1-1-2019

Probabilistic Record Linkage with Elliptic Curve Operations

Shreya Dhiren Patel
University of Denver

Probabilistic Record Linkage with Elliptic Curve Operations

A Thesis
Presented to
the Faculty of the
Daniel Felix Ritchie School of
Engineering and Computer Science
University of Denver

In Partial Fulfillment
of the Requirements for the Degree
Master of Science

---

by
Shreya Dhiren Patel
March 2019
Advisor: Rinku Dewri

Author: Shreya Dhiren Patel
Title: Probabilistic Record Linkage with Elliptic Curve Operations
Advisor: Rinku Dewri
Degree Date: March 2019

Abstract

Federated query processing for an electronic health record infrastructure enables large epidemiology studies using data integrated from geographically dispersed medical institutions. However, government imposed privacy regulations prohibit disclosure of patient's health record outside the context of clinical care, thereby making it difficult to determine which records correspond to the same entity in the process of query aggregation.

Privacy-preserving record linkage is an actively pursued research area to facilitate the linkage of database records under the constraints of regulations that do not allow the linkage agents to learn sensitive identities of record owners. In earlier works, scalability has been shown to be possible using traditional cryptographic transformations such as Pohlig-Hellman ciphers, precomputations, data parallelism, and probabilistic key reuse approaches.

This work proposes further optimizations to improve the runtime of a linkage exercise by adopting elliptic curve based transformations that are mostly additive and multiplicative, instead of exponentiations. The elliptic curve operations are used to improve the precomputation time, eliminate memory intensive comparisons of encrypted values and introduce data structures to detect negative comparisons. This method of record linkage is able to link data sets of the order of a million rows within 15 minutes. The approach has been gauged using synthetic and real world demographics data with parametric studies. We have also assessed the residual privacy risk of the proposed approach.

Acknowledgements

I would like to express my sincere gratitude towards my thesis advisor, Dr. Rinku Dewri, for guiding me through every stage of my research work. His encou-ragement and guidance always motivated me to stay passionately indulged in the research. I was honoured to work with him and gain a profoundly enriched experience.

I am very thankful to Dr. Matt Rutherford and Dr. Yun-Bo Yi for being a part of my Oral Defense Committee. Their valuable insights for my work helped me extensively in ameliorating the work.

I am very grateful to be a part of University of Denver and I am extremely thankful to the Department of Computer Science at University of Denver for providing me with the resources and a friendly environment to successfully complete my research work.

I would like to heartily thank my parents, Dhiren and Dimple, and my sister, Pearl, for their constant support, love and encouragement throughout my thesis. I am also very thankful to my friends and colleagues who have been extremely supportive during my thesis.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

The Electronic Health Record (EHR) infrastructure has made it easy to perform data analysis over medical records and enable comprehensive epidemiology studies in the field of medical research. Federated Query Processing systems provide a useful interface for such medical research over geographically distributed health records. Federated query processing requires standardization of the dispersed data, by removing the presence of duplicate entities and thereby, linking data from different data sources to unique entities. The process of linking the records present in different data sets but referring to a single entity, where the records may or may not share common identifiers, is referred to as record linkage.

However, patient's medical records contain sensitive private information and holds constraints over sharing of the data with multiple entities. Many countries have strict laws over the sharing of medical information of an individual, which restricts the data access exclusively to the health-care provider (health professional) and the patient himself. For example, the Health Insurance Portability and Account-ability Act (HIPAA) [1] prohibits the disclosure of personally identifiable health information that can be used to contact, identify or locate a unique individual. The identifiers protected by HIPAA includes name, address, dates (date of birth, date of death (if applicable), admission date, discharge date), telephone number, finger/voice print, vehicle number, etc. This challenges data analysis over medical records, since different records belonging to the same entity may be present

at multiple locations and conventional record linkage techniques require the information to be revealed to a third party. Hence, a privacy-preserving record linkage is substantially important in the federated query processing over medical data sets. The idea of record linkage was proposed way ahead of time but record linkage in a privacy-preserving setting was explored as the need to maintain data privacy increased.

The problem statement that we address in this thesis is as follows: Consider two sites or parties, site S_A and site S_B with respective databases, D_A and D_B. D_A and D_B contains multiple sensitive records with one or more attributes. Sites S_A and S_B want to determine what percentage of data overlap exists between the data sets D_A and D_B, without revealing their confidential data to each other or to a third party. The third party, a Linkage Agent, performs the process of record linkage using the encoded databases, received from S_A and S_B.

The initial approaches for private record linkage used the one-way hash functions to encode the data before the process of record linkage. But encoding the data with one-way hash functions was insufficient and unreliable due to the presence of data inconsistencies like data entry errors and data duplication. Hence, optimizations were made which involved switching from exact matching approaches to approximate matching approaches using n-grams (substrings of length n). Due to the lack of privacy guarantees to the approximate matching approach against dictionary attacks and frequency-based attacks, the technique of using Bloom filters was explored but at the expense of linkage performance. Later, it was proposed to use a commutative encryption method with modular exponentiation to compute private set intersections. But with higher security, comes heavy computation and communication complexity while handling big data. Such an approach made the commutative encryption process highly expensive and infeasible with the available hardware back then. In 2016, Dewri et al. proposed the Pohlig-Hellman approach which uses the Pohlig-Hellman exponentiation cipher in a set intersection protocol, and obtained improvements with precomputations and data parallelism, making the task of private record linkage feasible with the current hardware advancements under a commutative encryption scheme [9].

The work in this thesis proposes an Elliptic Curve (EC) based approach with various upgrades that can be implemented against the Pohlig-Hellman approach and intends to show a significant improvement in the record linkage time for big data along with detailed analysis of the residual privacy risk of the approach. The primary contributions of the

EC-based approach are as follows: First, we optimize the precomputation procedure by replacing modular exponentiations of the Pohlig-Hellman approach with EC-based additive and multiplicative one-way transformations. Second, we eliminate the need to communicate huge precomputed tables (containing EC points of long memory bytes) of encrypted data values and replace them with lookup tables based on indices. This avoids huge memory comparisons of the encrypted values. Third, we improve the linkage time by using hash maps to avoid unnecessary comparisons in the matching process.

1.1 Thesis Outline

The overview of the thesis is as follows:

- Related Work: We discuss previous works in the field of record linkage and relevant contributions on improving and securing the process of private record linkage in this chapter.
- Background: This chapter covers description of the background to understand the proposed framework in the thesis. It explains the architecture followed in the Federated Query Processing System and the matching algorithm underlying the system.
- Private Record Linkage using Key Rings: This chapter discusses the methods used in the EC-based approach of record linkage along with the implementation details used to generate the results in the thesis.
- Results: This chapter presents the results obtained by performing different parametric experiments, accompanied with time based comparisons of previous work and the current work.
- Conclusions and Future Work: This chapter summarizes the contributions of the work in the thesis and briefly discusses the different directions that can be explored for possible optimizations of the matching process.

Chapter 2

Related Work

2.1 Initial Work

Dunn initialized the idea of record linkage in the year 1946, metaphorizing it with "the process of assembling the pages of the Book of Life into a volume" [3]. His paper emphasizes on how to perform a record linkage over the distributed records of individuals that can help infer coordinated statistical information for the health and welfare organizations, using the idea of linking data that was used by the Canadian system for distributing family allowances, proposed in the Family Allowances Act, Canada, 1944 [15]. Fellegi and Sunter proposed a model for record linkage with statistical and probabilistic linkage rules [4]. These linkage rules are similar to mathematical decision rules (mapping of an observation into an action). They suggested that due to the presence of errors and incompleteness in the records of two databases, there will be cases where the database linking might wrongly conclude two unmatched records to be identical and conversely, two matched records to be non-identical, resulting in unreliable matchings. Hence it is important to consider the error levels while devising an optimal linkage rule. A brief explanation of their method is as follows.

Consider two databases D_A and D_B with some common attributes. Let record $a \in D_A$ and record $b \in D_B$. The set of ordered pairs of records is generated as follows.

$$D_A \times D_B = \{(a,b); a \in D_A, b \in D_B\}$$

The process to link records involves comparing the record pairs and categorizing them into a matched pair or an unmatched pair which can be defined as follows.

$$M = \{(a,b); a = b, a \in D_A, b \in D_B\}$$

$$U = \{(a,b); a \neq b, a \in D_A, b \in D_B\}$$

Each record pair can be linked to a status, positive link P, where P is the random variable defined as:

$$(a,b) \in M \Longrightarrow P = 1$$

$$(a,b) \in U \Longrightarrow P = 0$$

The comparison of records is done with a comparison function $\gamma(a,b)$ on each attribute. The set of all possible observations of the function γ is defined as the comparison space τ. The function $\gamma(a,b)$ will decide whether a record pair (a,b) is a positive link $((a,b) \in M)$, (a,b) is a positive non-link $((a,b) \in U)$ or (a,b) is a possible link (when there is not enough information to put it into either sets M or U). Thus, based on the observation of the function $\gamma(a,b)$, either of the three actions will be performed denoted by A_1, A_2 and A_3 respectively.

Considering this categorization, a linkage rule L, based on conditional probabilities, will be defined as a mapping from the comparison space τ onto a set of random decision functions $d(\gamma)$. The decision function $d(\gamma)$ will be defined as:

$$d(\gamma) = \{P(A_1|\gamma), P(A_2|\gamma), P(A_3|\gamma)\}; \gamma \in \tau$$

$$\sum_{i=1}^{3} P(A_i|\gamma) = 1$$

A linkage rule may have either of the following 2 types of errors:

- Unmatched comparison is concluded as a positive link:

$$\mu = P(A_1|P = 0)$$

- Matched comparison is concluded as a positive non-link:

$$\lambda = P(A_2|P = 1)$$

A linkage rule on the comparison space τ with error levels μ and λ where the values $\mu \in (0,1)$ and $\lambda \in (0,1)$ is defined as:

$$L(\mu, \lambda, \tau)$$

The optimal linkage rule will be the one that minimizes the probability of the possible link action A_3 for the error levels μ and λ.

Winkler elaborately discussed and performed surveys [16] over the different cases (records being a match or non-match) arising in the process of performing record linkage using Fellegi and Sunter's approach. He also suggested suitable methods that could be applied in those cases. But these approaches did not consider preserving data privacy.

2.2 Using Hash Functions

The proposal to perform record linkage in a privacy-preserving setting was initiated by Dusserre et al. [5], Bouzelat et al. [27] and Grannis et al. [40], who suggested the use of non-reversible one-way hash functions in enciphering the data. A basic protocol to perform a record linkage using hash functions is as follows:

- Consider two sites S_A and S_B who want to perform a record linkage where a linkage agent *LA* is the third party who computes the record linkage.
- S_A and S_B will decide upon a secure hash function to be used (example, SHA-Secure Hash Algorithm).
- S_A and S_B compute the hash values of their data and send them to *LA*.
- *LA* compares the hash values received from both sites. If two hash values are same, *LA* concludes that the source values coming from the two data sets is a match.

Schadow and Grannis gave a privacy-preserving framework in a distributed network using the HMAC (Hash-based Message Authentication Code) algorithm [17] for record linkage. According to Schadow and Grannis, since the traditional approach to perform research on medical databases involves aggregating all the data linked to a patient at a single site

and then running required queries over the accumulated data, such large collection of patient's data puts the patient's privacy in danger [10]. Hence, it is more desirable to perform a private record linkage over the distributed databases. Considering this approach, they designed a method based on a strong keyed-hash algorithm (HMAC) to perform bio-medical research on de-identified clinical cases that are distributed across a large network.

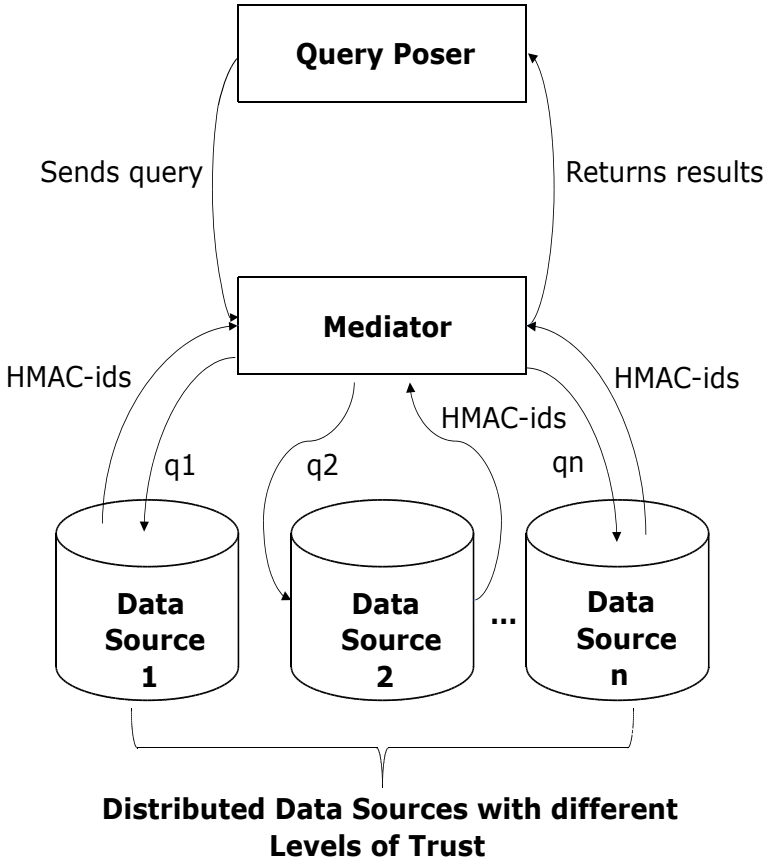


Figure 2.1: Distributed architecture based on HMAC [10]

Figure 2.1 depicts the method proposed by Schadow and Grannis. The framework includes a Query Poser (sends and receives results from the mediator), a Mediator (acts as a linkage agent) and data sources. The HMAC algorithm is used to encrypt the patient's identifiers and then deterministic linkage techniques are applied. The data entry errors (like typographical errors), likely to be present in the databases, are not taken into consideration in the design of this approach. Also, the framework is exposed to dictionary and frequency-based attacks [8].

Churches and Christen elaborated on two different approaches to perform rec-ord matching: exact matching and approximate matching [6]. The exact matching approach involves comparing the strings as a whole (without breaking them into n-grams). The approximate matching approach breaks a string into n-gram substrings and compares the substrings while performing a record matching. Exact matching method of string comparison has the drawback that even a single character difference would result in the algorithm concluding that the two strings are a non-match since they will produce two completely different hash values (in HMAC approach). Durham et al. also confirms that approximate matching approach in privacy-preserving setting is more reliable than the exact matching approach [19].

Churches and Christen present the n-gram similarity comparison technique for secret strings or sequences, to overcome the problem of performing exact matching of confidential data with data entry errors [6, 18]. They present the method where:

- The 2 parties willing to perform a record linkage mutually agree on a secret key, a secure one-way hash function, a standard protocol to preprocess strings (for example, convert all characters to lower case and, remove or replace punctuation and whitespace) and a standard method to perform encryptions using the secret key and the hash function.
- The strings are broken into a set of n-grams. Precomputations are performed which includes building the power set ($2^b - 1$ subsets of the n-gram set, where b is the number of n-grams of the string) of each set of n-grams. For example, the bigram set β for the string "cats" would be $\beta = \{$"ca", "at", "ts"$\}$ and the power set for β, $P(\beta)$ would comprise of

  ("ca"), ("at"), ("ts"),

  ("ca", "at"), ("at", "ts"), ("ca", "ts"),

  ("ca", "at", "ts")

  Each of the subsets are sorted and hash values are generated for them. After necessary precomputations are performed on the set of n-grams, the sets are sent to a trusted third party who performs the matching by calculating the bigram score as follows.

$$bigram\_score = \frac{|\alpha \cap \beta|}{0.5 \times (|\alpha| + |\beta|)}$$

α = set of bigrams of an attribute value of a record in first party

β = set of bigrams of an attribute value of a record in second party

Two strings are considered to be a match if there exist a significant overlap between them (bigram score is high). However, hash function based approaches fail to guarantee protection against dictionary attacks and frequency attacks because of their deterministic nature. Karakasidis and Verykios propose to use phonetic codes for comparisons to perform a probabilistic data linkage [28]. Other probabilistic approaches include the creation of linkage keys from specific indices of data values [41] and embedding data values in a multi-dimensional space [42].

2.3 Using Bloom Filters

To avoid the threat of constructing the original strings from the set of n-grams, Schnell et al. proposed the idea to use Bloom filters to calculate the similarity scores between the set of n-grams of two strings [7, 19].

A Bloom filter is a binary data structure which is used to test the presence of an element in a set. The bits in the Bloom filter are set according to the output of the hash functions applied in encoding the n-grams of a string. Hence, to compare two strings, two respective Bloom filters are set and compared. Running a query on a Bloom filter either returns a false positive (element may be present in the set) or a true negative (element is definitely not present in the set). In the process of record linkage, the two parties can compute their respective Bloom filters and send them to the third party. The third party compares the Bloom filters and generates appropriate results for the two parties. However, feasible attacks to reverse the Bloom filter encodings have been shown to be possible [29–32].

The frequency attacks may be made less feasible by using more number of hash functions [33]. Taking an account of the issues with using a single Bloom filter, Durham et al. [21] and Schnell et al. [34] proposed to use a combination of multiple Bloom filters while calculating the n-gram similarity scores between two strings. Since Bloom filters produce false positives, using multiple bloom filters would lower the performance of record linkage by trading off with the correctness of the method. Cryptanalysis attacks on composite Bloom filters have also been demonstrated [35, 36]. Although Bloom filters seem to be the fastest method for record linkage, much remains to be explored due to their exposure to privacy risks.

2.4 Blocking Procedure

Homomorphic encryption is another approach for record linkage that is used in testing the similarity distances of data values [43]. But the application of this approach to huge data sets comes at a huge expense. In general, when data sets scale to the order of millions, linkage time becomes a matter of concern. Hence, a blocking procedure is used to address the issue [37, 43]. Under this procedure, huge data sets are divided into smaller disjoint data subsets based on either an attribute value or partial information composed of multiple attributes. Under the blocking procedure, creation of empty subsets may leak information and jeopardize data privacy. Hence, privacy treatments are enforced to nullify the issue [44, 45]. Homomorphic encryption has been utilized in optimizing private set intersection protocols too [46].

2.5 Using Secure Multi-party Computation (SMC)

Different private set intersection (PSI) approaches have been proposed using the concept of SMC with oblivious transfer. PSI protocols are relatable to private record linkage since they compute the intersection of sets of two parties without revealing anything other than the intersection.

SMC is a part of cryptography where two or more parties jointly desire to compute a function of their inputs such that their inputs are not publicly disclosed. Thus, SMC can be used to build almost any type of online cryptographic application which involves mutually distrusted parties. Consider an example, the Millionaire's problem [24], involving parties with mutual distrust, described as follows.

2 millionaires, Alice and Bob want to find out who is richer among the both of them without disclosing their wealth. The total valuation of their assets are a and b, respectively. They want to devise a function that calculates the maximum of a and b.

$$F(a,b) = maximum(a,b)$$

Conventionally, they can choose a trusted third party, Tony (who is good at keeping secrets) and each of them can reveal their wealth to Tony. Tony would compare the values and declare who is richer. Unfortunately, this method would involve declaring private data

to a third party or an adversary, which is not wise. SMC helps in devising a function such that the following criteria are satisfied:

- Communication happens only between two parties and the idea of a third party is completely eliminated.
- No party knows about the input of the other party (their wealth remains hidden).

In order to devise protocols for SMC, Rabin introduced the concept of oblivious transfer [25]. Oblivious transfer is a powerful cryptographic protocol to transfer one of the many pieces of information from a sender to a receiver such that the sender has no knowledge of what piece of information (if any) has been received.

Andrew Yao presented the first protocol for two-party secure computation in 1986, to solve the Millionaire's Problem [24]. Essentially, Yao's protocol is a compiler that takes a circuit as its input and transforms it into one which returns nothing but the final output. The functionality is represented in the form of a garbled circuit. A garbled circuit is made of encrypted/garbled logic gates. Garbled gates are similar to regular logic gates, except they operate using encrypted sampled keys instead of bits. It can be described as an encrypted circuit with a pair of keys for every wire such that for any gate, given one key on every input wire :

- One can compute the key of the corresponding gate output.
- Nothing else is learnt.

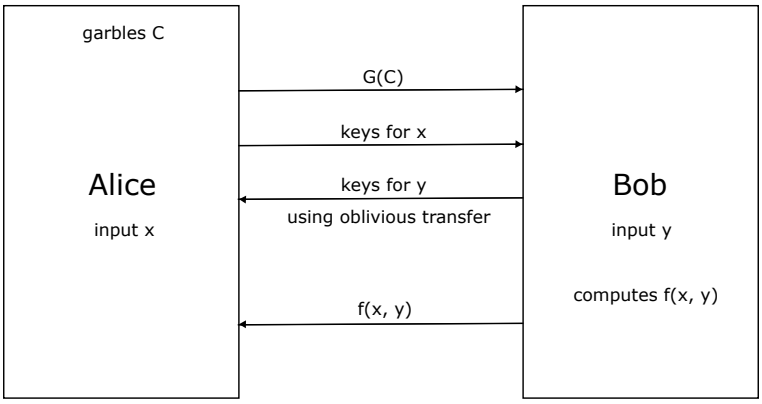


Figure 2.2: Garbled circuit protocol

The garbled circuit protocol is described as follows:

1. The function is described in the form of a boolean circuit. It is known to both parties.

2. Alice constructs a garbled circuit $G(C)$ by encrypting (garbling) the given boolean circuit C.
3. Alice sends $G(C)$ to Bob.
4. Alice sends the keys associated with her input x, to Bob. Bob does not learn the boolean value corresponding to the keys.
5. Alice and Bob run an oblivious transfer protocol so that only Bob learns the keys corresponding to his input y.
6. Bob evaluates the circuit $G(C)$ to retrieve $f(x,y)$.
7. Bob sends the result $f(x,y)$ back to Alice.

Pinkas et al. gave various optimizations to the existing protocol covering efficient ways to implement the Yao's garbled circuit protocol in different settings [23]. But PSI protocols focus on performing few intersections of large sets but are not scalable while performing huge number of intersections over smaller sets which becomes a bottleneck for the private record linkage.

2.6 Using Commutative Encryption

Agrawal et al. first proposed the use of commutative encryption in record linkage [8]. They addressed the problem to perform database linking with minimal sharing using commutative encryption over the data at respective sites. A general encryption scheme to encrypt a data value x using a key k and an encryption function f would be $f(k, x)$. Let there be two keys k_1 and k_2, then using two keys, the encryption can be described as follows.

$$f(k_2, f(k_1, x))$$

where x is first encrypted with key k_1 and the encrypted value is again encrypted with key k_2. Under the commutative property of the encryption function, the following holds true.

$$f(k_2, f(k_1, x)) = f(k_1, f(k_2, x))$$

However, Malin and Airoldi [22] claimed that there exist scalability issues with Agrawal's approach. Dewri et al. proposed a scalable and efficient approach to incorporate commutative encryption scheme in record linkage of big data using precomputed tables [9].

We have used various aspects of their approach as the foundation of EC-based approach proposed in this thesis.

Chapter 3

Background

3.1 Federated Query Processing

A federated query processing system is an information retrieval system which enables concurrent search across multiple distributed data sources. It is used to integrate information from heterogeneous data sources and present it in a standard or partially homogeneous form. A user submits a query to the system and the system compiles results of the query from geographically distributed databases and query engines, involved in the federation.

Figure 3.1 represents a distributed architecture for federated query processing, based on the SAFTINet architecture that provides a scalable and sustainable system to support federated query processing for electronic health records [2]. Authorized users can request data from the partner grid nodes via a web-based query portal. The query portal transmits the queries to a federated query portal (FQP). The FQP contacts each grid node selected by the user and submits the user's query to those grid nodes. Query results are then compiled on FQP and presented to the user. The FQP and each grid node maintain their own list of authorized groups and users, and interact via an intermediate authentication, authorization and accounting (AAA) service. The main components of the architecture are as follows.

- Federated query portal: It is an electronic health record portal which combines geographically dispersed data across medical institutions.
- Authentication, Authorization and Accounting (AAA) service: AAA is a framework to intelligently control access to computer resources, enforce policies, audit usage, and provide necessary billing information for the services.

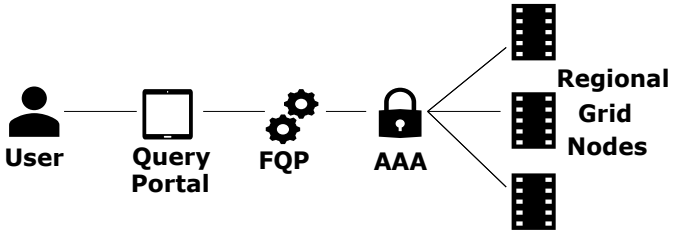


Figure 3.1: Distributed architecture of federated query processing [9]

- Authentication: provides a way to identify the user.
- Authorization: permits to do certain task(s).
- Accounting: measures the resources a user consumes during the access.

Figure 3.2 represents the structure of a regional grid node. Each of the regional grid nodes has a structure similar to the federated query processing system. The regional grid nodes consist of a linkage agent, rFQP (regional FQP), rAAA (regio-nal AAA) service and multiple data sources.

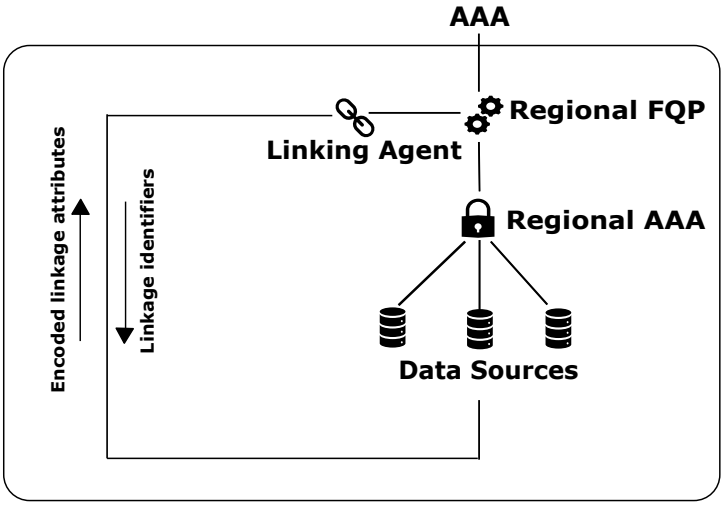


Figure 3.2: Regional grid node [9]

- Data sources: Data sources are made up of a local query engine and a standardized database. The local query engine fetches data from the corresponding database.

- rFQP: A regional FQP receives queries from the FQP of the main system and compiles/forwards the results evaluated from multiple regional data sources. All the query specific results compiled by the rFQP are encrypted and availed only to the authorized query user. No component of the system can view any unencrypted health record.
- rAAA: A regional AAA handles the communication between the rFQP and data sources.
- LA: Linkage agent helps the rFQP to resolve data conflicts like data duplication or data inconsistency in the data fetched from the data source.

The LA defines the uniqueness of the data based on some linkage identifiers (first name, last name, address, phone number, etc.). LA asks the data sources for results on these identifiers. The privacy preserving data source returns encrypted results and then LA resolves the data overlap and inconsistency issues without viewing the data and working with the available encrypted data. On query request, encrypted records are passed to the LA who then provides required results to the rFQP and those results are presented to the user. Federal entities like the National Center for Health Statistics can play the role of a linkage agent in the system.

3.2 Detecting Distributed Records

Record linkage is the procedure to find the records that refer to the same entity, from data sets across distributed data sources with or without the presence of common identifiers in the data sets. Conventionally, record linkage can be performed across multiple databases if the entities of the data sets can be uniquely identified by a primary key identifier or combination of Personally Identifying Information (PII) like name, contact number, address, etc. For instance, in the USA, the Social Security Number (SSN) acts as a national identification number, which is issued by the Social Security Administration and used to keep track of social security purposes (taxation, government benefits, etc.) of U.S citizens, permanent residents and temporary residents in the United States. But there does not exist any medical identifier and hence one has to use a set of PII to identify an entity uniquely, across multiple databases.

3.3 Linkage Issues

Privacy regulations like HIPAA prohibits disclosing personal health information to unauthorized organizations and sets limitations to use a patient's health data without authorization. Because of this, clear text record linkage using multiple identifiers becomes infeasible. This leads us to pursue private record linkage methods. Private record linkage attempts to perform a record linkage between databases of two parties without the need to reveal clear text PII to each other. Ideally, a person's PII is identical at two different databases. In practice, databases might carry data inconsistencies arising due to various aspects like data entry errors, data duplication, difference in data encoding, etc. This motivates us to apply a non-exact matching (approximate matching) in the process of private record linkage instead of an exact matching. Unlike in exact matching, where full strings of data values are compared with each other to check equality between records of two databases, non-exact matching breaks down the strings into sets of n-grams and determines the string equalities based on the extent of overlap between the two n-gram sets.

3.4 Similarity Scores

The underlying n-gram matching algorithm requires to compute similarity scores at attribute and record level of the data sets. We use bigrams ($n = 2$) for the method and calculate similarity scores by breaking down the strings in the data set into a set of bigrams. The similarity scores are calculated as follows.

Let γ denote an ordered set of all possible bigrams constructed from the following 69 characters: A B...Z 0 1...9 ~ ‘ ! @ # $ % ^ & * ( ) - _ + = { } [ ] — \: ; “ ‘ <> , . ? / and the blank space. There will be a total of 4761 possible bigrams in the set γ. Consider two databases D_A and D_B sharing z attributes and owned by sites S_A and S_B respectively. The objective is to perform a private record linkage between these two databases. The process involves performing a privacy preserving join between the given databases considering the z attributes shared by the databases and taking into account the presence of data inconsistencies.

The process of linking the databases under non-exact matching, involves computing record similarity scores between record pairs in the Cartesian product of the records in the databases D_A and D_B. To compute record similarity, attribute similarity score is calculated for corresponding attribute values of the record pair.

3.4.1 Attribute Similarity (Dice's Coefficient)

Attribute similarity between records r_a and r_b in a given attribute i is calculated using Dice's coefficient [9] as follows:

$$S_{Attr}(r_a[i], r_b[i]) = \frac{2|\alpha \cap \beta|}{|\alpha| + |\beta|}$$

where,

α = set of bigrams from $r_a[i]$

β = set of bigrams from $r_b[i]$

The attribute values are divided into bigram sets and the intersection score is calculated by counting the number of common bigrams between the bigram sets. The process to do so is elaborately explained in later sections.

3.4.2 Record Similarity

Record similarity is the similarity value measured between two records of the databases. A weighted similarity across the attributes is considered here, since, not all the attributes or identifiers share the same level of importance (discriminatory power) in record matching. For instance, attributes like "day of birth" (up to 31 possible values) have a higher discriminatory power than "gender information" (only two possible values). Hence, the "day of birth" attribute will hold a higher weightage in contributing towards the record similarity score than some trivial attribute like "gender information" which will have much lower weightage. Record similarity score is calculated as follows.

Let r_a and r_b be two records from databases D_A and D_B respectively. The similarity score S between the two records sharing z attributes is evaluated as:

$$S(r_a, r_b) = \sum_{i=1}^{z} w_i S_{Attr}(r_a[i], r_b[i])$$

where,

w_i = weight of attribute i (float value from [0,1])

$r_x[i]$ = string value in i^{th} attribute of record x

S_{Attr} = similarity score or measure between two strings (one from each record)

It can be seen in Figure 3.3 that a person John Doe can appear in multiple possible ways in different databases due to data inconsistencies. Using the *n*-grams approach and breaking down the strings into bigrams gives more reliable results than performing string comparisons over whole strings. The number of common bigrams is 4 which is then used to calculate attribute similarity and then, record similarity.

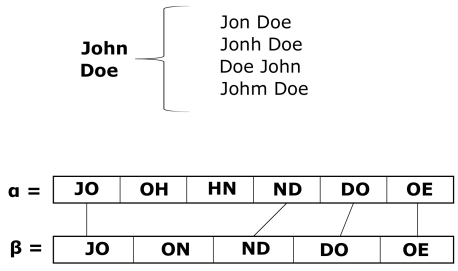


Figure 3.3: Determining common bigrams

3.5 Matching Algorithm

A standard matching algorithm performs a pairwise comparison of the records from the two sites. With each record comparison, a similarity score is assigned and based on the similarity score and pre-decided threshold values, record pairs are divided into three categories:

- Matches: Record pairs with similarity score more than the upper threshold value
- Non-matches: Record pairs with similarity score less than the lower threshold value
- Undecidable: Record pairs with similarity score between the lower and upper threshold values

Threshold values decide what percentage of overlap between a record pair in the record matching is acceptable for a pair to be a match or non-match. Since pairwise comparison of each record from one site is done with each record on another site, the algorithm is a greedy matching algorithm. The matching algorithm is described in Figure 3.4 as follows:

1. Choose a record r_a from D_A.
2. For all records r_b in D_B, compute similarity score $S(r_a, r_b)$ and update the current maximum similarity score $max(a)$ (score for maximum similarity for record in D_A

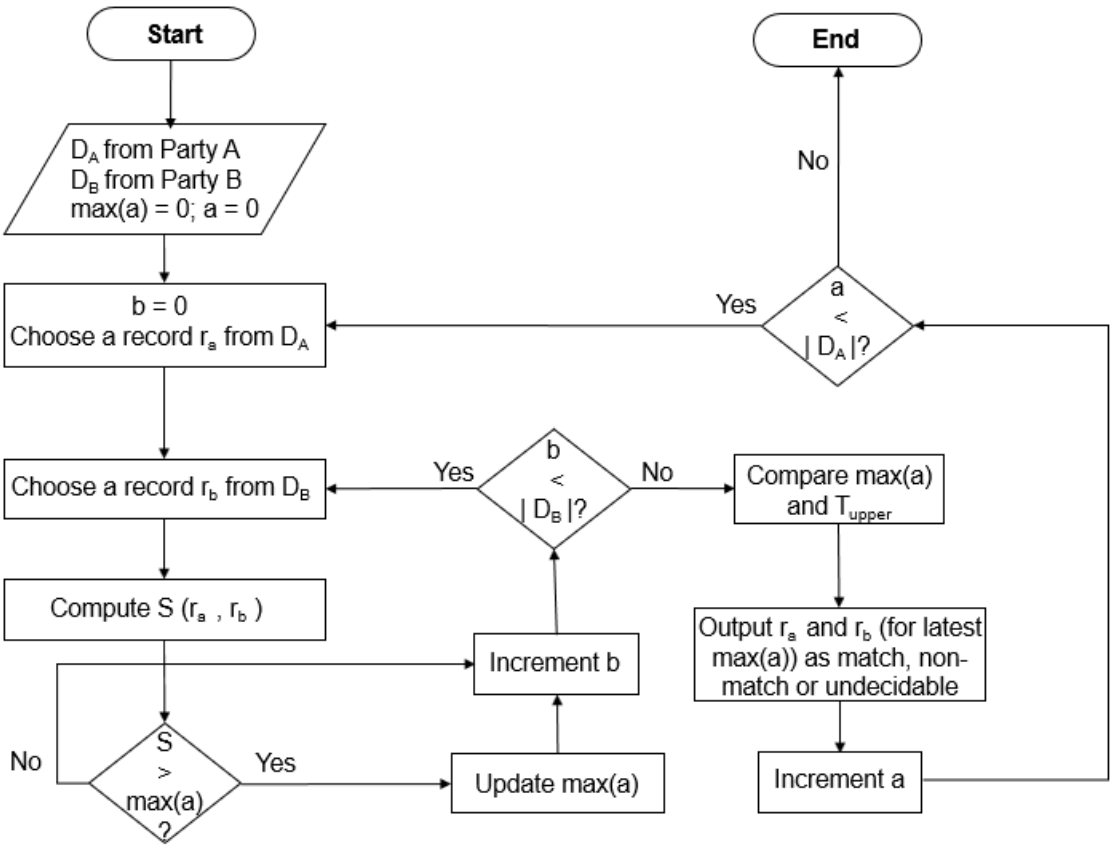


Figure 3.4: Flowchart for matching algorithm

with record in D_B) if a record pair with higher value of S is encountered. Ties are resolved by picking the first record found with the maximum score.

3. Given an upper threshold T_{upper}, if $max(a) \geq T_{upper}$, then output (r_a, r_b) as a match.
4. Repeat step 1 to step 3 until D_A is empty.

3.6 Cryptographic Protocols

The security strength of public-key cryptography is retained due to the intractability of respective mathematical problems. It is infeasible to factor a large integer which is composed of two or more prime numbers. This characteristic is exploited to enhance the security of the public-key systems. RSA [11] is one of the most used public-key cryptosystem, however has costly computations for encryption and decryption due to its large key size

and exponentiation functions involved. Security of subverting RSA is linked to the key size (small is less secure) which typically ranges from 1024 bits to 2048 bits.

3.6.1 Pohlig-Hellman Exponentiation Cipher

Dewri et al. [9] uses the Pohlig-Hellman exponentiation cipher scheme [12] in the context of the matching algorithm proposed for bigram messages to map each bigram to unique values as follows.

Let H be a mapping such that each bigram $b_i \in \gamma$ is mapped to a unique value $1 \leq g_i \leq p-1$; g_i is a primitive root of p, and p is a large prime number. For a key K, encoding E_K of b_i will be computed as:

$$E_K(b_i) = (H(b_i)^K) \, mod \, p$$

After computing transformations for the bigrams on both sites, the encoded sets from both parties are sent to the linkage agent who then performs the linkage process.

But this method imposes limitations over huge data sets due to the usage of exponentiation functions on each bigram. Performing such encodings on each bigram in a huge data set and comparisons on those encrypted values makes the whole process very time consuming and results into high computational costs. The linkage time would significantly reduce if the modular exponentiations can be replaced by simpler mathematical operations like addition and multiplication. Taking this idea into account, we switch from the Pohlig-Hellman approach to the elliptic curve cryptography.

3.6.2 Elliptic Curve Cryptography

Elliptic Curve Cryptography (ECC) provides similar security strength as that of RSA and at a smaller key size of about 256 bits to 360 bits, with additive and multiplicative operations instead of exponentiations, and hence preferred in many applications [13]. Table 3.1 represents a comparison of the security strengths of different cryptographic algorithms.

Security strength of n bits means it would require 2^n operations to break the algorithm for a brute force approach. It can be observed in Table 3.1 that the elliptic-curve based algorithms provide much higher security against their non-elliptic curve equivalents. For example, for a security strength of 256 bits, the asymmetric key algorithms require to use a key of 15360 bits which becomes infeasible due to the need of high computational power, while elliptic curve algorithms require a feasible size of only 512 bit key.

Symmetric Key Algorithms	Asymmetric Key Algorithms	Elliptic Curve Algorithms
80	1024	160
112	2048	224
128	3072	256
192	7680	384
256	15360	512

Table 3.1: Comparable estimated security strength of cryptographic algorithms based on varying key sizes [14]

ECC is public-key cryptography that uses an algebraic elliptic curve over finite fields. In order to understand the idea of elliptic curve, following are some definitions.

Definition 3.6.1. Elliptic curve: An elliptic curve is an algebraic curve in the Euclidean plane defined over four elements in some field with the following equation:

$$y^2 = x^3 + ax + b \tag{3.1}$$

Figure 3.5 shows a sample elliptic curve over the Euclidean plane with the constant values $a = -2$ and $b = 2$ in equation 3.1.

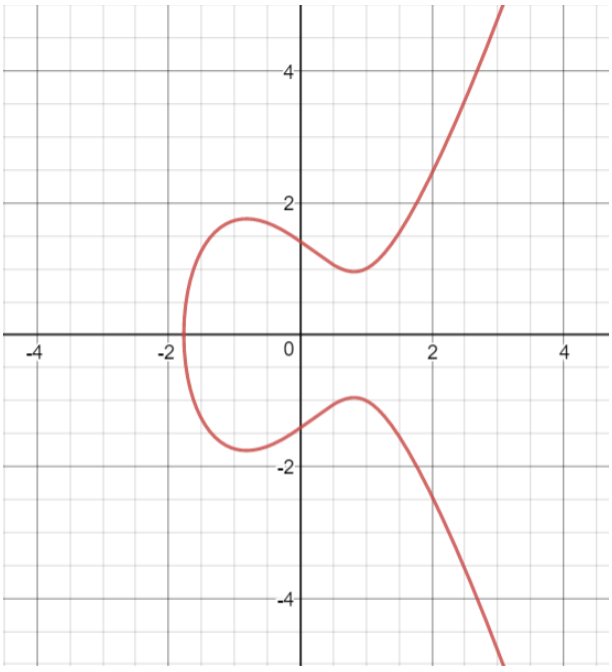


Figure 3.5: Elliptic curve with $a = -2$ and $b = 2$

For ECC, the elements x, y, a and b in equation 3.1 are restricted to a finite field. The points on the curve in terms of (x, y) coordinate hold properties called Point Addition and Point Multiplication. Some of the relevant concepts used in ECC are defined as follows.

Definition 3.6.2. Finite Field: A finite field is a set of $finite$ number of elements with operations like addition, multiplication, subtraction and division defined on them and holding commutative, associative, identity, and distributive properties. For example, the set of integers modulo p is a finite field, denoted by $\mathbb{Z}/p$ or F_p; p is a prime number.

Definition 3.6.3. Group: A group is a set of elements with a binary operation where two elements can be combined to form a third element such that it satisfies the properties of closure, associativity, identity and invertibility. If the group also holds the property of commutativity, then it is called an Abelian group. For exam-ple, the set of integers $\mathbb{Z}$ is an Abelian group but the set of natural numbers $\mathbb{N}$ is not, since it does not hold the commutative property.

Definition 3.6.4. Elliptic curve over finite field: An elliptic curve $\mathscr{E}$ over a finite field F_p of integers modulo a prime number p constitutes the set of points $\{0\} \cup \{(x, y) \in F_p \times F_p\}$ such that

$$y^2 \equiv x^3 + ax + b(mod\ p), \tag{3.2}$$

where, $a, b \in F_p$ and $4a^2 + 27b^2 \not\equiv 0(mod\ p)$. Point manipulations in $\mathscr{E}$ are defined using the operations of Point Addition and Point Multiplication.

Definition 3.6.5. Generator: The generator G is a point (base point) such that the multiples of G produces a subgroup of $\mathscr{E}$, $\mathscr{E}(G)$. If G generates a subgroup with co-factor 1, then every point in $\mathscr{E}$ can be represented as some multiple of G.

Definition 3.6.6. Point addition: It is the operation of adding two points on the elliptic curve, which yields a third point on the curve. For example, for two points P and Q on $\mathscr{E}$, the line drawn passing through these points intersects $\mathscr{E}$ at a third point R, and the three points are aligned irrespective of the order as shown in Figure 3.6. Algebraically, if $P = (x_P, y_P)$ and $Q = (x_Q, y_Q)$ are two distinct points, then the line passing through these two points has the slope

$$m = \frac{y_P - y_Q}{x_P - x_Q}$$

The line with slope m intersects $\mathscr{E}$ at a point $R = (x_R, y_R)$. The coordinates of R can be calculated as:

$$\begin{aligned} x_R &= m^2 - x_P - x_Q \\ y_R &= y_P + m(x_R - x_P) \end{aligned}$$

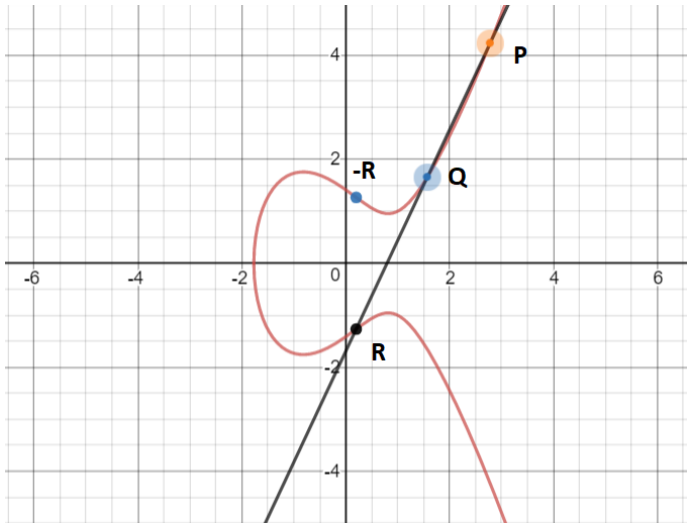


Figure 3.6: Elliptic curve with three aligned points

Since we are considering a finite field F_p, the equation of the line in F_p will be: $ax + by + c \equiv 0 \pmod{p}$ and accordingly, the equations to calculate the slope m and coordinates of point R will change as follows:

$$\begin{aligned} m &= \frac{y_P - y_Q}{x_P - x_Q} \, mod \, p \\ x_R &= (m^2 - x_P - x_Q) \, mod \, p \\ y_R &= [y_P + m(x_R - x_P)] \, mod \, p \end{aligned}$$

If P and Q are not distinct points ($P = Q$), then the slope will be calculated as:

$$m = \frac{3x_P^2 + a}{2y_P} \, mod \, p$$

Definition 3.6.7. Point multiplication: It is the operation of successively adding a point to itself along an elliptic curve (performing a series of point addition operations). If x is

scalar, point multiplication is denoted as xP where $P \in \mathscr{E}$, such that

$$xP = \underbrace{P + P + ... + P}_{\text{x times}}$$

ECC uses the idea of point multiplication to generate a one-way function since performing a point addition operation along the curve generates a third point, whose location does not reveal any information to trace back to the points that were added in the point addition operation. Elliptic curve based protocols take advantage of the computational intractability of the discrete logarithm problem that exists in elliptic Curves. The Elliptic Curve Discrete Log Problem (ECDLP) problem is defined as follows.

Definition 3.6.8. Elliptic Curve Discrete Log Problem: Suppose $\mathscr{E}$ is an elliptic curve over a finite field F_p and there exist points $P, Q \in \mathscr{E}(G)$. Given Q, as a multiple of P, the problem is to find $n \in \mathbb{Z}$ such that $Q = nP$.

There is no existence of a conventional algorithm to solve this problem in polynomial time yet. It is hypothesized that it is not feasible to find the discrete logarithm of a random elliptic curve element with respect to a publicly known point.

Chapter 4

Private Record Linkage using Key Rings

4.1 Introduction

Chapter 3 discusses the Federated query processing architecture, cryptographic protocols that will be used in data transformations and underlying aspects of the matching algorithm for private record linkage. Figure 3.2 gives an abstract view of the communication between the regional grid node components. Figure 4.1 depicts the workflow of the data and precomputed files between the two sites and the linkage agent in the regional grid node.

The sites that intend to perform the record linkage using the opposite site's encrypted data uses the publicly known bigram mapping H, which is in the form of a level-0 precomputation file, $L0$. Each site uses the mapping H to perform a level-1 precomputation and generate $L1$ files. Both sites exchange the respective $L1$ files and perform a level-2 precomputation to generate $L2$ files. Both sites also generate multiple keys and permutations, and use them during the precomputations, and to encode their data files. Each site sends their encoded data files and $L2$ files to the linkage agent. The linkage agent performs the record linkage and sends back the results (unique linkage identifiers pertaining to single entity) to each site. Details of these steps are presented next.

4.2 Proposed Methodology

4.2.1 Trivial Approach

Section 3.4 describes how to calculate the similarity score between attribute values of two records using Dice's coefficient. It requires finding the number of common bigrams

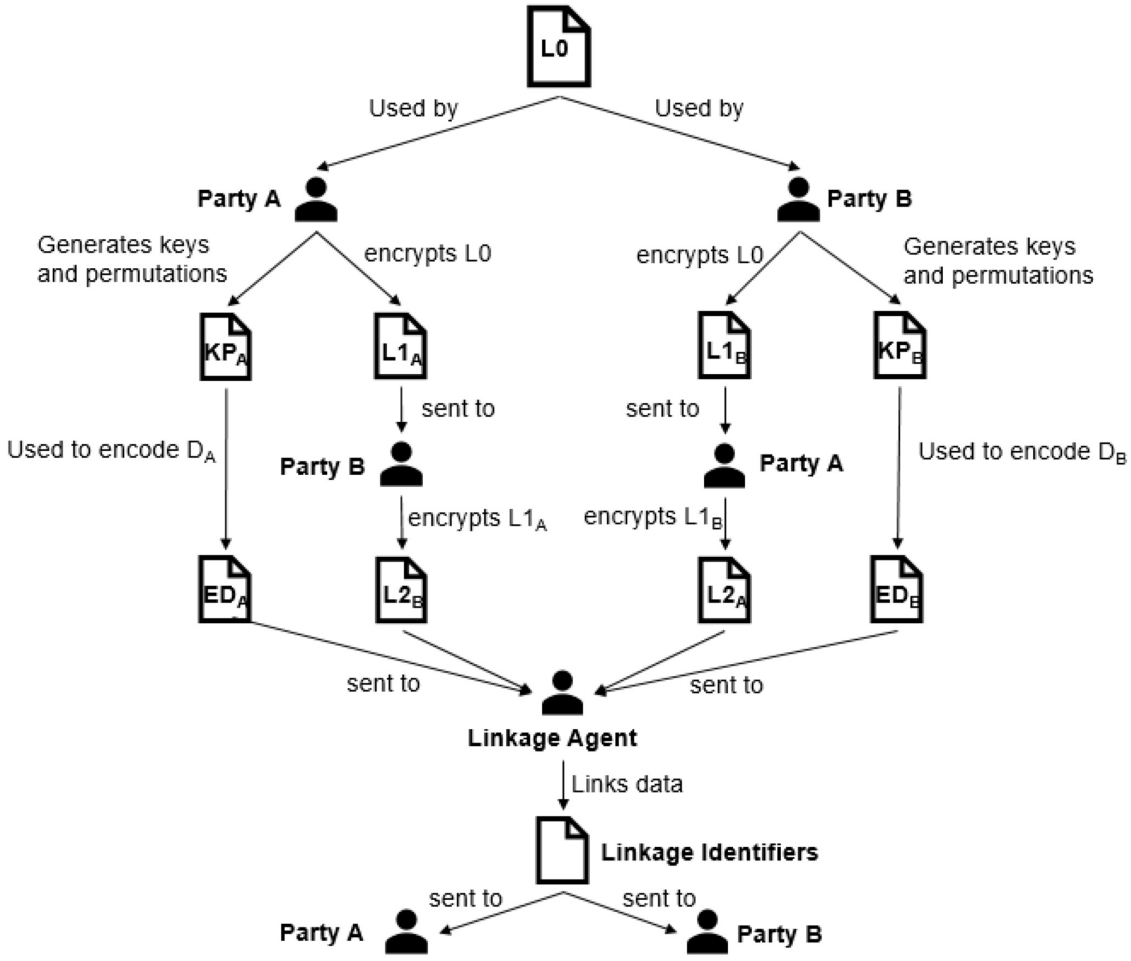


Figure 4.1: Private record linkage workflow

between the bigram sets of the attribute values. It is easier to perform such a set intersection in unencrypted data where one can simply divide the attribute values into bigram sets and compare the bigram strings to count the number of common bigrams. However, in a privacy preserving setting, the data is in the encrypted form. String comparisons become infeasible in such an environment. The following algorithm describes how to perform a set intersection with encrypted databases.

Let r_a and r_b be the two records from sites S_A and S_B respectively. Let α and β be bigram sets of the data value corresponding to an attribute in the records r_a and r_b respectively. Assume that the sites S_A and S_B have decided on a mapping H, such that all bigrams are represented by some point on an elliptic curve. Let $E_{KS_{site}}$ be the commutative encryption

function applied to the bigrams using key KS_{site}, $site \in \{A, B\}$. $|\alpha \cap \beta|$ can be calculated using the following algorithm.

1. For every pair of bigrams $(b_i, b_j) \in \alpha \times \beta$,
   (a) The sites S_A and S_B pick secret keys (randomly generated) KS_A and KS_B respectively, $KS_A, KS_B \in \mathbb{Z}$.
   (b) S_A computes encrypted value $e_A = E_{KS_A}(H(b_i))$ and S_B computes encrypted value $e_B = E_{KS_B}(H(b_j))$ using the mapping H and keys picked in step a. The sites then exchange the encoded values.
   (c) Both sites apply their respective keys to the encoded values that came from the opposite site. Thus, S_A computes $e'_A = E_{KS_A}(e_B)$ and S_B computes $e'_B = E_{KS_B}(e_A)$. The sites again exchange the new encoded values.
   (d) S_A and S_B conclude that b_i and b_j matches if $e'_A = e'_B$.
2. S_A and S_B calculate $|\alpha \cap \beta|$ as the number of matches found in step 1.

The correctness of the algorithm lies in the commutative properties of the transformation function $E_{KS_{site}}$. However, the need to perform cryptographic transformation for every comparison, often repeatedly on the same input, makes this an infeasible approach.

4.2.2 Linkage using Precomputations

We use 69 characters as the alphabet which would generate a total of 4761 possible bigrams. Since the number of bigrams are limited, it would be helpful to reduce the number of keys and precompute the values instead of computing new encrypted values during the process of linking.

To implement the described protocol efficiently, we restrict the domain of keys at each site to a smaller set and precompute lookup tables for the values encrypted using elliptic curve transformations. While performing the private set intersection over encoded records, a site picks keys from its corresponding key ring and then refers to the precomputed tables to encode the data instead of performing an encryption every single time. We use a commutative encryption function based on elliptic curve transformations instead of a keyed hash function.

4.2.3 Precomputing using a Key Ring

Figure 4.2 depicts the data exchanges between the sites at different levels of precomputation. As opposed to the Pohlig-Hellman approach, we replace the computations based on modular exponentiations with one-way transformations based on the infeasibility of the ECDLP. The transformations in the form of point additions and scalar multiplications optimize the precomputation time significantly. A probabilistic approach to record linkage with key rings begins with the independent selection of keys by the two sites. A key ring is a fixed set of keys, each key corresponding to a point on the elliptic curve. As discussed in Section 3.6.4, a point in $\mathscr{E}(G)$ can be represented as kG for some integer k. Each party decides the key ring size (number of keys) and accordingly performs a uniform random sampling to represent the keys in their respective key rings. Let KS_A and KS_B be the key rings for sites S_A and S_B respectively. Both sites decide on w secret permutations of the bigram set γ, one for each key, where γ is the set of all possible bigrams and w is the size of the respective key ring of each site. Each of the w permutations have a different shuffling of the bigram sequence. An element at index i in the set γ will be at index $\pi(i)$ in the shuffling of the sequence, where π denotes a generic permutation of the numbers $1,2,3,...,|\gamma|$.

Level-0 Precomputation

For all possible 4761 bigrams in the bigram set $\gamma = \{AA, AB, AC, ...\}$, the $L0$ file contains the encoded values for these bigrams (mappings between bigrams and EC points). Linkage agent generates a $L0$ file using the protocol described as follows.

For each bigram $b_i \in \gamma$, a new EC point P_i is generated using key $k_i \in \mathbb{N}$ (a positive random number) and generator G. Hence,

$$P_i = k_i \times G \tag{4.1}$$

Thus we have,
$P_1 = k_1 G$ for bigram b_1
$P_2 = k_2 G$ for bigram b_2
$P_3 = k_3 G$ for bigram b_3
and so on. Thus the $L0$ file comprises of

$$L0[i] = \langle b_i, P_i \rangle,$$

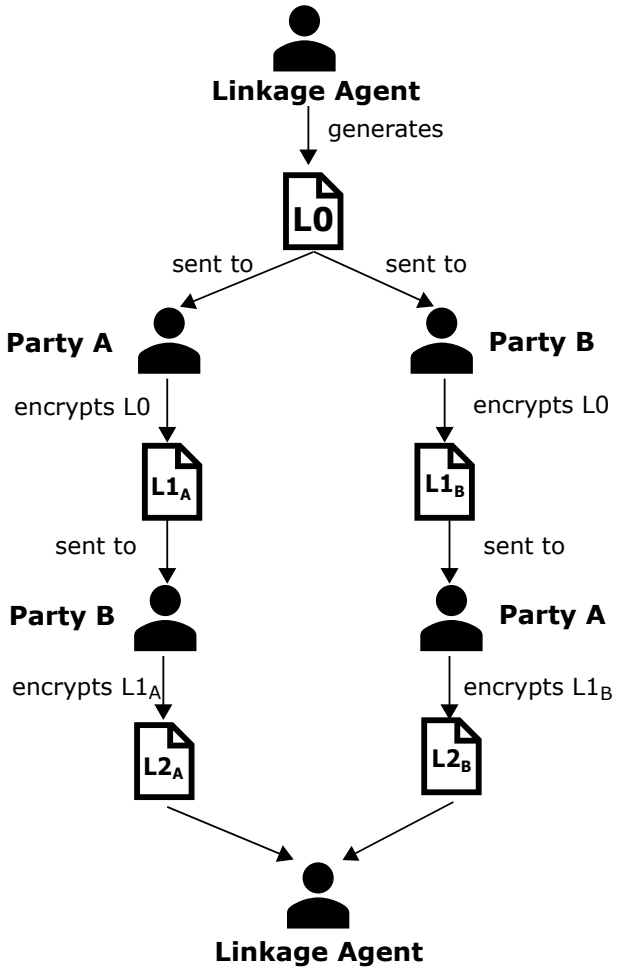


Figure 4.2: Generation of precomputation files

where, $b_i \in \gamma$ and $P_i \sim \mathscr{E}(G)$ where $\sim$ indicates uniform random sampling. The $L0$ file is made publicly available. $L0[i].b$ and $L0[i].P$ refers to a tuple's values.

Example: Consider there are three bigrams, "*ab*", "*bc*", "*cd*" which are transformed to EC-points using H to 10, 20, 30, respectively. Thus, $L0$ will contain these three points. $L0$ is downloaded by both sites S_A and S_B.

Level-1 Precomputation

Site S_A and site S_B download the $L0$ file and use it to perform the level-1 precomputation to generate $L1$ files. The level-1 precomputation is performed individually by each site. Level-1 precomputation is computation of the one-way transformations of the shuffled $L0$ set, using the keys in the respective key rings (w keys) of each site. Let $\mathsf{OWF}_K(m)$ denote

the one-way transformation function for message m using key K. $\mathsf{OWF}_K(m)$ is based on ECDLP and hence difficult to invert.

Let $\pi_A^1, \pi_A^2, ..., \pi_A^w$, and $\pi_B^1, \pi_B^2, ..., \pi_B^w$ denote the permutations generated by the sites. Site S_A will compute $L1_A$ messages as

$$L1_A = \{\mathsf{OWF}_{K_A^j}(b_{\pi(i)}) | i \in 1, ..., |\gamma|, K_A^j \in KS_A, \pi = \pi_A^j, j = 1, ..., |KS_A|\}$$

where,
$\mathsf{OWF}_{K_A^j}(b_{\pi(i)}) = (K_A^j \times P_{\pi(i)})$; encoded value of permuted bigram $b_{\pi(i)}$ using key $K_A^j \in KS_A$.
Site S_B will compute $L1_B$ messages:

$$L1_B = \{\mathsf{OWF}_{K_B^j}(b_{\pi(i)}) | i \in 1, ..., |\gamma|, K_B^j \in KS_B, \pi = \pi_B^j, j = 1, ..., |KS_B|\}$$

where,
$\mathsf{OWF}_{K_B^j}(b_{\pi(i)}) = (K_B^j \times P_{\pi(i)})$; encoded value of permuted bigram $b_{\pi(i)}$ using key $K_B^j \in KS_B$.

We will refer to a specific element in the sets $L1_A$ and $L1_B$ using a key index and a permuted index of the bigram. For example, $L1_A$ comprises of:

$$L1_A[p, \pi_A^p(L0[q].b)] = \mathsf{OWF}_{K_A^p}(L0[q].P)$$

where, $p = 1, ..., |KS_A|$ and $q = 1, ..., |\gamma|$. Similarly, $L1_B$ comprises of:

$$L1_B[p', \pi_A^{p'}(L0[q'].b)] = \mathsf{OWF}_{K_B^{p'}}(L0[q'].P)$$

where, $p' = 1, ..., |KS_B|$ and $q' = 1, ..., |\gamma|$.

Example: Let S_A have the key ring $KS_A = \{3, 5\}$ and S_B have the key ring $KS_B = \{7, 11\}$. Now, each site will use the $L0$ file and their respective key rings to generate the $L1$ files containing permuted messages as shown in Tables 4.1 and 4.2. The permutations used by site S_A for two keys in its key ring are:

$$\pi_A^1 : 1 \to 3,\ 2 \to 2,\ 3 \to 1$$
$$\pi_A^2 : 1 \to 2,\ 2 \to 3,\ 3 \to 1$$

where, $\pi_A^1 : 1 \rightarrow 3$ denotes that the bigram at index 1 is permuted to third position when first key is used. Similarly, the permutations used by site S_B for two keys in its key ring are:

$$\pi_B^1 : 1 \rightarrow 2,\ 2 \rightarrow 3,\ 3 \rightarrow 1$$
$$\pi_B^2 : 1 \rightarrow 1,\ 2 \rightarrow 3,\ 3 \rightarrow 2$$

Key-Message indices	Key	Bigram	OWF	L1 message
(1, 1)	3	10	30	(1, 1) : 90
(1, 2)	3	20	60	(1, 2) : 60
(1, 3)	3	30	90	(1, 3) : 30
(2, 1)	5	10	50	(2, 1) : 150
(2, 2)	5	20	100	(2, 2) : 50
(2, 3)	5	30	150	(2, 3) : 100

Table 4.1: Level-1 precomputation at site S_A

Key-Message indices	Key	Bigram	OWF	L1 message
(1, 1)	7	10	70	(1, 1) : 210
(1, 2)	7	20	140	(1, 2) : 70
(1, 3)	7	30	210	(1, 3) : 140
(2, 1)	11	10	110	(2, 1) : 110
(2, 2)	11	20	220	(2, 2) : 330
(2, 3)	11	30	330	(2, 3) : 220

Table 4.2: Level-1 precomputation at site S_B

Level-2 Precomputation

The sites S_A and S_B exchange their corresponding level-1 precomputation files $L1_A$ and $L1_B$ with each other. These files are then used for a level-2 precomputation at respective sites. Since the sites have each other's level-1 precomputation file, there lies a risk of backtracking to the original bigrams from the permuted values in case a site has knowledge of the permutation function that was used in level-1 precomputation by the opposite site. In absence of this knowledge, it is not possible to do so. Also, LA should not have direct access to raw $L1$ files. Hence, $L1$ files are encrypted and then exchanged by the two sites.

The level-2 precomputation is specific to a pair of sites. Hence, a site must perform a level-2 precomputation for every other site with which it seeks to perform the record

linkage. Level-2 precomputation at a site is the encoding of values obtained from opposite site's level-1 precomputation, using each key from its own key ring (without applying the permutations). Site S_A will compute $L2_A$ messages:

$$L2_A = \{\mathsf{OWF}_{K_A^j}(l_B) | l_B \in L1_B, K_A^j \in KS_A\}$$

Site S_B will compute $L2_B$ messages:

$$L2_B = \{\mathsf{OWF}_{K_B^j}(l_A) | l_A \in L1_A, K_B^j \in KS_B\}$$

A specific element in the sets $L2_A$ and $L2_B$ can be referred to, using a key index from site S_A, a key index from site S_B and a permuted bigram index. For example, $L2_A$ comprises of:

$$L2_A[p,q,r] = \mathsf{OWF}_{K_A^p}(L1_B[q,r])$$

where, $p = 1,...,|KS_A|$, $q = 1,...,|KS_B|$ and $r = 1,...,|\gamma|$. Similarly, $L2_B$ comprises of:

$$L2_B[q',p',r'] = \mathsf{OWF}_{K_B^{q'}}(L1_A[p',r'])$$

where, $q' = 1,...,|KS_B|$, $p' = 1,...,|KS_A|$ and $r' = 1,...,|\gamma|$.

Example: The $L2$ files at both sites will be as shown in Tables 4.3 and 4.4.

(LK, EK, PM) indices	Local key	L1 message	OWF	L2 message
(1, 1, 1)	3	(1, 1) : 210	630	(1, 1, 1) : 630
(1, 1, 2)	3	(1, 2) : 70	210	(1, 1, 2) : 210
(1, 1, 3)	3	(1, 3) : 140	420	(1, 1, 3) : 420
(1, 2, 1)	3	(2, 1) : 110	330	(1, 2, 1) : 330
(1, 2, 2)	3	(2, 2) : 330	990	(1, 2, 2) : 990
(1, 2, 3)	3	(2, 3) : 220	660	(1, 2, 3) : 660
(2, 1, 1)	5	(1, 1) : 210	1050	(2, 1, 1) : 1050
(2, 1, 2)	5	(1, 2) : 70	350	(2, 1, 2) : 350
(2, 1, 3)	5	(1, 3) : 140	700	(2, 1, 3) : 700
(2, 2, 1)	5	(2, 1) : 110	550	(2, 2, 1) : 550
(2, 2, 2)	5	(2, 2) : 330	1650	(2, 2, 2) : 1650
(2, 2, 3)	5	(2, 3) : 220	1100	(2, 2, 3) : 1100

Table 4.3: Level-2 precomputation at site S_A; LK = local key, EK = external key, PM = permuted message

(LK, EK, PM) indices	Local key	L1 message	OWF	L2 message
(1, 1, 1)	7	(1, 1) : 90	630	(1, 1, 1) : 630
(1, 1, 2)	7	(1, 2) : 60	420	(1, 1, 2) : 420
(1, 1, 3)	7	(1, 3) : 30	210	(1, 1, 3) : 210
(1, 2, 1)	7	(2, 1) : 150	1050	(1, 2, 1) : 1050
(1, 2, 2)	7	(2, 2) : 50	350	(1, 2, 2) : 350
(1, 2, 3)	7	(2, 3) : 100	700	(1, 2, 3) : 700
(2, 1, 1)	11	(1, 1) : 90	990	(2, 1, 1) : 990
(2, 1, 2)	11	(1, 2) : 60	660	(2, 1, 2) : 660
(2, 1, 3)	11	(1, 3) : 30	330	(2, 1, 3) : 330
(2, 2, 1)	11	(2, 1) : 150	1650	(2, 2, 1) : 1650
(2, 2, 2)	11	(2, 2) : 50	550	(2, 2, 2) : 550
(2, 2, 3)	11	(2, 3) : 100	1100	(2, 2, 3) : 1100

Table 4.4: Level-2 precomputation at site S_B; LK = local key, EK = external key, PM = permuted message

Verifying Commutative Property

Note that in level-1 precomputation,

$$\mathsf{OWF}_{K_1}(P) = (K_1 \times P)$$

where K = local key and P = bigram message in the form of EC point. Also, in level-2 precomputation with two keys K_1 (external key) and K_2 (local key),

$$\begin{aligned}
\mathsf{OWF}_{K_2}(\mathsf{OWF}_{K_1}(P)) &= \mathsf{OWF}_{K_2}(K_1 \times P) \\
&= K_2 \times (K_1 \times P) \\
&= K_1 \times (K_2 \times P) \\
&= \mathsf{OWF}_{K_1}(K_2 \times P) \\
&= \mathsf{OWF}_{K_1}(\mathsf{OWF}_{K_2}(P))
\end{aligned}$$

Hence, the order of computations does not matter and the commutative property is retained in the precomputation values of the bigrams. Due to the commutative properties of the transformations,

$$L2_A[p,q,r] = L2_B[q',p',r']$$

where,

$$\begin{aligned}
p &= p' \\
q &= q' \\
r &= \pi_B^q(b_i) \\
r' &= \pi_A^{p'}(b_i)
\end{aligned}$$

Example: Refer to Table 4.3. Consider $L2_A[p,q,r] = (1, 2, 3) : 660$. Then, $r = \pi_B^q(b_i)$ suggests that the $L2$ message 660 comes from $L1$ message of $L1_B$ where the original bigram was permuted with second key. It can be verified from Tables 4.4 and 4.2 that 660 is the encoded bigram value for the bigram "*bc*" represented as 20 in the $L0$ file. Also, it can be observed in Table 4.4 that $L2_B[q',p',r'] = (2, 1, 2) : 660$ is the equivalent representation of the bigram message "*bc*".

Hence, if one of the sites gets hold of a level-2 precomputation set of the other site, it can cross-reference the values in that set with its own level-2 precomputation set values, and reverse the permutations used in the level-1 precomputation set of the other site. For example, site S_A can figure out the $L1_B$ set values if it obtains $L2_B$ set. Hence, the sites do

not exchange the level-2 precomputation sets $L2_A$ and $L2_B$, but instead, they are sent to the linkage agent along with the encoded data files for the process of set intersection.

4.2.4 Data Set Encoding

The data at a site is encoded by the site before sending the data to the linkage agent. Data encoding is performed using the publicly known mapping which comes from the linkage agent in the form of level-0 precomputation, $L0$ file. A site picks a key $k_j \in KS_{site}$, where $site$ is either A or B and KS_{site} is the respective key ring. The bigram $b \in \beta$ is then encoded to a tuple $\langle j, \pi^j_{site}(b)\rangle$ of the form $\langle$key index, permuted bigram index$\rangle$. Thus for each bigram $b_i \in \alpha$, site S_A chooses $j \in \{1,...,|KS_A|\}$ and computes the encoded bigram value $\langle j, \pi^j_A(b_i)\rangle$ for bigram sets of attributes in each record. Similarly, for each bigram $b_i \in \alpha$, site S_B chooses $j \in \{1,...,|KS_B|\}$ and computes the encoded bigram value $\langle j, \pi^j_B(b_i)\rangle$ for bigram sets of attributes in each record. Encoded values of the bigram set of each attribute are shuffled and the encoded data sets from respective sites are sent to the linkage agent for further computation.

Example: Let there be a string "*abc*" in a record r_a of D_A for site S_A. Hence bigram set $\alpha = \{$"*ab*","*bc*"$\}$. Recall that $KS_A = \{3,5\}$. Thus, choosing a key and its respective permutation for the bigram would give the encoded set, for example, $\alpha' = \{\langle 1, 3\rangle, \langle 2, 3\rangle\}$ which suggests that "ab" is permuted to third position using first key and "bc" is permuted to third position using second key.

Frequency Smoothing

In the text of English language, certain bigrams appear with higher frequency than others. Figure 4.3 gives an idea about the distribution of bigrams, generated using the Google corpus for texts written in English. It can be seen that out of the 676 possible bigrams that can be generated from the alphabet character set of English, most of them barely contribute to the frequency of usage of bigrams. This gives rise to a potential privacy risk if the bigrams are encoded using single key since the distribution pattern will still hold for the encoded values which can be compared with the known distribution of the 676 bigrams. Using multiple keys randomly to encode different bigrams is also not a reliable solution, the same distribution will appear in each key. Frequency smoothing is a process used to maintain similar frequencies for the encoded values which helps to hide the distribution pattern of the encoded bigrams.

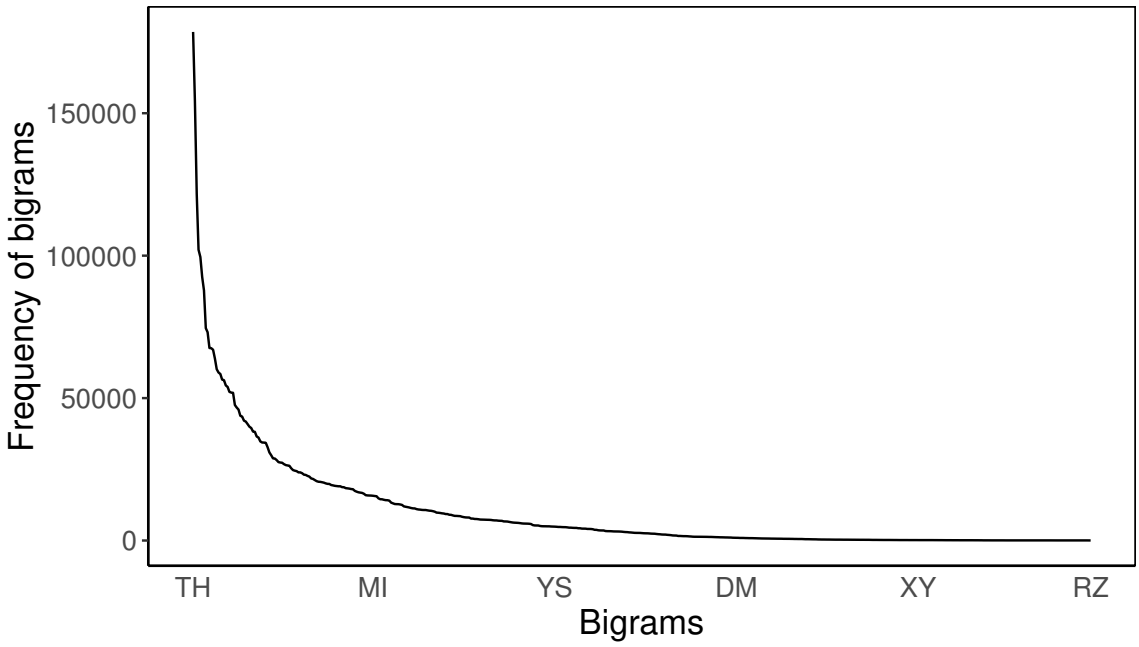


Figure 4.3: Frequency distribution of most common bigrams in English text

For the data encrypted without frequency smoothing, the frequency distribution holds and leaks considerable amount of information. The linkage agent can learn the frequency of different bigrams on a per key basis and trace back the private data. Frequency smoothing will ensure the distribution of high frequency bigrams over larger number of keys and lower frequency bigrams over smaller number of keys such that the encoded value of those bigrams appear uniformly in the encoded data set. The process of frequency smoothing is described as follows. Let $f_i(b)$ be the normalized frequency of a bigram $b \in \gamma$ for field index i. Let $max_{b\in\gamma} f_i(b)$ be the maximum frequency of a bigram in field i. Then, for a key ring K, a site can choose keys for encoding such that a key is used for

$$\delta = \frac{max_{b\in\gamma} f_i(b)}{|K|}$$

fraction of occurrence of a bigram. To achieve frequency smoothing for a bigram b, a key index is chosen uniformly from $[1, K_i]$ where $K_i = \lceil \frac{f_i(b)}{\delta} \rceil$. It can be noted that δ will be higher for higher frequency bigrams and hence they will be encoded with higher number of keys unlike lower frequency bigrams. This process of smoothing fails to hide bigrams whose frequency is less than δ but the focus here is to hide the distribution of higher

frequency bigrams. Figures 4.4 and 4.5 depict the effects of frequency smoothing over the bigrams.

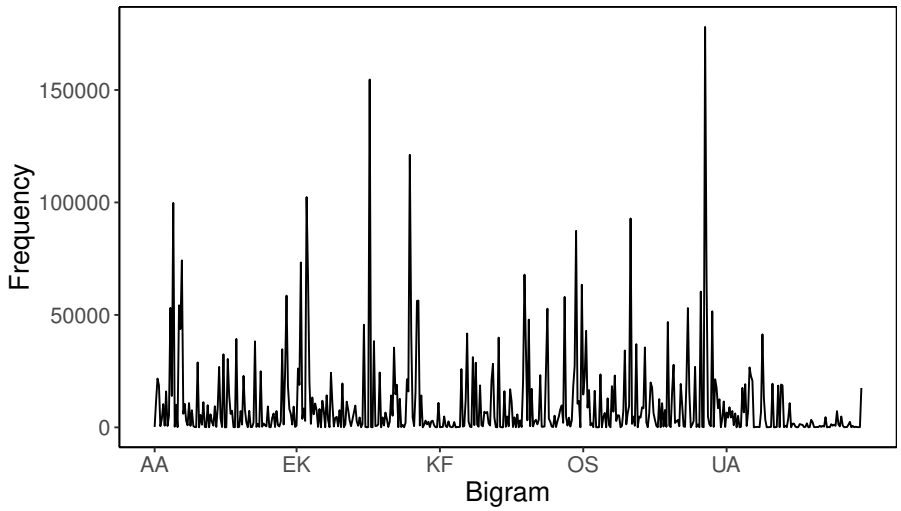


Figure 4.4: Before frequency smoothing

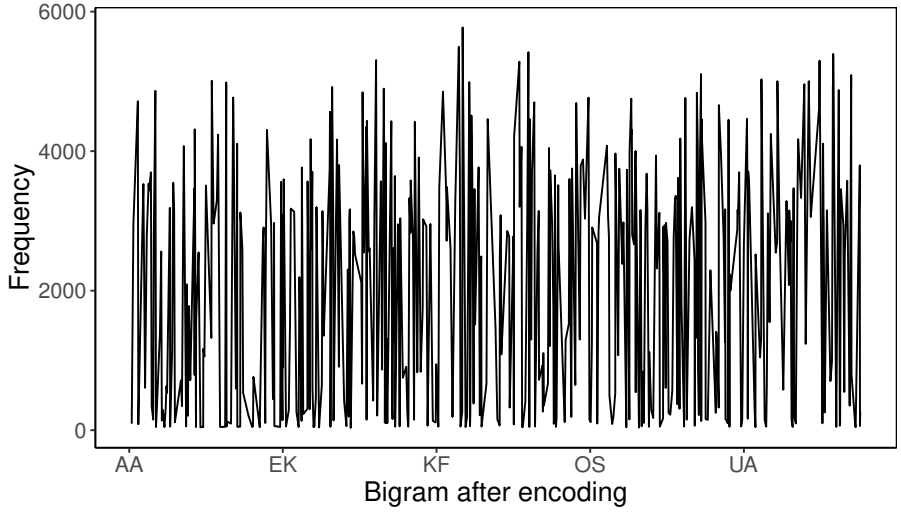


Figure 4.5: After frequency smoothing

Exposure Risk

Although the process of frequency smoothing hides the bigram distribution pattern, the bigram frequency information may leak through the number of key indices used for a particular bigram. Figure 4.6 shows the keys used while encoding bigrams and verifies the

concern of information leak. We assume that the adversary is aware of the key ring size and the field-wise bigram distribution of the client's data set. The risk exposure is calculated on the basis of the probability of identifying a bigram from its encoded values and thus, getting exposed to identifying a fraction of the bigrams.

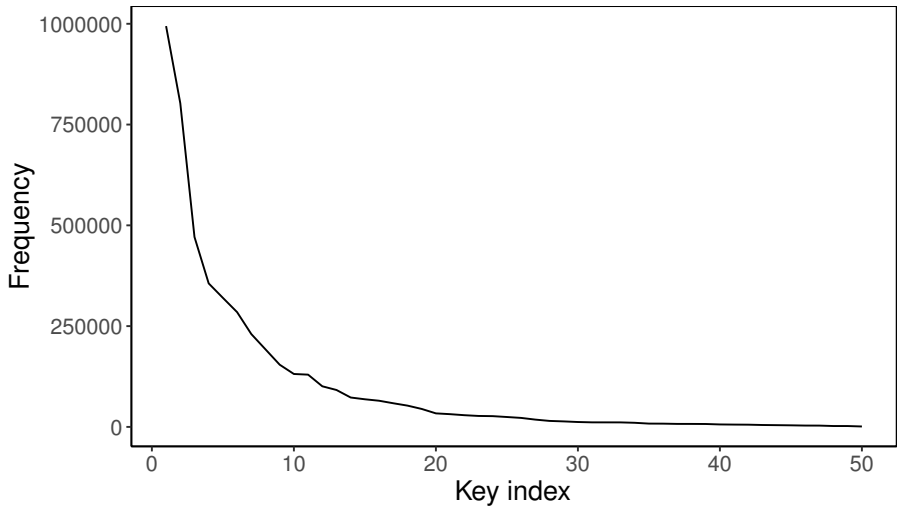


Figure 4.6: Key usage in encoding bigrams

Let $f(b)$ be the frequency of the bigram b in some field. $f(b)$ is known to the adversary. The prior exposure of the bigram is measured as the reciprocal of the number of bigrams occurring with the same frequency.

$$Pr_{pre}(b_i) = \begin{cases} \frac{1}{|\{b_j | f(b_j) = f(b_i)\}|} & , f(b_i) \neq 0 \\ 0 & , f(b_i) = 0 \end{cases} \tag{4.2}$$

Hence, it is easy to identify the bigrams if all the bigrams occur with unique frequencies even after encoding them. It is also obvious that using only one key index results in the same exposure before and after applying frequency smoothing to the bigrams encoded with that key. The rest of the bigrams will occur with frequency δ in each key. For a key index u, let e_{iu} denote the encoding pair for bigram b_i. The uncertainty faced by the adversary depends on how many other bigrams could be using key index u. Hence, we define the

posterior exposure of bigram b_i when using key index u as

$$Pr(e_{iu} \rightarrow b_i) = \begin{cases} \frac{1}{|\{b_j | K_j \geq K_i\}|} & , K_i > 1 \\ Pr(b_i) & , K_i = 1 \end{cases}. \tag{4.3}$$

Since b_i may be encoded using K_i possible key indices chosen uniformly at random, and each key has a different permutation, correlating encodings of the same bigram across different keys is not possible. Therefore, we compute the expected posterior exposure of b_i as

$$Pr_{post}(b_i) = \sum_{j=1}^{K_i} \frac{Pr(e_{ij} \rightarrow b_i)}{K_i}. \tag{4.4}$$

Given a string in a field made of bigrams $b_{i_1}, b_{i_2}, ..., b_{i_l}$ arranged in decreasing order of their exposure probability, we compute the exposure risk associated with identifying the m ($\leq l$) highest exposed bigrams as

$$Exposure_m(s) = \prod_{j=1,...,m} Pr_{pre/post}(b_{i_j}). \tag{4.5}$$

4.2.5 Linking Data Sets

Linkage map : Using Precomputed Tables

Both sites can send their respective $L2$ files to the Linkage Agent (LA) as followed by the Pohlig-Hellman approach. But, comparison of the transformations in the $L2$ files, which are EC points (long memory bytes) in the EC-based approach, becomes relatively expensive than performing a lookup to check for bigram similarity. Hence, we perform an optimization where the $L2$ files are sorted based on the EC points in the files. The OpenSSL library provides a serialized byte representation of the EC points which is used to sort the $L2$ files.

The probability for two $L2$ messages to represent the same EC point in a large elliptic curve is negligible unless the two keys and the represented bigrams in the nested one-way transformation function OWF are the same. Let $L2_A[p,q,r]_i$ and $L2_B[q',p',r']_i$ denote the i^{th} message in the respective files after independently sorting them. It holds they are equal. Also, $p' = p$, $q' = q$, and, the messages represent the same bigram. Thus, both the sites send indexing triplets with sorted EC points to the LA. Let T be the file representing the

set of indexing triplets. Then,

$$T_A[i] = (p,q,r)_i \text{ and } T_B[i] = (q',p',r')_i$$

Example: Table 4.5 shows the sorted $L2$ indexing triplets received at the linkage agent's side. The processing of $L2$ files to T files at the sites avoids having to send huge $L2$ files to

$L2_A$ message	$L2_B$ message
(1, 1, 2)	(1, 1, 3)
(1, 2, 1)	(2, 1, 3)
(2, 1, 2)	(1, 2, 2)
(1, 1, 3)	(1, 1, 2)
(2, 2, 1)	(2, 2, 2)
(1, 1, 1)	(1, 1, 1)
(1, 2, 3)	(2, 1, 2)
(2, 1, 3)	(1, 2, 3)
(1, 2, 2)	(2, 1, 1)
(2, 1, 1)	(1, 2, 1)
(2, 2, 3)	(2, 2, 3)
(2, 2, 2)	(2, 2, 1)

Table 4.5: Indices of sorted $L2$ messages from both sites

the LA.

Linking

Unlike the algorithm discussed in Section 4.2.1, the linkage agent can now use the encoded data files and indexing triplet files to determine the number of common bigrams for corresponding attributes, and compute the intersection size of the bigram sets to be used in the matching algorithm (Section 3.5). The modified approach is as follows:

1. As discussed in Section 4.2.4, each bigram set corresponding to an attribute's value is encoded before being sent to the linkage agent. For a given attribute, site S_A has set α' of encoded bigram tuples (p_A, r_A) and site S_B has set β' of encoded bigram tuples (p_B, r_B).
2. Let T_A and T_B be the indexing triplets received from the sites. The private record linkage begins with building a lookup table (linkage map) T by merging T_A and T_B.
   - For entries $T_A[i] = (p,q,r)$ and $T_B[i] = (q',p',r')$ ($T_A[i]$ and $T_B[i]$ represent the same bigram; $p = p'$ and $q = q'$), an entry $T[p,q,r] = r'$ is inserted in T.

3. For each pair $(< p_A, r_A >, < p_B, r_B >) \in \alpha' \times \beta'$, if $T(p_A, p_B, r_B) = r_A$, it is concluded that a match exist since the pair represent the same bigram.
4. $|\alpha' \cap \beta'|$ = number of matches found in step 5.

Example: Table 4.6 depicts the linkage map built by the linkage agent using the indexing triplets received from the sites. Consider the case where $T[p, q, r] = (1, 2, 3)$ in Table 4.6 and recall the respective permutations used by both sites. If S_A uses key 1 and permuted bigram index 2, and S_B uses key 2 and permuted bigram index 3, then they both are referring to bigram "*bc*".

$T[p,q,r]$	r'
(1, 1, 2)	3
(1, 2, 1)	3
(2, 1, 2)	2
(1, 1, 3)	2
(2, 2, 1)	2
(1, 1, 1)	1
(1, 2, 3)	2
(2, 1, 3)	3
(1, 2, 2)	1
(2, 1, 1)	1
(2, 2, 3)	3
(2, 2, 2)	1

Table 4.6: Linkage map built using the indexing triplets

Hash Maps

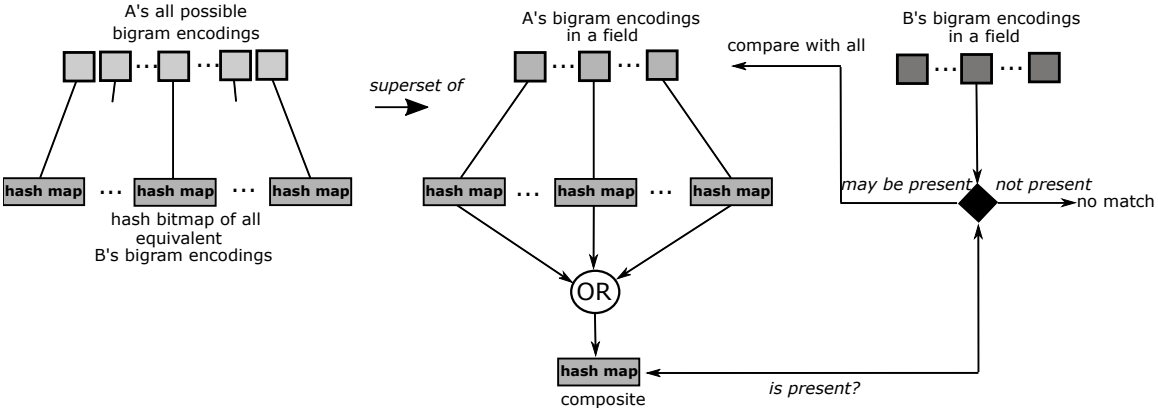


Figure 4.7: Hash map generation and usage during record linkage

The process to find similarity between record pairs of the two sites involves a greedy matching scheme and hence the algorithm is quadratic in nature. Considering the possible data values and data inconsistencies in the data sets, it can be inferred that during the encoded bigram set comparison for two attribute values, most of the bigrams will not result into a match. Even if there is a case where the two strings are exactly the same (with length l), the number of bigram matches would be at most $(l-1)$, whereas, the number of comparisons made would be l^2. Hence, comparing bigrams with a brute force approach is undesirable. To reduce the computational cost here, we have used hash maps which are described as follows.

Figure 4.7 depicts the usage of hash map in the linkage process. Let $EB_{\alpha'}$ and $EB_{\beta'}$ represent all possible bigram encodings used by site S_A and site S_B respectively. Let there be a hash function $Hash$ that outputs integers from $[1,h]$. We create a bitmap $H(p,r)$ of size h for each $(p,r) \in EB_{\alpha'}$, such that bit t is set (= 1) if there exist any $(q',r') \in EB_{\beta'}$ with $Hash((q',r')) = t$ and $T[p,q',r'] = r$. Thus, we create a bitmap for each bigram encoding from site S_A. For each bigram encoding from site S_B that represent the same bigram as that of site S_A, a bit is set in the bitmaps. These bitmaps are referred to as *Hash maps* and are created while building the linkage map.

Consider the case where a record r_A from site S_A and r_B from site S_B is being compared for similarity. r_A comprises of f number of $\alpha^{'}$ bigram encoding sets (f = number of fields). Thus, $r_A = \{\alpha_1^{'}, \alpha_2^{'}, ..., \alpha_f^{'}\}$. We create a bitmap $\mathscr{H}_i$ corresponding to each $\alpha_i^{'}$ by merging the hash maps of each bigram encodings in $\alpha_i^{'}$ (performing a bitwise-or operation). Thus, $\mathscr{H}_i$ can be represented as a composite hash map as follows:

$$\mathscr{H}_i = \vee_{(p,r) \in \alpha_i^{'}} H_{(p,r)}$$

Thus, for all possible bigram encodings in set $\beta^{'}$ of a field i in a r_B, $\mathscr{H}_i$ represents a composite hash map that can match some bigram encoding in $\alpha^{'}$ in r_A. While computing $|\alpha^{'} \cap \beta^{'}|$ in the process of linking for a field i, each bigram encoding (q',r') from site S_B is being compared to an encoding from site S_A only if $\mathscr{H}_i[Hash((q',r'))] = 1$ (bit t is set to 1 using S_B's bigram). If it is not 1, it can be concluded that no bigram in α (S_A's bigram set) can match any bigram in β (S_B's bigram set). We can hence infer that the false negative rate is zero. Due to existence of collisions in the hash function, $\mathscr{H}_i[Hash((q',r'))] = 1$ means

that there *maybe* a match. The true positive rate (if $\mathscr{H}_i[Hash((q',r'))] = 1$, then a match is indeed present) can be increased by decreasing the number of collisions in the hash map, henceforth, increasing the size of the hash map.

Hash Map Size

The hash map size (h bits) needs to be a balance between memory demand and probability of collisions. Large value of h will require more memory and small value of h will generate more collisions and hence resulting in more comparisons. We create $|KS_A| \times |\beta|$ hash maps and in each hash map, at most $|KS_B|$ bits are set. Some of these will be combined to form a composite hash map.

If q is the average number of bigrams in an attribute value, then q hash maps of h bits will form a composite hash map of h bits which is queried at q locations. Collisions at any other locations are not significant. It can be concluded that the size of the hash map should be sufficient enough such that q hash maps can avoid at least q collisions. Assuming the worst case where at most $q(|KS_B|-1)$ locations may already have been set by encodings that are not queried, the first of the q queried locations have no collision with probability $\frac{h-q(|KS_B|-1)}{h}$, the second with probability $\frac{h-q(|KS_B|-1)-1}{h}$, and so on. Hence, the probability that multiple encodings does not set each of the q queried locations is calculated as:

$$Pr(\text{no collision} \geq q) \geq \prod_{i=0}^{q-1} \frac{h-q(|KS_B|-1)-i}{h}. \tag{4.6}$$

We set the hash map size h as a multiple of $|KS_B|$, i.e $h = x|KS_B|$ for some positive integer x, and determine the smallest x such that

$$\prod_{i=0}^{q-1} \frac{x|KS_B|-q(|KS_B|-1)-i}{x|KS_B|} \geq p \tag{4.7}$$

for a desired true positive rate p. The value of x is found by increasing its value until the constraint is satisfied.

4.3 Implementation Details

Figure 4.1 represents the schematics of the process of privacy-preserving record linkage. This section gives an insight into the implementation details that is used to execute the

proposed framework. Figure 4.8 depicts different components and inputs-outputs involved in the pre-computing part of the method. The four components involved are :

1. Data encoder: It takes in a raw data file, a data frequency file and a keyperm (keys and permutations of bigrams) file. The raw data file is encoded by the data encoder using the frequency and the keyperm file.
2. Frequency extractor: It takes in a raw data file and generates a frequency file.
3. $L0$ encoder: It maps all possible bigrams to points on elliptic curve and generates a $L0$ file.
4. $L1$ encoder: It takes in a frequency file, a $L0$ file and a parameter N specifying the key ring size. The process outputs a file with generated keys and permutations (keyperm file), and a $L1$ file.
5. $L2$ encoder: It takes in a keyperm file and a $L1$ file to generate a $L2$ file.

4.3.1 File Structures

This section gives an idea about the format and structures of the various input and output files described in the framework.

Raw Data File

The raw data file contains the records of respective sites in clear-text form (not encrypted). Before the process of record linkage, these data files are expected to be in a standardized CSV format. The format of the file is as follows:

- First row contains the name of the attributes.
- Records (each row) are separated by the newline character.
- Data values of each field are separated by commas.

Frequency File

The frequency file is generated from the frequency extractor component of the framework. It contains the frequency of each bigram corresponding to each attribute. The format of the file is as follows:

- First row contains the string “bigram”, followed by name of the attributes, all separated by commas.

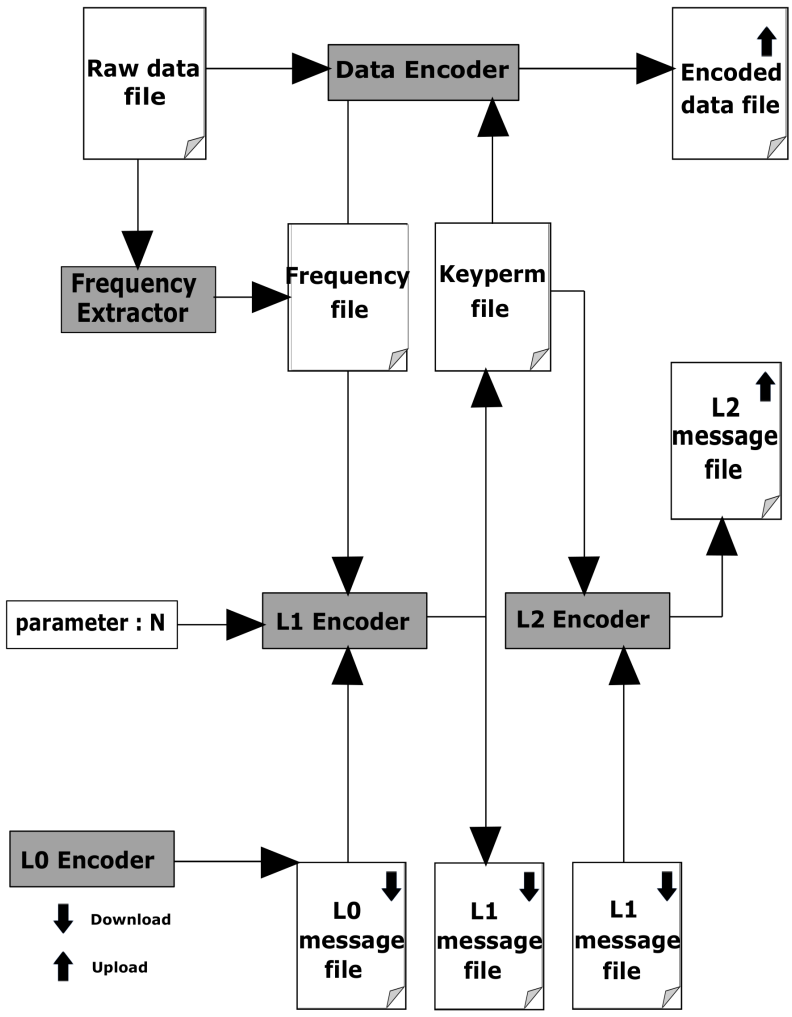


Figure 4.8: Component dependency for privacy preserving method

- Each row contains a bigram, followed by the frequencies of the corresponding bigram in each of the fields. Values are separated by commas.

Keyperm File

The keyperm file is generated by the $L1$ encoder. It consists of the permuted values of the bigram messages for respective keys. Detailed structure of the Keyperm file is depicted

in Figure 4.9. $Perm(x)$ in the keyperm file structure indicates permutation of x^{th} bigram; $x \in [0, B)$.

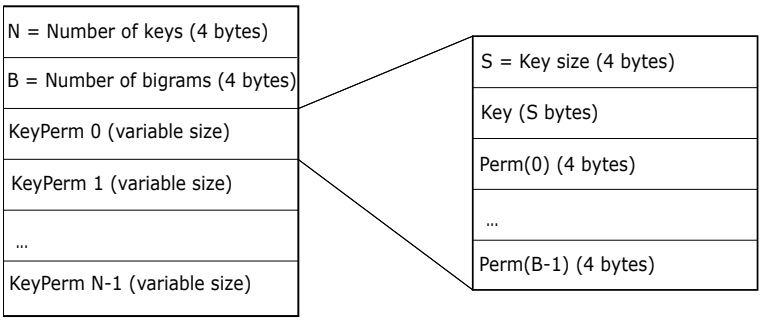


Figure 4.9: Keyperm file structure

L Message Files

The $L0$, $L1$, $L2$ message files are precomputed using the ECC approach described in Section 4.2.3 by the respective encoders. $L0$ file comprises of the numeric representation of bigrams (bigrams mapped to elliptic curve points). $L1$ file comprises of the $\langle$ local key index, encoded message index$\rangle$ tuples (built using $L0$ data). Recall that $L1$ files are exchanged between the parties to generate $L2$ files. Hence the local key index in $L1$ file acts as the external key index in $L2$ file. Detailed structure of the files are depicted in Figures 4.10 and 4.11.

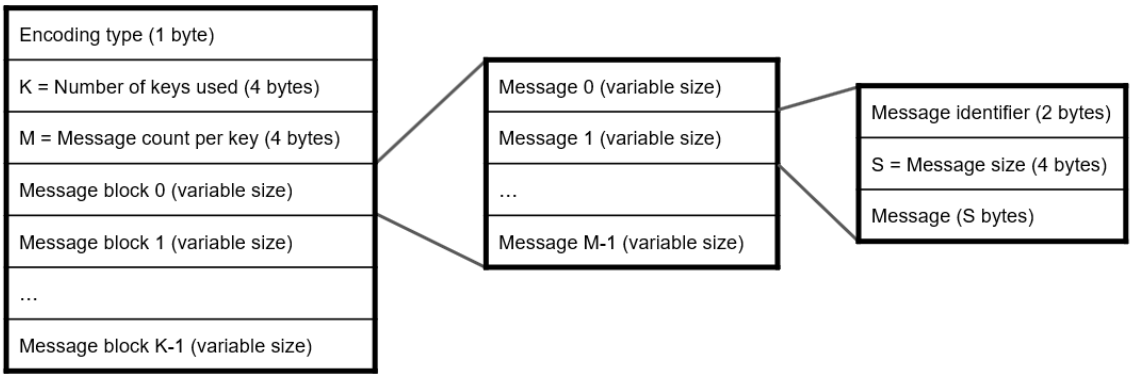


Figure 4.10: L0 - L1 file structure; Encoding type is 0 for L0 and 1 for L1; Message identifier is ASCII representation of bigram in L0 and 0x0000 in L1

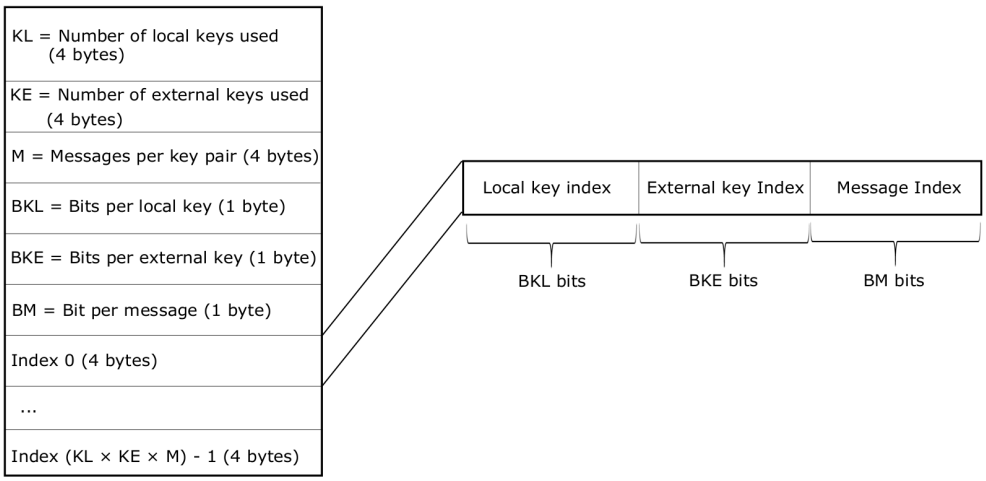


Figure 4.11: L2 file structure

Encoded Data File

The encoded data file is generated by the data encoder. It consists of the permuted bigram messages appended with the respective key used to apply permutations in the data encoder. Bigram encodings are represented in the form of a bit string of length $\lceil log_2(key\,ring\,size) \rceil + 13\,(2^{13} > |\beta|)$ rounded to the nearest multiple of 4.

4.3.2 Environment

We used $C++$ to implement the discussed privacy-preserving record linkage algorithms. It is assumed that the raw data sets at the respective sites are in a standardized form as described in Section 4.3.1. The implementation is divided into three multi-threaded programs: Precomputing L1 and L2 files, Data set encoding and Linking. We use the curve $secp521r1$ [26] from the OpenSSL library to perform the EC operations. All levels of precomputation and key generation functions use this curve. We use RAND_*bytes* function from the OpenSSL library to perform random number generation and the Knuth shuffle algorithm [38] to generate permutations in the precomputation code.

The program for data set encoding reads the data files and generates encoded versions with frequency smoothing. The linkage program uses the 128-bit $Mur-murHash3$ [37] hash function for the computations related to the hash maps. The programs to precompute

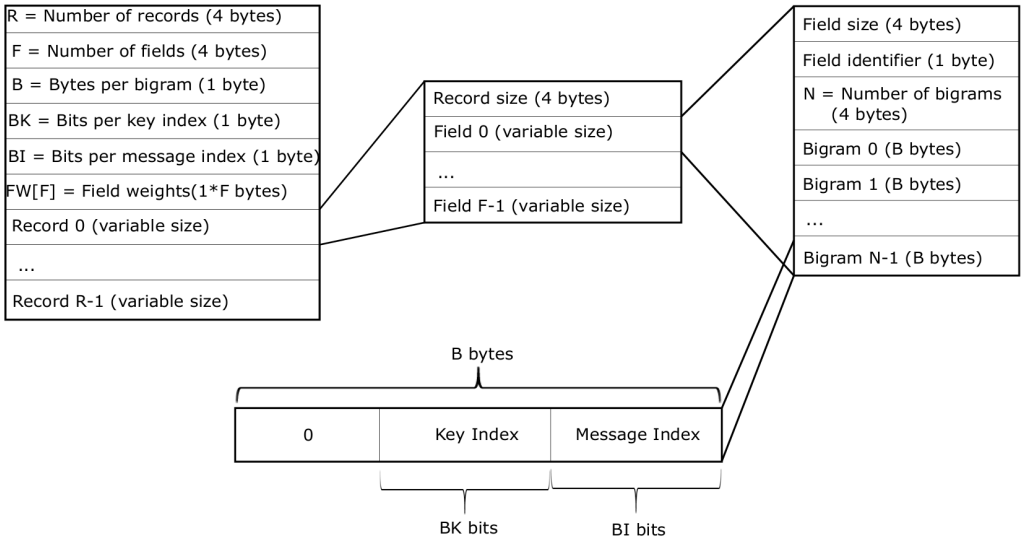


Figure 4.12: Encoded data file structure

the $L1/L2$ files and data set encoding are client side and the linking code is server side. The networking phase between the client and the server to transfer data has not been implemented yet. All the programs have been compiled with the $-Ofast$ and $-march = native$ flags in g++ version 5.4 using *pthreads* threading library.

Chapter 5

Results

The record linkage framework using the proposed elliptic curve cryptography protocol has shown encouraging results and has improved the linkage time by a significant amount. We perform various parametric experiments on synthetic and non-synthetic data sets to correlate this claim.

5.1 Data Sets

To perform various experiments, we used subsets of data from the North Carolina Voters Registration (NCVR) database. NCVR has more than 7 million records containing demographics data of individuals, example names, mailing address, phone number, gender, age, etc. [20]. We create the data subsets by randomly sampling records with five attributes - first name, last name, street address, city and zip code and using various sizes for the subsets. We also synthesize the sampled data subsets with two parameters: %overlap (records that overlap/match in two data sets) and %error (records undergoing synthetic error insertion). To gauge the linkage execution time for parametric studies, we consider the subset with 25% overlap as the default and to demonstrate linkage accuracy, we use sampled subsets with 25% and 5% data overlap. We simulate the data entry errors by introducing errors like character insertion/deletion, missing attribute, character substitution, adjacent character transposition and data entry in wrong field. The errors were introduced with varying probabilities to simulate the three categories of data being either relatively clean (5% error), moderately clean (30% error) or dirty (50% error). For our experiments, we use the

moderately clean data sets as the default. The encoded data files of the data sets vary from 200KB to 230MB in size.

We also built a synthetic data generator and encoder which uses a given frequency distribution of bigrams to generate synthetic data sets of a specified size (size based on number of records, number of fields, average number of bigrams in each field). To generate synthetic data, bigrams are randomly picked from the total possible bigrams of the given character set based on the probabilities of their occurrence in the standard frequency distribution of bigrams in English text. To uniformly pick bigrams for the given distribution, we used the algorithm based on converting non-uniform probabilities of the bigram occurrences to binary distributions [39].

5.2 Precomputation

Figure 5.1 depicts the execution time for computing level-1 ($L1$) and level-2 ($L2$) precomputation files for different key rings (1, 15, 25, 30, 50 keys) for 4761 bigrams (69×69) in the $L0$ file. The size of $L1$ files vary from 0.6-33 MB, and between 10-46 MB for the indexing triplets. It can be observed that precomputing $L1$ file takes significantly less time for varying key ring sizes against precomputing $L2$ file which takes more time as the size of key ring increases, since it has to perform the amount of work needed to generate $L1$ file once for each key in the key ring. The graph shows a quadratic increase in the execution time with increasing key ring size. Precomputation using modular exponentiations take more than two hours for a key ring of size 50.

5.3 Linkage Execution Time

To measure the linkage execution time of NCVR data subsets of different size, we run the linkage exercise on four standalone machines and four Amazon AWS EC2 compute optimized instances. Table 5.1 describes the machine configurations differentiated on the basis of processor type and speed, number of virtual CPUs, and memory.

Table 5.2 shows the linkage execution times for different machines with different sizes of the NCVR data subsets. It can be observed that with a fair amount of virtual CPUs (16 to 20), a quadratic linkage of $10^5 \times 10^5$ record pairs is executed within 8 minutes. With lesser number of virtual CPUs, the linkage is executed in about 15 minutes. Also, multi-threading proves to be beneficial in bigger data sets only. It degrades the linkage

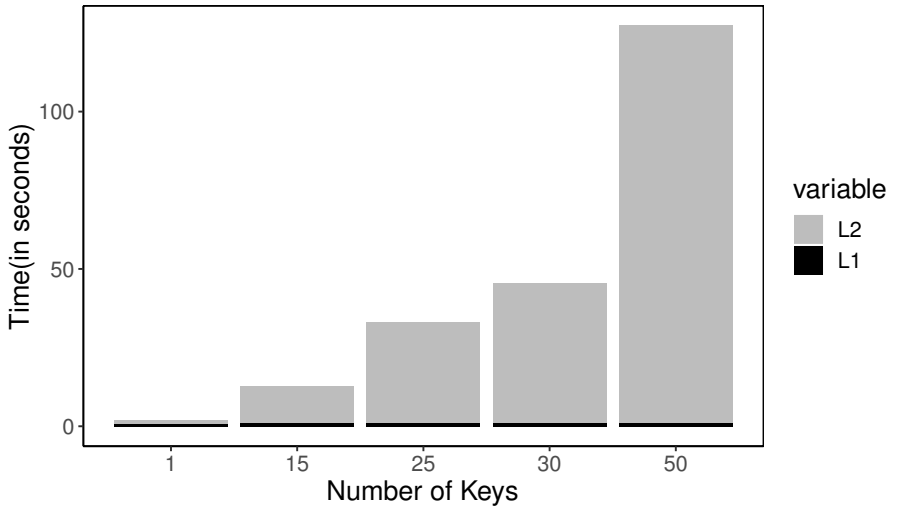


Figure 5.1: Precomputation time for different number of keys

Name	Type	CPU	vCPUs	Memory
S1	Apple Mac Pro	Intel Xeon E5-1620v2 3.7GHz	8	16GB
S2	Dell Precision T5810	Intel Xeon E5-1620v4 3.5GHz	8	8GB
S3	Dell Precision T5820	Intel Xeon W-2155 3.3GHz	20	16GB
S4	Dell PowerEdge R731	Intel Xeon E5-2695v4 2.1GHz	36	128GB
C1	AWS EC2 c5.2xlarge	Intel Xeon Platinum 8124M 3.0 GHz	8	16GB
C2	AWS EC2 c5.4xlarge		16	32GB
C3	AWS EC2 c5.9xlarge		36	72GB
C4	AWS EC2 c5.18xlarge		72	144GB

Table 5.1: Configurations of machines used to benchmark execution times

performance in smaller data sets. The EC based approach is feasible on data sets of the order of millions of records, but with increasing size, additional treatment to the data might be needed to make the linkage time acceptable.

We performed different parametric studies to measure the impacts on the linkage execution time, which are described in the further sections.

5.3.1 Hash Map Efficiency

Figure 5.2 depicts the linkage execution time for different hash map sizes. The p-value (desired true positive rate) varies from 0.1-0.9. The tests are performed on a synthetic data set with 100,000 records consisting of 10 fields, with 10 bigrams per data value, encoded with 50 keys and executed with 8 threads. p-value is the true positive rate in a composite

	Number of records in a data set						
Machine	1000	5000	10000	50000	100000	500000	1000000
S1	0.39 s	3.71 s	12.95 s	5.11 m	19.91 m	7.33 h	28.37 h
S2	0.38 s	4.28 s	17.18 s	5.18 m	16.78 m	6.07 h	23.93 h
S3	0.30 s	3.82 s	6.73 s	1.51 m	5.86 m	2.11 h	7.9 h
S4	0.49 s	2.77 s	7.11 s	1.45 m	5.75 m	2.2 h	8.1 h
C1	0.32 s	3.34 s	10.84 s	3.72 m	14.49 m	5.0 h	19.75 h
C2	0.27 s	2.32 s	7.27 s	2.0 m	7.74 m	2.74 h	10.12 h
C3	0.27 s	1.68 s	4.95 s	1.07 m	3.76 m	1.32 h	4.86 h
C4	0.49 s	2.57 s	5.79 s	0.91 m	2.75 m	0.77 h	3.02 h

Table 5.2: Linkage time in different machine configurations; Data sets have 25% overlap and 30% of the records in one data set has errors (s: seconds, m: minutes, h: hours)

hash map. It can be seen that smaller the p-value, higher the execution time since more collisions occur in such a case, thereby resulting in more number of bigram comparisons. With a higher p-value, lesser collisions occur in the hash maps and more number of comparisons are avoided.

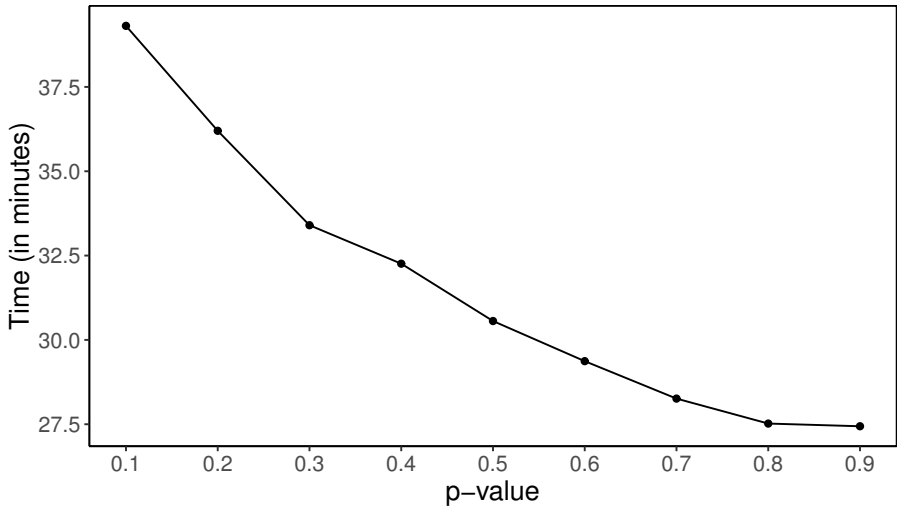


Figure 5.2: Linkage time for different hash map size

We also collect the following statistics to study the improvement produced by using hash maps in a linkage exercise.

- Number of bigram comparisons performed in absence of hash maps
- Number of bigram comparisons performed in presence of hash maps
- Number of times a query on a composite hash map returned a positive answer

- Number of times a positive query result actually finds a match

The results obtained tells us the percentage comparisons avoided by incorporating hash maps in the process, and the observed true positive rate of the queries. It can be observed in Figure 5.3 that using hash maps avoids approximately 70% to 90% comparisons. Also, the observed true positive rate increases with an increase in hash map size. The time reduction varies from about 20% to 50%. Using larger hash maps gives better optimizations but increases the memory requirements. A hash map with p-value 0.5 to 0.7 gives fairly improved results, and requires under 1 GB of memory for 50 keys.

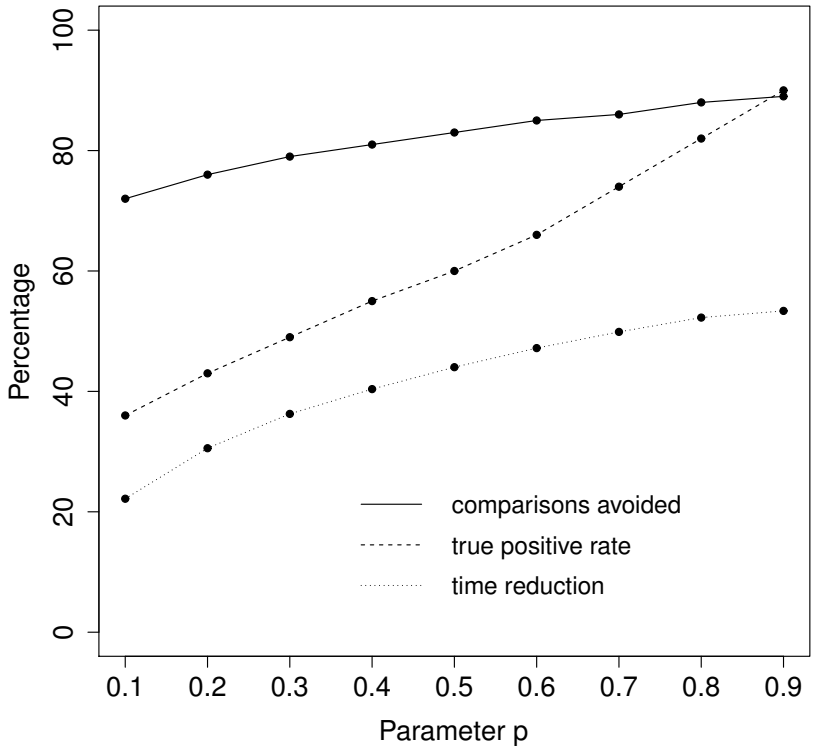


Figure 5.3: Hash map efficiency in avoiding bigram comparisons and reducing execution time

5.3.2 Different Number of Bigrams

Figure 5.4 depicts the linkage execution time for varying average number of bigrams in a field. The number of bigrams vary from 5-25. The tests are performed on a Synthetic data set with 100,000 records consisting of 10 fields, p-value 0.7, encoded with 50 keys, and executed with 8 threads. It can be observed that higher the number of bigrams, linkage

time increases considerably since number of comparisons to be made increases. Note that bigrams more than 15 takes more than 100 minutes which slows down the process but in demographics data, it is very unlikely that the average number of bigrams would be more than 10.

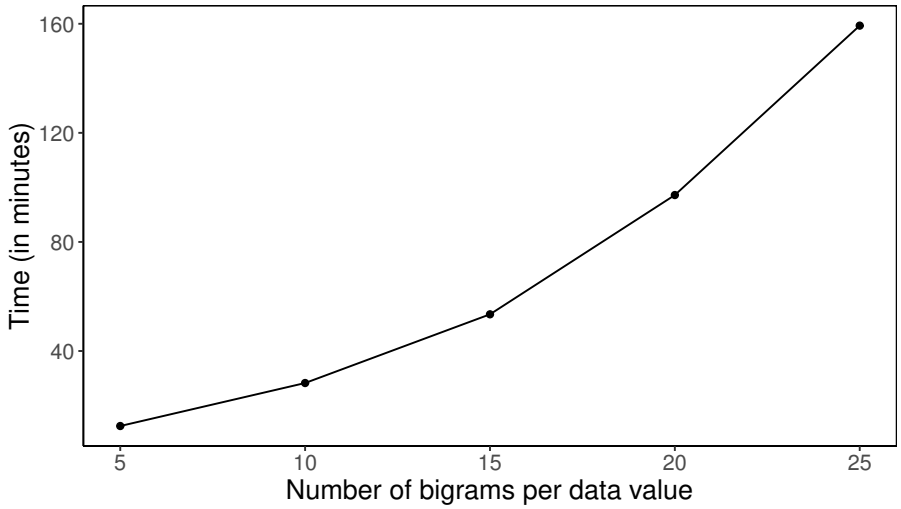


Figure 5.4: Linkage time for different number of bigrams per data value

5.3.3 Pohlig-Hellman versus ECC Approach

We compared the linkage execution time for the Pohlig-Hellman approach and the EC-based approach. The linkage times for the two approaches on 100,000 records with 5 fields of the NCVR data set with 25% overlap and 30% error in one data set were 165.6 *minutes* and 16.78 *minutes* respectively. The Pohlig-Hellman approach uses 38 keys and the EC-based approach uses 50 keys. Both experiments have were executed on machine S2. With the proposed improvements in the EC-based approach (usage of hash maps, post-processing of *L*2 files and replacement of modular exponentiations with EC-based transformations), we saw an improvement of 10 folds in the linkage time.

5.4 Exposure Risk

Figure 5.5 depicts the exposure risk of different key ring sizes for the NCVR data subset. Section 4.2.4 discusses how the exposure risk of the encoded bigrams is calculated as the probability of identifying half the bigrams in a string. Without frequency smoothing or

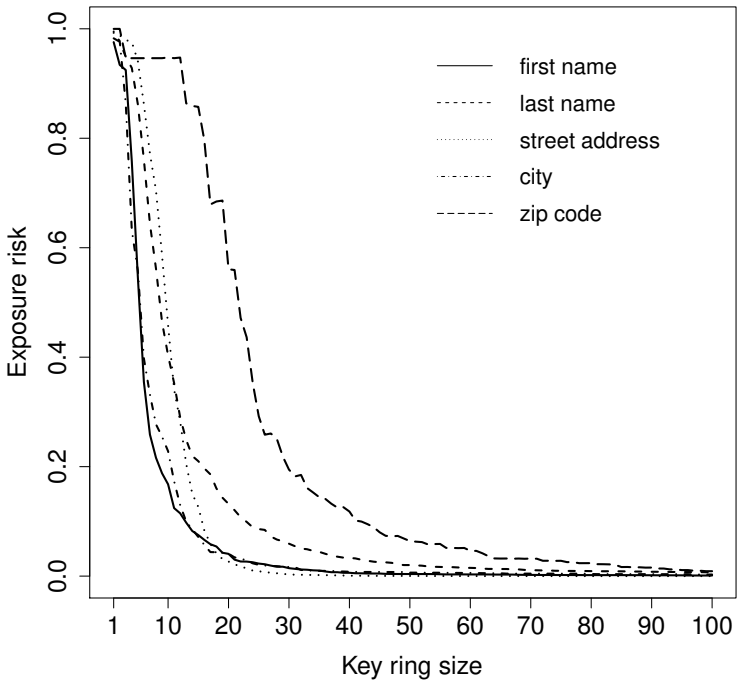
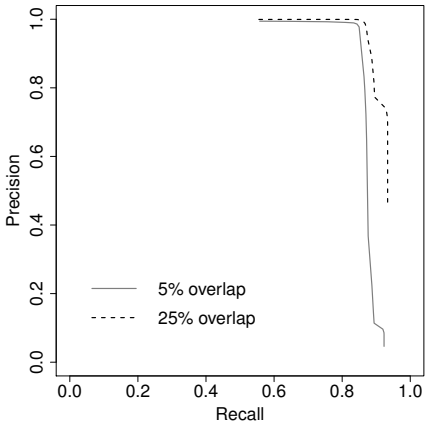


Figure 5.5: Exposure risk associated with identifying half the bigrams in a field with different key ring sizes

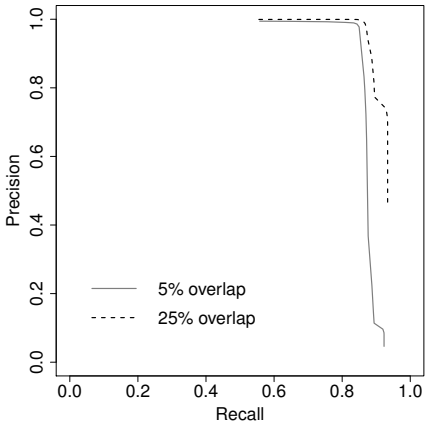
using only one key index exposes half of the bigrams with a probability of 0.99. With increasing size of the key ring, the exposure probability decreases. We chose the default key ring size as 50, and with that, half of the bigrams in the first name have exposure probability 0.0038, in last name - 0.02, street address - 0.0003, city - 0.064 and zip code - 0.064 (smaller bigram domain). Thus with 50 keys, the exposure probability is reasonably small in all the fields. A key ring of size 95 brings down the exposure probability to less than 1%.

5.5 Linkage Accuracy

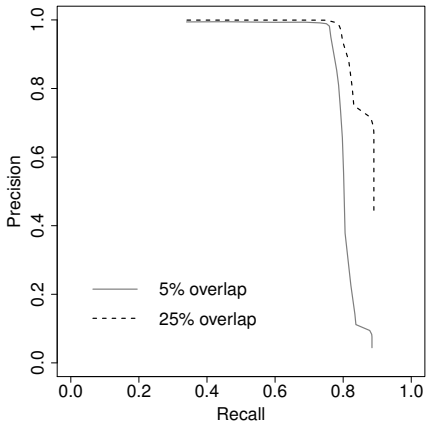
Figure 5.6 depicts the precision-recall assessment for matching thresholds in the range 0 to 1. Precision and recall are used to gauge the linkage accuracy. Precision is the fraction of correct matches in the total number of matched pairs. It assesses the correctness of the linkage algorithm. Recall is the fraction of true match record pairs in the data that are identified as matches by the algorithm. It assesses coverage. Both the precision and recall rates can be maintained at 96% with low error rates by using a high threshold (0.9). But we



(a) Low error (5% records)



(b) Moderate error (30% records)



(c) High error (50% records)

Figure 5.6: Precision and recall of a quadratic record linkage procedure for varying similarity thresholds in $[0, 1]$ (Equal weights are assigned to all fields)

observed that the recall rates drop to approximately 83% for such a high threshold in case of moderate date errors. We noted that it is difficult to maintain high precision and recall rates with low data set overlap. The threshold must be lowered to maintain a balance between precision and recall rates for data sets with high error rates. Thus, discriminatory power of the attributes and the weight of the fields have high impact on the probabilistic record linkage approach. For example, assigning high weightage to street address can maintain high precision and recall rates of about 95% at lower threshold values like 0.7 and 0.8.

5.6 Linkage with Blocking

We discussed briefly about the blocking procedure in Chapter 2. In Table 5.2, we can see that for huge number of records, the EC-based approach is less feasible. Hence, we propose to incorporate the blocking procedure to improve those results. The process to divide a large data set into smaller subsets with respect to a blocking value is called blocking of a field. For instance, we can generate a subset of records with equal attribute values (same zip codes). The linkage exercise is then performed on subsets from both the sites with matching blocking values. We performed blocking of our data set containing 1,000,000 records with blocking on zip codes and name initials. Blocking on zip codes generated 860 subset pairs, each containing 1150 records on an average. Blocking on name initials generated 725 subset pairs, each containing 1370 records on an average. We link each subset pair independently and then combine the results to produce the final linkage result over the larger data set. Table 5.3 shows the linkage time after incorporating the blocking procedure in the linkage exercise.

	Number of vCPUs		
Blocking variable	8	4	1
Zip code	13.76 m	18.26 m	25.62 m
First and last initials	14.54 m	18.93 m	28.66 m

Table 5.3: Execution time (in m: minutes) to link two data sets with 1,000,000 records in each using blocking; Machine S3 is used

Chapter 6

Conclusions and Future Work

The EC-based approach shows substantial improvement in the record linkage time against the Pohlig-Hellman approach. Various optimizations contributed towards the optimized performance of the linkage exercise. Removing the exponentiation operations from data encoding and replacing them with EC-based addition and multiplication operations showed considerable decrease in the precomputation time. Replacing the greedy matching algorithm with the hash map based approach effectively reduced the number of look-ups in the linkage map, which reduced the time to compare a record pair. The effective results of the blocking procedure suggests that private record linkage over a distributed network is feasible even for large data sets using current hardware.

In the experiments, we have assumed the data to be in a standardized format and the matching algorithm makes comparisons with corresponding attributes/fields. The work can be directed further to perform the linkage in non-standardized data sets. We consider two sites along with a third-party, the linkage agent, to perform the record linkage. However, the approach may be scalable to multiple sites and taken further to build a multi-site record linkage protocol. To develop a complete and sustainable FQP system, network layer aspects also need to be taken into consideration. We have emphasized on improving the linkage time performance using the EC-based approach. But a complete tool chain involves other components for a full fledged client-server web application. Client side application would involve discovery of peers, stream-lined execution of precomputation workflow, database schema integration and secure upload/download of files. Server side application would require a scheduler to load balance the linkage tasks to enable real time federated query

processing systems. These aspects remain to be explored and built which may reveal multiple technical issues.

We compute the exposure risk based on static frequency analysis (frequencies are considered independently). It is possible to perform a dynamic frequency analysis using frequency distribution of one or more known bigrams. A demographics dictionary can be used for inference of bigrams from their encodings, similar to the frequency attacks using a dictionary. Assessment based on such exposure risk is an attractive future direction.

Bibliography

[1] 104^{th} Congress, United States. "Health Insurance Portability and Accountability Act of 1996." *Public Law 104-191*, 1996.

[2] L. M. Schilling et al. "Scalable Architecture for Federated Translational Inquiries Network (SAFTINet) technology infrastructure for a distributed data network." *eGEMs(Generating Evidence and Methods to improve patient outcomes)*, 1(1):1027, 2013.

[3] H. A. Dunn. "Record linkage." *American Journal of Public Health and the Nations Health*, 36(12):1412–1416, 1946.

[4] I. Fellegi and A. Sunter. "A theory for record linkage". *Journal of the American Statistical Society*, 64:1183-1210, 1969.

[5] L. Dusserre, C. Quantin, and H. Bouzelat. "A one way public key cryptosystem for the linkage of nominal files in epidemiological studies." *MedInfo*, 8 (Pt 1):644–647, 1995.

[6] T. Churches and P. Christen. "Blind data linkage using n-grams similarity comparisons." *In Advances in Knowledge Discovery and Data Mining*, pages 121–126, 2004.

[7] R. Schnell, T. Bachteler, and J. Reiher. "Privacy-preserving record linkage using bloom filters." *BMC Medical Informatics and Decision Making*, 9:41, 2009.

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

[9] R. Dewri, T. Ong, R. Thurimella. "Linking Health Records for Federated Query Processing". *Proceedings on Privacy Enhancing Technologies*, (3):4-23, 2016

[10] G. Schadow, S. J. Grannis, C. J. McDonald. "Privacy preserving distributed queries for a clinical re-search network." *IEEE International Conference on Data Mining Workshop on Privacy, Security, and Data Mining*, 2002.

[11] R. Rivest, A. Shamir, L. Adleman. "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems" *Communications of the ACM*, 21 (2): 120–126, 1978.

[12] S. Pohlig and M. Hellman. "An Improved Algorithm for Computing Logarithms over GF(p) and its Cryptographic Significance." *IEEE Transactions on Information Theory*, (24): 106–110, 1978.

[13] U.S. National Security Agency. "Commercial National Security Algorithm Suite and Quantum Computing FAQ." 2016.

[14] OpenSSL. "Elliptic Curve Cryptography." URL: `https://wiki.openssl.org/index.php/Elliptic_Curve_Cryptography`.

[15] Canadian Parliament. "Family Allowances Act", 1944.

[16] W. E. Winkler. "The state of record linkage and current research problems." *Technical report, Statistical Research Division*, U.S. Census Bureau of the Census, 1999.

[17] M. Bellare, R. Canetti, H. Krawczyk. "Keying Hash Functions for Message Authentication." 1996.

[18] T. Churches and P. Christen. "Some methods for blindfolded record linkage." *BMC Medical Informatics and Decision Making*, 4:9, 2004.

[19] E. Durham, Y. Xue, M. Kantarcioglu, and B. Malin. "Private medical record linkage with approximate matching." *AMIA Annual Symposium Proceedings*, pages 182–186, 2010.

[20] North Carolina State Board of Elections. URL: `https://dl.ncsbe.gov/index.html?prefix=Voter_Registration/`.

[21] E. A. Durham, et al. "Composite bloom filters for secure record linkage." *IEEE Transactions on Knowledge and Data Engineering*, 26(12):2956–2968, 2013.

[22] B. Malin and E. Airoldi. "Confidentiality preserving audits of electronic medical record access." *Studies in Health Technology and Informatics*, 129(1):320–324, 2007.

[23] B. Pinkas, T. Schneider, and M. Zoner. "Faster private set intersection based on OT extension." *23rd USENIX Conference on Security Symposium*, pages 797–812, 2014.

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

[25] Michael O. Rabin. “How to exchange secrets by oblivious transfer.” *Technical Report TR-81, Aiken Computation Laboratory*, Harvard University, 1981.

[26] Certicom. “SEC 2: Recommended elliptic curve domain parameters”. *Technical report, Certicom Research*, 2000.

[27] H. Bouzelat, C. Quantin, and L. Dusserre. “Extraction and anonymity protocol of medical file.” *AMIA Annual Fall Symposium*, pages 323–327, 1996.

[28] A. Karakasidis and V. S. Verykios. “Privacy preserving record linkage using phonetic codes.” *Balkan Conference in Informatics*, pages 101–106, 2009.

[29] M. Kuzu, M. Kantarcioglu, E. Durham, and B. Malin. “A constraint satisfaction cryptanalysis of bloom filters in private record linkage.” *International Conference on Privacy Enhancing Technologies*, pages 226–245, 2011.

[30] 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*, 20(2):285–292, 2013.

[31] W. Mitchell, R. Dewri, R. Thurimella, and M. Rosckhe. “A graph traversal attack on bloom filter based medical data aggregation.” *International Journal of Big Data Intelligence*, 4(4):217–226, 2017.

[32] F. Niedermeyer, S. Steinmetzer, M. Kroll, and R. Schnell. “Cryptanalysis of basic bloom filters used for privacy preserving record linkage.” *Journal of Privacy and Confidentiality*, 6(2):59–79, 2014.

[33] D. Vatsalan, P. Christen, and V. S. Verykios. “An efficient two-party protocol for approximate matching in private record linkage.” *In Proceedings of the 9th Australasian Data Mining Conference*, pages 125–136, 2011.

[34] R. Schnell, T. Bachteler, and J. Reiher. “A novel errortolerant anonymous linking code.” *Technical Report WPGRLC-2011-02, German Record Linkage Center*, 2011.

[35] P. Christen, R. Schnell, D. Vatsalan, and T. Ranbaduge. “Efficient cryptanalysis of bloom filters for privacy-preserving record linkage.” *Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 628–640, 2017.

[36] M. Kroll and S. Steinmetzer. "Who is 1011011111...1110110010? Automated cryptanalysis of bloom filter encryptions of databases with several personal identifiers." *Proceedings of the International Joint Conference on Biomedical Engineering Systems and Technologies*, pages 341–356, 2015.

[37] A. Appleby. "MurMurHash3 on Github." URL: `https://github.com/aappleby/smhasher/blob/master/src/MurmurHash3.cpp`, 2008.

[38] D. Knuth. "Knuth shuffle algorithm." *The Art of Computer Programming, Volume 2, Seminumerical algorithms*, pp. 139–140, 1969.

[39] M. Vose. "A linear algorithm for generating random numbers with a given distribution." *IEEE Transactions on Software Engineering*, 17(9): 972-975, 1991.

[40] S. J. Grannis, J. M. Overhage, and C. McDonald. "Analysis of identifier performance using a deterministic linkage algorithm." *AMIA Annual Symposium Proceedings*, pages 305–309, 2002.

[41] E. van Eycken et al. "Evaluation of the encryption procedure and record linkage in the Belgian National Cancer Registry." *Archives of Public Health*, 50(6):281–294, 2000.

[42] L. Bonomi, L. Xiong, R. Chen, and B. Fung. "Frequent grams based embedding for privacy preserving record linkage." *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, pages 1597–1601, 2012.

[43] E. Ioannou, W. Nejdi, C. Niederee, and Y. Velegrakis. "On-the-fly entity-aware query processing in the presence of linkage." *Proceedings of the VLDB Endowment*, 3(1-2):429–438, 2010.

[44] A. Inan, M. Kantarcioglu, E. Bertino, and M. Scannapieco. "A hybird approach to private record linkage." *International Conference in Data Engineering*, pages 496–505, 2008.

[45] A. Inan, M. Kantarcioglu, G. Ghinita, and E. Bertino. "Private record matching using differential privacy." *International Conference on Extending Database Technology*, pages 123–134, 2010.

[46] H. Chen, K. Laine, and P. Rindal. "Fast private set intersection from homomorphic encryption." *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 1243–1255, 2017.