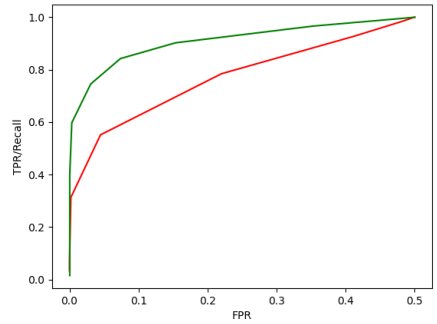
(a) Precision



(b) Recall

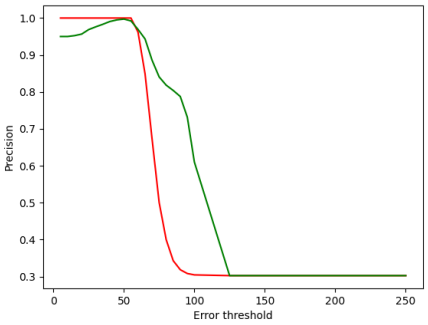


(c) Precision-recall curve

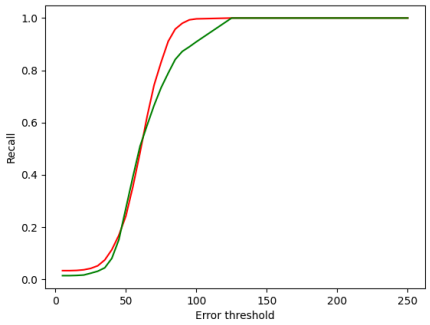


(d) ROC curve

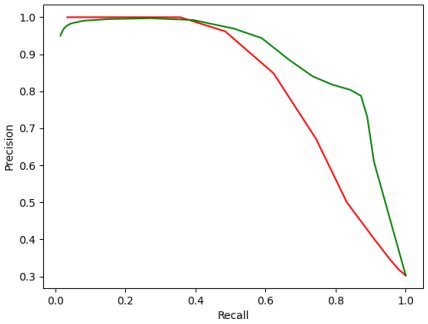
Figure 10.1: Metrics on the Amazon-Google dataset, vector length 128. Matched using titles (red). Matched using product description (green).



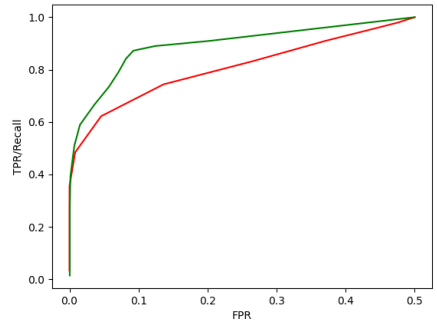
(a) Precision



(b) Recall

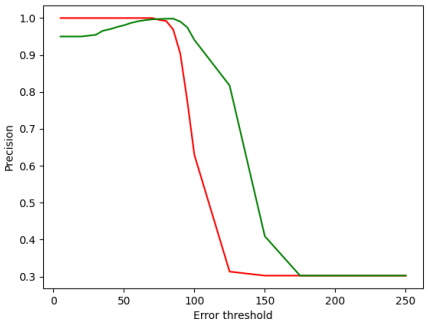


(c) Precision-recall curve

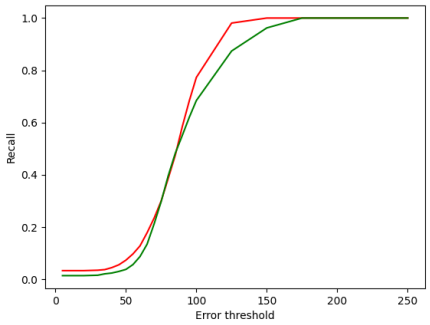


(d) ROC curve

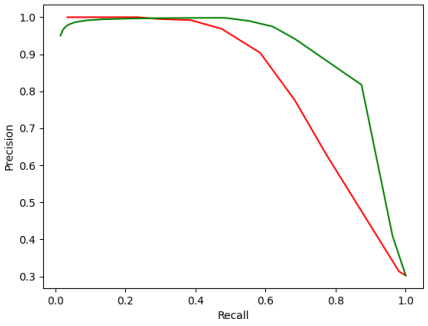
Figure 10.2: Metrics on the Amazon-Google dataset, vector length 256. Matched using titles (red). Matched using product description (green).



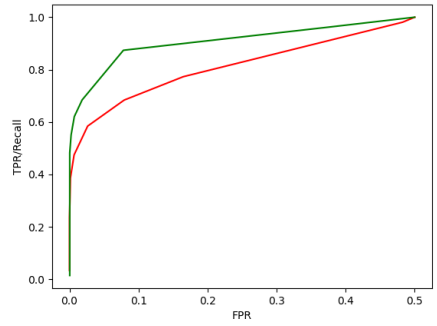
(a) Precision



(b) Recall

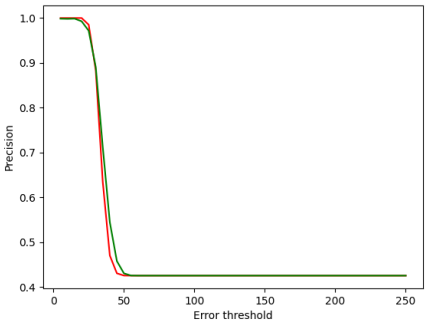


(c) Precision-recall curve

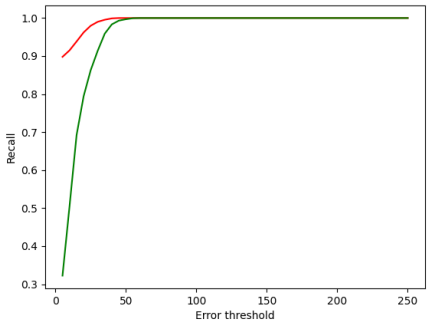


(d) ROC curve

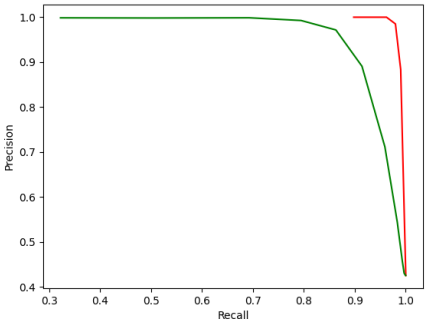
Figure 10.3: Metrics on the Amazon-Google dataset, vector length 384. Matched using titles (red). Matched using product description (green).



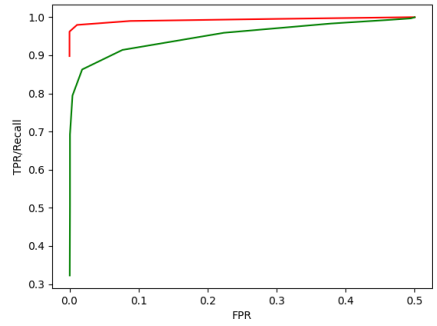
(a) Precision



(b) Recall

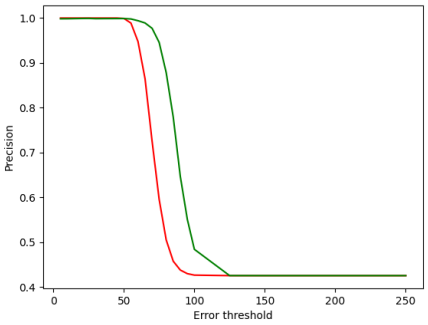


(c) Precision-recall curve

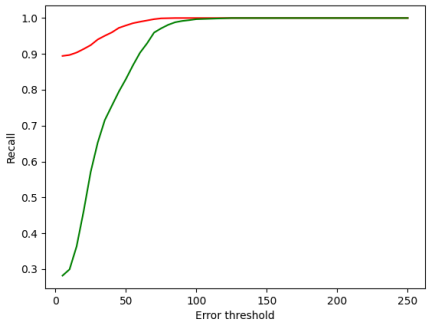


(d) ROC curve

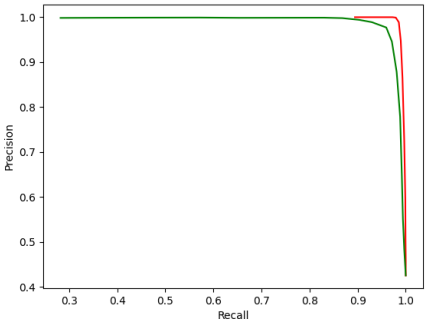
Figure 10.4: Metrics on the DBLP-ACM dataset, vector length 128. Matched using titles (red). Matched using authorlist (green).



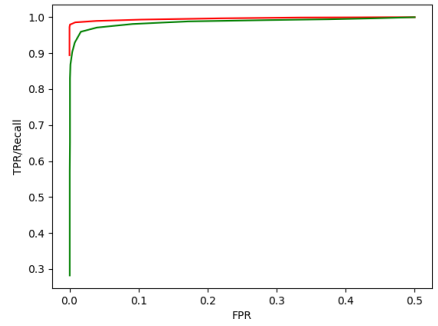
(a) Precision



(b) Recall

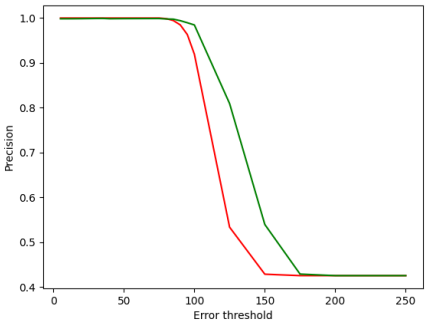


(c) Precision-recall curve

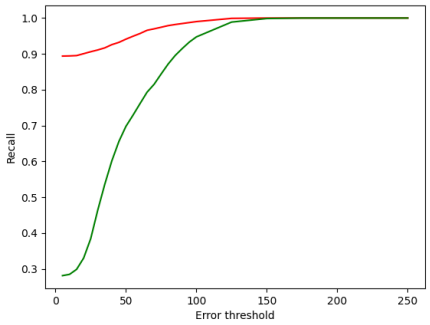


(d) ROC curve

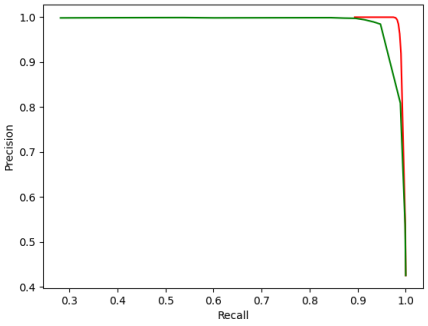
Figure 10.5: Metrics on the DBLP-ACM dataset, vector length 256. Matched using titles (red). Matched using authorlist (green).



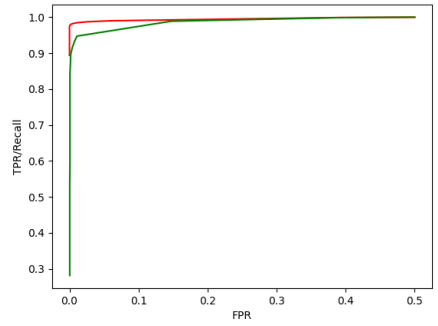
(a) Precision



(b) Recall

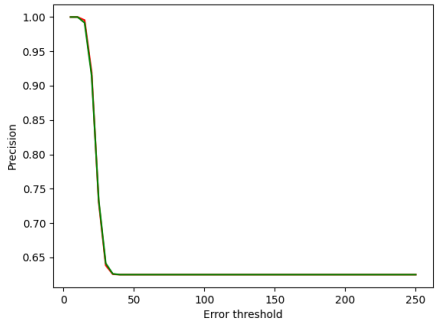


(c) Precision-recall curve

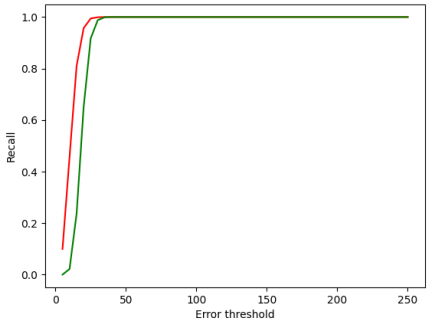


(d) ROC curve

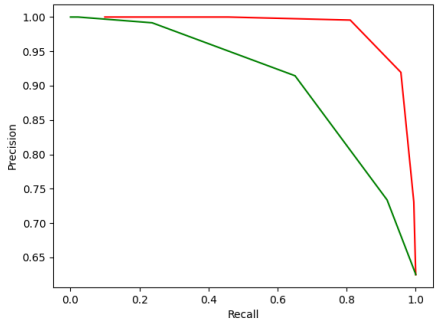
Figure 10.6: Metrics on the DBLP-ACM dataset, vector length 384. Matched using titles (red). Matched using authorlist (green).



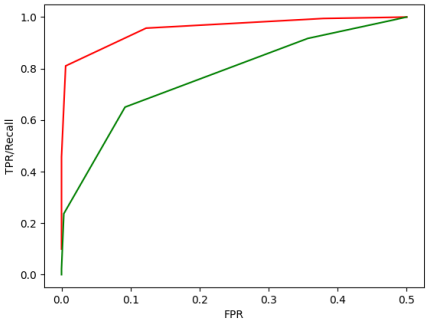
(a) Precision



(b) Recall

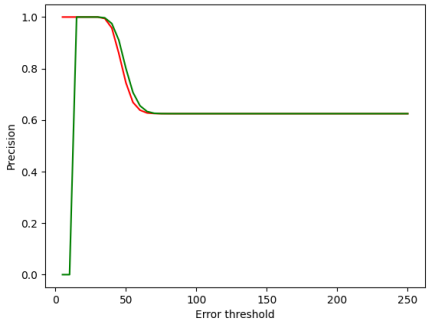


(c) Precision-recall curve

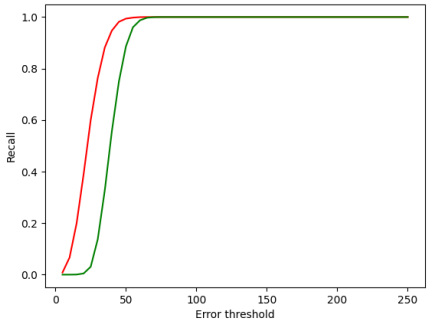


(d) ROC curve

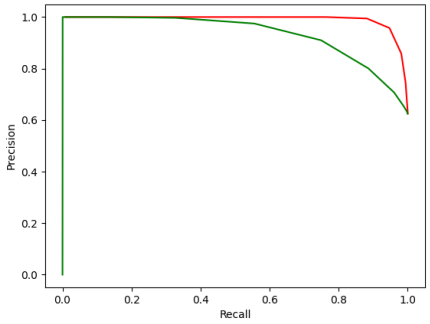
Figure 10.7: Metrics on the FEBRL dataset, vector length 128. Matched using full records (red). Matched using full records from one database, and records with a chosen small subset of attributes in the second database (green).



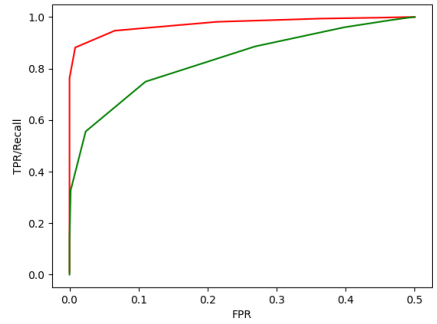
(a) Precision



(b) Recall

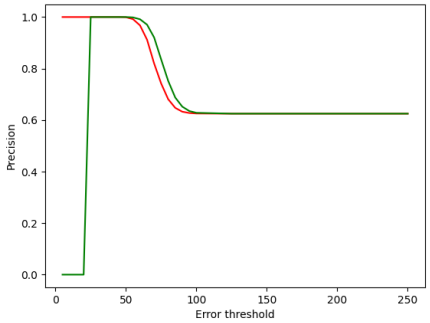


(c) Precision-recall curve

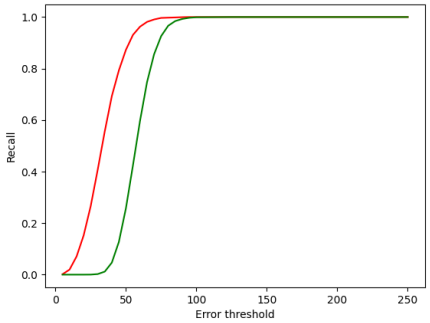


(d) ROC curve

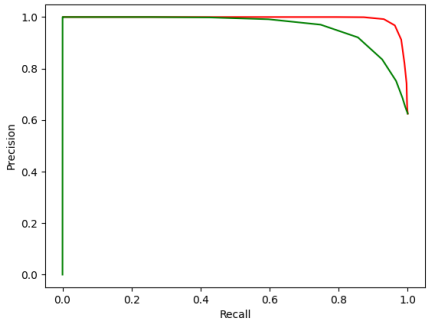
Figure 10.8: Metrics on the FEBRL dataset, vector length 256. Matched using full records (red). Matched using full records from one database, and records with a chosen small subset of attributes in the second database (green).



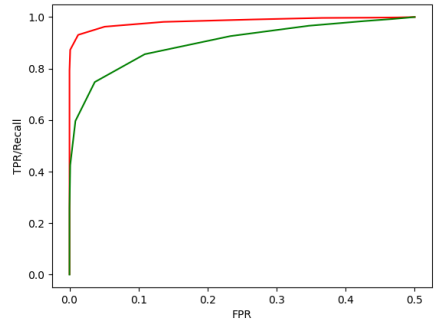
(a) Precision



(b) Recall



(c) Precision-recall curve



(d) ROC curve

Figure 10.9: Metrics on the FEBRL dataset, vector length 384. Matched using full records (red). Matched using full records from one database, and records with a chosen small subset of attributes in the second database (green).

tuned further to suit the current task, thus even higher accuracy is possible. We leave this task specific parameter tuning to future work and implementation.

During experiments on the DBLP-ACM dataset, a small number (At most 2 false positives out of 3000 ground truth non-matches, at strict thresholds) of false positives occur while using the author-list attribute due to non matching records with the same or near same set of authors. The precision graph for an attribute that has a constant number of false-positives tends to shift downwards as the threshold gets more restrictive, because while the number of false positives is constant, the number of true positives declines and hence the ratio also declines. This cannot be clearly seen in the graphs for the DBLP-ACM dataset as the number of true positives is comparatively high even at strict thresholds (1110 true positives at an error threshold of 5).

For the Amazon-Google dataset when using item descriptions, we again observe a constant number of false positives at stricter thresholds, because some item descriptions are empty, which will match according to the matching criteria but are not actual matches. The trend of the graph shifting downwards at stricter thresholds can be seen in Figures 10.1, 10.2 and 10.3.

As can be seen from the graphs, the optimal error threshold shifts upwards as the vector length increases, and the precision-recall and ROC curve are closer to ideal. The precision-recall curve and ROC curve demonstrates that using the element inequality distance metric to match records encoded using BERT offer high usability for record linkage tasks.

10.0.3 Performance

We test the performance metrics of our protocol versus the protocol based on FHIPE. We do this through micro-benchmarks, using the vector lengths chosen for accuracy. Error thresholds are chosen approximately based on accuracy metrics from the three databases tested. We now discuss the performance of the two protocols separately.

FHIPE protocol: The offline phase of the protocol is computationally intensive [26], and using simulated parameters (specifically matrices) has no impact on the inner product operations. Thus, we choose to use simulated parameters in our experiments.

The running time for single inner products for various vector lengths needs to be multiplied by the size of the threshold, since each product can only test if the inner product corresponds to a single integer value in the interval.

Vector length	Error threshold	Berlekamp-Welch Compare()	FHIPE Compare()
128	40	3.6	112.9
256	60	30.2	340.3
384	80	102.5	684.1

Table 10.1: Compute time comparison, all times are in seconds

Our protocol: The Berlekamp-Welch algorithm runs in $O(l^3)$. We use a micro-benchmark averaged over 100 runs to check the running time of a single matching operation. The degree of the polynomial was chosen using the formula for the maximum number of errors that can be corrected, and the threshold calculated from the results of the accuracy section.

Chapter 11

Conclusion

In this paper, we construct a protocol for performing private record linkage in the streaming data model. We show that this protocol is secure in the semi-honest data model. We use Google BERT to generate embeddings for records, and show that by using the element inequality metric, we can achieve high accuracy metrics, and enable the usage of a secure approximate equality operator to perform private record linkage.

Future work includes task specific tuning of parameters for even higher accuracy metrics, and testing of possibly more optimized decoding strategies using the distribution of number of errors in matches vs non-matches. An implementation in a higher performance programming language can also yield better performance. Our secure approximate equality operator has other applications than record linkage, whose exploration is future work.

References

[1] H. L. Dunn: Record linkage. *Amer. J. Public Health Nations Health, vol. 36, no. 12, pp. 1412–1416, 1946.*

[2] William E. Winkler: The state of record linkage and current research problems. *Statistical Research Division, US Census Bureau. 1999.*

[3] Ivan P. Fellegi, and Alan B. Sunter: A theory for record linkage. *Journal of the American Statistical Association 64, no. 328 (1969): 1183-1210.*

[4] William E. Winkler: Overview of record linkage and current research directions. *Bureau of the Census 2006.*

[5] Dinusha Vatsalan, Peter Christen, Vassilios S.Verykios: A taxonomy of privacy-preserving record linkage techniques. *Information Systems. 2013 Sep 1;38(6):946-69.*

[6] Aris Gkoulalas-Divanis, Dinusha Vatsalan, Dimitrios Karapiperis, and Murat Kantarcioglu: Modern Privacy-Preserving Record Linkage Techniques: An Overview. *IEEE Transactions on Information Forensics and Security (2021).*

[7] R. Schnell, T. Bachteler and J. Reiher: Private record linkage with bloom filters. *Social Statistics: The Interplay among Censuses, Surveys and Administrative Data (2010): 304-9.*

[8] E. Durham, M. Kantarcioglu, Y. Xue, C. Toth, M. Kuzu and B. Malin: Composite bloom filters for secure record linkage. *IEEE transactions on knowledge and data engineering 26.12 (2013): 2956-2968.*

[9] Xi He, Ashwin Machanavajjhala, Cheryl Flynn, Divesh Srivastava: Composing Differential Privacy and Secure Computation: A Case Study on Scaling Private

Record Linkage. *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. 2017.*

[10] F.-Y. Rao, J. Cao, E. Bertino and M. Kantarcioglu: Hybrid private record linkage: Separating differentially private synopses from matching records, *ACM Transactions on Privacy and Security, vol. 22, no. 3, 2019.*

[11] Basit Khurram, Florian Kerschbaum: SFour: A Protocol for Cryptographically Secure Record Linkage at Scale, *2020 IEEE 36th International Conference on Data Engineering (ICDE).*

[12] Brassard, Gilles, Claude Crépeau, Jean-Marc Robert: All-or-nothing disclosure of secrets. *Conference on the Theory and Application of Cryptographic Techniques. Springer, Berlin, Heidelberg, 1986.*

[13] Peter Christen, Rainer Schnell, Dinusha Vatsalan, Thilina Ranbaduge: Efficient Cryptanalysis of Bloom Filters for Privacy-Preserving Record Linkage. *Pacific-Asia Conference on Knowledge Discovery and Data Mining. Springer, Cham, 2017.*

[14] P. Christen, T. Ranbaduge, D. Vatsalan and R. Schnell: Precise and fast cryptanalysis for bloom filter based privacy-preserving record linkage. *IEEE Transactions on Knowledge and Data Engineering, 2018.*

[15] P. Christen, A. Vidanage, T. Ranbaduge and R. Schnell: Pattern-mining based cryptanalysis of bloom filters for privacy-preserving record linkage. *PAKDD, 2018.*

[16] M. Kroll, S. Steinmetzer: Automated cryptanalysis of bloom filter encryptions of health records. *Proceedings of the International Joint Conference on Biomedical Engineering Systems and Technologies - Volume 5 BIOSTEC 2015, 2015.*

[17] Yehuda Lindell, Benny Pinkas, Secure Multiparty Computation for Privacy-Preserving Data Mining. *Journal of Privacy and Confidentiality 2008.*

[18] D. Evans, V. Kolesnikov, M. Rosulek: A pragmatic introduction to secure multi-party computation, *Foundations and Trends in Privacy and Security, vol. 2, 2018.*

[19] Radhika Bhargava, Chris Clifton: When Is a Semi-honest Secure Multiparty Computation Valuable?, *Decision and Game Theory for Security, 10th International Conference, GameSec 2019.*

[20] Kuzu M, Kantarcioglu M, Durham E, Malin B: A constraint satisfaction cryptanalysis of Bloom filters in private record linkage. *International Symposium on Privacy Enhancing Technologies Symposium 2011 Jul 27 (pp. 226-245). Springer, Berlin, Heidelberg.*

[21] Debabrata Dey, Vijay Mookerjee, Dengpan Liu: Efficient Techniques for Online Record Linkage. *IEEE Transactions on Knowledge and Data Engineering (Volume: 23, Issue: 3, March 2011)* .

[22] Dimitrios Karapiperis, Aris Gkoulalas-Divanis, Vassilios S. Verykios: Fast schemes for online record linkage *Data Mining and Knowledge Discovery. 2018.*

[23] Dimitrios Karapiperis, Aris Gkoulalas-Divanis, and Vassilios S. Verykios: LSHDB: a parallel and distributed engine for record linkage and similarity search. *2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW). IEEE, 2016.*

[24] Dimitrios Karapiperis, Aris Gkoulalas-Divanis, Vassilios S. Verykios :Summarization Algorithms for Record Linkage." *EDBT. 2018.*

[25] Banda Ramadan, Peter Christen, Huizhi Liang, Ross W. Gayler: Dynamic Sorted Neighborhood Indexing for Real-Time Entity Resolution. *Journal of Data and Information Quality Volume 6 Issue 4 October 2015.*

[26] Sam Kim, Kevin Lewi, Avradip Mandal, Hart Montgomery, Arnab Roy, and David J. Wu: Function-Hiding Inner Product Encryption is Practical. *International Conference on Security and Cryptography for Networks. Springer, Cham, 2018.*

[27] Martin Franke, Ziad Sehili,Marcel Gladbach, Erhard Rahm: Post-processing Methods for High Quality Privacy-Preserving Record Linkage. *Data privacy management, Cryptocurrencies and Blockchain technology. Springer, Cham, 2018. 263-278.*

[28] Adelene Y.L. Sim, Andrew Borthwick: Record2Vec:Unsupervised Representation Learning for Structured Records. *2018 IEEE International Conference on Data Mining (ICDM). IEEE, 2018.*

[29] Jeremy Foxcroft, Adrian d'Alessandro, Luiza Antonie: Name2Vec:Personal Names Embeddings. *Canadian Conference on Artificial Intelligence. Springer, Cham, 2019.*

[30] Philipp Behnen, Felix Kruse and Jorge Marx Gómez: Enhancement of Record Linkage by Using Attributes containing Natural Language Text. *AAAI Spring Symposium: Combining Machine Learning with Knowledge Engineering. 2021.*

[31] Adi Shamir: How to share a secret. *Communications of the ACM 22.11 (1979): 612-613.*

[32] Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova: BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *NAACL 2019.*

[33] Masoumeh Shafieinejad, Suraj Gupta, Jin Yang Liu, Koray Karabina, Florian Kerschbaum: Equi-Joins over Encrypted Data for Series of Queries. *arXiv preprint arXiv:2103.05792 (2021).*

[34] Hanna Köpcke, Andreas Thor, Erhard Rahm: Learning-Based Approaches for Matching Web Data Entities. *IEEE Internet Computing 14.4 (2010): 23-31.*

[35] Hanna Köpcke, Andreas Thor, Erhard Rahm: Evaluation of entity resolution approaches on real-world match problems. *Proceedings of the VLDB Endowment 3.1-2 (2010): 484-493.*

[36] Peter Christen: Febrl – A Freely Available Record Linkage System with a Graphical User Interface. *KDD '08: Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining.*

[37] Python record linkage toolkit;Datasets: https://recordlinkage.readthedocs.io/en/latest/ref-datasets.html

[38] Sam Kim, Kevin Lewi, Avradip Mandal, Hart Montgomery, Arnab Roy, and David J. Wu: Function Hiding Inner Product Encryption is practical (repository) https://github.com/kevinlewi/fhipe

[39] Jeremy Kun: The Welch-Berlekamp Algorithm for Correcting Errors in Data; https://github.com/j2kun/welch-berlekamp/tree/master

[40] Nils Reimers, Iryna Gurevych Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. *arXiv preprint arXiv:1908.10084 (2019).* https://www.sbert.net/index.html