PRIVACY-ENHANCING TECHNIQUES FOR DATA ANALYTICS

A Dissertation

Submitted to the Faculty

of

Purdue University

by

Fang-Yu Rao

In Partial Fulfillment of the

Requirements for the Degree

of

Doctor of Philosophy

May 2019

Purdue University

West Lafayette, Indiana

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF DISSERTATION APPROVAL

Dr. Elisa Bertino, Chair

Department of Computer Science

Dr. Ninghui Li

Department of Computer Science

Dr. Sonia Fahmy

Department of Computer Science

Dr. Dan Goldwasser

Department of Computer Science

Approved by:

Dr. Voicu Popescu

Head of the Graduate Program, Department of Computer Science

To my family.

ACKNOWLEDGMENTS

I would like to first thank Prof. Elisa Bertino, my academic advisor, as well as a role model that inspires and motivates me to remain persistent and determined when facing difficulties in research. This dissertation would not be possibly completed without her continual encouragement and financial support for the past five and a half years. I also feel very fortunate to have a chance to work with many bright collaborators including Dr. Jianneng Cao, Prof. Gabriel Ghinita, Prof. Murat Kantarcioglu, Prof. Bharath Samanthula, and Prof. Xun Yi, each of whom has provided great help in various research projects I have been involved in. In additon, I would like to express my appreciation to Prof. Sonia Fahmy, Prof. Dan Goldwasser, and Prof. Ninghui Li, who served on the committee of my preliminary and final exams, and made many helpful suggestions as well. I would also like to extend my sincere thanks to Prof. Greg Frederickson, Prof. Samuel Wagstaff, Prof. Bernd Ulrich, and Prof. William Heinzer. The courses they offered help me develop necessary technical knowledge to conduct the research. I am also grateful to the Department of Computer Science at Purdue for supporting me as a teaching assistant in my first few years so that I was able to focus on my studies. Another person to whom I owe a debt of gratitude is the advisor of my master's thesis Prof. Dah-Jyh Guan. I can never be accepted to Purdue in the first place without his letter of recommendation. I have to additionally thank Lacey Siefers, Shelley Straley, and Monica Shively in my department for helping me with the forms that have to be generated and approved prior to my dissertation defense. I am also grateful for the funding provided by the Purdue Cyber Center and the National Science Foundation under grant CNS-1111512 that supported the research in this dissertation.

My appreciation also goes to the friends I met while I was at Purdue, which include Kurt Aikens, Chun-Mei Chiu, Wei-Chiu Chuang, Wei-Yen Day, Chao-Jung

Hsu, Terry Ching-Hsiang Hsu, Enyi Jen, Hou-Jen Ko, I-Fan Lin, Dong Su, Yu-Chi Su, Tianhao Wang, Sherry Wei, Jia-Jing Wu, Meng-Lin Wu, and Wei-Min Yao. Each of them was willing to offer generous help every time when I needed a hand. My life at Purdue would have been much more difficult without anyone of them. I also have a deep gratitude towards Hung-Han Chen and Chiun-Teh David Ho, whom I first met in 2009 because of Lollapalooza, an annual music festival based in Chicago. Their constant caring over these years always makes me feel supported. Moreover, I am especially thankful for the help from my previous labmate Wei-Yen Day and my previous landlord Michelle Lam during my stay in the Bay Area last summer. Their help made it much easier for me to adapt to the new environment. I also have to express my gratitude to my colleagues at Cloudera, including Wei-Chiu Chuang, Tony Tuan, Paul Rogers, and Adrian Ng. Their supportiveness and thoughtfulness make it possible for me to finish this dissertation on time. On the other hand, I would also like to thank the following friends in Taiwan: Matt Chang, Hung-Ting Chen, Shao-Yen Chen, Ming-Chen Juan, Chi-Ming Lee, Hsiao-Ying Lin, Sasha Lin, Ching-Fu Chen, Song-Kai Su, Chun-Yuan Chu, Chang-Ching Yeh, Fan-Hao Lin, and Ting-Yu Shi. Their hospitality made my every visit to Taiwan a memorable experience and also made me feel less isolated.

Last but not least, I have to thank my parents for their unconditional support and love during this long course of my studies at Purdue. Their encouragement always help me find the strength to persist with my studies in spite of difficulties in research.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Rao, Fang-Yu Ph.D., Purdue University, May 2019. Privacy-Enhancing Techniques for Data Analytics. Major Professor: Dr. Elisa Bertino.

Organizations today collect and aggregate huge amounts of data from individuals under various scenarios and for different purposes. Such aggregation of individuals' data when combined with techniques of data analytics allows organizations to make informed decisions and predictions. But in many situations, different portions of the data associated with individuals are collected and curated by different organizations. To derive more accurate conclusions and predictions, those organization may want to conduct the analysis based on their joint data, which cannot be simply accomplished by each organization exchanging its own data with other organizations due to the sensitive nature of data. Developing approaches for collaborative privacy-preserving data analytics, however, is a nontrivial task. At least two major challenges have to be addressed. The first challenge is that the security of the data possessed by each organization should always be properly protected during and after the collaborative analysis process, whereas the second challenge is the high computational complexity usually accompanied by cryptographic primitives used to build such privacy-preserving protocols.

In this dissertation, based on widely adopted primitives in cryptography, we address the aforementioned challenges by developing techniques for data analytics that not only allow multiple mutually distrustful parties to perform data analysis on their joint data in a privacy-preserving manner, but also reduce the time required to complete the analysis. More specifically, using three common data analytics tasks as concrete examples, we show how to construct the respective privacy-preserving protocols under two different scenarios: (1) the protocols are executed by a collaborative

process only involving the participating parties; (2) the protocols are outsourced to some service providers in the cloud. Two types of optimization for improving the efficiency of those protocols are also investigated. The first type allows each participating party access to a statistically controlled leakage so as to reduce the amount of required computation, while the second type utilizes the parallelism that could be incorporated into the task and pushes some computation to the offline phase to reduce the time needed for each participating party without any additional leakage. Extensive experiments are also conducted on real-world datasets to demonstrate the effectiveness of our proposed techniques.

1 INTRODUCTION

Organizations today collect and aggregate huge amounts of data from individuals under various scenarios and for different purposes. Such aggregation of individuals' data combined with techniques of data analytics enables organizations to make informed decisions and predictions. However, in many scenarios different portions of the data associated with individuals are collected by different organizations. As a result, no single organization has a comprehensive data view of the user population of interest. However, without such a comprehensive data view, the derived conclusions and predictions could be biased if not totally off the mark. Therefore, it is crucial for organizations to be able to collaboratively perform computations over their joint data, thus creating a win-win situation for each participating organization. Notice that such an approach can also be relevant for single organizations when these organizations have separate organizational units that for privacy, security or compliance reasons cannot share their own data in the clear.

Developing approaches for collaborative privacy-preserving computations requires addressing two major challenges. The first challenge is that the security of the data owned by each organization should be properly protected during and after the collaborative analysis process. This challenge arises from the fact that data sharing among different organizations is usually subject to restrictions imposed because of the sensitive nature of data. Under such restrictions directly revealing the data possessed by one organization to other organizations for analysis is out of question. In an ideal situation, each organization would securely transmit its own data to a trusted third party, who would then perform the analysis on the joint data so that in the end each organization solely retrieves from the trusted party the output of the analysis. Such a trusted party, however, does not always exist in the real world. Without such a trusted party conducting the analysis of the joint data, suitable privacy-preserving

protocols allowing multiple organizations to realize the functionality executed by the trusted party should thus be devised. On the other hand, it is not always the case that each organization has the relevant expertise for developing or deploying such protocols and hence an organization may even want to outsource the task of privacy-preserving analysis to a service provider, adding more complexity to the design of such protocols. In either case, a privacy-preserving protocol should always provide the guarantee that each organization only receives the results of analysis as if there were a trusted party performing the computation on the joint data for those organizations.

The second challenge is the high computational complexity of cryptographic primitives that allow one to build protocols for analyzing the joint data belonging to different organizations in a privacy-preserving manner. Such primitives, although being provably secure based on some computational hardness assumptions, are not always practical especially for large datasets as their computational complexity often grows linearly with the bit lengths of input data. Quite a few primitives even involve computationally intensive operations, e.g., modular exponentiation for large numbers, hindering an organizations's willingness to participate in such a collaborative analysis process.

In this dissertation, based on widely adopted primitives in cryptography, we address the aforementioned challenges by developing privacy-preserving techniques for data analytics that not only enable multiple mutually distrustful parties to conduct data analytics of their joint data in a privacy-preserving way, but also allow those parties to carry out the analysis in much shorter time than the time needed if those primitives were directly applied. In particular, using three common data analytics tasks as concrete examples, we show how to construct the respective privacy-preserving protocols with formal security and privacy guarantees under two different scenarios: (1) the protocols are executed by a collaborative process that only involves the participating parties; (2) the protocols are outsourced to some service providers in the cloud. On the other hand, we also investigate two types of techniques that could be employed to boost the efficiency of those protocols. The first type of techniques

reduces the amount of required computation by allowing each participating party access to a statistically controlled leakage about the other parties' data, whereas the second type reduces the amount of waiting time by exploiting the parallelism as well as pushing some computation to the offline phase without sacrificing the privacy of data, i.e., no additional leakage to any participating party.

To be more specific, we focus on the following three specific data analytics tasks to demonstrate how the techniques described above can be applied to design protocols with provable security guarantees without sacrificing efficiency.

1. **Hybrid Private Record Linkage**. This protocol allows multiple parties to exchange matching records, which refer to the same entities or have similar values, while keeping the non-matching ones secret. Conventional protocols are based on computationally expensive cryptographic primitives and therefore do not scale. To address scalability, previously, hybrid protocols have been proposed that directly combine differential privacy techniques with secure multiparty computation techniques. However, a drawback of such protocols is that they disclose to the parties both the matching records and the differentially private synopses of the datasets involved in the linkage. Consequently, differential privacy is no longer always satisfied. To address this issue, we propose a novel framework, which separates the private synopses from the matching records. With the help from a semi-honest third party, the two parties do not access the synopses directly, but still use them to efficiently link records. We theoretically prove the security of our framework under the state-of-the-art privacy notion of differential privacy for record linkage (DPRL). In addition, we develop a simple but effective strategy for releasing private synopses. Extensive experimental results show that our framework is superior to existing methods in terms of efficiency.

2. **Hybrid Private String Matching**. Unlike the previous protocol that privately matches data records with numerical attributes, this protocol enables

multiple parties to exchange records referring to similar entities based on string-typed attributes. Our protocol is based on the same hybrid framework that incorporates a semi-honest third party which helps in enforcing the privacy guarantee throughout the protocol execution. String-valued data are partitioned into groups, and each partition is represented by a synopsis. To facilitate partitioning, strings are first embedded into a metric space which preserves the relative distance among them, and then they are mapped onto multi-dimensional points. We propose two partitioning algorithms: the first one partitions the data solely based on Lipschitz embedding coordinates, whereas the second one also leverages string length information. Extensive experimental results on a real dataset demonstrate that our solution is efficient. We also show that our solution can be adapted to a newer protection model, i.e., DPRL.

3. **Privacy-Preserving and Outsourced Multi-User k-Means Clustering**. Unlike the previous two protocols where the main computation is performed at the participating parties owning the data, here we consider the scenario where the data owners would like to outsource the task to the service providers in the cloud. We propose a novel and efficient solution for executing the k-means clustering algorithm based on the data records from multiple mutually distrustful parties. The main novelty of our solution is that it does not require the execution of secure division operations at the cloud through efficient transformation techniques. The proposed solution protects data confidentiality of all the participating entities under the standard semi-honest model. Also, we discuss two strategies, namely offline computation and pipelined execution, that aim to boost the performance of our protocol. We implement our protocol on a cluster of 16 nodes and demonstrate how our two strategies combined with parallelism can significantly improve the performance of our protocol through extensive experiments using a real dataset.

The rest of the dissertation is organized as follows. In Chapter 2 we first provide the necessary background knowledge about the primitives we use to construct our protocols. Chapter 3 and Chapter 4 describe in more detail our protocols for private record linkage for numerical and string-valued data, respectively. A detailed description of the protocol for outsourced k-means clustering is given in Chapter 5. We finally conclude the dissertation with possible directions for future research in Chapter 6.

2 PRELIMINARIES

In this chapter, we provide background information about each primitive we use for constructing our privacy-preserving data analytics protocols.

2.1 Secure Multi-Party Computation (SMC)

SMC [1] enables multiple parties to jointly compute a function over their respective inputs, while maintaining the privacy of all the inputs. Let $f : (\{0,1\}^*)^\ell \rightarrow (\{0,1\}^*)^\ell$ be an ℓ-ary function, and $m_1, m_2, \ldots, m_\ell$ be the respective inputs of ℓ parties. SMC keeps all m_i's confidential, but computes $f(m_1, m_2, \ldots, m_\ell) = \{f_i(m_1, m_2, \ldots, m_\ell)\}_{i\in\{1,2,\ldots,\ell\}}$, where $f_i(m_1, m_2, \ldots, m_\ell)$ is the output to the i-th party.

Simulation [1] is a standard methodology to prove the security of an SMC protocol. An SMC protocol is simulatable if there exists a simulator for each party P_i $(i = 1, 2, \ldots, \ell)$, such that P_i's view (i.e., messages received from the other ℓ-1 parties) in the protocol is *indistinguishable* from a simulation of this view using $(m_i, f_i(m_1, m_2, \ldots, m_\ell))$, where m_i is the input of P_i and $f_i(m_1, m_2, \ldots, m_\ell)$ is the protocol's output to P_i. Intuitively, an SMC protocol is simulatable if each participating party gains no additional knowledge other than its respective input and output after interacting with other parties in the protocol.

To be specific, let $\prod$ be an SMC protocol for privately computing function f on $(m_1, m_2, \ldots, m_\ell)$. The *view* of P_i during a real execution of $\prod$ is $\text{VIEW}_i^{\prod}(m_1, \cdots, m_\ell)$, equal to $(m_i, r_i, V_i^1, \cdots, V_i^t)$, where r_i is the internal randomness of P_i, V_i^j is the j-th message received by P_i in the protocol, and $1 \leq j \leq t$. We say $\prod$ is simulatable, if there exists a polynomial-time simulator S_i, such that the distribution of

$S_i(m_i, f_i(m_1, \cdots, m_\ell))$ is *indistinguishable* from that of $\text{VIEW}_i^{\Pi}(m_1, \cdots, m_\ell)$ [1]. The indistinguishability can be computational or statistical as defined below.

Definition 2.1.1 *[2] Let $\Omega_\pi = \{0,1\}^\pi$ be a finite domain of π-bit numbers, and X and Y be two random variables over Ω_π. Let G_π be a probabilistic algorithm, which outputs either 0 or 1 given a value in Ω_π. We say that the distribution of X is computationally indistinguishable from that of Y, if for any probabilistic polynomial-time distinguisher G_π, any positive polynomial $p(\pi)$, and all sufficiently large π's*

$$|\Pr[G_\pi(X) = 1] - \Pr[G_\pi(Y) = 1]| < \frac{1}{p(\pi)}.$$

Let $t(\pi)$ be a polynomial in π. Suppose that X_i is computationally indistinguishable from Y_i, for $1 \leq i \leq t(\pi)$. Then we note that $(X_1, \ldots, X_{t(\pi)})$ and $(Y_1, \ldots, Y_{t(\pi)})$ are computationally indistinguishable as well according to the hybrid argument [3]. Briefly speaking, if there exists a polynomial-time distinguisher G_π that could distinguish $(X_1, \ldots, X_{t(\pi)})$ from $(Y_1, \ldots, Y_{t(\pi)})$, it would imply that G_π could be used to distinguish X_i from Y_i for some i, a contradiction.

Definition 2.1.2 *[3] Let $\Omega_\pi = \{0,1\}^\pi$ be a finite domain of π-bit numbers, and X and Y be two random variables over Ω_π. We say that X and Y are statistically indistinguishable, if for any positive polynomial $p(\pi)$ and all sufficiently large π's, their statistical distance is negligible, i.e.,*

$$\frac{1}{2} \sum_{v \in \Omega_\pi} |\Pr[X = v] - \Pr[Y = v]| < \frac{1}{p(\pi)}.$$

Note that if the distributions of X and Y are statistically indistinguishable, then they are also computationally indistinguishable [3].

Secure Integer Comparison. Yao's garbled circuit [4] is an instantiation of SMC. Kolesnikov et al. [5] apply it for secure integer comparison—given two integers x and y of two parties, respectively, the solution [5] outputs whether $x \geq y$ without disclosing their values.

[1] r_i is uniformly random in honest-but-curious setting [1]. So in the proof we focus on P_i's received messages.

2.2 Paillier Cryptosystem

The Paillier cryptosystem [6] is one of the widely adopted partially homomorphic public-key cryptosystems. Let $n = pq$ be a large RSA modulus for two large primes p and q, $\mathbb{Z}_n = \{x \in \mathbb{Z} \mid 0 \leq x \leq n-1\}$, and $\mathbb{Z}_n^* = \{x \in \mathbb{Z}_n \mid \gcd(x, n) = 1\}$. Other than n, the public key pk also includes $g \in \mathbb{Z}_{n^2}^*$, which is an element of order equal to a non-zero multiple of n modulo n^2. The private key sk of the system is (λ, μ), where $\lambda = \text{lcm}(p-1, q-1)$, and $\mu = [(g^\lambda \bmod n^2 - 1)/n]^{-1} \bmod n$. To encrypt a message $m \in \mathbb{Z}_n$ using the public key $pk = (g, n)$, we compute its encryption $E_{(g,n)}(m)$ as

$$E_{(g,n)}(m) = g^m r^n \bmod n^2, \tag{2.1}$$

where r is drawn from $\mathbb{Z}_n^*$ uniformly at random. The decryption of a ciphertext $c \bmod n^2$ works as

$$D_{(\lambda,\mu)}(c) = [(c^\lambda \bmod n^2 - 1)/n] \cdot \mu \bmod n. \tag{2.2}$$

The Paillier cryptosystem supports addition over ciphertexts, that is, $E_{pk}(m_1)$ $\cdot\, E_{pk}(m_2) \bmod n^2$ corresponds to a ciphertext of $m_1 + m_2 \bmod n$. Suppose that (pk, sk) is a pair of public and private keys corresponding to the Paillier cryptosystem. Another useful property resulting from the property above is that for any message $m \in \mathbb{Z}_n$ and any integer i, we have

$$D_{sk}(E_{pk}(m)^i) = i \cdot m \bmod n. \tag{2.3}$$

The parameter g in practice could be chosen as $g = (1 + n)$ without sacrificing the security of the scheme. For the simplicity of notations, we sometimes omit modulo operation in our equations. Paillier cryptosystem is probabilistic; repeated encryptions of the same plaintext produce different ciphertexts with high probability. Furthermore, it is semantically secure under the intractability assumption of decisional composite residuosity. Specifically, for any given plaintexts m_1 and m_2, the distributions of their respective ciphertexts are computationally indistinguishable (Definition 2.1.1).

2.3 Differential Privacy

Differential privacy [7] requires that the output of a function on the dataset be essentially the same, even if any single record is added to or removed from the dataset. Such a concept ensures that adversarial attacks against a single individual are essentially equally likely to occur, whether that individual's record is in the database or not. Therefore, differential privacy provides a strong privacy guarantee, and has become the de facto standard notion of privacy.

Definition 2.3.1 (differential privacy [7]) *Let r be a record, and (D, D') to be a pair of neighboring datasets, such that $D = D' \cup \{r\}$ or $D' = D \cup \{r\}$. A randomization mechanism $\mathcal{M}$ satisfies ϵ-differential privacy, if for any pair (D, D') and any subset $O \subseteq Range(\mathcal{M})$,*

$$\Pr[\mathcal{M}(D) \in O] \leq e^{\epsilon} \cdot \Pr[\mathcal{M}(D') \in O]. \tag{2.4}$$

Let F be a set of functions. Its L_1 *sensitivity* [7] is defined to be

$$\mathsf{Sen}\,(F) = \max_{\forall (D,D')} \left(\sum_{f \in F} |f(D) - f(D')| \right), \tag{2.5}$$

where D and D' are two neighboring datasets and f is any function in F. Dwork et al. [7] have proved that differential privacy can be achieved by adding to each function output a noise following Laplace distribution. Specifically, the noise is a random variable $\mathsf{Lap}\,(\lambda)$ with probability density function

$$\mathsf{pdf}\,[\mathsf{Lap}\,(\lambda) = z, |\mu, \lambda] = \frac{1}{2\lambda} \exp\left(-\frac{|z - \mu|}{\lambda} \right), \tag{2.6}$$

where μ is the mean and λ is the scale. Usually, μ is set to 0. A randomization mechanism complies with ϵ-differential privacy, if it adds to each function $f \in F$ independent Laplace noise $\mathsf{Lap}\,(\lambda)$, where $\lambda = \frac{\mathsf{Sen}(F)}{\epsilon}$.

The exponential mechanism is more general, and is used for answering queries that have non-numerical outputs [8]. It draws samples from the output range such that the probability of an output being drawn increases exponentially with its *quality score*. That is, a *better* output will be selected with a substantially higher probability.

Specifically, let $\mathcal{D}$ be the set of all possible databases that could exist, and let $\mathcal{R}$ be the set of all possible output values. According to a quality function $q : \mathcal{D} \times \mathcal{R} \to \mathbb{R}$, ϵ-differential privacy could be achieved if an output $\rho \in \mathcal{R}$ is chosen with probability proportional to $\exp(\epsilon q(D, \rho)/(2S(q)))$, where $S(q)$ is the sensitivity of q, defined as the largest possible difference in the output of q for any possible output ρ and any pair of neighboring databases D and $D' \in \mathcal{D}$, i.e.,

$$S(q) = \max_{\forall (D, D'), \rho} (|q(D, \rho) - q(D', \rho)|). \tag{2.7}$$

Composability. Differential privacy has the property of *composability*. Given a set of mechanisms, which satisfy differential privacy with regard to privacy parameters $\epsilon_1, \epsilon_2, \ldots, \epsilon_n$, respectively, then they as a whole satisfy ϵ-differential privacy for $\epsilon = \sum_{i=1}^{n} \epsilon_i$. Parameter ϵ is usually referred to as the total privacy budget. Given a task that consists of multiple steps, to satisfy ϵ-differential privacy, a portion of privacy budget is assigned to each step so that the summation of all the portions is at most ϵ.

3 HYBRID PRIVATE RECORD LINKAGE

Data integration across multiple organizations is crucial for many different applications, including e-commerce, health care, and national security. However, as data often encodes sensitive information, it is critical to make sure that no information more than necessary is exchanged [2]. As an example, consider two hospitals that, in order to conduct research on how to enhance diagnosis services to patients, need to exchange records of patients with similar background (e.g., similar values on sex, age, occupation, and disease). To protect patients' privacy, only matching records are exchanged, whereas all the non-matching records remain private.

Private linkage protocols have been proposed to address the above requirement. They allow multiple parties to exchange matching records (i.e., records referring to the same entities or records with similar values), while hiding non-matching ones. Secure multi-party computation (SMC) techniques are commonly used to implement private linkage protocols. In a typical SMC protocol, given two datasets belonging to two parties respectively, a record in one dataset is securely compared with every record in the other dataset. Such a protocol has a time complexity of $O(m \times n)$, where m and n are the sizes of the two datasets. Furthermore, each comparison involves computationally expensive cryptographic operation. Thus, traditional SMC-based private linkage protocols are prohibitively expensive. Experimental results show that it takes a single Intel Core i7-2600 CPU machine 9.16 days to complete the private linkage of two datasets, each having 6,000 records and 5 attributes.

To enhance the scalability of private record linkage protocols, Ali et al. [9] proposed a hybrid approach, under which each party partitions its own dataset into subsets and releases a synopsis of each such subset (i.e., the subset extent and size) to the other party. Record matching between 'faraway' subsets is then pruned, and thus efficiency is greatly improved. To protect the privacy of the records in each subset, the subset

size is randomized by differential privacy [7] before being released. Experimental results [9] show that the hybrid approach improves the linkage efficiency by 2 orders of magnitude. Such an approach has, however, the drawback of not always satisfying differential privacy. At the end of the private linkage, each party obtains both the matching records and the differentially private sizes of the subsets. When these two sources of information are combined together, differential privacy for the non-matching records does not hold any more. Refer to Section 3.2 for details.

To address such a drawback, we propose a novel hybrid private record linkage framework [10]. Our framework consists of three parties: Alice and Bob are data owners, and Charlie is a third party. The three parties are *honest-but-curious* [1]. Alice and Bob partition their datasets into subsets and send to Charlie differentially private synopses of the subsets. Charlie coordinates the record matching between Alice and Bob. Based on the received synopses, he prunes the record matching between subsets with a distance beyond a threshold specified by Alice and Bob. During the entire protocol, Charlie does not access any record value. At the end of the protocol, his knowledge remains limited to the private synopses released to him by Alice and Bob. At the same time, Alice and Bob obtain the matching records. But they neither learn the attribute values of non-matching records nor the private synopses of the subsets. Our framework separates the differentially private synopses from matching records and thus mitigate the aforementioned information leakage in [9]. On the other hand, He et al. in a follow-up work [11] investigate the problem of record linkage in the absence of the third party. Specifically, they propose the notion of differential privacy for record linkage (DPRL), an instance of output-constrained differential privacy. This new notion of DPRL is important since it allows one to construct efficient record linkage protocols that offer an end-to-end privacy guarantee for any non-matching record after the matching record pairs have been identified. However, the 2-party record linkage protocol proposed by He et al. [11] suffers from the disadvantage that each data owner learns the noisy number of *non-matching* records in each partition of the other party after the linkage process, which may be

undesirable. In addition, the indexing scheme used by He et al. [11] to partition the data owners' datasets is not adaptive. More precisely, data points are simply partitioned according to a predefined uniform grid, which may not always produce an optimal partitioning. Our approach, like the framework in [10], addresses those two shortcomings by adopting an honest-but-curious third party Charlie in charge of matching coordination and by incorporating the notion of DPRL into scalable private record linkage between two data owners. To be more specific, our main contributions include:

1. We propose a new framework with an end-to-end privacy guarantee that allows a data owner to further hide the statistics of non-matching records from the other data owner after record linkage. After the linkage, except for the matching record pairs, each data owner only learns the number of required secure comparisons conforming to DPRL, the state-of-the-art privacy notion for record linkage. On the other hand, Charlie, as a third party, only observes the synopses of subsets conforming to DPRL.

2. Moreover, our framework allows the data owners to adopt any adaptive indexing scheme compatible with DPRL to generate the partitionings of their respective datasets. Specifically, as long as it can be proved that the number of required secure comparisons resulting from the adopted indexing scheme complies with DPRL, the private record linkage protocol as a whole satisfies DPRL. As far as we know, this is the first hybrid private linkage solution that allows one to adopt an adaptive indexing scheme that is compatible with DPRL.

3. We prove the compatibility of our private indexing scheme [10] with DPRL. Our scheme hierarchically partitions a dataset, forming a tree. Whenever a node in the tree needs to be split, our scheme dynamically allocates a privacy budget based on the node size, in such a way that the magnitude of added noise does not dominate the node size. In other words, our scheme optimizes the accuracy of the noisy node size in relative terms.

4. We have carried out an extensive experimental evaluation under the privacy notion of DPRL. The experimental results show that our approach is superior to existing approaches with respect to efficiency in the context of private record linkage.

The remaining of this chapter is organized as follows. The next section introduces background knowledge. In Section 3.2 we analyze the security of the existing hybrid approach. In Section 3.3 we propose our new framework for private record linkage. We present a simple but effective scheme to privately partition data in Section 3.4. Section 3.5 reports the experimental results. We survey the related work in Section 3.6.

3.1 Background

This section provides background about private record linkage, differential privacy, and cryptographic techniques.

3.1.1 Record Linkage

We assume that Alice and Bob share a data schema (A_1, A_2, ..., A_d, A_{d+1}, ..., A_ℓ), of which the first d attributes are used for the linkage. The two parties jointly define distance functions on the linking attributes. Let $\mathrm{Dom}(A_i)$ be the domain of attribute A_i. The distance function on A_i is defined:

$$\mathsf{Dist}_i : \mathrm{Dom}(A_i) \times \mathrm{Dom}(A_i) \to \mathbb{R}_0^+,$$

where $\mathbb{R}_0^+ = \{x \in \mathbb{R} \mid x \geq 0\}$. Based on distance functions, a decision rule determines whether two records match or not. Here, we give a decision rule based on the Fellegi-Sunter model [12], which is widely used for record linkage.

Definition 3.1.1 (Decision Rule) *Let Z be the set of all possible records. A decision rule is a predicate that is* true *if two records in Z match in the linking attributes.*

$$\mathsf{Rule} : Z \times Z \to \{\mathsf{true}, \mathsf{false}\}.$$

The distance function and decision rule defined above are general. Prior to record linkage, they need to be instantiated according to application requirements. In the following, we provide an instantiation.

Instantiation. Let x and y be two records, and $A_1, A_2, \ldots, A_d$ be the linking attributes. We scale the domains of all attributes to $[L, U]$, where $L, U \geq 0$, and scale the attribute values accordingly. We define the attribute distance on A_i using *normalized squared Euclidean distance*

$$\mathsf{Dist}_i\left(x.A_i, y.A_i\right) = \left(\frac{x.A_i - y.A_i}{U - L}\right)^2. \tag{3.1}$$

The decision rule is then instantiated to the weighted sum of the attribute differences

$$\mathsf{Rule}_\theta(x, y) = \sum_{i=1}^{d} w_i \cdot \delta_i \leq \theta, \tag{3.2}$$

where $\delta_i = \mathsf{Dist}_i\left(x.A_i, y.A_i\right)$, and weight w_i and threshold θ are pre-defined by Alice and Bob. We denote this instantiation by SSE, an acronym for sum of weighted squared Euclidean distance.

3.1.2 An Extended Notion of Differential Privacy for Record Linkage

Recently He et al. [11] propose the notion of *Differential Privacy for Record Linkage* (DPRL), which is an instance of output-constrained differential privacy specifically targeted at the matching problem. DPRL is able to provide an end-to-end quantifiable privacy guarantee for non-matching records after the matching record pairs have been identified. In [11], DPRL is adopted in a two-party setting where two dataset owners Alice and Bob interact directly with each other, whereas in this chapter, we consider the scenario in which a third party Charlie coordinates the matching process between Alice and Bob. The knowledge of Charlie regarding Alice's and Bob's datasets thus has to be taken into consideration as well.

We start by formally defining the functionality of a record linkage protocol. Given datasets D_A and D_B owned by Alice and Bob, let f_{Rule} be the function that computes

the set $\{(x, y) \mid x \in D_A, y \in D_B, \mathsf{Rule}(x, y) = \mathsf{true}\}$ of matching record pairs according to decision rule *Rule*. We then can define the neighboring datasets for record linkage, followed by the formal definition of DPRL.

Definition 3.1.2 *(Neighboring Datasets for Record Linkage [11]) Two datasets D_B and D'_B of Bob are neighboring datasets if $|D_B| = |D'_B|$ and D_B and D'_B differ only in a pair of distinct non-matching records r_B and r'_B with respect to f_{Rule} and D_A, i.e., $D'_B = D_B \backslash \{r_B\} \cup \{r'_B\}$, $r_B \neq r'_B$, and $f_{\mathsf{Rule}}(D_A, D_B) = f_{\mathsf{Rule}}(D_A, D'_B)$.*

The protocol Π for private record linkage we consider in this chapter involves a third party Charlie in addition to the two data owners Alice and Bob. The two data owners first privately partition their respective datasets D_A and D_B into disjoint subsets. Dummy records are then added to each subset to hide the existence of any non-matching record. For each subset, each data owner encrypts the records (including those dummy records) using her/his public key and sends those ciphertexts to Charlie. Furthermore, each data owner sends to Charlie the *synopses* of the subset that enable Charlie to prune unnecessary secure comparisons between records residing in subsets that are far away according to the decision rule Rule. The *synopsis* of a subset includes its *extent* and *noisy counts*, each of which will be formally defined in Section 3.2. After the pruning process, Charlie notifies both parties of the required number of secure comparisons. Charlie then *randomizes* the ciphertexts of each potentially matching record pair. All potentially matching record pairs randomized by Charlie will be sent to both data owners. The two data owners finally invoke a cryptographic matching protocol to jointly compute the matching records.

According to the description above, Alice's view $\mathrm{VIEW}_A^{\Pi}(D_A, D_B)$ during the execution of the protocol Π on input (D_A, D_B) includes: (1) the number of required secure comparisons, and (2) the *messages* received from Bob due to the invocation of the cryptographic matching protocol. Bob's view $\mathrm{VIEW}_B^{\Pi}(D_A, D_B)$ during the execution of the protocol can be similarly defined. On the other hand, Charlie's view $\mathrm{VIEW}_C^{\Pi}(D_A, D_B)$ during the execution of the protocol consists of: (1) the *extent* and

the *noisy count* of each partition derived from both data owners' datasets, and (2) the respective ciphertexts in each partition encrypted under the public key of the corresponding dataset owner.

To protect the privacy of any non-matching record, we need to guarantee that the view of each party involved in protocol Π does not vary too much even when we replace any non-matching record r_B with another distinct non-matching record r'_B. Specifically, we have the following definition.

Definition 3.1.3 *(DPRL) A private record linkage protocol* Π *for computing* f_{Rule} *satisfies* $(\epsilon, \delta, f_{\mathsf{Rule}})$*-DPRL if for any neighboring datasets* (D_B, D'_B) *with respect to private record linkage with the decision rule* Rule*, the view of Alice during the execution of* Π *when given as input to any given probabilistic polynomial time distinguisher* G *satisfies the following inequality*

$$\Pr[G(\mathrm{VIEW}_A^{\Pi}(D_A, D_B)) = 1] \leq e^{\epsilon} \Pr[G(\mathrm{VIEW}_A^{\Pi}(D_A, D'_B)) = 1] + \delta, \tag{3.3}$$

where $\mathrm{VIEW}_A^{\Pi}(D_A, D_B)$ *denotes the view of Alice during the execution of* Π *on* (D_A, D_B)*. Similarly, the view* $VIEW_C^{\Pi}(D_A, D_B)$ *of Charlie under* D_B *has to be similar to that under* D'_B*, i.e.,*

$$\Pr[G(\mathrm{VIEW}_C^{\Pi}(D_A, D_B)) = 1] \leq e^{\epsilon} \Pr[G(\mathrm{VIEW}_C^{\Pi}(D_A, D'_B)) = 1] + \delta. \tag{3.4}$$

The above must also hold with respect to any pair of Alice's neighboring datasets (D_A, D'_A)*.*

3.2 Analyzing the Hybrid Approach

Inan et al. [9] propose a hybrid approach to boost the efficiency of private linkage. Alice and Bob first partition their respective datasets into subsets. For each subset, a private synopsis is generated as defined below.

Definition 3.2.1 (Private Synopsis) *Let* c *be a subset of records in a* d*-dimensional dataset. The synopsis of* c *contains: 1) the extent of the subset* $\{[A_i^{\ell}, A_i^{u}]$

$\mid 1 \leq i \leq d\}$, *where* A_i^ℓ *and* A_i^u *are the lower and upper bounds of* c *along the* i*-th dimension respectively, and 2) the differentially private subset size of* c.

The randomization of the subset size via differential privacy is for privacy, since a synopsis corresponds to a function (or a query), which counts the number of records inside the subset. Let t be true size of a subset. Its randomized size is

$$\widetilde{t} = t + \mathrm{rpi}(\mathsf{Lap}\,(\lambda)),$$

where $\mathrm{rpi}(x) = (\lfloor x + 1/2 \rfloor)$ and $\mathsf{Lap}\,(\lambda)$ is the Laplace noise as defined in Equation 2.6. Laplace noise is of real value. To ensure that the support of any output integer is non-zero, $\mathsf{Lap}\,(\lambda)$ is rounded half up to $\mathrm{rpi}(x)$. Because of the injection of noise, the subset is adjusted accordingly. If the rounded noise is positive, a number of dummy records equal to the value of rounded noise are added to the subset. Otherwise, a number of records equal to the absolute value of the rounded noise are selected uniformly at random from the subset and suppressed. Note that to ensure that dummy records do not appear in the matching result, the attribute values of dummy records of Alice are set to $-U$ and those of Bob are set to $2U$, where U is the upper bound of attribute value. The addition or removal of records guarantees that the set of records later participating in the private linkage protocol are consistent with the released synopses.

Alice and Bob exchange the private synopses. Given a pair of subsets c_1 and c_2 of Alice and Bob, respectively, a *lower (upper) bound distance* between any record $x \in c_1$ and $y \in c_2$ is calculated based on the extents of c_1 and c_2. If the lower bound is bigger than the threshold defined in the decision rule, Alice and Bob safely conclude that the pair does not contain matching records. Instead, if the upper bound is smaller than the threshold, then all the records in the pair of subsets match. In both cases, the private linkage is pruned and efficiency is thus improved. For the remaining pairs of subsets after pruning, SMC is then applied to find matching records.

The hybrid approach reveals two types of information. The first is the subset synopses revealed directly to boost linkage efficiency as described above. The second

is about the non-matching records, which is inferred after linkage as explained below. Differential privacy is used in [9] for protecting privacy. But the following analysis shows that the second type of information does not satisfy differential privacy.

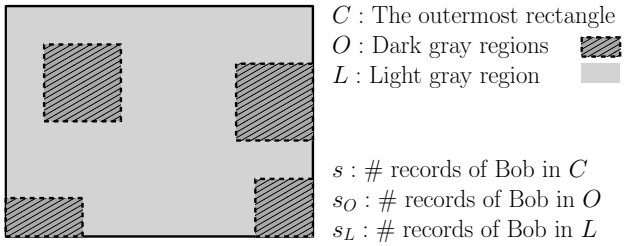


Figure 3.1.: An Example of the Hybrid Approach

Without loss of generality, consider a subset of Bob c. Assume that the solid line (outermost) rectangle in Figure 3.1 is the extent of the subset, and its size is randomized as

$$\widetilde{s} = s + \text{rpi}((\mathsf{Lap}\,(\lambda))), \tag{3.5}$$

where s is the true size of c. This is the first type of information (i.e., private synopsis) revealed.

Let $\widetilde{s_O}$ be the number of matching records (that are not suppressed) in Bob's subset c. These records are sent to Alice at the end of the protocol. With the matching records and also the private synopsis of c, Alice is able to infer information about the non-matching records in c. In particular, for each record x (that is not suppressed) in Alice's dataset, she calculates a *matching region*[1] according to the decision rule, such that Bob's records in the region match x. Suppose that the dark gray rectangles in Figure 3.1 are the overlapping areas between the extent of c and the matching regions of Alice's records. Denote the extent of c by C, the overlapping areas by O, and the non-overlapping areas by $L = C - O$. Then, Bob's records in O are matching records, and those in L are non-matching. Alice easily computes the noisy number of non-matching records in L as $\widetilde{s_L} = \widetilde{s} - \widetilde{s_O}$.

[1]The matching region in [9] is a rectangle, since distance threshold is specified on each attribute.

The value of $\widetilde{s_L}$ should satisfy differential privacy to guarantee the privacy of non-matching records. But this is not always the case. Let s_O be the true number of records of Bob in O, and s_L be that in L. Thus, $s = s_O + s_L$. The noise $\mathsf{Lap}(\lambda)$ in Equation 3.5 can be positive or negative. Consider the negative case, where Bob selects $-\text{rpi}(\mathsf{Lap}(\lambda))$ records in c uniformly at random for suppression. Thus, on average $[s_O/(s_O + s_L)](-\text{rpi}(\mathsf{Lap}(\lambda)))$ of Bob's matching records will be suppressed. More specifically,

$$\widetilde{s_O} = s_O + \frac{s_O}{s_O + s_L} \cdot (\text{rpi}(\mathsf{Lap}(\lambda))), \tag{3.6}$$

$$\widetilde{s_L} = s_L + \frac{s_L}{s_O + s_L} \cdot (\text{rpi}(\mathsf{Lap}(\lambda))). \tag{3.7}$$

Clearly, $\widetilde{s_L}$ satisfies ϵ-differential privacy if $s_O = 0$, that is, Bob has no matching record in the subset c. However, in the case when $s_O > 0$, it is impossible for Bob to distinguish matching records from those non-matching ones in advance. As a consequence, the magnitude of the added negative noise in $\widetilde{s_L}$ as suggested in [9] is insufficient to ensure ϵ-differential privacy[2].

In the above we have theoretically proven that the hybrid approach [9] does not satisfy differential privacy. In the following we illustrate this by an example.

Example 3.2.1 *Suppose that Alice and Bob would like to privately link their respective datasets, for which "Age" and "Weight" are the linking attributes. Suppose that the two datasets are partitioned as in Figure 3.2. Furthermore, suppose that Laplace noises are added to subsets as shown in Table 3.1(b) and Table 3.2(b), and differentially private synopses are generated accordingly as in Table 3.1(a) and Table 3.2(a). Take the synopsis of* c_{21} *for example. The extent is* $[40, 50)$ *on the dimension of "Age" and* $[200, 300)$ *on the dimension of "Weight", and the noisy subset size is* 4 *because negative Laplace noise* -1 *is added to the true size* 5*. The data partitioning improves*

[2]We emphasize that the definition of differential privacy is indistinguishability-based. It does not matter that Alice as an adversary does not know the matching records beforehand. Independent of any background knowledge Alice may possess, differential privacy requires that the disclosed number of non-matching records should satisfy Inequality 2.4 for any pair of Bob's *neighboring* datasets differing in only one *non-matching* record.

the efficiency of private linkage, since it requires record matching only between records in 'nearby' subsets. Consider c_{21} again. Without the data partitioning, records in c_{21} need to be compared with all the records of Alice. With the synopses provided, they only need to be matched with records in the subsets close to c_{21}.

Assume that based on the decision rule, records in c_{21} only need to be compared with records in c_{11}, c_{12}, c_{14}, and c_{15}. Also assume that the true matching record pairs for c_{21} are $(x_1, y_1), (x_2, y_3), (x_4, y_4)$. However, Laplace noises are added–positive noises introduce dummy records that do not match any input record, and negative noises suppress real records. Assume that y_3 is randomly selected and suppressed because of the negative noise -1 added to c_{21}. Consequently, Alice and Bob only get matching record pairs (x_1, y_1) and (x_4, y_4). With the matching records and the private synopsis of c_{21} (Table 3.2(a)), Alice could infer knowledge about the non-matching records in c_{21}. Based on the decision rule, she first develops a region for each of her non-suppressed input records in c_{11}, c_{12}, c_{14}, and c_{15}, such that all of Bob's records in these regions match. Suppose that the overlap between these regions and c_{21} is as shown in Figure 3.3. Alice is able to infer the number of non-matching records in $L_1 = c_{21} - R_1$, where R_1 is the union of the matching regions of Alice's non-suppressed input records and L_1 contains y_2 and y_5. This statistic (i.e., 2) about L_1 should be protected by Laplace noise as well. However, in this case, the noise is absorbed by y_3, a matching but suppressed record. As a consequence, no noise is added to L_1, and its statistic does not satisfy differential privacy.

The example above clearly demonstrates that the approach in [9] cannot guarantee adequate negative Laplace noises being added to non-matching records to satisfy differential privacy. Moreover, He et al. [11] carry out an analysis and prove that Inan et al.'s approach violates the notion of $(\epsilon, \delta, f_{\mathsf{Rule}})$-DPRL as described in Section 3.1.2.

Summary. The above problem happens, because each party (i.e., Alice and Bob) not only releases the synopses of its data but also exchanges the matching records. When the synopses and the matching records are combined, each party infers information

about the non-matching records, and thus violates differential privacy. Mathematically, the approaches [9, 13] release actually 3 queries (i.e., Equations 3.5, 3.6, and 3.7), although they directly only release a single one (i.e., Equation 3.5). The directly released query is properly randomized. The other two are not, since before private linkage neither party knows which records are matching and which are not. This is the root cause of privacy violation.

Table 3.1.: Subsets of Alice

(a) Differentially private synopses

Subset	Age	Weight	Noisy Count
c_{11}	$[40, 50)$	$[200, 300)$	2
c_{12}	$[50, 60)$	$[200, 300)$	2
c_{13}	$[60, 70)$	$[200, 300)$	4
c_{14}	$[40, 50)$	$[100, 200)$	3
c_{15}	$[50, 60)$	$[100, 200)$	1
c_{16}	$[60, 70)$	$[100, 200)$	1
c_{17}	$[40, 50)$	$[0, 100)$	1
c_{18}	$[50, 60)$	$[0, 100)$	3
c_{19}	$[60, 70)$	$[0, 100)$	1

(b) True Count / Noise

True Count	Noise
2	0
1	1
3	1
3	0
1	0
2	-1
2	-1
2	1
2	-1

Table 3.2.: Subsets of Bob

(a) Differentially private synopses

Subset	Age	Weight	Noisy Count
c_{21}	$[40, 50)$	$[200, 300)$	4
c_{22}	$[50, 60)$	$[200, 300)$	3
c_{23}	$[60, 70)$	$[200, 300)$	3
c_{24}	$[40, 50)$	$[100, 200)$	4
c_{25}	$[50, 60)$	$[100, 200)$	2
c_{26}	$[60, 70)$	$[100, 200)$	4
c_{27}	$[40, 50)$	$[0, 100)$	0
c_{28}	$[50, 60)$	$[0, 100)$	3
c_{29}	$[60, 70)$	$[0, 100)$	1

(b) True Count / Noise

True Count	Noise
5	-1
2	1
2	1
3	1
2	0
3	1
1	-1
2	1
2	-1

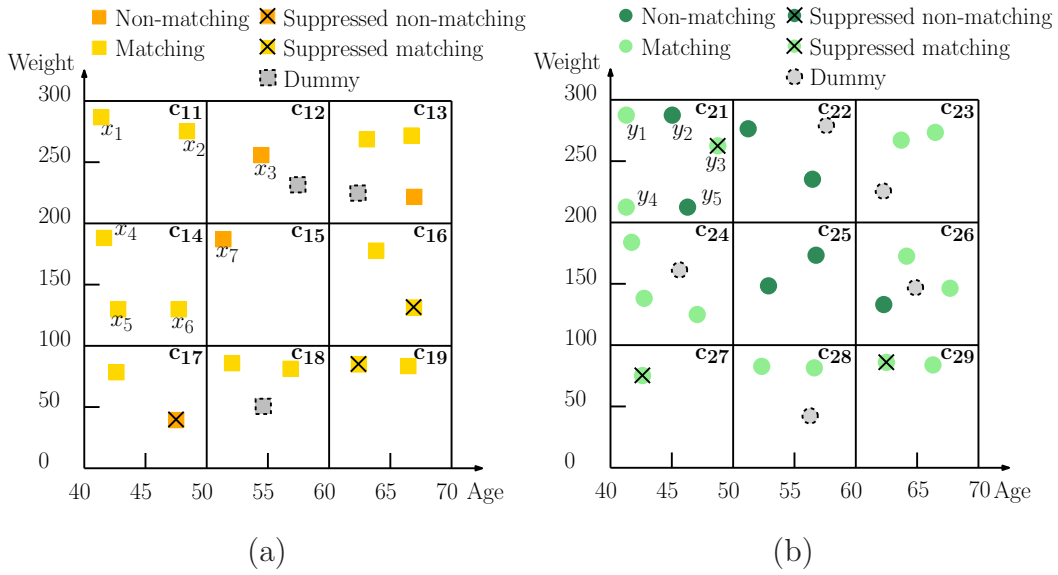


Figure 3.2.: Statistics Publication (An Example)

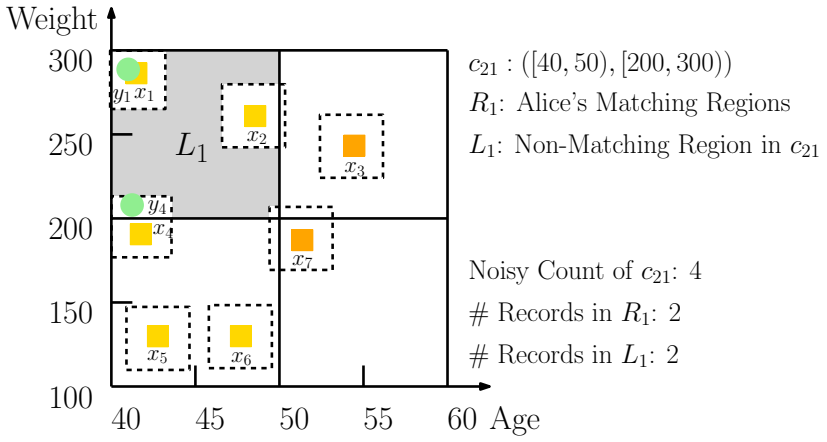


Figure 3.3.: Inference by Alice (An Example)

3.3 The Hybrid Scheme

Now we present the general framework of our hybrid scheme. For presentation clarity we will use the SSE (Inequality 3.2) decision rule as an example.

3.3.1 The Private Linkage Protocol

Our private linkage protocol consists of three parties. Alice and Bob are data owners. They will exchange matching records that have similar values with respect to linking attributes. Charlie is a third party, coordinating record matching between Alice and Bob. We assume that the three parties are *honest-but-curious* and model them as polynomial-time Turing machines for security analysis. That is, they strictly follow the protocol and will not collude with any other party, but are interested in inferring additional knowledge using polynomial-time computations.

In the protocol, Alice and Bob release to Charlie the synopses of their data that satisfy differential privacy for record linkage. According to the synopses, Charlie prunes the matching of pairs of records, which definitely mismatch. For all the remaining pairs that potentially match, Alice and Bob carry out the secure multi-party computation (SMC) to check if they really match. Alice and Bob exchange each pair of matching records secretly without informing Charlie. Our protocol thus separates the private synopses disclosure from matching records. Its input, output, and security requirements are as follows.

Protocol input from each party

- *Alice (Bob)*: Her (his) input records, the differentially private synopses for record linkage of her (his) records, and her (his) secret key for the cryptosystem.
- *Charlie*: None.

Protocol output to each party

- *Alice (Bob)*: The matching records, and the total number of pairs of records being securely compared in the protocol conforming to differential privacy for record linkage (DPRL).
- Charlie: Private synopses conforming to DPRL from Alice and Bob.

Security requirements

- *Alice (Bob)*: She (he) learns no additional information except for the protocol output to her (him).

- *Charlie*: At the end of the protocol, Charlie's knowledge about Alice's and Bob's records remains limited to the differentially private synopses released to him.

Figure 3.4 gives an overview of the protocol. It consists of two phases: *pruning* in Section 3.3.1.1 and *record matching via SMC* in Section 3.3.1.2.

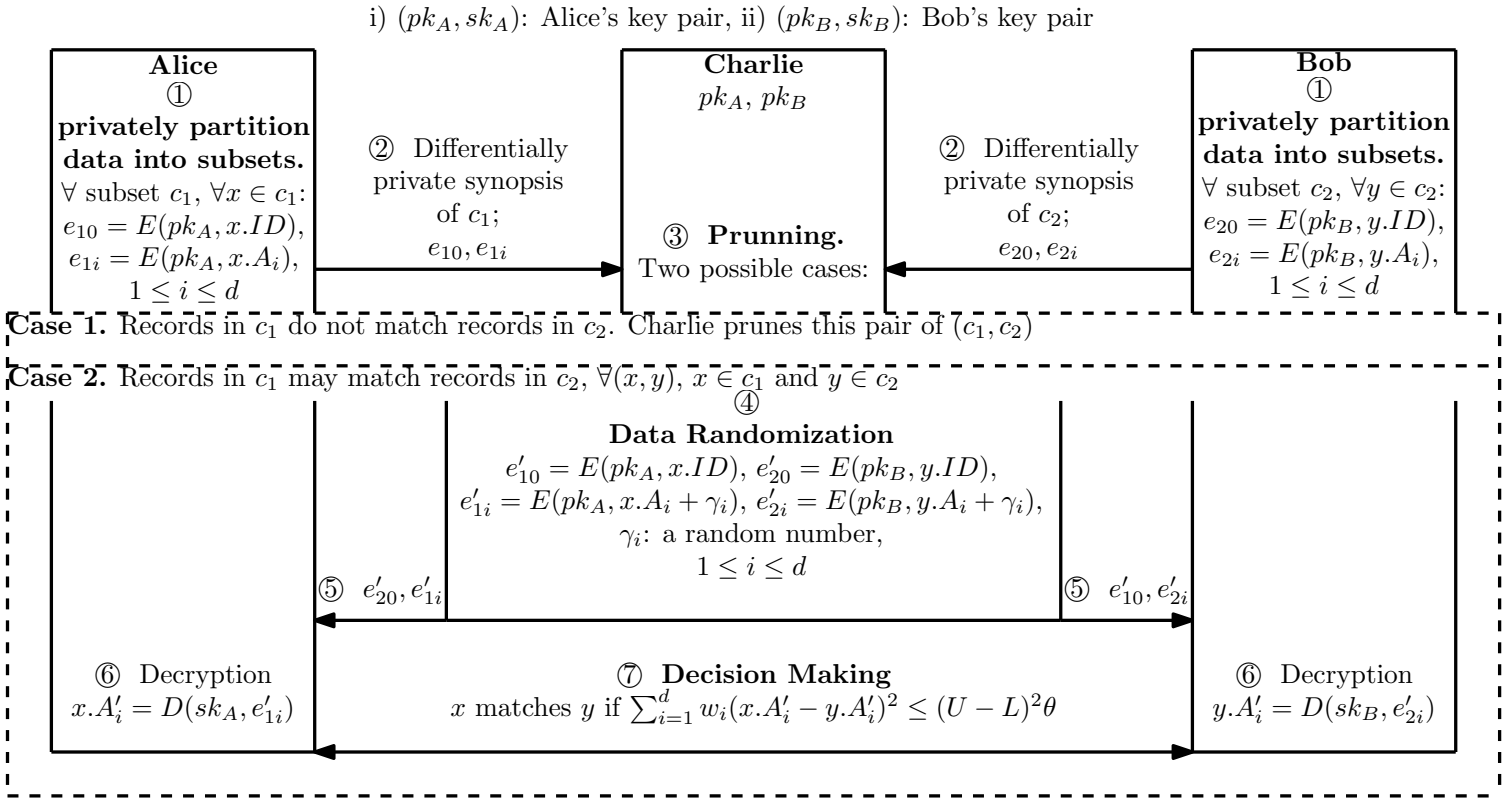


Figure 3.4.: The Private Linkage Protocol

3.3.1.1 Pruning

Alice and Bob prepare their protocol inputs before the pruning phase. The preparation process is the same for Alice and Bob. Alice (Bob) partitions her (his) data into non-overlapping subsets (details in Section 3.4). For each subset, she (he) generates a private synopsis, and adjusts the subset accordingly by adding dummy records as we discuss in detail in Section 3.4.2.

After the input is ready, the pruning phase starts. It consists of two steps. First, Alice and Bob encrypt their records, and send to Charlie the encrypted records as well as the private synopses. Second, Charlie prunes the record matching based on the received synopses and then reveals to both Alice and Bob the required number of potentially matching record pairs, i.e., the number of required secure comparisons. To facilitate the analysis, we will also use Π_1 to represent the the first phase of our protocol involving the steps described as follows in this subsection.

Record encryption. Alice uses Paillier cryptosystem [6] to encrypt her records. She randomly selects two large primes p_A and q_A to form the modulus $n_A = p_A \cdot q_A$, and follows the parameter configuration in [6] to generate a public-private key pair (pk_A, sk_A). Let c_1 be a subset of her data, and $x \in c_1$ be a record. She computes

$$\begin{cases} e_{10} &= E(pk_A, x.ID) \\ e_{1i} &= E(pk_A, x.A_i), \end{cases}$$

where $x.ID$ is the record ID of x, $x.A_i$ is the value of the i-th linking attribute of x, and $i = 1, 2, \ldots, d$.

Bob encrypts his data in the same way as Alice. He also generates a Paillier key pair (pk_B, sk_B) with the modulus n_B being the product of two large primes, and encrypts a record y in a subset c_2 of his data

$$\begin{cases} e_{20} &= E(pk_B, y.ID) \\ e_{2i} &= E(pk_B, y.A_i), \end{cases}$$

where $y.ID$ is the ID of y and $i = 1, 2, \ldots, d$.

Matching pruning by SSE. Alice and Bob send to Charlie the private synopses of subsets and the encrypted records. Let c_1 and c_2 be two subsets of Alice and Bob,

respectively. Charlie applies the decision rule to determine whether c_1 and c_2 may have matching records. Let $\{b_{1i} \mid 1 \leq i \leq d\}$ be the extent of c_1, where b_{1i} denotes the interval $[A_{1i}^{\ell}, A_{1i}^{u}]$, and A_{1i}^{ℓ} and A_{1i}^{u} are the lower and upper bounds of c_1 along the i-th attribute, respectively. Similarly, suppose that $\{b_{2i} \mid 1 \leq i \leq d\}$ is the extent of c_2, where b_{2i} denotes the interval $[A_{2i}^{\ell}, A_{2i}^{u}]$. Then, for each attribute Charlie computes

$$\delta_i^{\ell} = \min \left\{ \left(\frac{u - v}{U - L} \right)^2 \;\middle|\; u \in b_{1i} \wedge v \in b_{2i} \right\},$$

where U and L are the upper and lower bounds of all the attributes. According to the SSE (i.e., Inequality 3.2) decision rule, Charlie is thus able to calculate the lower bound of the distance between any pair of records from the two subsets

$$\theta^{\ell} = \sum_{i=1}^{d} w_i \cdot \delta_i^{\ell}.$$

Charlie classifies the private record linkage into two cases. *Case 1*: $\theta^{\ell} > \theta$. Records in c_1 definitely do not match those in c_2. Private linkage between c_1 and c_2 is pruned. *Case 2*: $\theta^{\ell} \leq \theta$. Records in c_1 may match those in c_2. After identifying each pair (c_1, c_2) of subsets that may contain matching record pairs, Charlie computes and then reveals to both data owners the total number of potentially matching record pairs, which is the total number of secure comparisons that have to be carried out. Alice and Bob securely compare records in c_1 and c_2 (see below).

Time complexity. Let d be the number of linking attributes, and m_A and m_B the number of subsets from Alice and Bob, respectively. The time cost of matching pruning by Charlie is $O(d \cdot m_A \cdot m_B)$. Both Alice and Bob need to encrypt records before sending them to Charlie. Each Paillier encryption takes 1 modular exponentiation, when the encryption is simplified to $(1 + nm) \cdot r^n \mod n^2$, where m is the message in $\mathbb{Z}_n = \{x \in \mathbb{Z} \mid 0 \leq x \leq n - 1\}$ and r is a random number in $\mathbb{Z}_n^* = \{x \in \mathbb{Z} \mid 0 \leq x \leq n - 1 \text{ and } \gcd(x, n) = 1\}$. Hence, for each record of Alice/Bob, encryption takes her/him $(d + 1)$ modular exponentiations.

3.3.1.2 Record Matching via SMC

The record matching decides whether two records match or not. It consists of two steps. First, Charlie randomizes the two records, while keeping the fact (i.e., whether they match or not) unchanged. Second, Alice and Bob carry out a 2-party computation protocol to securely decide whether the two records match. For ease of analysis, we will also use Π_2 to represent the second phase of our protocol involving the steps detailed as follows in this subsection.

Data randomization. Let $x \in c_1$ and $y \in c_2$ be two records of Alice and Bob, respectively. Charlie randomizes the ciphertexts of them. Let κ be a statistical security parameter and $\tau = \kappa + \lceil \log_2(2U) \rceil$, such that 2^τ is much larger than $2U$ but much smaller than $\min\{n_A, n_B\}$, where n_A (n_B) is the modulus of Alice's (Bob's) Paillier cryptosystem. Charlie uniformly selects a random value γ_i from $\{U, U + 1, \cdots, U + 2^\tau - 1\}$ and computes

$$\begin{cases} e'_{1i} &= e_{1i} \cdot E(pk_A, \gamma_i) &= E(pk_A, x.A_i + \gamma_i) \\ e'_{2i} &= e_{2i} \cdot E(pk_B, \gamma_i) &= E(pk_B, y.A_i + \gamma_i), \end{cases}$$

where $i = 1, 2, \ldots, d$. Charlie also re-encrypts the IDs of x and y

$$\begin{cases} e'_{10} &= e_{10} \cdot E(pk_A, 0) \\ e'_{20} &= e_{20} \cdot E(pk_B, 0). \end{cases}$$

The above setting of κ ensures that $x.A'_i = x.A_i + \gamma_i$ and $y.A'_i = y.A_i + \gamma_i$ are much smaller than n_A and n_B, respectively. Therefore, given D the Paillier decryption function, $D(sk_A, e'_{1i}) = x.A'_i \mod n_A = x.A'_i$, and $D(sk_B, e'_{2i}) = y.A'_i \mod n_B = y.A'_i$. Therefore, the randomization does not change attribute distance. That is, $\mathsf{Dist}_i(x.A_i, y.A_i) = \mathsf{Dist}_i(x.A'_i, y.A'_i)$. Charlie switches the IDs of the two records and sends back the randomized ciphertexts, i.e., sending $(e'_{20}, e'_{11}, \ldots, e'_{1d})$ to Alice and $(e'_{10}, e'_{21}, \ldots, e'_{2d})$ to Bob.

Lemma 3.3.1 *Let $(e'_{20}, e'_{11}, \ldots, e'_{1d})$ be the randomized ciphertext sent to Alice from Charlie. Then, Alice cannot distinguish it from $(z_{20}, z_{11}, \ldots, z_{1d})$* [3]*, where $z_{20} = E(pk_B, b)$, $z_{1i} = E(pk_A, a_i)$ for $1 \leq i \leq d$, and both b and a_i are random values in $\{U, U+1, \cdots, U+2^\tau - 1\}$.*

Proof In the randomized ciphertext, e'_{20} is encrypted using Bob's public key. Alice cannot decrypt it. She cannot distinguish it from z_{20} (the ciphertext of a random value) either, since Paillier cryptosystem is semantically secure. She decrypts e'_{1i} to get $x.A_i + \gamma_i$ for $1 \leq i \leq d$. Let $D^x = \{-U, 0, 1, \ldots, U\}$ be the domain of $x.A_i$, where $-U$ is included because dummy records with attribute value $-U$ are added. Clearly, both a_i and $x.A_i + \gamma_i$ are in the domain of $[0,\ 2U + 2^\tau - 1]$. By Definition 2.1.2, the statistical distance between $x.A_i + \gamma_i$ and a_i is

$$\frac{1}{2} \sum_{v=0}^{2U+2^\tau-1} |\Pr[a_i = v] - \Pr[x.A_i + \gamma_i = v]|,$$

where 0 and $2U + 2^\tau - 1$ are the lower and upper bounds of $x.A_i + \gamma_i$, respectively. For any v, we have

$$\Pr[x.A_i + \gamma_i = v] = \sum_{\ell \in D^x} \Pr[x.A_i = \ell] \Pr[\gamma_i = v - \ell] \leq \frac{1}{2^\tau} \sum_{\ell \in D^x} \Pr[x.A_i = \ell] = \frac{1}{2^\tau}.$$

If $2U \leq v \leq 2^\tau - 1$, then $\Pr[x.A_i + \gamma_i = v] = \frac{1}{2^\tau}$. Based on the setting of a_i, it follows that $\Pr[a_i = v] = \frac{1}{2^\tau}$ if $U \leq v \leq U + 2^\tau - 1$ and $\Pr[a_i = v] = 0$ otherwise. Therefore,

$$\begin{aligned}
&\frac{1}{2} \sum_{v=0}^{2U+2^\tau-1} |\Pr[a_i = v] - \Pr[x.A_i + \gamma_i = v]| \\
&\leq \frac{1}{2} \left(\sum_{v=0}^{U-1} \left| 0 - \frac{1}{2^\tau} \right| + \sum_{v=U}^{2U-1} \left| \frac{1}{2^\tau} - 0 \right| + \sum_{v=2U}^{2^\tau-1} \left| \frac{1}{2^\tau} - \frac{1}{2^\tau} \right| \right. \\
&\quad \left. + \sum_{v=2^\tau}^{U+2^\tau-1} \left| \frac{1}{2^\tau} - 0 \right| + \sum_{v=U+2^\tau}^{2U+2^\tau-1} \left| 0 - \frac{1}{2^\tau} \right| \right) \leq \frac{1}{2^\kappa}.
\end{aligned}$$

[3] Similarly, we can prove that $(e'_{10}, e'_{21}, \ldots, e'_{2d})$, the randomized ciphertext sent to Bob from Charlie, is indistinguishable from a ciphertext of randomly selected values. Since the proof is essentially the same as that for Lemma 3.3.1, it is omitted.

Set $\pi = \lfloor \log_2(2U + 2^\tau - 1) \rfloor$, and we can verify that $\frac{1}{2^\kappa} \leq \frac{8U}{2^\pi} \leq \frac{1}{p(\pi)}$, where $p(\pi)$ is any positive polynomial with sufficiently large π. Therefore, e'_{1i} is statistically indistinguishable from z_{1i}, which in turn implies that e'_{1i} is computationally indistinguishable from z_{1i} [3]. ■

The data randomization hides the connection between the input records and their corresponding randomized ciphertexts. Let χ be the set of Alice's records with cardinality greater than 1. Then, in the case where a pair of randomized records does not match, Alice is unable to tell which $x \in \chi$ is involved in this matching. Furthermore, in order to make sure that Alice (Bob) cannot tell exactly the subset to which x (y) belongs, Charlie shuffles the sequence of pairs of potentially matching records[4].

2-party decision making by SSE. Alice and Bob decide if x matches y. They decrypt the randomized ciphertexts. Alice gets $x.A'_i = x.A_i + \gamma_i$ and Bob gets $y.A'_i = y.A_i + \gamma_i$ for $i = 1, 2, \ldots, d$. The randomization does not change the attribute distance. Thus, according to the instantiated decision rule in Equation 3.2, x and y match if $\Delta \leq \Theta$, where

$$\begin{cases} \Theta &= (U - L)^2\theta \\ \Delta &= \sum_{i=1}^{d} w_i (x.A'_i - y.A'_i)^2. \end{cases}$$

A straightforward solution is for Alice and Bob to exchange $x.A'_i$ and $y.A'_i$. However, this solution discloses the real attribute distance. In the case that x and y do not match, such additional information leak is not desirable.

We develop a more sophisticated solution. Let ρ be a random number only known to Bob. Our solution ensures that Alice knows $\Delta + \rho$ but does not learn Δ, and Bob knows ρ but does not learn $\Delta + \rho$. Then, Bob computes $\Theta + \rho$, and securely compares it with Alice's $\Delta + \rho$. The two records x and y match, if and only if $\Delta + \rho \leq \Theta + \rho$.

The details of the solution are as follows. Let V^Δ be the upper bound of Δ, κ the security parameter as set for data randomization, and $\psi = \lceil \kappa + \log_2(V^\Delta) \rceil$,

[4]We would like to point out that in the case when Charlie colludes with one of the data owners, e.g., Alice, the privacy guarantee provided by our framework would be the same as that in [11], since in this case Charlie could inform Alice of the noisy counts of Bob's subsets so that in the end of record linkage, Alice learns the noisy number of non-matching records in each of Bob's subsets.

such that both $2^\psi + \Theta$ and $2^\psi + \Delta$ are much smaller than the modulus (i.e., n_A) of Alice's Paillier cryptosystem. Alice first sends to Bob $f_1 = E(pk_A, \sum_{i=1}^{d} w_i \cdot x.A_i'^2)$ and $f_{2i} = E(pk_A, -2w_i \cdot x.A_i')$ for $1 \leq i \leq d$. Bob then computes an encryption of $\Delta + \rho$:

$$E(pk_A, \Delta + \rho) = f_1 \cdot \prod_{i=1}^{d} f_{2i}^{y.A_i'} \cdot E\left(pk_A, \left(\sum_{i=1}^{d} w_i \cdot y.A_i'^2\right) + \rho\right),$$

where $\rho \in \{0, 1, \cdots, 2^\psi - 1\}$ is a random number chosen by Bob and kept secret from Alice. Bob sends to Alice the ciphertext. Alice decrypts it and gets $D(sk_A, E(pk_A, \Delta + \rho)) = \Delta + \rho \bmod n_A$. Since $2^\psi + \Delta$ is much smaller than n_A, it follows that $\Delta + \rho \bmod n_A = \Delta + \rho$. Meanwhile, Bob computes $\Theta + \rho$.

The two parties then apply a secure integer comparison protocol (Section 2.1) to compare $\Delta+\rho$ and $\Theta+\rho$. If $\Delta+\rho \leq \Theta+\rho$, i.e., the record pair matches, Alice and Bob exchange their encrypted IDs (i.e., e'_{10} and e'_{20}). They then use the decrypted IDs to trace the original pair (x, y) and exchange them. Otherwise, the record pair does not match. The next lemma shows that Alice and Bob learn no additional information than the output.

Lemma 3.3.2 *In the 2-party decision making, Alice (Bob) only learns the protocol output to her(him).*

Proof We first formalize the output of the 2-party decision making. Let $x' = (x.A'_1, x.A'_2, \cdots, x.A'_d)$ and $y' = (y.A'_1, y.A'_2, \cdots, y.A'_d)$ be a pair of records randomized by Charlie, and $R \in \{0, 1\}$ be their matching result. If x' matches y', the protocol's output to Alice/Bob is $(x, y, R = 1)$ and $(R = 0)$ otherwise, where x and y are the input records of x' and y', respectively.

We now use simulation (Section 2.1) to prove the lemma. We will show that the messages each party receives from the other party within the protocol are *indistinguishable* from messages randomly drawn from a uniform distribution. Thus, intuitively each party only learns the protocol output to her/him. In the following, we first build a simulator $S_A^{DM}(x', O)$ for Alice, where O represents the protocol output

to her. Let $V_A^1, V_A^2, \ldots, V_A^{t_1}$ be the sequence of messages she receives from Bob when matching x' and y', where $V_A^1 = \Delta + \rho$ and V_A^i $(2 \leq i \leq t_1)$ are the messages she receives for securely comparing $\Delta + \rho$ with $\Theta + \rho$. To simulate V_A^1, simulator S_A^{DM} randomly selects $\widetilde{V_A^1} \in \{0, 1, \cdots, 2^\psi - 1\}$. In a similar way as the proof of Lemma 3.3.1, we can prove that $\widetilde{V_A^1}$ is statistically indistinguishable from V_A^1. The work [14] has already built a simulator S_A^{IC} to simulate the messages Alice receives from Bob for securely comparing $\Delta+\rho$ with $\Theta+\rho$. Simulator S_A^{DM} calls $S_A^{IC}(\widetilde{V_A^1}, R)$ to simulate all V_A^i's for $2 \leq i \leq t_1$.

Next, we build a simulator $S_B^{DM}(y', O)$ for Bob, where O is the same as that to Alice. Let $V_B^1, V_B^2, \ldots, V_B^{t_2}$ be the sequence of messages Bob receives from Alice when matching x' and y', where $V_B^1 = (f_1, f_{21}, f_{22}, \ldots, f_{2d})$, and V_B^j for $2 \leq j \leq t_2$ are messages he receives from Alice for securely comparing $\Delta+\rho$ with $\Theta+\rho$. To simulate V_B^1, simulator S_B^{DM} selects random values r_j's for $j = 1, 2, \ldots, d+1$, and generates $\widetilde{f_1} = E(pk_A, r_1)$ and $\widetilde{f_{2i}} = E(pk_A, r_{i+1})$ for $i = 1, 2, \ldots, d$. By the semantic security of Paillier cryptosystem, it follows that $\widetilde{V_B^1} = (\widetilde{f_1}, \widetilde{f_{21}}, \widetilde{f_{22}}, \ldots, \widetilde{f_{2d}})$ is computationally indistinguishable from V_B^1. The work [14] has also built a simulator S_B^{IC} to simulate the messages Bob receives from Alice for comparing $\Delta + \rho$ with $\Theta + \rho$. Simulator S_B^{DM} randomly selects $\widetilde{\rho} \in \{0, 1, \cdots, 2^\psi - 1\}$, and calls $S_B^{IC}(\Theta + \widetilde{\rho}, R)$ to simulate V_B^j for $2 \leq j \leq t_2$.

■

Complexity and Comparison with the 2-Party Protocol. We analyze both the computational and communication complexity of comparing a pair of records. The matching has two steps. *Step 1: Data randomization.* Charlie randomizes the ciphertexts generated by Alice and Bob. It requires in total $2(d+1)$ modular exponentiations. *Step 2: 2-party decision making.* Alice (Bob) decrypts the randomized record attribute values, which takes d modular exponentiations. Then, Alice and Bob jointly compute an encryption of $\Delta + \rho$, which requires $2d + 2$ exponentiations. Alice needs an additional exponentiation to obtain $\Delta + \rho$. Hence, $6d + 5$ modular exponentiations are needed, before the secure integer comparison [5] is applied. Let

$\mathbb{Z}_{n^2}$ denote the computational cost of an exponentiation in $\mathbb{Z}_{n^2}$ with the exponent being a $|\mathbb{Z}_n|$-bit long integer, where $|\mathbb{Z}_n|$ denotes the bit-length of an element in $\mathbb{Z}_n$. Furthermore, we use Γ_A and Γ_B to represent the computational cost of an AND gate incurred on Alice and Bob, respectively. On the other hand, we use k to denote the bit length of each garbled wire label in the garbled circuit. Since $3k(\psi+1)$ bits are required for transmitting the input wire labels and the garbled truth table of an AND gate consists of 4 entries, in total $7k(\psi+1)$ bits have to be transferred. Recall that in our protocol, $\tau = \lceil \kappa + \log_2(2U) \rceil$, $\psi = \lceil \kappa + \log_2(V^{\Delta}) \rceil$, where U and V^{Δ} denote the upper bounds on a scaled attribute value and the distance between any pair of records. In Table 3.3 and Table 3.4, we report the detailed computational and communication overheads incurred by our 3-party protocol. We note that the computational overheads on Bob are lower than Alice, due to the fact that the bit-length τ of Bob's randomized attribute $y.A'_i$ is much smaller than $|\mathbb{Z}_n|$ and that the computational complexity of exponentiations grows linearly in the bit-length of the exponent.

For completeness, we also compare our 3-party protocol with the 2-party protocol, where each data owner agrees to release to the other party the noisy number of non-matching records within each subset. In this case, the third party Charlie is not required. According to Table 3.3, at least half of the exponentiations could be saved in the 2-party protocol per record pair, since Charlie does not have to perform the record randomization and hence the data owners do not have to decrypt the corresponding ciphertexts. More than half of the communication cost incurred to transmit the Paillier ciphertexts could thus be saved. On the other hand, the computational and communication costs due to Yao's garbled circuits are unchanged, because the data randomization performed by Charlie does not alter the distance between any pair of records with respect to the decision rule SSE.

Table 3.3.: Computational Cost of Each Party for Each Compared Record Pair

Protocol	**Charlie**	**Alice**	**Bob**
3-Party (Ours)	$(2d+2)\mathbb{Z}_{n^2}$	$(2d+2)\mathbb{Z}_{n^2} + (\psi+1)\Gamma_A$	$\left(\left(1+\frac{\tau}{\lvert\mathbb{Z}_n\rvert}\right)d+1\right)\mathbb{Z}_{n^2} + (\psi+1)\Gamma_B$
2-Party	N/A	$(d+2)\mathbb{Z}_{n^2} + (\psi+1)\Gamma_A$	$\left(\left(\frac{\lceil\log_2(2U)\rceil}{\lvert\mathbb{Z}_n\rvert}\right)d+1\right)\mathbb{Z}_{n^2} + (\psi+1)\Gamma_B$

Table 3.4.: Communication Cost for Each Compared Record Pair

Protocol	**Communication Cost**
3-Party (Ours)	$(3d+4)\lvert\mathbb{Z}_{n^2}\rvert + 7k(\psi+1)$
2-Party	$(d+2)\lvert\mathbb{Z}_{n^2}\rvert + 7k(\psi+1)$

3.3.2 The Protocol Analysis

This section is to verify the goals set at the beginning of Section 3.3.1. Namely, each party acquires his/her respective output from the protocol, while at the same time, all the security requirements are met.

Theorem 3.3.1 (Correctness) *For each pair of records being securely matched by Alice and Bob, the decision rule is correctly evaluated.*

Proof The correctness of the theorem has already been clarified in the protocol. Here, we give a brief summarization. Let x and y be two records of Alice and Bob, respectively. Then, according to the decision rule (Definition 3.1.1 and Equations 3.1 and 3.2), x and y match iff $\sum_{i=1}^{d} w_i(x.A_i - y.A_i)^2 \leq \Theta$, where $\Theta = (U - L)^2\theta$. In the protocol, Charlie randomizes $x.A_i$ and $y.A_i$ such that $x.A'_i = x.A_i + \gamma_i$ and $y.A'_i = y.A_i + \gamma_i$, respectively, for $i = 1, 2, \ldots, d$. Alice and Bob then determine x and y match iff $\Delta + \rho \leq \Theta + \rho$, where $\Delta = \sum_{i=1}^{d} w_i(x.A'_i - y.A'_i)^2$. Clearly, $\Delta + \rho \leq \Theta + \rho$ holds iff $\sum_{i=1}^{d} w_i(x.A_i - y.A_i)^2 \leq \Theta$. This concludes the proof. ■

We now prove that as long as the view of Alice after the first phase (Step 1 through Step 3 in Figure 3.4) Π_1 of our protocol does not vary too much, $(\epsilon, \delta, f_{\mathsf{Rule}})$-DPRL holds after the execution of the second phase (Step 5 through Step 7 in Figure 3.4) Π_2. That is, we prove that for each dataset owner, her/his view after the execution of Π_2 following Π_1 satisfies $(\epsilon, \delta, f_{\mathsf{Rule}})$-DPRL no matter which one of the neighboring dataset is used as input by the other dataset owner. To facilitate the proof, we first prove that Alice's view after the execution of Π_2 will not change too much for any pair of Bob's neighboring datasets . More precisely, we have the following lemma.

Lemma 3.3.3 *Let D_B and D'_B be a given pair of Bob's neighboring datasets for record linkage such that after the first phase, both of them result in ν potential matching record pairs (including those pairs induced by dummy records), where ν is upper bounded by a polynomial in π. Then, for any polynomial time distinguisher G, any positive polynomial $p(\pi)$, and any sufficiently large π, we have*

$$\Pr[G(VIEW_A^{\Pi_2}(\nu, D_A, D_B)) = 1] \leq \Pr[G(VIEW_A^{\Pi_2}(\nu, D_A, D'_B)) = 1] + \frac{1}{p(\pi)}, \quad (3.8)$$

where $VIEW_A^{\Pi_2}(\nu, D_A, D_B)$ *denotes Alice's view during the execution of* Π_2 *on some fixed input* ν*,* D_A*, and* D_B*, and* $VIEW_A^{\Pi_2}(\nu, D_A, D'_B)$ *is defined similarly.*

Proof According to Lemma 3.3.1 and Lemma 3.3.2, we are able to simulate Alice's view during the execution of Π_2 on input (ν, D_A, D_B) and (ν, D_A, D'_B) using $(D_A, \nu, f_{\mathsf{Rule}}(D_A, D_B))$ and $(D_A, \nu, f_{\mathsf{Rule}}(D_A, D'_B))$, respectively. Specifically, we are able to construct a simulator S_A to simulate Alice's view.

Consider the j-th pair of records (x'_j, y'_j) for $j = 1, 2, \ldots, \nu$. S_A first prepares a list $\mathcal{O}$ of matching results for each of those ν decision making processes between Alice and Bob. Specifically, for each matching record pair $(x, y) \in f_{\mathsf{Rule}}(D_A, D_B)$, S_A adds $(x, y, 1)$ to $\mathcal{O}$. A number of (0)'s are then added to $\mathcal{O}$ so that there are exactly ν elements in $\mathcal{O}$[5]. S_A now randomly permutes the elements on the list $\mathcal{O}$.

Let $V_j^1, V_j^2, \ldots, V_j^{t_3}$ be the sequence of messages Alice receives from Charlie and Bob for matching the j-th pair, where $V_j^1 = (e'_{j,20}, e'_{j,11}, \ldots, e'_{j,1d})$ is received from Charlie in data randomization and V_j^k for $2 \leq k \leq t_3$ is received from Bob in the 2-party decision making. To simulate V_j^1, the simulator randomly selects b_j and $a_{j,i}$ in $\{U, U+1, \cdots, U+2^\tau-1\}$, and generates $z_{j,20} = E(pk_B, b_j)$, $z_{j,1i} = E(pk_A, a_{j,i})$ for $1 \leq i \leq d$. According to Lemma 3.3.1, $\widetilde{V_j^1} = (z_{j,20}, z_{j,11}, \ldots, z_{j,1d})$ is indistinguishable from V_j^1. The simulator S_A also needs to simulate the messages Alice receives from Bob in the 2-party decision making. This is done by invoking the simulator S_A^{DM} we built in Lemma 3.3.2 on input $(\widetilde{x'_j}, O_j)$, where $\widetilde{x'_j} = (a_{j,1}, a_{j,2}, \ldots, a_{j,d})$ and O_j denotes the j-th element on the randomly permuted list $\mathcal{O}$.

From the construction of S_A described above, based on the hybrid argument [3], it thus holds that for any polynomial time distinguisher G, any positive polynomial $q(\pi)$, and any sufficiently large π,

$$\left|\Pr[G(S_A(D_A, \nu, f_{\mathsf{Rule}}(D_A, D_B))) = 1] - \Pr[G(\mathrm{VIEW}_A^{\Pi_2}(\nu, D_A, D_B)) = 1]\right| < \frac{1}{q(\pi)}.$$

Similarly,

$$\left|\Pr[G(S_A(D_A, \nu, f_{\mathsf{Rule}}(D_A, D'_B))) = 1] - \Pr[G(\mathrm{VIEW}_A^{\Pi_2}(\nu, D_A, D'_B)) = 1]\right| < \frac{1}{q(\pi)}.$$

[5]In our protocol, since both data owners only add dummy records, it is true that $\nu \geq |f_{\mathsf{Rule}}(D_A, D_B)|$

Since D_B and D'_B are neighboring datasets for record linkage, it holds that $f_{\mathsf{Rule}}(D_A, D_B) = f_{\mathsf{Rule}}(D_A, D'_B)$, implying that $\Pr[G(S_A(D_A, \nu, f_{\mathsf{Rule}}(D_A, D_B))) = 1]$ and $\Pr[G(S_A(D_A, \nu, f_{\mathsf{Rule}}(D_A, D'_B))) = 1]$ are identically distributed, which in turn implies

$$\left|\Pr[G(\mathrm{VIEW}_A^{\Pi_2}(\nu, D_A, D_B)) = 1] - \Pr[\Pr[G(\mathrm{VIEW}_A^{\Pi_2}(\nu, D_A, D'_B)) = 1]\right| < \frac{2}{q(\pi)}.$$

Hence, for any polynomial time distinguisher G, any positive polynomial $p(\pi)$, and any sufficiently large π, we have

$$\Pr[G(\mathrm{VIEW}_A^{\Pi_2}(\nu, D_A, D_B)) = 1] \leq \Pr[G(\mathrm{VIEW}_A^{\Pi_2}(\nu, D_A, D'_B)) = 1] + \frac{1}{p(\pi)}.$$

■

We are now ready to prove that for each data owner, her/his view after the execution of Π_1 followed by Π_2 satisfies $(\epsilon, \delta, f_{\mathsf{Rule}})$-DPRL as long as her/his view after the execution of Π_1, i.e., the number of required secure comparisons, does not change too much no matter which one of the neighboring datasets for record linkage is used by the other party. In the following we prove the case for Alice's view. The case for Bob is essentially the same and thus the proof is omitted.

Theorem 3.3.2 *During the execution of the first phase Π_1 (Step 1 through Step 3 in Figure 3.4) followed by the second phase Π_2 of our protocol (Step 4 through Step 7 in Figure 3.4), it holds that for any pair of neighboring datasets (D_B, D'_B) for record linkage, each probabilistic polynomial time distinguisher G, any positive polynomial $p(\pi)$, and any sufficiently large π,*

$$\Pr[G(\mathit{VIEW}_A^{\Pi_2,\Pi_1}(D_A, D_B)) = 1] \leq \exp(\epsilon)\Pr[G(\mathit{VIEW}_A^{\Pi_2,\Pi_1}(D_A, D'_B)) = 1] + \delta + \frac{1}{p(\pi)}.$$

as long as for any non-negative integer ν,

$$\Pr[\mathit{VIEW}_A^{\Pi_1}(D_A, D_B) = \nu] \leq \exp(\epsilon)\Pr[\mathit{VIEW}_A^{\Pi_1}(D_A, D'_B) = \nu] + \delta,$$

where $\mathit{VIEW}_A^{\Pi_1}(D_A, \cdot)$ denotes Alice's view during the first phase Π_1 of the protocol, i.e., the number of required secure comparisons revealed by Charlie.

Proof In the first phase Π_1 of our protocol, each data owner privately partitions her/his own dataset and sends the respective private synopses to the third party Charlie, which in turn computes the number ν of required secure comparisons. Charlie then notifies both Alice and Bob of the required number of secure comparisons and initializes the second phase Π_2 of the protocol. Let $\mathbb{Z}_0^+ = \{x \in \mathbb{Z} \mid x \geq 0\}$, i.e., the set of non-negative integers. For any pair of neighboring datasets (D_B, D'_B) for record linkage, any polynomial time distinguisher G, any positive polynomial $p(\pi)$, and any sufficiently large π, we therefore have

$$
\begin{aligned}
&\Pr[G(\mathrm{VIEW}_A^{\Pi_2,\Pi_1}(D_A, D_B)) = 1] \\
&= \sum_{\nu \in \mathbb{Z}_0^+} \Pr[G(\mathrm{VIEW}_A^{\Pi_2}(\nu, D_A, D_B)) = 1] \Pr[\mathrm{VIEW}_A^{\Pi_1}(D_A, D_B) = \nu] \\
&\leq \sum_{\nu \in \mathbb{Z}_0^+} \left(\Pr[G(\mathrm{VIEW}_A^{\Pi_2}(\nu, D_A, D'_B)) = 1] + \frac{1}{p(\pi)} \right) \Pr[\mathrm{VIEW}_A^{\Pi_1}(D_A, D_B) = \nu] \\
&= \left(\sum_{\nu \in \mathbb{Z}_0^+} \Pr[G(\mathrm{VIEW}_A^{\Pi_2}(\nu, D_A, D'_B)) = 1] \Pr[\mathrm{VIEW}_A^{\Pi_1}(D_A, D_B) = \nu] \right) + \frac{1}{p(\pi)} \\
&\leq \left[\sum_{\nu \in \mathbb{Z}_0^+} \Pr[G(\mathrm{VIEW}_A^{\Pi_2}(\nu, D_A, D'_B)) = 1] \left(\exp(\epsilon) \Pr[\mathrm{VIEW}_A^{\Pi_1}(D_A, D'_B) = \nu] + \delta \right) \right] \\
&\quad + \frac{1}{p(\pi)} \\
&\leq \left(\sum_{\nu \in \mathbb{Z}_0^+} \exp(\epsilon) \Pr[G(\mathrm{VIEW}_A^{\Pi_2}(\nu, D_A, D'_B)) = 1] \Pr[\mathrm{VIEW}_A^{\Pi_1}(D_A, D'_B) = \nu] \right) \\
&\quad + \delta + \frac{1}{p(\pi)} \\
&= \exp(\epsilon) \Pr[G(\mathrm{VIEW}_A^{\Pi_2,\Pi_1}(D_A, D'_B)) = 1] + \delta + \frac{1}{p(\pi)}.
\end{aligned}
$$

We note that the first inequality is due to Lemma 3.3.3 and that in the case when δ is a positive constant independent of π, the term $1/p(\pi)$ on the right-hand side of the inequality could be dropped since δ is the dominant term compared to $1/p(\pi)$. ■

Next, we prove that for any pair of Bob's neighboring datasets (D_B, D'_B) with respect to record linkage, Charlie's view during the protocol does not change too much[6].

Theorem 3.3.3 *For each pair of Bob's neighboring datasets (D_B, D'_B), any probabilistic polynomial time distinguisher G, any positive polynomial $p(\cdot)$, and any sufficiently large π,*

$$\Pr[G(\mathit{VIEW}_C^{\Pi}(D_A, D_B) = 1] \leq \exp(\epsilon) \Pr[G(\mathit{VIEW}_C^{\Pi}(D_A, D_B) = 1] + \delta + \frac{1}{p(\pi)}.$$

as long as for any set V_B of synopses

$$\Pr[\mathcal{M}_B(D_B) = V_B] \leq \exp(\epsilon) \Pr[\mathcal{M}_B(D'_B) = V_B] + \delta,$$

where $\mathcal{M}_B(\cdot)$ denotes the partitioning algorithm adopted by Bob which takes as input his dataset and outputs a corresponding set V_B of synopses consisting of the extent and the noisy count for each partition.

Proof In our protocol, Alice and Bob privately partitions their respective datasets. Let $\mathcal{M}_A(\cdot)$ and $\mathcal{M}_B(\cdot)$ denote Alice's and Bob's algorithms that partition their respective input datasets and generate the corresponding sets of synopses V_A and V_B. Next, both data owners generate the ciphertexts for the records within each partition using their respective public keys pk_A and pk_B in the Paillier cryptosystem. Suppose that $\mathcal{W}$ denotes the set of all possible sets of synopses. Then, for any pair of Bob's

[6]The case for Alice's neighboring datasets (D_A, D'_A) could be similarly proved and thus the proof is omitted.

neighboring datasets (D_B, D'_B), any polynomial time distinguisher G, any positive polynomial $p(\cdot)$, and any sufficiently large π, we have

$$
\begin{aligned}
&\Pr[G(\mathrm{VIEW}_C^{\Pi}(D_A, D_B)) = 1] \\
&= \sum_{V_A, V_B \in \mathcal{W}} \big(\Pr[G(\mathrm{VIEW}_C^{\Pi}(V_A, V_B, D_A, D_B)) = 1] \\
&\quad \times \Pr[\mathcal{M}_A(D_A) = V_A] \Pr[\mathcal{M}_B(D_B) = V_B]\big) \\
&\leq \sum_{V_A, V_B \in \mathcal{W}} \left[\left(\Pr[G(\mathrm{VIEW}_C^{\Pi}(V_A, V_B, D_A, D'_B)) = 1] + \frac{1}{p(\pi)}\right)\right. \\
&\quad \left. \times \Pr[\mathcal{M}_A(D_A) = V_A] \Pr[\mathcal{M}_B(D_B) = V_B]\right] \\
&= \left(\sum_{V_A, V_B \in \mathcal{W}} \Pr[G(\mathrm{VIEW}_C^{\Pi}(V_A, V_B, D_A, D'_B)) = 1]\right. \\
&\quad \left. \times \Pr[\mathcal{M}_A(D_A) = V_A] \Pr[\mathcal{M}_B(D_B) = V_B]\right) + \frac{1}{p(\pi)} \\
&\leq \left[\sum_{V_A, V_B \in \mathcal{W}} \Pr[G(\mathrm{VIEW}_C^{\Pi}(V_A, V_B, D_A, D'_B)) = 1]\right. \\
&\quad \left. \times \Pr[\mathcal{M}_A(D_A) = V_A] \left(\exp(\epsilon) \Pr[\mathcal{M}_B(D'_B) = V_B] + \delta\right)\right] + \frac{1}{p(\pi)} \\
&\leq \left(\sum_{V_A, V_B \in \mathcal{W}} \exp(\epsilon) \Pr[G(\mathrm{VIEW}_C^{\Pi}(V_A, V_B, D_A, D'_B)) = 1]\right. \\
&\quad \left. \times \Pr[\mathcal{M}_A(D_A) = V_A] \Pr[\mathcal{M}_B(D'_B) = V_B]\right) + \delta + \frac{1}{p(\pi)} \\
&= \exp(\epsilon) \Pr[G(\mathrm{VIEW}_C^{\Pi}(D_A, D'_B)) = 1] + \delta + \frac{1}{p(\pi)}.
\end{aligned}
$$

We note that the first inequality stems from the fact that given the respective synopses V_A and V_B from Alice and Bob, we are able to simulate the ciphertexts observed by Charlie no matter which one of D_B or D'_B is used by Bob as input, which in turn implies that the ciphertexts derived from (V_A, V_B, D_A, D_B) and (V_A, V_B, D_A, D'_B) are computationally indistinguishable. More formally, based on the synopses we can build a simulator to simulate the encrypted records of Alice and Bob. Without loss of generality, we demonstrate how to simulate the encrypted records of Alice. Let

x be a record of Alice's data subset c_1 in a real execution of the protocol. Suppose that $e_{10} = E(pk_A, x.ID)$ and $e_{1i} = E(pk_A, x.A_i)$ are x's encryption for $i = 1, 2, \ldots, d$. We simulate the encrypted x by randomly generating a record $\tilde{x}$ in the extent of c_1, computing $\widetilde{e_{10}} = E(pk_A, \tilde{x}.ID)$ and $\widetilde{e_{1i}} = E(pk_A, \tilde{x}.A_i)$, $1 \leq i \leq d$. Because of the semantic security of Paillier cryptosystem, $(\widetilde{e_{10}}, \widetilde{e_{11}}, \widetilde{e_{12}}, \ldots, \widetilde{e_{1d}})$ is computationally indistinguishable from $(e_{10}, e_{11}, e_{12}, \ldots, e_{1d})$. Last, as pointed out in the proof of Theorem 3.3.2, the term $1/p(\pi)$ on the right-hand side of the inequality could be dropped in the case when δ is a positive constant independent of π. ■

From Theorem 3.3.2 and Theorem 3.3.3, we can see that our protocol satisfies $(\epsilon, \delta, f_{\mathsf{Rule}})$-DPRL if for each pair of Alice's and Bob's respective neighboring datasets (D_A, D'_A), (D_B, D'_B), the probabilities of generating any synopses (observed by Charlie) as well as the number of required secure comparisons (observed by both data owners) do not change too much. More precisely, we have the following corollary.

Corollary 3.3.1 *Our protocol satisfies $(\epsilon, \delta, f_{\mathsf{Rule}})$-DPRL if for each pair of Alice's and Bob's respective neighboring datasets with respect to f_{Rule}, i.e., (D_A, D'_A), (D_B, D'_B), any non-negative integer ν, and any sets of synopses V_A, V_B, we have*

1. $\Pr[VIEW_A^{\Pi_1}(D_A, D_B) = \nu] \leq \exp(\epsilon) \Pr[VIEW_A^{\Pi_1}(D_A, D'_B) = \nu] + \delta$,
2. $\Pr[VIEW_B^{\Pi_1}(D_A, D_B) = \nu] \leq \exp(\epsilon) \Pr[VIEW_B^{\Pi_1}(D'_A, D_B) = \nu] + \delta$,
3. $\Pr[\mathcal{M}_A(D_A) = V_A] \leq \exp(\epsilon) \Pr[\mathcal{M}_A(D'_A) = V_A] + \delta$, *and*
4. $\Pr[\mathcal{M}_B(D_B) = V_B] \leq \exp(\epsilon) \Pr[\mathcal{M}_B(D'_B) = V_B] + \delta$.

In other words, as long as the partitioning algorithms adopted by both data owners allow us to prove the validity of those four inequalities, the protocol as a whole satisfies $(\epsilon, \delta, f_{\mathsf{Rule}})$-DPRL[7].

[7]We would like to emphasize that if both data owners agree to release to the other party the noisy number of non-matching records for each subset, then as long as the third and the fourth inequalities in Corollary 3.3.1 hold, the protocol which allows each data owner to adaptively partition her/his dataset still satisfies $(\epsilon, \delta, f_{\mathsf{Rule}})$-DPRL, because it is sufficient to use only V_A, V_B, and $f_{\mathsf{Rule}}(D_A, D_B)$ to simulate Alice's (Bob's) view during the protocol execution without access to Bob's (Alice's) private input D_B (D_A) using a similar argument in Lemma 3.3.3.

3.4 Noise-adaptive Data Partition

In this section we present a differentially private indexing scheme, which partitions data into non-overlapping subsets.

3.4.1 Non-Dominating Rule

An important issue in building a private index is how to allocate privacy budget to its nodes. Past approaches [9, 15] allocate privacy budget evenly to each level of nodes in the index. A more recent approach [16] adopts a geometric progression, where the ratio of privacy budget between a child node and its parent is a user defined constant greater than 1. As far as we know, none of existing approaches considers data distribution when allocating privacy budget to nodes.

We allocate the privacy budget based on node size. We use a relative term to measure the accuracy of the approximate node size after adding noise. Suppose that s is the node size, and $\mathsf{Lap}\,(\lambda)$ is the Laplace noise with 0 mean and scale of λ. We set a threshold on the ratio between the expected absolute value of the noise and the node size. That is,

$$\frac{\mathbb{E}[|\mathsf{Lap}\,(\lambda)\,|]}{s} \leq \eta, \tag{3.9}$$

where η is a threshold. We refer to Inequality 3.9 as the *non-dominating rule*.

Let ϵ be the privacy budget assigned to a node. Since the sensitivity of counting node size is 1, the Laplace noise added to node size has the scale $\lambda = \frac{1}{\epsilon}$. According to the Laplace distribution in Equation 2.6, the expected absolute value of the Laplace noise is

$$\mathbb{E}(|\mathsf{Lap}\,(\lambda)\,|) = \frac{1}{2\lambda}\int_{-\infty}^{\infty} |z| e^{-|z|/\lambda} dz = \lambda.$$

Substitute it into Inequality 3.9, and it follows that

$$\epsilon \geq \frac{1}{s \cdot \eta}, \tag{3.10}$$

which gives the lower bound on the privacy budget to ensure that the noise on average is at most η times as big as the node size. The inequality also indicates that the

privacy budget is inversely proportional to the node size. This is reasonable, since a bigger node size can tolerate higher noise without reducing the relative accuracy.

3.4.2 The Algorithm

Our algorithm for building the private index is guided by the non-dominating rule. We consider a data space by taking each linking attribute as a dimension. If the attribute is categorical, we assign numerical labels to its values. Given a dataset, the algorithm recursively partitions the data space in a top-down way, forming a tree. It starts from a single node (i.e., the root) that covers all the data. Then, it recursively partitions the nodes. Before reaching the leaves, a node is always partitioned into two child nodes. If a node is to be partitioned into leaf nodes, then the number of its children is not fixed. Instead the number is up to the node size and also the available privacy budget. The resultant leaves of the tree are the subsets that participate in the hybrid private linkage protocol in Section 3.3.1.

Given privacy parameters ϵ and δ, our partitioning algorithm divides each of them in half, and ensures that the privacy budget consumed along each root-to-leaf path is upper-bounded by $\epsilon/2$, for which the reason shall be clear later in our analysis. We then divide the total privacy budget along each root-to-leaf path into two categories: *probing budget* is to test whether to split nodes, and *populating budget* is to count leaf sizes. Specifically, a budget of $\epsilon_{\mathsf{leaf}}/2$ is reserved for the populating budget, and the remaining $(\epsilon - \epsilon_{\mathsf{leaf}})/2$ is used for probing, where ϵ_{leaf} is set to $\epsilon/2$ unless otherwise specified. We reserve a big portion of privacy budget for leaves, since only leaves participate in the private linkage. This is different from other indexing approaches [15, 16], in which an adequate budget is allocated to internal nodes to support range queries that cover internal nodes. In addition, the sensitivity of the private count query used in the Laplace mechanism is 1. To generate noisy counts for the produced leaf nodes, the mean of the Laplace distribution with the scale equal to λ is shifted to the positive direction by $\lceil \mu' \rceil = \lceil -\lambda \ln(2\delta') \rceil$, where $\delta' = \delta/2$, to ensure

that the probability of generating a negative noise is upper-bounded by $\delta/2$. We do *not* suppress any records once a negative noise is drawn. This way, we avoid record suppression due to negative noise, which would lead to privacy violation, as shown in [10, 11].

Let C be a node. If it is to be split into two internal nodes, we allocate the privacy budget as follows. Let $C.\widetilde{s}$ be the noisy size of node C. We assume that the two children of C are approximately of equal size. Then, according to the non-dominating rule (Inequality 3.10), the privacy budget ϵ_x allocated to each child is $\epsilon_x = 2/(C.\widetilde{s} \times \eta)$. Let $C.B$ be the remaining probing budget at node C (the probing budget at the root of the tree is initialized to $(\epsilon - \epsilon_{\mathsf{leaf}})/2$). If $C.B \geq \epsilon_x$, then C is split into two children. Correspondingly, ϵ_x is subtracted from $C.B$, which is then passed to each child of C as the remaining probing budget.

We select one dimension A_i to split node C. Like when building a kd-tree, we select A_i in a round-robin way based on the depth of C in the tree. The extent of node C along A_i is considered, and its middle point (i.e., the average of the lower and upper bounds of the extent of C along A_i) is taken as the splitting point. All the records of C with A_i values less than the splitting point are pushed to the left child, and all the remaining records in C are pushed to the right child. The noisy sizes of the children are computed by adding Laplace noise $\mathsf{Lap}\,(1/\epsilon_x)$.

We thus recursively split the nodes. The probing budget decreases while the tree grows. At the point when the remaining probing budget $C.B$ at node C is smaller than ϵ_x, we partition C into a set of leaves. The total unused privacy budget, including the remaining probing budget and the reserved populating budget, is $\epsilon_u = C.B + \epsilon_{\mathsf{leaf}}/2$. Suppose that C is partitioned into H leaves, which are approximately of equal size. Then, according to non-dominating rule, we have

$$\frac{\mathbb{E}(|\mathsf{Lap}\,(1/\epsilon_u)\,|)}{C.\widetilde{s}/H} \leq \eta,$$

where $\tilde{s}$ is the noisy count of node C. Setting the inequality to be equal, we obtain

$$H = \left\lfloor C.\widetilde{s} \times \left(C.B + \frac{\epsilon_{\mathsf{leaf}}}{2} \right) \times \eta \right\rfloor.$$

Each dimension of node C is split to generate the leaves. Given a dimension, we partition the extent of node C on the dimension into $\lfloor \sqrt[d]{H} \rfloor$ sub-intervals of equal length. If the dimension is categorical, it is split into $\min\{I, \lfloor \sqrt[d]{H} \rfloor\}$ sub-intervals, where I is the number of distinct values in C on that dimension. Therefore, the number of resultant leaves may be smaller than H.

Algorithm 3.1 adTree-NDR: Construct an adaptive Tree based on Non-Dominating Rule

Input: D, a dataset; ϵ, total privacy budget; ϵ_{leaf}, total privacy budget for counting leaf nodes; δ, upper bound on the probability of violating differential privacy; η, a threshold for Non-Dominant Rule

Output: $\tilde{D}$, noisy counts of leaf nodes

1: $\tilde{D} \leftarrow \{\}$
2: Create the tree root r
3: $r.\tilde{s} \leftarrow |D|$
4: $r.B \leftarrow (\epsilon - \epsilon_{\mathsf{leaf}})/2$
5: Add r to an empty FIFO $\mathbb{Q}$
6: **while** $\mathbb{Q} \neq \phi$ **do**
7: $\quad C \leftarrow \mathbb{Q}$.remove()
8: $\quad \epsilon_x \leftarrow \frac{2}{C.\tilde{s} \times \eta}$
9: $\quad$ **if** $\epsilon_x \leq C.B$ **and** $\epsilon_x > 0$ **then**
10: $\quad\quad i \leftarrow (\mathsf{Depth}(C) \bmod d) + 1$
11: $\quad\quad$ Let m_i be the middle point of $\mathsf{Range}(C.A_i)$
12: $\quad\quad L \leftarrow \mathsf{Create}$(tuples in C with A_i values $< m_i$)
13: $\quad\quad R \leftarrow \mathsf{Create}$(tuples in C with A_i values $\geq m_i$)
14: $\quad\quad L.\tilde{s} \leftarrow L.s + \mathsf{Lap}\,(1/\epsilon_x)$; $R.\tilde{s} \leftarrow R.s + \mathsf{Lap}\,(1/\epsilon_x)$
15: $\quad\quad L.B \leftarrow C.B - \epsilon_x$; $R.B \leftarrow C.B - \epsilon_x$
16: $\quad\quad$ Add L and R to $\mathbb{Q}$

Algorithm 3.1 adTree-NDR: Construct an adaptive Tree based on Non-Dominating Rule

17: **else**
18: $W \leftarrow \max(\lfloor C.\tilde{s} \times (C.B + \frac{\epsilon_{\mathsf{leaf}}}{2}) \times \eta \rfloor, 1)$
19: Partition C into a grid with W cells
20: **for** each cell c in the grid **do**
21: $c.\epsilon_u \leftarrow C.B + \epsilon_{\mathsf{leaf}}/2$
22: Add c to $\tilde{D}$
23: **end for**
24: **end if**
25: **end while**
26: **for** each $c \in \tilde{D}$ **do**
27: $\lambda \leftarrow 1/c.\epsilon_u$
28: $c.\widetilde{s} \leftarrow |c| + \mathrm{rpi}(\mathsf{Lap}(\lambda)) + \lceil -\lambda \ln(\delta) \rceil$
29: **end for**
30: **return** $\tilde{D}$

Comments

- Step 3 assumes the whole dataset size is a public parameter.
- Steps 10–16 partition a node into two children (Step 13), if the node is not a parent of leaf nodes.
- Steps 18–23 partition a node into leaf nodes.

Lemma 3.4.1 *Given any set T of extents of leaf nodes corresponding to a partitioning tree generated using the probing budget in the first stage $\mathcal{M}_1$ (Steps 6 to 25) of our partitioning algorithm, the ratio of probabilities of producing T given a pair of*

Bob's neighboring datasets (D_B, D'_B) *with respect to* f_{Rule} *and Alice's dataset* D_A*, can be upper bounded by* $\exp(\epsilon - \epsilon_{\mathsf{leaf}})$*. Specifically,*

$$\frac{\Pr[\mathcal{M}_1(D_B) = T]}{\Pr[\mathcal{M}_1(D'_B) = T]} \leq \exp(\epsilon - \epsilon_{\mathsf{leaf}}).$$

Proof Suppose that there are γ internal tree nodes in the partitioning tree corresponding to the given set T of extents of leaf nodes. Without loss of generality, let those γ internal tree nodes be $C_1, \ldots, C_\gamma$, where C_i is the i-th generated internal tree node by the first stage $\mathcal{M}_1$ our partitioning algorithm. During the execution of $\mathcal{M}_1(D_B)$ and $\mathcal{M}_1(D'_B)$, the random variables $\mathcal{E}_i^{D_B}$ and $\mathcal{E}_i^{D'_B}$ are used to determine the amount of privacy budget ϵ_i to generate the noisy count $C_i.\tilde{s}$ according to instances of Laplace mechanism denoted by the random variables $\mathcal{A}_{\epsilon_i}^{D_B}$ and $\mathcal{A}_{\epsilon_i}^{D'_B}$ satisfying $\Pr[\mathcal{A}_{\epsilon_i}^{D_B} = C_i.\tilde{s}] \leq e^{\epsilon_i} \Pr[\mathcal{A}_{\epsilon_i}^{D'_B} = C_i.\tilde{s}]$. Let $\phi = (\epsilon_1, C_1.\tilde{s}, \ldots, \epsilon_\gamma, C_\gamma.\tilde{s})$ be a tuple generated by our partitioning algorithm adTree-NDR that results in the given set T of leaf node extents and that $P_T^{D_B}(\phi)$ be the probability of generating T given D_B according to ϕ. We can see that $\Pr[\mathcal{M}_1(D_B) = T] = \sum_{\phi \in \Phi} P_T^{D_B}(\phi)$, where Φ represents the set of all possible tuples generated by adTree-NDR that result in T. Thus, it can be seen that as long as it is true that for each $\phi \in \Phi$, $P_T^{D_B}(\phi)/P_T^{D'_B}(\phi) \leq \exp(\epsilon - \epsilon_{\mathsf{leaf}})$, $\Pr[\mathcal{M}_1(D_B) = T]/\Pr[\mathcal{M}_1(D'_B) = T]$ will be upper-bounded by $\exp(\epsilon - \epsilon_{\mathsf{leaf}})$ as well.

To see this, we know that $P_T^{D_B}(\phi)$ can be expressed as

$$P_T^{D_B}(\phi) = \prod_{i=1}^{\gamma} \Big(\Pr[\mathcal{A}_{\epsilon_i}^{D_B} = C_i.\tilde{s} \mid \mathcal{E}_i^{D_B} = \epsilon_i, \mathcal{A}_{\epsilon_{i-1}}^{D_B}, \mathcal{E}_{i-1}^{D_B}, \cdots, \mathcal{A}_{\epsilon_1}^{D_B}, \mathcal{E}_1^{D_B}] \\ \Pr[\mathcal{E}_i^{D_B} = \epsilon_i \mid \mathcal{A}_{\epsilon_{i-1}}^{D_B}, \mathcal{E}_{i-1}^{D_B}, \cdots, \mathcal{A}_{\epsilon_1}^{D_B}, \mathcal{E}_1^{D_B}] \Big) \tag{3.11}$$

We notice that

$$\Pr[\mathcal{A}_{\epsilon_i}^{D_B} = C_i.\tilde{s} \mid \mathcal{E}_i^{D_B} = \epsilon_i, \mathcal{A}_{\epsilon_{i-1}}^{D_B}, \mathcal{E}_{i-1}^{D_B}, \cdots, \mathcal{A}_{\epsilon_1}^{D_B}, \mathcal{E}_1^{D_B}] = \Pr[\mathcal{A}_{\epsilon_i}^{D_B} = C_i.\tilde{s}], \tag{3.12}$$

and that

$$\Pr[\mathcal{E}_i^{D_B} = \epsilon_i \mid \mathcal{A}_{\epsilon_{i-1}}^{D_B}, \mathcal{E}_{i-1}^{D_B}, \cdots, \mathcal{A}_{\epsilon_1}^{D_B}, \mathcal{E}_1^{D_B}] = \Pr[\mathcal{E}_i^{D_B} = \epsilon_i \mid \mathcal{A}_{\epsilon_{p(i)}}^{D_B} = C_{p(i)}.\tilde{s}] = 1, \tag{3.13}$$

where $C_{p(i)}$ is the parent node of C_i. The Equation 3.13 holds since the amount of privacy budget $\mathcal{E}_i^{D_B}$ used to count C_i is deterministically determined by $C_{p(i)}.\tilde{s}$ using the non-dominating rule. Now, define $I = \{i \mid 1 \leq i \leq \gamma\}$. We have

$$P_T^{D_B}(\phi) = \prod_{i=1}^{\gamma} \Pr[\mathcal{A}_{\epsilon_i}^{D_B} = C_i.\tilde{s}] = \prod_{i \in J} \Pr[\mathcal{A}_{\epsilon_i}^{D_B} = C_i.\tilde{s}] \prod_{i \in \overline{J}} \Pr[\mathcal{A}_{\epsilon_i}^{D_B} = C_i.\tilde{s}], \quad (3.14)$$

where $J = \{j \mid \text{record } r_B \in C_j \text{ or record } r'_B \in C_j\}$, and $\overline{J} = I \backslash J$. The probability $P_T^{D'_B}(\phi)$ of generating T given D'_B can be derived similarly. Note that for $i \in \overline{J}$, it holds that $\Pr[\mathcal{A}_{\epsilon_i}^{D_B} = C_i.\tilde{s}] = \Pr[\mathcal{A}_{\epsilon_i}^{D'_B} = C_i.\tilde{s}]$, since the records within these internal nodes are exactly the same no matter which one of D_B and D'_B is used by Bob. On the other hand, for each $i \in J$, it holds that $\Pr[\mathcal{A}_{\epsilon_i}^{D_B} = C_i.\tilde{s}] \leq e^{\epsilon_i} \Pr[\mathcal{A}_{\epsilon_i}^{D'_B} = C_i.\tilde{s}]$. Thus, the ratio $(P_T^{D_B}(\phi)/P_T^{D'_B}(\phi))$ can be rewritten as

$$\frac{\prod_{i \in J} \Pr[\mathcal{A}_{\epsilon_i}^{D_B} = C_i.\tilde{s}]}{\prod_{i \in J} \Pr[\mathcal{A}_{\epsilon_i}^{D'_B} = C_i.\tilde{s}]} \leq \exp\left(\sum_{i \in J} \epsilon_i\right). \quad (3.15)$$

To ensure that the ratio is upper-bounded by $\exp(\epsilon - \epsilon_{\mathsf{leaf}})$, it should hold that $\exp(\epsilon - \epsilon_{\mathsf{leaf}}) = \exp\left(\sum_{i \in J} \epsilon_i\right)$. However, recall that the two root-to-leaf paths involving r_B and r'_B are not known in advance, which means we do not know from which internal tree node r_B and r'_B initially split. That is, we do not know the lowest internal tree node C_i that satisfies both $r_B \in C_i$ and $r'_B \in C_i$. Thus, the only way to distribute the budget $(\epsilon - \epsilon_{\mathsf{leaf}})$ is to evenly distribute it between the two paths, as we do in our partitioning algorithm described earlier. By doing this, we ensure that for any given root-to-leaf path, the ratio of the probability of generating the noisy counts associated with this path given D_B to that given D'_B is upper-bounded by $\exp((\epsilon - \epsilon_{\mathsf{leaf}})/2)$. Together with the fact that there are *at most* two such paths, the multiplicative factor is upper-bounded by $\exp(\epsilon - \epsilon_{\mathsf{leaf}})$. Lastly, we explain how we guarantee that $\Pr[\mathcal{A}_{\epsilon_i}^{D_B} = C_i.\tilde{s}] \leq e^{\epsilon_i} \Pr[\mathcal{A}_{\epsilon_i}^{D'_B} = C_i.\tilde{s}]$. This inequality can be satisfied if we set the sensitivity of the query to 1 when counting an internal node using the Laplace mechanism. The sensitivity is reached when this internal node is on the root-to-leaf path corresponding to r_B or r'_B ■

Lemma 3.4.2 *Given the set $\{C_1, \ldots, C_\psi\}$ of leaf nodes and their corresponding set T of extents produced by the first stage $\mathcal{M}_1$ of our partitioning algorithm, for any given noisy counts $C_1.\tilde{s}, \ldots, C_\psi.\tilde{s}$ produced using the populating budget in the second stage $\mathcal{M}_2$ (Steps 26 to 29) of our partitioning algorithm, with probability $1-\delta$, the ratio of probabilities of producing these noisy counts given any pair of Bob's neighboring datasets (D_B, D'_B) with respect to f_{Rule} and Alice's dataset D_A, can be upper bounded by $\exp(\epsilon_{\mathsf{leaf}})$. To be precise,*

$$\frac{\Pr[\mathcal{M}_2(T) = (C_1.\tilde{s}, \ldots, C_\psi.\tilde{s}) \mid \mathcal{M}_1(D_B) = T]}{\Pr[\mathcal{M}_2(T) = (C_1.\tilde{s}, \ldots, C_\psi.\tilde{s}) \mid \mathcal{M}_1(D'_B) = T]} \leq \exp(\epsilon_{\mathsf{leaf}}).$$

Proof Define $I = \{i \mid 1 \leq i \leq \psi\}$, $J = \{j \mid \text{record } r_B \in C_j \text{ or record } r'_B \in C_j\}$, and $\overline{J} = I \backslash J$. Let $\mathcal{A}_{\epsilon_i}^{D_B}$ and $\mathcal{A}_{\epsilon_i}^{D'_B}$ denote the random variables corresponding to the noisy counts of C_i produced by the Laplace mechanism given D_B and D'_B satisfying $\Pr[\mathcal{A}_{\epsilon_i}^{D_B} = C_i.\tilde{s}] \leq e^{\epsilon_i} \Pr[\mathcal{A}_{\epsilon_i}^{D'_B} = C_i.\tilde{s}]$ with probability at least $1 - \delta/2$. Similar to Lemma 3.4.1, the ratio of the probabilities to generate noisy counts $(C_1.\tilde{s}, \ldots, C_\psi.\tilde{s})$ given D_B or D'_B as input, can be expressed as

$$\frac{\prod_{i \in J} \Pr[\mathcal{A}_{\epsilon_i}^{D_B} = C_i.\tilde{s}]}{\prod_{i \in J} \Pr[\mathcal{A}_{\epsilon_i}^{D'_B} = C_i.\tilde{s}]} \leq \prod_{i \in J} e^{\epsilon_i} = \exp\left(\sum_{i \in J} \epsilon_i\right). \tag{3.16}$$

To ensure that the ratio is upper-bounded by $\exp(\epsilon_{\mathsf{leaf}})$, it must hold that $\exp(\epsilon_{\mathsf{leaf}}) = \exp\left(\sum_{i \in J} \epsilon_i\right)$. Not knowing whether or not r_B and r'_B fall into the same leaf node, our partitioning algorithm divide ϵ_{leaf} evenly and we guarantee that the ratio of the probability of generating the noisy count for a leaf node corresponding to some element in J given D_B to that given D'_B is upper-bounded by $\exp(\epsilon_{\mathsf{leaf}}/2)$.

Finally, to guarantee that $\Pr[\mathcal{A}_{\epsilon_i}^{D_B} = C_i.\tilde{s}] \leq e^{\epsilon_i} \Pr[\mathcal{A}_{\epsilon_i}^{D'_B} = C_i.\tilde{s}]$ with probability at least $1-\delta/2$, we shift the mean of the Laplace distribution to the positive direction such that the chance of generating a negative noise is upper-bounded by $\delta' = \delta/2$. Specifically, we solve the following inequality for μ' where $p(z \mid 0, \lambda) = \frac{1}{2\lambda} e^{-|z|/\lambda}$ is the probability density function of the Laplace distribution with the mean equal to 0, and then set $\lceil \mu' \rceil$ as the shifted mean:

$$\frac{1}{2} + \int_0^{\mu'} \frac{1}{2\lambda} e^{-z/\lambda} dz \geq 1 - \delta' \Rightarrow 1 - \frac{1}{2} e^{-\mu'/\lambda} \geq 1 - \delta' \Rightarrow \mu' \geq -\lambda \ln(2\delta'). \tag{3.17}$$

Since there are at most two elements in J, each corresponding to a leaf node having either r_B or r'_B, the probability of generating a negative noise for at least one leaf node is upper-bounded by $(\frac{\delta}{2})2 = \delta$ according to the union bound. ■

Lemma 3.4.3 *Let $\mathcal{M}_1$ be the first stage of our partitioning algorithm (Steps 6 to 25) producing a set T of extents of ψ leaf nodes and $\mathcal{M}_2$ be the second stage of our partitioning algorithm (Steps 26 to 29) that produces the corresponding noisy counts $(C_1.\tilde{s}, \ldots, C_\psi.\tilde{s})$. It holds that for any given set $\mathcal{S}$ containing tuples in the form of $(T, (C_1.\tilde{s}, \ldots, C_\psi.\tilde{s}))$ and any given pair of Bob's neighboring datasets (D_B, D'_B) with respect to the functionality f_{Rule} and Alice's dataset D_A,*

$$\Pr[\mathcal{M}_2(\mathcal{M}_1(D_B)) \in \mathcal{S}] \leq \exp(\epsilon) \Pr[\mathcal{M}_2(\mathcal{M}_1(D'_B)) \in \mathcal{S}] + \delta.$$

Proof We first note that due to Lemma 3.4.1 and Lemma 3.4.2, for any $(T, (C_1.\tilde{s}, \ldots, C_\psi.\tilde{s}))$, with probability at least $1 - \delta$,

$$\begin{aligned} &\frac{\Pr[\mathcal{M}_2(\mathcal{M}_1(D_B)) = (T, (C_1.\tilde{s}, \ldots, C_\psi.\tilde{s}))]}{\Pr[\mathcal{M}_2(\mathcal{M}_1(D'_B)) = (T, (C_1.\tilde{s}, \ldots, C_\psi.\tilde{s}))]} \\ &= \left(\frac{\Pr[\mathcal{M}_2(T) = (C_1.\tilde{s}, \ldots, C_\psi.\tilde{s}) \mid \mathcal{M}_1(D_B) = T]}{\Pr[\mathcal{M}_2(T) = (C_1.\tilde{s}, \ldots, C_\psi.\tilde{s}) \mid \mathcal{M}_1(D'_B) = T]}\right) \left(\frac{\Pr[\mathcal{M}_1(D_B) = T]}{\Pr[\mathcal{M}_1(D'_B) = T]}\right) \\ &\leq \exp(\epsilon_{\mathsf{leaf}}) \exp(\epsilon - \epsilon_{\mathsf{leaf}}) = \exp(\epsilon). \end{aligned} \tag{3.18}$$

Now, let $I = \{i \mid 1 \leq i \leq \psi\}$, $J = \{j \mid \text{record } r_B \in C_j \text{ or record } r'_B \in C_j\}$, and $\mathcal{F} = \{(T, (C_1.\tilde{s}, \ldots, C_\psi.\tilde{s})) \mid \exists i \in J \text{ such that } C_i.\tilde{s} < 0\}$. That is, $\mathcal{F}$ is the set that contains the tuples corresponding to the events when there is at least one negative noise drawn in $\mathcal{M}_2$ for leaf nodes containing r_B or r'_B, which happens with a probability of at most δ. On the other hand, for any given set $\mathcal{S}$, we know that $\Pr[\mathcal{M}_2(\mathcal{M}_1(D_B)) \in (\mathcal{S} \backslash \mathcal{F})] \leq \exp(\epsilon) \Pr[\mathcal{M}_2(\mathcal{M}_1(D'_B)) \in (\mathcal{S} \backslash \mathcal{F})]$. Thus, for any given set $\mathcal{S}$ containing any tuples output by our partitioning algorithm, we have

$$\begin{aligned} &\Pr[\mathcal{M}_2(\mathcal{M}_1(D_B)) \in \mathcal{S}] \leq \Pr[\mathcal{M}_2(\mathcal{M}_1(D_B)) \in (\mathcal{S} \backslash \mathcal{F})] + \Pr[\mathcal{M}_2(\mathcal{M}_1(D_B)) \in \mathcal{F}] \\ &\leq \exp(\epsilon) \Pr[\mathcal{M}_2(\mathcal{M}_1(D'_B)) \in (\mathcal{S} \backslash \mathcal{F})] + \delta \leq \exp(\epsilon) \Pr[\mathcal{M}_2(\mathcal{M}_1(D'_B)) \in \mathcal{S}] + \delta. \end{aligned}$$

■

Using Lemma 3.4.3, we are able to prove that with the help from the third party Charlie, the number of (non-matching) records in each leaf node of Bob (Alice) can be further hidden from Alice (Bob) via the coordination of Charlie such that the resulting number of secure comparisons satisfies the first two inequalities in Corollary 3.3.1 as well. Specifically, we have the following theorem that proves that the probability of any given number of required secure comparisons does not vary too much given any pair of Bob's neighboring datasets. The proof for the probability given any pair of Alice's neighboring datasetsets is essentially the same and thus is omitted.

Theorem 3.4.1 *Let $\mathcal{M}$ denote our partitioning algorithm and $\mathcal{N}_{\mathsf{Rule}}$ be the functionality computed by the third party Charlie that returns the number of required secure comparisons with respect to the decision rule* Rule *according to the synopses produced by $\mathcal{M}$ based on Alice's and Bob's respective datasets. Then, for any number ν of required secure comparisons and any given pair of Bob's neighboring datasets (D_B, D'_B) with respect to the functionality f_{Rule} and Alice's dataset D_A,*

$$\Pr[\mathcal{N}_{\mathsf{Rule}}(\mathcal{M}(D_B), \mathcal{M}(D_A)) = \nu] \leq \exp(\epsilon) \Pr[\mathcal{N}_{\mathsf{Rule}}(\mathcal{M}(D'_B), \mathcal{M}(D_A)) = \nu] + \delta. \tag{3.19}$$

Proof Let $\mathcal{W}$ be the set of all possible sets of synopses that could be output by our partitioning algorithm $\mathcal{M}$, $\mathcal{N}_{\mathsf{Rule}}{}^{-1}(\nu) = \{(V_B, V_A) \in (\mathcal{W} \times \mathcal{W}) \mid \mathcal{N}_{\mathsf{Rule}}(V_B, V_A) = \nu\}$,

$\mathcal{S}_A = \{V_A \in \mathcal{W} \mid \exists (V_B, V_A) \in {\mathcal{N}_{\mathsf{Rule}}}^{-1}(\nu)\}$, and $\mathcal{S}_{B,V_A} = \{V_B \in \mathcal{W} \mid (V_B, V_A) \in {\mathcal{N}_{\mathsf{Rule}}}^{-1}(\nu)\}$. Hence, we can see that

$$
\begin{aligned}
&\Pr[\mathcal{N}_{\mathsf{Rule}}(\mathcal{M}(D_B), \mathcal{M}(D_A)) = \nu] = \Pr[(\mathcal{M}(D_B), \mathcal{M}(D_A)) \in {\mathcal{N}_{\mathsf{Rule}}}^{-1}(\nu)] \\
&= \sum_{V_A \in \mathcal{S}_A} \Pr[\mathcal{M}(D_B) \in \mathcal{S}_{B,V_A} \mid \mathcal{M}(D_A) = V_A] \Pr[\mathcal{M}(D_A) = V_A] \qquad (3.20) \\
&= \sum_{V_A \in \mathcal{S}_A} \Pr[\mathcal{M}(D_B) \in \mathcal{S}_{B,V_A}] \Pr[\mathcal{M}(D_A) = V_A] \\
&\leq \sum_{V_A \in \mathcal{S}_A} (\exp(\epsilon) \Pr[\mathcal{M}(D'_B) \in \mathcal{S}_{B,V_A}] + \delta) \Pr[\mathcal{M}(D_A) = V_A] \\
&= \left[\sum_{V_A \in \mathcal{S}_A} (\exp(\epsilon) \Pr[\mathcal{M}(D'_B) \in \mathcal{S}_{B,V_A}] \Pr[\mathcal{M}(D_A) = V_A]) \right] + \delta \qquad (3.21) \\
&= \exp(\epsilon) \Pr[\mathcal{N}_{\mathsf{Rule}}(\mathcal{M}(D'_B), \mathcal{M}(D_A)) = \nu] + \delta.
\end{aligned}
$$

We note that the inequality is the result from Lemma 3.4.3. ■

According to Lemma 3.4.3 and Theorem 3.4.1, we can see that the partitioning algorithm adTree-NDR when adopted by both data owners to create their respective subsets for record linkage allows us to prove the validity of the four inequalities in Corollary 3.3.1, which is sufficient to prove that the protocol as a whole satisfies $(\epsilon, \delta, f_{\mathsf{Rule}})$-DPRL.

Corollary 3.4.1 *Our protocol for record linkage satisfies $(\epsilon, \delta, f_{\mathsf{Rule}})$-DPRL when both data owners adopt the partitioning algorithm adTree-NDR to generate their respective subsets.*

Time complexity. We analyze the time complexity of building the private indexing tree in terms of the tree size. Let the number of records in the dataset be n. The node splitting of the index depends on the data distribution. For simplicity we assume that data are uniformly distributed. Suppose the number of leaves is x. We reserved a privacy budget of $\epsilon_{\mathsf{leaf}}/2 = \epsilon/4$ for each leaf node. Thus, according to the non-dominating rule in Inequality 3.9, $x \leq \frac{n\eta\epsilon}{4}$. Since the total number of nodes in the tree is at most twice of the number of leaves, the complexity is $O(n\eta\epsilon)$.

3.5 Experimental Analysis

In this section we evaluate our approach. We configured the datasets of Alice and Bob by the *Adult* dataset [17]. We choose *occupation*, *relationship*, *sex*, *fnlwgt*, and *hours-per-week* as the linking attributes for private record linkage. For each categorical attribute, such as *occupation*, we order its values and then assign sequential integers to these values. Table 3.5 gives the original domain ranges of these 5 attributes. We randomly sample a subset of records from the Adult dataset and partition them into 3 equal-sized subsets d_1, d_2, and d_3. Then, $d_1 \cup d_2$ forms the dataset of Alice, and $d_2 \cup d_3$ forms that of Bob. In this way we create 4 datasets for Alice (Bob) with the sizes of 1,500, 3,000, 6,000, and 12,000, respectively. We set 6,000 to be the default dataset size. Our differentially private data partitioning algorithm controls the magnitude of Laplace noise added to each node in the indexing tree. It requires that the expectation of absolute noise be at most η times as big as the node size. Unless specified, we fix η to 0.1. We set the default privacy budget ϵ to 1.0 and δ to 10^{-4}.

Table 3.5.: Domain Ranges of Attributes

Attribute	**Type**	**Lower Bound**	**Upper Bound**
occupation	Categorical	0	16
relationship	Categorical	0	8
sex	Categorical	0	2
fnlwgt	Numerical	0	1,600,000
hours-per-week	Numerical	0	128

We use the *reduction ratio* to evaluate the efficiency our approach. Let N_A and N_B be the sizes of the datasets of Alice and Bob, respectively. Without optimization, in total $N_A \times N_B$ SMC invocations (i.e., one SMC invocation corresponds to the secure

matching of one pair of records) are needed. Let S be the number of SMC invocations needed by a hybrid approach. Then, the reduction ratio by the hybrid approach is

$$reduction = 1 - \frac{S}{N_A \times N_B}.$$

A higher reduction ratio results in fewer invocations of SMC, which in turn implies higher efficiency of the hybrid approach. Note that our protocol does not have false positives, since dummy records with attribute values (i.e., $-U$ for Alice and $2U$ for Bob) do not match others.

The prototype of our approach for data partitioning was implemented in Java, and most experiments were carried out on an Intel Core i7-2600 3.40GHz CPU machine with 8G RAM running Linux 3.4.13. Since Laplace density function is probabilistic, for each configuration of parameters, we ran the data partitioning algorithm 50 times, each time with a different randomization seed to generate the noise, and then reported the average.

3.5.1 The Parameter Tuning

Our approach has parameters: ϵ—the total privacy budget, δ—the probability that DPRL is violated, η—the threshold that controls the magnitude of Laplace noise added to a tree node, and θ—the threshold that decides whether two records match. We tune these parameters to study their effects. Our private indexing tree to partition data is adaptive on the basis of the non-dominating rule (Inequality 3.9). Thus, we use *adTree-NDR* to represent our approach, and *Tree-Alice* and *Tree-Bob* to represent the private indexing trees built on the datasets of Alice and Bob, respectively. The decision rule SSE has parameters θ and w_i for $1 \leq i \leq 5$. By default, we set $\theta = 0$ and $w_i = 1$.

We first vary ϵ. A larger ϵ allows more node splitting when building the private indexing tree from either the dataset of Alice or that of Bob. Thus, as shown on the left-hand side in Figure 3.5, when ϵ grows, the number of leaf nodes in the tree of Alice (Bob) becomes larger. Therefore, on average leaf nodes become smaller, and

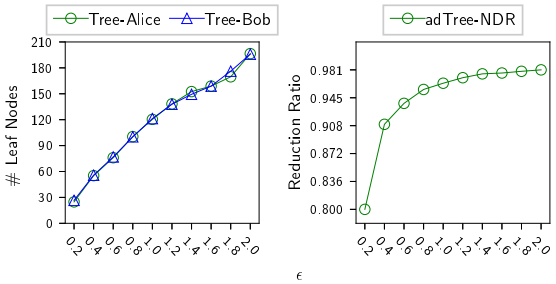


Figure 3.5.: The Effect of Varying ϵ

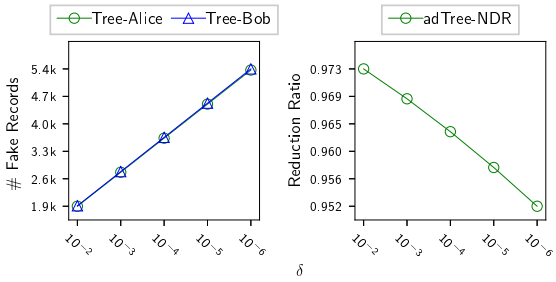


Figure 3.6.: The Effect of Varying δ

a record of Alice needs to be compared with fewer records of Bob. The right-hand side in Figure 3.5 confirms this—as ϵ increases, the total number of SMC invocations decreases (i.e., the reduction ratio increases).

Now, we study the effect of δ. Recall that in Inequality 3.17, the shifted mean μ' is computed as $-\lambda \ln(\delta)$. Hence, it is not difficult to see that when δ decreases exponentially, the shifted mean μ' would grow linearly, which is consistent with the number of fake records each data owner has generated as reported on the left-hand side in Figure 3.6. This fact in turn results in a linear decrease in the reduction ratio that we observe on the right-hand side in Figure 3.6.

Next, we study the effect of η. Given a privacy budget allocated to a node in the indexing tree, as η increases, the node size could be smaller (Inequality 3.10). Thus, on the left-hand side in Figure 3.7 the number of leaf nodes in the indexing tree of Alice (Bob) grows as a function of η. Therefore, like the result on the right-hand side in Figure 3.5, the reduction ratio improves (Figure 3.7).

We now vary the threshold θ in the decision rule, and investigate its effect. A larger θ relaxes the decision rule. A record of one party thus needs to be compared with more records from the other party. Thus, as θ increases from 0.000 to 0.075, in Figure 3.8, the reduction ratio decreases.

3.5.2 The Efficiency Evaluation

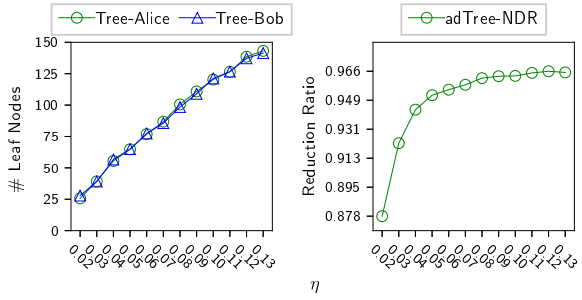


Figure 3.7.: The Effect of Varying η

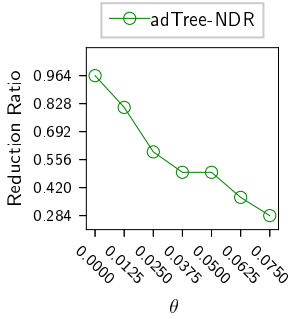


Figure 3.8.: The Effect of Varying θ

We now evaluate the efficiency of our approach. We consider the cost of: I) building private indexes (Section 3.4) to partition data, II) encrypting the records (Section 3.3.1.1), III) pruning the linkage of non-matching records (cost incurred by Charlie), and IV) matching the records via SMC (Section 3.3.1.2). We downloaded a C-based library [18] to instantiate the Paillier encryption scheme [19]. The moduli used by both parties are 1,024-bit long, and the statistical security parameter κ is set to 40. The garbled circuit we construct for secure comparison is based on FastGC [20], one of the most widely used Java-based implementations for garbled circuits.

It took each party on average 35 ms to privately partition the data. Alice and Bob send encrypted records to Charlie. The encryption of a record took Alice (Bob) around 16.8 ms. Thus, the elapsed time for encrypting the whole dataset was about 101 seconds. When instantiating the decision rule, we scale the domain of each attribute to $[0, 160]$ and set the weight w_i for the i'th linking attribute to 1. After Charlie received the private synopses from Alice and Bob, it took Charlie around 10 ms to carry out the pruning.

The record matching via SMC is pair-wise. It consists of data randomization and 2-party decision making. Given a pair of records, Charlie spends 33.7 ms to randomize their values. The maximum squared Euclidean distance between any two records is $V^{\Delta} = 5(2 \cdot 160 - (-160))^2$, which is reached when a dummy record of Alice is compared with a dummy record of Bob. Since $\lceil \log_2(V^{\Delta}) \rceil = 21$ and $\kappa = 40$, in the decision making process, Alice and Bob jointly compare two integers of at most $\psi + 1 = (40 + 21) + 1 = 62$ bit long. Here we assume that Charlie concurrently performs the data randomization with Alice and Bob. Specifically, we assume that Charlie sequentially computes e'_{1i}, e'_{2i}, $1 \leq i \leq d$, e'_{10}, e'_{20} and that each ciphertext is transmitted immediately once it has been generated. In addition, we assume that Bob first computes $E(pk_A, (\sum_{i=1}^{d} w_i \cdot y.A_i'^2) + \rho)$ without waiting for those $d+1$ ciphertexts from Alice. The matching thus takes around 38.94 ms per pair[8].

[8]The elapsed time is much less than that (97.84 ms) in the conference version of this chapter [10]. Previously, the TCP/IP socket is set in a buffered fashion. It sends data only if enough data is in the queue. Now, it sends data once data is available in the queue.

Here we also give an estimated cost for matching a pair of records in the 2-party case. In the 2-party protocol, Alice and Bob do not have to carry out the decryption of those d ciphertexts received from Charlie as in the 3-party case. The computational cost thus reduces from 38.94 ms to 21.99 ms—that is, 43.52% of the elapsed time could be saved if both parties agree to reveal the noisy number of non-matching records within each subset. In addition, since $|\mathbb{Z}_{n^2}| = 2{,}048$ and the bit-length of a wire label in FastGC [20] is $k = 160$, the communication cost reduces from 108.35 kbits to 83.78 kbits in the 2-party case—22.68% less than the 3-party case.

We first evaluate the elapsed time of private record linkage on a *single* computer. When the dataset size of Alice (Bob) is 6,000 and the privacy budget ϵ is 1.0 with δ set to 10^{-4}, our hybrid approach needs to compare 1.31×10^6 pairs of records via SMC. It took 14.19 hours. The traditional 2-party private linkage approach, which compares 3.6×10^7 pairs of records, would take 219.9 hours. This is more than an improvement of 1 order of magnitude. On the left-hand side in Figure 3.9 we report the result for other dataset sizes.

Our proposed approach allows for parallel execution. Charlie could partition the pairs of records (that need to be securely matched) into a batch of jobs, and run each job independently on a single core of a CPU. We thus also conducted the record matching on a cluster of computers. The cluster has 326 nodes, each consisting of two Quad-Core Intel E5410 CPUs. We partitioned the pairs of records into 100 jobs and submitted them to the cluster. On the right-hand side in Figure 3.9 it can be seen that on average, when the dataset size of Alice (Bob) is 6,000 and privacy budget ϵ is 1.0 with $\delta = 10^{-4}$, it takes 35.48 minutes to complete the record matching.

3.5.3 Comparative Study

We compare our approach with two benchmarks. The first is one of the state-of-the-art hybrid private linkage schemes [9]. We name it *DPkdT-basic*, since it uses a kd-tree to privately partition data. The second is one of the state-of-the-art pri-

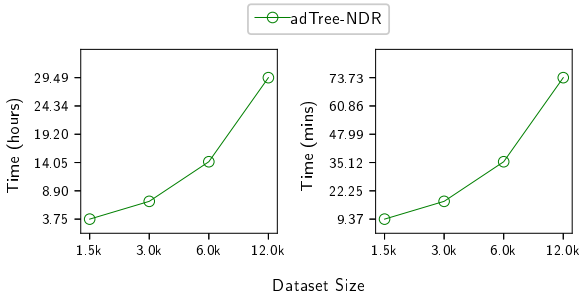


Figure 3.9.: Elapsed Time by Varying Dataset Size (Left: Sequential, Right: Parallel)

vate indexing schemes [16]. We name it *DPkdT-GB*, since it allocates geometrically increasing privacy budgets to nodes when building a private kd-tree. On the other hand, since neither of the two benchmarks takes into consideration the privacy notion of $(\epsilon, \delta, f_{\mathsf{Rule}})$-DPRL when constructing a partitioning tree, we have to modify their implementations to ensure that the consumption of the privacy budget along each root-to-leaf path is upper bounded by $\epsilon/2$ and that the probability that a negative noise is drawn when counting the size of a leaf node is also bounded above by $\delta/2$.

Since we would like to provide a comprehensive comparison of the efficiency of our partitioning algorithm with the other two under all possible settings of privacy budget ϵ and dataset size, we first tune the approaches to create a level playing field. Specifically, for each possible dataset size N and privacy budget ϵ, we identify a suitable parameter to achieve high efficiency. The results are provided in Figure 3.10 through Figure 3.13 for each possible dataset size. In each figure, we only report the results for $\epsilon = 0.6$ and $\epsilon = 1.6$ for clarity.

As seen in Section 3.5.1, the number of leaf nodes grows with η, the parameter that controls the ratio of the noise magnitude to the estimated sizes of two child nodes when trying to split a parent node C. A larger η implies more leaf nodes, which in turn improves the reduction ratio. This can be corroborated from the results of the leftmost graphs in Figure 3.10 through Figure 3.13. We observe that the reduction ratio is a bit fluctuating for the first few η values when the dataset size $N = 1{,}500$.

The reason is that a parent node C is more sensitive to the noise used to perturb its size when N is smaller. Recall that we use $2/(C.\tilde{s} \times \eta)$ to estimate the amount of budget to count C's two child nodes. When $C.\tilde{s}$ becomes too small due to a previously added negative noise, the splitting of the node C would stop, resulting in fewer leaf nodes in the end. This situation improves when we increase η to at least 0.05, after which the number of leaf nodes monotonically increases with η. However, the increase in reduction ratio is not monotonic when we keep enlarging η, since generating more leaf nodes could also increase the number of required secure comparisons due to those fake records. In general, we found that when $\eta = 0.10$, adTree-NDR achieves a high reduction ratio for each dataset size and privacy budget. We thus set η to 0.10 for adTree-NDR for *all* possible dataset sizes and privacy budgets.

The DPkdT-basic approach assigns the privacy budget uniformly at each tree level, when building the private kd-tree. Hence, the tree height plays an important role in its performance, since a larger tree height results in more leaf nodes. On the other hand, as discussed above, the increase in reduction ratio when increasing the number of leaf nodes is not monotonic due to the addition of fake records. We thus have to identify a suitable tree height optimized for DPkdT-basic under each combination of possible dataset size and privacy budget. The middle graphs in Figure 3.10 through Figure 3.13 report the results of $\epsilon = 0.6$ and $\epsilon = 1.6$. From these results we can see that a larger tree height would be allowed for if we had either (i) a bigger privacy budget ϵ, or (ii) a larger dataset size N, both of which are expected. When ϵ is larger, the number of fake records added to each leaf node would be smaller, and hence more leaf nodes could be allowed for without a decrease in reduction ratio. Take $N = 1{,}500$, in Figure 3.10, we can see that when $\epsilon = 0.6$, a smaller tree height of 2 yields a better reduction ratio, whereas a larger tree height of 3 produces a better reduction ratio when $\epsilon = 1.6$. On the other hand, when N is larger, the magnitude of noise would be comparatively smaller (to N), making it possible to choose a larger tree height in order to improve the reduction ratio. For instance, when $\epsilon = 0.6$, the best tree height when $N = 6{,}000$ is 3, whereas it could be increased to 4 once the

dataset size is increased to 12,000. For the sake of completeness, in Table 3.6, we also identify the best tree height for other possible combinations of N and ϵ.

The DPkdT-GB approach assigns the privacy budget geometrically to the nodes, when building a differentially private kd-tree. More precisely, the budget assigned to a child node is $\sqrt[3]{2}$ times as large as that to its parent node. DPkdT-GB reserves a portion of privacy budget to select medians of kd-tree nodes, which are to be split. It also sets a threshold for the maximum tree height. We tuned these parameters to optimize its performance. We found that in general DPkdT-GB performs well when 30% percent privacy budget is reserved for median selection. Based on this, we identify the best tree height for DPkdT-GB given each possible combination of N and ϵ. The rightmost graphs in Figure 3.10 through Figure 3.13 report the results for $\epsilon = 0.6$ and $\epsilon = 1.6$. Similar to the tuning process of DPkdT-basic, we also give the best identified tree height under various combinations of N and ϵ in Table 3.7.

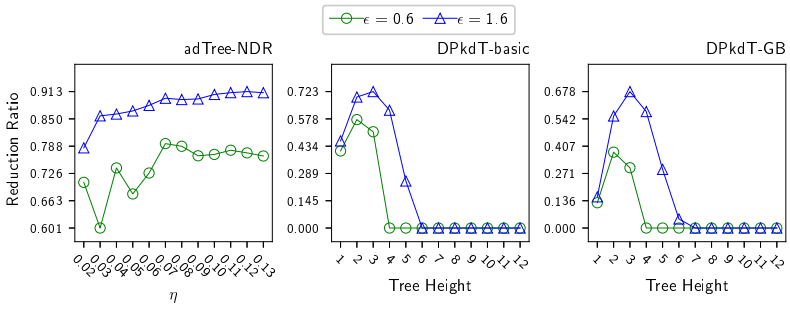


Figure 3.10.: Parameter Tuning when Database Size N = 1,500, $\delta = 10^{-4}$, and $\theta = 0.0$

Based on the suitable parameters identified above for each approach, we compare their efficiency under various privacy budgets and dataset sizes, where $\delta = 10^{-4}$ and $\theta = 0$. More precisely, for adTree-NDR, we set η to 0.10 for all possible ϵ values and dataset sizes, whereas we adopt the settings given in Table 3.6 and Table 3.7 for DPkdT-basic and DPkdT-GB, respectively.

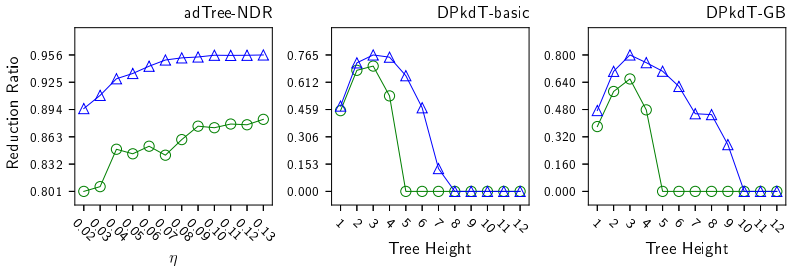


Figure 3.11.: Parameter Tuning when Database Size N = 3,000, $\delta = 10^{-4}$, and $\theta = 0.0$

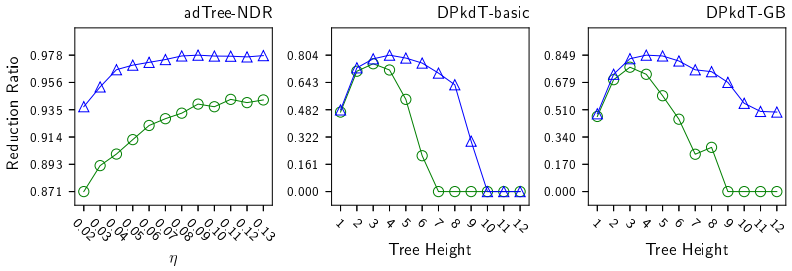


Figure 3.12.: Parameter Tuning when Database Size N = 6,000, $\delta = 10^{-4}$, and $\theta = 0.0$

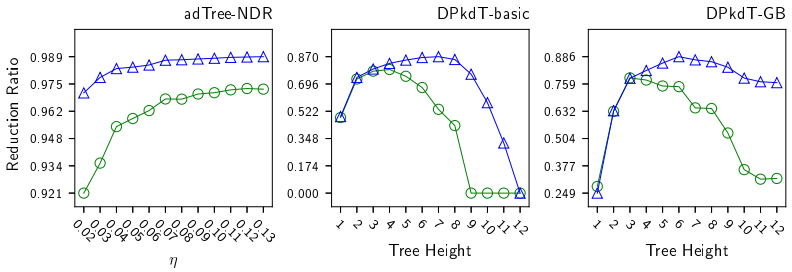


Figure 3.13.: Parameter Tuning when Database Size N = 12,000, $\delta = 10^{-4}$, and $\theta = 0.0$

Figure 3.14 and Figure 3.15 reports the efficiency of the three approaches by varying ϵ and dataset size N. In Figure 3.14, adTree-NDR clearly outperforms the other two approaches with respect to the reduction ratio, in most cases by at least 8.83%.

Table 3.6.: Setting of Tree Height for DPkdT-basic

N \ ϵ	0.2	0.4	0.6	0.8	1.0	1.2	1.4	1.6	1.8	2.0
1,500	1	2	2	2	3	3	3	3	3	3
3,000	2	2	3	3	3	3	3	3	3	4
6,000	2	3	3	3	4	4	4	4	4	4
12,000	3	3	4	4	4	6	6	7	7	7

Table 3.7.: Setting of Tree Height for DPkdT-GB

N \ ϵ	0.2	0.4	0.6	0.8	1.0	1.2	1.4	1.6	1.8	2.0
1,500	1	2	3	3	3	3	3	3	3	3
3,000	1	3	3	3	3	3	3	3	3	3
6,000	2	3	3	3	3	4	4	4	5	5
12,000	3	3	3	6	6	6	6	6	6	6

The efficiency improvement of adTree-NDR is even more obvious, if we evaluate the three approaches in terms of elapsed time. For example, when $N = 6{,}000$, $\epsilon = 1.0$, $\delta = 10^{-4}$, the DPkdT-basic (DPkdT-GB) approach requires more than 87.62 (72.22) hours, whereas the adTree-NDR approach needs less than 14.20 hours, a 6-fold (5-fold) improvement (Figure 3.15). For most ϵ and N values, adTree-NDR is at least twice as fast as the other two approaches.

In addition, we found that the advantage of our proposed scheme over the other two is more evident when the dataset is larger: when $\epsilon = 1.0$, and the dataset size N is 12,000, our scheme takes less than 29.50 hours, whereas DPkdT-basic (DPkdT-GB) needs at least 289.55 (253.87) hours to finish (the rightmost figure in Figure

3.15). Clearly, adTree-NDR is more scalable to the dataset size than the other two approaches.

Furthermore, Figure 3.14 shows that in most cases the difference between DPkdT-basic and DPkdT-GB on the reduction ratio is small (less than 5%) and that a larger dataset size N and a bigger privacy budget ϵ favor DPkdT-GB over DPkdT-basic. Taking a closer look at the generated leaf nodes, we found there are two main reasons for this phenomenon. First, due to the strategies these two approaches adopt to allocate the privacy budget, the amount of privacy budget used to count a leaf node in DPkdT-GB surpasses that of DPkdT-basic, when the tree height of both approaches is greater than or equal to 4, resulting in fewer fake records injected per leaf node by DPkdT-GB than DPkdT-basic. Second, under the same tree height and the same combination of N and ϵ, DPkdT-basic always produces fewer leaf nodes, which has a negative effect on the reduction ratio when more leaf nodes could actually be generated to improve the efficiency. This is attributed to the fact that the quality of the splitting point chosen by DPkdT-basic is worse than that by DPkdT-GB when splitting a parent node into two child nodes. To be precise, we found that DPkdT-basic fairly often chooses a worse splitting point especially on the dimension with very few values in the domain, e.g., sex. In this case, a worse splitting point can only generate one child node, i.e., which consists of all the records from its parent. Recall that unlike DPkdT-GB where the exponential mechanism is adopted to choose a private median, the splitting point selected by DPkdT-basic is based on the Laplace mechanism, where a random noise is drawn to perturb the true count of each possible value in the domain, followed by the computation of private median from those noisy counts. Such an observation demonstrates the advantage of the exponential mechanism for median selection. Although reserving an amount of privacy budget specifically for the exponential mechanism is in general preferable, we can also see that when N is very small, e.g., 1,500, DPkdT-basic is always better than DPkdT-GB, due to a smaller tree height adopted by both approaches, which results in less privacy budget being allocated by DPkdT-GB to count its leaf nodes.

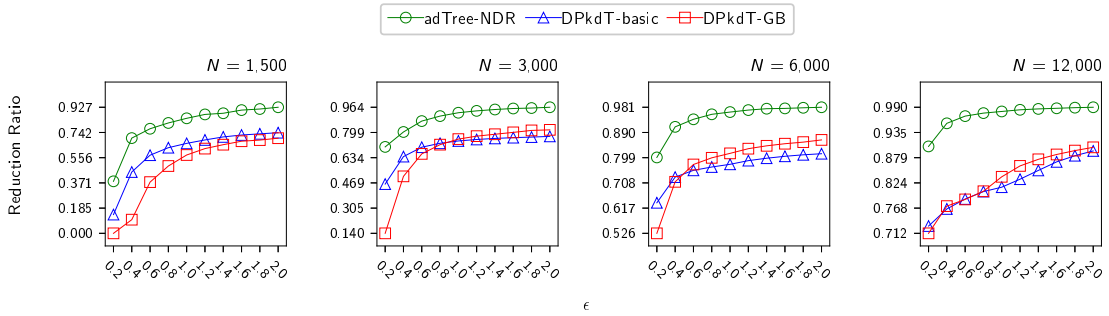


Figure 3.14.: Varying privacy budget ϵ

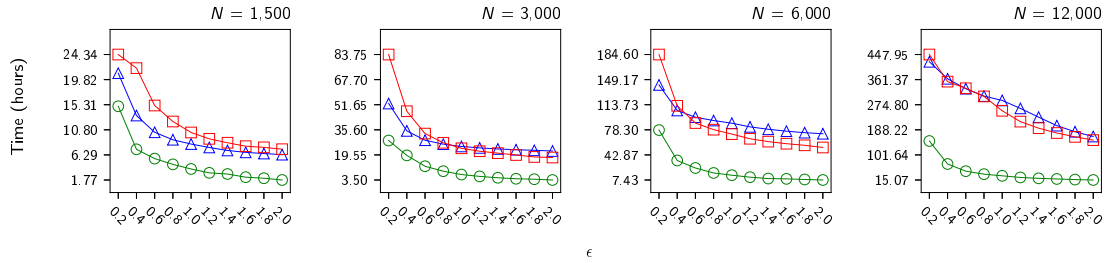


Figure 3.15.: Varying privacy budget ϵ

3.6 Related work

The research community has proposed different approaches to implement private linkage [21]. One category of methods first embed records into vectors before the matching [22–27]. They are developed mainly for private string matching. Specifically, methods [23,25,27] embed records in the Euclidean space. The record matching is thus reduced to the distance computation between points in the Euclidean space. Rather than embedding records in the Euclidean space, Al-Lawati et al. [22] use hash function to transform records into hash signatures, and Karakasidis et al. [24] encode the records using phonetic codes. Schnell et al. [26] instead apply Bloom filters [28] to encode the strings before the actual matching. Those approaches are efficient, but their matching result contains false positives. In addition, the record matching is carried out via the encoded vectors. It is not clear how much information these

vectors disclose. Therefore, the privacy guarantee of those approaches is not well defined either.

Another category of private linkage methods is based on cryptography [2, 29, 30]. One pioneering work in this category is given in [2]. Here the authors apply commutative encryption to carry out equijoin and set intersection with respect to a single linking attribute. Homomorphic cryptosystem like Paillier encryption system [6] is also applied for private record linkage. The methods [29,30] are the representatives. Nevertheless, like [2], they support only exact matching on a single attribute. Compared with the first category of methods [22–27], those approaches provide a well defined privacy guarantee since they are based on securely proven cryptographic schemes. However, they are computationally expensive since they need intensive operations of modular exponentiation.

Hybrid approaches [9, 13, 22] have been proposed to improve the efficiency of cryptography-based private record linkage. They partition records into subsets, and privately match records only in subsets of similar records. As far as we know, Al-Lawati et al. [22] propose the first hybrid approach. Still, their approach does not give a privacy guarantee of information leak by the record partitioning. For this, Inan et al. [9] apply differential privacy [7] to privately partition the input records. Kuzu et al. [13] extend [9] to private string matching. However, as analyzed in Section 3.2, neither of the approaches [9, 13] satisfies differential privacy at the end of private record linkage. The work in [11] is the first that proposes a privacy notion specifically tailored for private record linkage. Their solution provides an end-to-end privacy guarantee for any non-matching record. However, the partitioning scheme used in [11] is not adaptive. Instead, a predefined uniform grid is used to generate the subsets of the given dataset, which may not produce a partitioning optimized for record linkage if data points are not evenly distributed in the data space.

Private indexing schemes [9, 15, 16, 31] are also closely related to our scheme, since they can be used to privately partition data into subsets. Inan et al. [9] build a differentially private kd-tree with a pre-configured tree height. Hay et al. [15]

improve private indexing schemes by enforcing consistency between a parent node and its child nodes. That is, the size of a parent node is equal to the summation of the sizes of its child nodes. Different from [9, 15] that allocate privacy budget flatly to all the levels in the indexing, Cormode et al. [16] adopt a geometric series to allocate the privacy budgets. Still, like [9, 15], it pre-configures the tree height. Qardaji et al. [31] partition the data space into a grid of uniform cells. However, the developed approach is tailored to the 2-dimensional geospatial dataset. It is not clear how to extend it to higher dimensional data with both numerical and categorical attributes. Zhang et al. [32], on the other hand, give a private spatial decomposition algorithm that does not rely on a pre-defined tree height when building the tree. Each time when determining whether or not to further split a given tree node, a carefully calculated positive constant is subtracted from the true count of an internal tree node before the addition of a Laplace noise of some constant scale that is independent of the tree height. Like our proposed spatial decomposition method, their algorithm also allocates half of the privacy budget along any root-to-leaf path for producing the noisy counts of leaf nodes. We leave the evaluation and comparison with this indexing scheme as our future work.

There are also other protocols that introduce differential privacy in the context of secure multi-party computation. Mohammed et al. [33] propose a solution to private data publishing where different attributes for the same set of individuals are entrusted to two curators. Based on Yao's garbled circuits, they [33] give a construction of distributed exponential mechanism that allows two curators to privately determine how the integrated data table should be partitioned. The two curators then participate in a secure scalar product protocol to compute for each subset the cardinality, followed by a two-party distributed Laplace mechanism perturbing the size of each subset. Hong et al. [34] consider the problem of privacy-preserving publication of user search logs collected from different data curators. Their scheme is sampling-based, that is, each data curator independently employs a differentially-private sampling-based process to generate locally synthetic user search logs from his/her own user search logs.

To further improve the resulting utility of the published user logs, they [34] devise a secure protocol that enables multiple curators to compute the global information needed in the sampling process without revealing the local information possessed by any single curator. Unlike this work and [11], [33, 34] do not intentionally leak differentially private statistics to trade for potential efficiency improvement. On the other hand, Mazloom and Gordan [35] propose a framework consisting of two computing servers for performing computation on graph-structured data, where each user is represented by a node in a graph. Mazloom and Gordan [35] observe that it is possible to greatly improve the efficiency by leaking differentially-private node degrees so that the more expensive operations of oblivious sort or ORAM access could be avoided. Like our approach and the one by He et al. [11], the approach by Mazloom and Gordon [35] makes a clear trade-off between controlled statistical leakage and computational efficiency.

This chapter is a major extension to our previous work [10] in that we propose a new framework offering an end-to-end privacy guarantee by incorporating the notion of DPRL into scalable private record linkage, which was not included in [10]. The new framework allows data owners to adopt any adaptive indexing scheme compatible with DPRL to generate the partitionings of their respective datasets. In addition, the compatibility of our private indexing scheme with DPRL is proved and a more extensive experimental evaluation is carried out under the state-of-the-art privacy notion of DPRL.

4 HYBRID PRIVATE STRING MATCHING

We in Chapter 3 proposed an efficient hybrid framework for *numerical* data that combines SMC with differential privacy. Three *semi-honest* [1] parties are involved in this framework: data owners Alice and Bob, and a non-colluding third-party Charlie. First, Alice and Bob partition their respective datasets and send to Charlie private synopses. Based on the synopses, Charlie prunes unnecessary comparisons. However, in many applications the data types of interest are not numerical. For instance, in order to jointly provide services at a lower cost, two companies may want to find common or similar individuals among their customer bases using surname or city of residence, which are strings. Several string comparison metrics exist, such as edit distance, Hamming distance and Jaccard distance. The popular edit distance measures the minimum number of single-character deletions, insertions, and substitutions required to transform one string into the other. One challenge in using the edit distance is that it does not impose a total order, which makes it more difficult for the third party to coordinate the matching. A common practice to enforce a total order on a data set is through embedding. Both data owners agree on a common set of reference objects and transform their data sets into multi-dimensional points in the embedded space. Next, synopses derived from the embedded points are used by the third party to perform the pruning.

The objective of the hybrid approach is to prune as much data as possible in the initial *blocking* phase, which is performed based on the synopses. Therefore, it is important to create an effective set of synopses that accurately represent the underlying data, while at the same time achieving a high blocking ratio. The embedding inherently introduces *distortion*, which may be reduced by using a large number of embedding dimensions. Past research has focused on private publication of high-dimensional data, e.g., [36–40]. In general, given a high-dimensional dataset, to over-

come the curse of dimensionality, the focus is on the publication of low-dimensional marginal tables, i.e., k-way marginals that try to capture the statistical properties of the raw input data. These k-way marginals are suitable for tasks such as range queries and classification, but it is not clear how they could be applied in the context where strings are embedded in a high dimensional space of dimensionality much larger than k.

In this chapter, we propose two privacy-preserving linkage techniques for string-valued data with edit distance. Both solutions rely on the hybrid framework proposed in Chapter 3. Our first solution tackles the challenge of high data dimensionality by employing Lipschitz embeddings and transforming the string-valued data into multi-dimensional points. Each data owner creates an index tree to partition its own data into disjoint subsets, which in turn are sent to the third party for pruning unnecessary secure comparisons. To further reduce the amount of required comparisons, the second technique augments the embedded strings by appending one extra dimension derived from the sanitized string length. Blocking ratio is improved because strings of significantly different lengths also have high edit distance. Extensive experimental results using the DBLP authors dataset indicate that our proposed solutions are able to achieve a high blocking ratio, thus improving considerably matching efficiency. Furthermore, we show that our techniques can be adapted to the recently-proposed protection model of output-constrained differential privacy [11] which is shown to suit well the private dataset matching problem.

Section 4.1 provides necessary background information. We give an overview of our approach in Section 4.2, followed by a detailed description of the algorithms in Section 4.3. We present experimental results in Section 4.4, followed by the extension to the output constrained differential privacy model in Section 4.5. We survey related work in Section 4.6.

4.1 Background

Data Model and Record Linkage Criterion. Alice and Bob share common schema $(A_1, \ldots, A_\omega, A_{\omega+1}, \ldots, A_\ell)$, where only the first ω attributes are used for the linkage. The two parties jointly define the distance functions on the linking attributes. Let $\mathrm{Dom}(A_i)$ be the domain of attribute A_i. The distance function on A_i is defined as:

$$\mathsf{Dist}_i : \mathrm{Dom}(A_i) \times \mathrm{Dom}(A_i) \to \mathbb{R}_0^+.$$

A *decision rule* determines whether or not two records match:

Definition 4.1.1 (Decision Rule) *A decision rule is a predicate that evaluates to* true *if two records match in the linking attributes.*

$$\mathsf{Rule} : \underbrace{\mathbb{R}_0^+ \times \mathbb{R}_0^+ \times \cdots \times \mathbb{R}_0^+}_{\omega} \to \{\mathsf{true}, \mathsf{false}\}. \tag{4.1}$$

The decision rule defined above is general, and must be instantiated according to application requirements. In this chapter, we focus on string linking attributes and the edit distance (or Levenshtein distance) [41], defined as the minimum number of single-character edits, i.e., insertions, deletions, or substitutions needed to transform one string into another.

Instantiation of Decision Rule. Given two records x and y, and a threshold θ of textual dissimilarity for A_i, $1 \leq i \leq \omega$, the decision rule is defined as

$$\mathsf{Rule}(x, y, \theta) = \sum_{i=1}^{\omega} \mathsf{Dist}(x.A_i, y.A_i) \leq \theta, \tag{4.2}$$

where Dist denotes edit distance and θ is application-specific.

Lipschitz Embedding. Let S be a set of objects. A *Lipschitz embedding* [42] is defined in terms of a set R of subsets of S, where $R = \{G_1, G_2, \cdots, G_k\}$. The subsets G_i are called *reference sets*. Let $d(o, G)$ be an extension of the distance function d to a subset $G \subset S$ such that $d(o, G) = \min_{x \in G}\{d(o, x)\}$. An embedding with respect to R is defined as function F such that $F(o) = (d(o, G_1), d(o, G_2), \cdots, d(o, G_k))$.

The Lipschitz embedding preserves information about distance between any objects o_1 and o_2 in S. For any $x \in S$, according to the triangle inequality, $|d(o_1, x) -$

$d(o_2, x)| \leq d(o_1, o_2)$. Hence, $|d(o_1, G) - d(o_2, G)|$ is a lower bound for $d(o_1, o_2)$. Using several reference sets, we can increase the probability that distance $d(o_1, o_2)$ in the original space is captured adequately by $\delta(F(o_1), F(o_2))$ in the embedding space, as defined next.

We define the distance function δ based on a Minkowski metric L_p. Specifically, we denote by k the number of reference sets, and define the distance function δ based on L_p as:

$$\delta(F(o_1), F(o_2)) = \frac{\left(\sum_{i=1}^{k} |d(o_1, G_i) - d(o_2, G_i)|^p\right)^{1/p}}{(k)^{1/p}}. \tag{4.3}$$

It can be proved that δ defined in Eq. 4.3 forms a lower bound on $d(o_1, o_2)$ for any choice of $p \geq 1$ [43]. In addition, for fixed o_1, o_2, the distance $\delta(F(o_1), F(o_2))$ increases monotonically with p. When p approaches ∞, it corresponds to the maximum difference. The use of L_∞ results in the largest value of $\delta(F(o_1), F(o_2))$ and leads to best pruning for similarity computation. Unless otherwise specified, in this chapter, we adopt L_∞ to compute the dissimilarity between objects in the embedding space.

4.2 The Hybrid Scheme

In this section, we provide an overview of our proposed private linkage protocol solution for string-valued attributes, and we identify the various phases of the algorithm. In Section 4.3, we look in detail at the data partitioning strategies we employ to obtain differentially-private synopses. Table 4.1 summarizes the notations used.

Table 4.1.: Summary of Notations

Symbol	Description
ω	Number of linking attributes
A_i	The i-th linking attribute
k_i	Number of reference sets for the i-th linking attribute
k	Total number of reference sets, i.e., $\sum_{i=1}^{\omega} k_i$
$G_{i,j(1\leq j\leq k_i)}$	The j-th reference set of the i-th linking attribute
ℓ_i	The maximum length of the i-th linking attribute
π_i	Number of buckets used for string length of the i-th attribute
$[b_{i,g}^{\ell}, b_{i,g}^{u})_{(1\leq g\leq \pi_i)}$	The g-th extent of lower and upper bounds on the string length along the i-th linking attribute
ϵ	Total privacy budget
$p_{\mathsf{len}}, p_{\mathsf{Lip}}$	Privacy budget fraction for dimensions related to string length and Lipschitz embeddings, resp.
α	Adjustment parameter for the lower bounds w.r.t. Lipschitz embeddings
ξ	Quality parameter for the exponential mechanism
η	Quality parameter for the Laplace mechanism
$C.\beta$	Available privacy budget for a tree node C
$C.s$, $C.\tilde{s}$	True count and noisy count of number of records within a tree node C, resp.
$C.\underline{A}_{i,j}^{\ell}, C.\underline{A}_{i,j}^{u}$	Lower and upper bounds along the j-th dimension for the i-th linking attribute of a tree node C

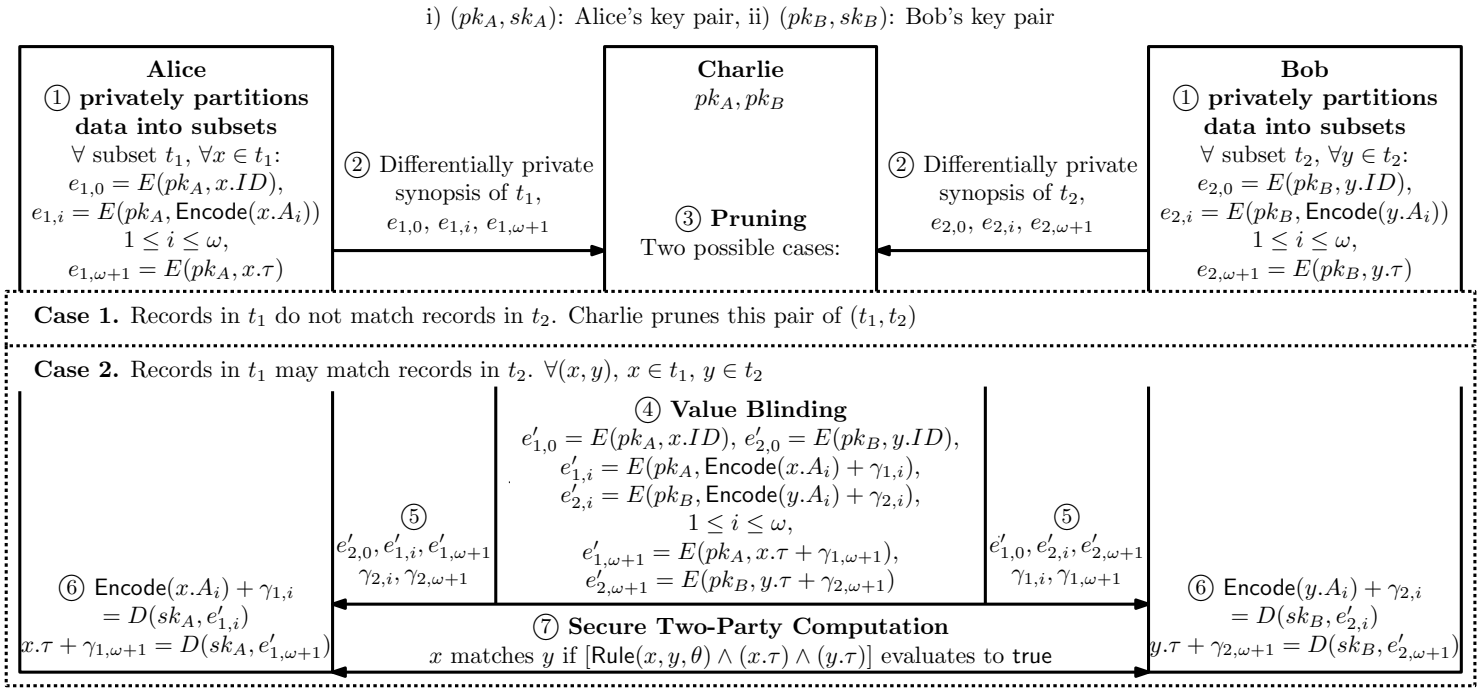


Figure 4.1.: Stages of Proposed Protocol

4.2.1 System Architecture

The work in [10] showed that a hybrid record linkage protocol in which only the two data owners participate does not satisfy differential privacy, because the two parties get access to both the private synopses and the set of matching records, which violates DP. To address this issue, a semi-honest third party is introduced in [10], and our solution follows the same architecture. Alice (A) and Bob (B) are the data owners and mutually agree on a decision rule for matching. Charlie (C) is a third party who coordinates the matching process based on the differentially-private synopses received from A and B. We assume that all parties are *honest-but-curious* and model them as polynomial-time Turing machines. They strictly conform to the protocol execution, but they attempt to infer additional knowledge from what they observe during matching. Moreover, we assume that Charlie does not collude with any of the data owners.

A and B release to C differentially-private synopses of their data. Using the synopses, C infers that some of the record pairs cannot match, and excludes them from the SMC matching phase, thus saving cost. Due to the privacy constraints, it is possible for false negatives to exist, but as we show later on in Section 4.4 the recall obtained is high. A and B then directly engage in the SMC phase to determine whether the remaining record pairs match or not. C does not gain access to matching records, hence DP is preserved. Next, we specify the input, output, and security requirements of our protocol.

Protocol Input from Each Party

- For each of A, B: Input records, the differentially-private synopses of the records (see Section 4.3), and a Paillier cryptosystem key pair (one for each).
- C: The public Paillier keys of A and B.

Protocol Output to Each Party

- A, B: The matching records (except for some false negatives due to the DP constraint), and the total number of record pairs included in the SMC phase.

- C: Differentially-private synopses from A and B.

Security Requirements

- A and B learn no additional information except for their protocol output.
- C's knowledge is restricted to the differentially-private synopses released by A and B.

Figure 4.1 provides an overview of our protocol, which consists of three main phases: *pre-processing* (Section 4.2.1.1), *pruning* (Section 4.2.1.2) and *record matching via SMC* (Section 4.2.1.3).

4.2.1.1 Pre-processing

A and B each partition their records into disjoint subsets, and generate for each set a DP-compliant synopsis. To achieve DP, a subset has to either be enlarged by adding dummy records, or shrunk by suppressing existing records. Next, A and B encrypt their records (including the dummy ones) under their respective public Paillier keys. The private synopses and encrypted records are sent to C.

Definition 4.2.1 (Private Synopsis) *Let t be a subset of records. The private synopsis of t contains: 1) the extent of t's records in the linking attributes space, determined according to differential privacy, and 2) the differentially private cardinality of t .*

Our focus is on string attributes, for which a total order does not exist according to the edit distance. To allow pruning in the absence of a total order, we adopt a common practice to embed strings before matching [25, 27]. We use the Lipschitz embedding, which allows C to compute the lower bound on the edit distance between strings located in different subsets.

Generation of Random Reference Sets. Lipschitz embedding requires parties to agree, for each linking attribute A_i, on a set of randomly generated reference sets

$G_{i,j}$, $1 \leq j \leq k_i$. In our scheme, the two parties first agree on the maximum length ℓ_i for each linking attribute A_i. Then, for each of those k_i reference sets, random strings of length ranging from 1 to ℓ_i are generated. This way, the coordinate value for attribute A_i will lie in the range $[0, \ell_i + 1)$.

Definition 4.2.2 (Extent of a Subset for String Attribute) *Let t be a set of records. For a string-valued linking attribute A_i, $1 \leq i \leq \omega$, assume that the string attribute A_i of each record $x \in t$ has been embedded in a k_i-dimensional space via an embedding scheme. The extent of t on attribute A_i is defined as $\{[\underline{A}^{\ell}_{i,j}, \underline{A}^{u}_{i,j}) \mid 1 \leq j \leq k_i\}$, where $\underline{A}^{\ell}_{i,j}$ and $\underline{A}^{u}_{i,j}$ denote the lower and upper bounds along the j-th dimension for attribute A_i in the embedding space, respectively.*

After generating private synopses for each subset, A and B encrypt the records within each of their respective subsets. The plaintext space of the Paillier cryptosystem is $\{0, 1, \cdots, n-1\}$, where n is the encryption function modulus. String-valued linking attributes must be encoded into this space, as shown next.

Record Encryption. Recall that A and B use Paillier cryptosystem [6] to encrypt records. Party A generates a public-private key pair (pk_A, sk_A). Let t_1 be the current subset of A's data to be encoded, and denote by $x \in t_1$ a record. A computes

$$\begin{cases} e_{1,0} & = E(pk_A, x.ID) \\ e_{1,i} & = E(pk_A, \mathsf{Encode}(x.A_i)),\ 1 \leq i \leq \omega \\ e_{1,\omega+1} & = E(pk_A, x.\tau), \end{cases} \tag{4.4}$$

where $x.\tau = \mathsf{true}$ if x is an actual record and false if x is a dummy record added to satisfy DP. B follows the same procedure. Let t_2 be a subset of B's data, and $y \in t_2$ be a record. B generates key pair (pk_B, sk_B) and encrypts a record as follows:

$$\begin{cases} e_{2,0} & = E(pk_B, y.ID) \\ e_{2,i} & = E(pk_B, \mathsf{Encode}(y.A_i)),\ 1 \leq i \leq \omega \\ e_{2,\omega+1} & = E(pk_B, y.\tau), \end{cases} \tag{4.5}$$

where $y.\tau = \mathsf{true}$ if y is an actual record and false otherwise.

4.2.1.2 Pruning Secure Comparisons at the Third Party

Let t_1 and t_2 be two subsets of A and B, respectively. C applies the decision rule to determine whether t_1 and t_2 may contain matching records. Specifically, C computes the lower bound on the edit distance for each linking attribute. Then, according to the decision rule in Eq. 4.2, the subset pair is pruned from further SMC matching if the sum of the lower bounds exceeds the matching threshold θ.

For linking attribute A_i, let $h_{1,i,j} = [\underline{A}^{\ell}_{1,i,j}, \underline{A}^{u}_{1,i,j})$ and $h_{2,i,j} = [\underline{A}^{\ell}_{2,i,j}, \underline{A}^{u}_{2,i,j})$ be the extents of the embedded records in t_1 and t_2 along the j-th embedding dimension for attribute A_i. C computes $\pi_j = \min\{|v_1 - v_2| \mid v_1 \in h_{1,i,j}, v_2 \in h_{2,i,j}\}$, for $1 \leq j \leq k_i$ and chooses the largest π_j (since L_∞ is used). The lower bound on the edit distance is

$$\delta^{\ell}_i = \max\{\pi_j \mid 1 \leq j \leq k_i\}. \tag{4.6}$$

Given δ^{ℓ}_i, $1 \leq i \leq \omega$, for those subsets t_1 and t_2 having $\sum_{i=1}^{\omega} \delta^{\ell}_i > \theta$, the comparisons between any records $x \in t_1$ and $y \in t_2$ are pruned.

4.2.1.3 Record Matching via SMC

After pruning, Alice and Bob engage in SMC matching for the remaining sets of record pairs. First, C must blind attribute values in each subset to prevent additional disclosure to A and B, as explained next. Finally, A and B engage in a secure two-party computation protocol to determine matching record pairs.

Value Blinding. In the absence of attribute value blinding, when record x does not match record y, A is able to infer that B's record y in the current instance of secure computation is similar to record x, because x is seen by A in the clear. To prevent this disclosure, C blinds attributes of both parties with large random integers. Let $x \in t_1$, and $y \in t_2$ be two records of A and B, respectively, and let κ be a statistical security parameter. C uses random numbers of appropriate bit-lengths to blind the linking attributes and field τ of x and y. Specifically, for the i-th linking attribute A_i, C generates a random integer uniformly distributed in $[0, 2^{\rho_i} - 1]$, where

$\rho_i = \kappa + \lfloor \log_2(U_i) \rfloor + 1$, and U_i denotes the upper bound of $\mathsf{Encode}(A_i)$. The field τ has to be blinded as well. The possible values of τ could be 1 or 0, denoting true or false, respectively. We denote by $U_{\omega+1} = 1$ the upper bound for this field, and we use a random integer from $[0, 2^{\rho_{\omega+1}} - 1]$ to hide the value of the field.

Specifically, C randomizes A's encrypted records by computing

$$\begin{cases} e'_{1,i} & = e_{1,i} \cdot E(pk_A, \gamma_{1,i}), 1 \leq i \leq \omega \\ e'_{1,\omega+1} & = e_{1,\omega+1} \cdot E(pk_A, \gamma_{1,\omega+1}), \end{cases} \tag{4.7}$$

where $\gamma_{1,i}$ is a random integer drawn from $[0, 2^{\rho_i} - 1]$. By the homomorphic property of the Paillier cryptosystem, $e'_{1,i}$, $e'_{1,\omega+1}$ will be ciphertexts for $(\mathsf{Encode}(x.A_i) + \gamma_{1,i})$, and $(x.\tau + \gamma_{1,\omega+1})$, respectively.

To produce ciphertexts of $(\mathsf{Encode}(y.A_i) + \gamma_{2,i})$, and $(y.\tau + \gamma_{2,\omega+1})$, C randomizes B's encrypted records as follows:

$$\begin{cases} e'_{2,i} & = e_{2,i} \cdot E(pk_B, \gamma_{2,i}), 1 \leq i \leq \omega \\ e'_{2,\omega+1} & = e_{2,\omega+1} \cdot E(pk_B, \gamma_{2,\omega+1}), \end{cases} \tag{4.8}$$

with $\gamma_{2,i}$, $\gamma_{2,\omega+1}$ randomly selected as described above.

C also re-encrypts the encrypted identifiers of x and y

$$\begin{cases} e'_{1,0} = e_{1,0} \cdot E(pk_A, 0) \\ e'_{2,0} = e_{2,0} \cdot E(pk_B, 0). \end{cases} \tag{4.9}$$

C sends back to A $e'_{2,0}, (e'_{1,1}, \cdots, e'_{1,\omega+1})$, $(\gamma_{2,1}, \cdots, \gamma_{2,\omega+1})$, and to B $e'_{1,0}, (e'_{2,1}, \cdots, e'_{2,\omega+1})$, $(\gamma_{1,1}, \cdots, \gamma_{1,\omega+1})$. In other words, A and B each receive the encrypted record identifiers of their counterparts. Furthermore, they receive their own blinded record attributes, and the masks used to blind their counterpart's record attributes.

Secure Two-Party Computation Step. Finally, A and B jointly decide if x matches y according to the randomized encrypted records and the corresponding random masks received from C. This step is implemented using Yao's garbled circuits. In short, being able to decrypt $e'_{1,i}$ $(e'_{2,i})$, $1 \leq i \leq \omega + 1$, A and B are able to remove the random masks added by C *within* the circuit. This is because $D(sk_A, e'_{1,i}) - \gamma_{1,i} = \mathsf{Encode}(x.A_i)$, and $D(sk_B, e'_{2,i}) - \gamma_{2,i} = \mathsf{Encode}(y.A_i)$, $1 \leq i \leq \omega$.

Similarly, the random masks added to hide $x.\tau$ ($y.\tau$) are removed within the circuit since $D(sk_A, e'_{1,\omega+1}) - \gamma_{1,\omega+1} = x.\tau$ and $D(sk_B, e'_{2,\omega+1}) - \gamma_{2,\omega+1} = y.\tau$. The encoded records x and y are then obliviously compared using the pre-defined decision rule. In addition to the circuit for secure comparison adopted from [10], a more complex circuit has to be constructed to enable the evaluation of string similarity. The circuit must evaluate the conjunction $[\mathsf{Rule}(x, y, \theta) \wedge (x.\tau) \wedge (y.\tau)]$, i.e., only the true matching pairs (x, y) satisfying the predicate would yield an output of true. In the case of matching, A sends to B $e'_{2,0}$, i.e., the encryption of $y.ID$, so that B can determine the ID of the matching record $y.ID$ in B's own dataset.

Security Analysis. Recall that in the steps of SMC, the only place where the attribute values are not protected by encryption is in the step of value blinding, where A and B observe the masked attribute values in the clear. However, we note that these attributes are still protected in that the above setting of $\gamma_{1,i}$ and $\gamma_{2,i}$ guarantees that the statistical distance between $\mathsf{Encode}(x.A_i) + \gamma_{1,i}$ and an integer drawn uniformly at random from $[0, 2^{\rho_i} - 1]$ is upper bounded by $1/2^{\kappa}$, i.e., with probability $1 - 1/2^{\kappa}$, the blinded attributes cannot be distinguished from a random number in the range $[0, 2^{\rho_i} - 1]$. A similar guarantee holds for $\mathsf{Encode}(y.A_i) + \gamma_{2,i}$ as well.

4.3 Data Partitioning

This section describes in detail how to create DP-compliant synopses for pruning non-matching record pairs. We propose two algorithms that partition records into disjoint subsets. The first one uses only the Lipschitz embeddings of data records, whereas the second algorithm also takes into account sanitized string lengths. Both algorithms follow a similar two-stage approach, where each stage is assigned a pre-specified amount of privacy budget. The first stage creates a DP-compliant data-dependent hierarchical index structure, whereas the second stage releases noisy (i.e., private) counts of records inside each leaf node of the index.

The first stage relies on the exponential mechanism (*EM*) of DP, and iteratively selects a split point along one of the string-attribute dimensions to grow the hierarchical index. The second stage employs the Laplace mechanism (*LM*) to compute noisy counts. However, we have found that using generic EM and LM instantiations leads to poor results, i.e., low blocking ratios and many false negatives. To alleviate these shortcomings, we first propose two revised versions of EM and LM customized to our problem setting.

4.3.1 EM with Predefined Output Quality

As seen in Section 4.1, given a quality function q and a set $\mathcal{R}$ of all possible output values, EM selects a result $r \in \mathcal{R}$ with a probability exponentially proportional to the quality score of r. If the quality function q is designed to output the quality score from a pre-defined domain, e.g., 0 to $-s$, with 0 being the highest possible score and $-s$ being the lowest possible score, then the ratio of the probability that the best solution is sampled to the probability that the worst solution is sampled is $\xi = \exp((\epsilon \cdot 0)/(2S(q)))/\exp((\epsilon \cdot (-s))/(2S(q))) = \exp(\epsilon \cdot s/(2S(q)))$, where $S(q)$ denotes the sensitivity of the quality function. The larger ξ is, the more effective the EM is in selecting a quality output. Hence, given a user-specified quality parameter ξ, the budget ϵ required by EM can be estimated as:

$$\exp\left\{\frac{\epsilon \cdot s}{2S(q)}\right\} = \xi \Rightarrow \epsilon = \frac{2S(q) \cdot \ln \xi}{s}. \tag{4.10}$$

We employ EM to determine a good splitting point for partitioning. Given a set C of records with noisy count $C.\tilde{s}$, and the j-th dimension of the i-th linking attribute having the extent $[\underline{A}^{\ell}_{i,j}, \underline{A}^{u}_{i,j})$, EM will privately sample a point sp. Records having an embedding coordinate less than sp along the splitting dimension are placed into the left subset, and the rest are placed into the right subset. We aim to identify an sp value that results in an even partitioning. We define the quality score of sp as $q(C, sp) = -|C_{L,sp}.s - C_{R,sp}.s|$, where $C_{L,sp}.s$ and $C_{R,sp}.s$ represent the numbers of records falling into the left and the right subsets according to sp, respectively.

The highest possible score of $q(C, sp)$ corresponds to a splitting point that separates C into two equal-sized subsets, i.e., $q(C, sp) = 0$, whereas the lowest possible score corresponds to a splitting point placing all records in a single set, with quality score $-C.s$. Note that, due to the DP constraint, $C.s$ is not directly available so we use instead the noisy value $C.\tilde{s}$. The sensitivity $S(q) = 1$, since the maximum change to the output of q due to the removal or the insertion of a record is at most 1. Thus, we estimate the amount of privacy budget allocated to EM as $2 \ln \xi / C.\tilde{s}$. In total, we have $k = \sum_{i=1}^{\omega} k_i$ dimensions for Lipschitz embeddings. We order those dimensions as $\underline{A}_1, \underline{A}_2, \cdots, \underline{A}_k$. We use $\mathsf{Select}(C.\underline{A}_m^{\ell}, C.\underline{A}_m^{u}, \epsilon)$ to denote an instance of EM that uses budget ϵ to select a splitting point for a tree node C having the extent $[C.\underline{A}_m^{\ell}, C.\underline{A}_m^{u})$ along the dimension $\underline{A}_m$.

4.3.2 LM with Controlled Noise Magnitude

The work in [10] proposed a strategy to determine the amount of privacy budget used to count the number s of records within a given subset. Specifically, the ratio of the magnitude of Laplace noise to s is bounded by a threshold η. Let $\mathsf{Lap}(\lambda)$ be a Laplace random variable with mean 0 and scale λ. Then, we require that:

$$\frac{\mathbb{E}(|\mathsf{Lap}(\lambda)|)}{s} \leq \eta. \tag{4.11}$$

The expected absolute value of the Laplace noise equals λ, and to achieve ϵ-differential privacy when counting a given set, the scale has to be set to $\lambda = S(f)/\epsilon$, where $S(f)$ signifies the sensitivity of f. In our case, $S(f) = 1$ so we rewrite the inequality above as

$$\epsilon \geq \frac{1}{s \cdot \eta}. \tag{4.12}$$

After partitioning a set C into subsets $C_{L,sp}$ and $C_{R,sp}$ according to an sp privately selected by EM, we must estimate the sizes of the resultant subsets. We replace s in Eq. 4.11 with $C.\tilde{s}/2$ under the assumption that the splitting point selected yields an even partitioning. The required amount of privacy budget is thus $2/(C.\tilde{s} \cdot \eta)$.

4.3.3 Lipschitz Embedding-based Partitioning

Algorithm 4.1 details the **DP-adTree-Lip** method which partitions data based on Lipschitz embeddings only. The records are recursively partitioned, starting with the root node that covers the entire dataspace. The data owner specifies the proportion p_{Lip} of the privacy budget used to determine split points and internal node counts at each index level. The rest of the budget $\epsilon_{\mathsf{leaf}} = \epsilon \cdot (1 - p_{\mathsf{Lip}})$ is used to compute noisy leaf counts.

Algorithm 4.1 (DP-adTree-Lip)

Input: dataset D, ϵ, p_{Lip}, ξ, η, $k = \sum_{i=1}^{\omega} k_i$

Output: tree T and its noisy leaf node counts

$\epsilon_{\mathsf{Lip}} \leftarrow \epsilon \cdot p_{\mathsf{Lip}}$; $\epsilon_{\mathsf{leaf}} \leftarrow \epsilon \cdot (1 - p_{\mathsf{Lip}})$
$r \leftarrow \mathsf{Create}$(records in D)
$r.\tilde{s} \leftarrow |D|$; $r.\beta \leftarrow \epsilon_{\mathsf{Lip}}$
$T \leftarrow \textsc{GrowTree}(r, \epsilon_{\mathsf{leaf}}, \xi, \eta, k)$
procedure $\textsc{GrowTree}(r, \epsilon_{\mathsf{leaf}}, \xi, \eta, k)$
    $T \leftarrow \{\}$, FIFO Queue $\mathbb{Q} \leftarrow \{r\}$
    while $(C \leftarrow \mathbb{Q}.\text{remove}()) \neq NIL$ **do**
        $\epsilon_{\mathsf{med}} \leftarrow 2 \ln \xi / C.\tilde{s}$; $\epsilon_{\mathsf{cnt}} \leftarrow 2/(C.\tilde{s} \cdot \eta)$
        if $C.\tilde{s} > 0$ **and** $C.\beta \geq \epsilon_{\mathsf{med}} + \epsilon_{\mathsf{cnt}}$ **then**
            $(C_L, C_R) \leftarrow \textsc{Split}(C, \epsilon_{\mathsf{med}})$
            $C_L \leftarrow \textsc{Count}(C_L, \epsilon_{\mathsf{cnt}})$
            $C_R \leftarrow \textsc{Count}(C_R, \epsilon_{\mathsf{cnt}})$
            Add C_L, C_R to $\mathbb{Q}$
        else if $C.\tilde{s} > 0$ **and** $C.\beta \geq \epsilon_{\mathsf{med}}$ **then**
            $(C_L, C_R) \leftarrow \textsc{Split}(C, \epsilon_{\mathsf{med}})$
            $C_L \leftarrow \textsc{Count}(C_L, C_L.\beta + \epsilon_{\mathsf{leaf}})$
            $C_R \leftarrow \textsc{Count}(C_R, C_R.\beta + \epsilon_{\mathsf{leaf}})$
            Add C_L, C_R to T
        else
            $C \leftarrow \textsc{Count}(C, C.B + \epsilon_{\mathsf{leaf}})$
            Add C to T
        end if
    end while
    return T
end procedure

Algorithm 4.1 (DP-adTree-Lip) (Continued)

26: **procedure** SPLIT(C, ϵ)
27: $m \leftarrow (\mathsf{Depth}(C) \bmod k) + 1$
28: $sp \leftarrow \mathsf{Select}(C.\underline{A}^{\ell}_m, C.\underline{A}^{u}_m, \epsilon)$
29: $C_L \leftarrow \mathsf{Create}(\text{records in } C \text{ with } \underline{A}_m < sp)$
30: $C_R \leftarrow \mathsf{Create}(\text{records in } C \text{ with } \underline{A}_m \geq sp)$
31: $C_L.\beta \leftarrow C.\beta - \epsilon$; $C_R.\beta \leftarrow C.\beta - \epsilon$
32: **return** (C_L, C_R)
33: **end procedure**
34: **procedure** COUNT(C, ϵ)
35: $C.\tilde{s} \leftarrow C.s + \mathsf{Lap}(1/\epsilon)$; $C.\beta \leftarrow C.\beta - \epsilon$
36: **return** C
37: **end procedure**

After each split, the remainder of the budget is passed to the children. The recursion stops when a node does not have sufficient budget to perform a split. Specifically, we check if the current node's privacy budget $C.\beta$ is larger than the bounds given in Eqs. 4.10 and 4.11. There are three possible outcomes: (i) if $C.\beta$ is sufficient for both splitting the current node and counting the resulting child nodes (lines 10-13), then we invoke SPLIT and COUNT, and then add the resulting nodes C_L, C_R to queue $\mathbb{Q}$; (ii) if the budget is only enough for the split operation (lines 15-18), we invoke SPLIT and then proceed to release noisy leaf node counts; (iii) otherwise (lines 20-21), we do not split the current node (i.e., current node is a leaf), and we release its noisy count.

4.3.4 Length-Aided Partitioning

We describe next the **DP-adTree** method (Algorithm 4.2), which uses sanitized string length information to improve the quality of Lipschitz-based partitioning. The

data owner must additionally specify the proportion p_{len} of privacy budget allocated for the length-based partitioning, summarized in procedure PartitionByLength.

Specifically, let $\mathcal{B} = \{B_i \mid 1 \leq i \leq \omega\}$ represent a set of string length buckets for attribute A_i, where $B_i = \{[b^{\ell}_{i,g}, b^{u}_{i,g}) \mid 1 \leq g \leq \pi_i\}$. Given dataset D and $\mathcal{B}$, procedure PartitionByLength starts with a root node (at depth 0) which contains all records in D and builds ω tree levels such that a node at depth $i-1$ produces π_i child nodes $\mathcal{C}_i = \{C_1, \cdots, C_{\pi_i}\}$. The g-th child node contains those records for which the i-th linking attribute has length between $b^{\ell}_{i,g}$ and $b^{u}_{i,g} - 1$. Procedure CreatePartitions creates the child nodes, and augments the extent of each child node at level i by prepending an additional dimension $\underline{A}_{i,0}$. The lower and upper bounds along this dimension are set for the g-th child to $b^{\ell}_{i,g}$ and $b^{u}_{i,g}$, respectively. This is equivalent to creating one additional reference set $G_{i,0} = \{\phi^{v'} \mid 1 \leq v' \leq \ell_i\}$ for the i-th attribute, consisting of ℓ_i reference strings, where ℓ_i is the maximum length of attribute A_i, and ϕ^v is the concatenation of v special characters ϕ (ϕ is not part of the alphabet for strings in attribute A_i). We observe that for a string o of length v upper bounded by ℓ_i, $d(o, G_{i,0}) = \min_{x \in G_{i,0}}\{d(o,x)\} = d(o, \phi^{v'}) = v$ with $1 \leq v' \leq v$.

After the length-based partitioning is completed, we compute the noisy count C_i for each subset (line 6), and we invoke procedure AdjustBounds which updates the lower and upper bounds of each subset along the dimensions of Lipschitz embeddings according to the range of string lengths within the subset. Recall from Section 4.2.1.1 that we generate reference sets for attribute A_i as random strings of length from 1 to ℓ_i, where ℓ_i is the maximum length of strings for attribute A_i agreed on by both parties. By doing so, it is possible to further reduce the extents with respect to the Lipschitz embeddings after the invocation of PartitionByLength.

Specifically, after range $[C.\underline{A}^{\ell}_{i,0}, C.\underline{A}^{u}_{i,0})$ of attribute A_i within subset C is known, it is certain that the upper bounds along any other embedded dimensions $\underline{A}^{u}_{i,j}$, $1 \leq j \leq k_i$ cannot exceed $C.\underline{A}^{u}_{i,0}$, since for any record having its i-th attribute o in C, its j-th dimension of the Lipschitz embeddings satisfies inequality $d(o, G_{i,j}) = \min_{x \in G_{i,j}}\{d(o,x)\} \leq \mathsf{strlen}(o)$. The inequality holds because for any random reference

string $x \in G_{i,j}$ with $\mathsf{strlen}(x) \leq \mathsf{strlen}(o)$, we have $d(o, x) \leq \mathsf{strlen}(o)$, and in $G_{i,j}$ there is at least one random reference string x of the same length as o. In the worst case, the minimum distance between o and any reference string x cannot be greater than $\mathsf{strlen}(o)$. In other words, such an update to the upper bound $C.\underline{A}^{u}_{i,j}$ takes into account the correlation between the length of a string and its corresponding Lipschitz embedding.

However, there is no deterministic lower bound on the value of $C.\underline{A}^{\ell}_{i,j}$ that could be inferred via the lower bound on the string length $C.\underline{A}^{\ell}_{i,0}$. According to [44], the average edit distance between two random strings of length v constructed from an alphabet of σ characters is at least $v(1 - e/\sqrt{\sigma})$, where e denotes Euler's number. Substituting $\sigma = 100$ into the formula, we see that $(1-e/\sqrt{\sigma}) \geq 0.728$, indicating that on average, there will not be too many matching characters between the two strings, resulting in an edit distance that is only slightly smaller than v. Since the reference strings in our scheme are randomly generated and the number of reference strings of length v in each reference set is strictly smaller than $O(\sigma^v)$, as a rule of thumb, the data owner could specify an adjustment parameter α close to 1 and update the lower bound on the j-th dimension $C.\underline{A}^{\ell}_{i,j}$ by $\alpha \cdot C.\underline{A}^{\ell}_{i,0}$. We note that the determination of the value α does not rely on the data owner's records and thus the adjustment to the lower bound is data-independent.

Algorithm 4.2 (DP-adTree)

Input: dataset D, ϵ, p_{len}, p_{Lip}, set of length buckets $\mathcal{B}$, ξ, η, k, α

Output: tree T and its noisy leaf node counts

$T \leftarrow \{\}$
$\epsilon_{\mathsf{len}} \leftarrow \epsilon \cdot p_{\mathsf{len}}$; $\epsilon_{\mathsf{Lip}} \leftarrow \epsilon \cdot p_{\mathsf{Lip}}$
$\epsilon_{\mathsf{leaf}} \leftarrow \epsilon \cdot (1 - p_{\mathsf{len}} - p_{\mathsf{Lip}})$
$\mathcal{C} \leftarrow \textsc{PartitionByLength}(D, \mathcal{B})$
for all $C_i \in \mathcal{C}$ **do**
    $C_i \leftarrow \textsc{Count}(C_i, \epsilon_{\mathsf{len}})$
    $C_i \leftarrow \textsc{AdjustBounds}(C_i, \alpha)$; $C_i.\beta \leftarrow \epsilon_{\mathsf{Lip}}$
    $T_i \leftarrow \textsc{GrowTree}(C_i, \epsilon_{\mathsf{leaf}}, \xi, \eta, k)$
    $T \leftarrow T \cup T_i$
end for
procedure $\textsc{PartitionByLength}(D, \mathcal{B})$
    $\mathcal{C} \leftarrow \{\}$; $r \leftarrow \mathsf{Create}(\text{records in } D)$
    Add r to an empty FIFO Queue $\mathbb{Q}$
    while $\mathbb{Q} \neq \{\}$ **do**
        $C \leftarrow \mathbb{Q}.\text{remove}()$
        $i \leftarrow \mathsf{Depth}(C) + 1$
        $\mathcal{C}_i \leftarrow \textsc{CreatePartitions}(C, B_i)$
        if $\mathsf{Depth}(C) < \omega$ **then**
            $\mathbb{Q}.\text{addAll}(\mathcal{C}_i)$
        else
            $\mathcal{C} \leftarrow \mathcal{C} \cup \mathcal{C}_i$
        end if
    end while
    return $\mathcal{C}$
end procedure

Algorithm 4.2 (DP-adTree) (Continued)

procedure CREATEPARTITIONS(C, B_i)
  $\mathcal{C}_i \leftarrow \{\}$
  for $g \leftarrow 1$ **to** π_i **do**
    $C_g \leftarrow$ Create(records in C with A_i's length in $[b^{\ell}_{i,g}, b^{u}_{i,g})$)
    $C_g.\underline{A}^{\ell}_{i,0} \leftarrow b^{\ell}_{i,g}$; $C_g.\underline{A}^{u}_{i,0} \leftarrow b^{u}_{i,g}$
    Add C_g to $\mathcal{C}_i$
  end for
  return $\mathcal{C}_i$
end procedure
procedure ADJUSTBOUNDS(C, α)
  for $i \leftarrow 1$ **to** ω **do**
    for $j \leftarrow 1$ **to** k_i **do**
      $C.\underline{A}^{\ell}_{i,j} \leftarrow \alpha \cdot C.\underline{A}^{\ell}_{i,0}$; $C.\underline{A}^{u}_{i,j} \leftarrow C.\underline{A}^{u}_{i,0}$
    end for
  end for
  return C
end procedure

4.3.5 Privacy Analysis

We emphasize that both Algorithm 4.1 and Algorithm 4.2 satisfy ϵ-DP. First, we notice that given any partitioning tree, addition or removal of any single record r to or from the input dataset only affects the results returned by procedures SPLIT and COUNT along the root-to-leaf path in which this record r is involved. Now, suppose that the root-to-leaf path involving r consists of nodes $C_0, C_1, C_2, \ldots, C_d$, where C_0 represents the root node covering all the records and C_d corresponds to the leaf that contains r. It is not hard to see that the consumption of privacy budget along such a path is ϵ in both algorithms. To be more precise, let $x_{i,\mathsf{med}}$ and $x_{i,\mathsf{cnt}}$ be the amounts of privacy budget consumed by node C_i due to the invocations of SPLIT and COUNT,

respectively. We have that $\sum_{i=0}^{d-1}(x_{i,\mathsf{med}} + x_{i,\mathsf{cnt}}) = \epsilon - \epsilon_{\mathsf{leaf}} - \epsilon'$, $x_{d,\mathsf{med}} = 0$, and that $x_{d,\mathsf{cnt}} = \epsilon' + \epsilon_{\mathsf{leaf}}$, where ϵ' denotes the privacy budget left over when there is not enough budget for further growing the internal nodes. According to the composability of DP [8], queries along any such path satisfy $\sum_{i=0}^{d}(x_{i,\mathsf{med}} + x_{i,\mathsf{cnt}})$-DP, hence both algorithms satisfy ϵ-DP.

4.4 Experiments

We evaluate empirically our approach using a subset of the popular DBLP dataset [45]. There are 205,404 entries in this dataset under the category of *inproceedings*, containing bibliographic information like title, authors, and venue logged for each publication. We use as linking attributes the names of the first two authors, i.e., $\omega = 2$, and we set $\ell = \ell_1 = \ell_2 = 39$ to be the upper bound on name length. For publications with at least two authors each consisting of at most $\ell = 39$ characters, we randomly draw 10,000, 20,000, 30,000, 40,000, and 50,000 records to produce the two datasets for parties Alice and Bob, respectively. For each set S of these sampled records, we randomly permute S and split it into three approximately equal-sized disjoint subsets d_1, d_2, and d_3. The union of subsets d_1 and d_2 constitutes the dataset of Alice, and $d_2 \cup d_3$ forms the dataset of Bob. For each linking attribute A_i, $1 \leq i \leq 2$, both parties agree on $k_i = 1$ reference sets consisting of 80 random strings of length ranging from 1 to $\ell = 39$. The proportion p_{Lip} is set to 0.65 in Algorithm 4.1, whereas $p_{\mathsf{Lip}} = 0.60$ in Algorithm 4.2 with $p_{\mathsf{len}} = 0.05$. Table 4.2 summarizes parameter settings, with default values in boldface.

For length-based partitioning, we set $\mathcal{B} = \{B_1, B_2\}$ with $B_i = \{[0, 11), [11, 13), [13, 14), [14, 15), [15, 16), [16, 17), [17, 19), [19, 40)\}$ for each linking attribute A_i (i.e., in Algorithm 4.2 records are partitioned into $\pi_1 = \pi_2 = 8$ disjoint subsets for each attribute). We focus on two metrics: *reduction ratio* and *recall rate*. Let D_A and D_B be the datasets of Alice and Bob, respectively. Without blocking, a total of $|D_A| \times |D_B|$ secure comparisons are required to identify matching records. Denote by

Table 4.2.: Experimental Parameter Settings

Parameter	Values
$\|S\|$	**10k**, $20k, 30k, 40k, 50k$
ω	**2**
θ	$0, 1, 2, 3$
k_1, k_2	**1**
ℓ_1, ℓ_2	**39**
π_1, π_2	**8**
ϵ	$0.2, 0.4, \mathbf{0.6}, 0.8, 1.0$
p_{len}	0.05
p_{Lip}	$0.60, 0.65$
α	**0.9**

n the number of secure comparisons needed by a hybrid approach. Furthermore, let M_A be the number of true matching record pairs and M_O the number of matching pairs output by the hybrid approach. Reduction and recall are defined as:

$$reduction = 1 - \frac{n}{|D_A| \times |D_B|}; recall = \frac{M_O}{M_A}$$

We implemented Algorithm 4.1, denoted by Lipschitz-Only (LO), and two configurations of Algorithm 4.2: the results produced without AdjustBounds are denoted as Length-Guided-N (LGN), whereas Length-Guided-W (LGW) represents the results when AdjustBounds is invoked. Results in each experiment are averaged over 30 distinct random seeds.

In our first experiment, we set $|S|$ to 10,000 and measure the accuracy of our algorithms for several matching threshold settings (θ) when varying privacy budget ϵ. Figure 4.2 shows how reduction ratio improves when we increase the total privacy budget ϵ. This is expected, since a larger ϵ results in less noise added, hence increased precision. More budget is available for building the partitioning tree, thus extent in-

formation associated with leaf nodes will be more accurate. The recall also improves when budget increases. Note that, the relative performance of the three algorithms is different in terms of reduction ratio and recall. A clear tradeoff emerges: the LO approach which achieves the lowest reduction ratio is best in terms of recall. Conversely, LGW has the highest reduction, but incurs the most false negatives. Nevertheless, the absolute values are satisfactory for all proposed values: when $\theta = 0$ and $\epsilon = 1.0$, LGW is able to reduce 99.39% of the secure comparisons and LO is able to retrieve 95.96% of the matching record pairs (with reduction ratio of 98.76%). Even for the larger $\theta = 3$ threshold and $\epsilon = 0.2$, LGW yields a reduction ratio of 74.96% and LO achieves a recall of 92.39% (with reduction ratio 61.96%).

Although reduction ratio and recall are the only metrics directly perceived by the users, we also present in Figure 4.4 how the number of leaf nodes in the index structure varies, as it has a significant impact on reduction ratio and recall. We notice that, in general, the recall rate is inversely proportional to the number of leaf nodes. Note that, the number of leaf nodes when $\epsilon = 0.8$ is a bit higher than when $\epsilon = 1.0$ for LO, but the corresponding reduction ratio is still higher when $\epsilon = 1.0$. When ϵ is larger, a larger amount of budget ϵ_{leaf} will be allocated for privately counting the number of records in the leaf nodes as well, yielding a smaller amount of noise. According to our results, if the Laplace noise is positive, on average 2.90 fake records will be added to a leaf node when $\epsilon = 0.8$, whereas only 2.34 fake records will be added when $\epsilon = 1.0$.

We also point out that for LO, when $\epsilon = 1.0$, the number of leaf nodes for $\theta = 0$ and 1 is smaller than that for 2 and 3, indicating that a partitioning tree producing fewer leaf nodes achieves a better reduction ratio when the matching threshold θ is smaller, which in turn implies that even though the latter cases ($\theta = 2, 3$) can lead to more leaf nodes, it does not necessarily result in a much higher reduction ratio. Taking a closer look at the generated leaf nodes, we found that in the top-25% of the leaf nodes, the former cases ($\theta = 0, 1$) and the latter cases ($\theta = 2, 3$) consist of 49.25% and 55.24% of the records, respectively, that is, the tree is more balanced in

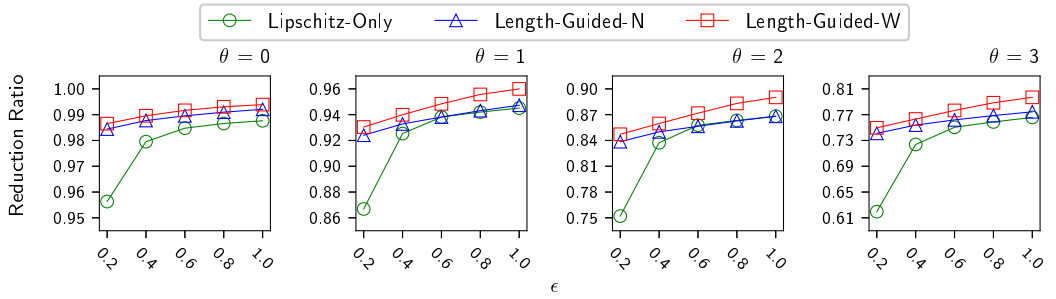


Figure 4.2.: Reduction Ratio (varying privacy budget ϵ and matching threshold θ)

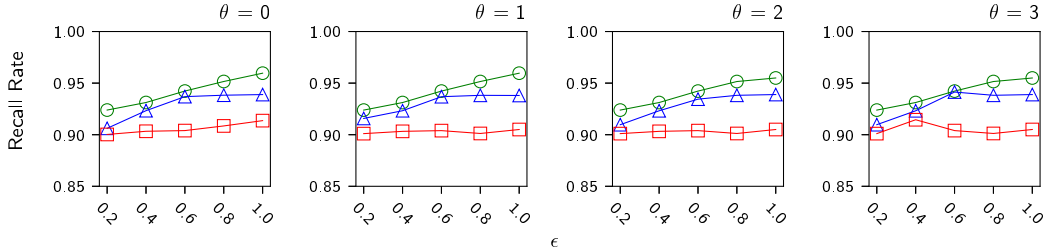


Figure 4.3.: Recall Rate (varying privacy budget ϵ and matching threshold θ)

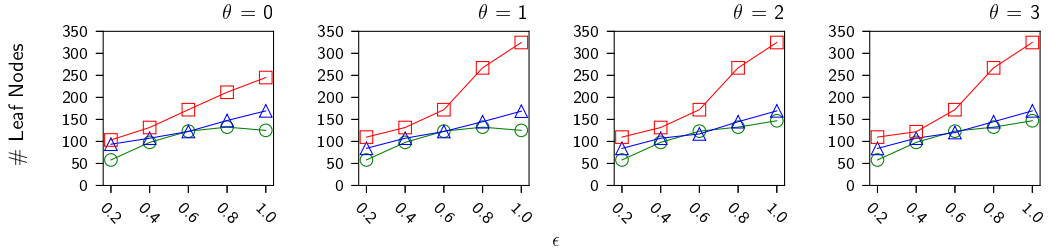


Figure 4.4.: Number of Leaf Nodes (varying privacy budget ϵ and matching threshold θ)

the former cases. For a smaller θ, records in each leaf node of one party only have to be compared with records in fewer leaf nodes of the other party, and thus a more balanced partitioning could achieve a better reduction ratio even though it has fewer leaf nodes. When θ is larger, the benefit of more balanced partitioning is offset since each record in a leaf node of one party will need to be compared with records in a lot more leaf nodes of the other party.

Next, we evaluate the effect of dataset size $|S|$ for fixed $\epsilon = 0.6$. Figures 4.5, 4.6, and 4.7 show the results. Similar to what we observed previously, LGW always yields

the best reduction ratio, whereas LO produces the highest recall. When $\theta = 0$ and $|S| = 50,000$, 99.60% of the secure comparisons can be pruned with LGW and 98.26% of the matching record pairs can be identified by LO (with the latter's reduction ratio of 98.96%). We also observe that the difference in the reduction ratio among these variants becomes more pronounced when we have a larger θ. Specifically, when $\theta = 0$ and $|S| = 30{,}000$, the reduction ratios of LO, LGN, and LGW are 98.91%, 99.45%, and 99.54%, respectively. However, when θ is increased to 3, their reduction ratios become 77.55%, 79.61%, and 81.36%, respectively. The number of leaf nodes generated by each algorithm directly affects its resulting reduction ratio. As seen in Figure 4.7, LGW always generates the largest number of leaf nodes. In addition, the number of leaf nodes generally increases with $|S|$, which is expected, because for a larger dataset size, a smaller amount of privacy budget is needed to select the median or count the respective sizes of the two child nodes. However, we also notice the increasing rate is different for these three variants, with LO being more flat and LGN and LGW being steeper. Examining the generated leaf nodes more closely, we find that for LO, when $|S| = 10{,}000$, 53.58% of the leaf nodes have unit length along each dimension of Lipschitz embeddings, whereas 65.16% of the leaf nodes are of unit length when $|S| = 50{,}000$. A leaf node of unit length cannot be further split and thus is not able to provide more specific information that allows third party Charlie to eliminate more unnecessary comparisons. On the other hand, when $|S| = 10{,}000$, only less than 5.00% of the leaf nodes of LGN and LGW are of unit length. In the case of $|S| = 50{,}000$, around 57.83% and 4.40% of the leaf nodes produced by LGW and LGN are of unit length, respectively. The fact that LGW has much more unit length leaf nodes results from the invocation of AdjustBounds, which greatly reduces the extent of the internal tree nodes before partitioning according to the Lipschitz embeddings. Hence, to further improve the reduction ratio, more dimensions other than the Lipschitz embedding would be helpful.

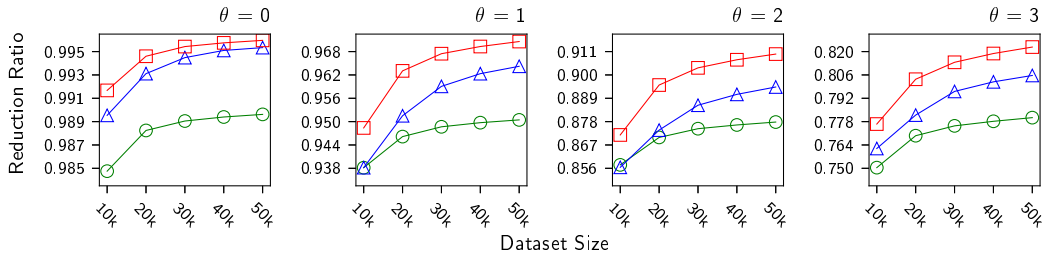


Figure 4.5.: Reduction Ratio (varying dataset size)

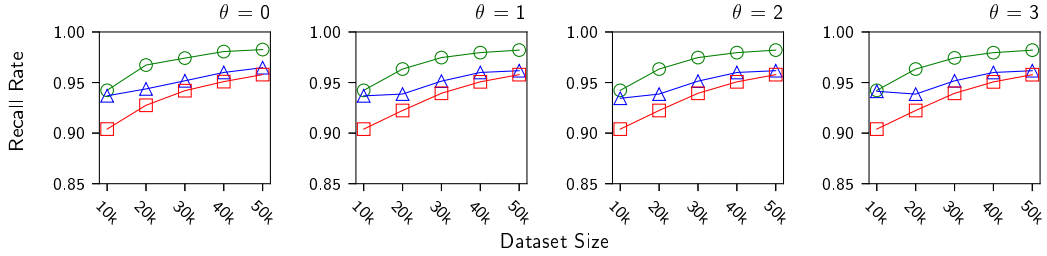


Figure 4.6.: Recall Rate (varying dataset size)

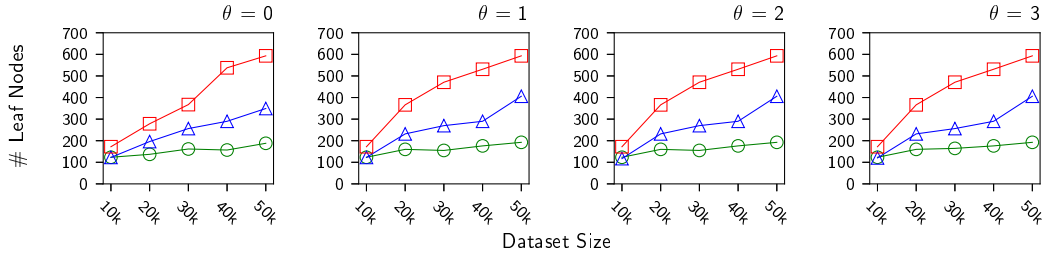


Figure 4.7.: Number of Leaf Nodes (varying dataset size)

4.5 Extension to DPRL Model

Recently, He et al [11] proposed the *Differential Privacy for Record Linkage (DPRL)* model, which is an instance of output-constrained differential privacy specifically targeted at the matching problem. DPRL considers a two-party protocol and provides a quantifiable privacy guarantee for non-matching records after the matching record pairs have been identified. In this section, we provide an overview of DPRL and show that our techniques can be extended to this novel model.

DPRL Model Overview. Given datasets D_A and D_B owned by parties A and B, let f_{Rule} be the function that computes the set $\{(x, y) \mid x \in D_A, y \in D_B, \mathsf{Rule}(x, y) = \mathsf{true}\}$ of matching record pairs according to decision rule *Rule*.

Definition 4.5.1 *(Neighboring Datasets with respect to Private Record Matching [11]) Two datasets D_B and D'_B of B are neighboring datasets if $|D_B| = |D'_B|$ and D_B and D'_B differ only in a pair of non-matching records r_B and r'_B with respect to f_{Rule} and D_A, i.e., $D'_B = D_B \backslash \{r_B\} \cup \{r'_B\}$ and $f_{Rule}(D_A, D_B) = f_{Rule}(D_A, D'_B)$.*

Definition 4.5.2 *(DPRL) A two-party private record matching protocol Π for computing f_{Rule} satisfies $(\epsilon, \delta, f_{Rule})$-constrained differential privacy if for any neighboring datasets (D_B, D'_B) with respect to private record matching with the decision rule* Rule*, the view of party A during the execution of Π when fed into any given probabilistic polynomial time distinguisher S_A satisfies*

$$\begin{aligned} &\Pr[S_A(\mathrm{VIEW}_A^{\Pi}(D_A, D_B)) = 1] \\ &\leq e^{\epsilon} \Pr[S_A(\mathrm{VIEW}_A^{\Pi}(D_A, D'_B)) = 1] + \delta \end{aligned} \tag{4.13}$$

The same must hold for the view $\mathrm{VIEW}_B^{\Pi}(\cdot, D_B)$ of B with respect to any pair of A's neighboring datasets (D_A, D'_A) and any probabilistic polynomial time distinguisher S_B.

The *view* VIEW_A^{Π} in the probabilistic statement above includes *(1)* the *noisy count* and the *extent* of each partition from B's dataset, and *(2)* the *messages* exchanged between A and B due to the invocation of the cryptographic matching protocol. We note that after the matching protocol, A will know the number of the *non-matching*

records in each of B's partitions. Thus, the view regarding the noisy counts could also be thought of as the noisy counts of non-matching records in B's partitions. In what follows, for ease of presentation, we omit the analysis about the security properties of those exchanged messages and focus on the noisy count and the extent of each partition revealed to A.

Extension of Proposed Partitioning Algorithms to DPRL. In the worst case scenario where the third party Charlie colludes with Alice[1], our partitioning algorithms are able to satisfy DPRL by adjusting privacy budgets. Specifically, consider that Charlie reveals to Alice *(1)* the synopses of the leaf nodes generated by Bob, i.e., extent and noisy counts, and *(2)* Bob's leaf node corresponding to each invocation of SMC that performs the decision making. Therefore, in the case of collusion between Alice and Charlie, Alice will know the number of *non-matching* records in each of Bob's leaf nodes on completion of the matching process.

Given privacy parameters ϵ and δ, our modified partitioning algorithm divides each of them in half, and ensures that the privacy budget consumed along each root-to-leaf path is upper-bounded by $\epsilon/2$. Recall that there are two stages in both of our partitioning algorithms, i.e., growing internal tree nodes, and producing noisy counts for each resulting leaf node. We allocate a budget of $(\epsilon - \epsilon_{\mathsf{leaf}})/2$ for the first stage, and the remainder $(\epsilon_{\mathsf{leaf}})/2$ for the second. In addition, the sensitivity of the quality function $q(C, sp)$ used for selecting the median is set to 2 instead of 1, according to Lemma 4.5.1 below, whereas the sensitivity of the private count query remains 1, as in our three-party protocol. To generate noisy counts for the produced leaf nodes, the mean of the Laplace distribution with the scale equal to λ is shifted to the positive direction by $\lceil \mu' \rceil = \lceil -\lambda \ln(2\delta') \rceil$, where $\delta' = \delta/2$, to ensure that the probability of generating a negative noise is upper-bounded by $\delta/2$. This way, we avoid record suppression due to negative noise, which would lead to privacy violation, as shown in [10, 11].

[1]When the third party Charlie colludes with either dataset owner (but not both), the resulting protocol is essentially equivalent to a 2-party one.

Next, we prove that for any partitioning tree T generated by our modified algorithms, the probability of generating T will not vary significantly given any pair of neighboring datasets. Specifically, we have the following lemma.

Lemma 4.5.1 *Given any partitioning tree T generated by our modified partitioning algorithms, where each internal tree node C is associated with a noisy count $C.\tilde{s}$ produced by the Laplace mechanism and a private median $C.sp$ selected by the exponential mechanism, the ratio of probabilities of producing T by our partitioning algorithms given a pair of neighboring datasets (D_B, D'_B) with respect to f_{Rule} and D_A, can be upper bounded by $\exp(\epsilon - \epsilon_{\mathsf{leaf}})$.*

Proof Let $C_1, \ldots, C_\gamma$ be the set of internal tree nodes in the partitioning tree T generated by our algorithm. Denote the noisy count and the selected median associated with node C_i by $C_i.\tilde{s}$, and $C_i.sp$, respectively. Define $I = \{i \mid 1 \leq i \leq \gamma\}$, $J = \{j \mid \text{record } r_B \in C_j \text{ or record } r'_B \in C_j\}$, and $\overline{J} = I \backslash J$. That is, J is the set containing the indices of internal tree nodes that contain either the non-matching record r_B or r'_B. Let $\mathcal{A}^{D_B}_{\epsilon_i,\mathsf{cnt}}$ and $\mathcal{A}^{D'_B}_{\epsilon_i,\mathsf{cnt}}$ denote the random variables corresponding to the noisy counts of C_i produced by the Laplace mechanism given D_B and D'_B satisfying $\Pr[\mathcal{A}^{D_B}_{\epsilon_i,\mathsf{cnt}} = C_i.\tilde{s}] \leq e^{\epsilon_{i,\mathsf{cnt}}} \Pr[\mathcal{A}^{D'_B}_{\epsilon_i,\mathsf{cnt}} = C_i.\tilde{s}]$. Also, let $\mathcal{A}^{D_B}_{\epsilon_{i,\mathsf{med}}}$ and $\mathcal{A}^{D'_B}_{\epsilon_{i,\mathsf{med}}}$ be the random variables corresponding to the medians produced for C_i by the exponential mechanism under D_B and D'_B satisfying $\Pr[\mathcal{A}^{D_B}_{\epsilon_{i,\mathsf{med}}} = C_i.sp] \leq e^{\epsilon_{i,\mathsf{med}}} \Pr[\mathcal{A}^{D'_B}_{\epsilon_{i,\mathsf{med}}} = C_i.sp]$. The probability $P^{D_B}_T$ of generating T given D_B is expressed as

$$
\begin{aligned}
P^{D_B}_T = \prod_{i=1}^{\gamma} \Pr[\mathcal{A}^{D_B}_{\epsilon_{i,\mathsf{cnt}}} = C_i.\tilde{s}, \mathcal{A}^{D_B}_{\epsilon_{i,\mathsf{med}}} = C_i.sp \mid \\
\mathcal{A}^{D_B}_{\epsilon_{1,\mathsf{cnt}}}, \mathcal{A}^{D_B}_{\epsilon_{1,\mathsf{med}}}, \ldots, \mathcal{A}^{D_B}_{\epsilon_{i-1,\mathsf{cnt}}}, \mathcal{A}^{D_B}_{\epsilon_{i-1,\mathsf{med}}}] \qquad (4.14) \\
= \prod_{i \in J} \Pr[\mathcal{A}^{D_B}_{\epsilon_{i,\mathsf{med}}} = C_i.sp] \Pr[\mathcal{A}^{D_B}_{\epsilon_{i,\mathsf{cnt}}} = C_i.\tilde{s}] \\
\prod_{i \in \overline{J}} \Pr[\mathcal{A}^{D_B}_{\epsilon_{i,\mathsf{med}}} = C_i.sp] \Pr[\mathcal{A}^{D_B}_{\epsilon_{i,\mathsf{cnt}}} = C_i.\tilde{s}]. \qquad (4.15)
\end{aligned}
$$

The probability $P^{D'_B}_T$ of generating T given D'_B can be derived similarly. Note that for $i \in \overline{J}$ it holds that $\Pr[\mathcal{A}^{D_B}_{\epsilon_{i,\mathsf{med}}} = C_i.sp] = \Pr[\mathcal{A}^{D'_B}_{\epsilon_{i,\mathsf{med}}} = C_i.sp]$ and that $\Pr[\mathcal{A}^{D_B}_{\epsilon_{i,\mathsf{cnt}}} =$

$C_i.\tilde{s}] = \Pr[\mathcal{A}^{D'_B}_{\epsilon_{i,\mathsf{cnt}}} = C_i.\tilde{s}]$ since the records within these internal nodes are exactly the same no matter which one of D_B and D'_B is used by Bob. On the other hand, for each $i \in J$, it holds that $\Pr[\mathcal{A}^{D_B}_{\epsilon_{i,\mathsf{med}}} = C_i.sp] \leq e^{\epsilon_{i,\mathsf{med}}} \Pr[\mathcal{A}^{D'_B}_{\epsilon_{i,\mathsf{med}}} = C_i.sp]$ and that $\Pr[\mathcal{A}^{D_B}_{\epsilon_{i,\mathsf{cnt}}} = C_i.\tilde{s}] \leq e^{\epsilon_{i,\mathsf{cnt}}} \Pr[\mathcal{A}^{D'_B}_{\epsilon_{i,\mathsf{cnt}}} = C_i.\tilde{s}]$. The ratio $(P_T^{D_B}/P_T^{D'_B})$ can be rewritten as

$$\frac{\prod_{i\in J} \Pr[\mathcal{A}^{D_B}_{\epsilon_{i,\mathsf{med}}} = C_i.sp] \Pr[\mathcal{A}^{D_B}_{\epsilon_{i,\mathsf{cnt}}} = C_i.\tilde{s}]}{\prod_{i\in J} \Pr[\mathcal{A}^{D'_B}_{\epsilon_{i,\mathsf{med}}} = C_i.sp] \Pr[\mathcal{A}^{D'_B}_{\epsilon_{i,\mathsf{cnt}}} = C_i.\tilde{s}]} \tag{4.16}$$

$$\leq \exp\left(\sum_{i\in J} \epsilon_{i,\mathsf{med}} + \sum_{i\in J} \epsilon_{i,\mathsf{cnt}}\right). \tag{4.17}$$

To ensure that the ratio is upper-bounded by $\exp(\epsilon - \epsilon_{\mathsf{leaf}})$, it should hold that $\exp(\epsilon - \epsilon_{\mathsf{leaf}}) = \exp\left(\sum_{i\in J} \epsilon_{i,\mathsf{med}} + \sum_{i\in J} \epsilon_{i,\mathsf{cnt}}\right)$. However, recall that the two root-to-leaf paths involving r_B and r'_B are not known in advance, which means we do not know from which internal tree node r_B and r'_B initially split. Thus, one way to distribute the budget $(\epsilon - \epsilon_{\mathsf{leaf}})$ is to evenly distribute it between the two paths, as we do in our modified partitioning algorithms described earlier. By doing this, we ensure that for any given root-to-leaf path, the ratio of the probability of generating the noisy counts and medians associated with this path given D_B to that given D'_B is upper-bounded by $\exp((\epsilon - \epsilon_{\mathsf{leaf}})/2)$. Together with the fact that there are *at most* two such paths, the multiplicative factor is upper-bounded by $\exp(\epsilon - \epsilon_{\mathsf{leaf}})$. The privacy guarantee above holds for a specific tuple $\phi = (C_1.sp, C_1.\tilde{s}, \ldots, C_\gamma.sp, C_\gamma.\tilde{s})$ in the set Φ containing all consistent[2] tuples with respect to T. The same guarantee still holds if we take the summation over all consistent tuples with respect to the partitioning tree T. Specifically,

$$\frac{\Pr[T \mid D_B]}{\Pr[T \mid D'_B]} = \frac{\sum_{\phi'\in\Phi} \Pr[\phi'|D_B]}{\sum_{\phi'\in\Phi} \Pr[\phi'|D'_B]} \leq \exp(\epsilon - \epsilon_{\mathsf{leaf}}). \tag{4.18}$$

Lastly, we explain how we guarantee that $\Pr[\mathcal{A}^{D_B}_{\epsilon_i,\mathsf{cnt}} = C_i.\tilde{s}] \leq e^{\epsilon_{i,\mathsf{cnt}}} \Pr[\mathcal{A}^{D'_B}_{\epsilon_i,\mathsf{cnt}} = C_i.\tilde{s}]$ and $\Pr[\mathcal{A}^{D_B}_{\epsilon_{i,\mathsf{med}}} = C_i.sp] \leq e^{\epsilon_{i,\mathsf{med}}} \Pr[\mathcal{A}^{D'_B}_{\epsilon_{i,\mathsf{med}}} = C_i.sp]$. For the first inequality to hold, we know that the sensitivity with respect to the count query in the Laplace

[2]A consistent tuple is defined as a tuple that could be produced by our partitioning algorithms according to the pre-defined quality parameters used in the private counting and median selection for internal tree nodes.

mechanism for any internal node is 1, which is reached when this internal node is on the root-to-leaf path corresponding to r_B or r'_B. On the other hand, recall that the quality function $q(C, sp)$ used for the median selection by the exponential mechanism is $-|C_{L,sp}.s - C_{R,sp}.s|$, and thus for the second inequality to hold, the sensitivity of the quality function $q(C, sp)$ in median selection should be 2, which is reached when the exponential mechanism is invoked on an internal node on the root-to-leaf path corresponding to both r_B and r'_B. ■

The second stage of our partitioning algorithms satisfies the extended privacy notion in [11] as well.

Lemma 4.5.2 *Let $\{C_1, \ldots, C_\psi\}$ be the set of leaf nodes and T the set of their extents, where each leaf node C_i is associated with a noisy count $C_i.\tilde{s}$ produced by the Laplace mechanism. Then, the ratio of probabilities of producing any $(C_1.\tilde{s}, \ldots, C_\psi.\tilde{s})$ in the second stage of our modified partitioning algorithms, for any pair of neighboring datasets (D_B, D'_B) with respect to f_{Rule} and D_A, can be upper bounded by $\exp(\epsilon_{\mathsf{leaf}})$ with probability $1 - \delta$.*

Proof Define $I = \{i \mid 1 \leq i \leq \psi\}$, $J = \{j \mid \text{record } r_B \in C_j \text{ or record } r'_B \in C_j\}$, and $\overline{J} = I \backslash J$. Let $\mathcal{A}^{D_B}_{\epsilon_{i,\mathsf{cnt}}}$ and $\mathcal{A}^{D'_B}_{\epsilon_{i,\mathsf{cnt}}}$ denote the random variables corresponding to the noisy counts of C_i produced by the Laplace mechanism given D_B and D'_B satisfying $\Pr[\mathcal{A}^{D_B}_{\epsilon_i,\mathsf{cnt}} = C_i.\tilde{s}] \leq e^{\epsilon_{i,\mathsf{cnt}}} \Pr[\mathcal{A}^{D'_B}_{\epsilon_i,\mathsf{cnt}} = C_i.\tilde{s}]$ with probability at least $1-\delta/2$. Similar to Lemma 4.5.1, the ratio of the probabilities to generate noisy counts $(C_1.\tilde{s}, \ldots, C_\psi.\tilde{s})$ given D_B or D'_B as input, can be expressed as

$$\frac{\prod_{i \in J} \Pr[\mathcal{A}^{D_B}_{\epsilon_{i,\mathsf{cnt}}} = C_i.\tilde{s}]}{\prod_{i \in J} \Pr[\mathcal{A}^{D'_B}_{\epsilon_{i,\mathsf{cnt}}} = C_i.\tilde{s}]} \leq \prod_{i \in J} e^{\epsilon_{i,\mathsf{cnt}}} = \exp\left(\sum_{i \in J} \epsilon_{i,\mathsf{cnt}}\right). \tag{4.19}$$

To ensure that the ratio is upper-bounded by $\exp(\epsilon_{\mathsf{leaf}})$, it must hold that $\exp(\epsilon_{\mathsf{leaf}}) = \exp\left(\sum_{i \in J} \epsilon_{i,\mathsf{cnt}}\right)$. Not knowing whether or not r_B and r'_B fall into the same leaf node, our modified partitioning algorithms divide ϵ_{leaf} evenly and we guarantee that the ratio of the probability of generating the noisy count of a leaf node corresponding to some element in J given D_B to that given D'_B is upper-bounded by $\exp(\epsilon_{\mathsf{leaf}}/2)$.

Finally, to guarantee that $\Pr[\mathcal{A}^{D_B}_{\epsilon_{i,\mathsf{cnt}}} = C_i.\tilde{s}] \leq e^{\epsilon_{i,\mathsf{cnt}}} \Pr[\mathcal{A}^{D'_B}_{\epsilon_{i,\mathsf{cnt}}} = C_i.\tilde{s}]$ with probability at least $1-\delta/2$, we shift the mean of the Laplace distribution to the positive direction such that the chance of generating a negative noise is upper-bounded by $\delta' = \delta/2$. Specifically, we solve the following inequality for μ' where $p(z \mid 0, \lambda) = \frac{1}{2\lambda} e^{-|z|/\lambda}$ is the probability density function of the Laplace distribution with the mean equal to 0, and then set $\lceil \mu' \rceil$ as the shifted mean:

$$\frac{1}{2} + \int_0^{\mu'} \frac{1}{2\lambda} e^{-z/\lambda} dz \geq 1 - \delta' \tag{4.20}$$

$$\Rightarrow 1 - \frac{1}{2} e^{-\mu'/\lambda} \geq 1 - \delta' \Rightarrow \mu' \geq -\lambda \ln(2\delta'). \tag{4.21}$$

Since there are at most two elements in J, each corresponding to a leaf node having r_B or r'_B, the probability of generating a negative noise for at least one leaf node is upper-bounded by $(\frac{\delta}{2})2 = \delta$ according to the union bound. ■

Using the sequential composition property of DPRL [11], it results by adding the respective budgets used in each stage that our modified partitioning algorithm satisfies $(\epsilon, \delta, f_{\mathsf{Rule}})$-constrained differential privacy.

4.5.1 Evaluation

We also performed a brief experimental evaluation of the proposed techniques under the DPRL model. Figures 4.8 and 4.9 report the reduction ratio when varying the dataset size and δ, respectively (privacy budget is set to $\epsilon = 0.8$). The δ used in Figure 4.8 is 10^{-5}, whereas the dataset size used in Figure 4.9 is 30,000. The modified partitioning algorithms still produce acceptable reduction ratios. For instance, when the dataset size is 30,000 and $\theta = 1$, the reduction ratio is at least 90.2%. Note that under the DPRL model, our partitioning algorithms do not suppress *any* records when the drawn Laplace noise is negative, and hence we can achieve a recall of 1, the same as what is shown in [11]. We also emphasize that if we further allow the noisy count of non-matching records in each leaf node to be revealed to the other party in the clear just as it is done in [11], then further reduction in the computation could be

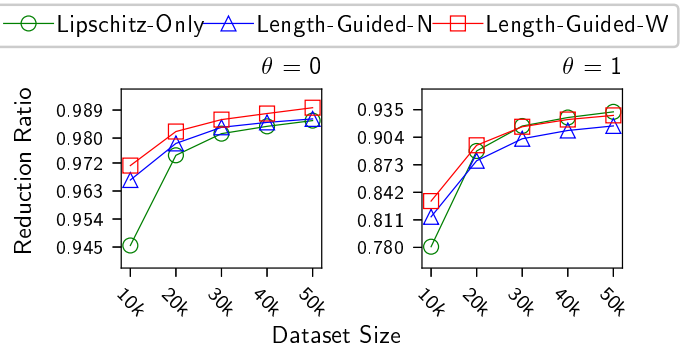


Figure 4.8.: Reduction Ratio (varying dataset size)

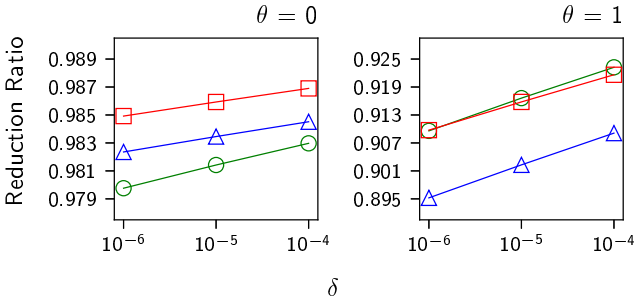


Figure 4.9.: Reduction Ratio (varying δ)

achieved by Alice and Bob exchanging the result every time a record pair is deemed as a match. In this case, we do not need Charlie to perform the step of value blinding as in our three-party protocol. Specifically, Alice could use Bob's matching record to match her dataset locally and the matching records identified this way do not need to participate in the subsequent SMC step.

4.5.2 Discussion

We showed how our proposed techniques may be used in either a three-party or a two-party setting. The two-party setting provides strong protection according to the output-constrained differential privacy model [11], whereas the three-party setting assumes the conventional differential privacy model. The third-party model uses the privacy budget more effectively, and as a result obtains a higher reduction ratio.

However, as shown in [11], in the case of collusion with the third party, there may be a small amount of leakage resulting from the fact that each party gains knowledge about the noisy count of non-matching records in each partition of the other side. Nevertheless, as long as the third-party Charlie does not collude with any of the data owners, the solution offers good protection. On the other hand, the two-party solution provides stronger protection, but the reduction ratio is lower, due to the fact that the partitioning and record counting budget is virtually halved compared to the third-party case.

4.6 Related Work

Private record linkage plays an important role for privacy-preser-ving statistical analysis, when data are distributed across multiple databases owned by different entities. Several schemes have been proposed to implement private linkage for relational data. Methods in [2, 29, 30] consider the data to be matched as set elements represented by integers. Those protocols provide well-defined security guarantees since they are based on SMC primitives. However, they incur prohibitive costs. Moreover, they do not support other data types, e.g., strings. The approach by Beck and Kerschbaum [46], considers privacy-preserving string matching in the two-party scenario where a client wants to know whether the edit distance between its string and that owned by the server is less than or equal to a predefined threshold. By transforming a string into grams and then using a Bloom filter to represent the string in a compact way, the approximate edit distance between two encoded strings is estimated. Their protocol is based on an additively homomorphic encryption scheme similar to the Paillier cryptosystem. Wang et al. [47] propose an efficient secure protocol for approximate human genome matching with respect to edit distance based on the key observation that 99.5% of human DNA sequences are identical, and that most of the edits from the reference genomes take place at non-adjacent locations. It is however not clear whether or not such a technique could be applied to strings constructed from

a larger alphabet. Protocols in [22, 23, 26, 27] address the problem of matching string data. These schemes first embed records into vectors, and the matching is performed on the encoded vectors in the clear. These protocols, although being efficient, do not provide formal guarantees of privacy.

To improve efficiency, hybrid approaches have been proposed [9,10,13,22]. In these approaches, records are first partitioned into subsets, and are securely matched with subsets containing similar records. Al-Lawati et al. [22] give the first hybrid approach but do not provide a privacy guarantee of information leakage due to record partitioning. In an attempt to provide a formal privacy guarantee, approaches proposed by Inan et al. [9] and Kuzu et al. [13] partition the records in a differentially-private manner [7]. But as shown by Cao et al. [10], differential privacy does not hold after the matching records and subset synopses are combined. Our work is the only hybrid approach for linking string data that provides provable privacy guarantees.

Private spatial decomposition algorithms [9,10,15,16,31] are also closely related to our work, as they privately partition data. Approaches [9,15] fix indexing tree height, and allocate privacy budget equally to all the levels in the tree. Cormode et al. [16] instead adopt a geometric series to allocate the privacy budgets, but they still fix the tree height. Qardaji et al. [31] partition the data space into a grid of uniform cells. However, the developed approach is tailored to two-dimensional geospatial datasets. It is not clear how to extend it to higher dimensional data.

5 PRIVACY-PRESERVING AND OUTSOURCED MULTI-USER K-MEANS CLUSTERING

Clustering [48] is one of the widely used techniques in various data mining applications in several fields, including information retrieval [49], machine learning [50], and pattern recognition [51, 52]. Real-life applications related to clustering include categorizing results returned by a search engine in response to a user's query, and grouping persons into categories based on their DNA information. In this chapter, we consider the problem where multiple users, such as companies, governmental agencies, and health care organizations, each holding a dataset, want to collaboratively perform clustering on their combined data and share the clustering results in a privacy-preserving way. This problem, referred to as privacy-preserving distributed clustering (PPDC), can be best explained by the following example:

- Consider two health providers each holding a dataset containing the disease patterns and clinical outcomes of their patients. Suppose that the providers want to cluster their combined datasets and identify interesting clusters that would enable directions for better disease prevention. However, suppose that due to the sensitive nature of the data, they are not able to share their data. Hence, they have to collaboratively perform the clustering task on their joint datasets in a privacy-preserving manner. Once the clustering task is done, they can exchange necessary information (after proper sanitization) if needed.

Existing PPDC methods (e.g., [53–56]) incur significant cost (computation, communication and storage) on the participating users and thus they are not suitable if the users do not have sufficient resources to perform the clustering task. This problem becomes even more serious when dealing with big data. Sometimes the users do not even possess the relevant expertise for developing or deploying such protocols. To

address these issues, it is more attractive for the users to outsource their data as well as the clustering task to the cloud. However, the cloud cannot be fully trusted by the users in protecting their data. Thus, to ensure data confidentiality, users can encrypt their databases locally (using a common public key) and then outsource them to the cloud. Then, the goal is for the cloud to perform clustering over the aggregated encrypted data. We refer to the above process as *privacy-preserving and outsourced distributed clustering (PPODC)*.

It is worth noting that if all the encrypted data resides on a single cloud, then the only way for the cloud to execute the clustering task (assuming that users do not participate in the clustering process), without ever decrypting the data, is when the data is encrypted using fully homomorphic encryption schemes (e.g., [57]). Past results [58], however, show that fully homomorphic encryption schemes are very expensive and their usage in practical applications is decades away.

In this chapter, we propose a new and efficient solution to the PPODC problem based on the standard k-means clustering algorithm [59,60] with the use of two cloud service providers (say Amazon and Google) which together form a federated cloud environment. Our proposed solution protects data confidentiality of all the participating users at all times. We emphasize that the concept of federated clouds is becoming increasingly popular and is also identified as one of the ten High Priority Requirements for U.S. cloud adoption in the NIST U.S. Government Cloud Computing Technology Roadmap [61]. Therefore, we believe that developing privacy-preserving solutions under federated cloud environments will become increasingly important in the near future. The main contributions of this work are five-fold:

- We propose new transformations in order to develop an order-preserving Euclidean distance function that enables our proposed protocol to securely assign the data records to the closest clusters, a crucial step in each iteration of the k-means clustering algorithm (see Sections 5.2.1 and 5.3.3.2). Also, we propose a novel transformation for the termination condition that enables our protocol

to securely evaluate the termination condition over encrypted data (see Sections 5.2.2 and 5.3.3.3).

- We discuss novel constructions of two cryptographic primitives, namely secure squared order-preserving Euclidean distance and secure minimum out of k numbers, which are used as building blocks in our proposed solution (see Section 5.3.2).

- The proposed solution protects the confidentiality of each user's data at all times and outputs the correct result. Specifically, we show that the proposed protocol is secure under the standard semi-honest model (see Section 5.3.4). We emphasize that, once a user's data is outsourced to the cloud, the user does not need to participate in any computations of the clustering task.

- We present two strategies to improve the performance of our protocol. The first strategy, referred to as offline computation, allows the cloud to pre-compute some data independent intermediate results, thus boosting online computation time. The second strategy, referred to as pipelined execution, relies on pipelining the underlying computations (see Section 5.3.6.2).

- By using a real dataset, we experimentally show that the above techniques can indeed boost the performance of our protocol. Moreover, we demonstrate how the performance of our protocol can be drastically improved using parallel implementation on a cluster of 16 nodes (see Section 5.4).

The remainder of this chapter is organized as follows. Section 5.1 introduces definitions and properties related to k-means clustering algorithm. Section 5.2 presents our new transformation techniques. Section 5.3 discusses our proposed PPODC solution in detail. Also, we analyze the security guarantees of PPODC and discuss two performance improvement strategies. Section 5.4 presents our experimental results based on a real-world dataset under different parameter settings, and Section 5.5 discusses related work.

5.1 Background

Definition 5.1.1 *Suppose $c = \{t_1, \ldots, t_h\}$ denote a cluster where $t_1, \ldots, t_h$ are data records with l attributes. Then, the center of cluster c is defined as a vector μ_c given by [56]:*

$$\mu_c[s] = \frac{t_1[s] + \ \cdots \ + t_h[s]}{|c|} = \frac{\lambda_c[s]}{|c|}, \ \textit{for } 1 \leq s \leq l \tag{5.1}$$

where $t_i[s]$ denotes the s^{th} attribute value of t_i and $\lambda_c[s]$ denotes the sum of s^{th} attribute values of all the data records in cluster c, for $1 \leq i \leq h$. Also, $|c|$ denotes the number of data records in c.

In the above definition, the s^{th} attribute value in μ_c is equivalent to the mean of the s^{th} attribute values of all the data records in cluster c. Note that, if the cluster contains a single data record, then the cluster center is the same as the corresponding data record.

Example 5.1.1 *Let c be a cluster with three data records given as: $t_1 = \{0, 2, 1, 0, 3\}$, $t_2 = \{1, 1, 3, 4, 2\}$, and $t_3 = \{0, 1, 0, 2, 0\}$. Based on Definition 5.1.1, the center of cluster c is given by $\mu_c[1] = 0.333$, $\mu_c[2] = 1.333$, $\mu_c[3] = 1.333$, $\mu_c[4] = 2$, $\mu_c[5] = 1.666$.* □

5.1.1 Euclidean Distance between t_i and c

We now discuss how to compute the similarity score between a given data record t_i and a cluster c. In general, the similarity score between any two records can be computed using one of the standard similarity metrics, such as Euclidean distance and Cosine similarity. In this chapter, we use the Euclidean distance as the underlying similarity metric since the standard k-means algorithm is based on this metric [56,62].

Definition 5.1.2 *For any given data record t_i and cluster c, let μ_c denote the cluster center of c (as per Definition 5.1.1). Then the Euclidean distance between t_i and c is given as*

$$\|t_i - c\| = \sqrt{\sum_{s=1}^{l} (t_i[s] - \mu_c[s])^2} = \sqrt{\sum_{s=1}^{l} \left(t_i[s] - \frac{\lambda_c[s]}{|c|}\right)^2}$$

Example 5.1.2 *Suppose that* $t_i = \{0, 1, 1, 3, 2\}$ *and* $\mu_c = \{0.333, 1.333, 1.333, 2, 1.666\}$. *Then, the Euclidean distance between* t_i *and* c, *based on Definition 5.1.2, is* $\|t_i - c\| = 1.201$. □

In a similar manner, the Euclidean distance between any two given clusters c and c' can be computed using their respective cluster centers. More specifically, $\|c - c'\|$ is given as

$$\sqrt{\sum_{s=1}^{l} (\mu_c[s] - \mu_{c'}[s])^2} = \sqrt{\sum_{s=1}^{l} \left(\frac{\lambda_c[s]}{|c|} - \frac{\lambda_{c'}[s]}{|c'|}\right)^2}$$

where μ_c and $\mu_{c'}$ denote the cluster centers of c and c', respectively. Also, $|c|$ and $|c'|$ denote the number of data records in c and c'.

5.1.2 Single Party k-Means Clustering

Consider a user U who wants to apply the k-means clustering algorithm [59,60] on his/her own database of m records, denoted by $\{t_1, \ldots, t_m\}$. Here we assume that U wants to compute k cluster centers, denoted by $\mu_{c'_1}, \ldots, \mu_{c'_k}$, as the output. However, other desired values, such as the final cluster IDs assigned to each data record can also be part of the output. Since k-means clustering is an iterative algorithm, U has to input a threshold value to decide when to stop the algorithm (termination condition). Without loss of generality, let β denote the threshold value. For simplicity, throughout this chapter, we assume that the initial set of k clusters are chosen at random (referred to as the Initialization step).

The main steps involved in the traditional (single party) k-means clustering task [59,60], using the Euclidean distance as the similarity metric, are given in Algorithm 5.1. Apart from the initialization step, the algorithm involves three main stages: (i) Assignment (ii) Update and (ii) Termination. In the initialization step, k data records are selected at random and assigned as the initial clusters $c_1, \ldots, c_k$ with their centers

Algorithm 5.1 k-means$(\{t_1, \ldots, t_m\}, \beta) \rightarrow \{\mu_{c'_1}, \ldots, \mu_{c'_k}\}$

Require: User U with m data records $\{t_1, \ldots, t_m\}$ and β

1: **Initialization**: Select k data records at random and assign them as initial clusters $c_1, \ldots, c_k$ with respective cluster centers as $\mu_{c_1}, \ldots, \mu_{c_k}$
2: **for** $j = 1$ **to** k **do**
3: $\quad c'_j \leftarrow \emptyset$, $\mu_{c'_j} \leftarrow \{\}$ and $sum \leftarrow 0$
4: **end for**
5: **for** $i = 1$ **to** m **do** ▷ Steps 5 to 10: **Assignment Stage**
6: $\quad$ **for** $j = 1$ **to** k **do**
7: $\quad\quad$ Compute $\|t_i - c_j\|$
8: $\quad$ **end for**
9: $\quad$ Add t_i to cluster c'_h such that $\|t_i - c_h\|$ is the minimum, for $1 \leq h \leq k$
10: **end for**
11: **for** $j = 1$ **to** k **do** ▷ Steps 11 to 13: **Update Stage**
12: $\quad$ Compute cluster center for c'_j and assign it to $\mu_{c'_j}$
13: **end for**
14: $sum \leftarrow \sum_{j=1}^{k} \|c_j - c'_j\|^2$ ▷ Steps 14 to 22: **Termination Stage**
15: **if** $sum \leq \beta$ **then**
16: $\quad$ Return $\{\mu_{c'_1}, \ldots, \mu_{c'_k}\}$
17: **else**
18: $\quad$ **for** $j = 1$ **to** k **do**
19: $\quad\quad$ $c_j \leftarrow c'_j$ and $\mu_{c_j} \leftarrow \mu_{c'_j}$
20: $\quad$ **end for**
21: $\quad$ Go to Step 5
22: **end if**

(or mean vectors) denoted by $\mu_{c_1}, \ldots, \mu_{c_k}$, respectively. In the assignment stage, for each data record t_i, the algorithm computes the Euclidean distance between t_i and each cluster c_j, for $1 \leq j \leq k$. The algorithm identifies the cluster corresponding to the minimum distance as the closest cluster to t_i (say c_h) and assigns t_i to a new

cluster c'_h, where $h \in [1, k]$. In the update stage, the algorithm computes the centers of the new clusters, denoted by $\mu_{c'_1}, \ldots, \mu_{c'_k}$. Finally, in the termination stage, the algorithm verifies whether the pre-defined termination condition holds. Specifically, the algorithm checks whether the sum of the squared Euclidean distances between the current and newly computed clusters is less than or equal to the threshold value β. If the termination condition holds, then the algorithm halts and returns the new cluster centers as the final output. Otherwise, the algorithm continues to the next iteration with the new clusters as input.

5.2 The Proposed Transformations

It is important to note that cluster centers (denoted by μ_c for a cluster c) are represented as vectors and the entries in the vectors can be fractional values. Since the encryption schemes typically support integer values, we should somehow transform the entries of the cluster centers into integer values without affecting their utility in the k-means clustering process. Therefore, we first define scaling factors for clusters and then discuss a novel order-preserving Euclidean distance function operating over integers. Also, we discuss how to transform the termination condition in k-means clustering algorithm with fractional values into an integer-valued one.

Definition 5.2.1 *Consider the cluster c_i whose center is denoted by μ_{c_i} (based on Definition 5.1.1). We know that μ_{c_i} is a vector and each entry can be a fractional value with denominator $|c_i|$, for $1 \leq i \leq k$. We define the scaling factor for c_i, denoted by α_i, as below:*

$$\alpha_i = \frac{\prod_{j=1}^{k} |c_j|}{|c_i|} = \prod_{j=1 \wedge j \neq i}^{k} |c_j|. \tag{5.2}$$

Also, we define $\alpha = \prod_{j=1}^{k} |c_j|$ as the global scaling factor.

5.2.1 Order-Preserving Euclidean Distance

In the assignment stage of k-means clustering, the first step is to compute the Euclidean distance between a data record t_i and each cluster c_j, denoted by $\|t_i - c_j\| = \sqrt{\sum_{s=1}^{l} \left(t_i[s] - \frac{\lambda_{c_j}[s]}{|c_j|}\right)^2}$. It is clear that $\|t_i - c_j\|$ involves the fractional value $\frac{\lambda_{c_j}[s]}{|c_j|}$. In order to compute the encrypted value of $\|t_i - c_j\|$, we need to avoid such fractional values without affecting the relative ordering among the k Euclidean distances $\|t_i - c_1\|, \ldots, \|t_i - c_k\|$, where $c_1, \ldots, c_k$ denote k clusters. Note that since t_i has to be assigned to the nearest cluster, it is important to preserve the relative ordering among the computed k Euclidean distances. For this purpose, we propose a novel order-preserving Euclidean distance function whose evaluation involves only integer values.

We define the order-preserving Euclidean distance (OPED) function between a data record t_i and a cluster c_j as follows:

$$\text{OPED}(t_i, c_j) = \sqrt{\sum_{s=1}^{l} \left(\alpha \times t_i[s] - \alpha_j \times \lambda_{c_j}[s]\right)^2} \tag{5.3}$$

where α and α_j denote the global and c_j's scaling factors, respectively. Observe that all the terms in the above equation are integer values (here each attribute is explicitly assumed to take only integer values). Moreover, following Definition 5.2.1, we can rewrite the above equation as:

$$\begin{aligned}\text{OPED}(t_i, c_j) &= \sqrt{\sum_{s=1}^{l} \left(\alpha \times t_i[s] - \frac{\alpha}{|c_j|} \times \lambda_{c_j}[s]\right)^2} \\ &= \alpha \times \|t_i - c_j\|.\end{aligned}$$

Since α remains constant for any given set of k clusters (in a particular iteration), we claim that the above OPED function preserves the relative ordering among cluster centers for any given data record. More specifically, given a data record t_i and two clusters c_j and $c_{j'}$, if $\|t_i - c_j\| \geq \|t_i - c_{j'}\|$, then it is guaranteed that $\text{OPED}(t_i, c_j) \geq \text{OPED}(t_i, c_{j'})$, for $1 \leq j, j' \leq k$ and $j \neq j'$.

5.2.2 Termination Condition - Transformation

In the k-means clustering process (see Algorithm 5.1), the termination condition is given by:

$$\sum_{j=1}^{k} \|c_j - c'_j\|^2 \leq \beta \tag{5.4}$$

where $c_1, \ldots, c_k$ and $c'_1, \ldots, c'_k$ denote the current and new set of clusters in an iteration, respectively. Remember that $\|c_j - c'_j\| = \sqrt{\sum_{s=1}^{l} \left(\frac{\lambda_{c_j}[s]}{|c_j|} - \frac{\lambda_{c'_j}[s]}{|c'_j|} \right)^2}$ and clearly it consists of fractional values. In order to evaluate this condition over encrypted values, we first need to transform the above termination condition so that all the components are integers. To achieve this, we use the following approach. We define a constant scaling factor (denoted by f) for the termination condition in such a way that by multiplying both sides of Equation 5.4 with f^2, we can cancel all the denominator values. More specifically, we define the scaling factor for the termination condition as $f = \prod_{j=1}^{k} |c_j| \times |c'_j|$. Also, we define the scaling factor for the cluster pair (c_j, c'_j) as $f_j = \frac{f}{|c_j| \times |c'_j|} = \prod_{i=1 \wedge i \neq j}^{k} |c_i| \times |c'_i|$. Then we define the new termination condition as follows:

$$\sum_{j=1}^{k} \sum_{s=1}^{l} \left(|c'_j| \times f_j \times \lambda_{c_j}[s] - |c_j| \times f_j \times \lambda_{c'_j}[s] \right)^2 \leq f^2 \times \beta. \tag{5.5}$$

Observe that the above equation consists of only integer values. Now we need to show that evaluating the above equation is the same as evaluating Equation 5.4. First, we divide the above equation by f^2 on both sides of the inequality. Note that since f^2 remains constant in a given iteration, multiplication of both sides of Equation 5.5 by f^2 has no effect on the inequality. That is, Equation 5.5 can be rewritten as:

$$\sum_{j=1}^{k} \sum_{s=1}^{l} \frac{\left(|c'_j| \times f_j \times \lambda_{c_j}[s] - |c_j| \times f_j \times \lambda_{c'_j}[s] \right)^2}{f^2} \leq \beta.$$

The left-hand side of the above equation can be expanded as below:

$$\sum_{j=1}^{k}\sum_{s=1}^{l}\left(\frac{|c'_j| \times f_j \times \lambda_{c_j}[s]}{f} - \frac{|c_j| \times f_j \times \lambda_{c'_j}[s]}{f}\right)^2$$
$$= \sum_{j=1}^{k}\sum_{s=1}^{l}\left(\frac{|c'_j| \times \lambda_{c_j}[s]}{|c_j| \times |c'_j|} - \frac{|c_j| \times \lambda_{c'_j}[s]}{|c_j| \times |c'_j|}\right)^2$$
$$= \sum_{j=1}^{k} \|c_j - c'_j\|^2.$$

Based on the above discussions, it is clear that evaluating the inequality $\sum_{j=1}^{k} \|c_j - c'_j\|^2 \leq \beta$ is the same as evaluating Equation 5.5. Hence, in our proposed PPODC protocol, we consider Equation 5.5 as the termination condition of k-means clustering and evaluate it in a privacy-preserving manner.

5.3 The Proposed Solution

In this section, we first introduce our system architecture and formalize the problem definition, followed by the discussion of a set of privacy-preserving primitives. Then, we present our novel PPODC protocol that utilizes the above transformation techniques and the privacy-preserving primitives as building blocks.

5.3.1 System Model and Problem Definition

In our problem, we consider N users denoted by $U_1, \ldots, U_N$. Assume that user U_i holds a database T_i with m_i data records and l attributes, for $1 \leq i \leq N$. Consider a scenario where the N users want to outsource their databases as well as the k-means clustering process on their combined databases to a cloud environment. In our system model, we consider two different entities: (i) the users and (ii) the cloud service providers. We assume that the users choose two cloud service providers C_1 and C_2 to perform the clustering task on their combined data.

We explicitly assume that C_1 and C_2 are semi-honest[1] [1] and they do not collude. After proper service level agreements with the users, C_2 generates a public-secret key pair (pk, sk) based on the Paillier cryptosystem [6] and broadcasts pk to all users and C_1. A more robust setting would be for C_1 and C_2 to jointly generate the public key pk based on the threshold Paillier cryptosystem (e.g., [19, 63]) such that the corresponding secret key sk is obliviously split between the two clouds. Under this case, the secret key sk is unknown to both clouds and only (random) shares of it are revealed to C_1 and C_2. For simplicity, we consider the former asymmetric setting where C_2 generates (pk, sk) in the rest of this chapter. Let $E_{pk}(\cdot)$ and $D_{sk}(\cdot)$ denote the encryption and decryption functions under Paillier cryptosystem and n denote the RSA modulus. We also recall that under the Paillier's encryption scheme, $E_{pk}(a) \times E_{pk}(b) \bmod n^2 = E_{pk}(a + b \bmod n), \forall\ a, b \in\ \mathbb{Z}_n$.

Given the above system architecture, we assume that user U_i encrypts T_i attribute-wise using pk and outsources the encrypted database to C_1. Another way to outsource the data is for users to split each attribute value in their database into two random shares and outsource the shares separately to each cloud (see Section 5.3.3 for more details). A detailed information flow between different entities in our system model is shown in Figure 5.1. Having outsourced the data, the main goal of a PPODC protocol is to enable C_1 and C_2 to perform k-means clustering over the combined encrypted databases in a privacy-preserving manner. More formally, we can define a PPODC protocol as follows:

$$\text{PPODC}(\langle T_1, \ldots, T_N \rangle, \beta) \rightarrow (S_1, \ldots, S_N) \tag{5.6}$$

where β is a threshold value agreed upon by all parties and used to check whether the termination condition holds in each iteration of the k-means algorithm as described in Section 5.1.2. Depending on the users' requirements, S_i (output received by U_i) can be the the global cluster centers and/or the final cluster IDs corresponding to the data records of U_i. In this chapter, we consider the former case under which S_i's

[1]Under the semi-honest model, each party follows the prescribed steps of the protocol, but is free to deduce any additional information based on the data it sees during the execution of the protocol.

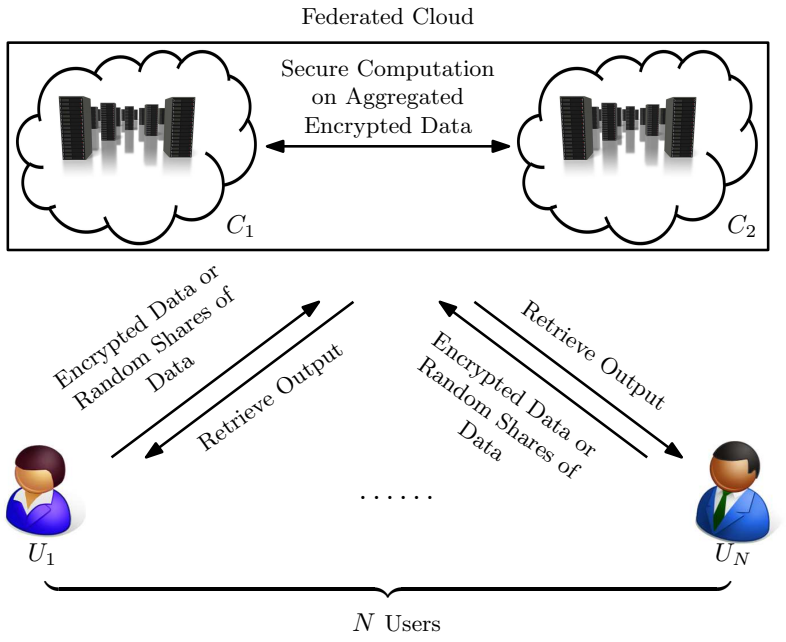


Figure 5.1.: The Proposed PPODC Architecture

are the same for all users. In general, a PPODC protocol should meet the following requirements:

- **Data Confidentiality:** The contents of U_i's database T_i should never be revealed to other users, C_1 and C_2.
- **Accuracy:** The output received by each party (i.e., S_i's) should be the same as in the standard k-means algorithm.
- **No Participation of Users:** Since the very purpose of outsourcing is to shift the users' load towards the cloud environment, a desirable requirement for any outsourced task is that the computations should be totally performed in the cloud. This will enable the users who do not have enough resources to participate in the clustering task to still get the desired results without compromising their privacy.

5.3.2 Privacy-Preserving Primitives

We discuss a set of privacy-preserving primitives under the above two-party (i.e., C_1 and C_2) computation model.

- Secure Multiplication (SMP): Assume that C_1 holds $\langle E_{pk}(a), E_{pk}(b)\rangle$ and C_2 holds sk, where $\langle a, b\rangle$ is unknown to both C_1 and C_2, the goal of the SMP protocol is to compute $E_{pk}(a \times b)$. During the execution of SMP, no information regarding the contents of a and b is revealed to C_1 and C_2.

- Secure Squared Euclidean Distance (SSED): In this protocol, C_1 holds two encrypted vectors $E_{pk}(X) = \langle E_{pk}(x[1], \ldots, E_{pk}(x[l])\rangle$ and $E_{pk}(Y) = \langle E_{pk}(y[1]), \ldots, E_{pk}(y[l])\rangle$. The goal of SSED is to compute the encryption of the squared Euclidean distance between X and Y. Specifically, the output is $E_{pk}((\|X - Y\|)^2)$. SSED reveals neither the contents of X and Y nor the Euclidean distance to C_1 and C_2.

- Secure Squared Order-Preserving Euclidean Distance (denoted by SSED_{OP}): Given that C_1 holds an encrypted data record, denoted by $E_{pk}(t_i)$, and an encrypted cluster, denoted by $E_{pk}(c_h)$, the goal of SSED_{OP} is for C_1 and C_2 to jointly compute $E_{pk}((\text{OPED}(t_i, c_h))^2)$. Here $E_{pk}(c_h) = \langle E_{pk}(\lambda_{c_h}), E_{pk}(|c_h|)\rangle$ and $E_{pk}(\lambda_{c_h}) = \langle E_{pk}(\lambda_{c_h}[1]), \ldots, E_{pk}(\lambda_{c_h}[l])\rangle$. Note that $\text{OPED}(t_i, c_h)$ denotes the Euclidean distance between data record t_i and cluster c_h based on the order-preserving Euclidean distance function defined in Equation 5.3. The output is revealed only to C_1.

- Secure Bit-Decomposition (SBD): Suppose C_1 holds $E_{pk}(z)$, where z is unknown to both parties and $0 \leq z < 2^w$, the goal of SBD is to compute encryptions of the individual bits of z. The output is $[z] = \langle E_{pk}(z_1), \ldots, E_{pk}(z_w)\rangle$, where z_1 (resp., z_w) denotes the most (resp. least) significant bit of z. In SBD, no contents regarding z is revealed to C_1 and C_2.

- Secure Comparison (SC): Given that C_1 holds $\langle [a], [b] \rangle$, the goal of SC is to securely compare a and b. Here $[a]$ and $[b]$ denote the encrypted bit vectors of a and b, respectively. The output is γ, where $\gamma = 1$ if $a \leq b$, and 0 otherwise. At the end, γ is known to both C_1 and C_2.

- Secure Minimum out of k Numbers (SMIN_k): In this protocol, we assume that C_1 holds k encrypted bit vectors, denoted by $[d_1], \ldots, [d_k]$, corresponding to integers $d_1, \ldots, d_k$. The goal of SMIN_k is to securely identify the array position corresponding to the minimum value among the k numbers. More specifically, if j^{th} integer is the minimum number among the k values, then the output of SMIN_k is an encrypted vector Γ such that Γ_j is $E_{pk}(1)$ and all the other entries contain encryptions of 0, where $j \in [1, k]$. The SMIN_k protocol should not reveal any information regarding the contents of k numbers (e.g., the minimum value or the array position corresponding to it, etc.) to C_1 and C_2.

Several solutions have been proposed for most of the above privacy-preserving primitives. Recently, Yousef et al. [64] discussed efficient implementations for SMP, SSED, and SBD. Also, an efficient solution to SC was proposed in [65]. We now propose implementations for SSED_{OP} and SMIN_k.

5.3.2.1 The SSED_{OP} Protocol

We discuss a novel solution to the SSED_{OP} problem using the SMP and SSED protocols as sub-routines. The main steps involved in the proposed SSED_{OP} protocol are highlighted in Algorithm 5.2. We assume that C_1 holds $\langle E_{pk}(c_1), \ldots, E_{pk}(c_k) \rangle$ and C_2 holds sk, where $c_1, \ldots, c_k$ denote k clusters and $E_{pk}(c_h) = \langle E_{pk}(\lambda_{c_h}), E_{pk}(|c_h|) \rangle$. Note that $E_{pk}(\lambda_{c_h}) = \langle E_{pk}(\lambda_{c_h}[1]), \ldots, E_{pk}(\lambda_{c_h}[l]) \rangle$. The goal is to compute $E_{pk}((\text{OPED}(t_i, c_h))^2)$ for a given input $E_{pk}(t_i)$ and $E_{pk}(c_h)$, where $1 \leq h \leq k$.

To start with, C_1 and C_2 securely compute the scaling factor for cluster c_h (in encrypted format based on Equation 5.2) using the extended secure multiplication protocol, denoted by SMP_{k-1}, that takes $k-1$ encrypted inputs and multiplies them

Algorithm 5.2 $\text{SSED}_{\text{OP}}(E_{pk}(t_i), E_{pk}(c_h))$

Require: C_1 has $E_{pk}(t_i)$, $E_{pk}(c_h) = \langle E_{pk}(\lambda_{c_h}), E_{pk}(|c_h|)\rangle$

1: C_1 and C_2:

(a) $b_h \leftarrow \text{SMP}_{k-1}(\tau_h)$, where $\tau_h = \cup_{j=1 \wedge j \neq h}^{k} E_{pk}(|c_j|)$

(b) $b' \leftarrow \text{SMP}(b_h, E_{pk}(|c_h|))$

(c) **for** $1 \leq s \leq l$ **do:**

- $a_i[s] \leftarrow \text{SMP}(b', E_{pk}(t_i[s]))$
- $a'_h[s] \leftarrow \text{SMP}(b_h, E_{pk}(\lambda_{c_h}[s]))$

(d) $E_{pk}((\text{OPED}(t_i, c_h))^2) \leftarrow \text{SSED}(a_i, a'_h)$

(within encryption). Specifically, they jointly compute $b_h = \text{SMP}_{k-1}(\tau_h)$, where $\tau_h = (E_{pk}(|c_j|))_{j \in [1,k] \wedge j \neq h}$. The important observation here is that $b_h = E_{pk}(\prod_{j=1 \wedge j \neq h}^{k} |c_j|)$ $= E_{pk}(\alpha_h)$, where α_h is the scaling factor for cluster c_h as defined in Equation 5.2. Then C_1 and C_2 securely multiply b_h with $E_{pk}(|c_h|)$ using SMP to get $b' = \text{SMP}(b_h, E_{pk}(c_h))$
$= E_{pk}(|c_1| \times \ldots \times |c_k|) = E_{pk}(\alpha)$, where α is the global scaling factor. After this, for $1 \leq s \leq l$, C_1 and C_2 jointly compute two encrypted vectors as follows:

$$a_i[s] = \text{SMP}(b', E_{pk}(t_i[s])) = E_{pk}(\alpha \times t_i[s]),$$
$$a'_h[s] = \text{SMP}(b_h, E_{pk}(\lambda_{c_h}[s])) = E_{pk}(\alpha_h \times \lambda_{c_h}[s]).$$

Finally, with the two encrypted vectors a_i and a'_h as C_1's input, C_1 and C_2 jointly compute the encrypted squared Euclidean distance between them using SSED. Specifically, the output of $\text{SSED}(a_i, a'_h)$ is $E_{pk}(\sum_{s=1}^{l}(\alpha \times t_i[s] - \alpha_h \times \lambda_{c_h}[s])^2)$. From Equation 5.3, it is clear that the output $\text{SSED}(a_i, a'_h)$ is equivalent to $E_{pk}((\text{OPED}(t_i, c_h))^2)$.

5.3.2.2 The SMIN_k Protocol

With respect to SMIN_k, the implementation discussed in [66] outputs the encryption of the minimum value (to C_1) among the k numbers as the output. On the other hand,

we require an encrypted vector as the output such that the entry corresponding to the minimum value is an encryption of 1 and all the other entries contain encryptions of 0. For this purpose, we consider the implementation given in [64] which is actually an extension to the SMIN_k protocol proposed in [66]. More specifically, the SMIN_k protocol given in [64] computes the encryption of the array index corresponding to the minimum value. We emphasize that the protocol in [64] can be easily extended to output the desired encrypted vector as follows:

- Let $E_{pk}(I_{\min})$ be the output (known only to C_1) computed using the SMIN_k protocol given in [64], where $I_{\min}$ denotes the index corresponding to the minimum value among the k input values. Now C_1 computes $E_{pk}(i - I_{\min})$ and randomizes it to get $\phi[i] = E_{pk}(r_i \times (i - I_{\min}))$, where r_i denotes a random number in $\mathbb{Z}_n$ and $1 \leq i \leq k$. Observe that exactly one of the entries in ϕ is equal to encryption of 0 (i.e., when $i = I_{\min}$) and the rest are encryptions of random values. Hereafter, $r \in_R \mathbb{Z}_n$ denotes a random number r in $\mathbb{Z}_n$.

- C_1 computes $u[\pi(i)] = \phi[i]$ and sends it to C_2, for $1 \leq i \leq k$. Here π is a random permutation function known only to C_1.

- Upon receiving u, C_2 decrypts it component-wise using sk to get $u'[i] = D_{sk}(u[i])$. After this, C_2 generates an encrypted vector U as follows. If $u'[i] = 0$, then $U[i] = E_{pk}(1)$, and $E_{pk}(0)$ otherwise. C_2 sends U to C_1.

- Finally, C_1 gets the desired encrypted vector as output by performing the inverse permutation on U, i.e., $\phi'[i] = U[\pi(i)]$, $1 \leq i \leq k$.

Example 5.3.1 *Suppose that $k = 5$ and the input to SMIN_k is $\langle [3], [6], [13], [2], [9] \rangle$, where $[x]$ denotes the encrypted bit vector corresponding to integer x. The output of the SMIN_k protocol given in [64] is $E_{pk}(I_{\min}) = E_{pk}(4)$ and it will be known only to C_1. Note that since '2' is the minimum among the five input values, the output is the encryption of the array index corresponding to '2' in the input list (i.e., $I_{\min} = 4$). After this, C_1 computes $\phi[1] = E_{pk}(r_1 \times (1-4))$, $\phi[2] = E_{pk}(r_2 \times (2-4))$, $\phi[3] = E_{pk}(r_3 \times (3-$*

4)), $\phi[4] = E_{pk}(r_4 \times (4-4))$, *and* $\phi[5] = E_{pk}(r_5 \times (5-4))$. *Without loss of generality, let the random permutation function* π *(known only to* C_1*) be as follows.* C_1 *computes* $u =$

$$\begin{array}{lcccccc} i & = & 1 & 2 & 3 & 4 & 5 \\ & & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\ \pi(i) & = & 2 & 5 & 1 & 3 & 4 \end{array}$$

$\langle \phi[3], \phi[1], \phi[4], \phi[5], \phi[2] \rangle$ *and sends* u *to* C_2. *Upon receiving,* C_2 *decrypts it using* sk *and identifies that* $D_{sk}(u[3]) = 0$. *Note that the rest of the values are random numbers. Then* C_2 *computes* $U = \langle E_{pk}(0), E_{pk}(0), E_{pk}(1), E_{pk}(0), E_{pk}(0) \rangle$ *and sends it to* C_1. *Finally,* C_1 *computes the final output as* $\phi' = \langle E_{pk}(0), E_{pk}(0), E_{pk}(0), E_{pk}(1), E_{pk}(0) \rangle$. □

In the rest of this chapter, SMIN_k refers to the implementation given in [64] combined with the above mentioned steps. We refer the reader to [64, 66] for more details.

5.3.3 The Proposed PPODC Protocol

In this sub-section, we discuss our proposed PPODC protocol which is based on the standard k-means algorithm discussed in Section 5.1.2. As mentioned in Section 5.3.1, our system model consists of N users denoted by $U_1, \ldots, U_N$. User U_j holds a database T_j of m_j data records with l attributes, for $1 \leq j \leq N$. Without loss of generality, let the aggregated database be $T = \bigcup_{j=1}^{N} T_j = \{t_1, \ldots, t_m\}$, where $m = \sum_{j=1}^{N} m_j$ denotes the total number of records in T. For simplicity, let $t_1 \ldots t_{m_1}$ belong to U_1, $t_{m_1+1}, \ldots, t_{m_1+m_2}$ belong to U_2, and so on. We assume that all users agree upon using two cloud service providers C_1 and C_2 for outsourcing their respective databases as well as the k-means clustering task. Remember that, in our system model, C_2 generates a public-secret key pair (pk, sk) based on the Paillier cryptosystem [6] and the public key pk is sent to all users and C_1.

After the users outsource their data (encrypted under pk) to C_1, the goal of PPODC is to enable C_1 and C_2 to jointly compute the global cluster centers using the aggregated encrypted data in a privacy-preserving manner. At a high level, our protocol computes the global cluster centers in an iterative manner until the pre-defined termination condition (given in Equation 5.5) holds.

The overall steps involved in the proposed PPODC protocol are given in Algorithms 5.3 and 5.4. The main steps are shown in Algorithm 5.3. Our protocol consists of the following three stages:

- **Stage 1 - Secure Data Outsourcing:**

  During this stage, each user U_j has to securely outsource an encrypted version of his/her database T_j to C_1. To minimize the data encryption costs of users, we achieve data outsourcing through randomization techniques. Note that this stage is run only once. At the end of this stage, only C_1 knows the (attribute-wise) encryptions of the N databases.

- **Stage 2 - Secure Computation of New Clusters:**

  In this stage, C_1 initially selects k data records at random (from the aggregated encrypted records) and assigns them as initial clusters (this step is the same as the initialization step in the traditional k-means algorithm). Then, C_1 and C_2 jointly assign each data record to a new cluster. After this, they compute the new cluster centers in encrypted format. The main goal of this stage is similar to the assignment and update stages given in Algorithm 1.

- **Stage 3 - Secure Termination or Update:**

  Upon computing the new cluster centers (in encrypted format), C_1 and C_2 securely verify whether the sum of the squared Euclidean distances between the current and new clusters is less than or equal to β (termination condition based on Equation 5.5). Here β denotes the pre-defined threshold value agreed upon by all the participating users. If the termination condition holds, then the protocol terminates returning the new cluster centers as the final output.

Otherwise, C_1 and C_2 update the current clusters to the new clusters and repeat Stages 2 and 3.

We emphasize that Stage 1 of PPODC is executed only once whereas Stages 2 and 3 are run in an iterative manner. We now discuss the steps in each of these three stages in detail.

5.3.3.1 Stage 1 - Secure Data Outsourcing (SDO)

Data are typically encrypted before being outsourced for privacy reasons. However, to avoid computation overhead on the users side due to having to encrypt their data, we consider the following approach for data outsourcing. User U_j generates two random shares for each attribute value of his/her data record t_i. More precisely, for the s^{th} attribute of data record t_i, U_j generates two random shares $(t_i^1[s], t_i^2[s])$ given by $t_i^1[s] = t_i[s] + r_i[s] \bmod n$ and $t_i^2[s] = n - r_i[s]$, where $r_i[s] \in_R \mathbb{Z}_n$ and $1 \leq s \leq l$. Observe that $t_i[s] = t_i^1[s] + t_i^2[s] \bmod n$. U_j outsources the random shares $t_i^1[s]$ and $t_i^2[s]$ to C_1 and C_2, respectively, instead of encrypting the database attribute-wise and outsourcing it to C_1. Thus, we are able to avoid heavy encryption costs on the users during the data outsourcing step. Here we assume that there exist secure communication channels, which can be established using standard mechanisms such as SSL, between user U_j and the two clouds C_1 and C_2. Each user U_j sends the random shares of his/her data to C_1 and C_2 separately through the secure communication channels.

Upon receiving the random shares of all data records, C_2 computes $E_{pk}(t_i^2[s])$ and sends it to C_1. Then, C_1 computes $E_{pk}(t_i[s]) = E_{pk}(t_i^1[s]) \times E_{pk}(t_i^2[s])$, for $1 \leq i \leq m$ and $1 \leq s \leq l$.

Algorithm 5.3 PPODC($\langle T_1, \ldots, T_N \rangle, \beta) \rightarrow (S_1, \ldots, S_N)$

Require: U_j holds a private database T_j; sk is known only to C_2

1: **for** $1 \leq i \leq m$**:** ▷ Step 1: **Secure Data Outsourcing**

(a) **for** $1 \leq s \leq l$**:**

- **if** $t_i \in T_j$ **then:**
  - U_j computes $t_i^1[s] = t_i[s] + r_i[s] \bmod n$, $t_i^2[s] = n - r_i[s]$, and $r_i[s]$ is a random number in $\mathbb{Z}_n^*$; sends $t_i^1[s]$ to C_1 and $t_i^2[s]$ to C_2
- C_2 sends $E_{pk}(t_i^2[s])$ to C_1
- C_1: $E_{pk}(t_i[s]) \leftarrow E_{pk}(t_i^1[s]) \times E_{pk}(t_i^2[s])$

2: C_1: ▷ Steps 2 to 4: **Secure Computation of New Clusters**

(a) Select k records at random and assign them to initial clusters denoted by $E_{pk}(\lambda_{c_1}), \ldots, E_{pk}(\lambda_{c_k})$, where $c_1, \ldots, c_k$ denote the current clusters

(b) $E_{pk}(|c_h|) \leftarrow E_{pk}(1)$, for $1 \leq h \leq k$

3: **for** $1 \leq i \leq m$ **do:**

(a) C_1 and C_2:

- $E_{pk}(d_i[h]) \leftarrow \text{SSED}_{\text{OP}}(E_{pk}(t_i), E_{pk}(c_h))$, for $1 \leq h \leq k$, where $E_{pk}(c_h) = \langle E_{pk}(\lambda_{c_h}), E_{pk}(|c_h|) \rangle$
- $[d_i[h]] \leftarrow \text{SBD}(E_{pk}(d_i[h]))$, for $1 \leq h \leq k$
- $\Gamma_i \leftarrow \text{SMIN}_k([d_i[1]], \ldots, [d_i[k]])$
- $\Lambda_{i,h}[s] \leftarrow \text{SMP}(\Gamma_{i,h}, E_{pk}(t_i[s]))$, for $1 \leq h \leq k$ and $1 \leq s \leq l$

Algorithm 5.3 PPODC($\langle T_1, \ldots, T_N \rangle, \beta$) $\rightarrow$ ($S_1, \ldots, S_N$) (Continued)

4: C_1:

(a) **for** $1 \leq h \leq k$ **do:**

- $W_h[s] \leftarrow \prod_{i=1}^{m} \Lambda_{i,h}[s]$, for $1 \leq s \leq l$
- $E_{pk}(|c'_h|) \leftarrow \prod_{i=1}^{m} \Gamma_{i,h}$

5: ▷ Steps 5 to 10: **Secure Termination or Update**

$\Omega \leftarrow \{\langle E_{pk}(\lambda_{c_1}), E_{pk}(|c_1|)\rangle, \ldots, \langle E_{pk}(\lambda_{c_k}), E_{pk}(|c_k|)\rangle\}$

6: $\Omega' \leftarrow \{\langle W_1, E_{pk}(|c'_1|)\rangle, \ldots, \langle W_k, E_{pk}(|c'_k|)\rangle\}$

7: $\gamma \leftarrow \text{SETC}(\Omega, \Omega')$ ▷ γ: the evaluation result of termination condition

8: **if** $\gamma = 1$ **then**, **for** $1 \leq h \leq k$ and $1 \leq s \leq l$

(a) C_1:

- $O_h[s] \leftarrow W_h[s] \times E_{pk}(r'_h[s])$ and $\delta_h \leftarrow E_{pk}(|c'_h|) \times E_{pk}(r''_h)$, where $r'_h[s]$ and $r''_h \in_R \mathbb{Z}_n$
- Send $O_h[s]$ and δ_h to C_2; $r'_h[s]$ and r''_h to each user U_j

(b) C_2: Send $O'_h[s] \leftarrow D_{sk}(O_h[s])$ and $\delta'_h \leftarrow D_{sk}(\delta_h)$ to each user U_j

9: **else for** $1 \leq h \leq k$

- $E_{pk}(\lambda_{c_h}) \leftarrow W_h$ and $E_{pk}(|c_h|) \leftarrow E_{pk}(|c'_h|)$
- Go to Step 3

10: U_j, **foreach** received pair $\langle O'_h, r'_h\rangle$ and $\langle \delta'_h, r''_h\rangle$ **do**:

(a) $\lambda_{c'_h}[s] = O'_h[s] - r'_h[s] \bmod n$, $1 \leq s \leq l$

(b) $|c'_h| \leftarrow \delta'_h - r''_h \bmod n$

(c) $\mu_{c'_h}[s] \leftarrow \frac{\lambda_{c'_h}[s]}{|c'_h|}$ and $S_j \leftarrow S_j \cup \{\mu_{c'_h}\}$

5.3.3.2 Stage 2 - Secure Computation of New Clusters

Given the (attribute-wise) encrypted versions of users data, during Stage 2 (denoted by SCNC), C_1 and C_2 jointly compute the new cluster centers in a privacy-preserving manner. To start with, C_1 randomly selects k encrypted data records (from the aggregated data) and assigns them as initial clusters. More specifically, the k encrypted data records are assigned to $E_{pk}(\lambda_{c_1}), \ldots, E_{pk}(\lambda_{c_k})$, respectively. For example, if the 3rd data record is selected as the first cluster c_1, then $E_{pk}(\lambda_{c_1}[s])$ is set to $E_{pk}(t_3[s])$, for $1 \leq s \leq l$. Also, $E_{pk}(|c_h|)$ is set to $E_{pk}(1)$ since each initial cluster c_h consists of only one data record, for $1 \leq h \leq k$.

For each encrypted data record $E_{pk}(t_i)$, C_1 and C_2 compute the squared Euclidean distance between t_i and all the clusters based on the order-preserving Euclidean distance function given in Equation 5.3. To achieve this, C_1 and C_2 jointly execute SSED_{OP} with $E_{pk}(t_i)$ and $E_{pk}(c_h)$ as C_1's private input, for $1 \leq i \leq m$ and $1 \leq h \leq k$, where $E_{pk}(c_h) = \langle E_{pk}(\lambda_{c_h}), E_{pk}(|c_h|)\rangle$. The output of SSED_{OP} is denoted by $E_{pk}(d_i[h])$, where $d_i[h] = (\text{OPED}(t_i, c_h))^2$. Now, C_1 and C_2 jointly execute the following set of operations:

- By using $E_{pk}(d_i[h])$ as C_1's private input to the secure bit-decomposition (SBD) sub-protocol, C_1 and C_2 securely compute encryptions of the individual bits of $d_i[h]$. The output $[d_i[h]] = \langle E_{pk}(d_{i,1}[h]), \ldots, E_{pk}(d_{i,w}[h])\rangle$ is known only to C_1, where $d_{i,1}[h]$ and $d_{i,w}[h]$ are the most and least significant bits of $d_i[h]$, respectively.

- For $1 \leq i \leq m$, with the k encrypted distances as C_1's private input to the secure minimum out of k numbers (SMIN_k) protocol, C_1 and C_2 compute an encrypted bit vector Γ_i. The important observation here is that $\Gamma_{i,g}$ is an encryption of 1 iff $d_i[g]$ is the minimum distance among $\langle d_i[1], \ldots, d_i[k]\rangle$. In this case, t_i is closest to cluster c_g, where $1 \leq g \leq k$. The rest of the values in Γ_i are encryptions of 0. Note that the output of SMIN_k, i.e., Γ_i, is known only to C_1.

- After this, C_1 and C_2 securely multiply $\Gamma_{i,h}$ with $E_{pk}(t_i[s])$ using the secure multiplication (SMP) sub-protocol. More precisely, C_1 and C_2 compute $\Lambda_{i,h}[s] = \text{SMP}(\Gamma_{i,h}, E_{pk}(t_i[s]))$. The observation here is that since $\Gamma_{i,g} = E_{pk}(1)$ only if t_i is closest to cluster c_g, $\Lambda_{i,g} = E_{pk}(t_i)$ denoting that t_i is assigned to new cluster c'_g. Also, $\Lambda_{i,h}$ is a vector of encryptions of 0, for $1 \leq h \leq k$ and $h \neq g$.

Next, C_1 computes the new cluster centers locally by performing homomorphic operations on $\Lambda_{i,h}$ and $\Gamma_{i,h}$ as follows:

- Compute (in encrypted format) the s^{th}-component of the numerator for the center of new cluster c'_h as $W_h[s] = \prod_{i=1}^{m} \Lambda_{i,h}[s]$, for $1 \leq h \leq k$ and $1 \leq s \leq l$. The observation here is that $W_h[s] = E_{pk}(\lambda_{c'_h}[s])$. Remember that $\mu_{c'_h}[s] = \frac{\lambda_{c'_h}[s]}{|c'_h|}$ denotes the center of cluster c'_h.

- Compute the encrypted number of data records that belong to the new cluster c'_h as $E_{pk}(|c'_h|) = \prod_{i=1}^{m} \Gamma_{i,h}$, for $1 \leq h \leq k$.

5.3.3.3 Stage 3 - Secure Termination or Update (STOU)

Given the new clusters (in encrypted format) resulting from Stage 2, the goal of Stage 3 is for C_1 and C_2 to verify whether the termination condition (based on Equation 5.5) holds in a privacy-preserving manner. If the termination condition holds, the new cluster centers are returned as the final output to U_j. Otherwise, the entire iterative process (i.e., Stages 2 and 3) is repeated by using the new clusters as the current clusters. The current and new clusters are $\Omega = \{\langle E_{pk}(\lambda_{c_1}), E_{pk}(|c_1|)\rangle, \ldots, \langle E_{pk}(\lambda_{c_k}), E_{pk}(|c_k|)\rangle\}$ and $\Omega' = \{\langle W_1, E_{pk}(|c'_1|)\rangle \ldots, \langle W_k, E_{pk}(|c'_k|)\rangle\})$, respectively.

First, by using the current and new clusters, C_1 and C_2 need to securely evaluate the termination condition (SETC) based on Equation 5.5.

The main steps involved in SETC are given in Algorithm 5.4 which we explain in detail below:

- To start with, C_1 and C_2 compute $\tau_i = E_{pk}(|c_i| \times |c'_i|)$ using $\langle E_{pk}(|c_i|), E_{pk}(|c'_i|)\rangle$ as C_1's private input to the SMP protocol, for $1 \leq i \leq k$. The output τ_i is known only to C_1.

- By using τ_i's, they compute $V_i = \text{SMP}_{k-1}(\tau'_i)$, where $\tau'_i = (\tau_j)_{j\in[1,k]\wedge j\neq i}$. Here SMP_{k-1} denotes the SMP protocol with $k-1$ encrypted inputs that need to be securely multiplied. More specifically, $V_i = E_{pk}(\prod_{j=1\wedge j\neq i}^{k} |c_i| \times |c'_i|)$, for $1 \leq i \leq k$. The important observation here is

$$V_i = E_{pk}\left(\prod_{j=1\wedge j\neq i}^{k} |c_i| \times |c'_i|\right) = E_{pk}(f_i)$$

  , where f_i is the scaling factor for cluster pair (c_i, c'_i) defined in Section 5.2.2. Then, they compute an encrypted value $Z_i = \text{SMP}(V_i, V_i) = E_{pk}(f_i^2)$.

- After this, they securely multiply V_1 and τ_1 using SMP protocol. The output of this step is

$$V = \text{SMP}(V_1, \tau_1) = E_{pk}\left(\prod_{j=1}^{k} |c_j| \times |c'_j|\right) = E_{pk}(f)$$

  , where f is the scaling factor for the termination condition as defined in Section 5.2.2. They compute $Y = \text{SMP}(V, V) = E_{pk}(f^2)$.

- For $1 \leq i \leq k$, C_1 and C_2 securely multiply each component in the current and new clusters with $|c'_i|$ and $|c_i|$, respectively. Specifically, for $1 \leq i \leq k$ and $1 \leq s \leq l$, they compute

$$G_i[s] = \text{SMP}(E_{pk}(\lambda_{c_i}[s]), E_{pk}(|c'_i|)) = E_{pk}(\lambda_{c_i}[s] \times |c'_i|),$$
$$G'_i[s] = \text{SMP}(W_i[s], E_{pk}(|c_i|)) = E_{pk}(\lambda_{c'_i}[s] \times |c_i|).$$

  Note that $W_i[s]$ computed in Stage 2 equals $E_{pk}(\lambda_{c'_i}[s])$.

- Now, by using the secure squared Euclidean distance (SSED) protocol with input vectors G_i and G'_i, C_1 and C_2 jointly compute $H_i = \text{SSED}(G_i, G'_i)$. More

precisely, they compute the encryption of squared Euclidean distance between vectors in G_i and G'_i given by

$$H_i = E_{pk}\left(\sum_{s=1}^{l}(\lambda_{c_i}[s] \times |c'_i| - \lambda_{c'_i}[s] \times |c_i|)^2\right).$$

- Given Z_i and H_i, C_1 and C_2 can securely multiply them using SMP to get

$$\begin{aligned} H'_i &= \text{SMP}(H_i, Z_i) \\ &= E_{pk}\left(f_i^2 \times \sum_{s=1}^{l}(\lambda_{c_i}[s] \times |c'_i| - \lambda_{c'_i}[s] \times |c_i|)^2\right). \end{aligned}$$

At the end of the above process, C_1 has $Y = E_{pk}(f^2)$ and H'_i, for $1 \leq i \leq k$. Now C_1 locally computes:

$$\begin{aligned} \beta' = Y^{\beta} &= E_{pk}(f^2 \times \beta) \quad \text{and} \\ M &= \prod_{i=1}^{k} H'_i \\ &= E_{pk}\left(\sum_{i=1}^{k}\sum_{s=1}^{l}(\lambda_{c_i}[s] \times f_i \times |c'_i| - \lambda_{c'_i}[s] \times f_i \times |c_i|)^2\right). \end{aligned}$$

At this point, C_1 has encryptions of the integers corresponding to both the left-hand and right-hand sides of the termination condition given in Equation 5.5. Therefore, the goal is to now securely compare them using the secure comparison (SC) protocol. However, the existing SC protocols (e.g., [65]) require encryptions of individual bits of the integers to be compared rather than the encrypted integers itself. Hence, we need to first convert the encrypted integers of the left-hand and right-hand sides into their respective encrypted bit representations using the secure bit-decomposition (SBD) protocol. Given β' and M, they can be securely compared as follows:

- C_1 and C_2 convert β' and M into their encrypted bit vectors using the SBD sub-protocol. Specifically, they compute $L = \text{SBD}(M) = [\sum_{i=1}^{k}\sum_{s=1}^{l}(\lambda_{c_i}[s] \times f_i \times |c'_i| - \lambda_{c'_i}[s] \times f_i \times |c_i|)^2)]$ and $R = \text{SBD}(\beta') = [f^2 \times \beta]$. Note that $[x]$ denotes the encrypted bit vector of an integer x. Also, remember that the outputs of SBD, i.e., L and R, are known only to C_1.

Algorithm 5.4 SETC(Ω, Ω')

Require: C_1 has $\Omega = \{\langle E_{pk}(\lambda_{c_1}), E_{pk}(|c_1|)\rangle, \ldots, \langle E_{pk}(\lambda_{c_k}), E_{pk}(|c_k|)\rangle\}$, $\Omega' = \{\langle W_1, E_{pk}(|c'_1|)\rangle \ldots, \langle W_k, E_{pk}(|c'_k|)\rangle\}$

1: C_1 and C_2:

(a) $\tau_i \leftarrow \text{SMP}(E_{pk}(|c_i|), E_{pk}(|c'_i|))$, for $1 \leq i \leq k$

(b) **for** $1 \leq i \leq k$ **do:**

- $V_i \leftarrow \text{SMP}_{k-1}(\tau'_i)$, where $\tau'_i = \cup_{j=1 \wedge j \neq i}^{k} \tau_j$
- $Z_i \leftarrow \text{SMP}(V_i, V_i)$

(c) $V \leftarrow \text{SMP}(V_1, \tau_1)$ and $Y \leftarrow \text{SMP}(V, V)$

(d) **for** $1 \leq i \leq k$ and $1 \leq s \leq l$ **do:**

- $G_i[s] \leftarrow \text{SMP}(E_{pk}(\lambda_{c_i}[s]), E_{pk}(|c'_i|))$
- $G'_i[s] \leftarrow \text{SMP}(W_i[s], E_{pk}(|c_i|))$

(e) $H_i \leftarrow \text{SSED}(G_i, G'_i)$, for $1 \leq i \leq k$

(f) $H'_i \leftarrow \text{SMP}(H_i, Z_i)$, for $1 \leq i \leq k$

2: C_1: $\beta' \leftarrow Y^{\beta}$ and $M \leftarrow \prod_{i=1}^{k} H'_i$

3: C_1 and C_2:

(a) $L \leftarrow \text{SBD}(M)$ and $R \leftarrow \text{SBD}(\beta')$

(b) $\gamma \leftarrow \text{SC}(L, R)$

- By using L and R as C_1's private input to the SC protocol, C_1 and C_2 securely evaluate the termination condition:

$$\sum_{i=1}^{k} \sum_{s=1}^{l} \left(\lambda_{c_i}[s] \times f_i \times |c'_i| - \lambda_{c'_i}[s] \times f_i \times |c_i|\right)^2 \leq f^2 \times \beta.$$

The output $\gamma = \text{SC}(L, R) = 1$ if the termination condition holds, and $\gamma = 0$ otherwise (here γ is known to C_1 and C_2).

Finally, once the termination condition has been securely evaluated, C_1 locally proceeds as follows:

- If $\gamma = 1$ (i.e., when the termination condition holds), the newly computed clusters are the final clusters which need to be sent to each user U_j. For this purpose, C_1 takes the help of C_2 to obliviously decrypt the results related to the new cluster centers. More specifically, C_1 initially picks random numbers $\langle r'_h[1], \ldots, r'_h[l], r''_h \rangle$ and computes $O_h[s] = W_h[s] \times E_{pk}(r'_h[s]) = E_{pk}(\lambda_{c'_h}[s] + r'_h[s] \bmod n)$ and $\delta_h = E_{pk}(|c'_h|) \times E_{pk}(r''_h) = E_{pk}(|c'_h| + r''_h \bmod n)$, for $1 \leq h \leq k$ and $1 \leq s \leq l$. After this, C_1 sends $O_h[s]$ and δ_h to C_2. In addition, C_1 sends $r'_h[s]$ and r''_h to each user U_j (through separate and secure communication channels).

- For $1 \leq h \leq k$, $1 \leq s \leq l$, C_2 successfully decrypts the received encrypted values using his/her secret share sk to get $O'_h[s] = D_{sk}(O_h[s])$ and $\delta'_h = D_{sk}(\delta_h)$ which it forwards to each user U_j (via secure communication channels). Observe that, due to the randomization by C_1, the values of $O'_h[s]$ and δ'_h are random numbers in $\mathbb{Z}_n$ from C_2's perspective.

- Upon receiving the entry pairs $\langle O'_h, r'_h \rangle$ and $\langle \delta'_h, r''_h \rangle$, user U_j removes the random factors to get $\lambda_{c'_h}[s] = O'_h[s] - r'_h[s] \bmod n$ and $|c'_h| = \delta'_h - r''_h \bmod n$, for $1 \leq h \leq k$ and $1 \leq s \leq l$. Finally, U_j computes the final cluster center as $\mu_{c'_h}[s] = \frac{\lambda_{c'_h}[s]}{|c'_h|}$ and adds it to his/her resulting set S_j.

- Otherwise, when $\gamma = 0$, C_1 locally updates the current clusters to new clusters by setting $E_{pk}(\lambda_{c_h}) = W_h$ and $E_{pk}(|c_h|) = E_{pk}(|c'_h|)$, for $1 \leq h \leq k$. Then, the above process is iteratively repeated until the termination condition holds, i.e., the protocol goes to Step 3 of Algorithm 5.3 and executes Steps 3 to 6 with the updated cluster centers as input.

5.3.4 Security Analysis

In this section, we show that the proposed PPODC protocol is secure under the standard semi-honest model [1]. Informally speaking, we stress that all the interme-

diate values seen by C_1 and C_2 in PPODC are either encrypted or pseudo-random numbers.

First, in the data outsourcing process (i.e., Step 1 of Algorithm 5.3), the values received by C_1 and C_2 are either random or pseudo-random values in $\mathbb{Z}_N$. At the end of the data outsourcing step, only C_1 knows the encrypted data records of all users and no information regarding the contents of T_j (the database of user U_j) is revealed to C_2. Therefore, as long as the underlying encryption scheme is semantically secure (which is the case in the Paillier cryptosystem [63]), the aggregated encrypted databases do not reveal any information to C_1. Hence, no information is revealed to C_1 and C_2 during Stage 1 of PPODC.

The implementations of SMP, SSED, SBD, and SMIN_k sub-protocols given in [64,67] are proven to be secure under the semi-honest model [1]. Also, the SC protocol given in [65] is secure under the semi-honest model. In the proposed SSED_{OP} protocol, the computations are based on using either SMP or SSED as a sub-routine. As a result, SSED_{OP} can be proven to be secure under the semi-honest model. Also, since SMIN_k is directly constructed from the solution given in [64], it can be proven secure under the semi-honest model. In short, the privacy-preserving primitives utilized in our protocol are secure under the semi-honest model.

We emphasize that the computations involved in Stages 2 and 3 of PPODC are performed by either C_1 locally or using one of the privacy-preserving primitives as a sub-routine. In the former case, C_1 operates on encrypted data locally. In the latter case, the privacy-preserving primitives utilized in our protocol are secure under the semi-honest model. Also, it is important to note that the output of a privacy-preserving primitive which is given as input to the next primitive is encrypted. Since we use a semantically secure Paillier encryption scheme [6], all the encrypted results (which are revealed only to C_1) from the privacy-preserving primitives do not reveal any information to C_1. Note that the secret key sk is unknown to C_1. By Composition Theorem [1], we claim that the sequential composition of the privacy-preserving primitives invoked in Stages 2 and 3 of our proposed PPODC protocol is secure under

the semi-honest model. Putting everything together, it is clear that PPODC is secure under the semi-honest model.

5.3.5 Complexity Analysis

In this subsection, we theoretically analyze the computational and communication costs incurred in each stage of the proposed PPODC protocol. The results regarding computational costs are given in Table 5.1. Here m denotes the sum of the numbers of data records of all users, l denotes the number of attributes, k denotes the number of clusters, $w_1 = 2k\log_2(m/k) + \log_2 l + 2\log_2(ub)$ is the maximum bit-length to express the order-preserving distance, and $w_2 = 4k\log_2(m/k) + \log_2 k + \log_2 l + 2\log_2(ub)$ is the maximum bit-length to express the left-hand side of Equation 5.5, where ub represents the maximum possible attribute value. It is important to note that Stage 1 of PPODC is run only once whereas Stage 2 and Stage 3 are run in an iterative manner until the termination condition holds.

Table 5.1.: Computational Costs for Different Stages in the Proposed PPODC Protocol

Stage	**Computational Costs**
Stage 1	$5m \cdot l$ multiplications
Stage 2 (per iteration)	$m \cdot (k \cdot (11l + 13w_1 + 10) - 10w_1 - 2)$ exponentiations $m \cdot (k \cdot (23l + 16w_1 + 1) - 16w_1 - 3) - k(l + 1)$ multiplications
Stage 3 (per iteration)	$k \cdot (16l + 15) + 16w_2 + 1$ exponentiations $k \cdot (32l + 30) + 17w_2 - 1$ multiplications

In addition, the total communication costs for each stage of PPODC are analyzed and the results are shown in Table 5.2. Here K denotes the size (in bits) of Paillier encryption key [6]. Following from our results, we can observe that the costs of Stage 2 are significantly higher than the costs incurred in Stage 3 in each iteration.

Table 5.2.: Communication Costs for the Proposed PPODC Protocol

Stage	**Communication Costs (in bits)**
Stage 1	$4m \cdot l \cdot K$
Stage 2	$(2m \cdot (6l \cdot k + 5w_1 \cdot k - 3(w_1 + 1)) + 20k) \cdot K$
Stage 3	$(3k^2 + 9k \cdot l + 5w_2 + 3k + 3) \cdot 2K$

5.3.6 Boosting the Performance of PPODC

We emphasize that a direct implementation of the PPODC protocol is likely to be inefficient due to involved expensive cryptographic operations. To address this issue, we propose two strategies to boost its performance: *(i) offline computation* and *(ii) pipelined execution*. In what follows, we extensively discuss how these two strategies are applicable to improving the performance of PPODC.

5.3.6.1 Offline Computation

In the Paillier cryptosystem [6], encryption of an integer $a \in \mathbb{Z}_n$ is given by $E_{pk}(a) = g^a \cdot r^n \bmod n^2$, where g is the generator, n is an RSA modulus, and r is a random number in $\mathbb{Z}_N^*$. It is clear that Paillier's encryption scheme requires two expensive exponentiation operations. In this chapter, we assume $g = 1+n$ (a commonly used setting that provides the same security guarantee as the original Paillier cryptosystem) as this allows for a more efficient implementation of Paillier encryption [68]. More specifically, when $g = 1+n$, we have $E_{pk}(a) = (1+n)^a \cdot r^n \bmod n^2 = (1+a \cdot n) \cdot r^n \bmod n^2$. As a result, an encryption under Paillier is reduced to one exponentiation operation. Our main observation from the above derivation is that the encryption cost under Paillier can be further reduced as follows. The exponentiation operation (i.e., $r^n \bmod n^2$) in the encryption function can be computed in an offline phase and thus the online cost of computing $E_{pk}(a)$ is reduced to two (inexpensive) multiplication operations[2].

[2]The time that takes to perform one exponentiation under $\mathbb{Z}_{n^2}$ is equivalent to $\log_2 n$ multiplication operations. Therefore, multiplication operation is inexpensive compared to exponentiation.

Additionally, the encryption of random numbers, 0s and 1s is independent of the underlying data and thus can be precomputed by the corresponding party (i.e., C_1 or C_2).

We emphasize that the actual online computation costs (with an offline phase) of the privacy-preserving primitives used in our protocol can be much less than their costs without an offline phase. For example, consider the secure multiplication (SMP) primitive with $E_{pk}(a)$ and $E_{pk}(b)$ as C_1's private input. During the execution of SMP, C_1 has to initially randomize the inputs and send them to C_2. That is, C_1 has to compute $E_{pk}(a)\cdot E_{pk}(r_1) = E_{pk}(a+r_1 \bmod n)$ and $E_{pk}(b)\cdot E_{pk}(r_2) = E_{pk}(b+r_2 \bmod n)$, where r_1 and r_2 are random numbers in $\mathbb{Z}_n$. This clearly requires C_1 to compute two encryptions: $E_{pk}(r_1)$ and $E_{pk}(r_2)$. However, since r_1 and r_2 are integers chosen by C_1 at random, the computation of $E_{pk}(r_1)$ and $E_{pk}(r_2)$ is independent of any specific instantiation of SMP. That is, C_1 can precompute $E_{pk}(r_1)$ and $E_{pk}(r_2)$ during the offline phase, thus boosting its online computation time. In a similar manner, C_1 and C_2 can precompute certain intermediate results in each privacy-preserving primitive.

5.3.6.2 Pipelined Execution

We are able to further reduce the online execution time by adopting the technique of pipelined execution. Take the execution of SMP for example, by which C_1 would like to compute $E_{pk}(a_1 \cdot b_1)$ and $E_{pk}(a_2 \cdot b_2)$ given $\langle E_{pk}(a_1), E_{pk}(b_1)\rangle$ and $\langle E_{pk}(a_2), E_{pk}(b_2)\rangle$, respectively. Here C_1 does not have to wait for C_2's response after sending $E_{pk}(a_1+r_{11})$ and $E_{pk}(b_1 + r_{12})$. Instead, after sending $E_{pk}(a_1 + r_{11})$ and $E_{pk}(b_1 + r_{12})$ to C_2, C_1 immediately computes $E_{pk}(a_2 + r_{21})$ and $E_{pk}(b_2 + r_{22})$. By observing that the time needed for C_2 to process $E_{pk}(a_1 + r_{11})$ and $E_{pk}(b_1 + r_{12})$ is approximately the same as the time needed for C_1 to process the intermediate result returned from C_2 afterward[3], we expect that we could further save at least half of the online execution time in the long run when we have a lot of SMP operations to perform.

[3]There will be two exponentiations for both C_1 and C_2.

Similarly, suppose that C_1 is given $E_{pk}(z_1)$ and $E_{pk}(z_2)$ and would like to invoke SBD to derive $\langle E_{pk}(z_{1,1}), \ldots, E_{pk}(z_{1,w})\rangle$ and $\langle E_{pk}(z_{2,1}), \ldots, E_{pk}(z_{2,w})\rangle$, where $z_{i,1}$ ($z_{i,w}$) denotes the most (resp. least) significant bit of z_i. We observe that C_1 does not have to sequentially compute $E_{pk}(z_{1,w})$, $\ldots$, $E_{pk}(z_{1,1})$, followed by the computation of $E_{pk}(z_{2,w}), \ldots, E_{pk}(z_{2,1})$[4]. Like what we have seen in the execution of SMP, to generate $E_{pk}(z_{1,w})$ from $E_{pk}(z_1)$, C_1 first computes $E_{pk}(z_1 + r_1)$ and sends it to C_2, followed by one exponentiation and some processing by C_2. The intermediate result generated by C_2 is then sent back to C_1 to derive $E_{pk}(z_{1,w})$ and the necessary information to derive $E_{pk}(z_{1,(w-1)})$ in the next iteration. By noticing that the time for C_2 to process $E_{pk}(z_1 + r_1)$ and the time for C_1 to process the intermediate result (from C_2) is approximately the same, i.e., one exponentiation on each party, C_1 could prepare $E_{pk}(z_2+r_2)$ immediately without waiting for the intermediate result from C_2 regarding $E_{pk}(z_1 + r_1)$. We expect to save around half of the online execution time by applying this technique to the SBD protocol.

5.4 Experimental Results

First of all, we emphasize that PPODC is 100% accurate in the sense that the outputs returned by PPODC and the standard k-means clustering algorithm (applied on the corresponding plaintext data) are the same. Therefore, in this section, we extensively analyze the running time of PPODC by performing various experiments using a real dataset under different parameter settings. Note that ours is the first work to address the PPODC problem requiring a federated cloud consisting of only two service providers and thus there exists no prior work to compare with our protocol.

[4]Recall that the SBD protocol securely extracts the encrypted bit vectors of z_1 and z_2 starting from the least significant bit to the most significant bit [67].

5.4.1 Implementation and Dataset Description

We implemented the protocols in C using the GNU Multiple Precision Arithmetic (GMP) library [69]. The experiments were conducted on a local cluster consisting of 16 computers [5], each with an Intel® Xeon™ CPU E5-5320 at 1.86GHz [6] and 8 GBytes of memory, running Linux version 3.17.7.

Figure 5.2 depicts our implementation model and shows how the cluster servers communicate with each other when executing the protocols over TCP/IP. The system consists of three components: 1) the master node, 2) a number (ψ) of servers performing the tasks of C_1, and 3) the same number of servers performing the tasks of C_2. Since we have 16 servers, we could have up to 8 pairs of servers performing the tasks needed in PPODC. The master node is directly connected with those servers of C_1, whereas each of C_1's server is paired with a corresponding server of C_2. The master node is in charge of the coordination of the execution of tasks needed in PPODC, i.e., within each iteration of k-means clustering, the master node needs to instruct each pair of servers about which task in an iteration to perform. When the task assigned to a pair of servers is complete, this pair will contact the master node for further instruction. Depending on the task just been accomplished, the master node would either assign the next task to this pair of servers, or put them to wait for others to finish.

For our experiments, we used the KEGG Metabolic Reaction Network (Undirected) dataset from the UCI KDD archive [70] that consists of 65,554 data records and 29 attributes. Since some of the attribute values are missing in the dataset, we removed the corresponding data records and the resulting dataset consists of 64,608 data records. As part of the pre-processing, we normalized the attribute values and scaled them into the integer domain $[0, 1000]$. Then we selected sample datasets (from the preprocessed data) by choosing data records at random based on the parameter

[5] We emphasize that this is not a dedicated cluster for our own computing tasks. It is a public cluster shared among users in the Computer Science department at Purdue University.

[6] We note that this is not a top-notch CPU. According to http://www.cpubenchmark.net/, it is at least 5 times slower than Intel® Core™ i7-4790K, a popular CPU widely used nowadays.

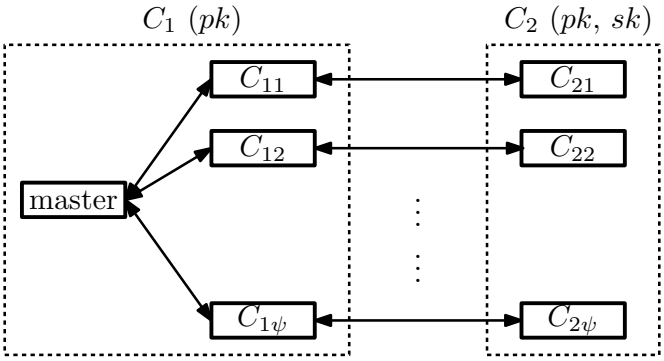


Figure 5.2.: A Cluster-Based Implementation Model

Table 5.3.: Computation time of privacy-preserving primitives when m = 6,000, $l = 10$, $k = 4$, and $\psi = 1$ (in milliseconds)

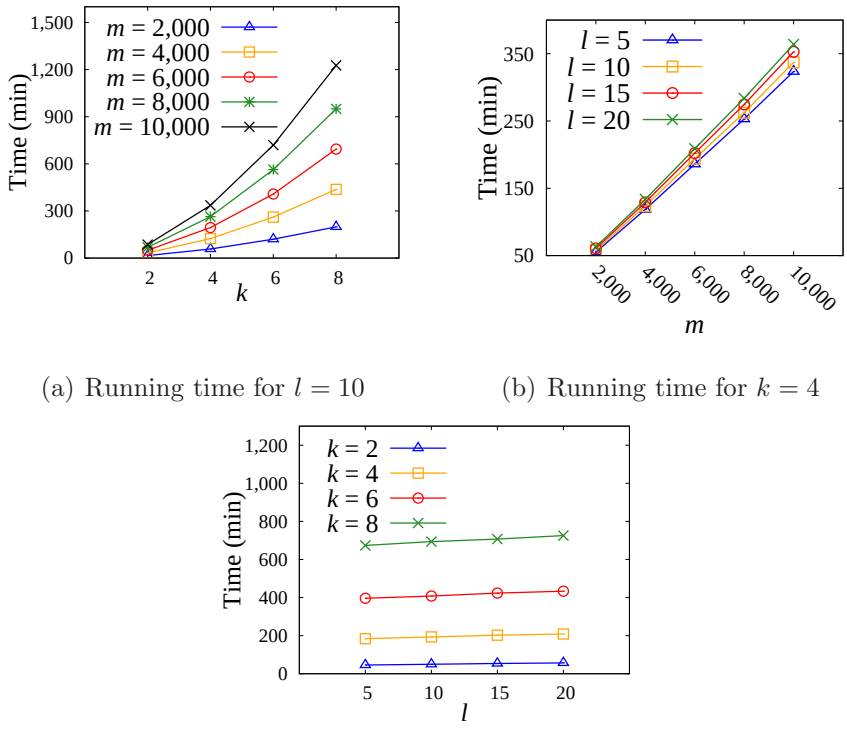
Primitive	Online + Offline	Online	Pipelined Online
SSED_{OP}	1,186	356	177
SBD	3,372	1,333	585
SMIN_k	33,486	16,021	11,897
SMP	59	23	11

values under consideration. We fixed the Paillier encryption key size to 1,024 bits (a commonly accepted key size) in all our experiments. For each sample dataset, we share each of its data record attribute-wise among the servers of C_1 and C_2 as mentioned in Stage 1 of PPODC. Note that the secret key sk is stored on C_2's servers.

5.4.2 Performance of PPODC

We evaluate the performance of our protocol based on the following parameters: the number of data records (m), the number of attributes in each data record (l), the number of clusters (k), and the number of pairs of servers (denoted by ψ) deployed when executing PPODC. On our local cluster, ψ varies from 1 to 8 as described previously. Also, we analyze the performance of PPODC based on different modes of execution: (i) the basic implementation without any optimization (denoted by *Online + Offline*), (ii) the implementation moving the computation of random ciphertexts to the offline phase (denoted by *Online*), (iii) the implementation that adopts the pipelined execution for SMP and SBD protocol assuming that all needed random ciphertexts are computed offline (denoted by *Pipelined Online*). We note that SMP is intensively used as a sub-protocol in both SSED_{OP} and SMIN_k. Therefore, to further reduce the online computation time, in the third execution mode of PPODC, we replace each invocation of the SMP protocol with its pipelined version. In the

following experiments, unless otherwise specified, we use $m = 6{,}000$, $l = 10$, $k = 4$, $\psi = 8$ and pipelined execution as the default parameters to execute PPODC.



(a) Running time for $l = 10$ (b) Running time for $k = 4$

(c) Running time for $m = 6{,}000$

Figure 5.3.: Pipelined online computation costs of PPODC for encryption key size 1,024 bits, $\psi = 8$, and varying values of l, k, and m

We first measure the running time of each secure primitive used in our protocol under different execution modes. To have a more accurate result, we execute each primitive on a single pair of servers for 100 times and compute the average. The results are shown in Table 5.3. From the results it can be seen that SSED_{OP}, SBD, and SMIN_k are much more expensive than SMP. This is as expected because the execution time needed by the first three primitives depends on m, l, and k as well. Precisely, SSED_{OP} depends on l, whereas SBD and SMIN_k are directly related to all those three parameters. Recall that the squared Order-Preserving Euclidean Distance

(OPED) between a record t_i and a cluster c_j could be expressed as $\alpha^2 \times \|t_i - c_j\|^2$, where $\alpha = \prod_{j=1}^{k} |c_j|$, the product of cardinalities of all clusters. It can be seen that $\text{OPED}(t_i, c_j)^2 \leq (m/k)^{2k} \times (ub^2) \times l$, where ub is the upperbound of attribute values for each dimension and is set to 1,000 in our experiment. In order for SMIN_k to correctly output a list of ciphertexts indicating the closest cluster for a given data record t_i, SBD should be invoked in a way that all the possible nonzero bits of $\text{OPED}(t_i, c_j)^2$ are preserved. For m = 6,000, l = 10, k = 4, we thus need to securely decompose $E_{pk}(\text{OPED}(t_i, c_j)^2)$ into $\langle E_{pk}(z_1), \ldots, E_{pk}(z_{108})\rangle$, where z_1 (resp., z_{108}) represents the most (resp., least) significant bit of $E_{pk}(\text{OPED}(t_i, c_j)^2)$. In Table 5.3, we can also observe that by moving the generation of random ciphertexts to the offline phase, we are able to save at least half of the online computation time for all 4 primitives. Furthermore, we see the effectiveness after pipelining the execution of SBD and SMP. Specifically, for SSED_{OP}, SBD, and SMP, we can save at least 81% of the online execution time if both those two optimizations are employed. For SMIN_k, we are able to save 64% of the online execution time as well.

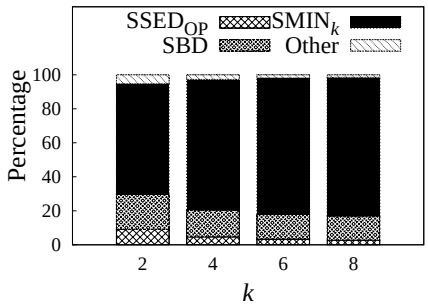


Figure 5.4.: Cost Breakdown when m = 6,000, l = 10, ψ = 8 under pipelined execution of SMP and SBD

Next, we break down the total execution time required by each sub-protocol used in PPODC. The results are presented in Figure 5.4. In Figure 5.4, we categorize the cost of PPODC into 4 parts, each denoting the time spent in each part respectively. We note that the part represented by 'Other' includes the time used by SMP, aggre-

gation of cluster information, as well as the secure evaluation of termination condition (SETC). From Figure 5.4 it can also be observed that when $m = 6{,}000, l = 10$, and $\psi = 8$, the percentage of the time consumed by SBD and SMIN_k grows for increasing k values. For instance, when $k = 2$, SBD and SMIN_k combined take 85% of the execution time, while they take 95% of the execution time when $k = 8$. The increased percentage of cost by SBD and SMIN_k comes from the fact that the time needed by these two protocols depends not only on the bit-length to express the order-preserving distances but also on the number of clusters. More precisely, when m and l are fixed, the bit-length to express the order-preserving distances is linear in $2k \cdot \log_2(m/k)$. Also, the number of encrypted distances to be bit-decomposed and then compared (by SMIN_k) also grows linearly in k. Hence, it can be seen that the time needed by SBD and SMIN_k grows quadratically in k, while the time needed in other parts (SSED_{OP} and SMP) mostly grows linearly in k.

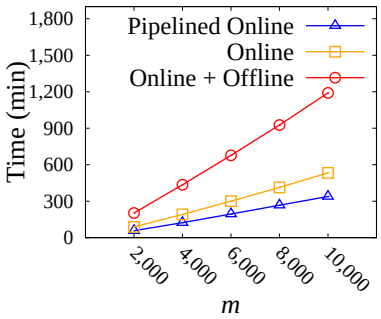


Figure 5.5.: m vs. execution mode for $l = 10, k = 4$, and $\psi = 8$

We also assess the execution time of PPODC for varying values of parameters m, l, and k. The results are given in Figure 5.3. In Figure 5.3(a), it can be seen that the online execution time of PPODC grows for increasing values of m and k. It is also obvious that the execution time is more sensitive to k than to m or l, which is expected because SMIN_k and SBD contribute to at least 85% of the execution time of PPODC and both of them have time complexity quadratic in k. In general, the total execution time of PPODC grows almost linear with the number of data records.

Similar behavior is seen in Figures 5.3(b) and 5.3(c). The total execution time is approximately linear in m but the variation in l only slightly affects the execution time.

We report the running time needed by PPODC under different execution modes for varying values of m in Figure 5.5. The execution time grows approximately linear with m. In addition, from Table 5.3, by moving the computation of random ciphertexts to the offline phase, it is clear that we save around 55% of the online execution time. If we further pipeline the execution of all invocations to SMP and SBD protocols, we save another 16% of the execution time compared to the basic implementation without any optimization. More specifically, it would take up to 1,190 minutes to perform one single iteration of PPODC (m = 10,000, $l = 10$, $k = 4$, and $\psi = 8$) if none of the optimizations is adopted, whereas only 337 minutes are needed when both optimizations are applied, a saving of 71% with respect to the online execution time. We further note that the average execution time is 293 minutes for each of C_1's server (excluding waiting time) and 183 minutes for each of C_2's server. The total execution time (337 minutes) needed is less than the sum of their respective execution time due to the pipelined execution of all instances of SMP and SBD protocols (including those invoked in SSED_{OP} and SMIN_k). To be specific, on average, 87% of the time is spent on computation for each of C_1's servers, whereas only 54% of the time is spent on computation for each of C_2's servers.

Lastly, we assess the effectiveness of parallel execution of the secure primitives involved in Stage 2 of our protocol. Recall that in our protocol we can almost achieve record level parallelism. To be precise, before the execution of PPODC, the master node will evenly distribute the m data records evenly to those ψ pairs of servers. Hence, when performing the Step 3 in PPODC, a pair of servers does not have to wait for other pairs of servers since a data record is assigned to exactly one pair of servers for processing. We thus expect the parallel execution of PPODC to be very effective, which is confirmed by Figure 5.6. When m = 10,000, $l = 10$, $k = 4$, it takes 2,676 minutes for 1 pair of servers to finish one iteration of PPODC, while the

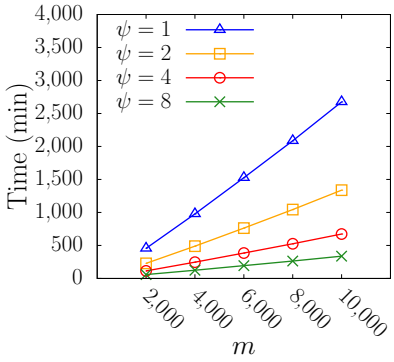


Figure 5.6.: m vs. number of server pairs for $l = 10$ and $k = 4$ under pipelined execution of SMP and SBD

time is reduced to 337 minutes when 8 pairs of servers are available. The speedup is 2,676/337 = 7.94, which is very close to 8.

5.5 Related Work

Several techniques have been proposed for the clustering task under the privacy-preserving data mining (PPDM) model (e.g., [53–56]). In PPDM, each user owns a piece of a dataset (typically a vertically or horizontally partitioned dataset) and the goal is for them to collaboratively perform the clustering task on the combined data in a privacy-preserving manner. On the other hand, our work is motivated by the cloud computing model where users can outsource their encrypted databases to a federated cloud environment. Under our problem setting, the federated cloud performs the clustering task over encrypted data and the users do not participate in any of the underlying computations. Hence, existing PPDM techniques for the clustering task are not applicable to the PPODC problem.

Only recently, researchers have started to focus on the clustering task in an outsourced environment (e.g., [62, 71, 72]). In [71], Upmanyu et al. give a solution to privacy-preserving k-means clustering for data records collected from n users based on secret sharing scheme where at least three non-colluding cloud service providers

are required to perform the computation. The work by Lin [72] proposes to utilize the randomized kernel matrix to protect the data privacy in the problem of outsourced kernel k-means computations. Only one service provider is needed in Lin's scheme. But, it is not clear that how much information is disclosed to the service provider since its security is not formally proved in the semi-honest model. The work by Liu et al. [62] is perhaps the most recent work along this direction. However, their solution has the following limitations: (i) it assumes that there is only a single user who wants to perform the clustering task on his/her own data and (ii) the user is required to execute certain intermediate computations and thus needs to be part of the clustering process. Unlike the approach in [62], our solution is proposed under the multi-user setting and the users can completely outsource the computations of the clustering task to a federated cloud environment in a privacy-preserving manner.

6 CONCLUSIONS AND FUTURE WORK

In this dissertation, based on widely adopted cryptographic primitives, we develop techniques that allow multiple mutually distrustful parties to conduct data analytics on their joint data. We demonstrate how to construct privacy-preserving protocols with formal security guarantees using those cryptographic primitives for three common tasks of data analytics under two different scenarios, i.e., the situation when the protocols are executed by the participating parties and the situation when the protocols are outsourced to some cloud service providers. In addition, two types of possible approaches to reducing the execution time needed to finish the process of collaborative data analytics are investigated. The first type of approaches allows for pruning unnecessary computation by disclosing a statistical controlled leakage to each participating party, whereas the second type of approaches reduces the required execution time of the participating parties by exploiting parallelism and pushing some computation to the offline phase without any additional leakage. The extensive experimental evaluation on real-world datasets shows that these approaches are effective.

Although in this dissertation we have investigated two types of approaches that allow us to reduce the required execution time of several common tasks of collaborative data analytics, these two types of approaches are still worth further exploration due to the ever increasing amount of data collected by different organizations under different scenarios. To be more specific, it is worth investigating whether or not we could identify other data analytics tasks where the notion of differential privacy can be properly applied to disclose statistics on input data to the participating parties in a controlled manner so as to reduce the invocations of cryptographic primitives. We have already seen in Chapter 3 and Chapter 4 that the notion of differential privacy could be very useful to boost the efficiency of private record linkage. Other than [35], which uses differential privacy in the scenario where two parties would like to perform

secure computation over graph-structured data, previous work has shown how such a privacy notion could be applied for constructing an efficient protocol for encrypted search [73], and an efficient protocol for ORAM execution [74]. All these examples show that approaches like ours have great potentials for improving the efficiency of secure multi-party computation.

On the other hand, it would also be beneficial to develop adaptive algorithms that automatically decide which cryptographic primitive to use given different types of tasks as well as different constraints on the computational and communication resources without sacrificing the data privacy. For instance, consider a simple task in which two parties each having an integer would like to compute the inner product of these two integers. Such a task could be carried out by using either the Paillier cryptosystem or Yao's garbled circuits. Suppose that the bit-length of each party's integer is at most 512-bit long and that the bit length of the RSA modulus used in the Paillier cryptosystem is 1,024. It can be seen that the computational complexity of the protocol based on the Paillier cryptosystem would be mostly dominated by one decryption operation, whereas the computational complexity of the protocol based on Yao's garbled circuits would exhibit a quadratic growth with respect to the bit-length of the integers. Hence, even though the garbled circuits based protocol incurs less computational overhead for smaller integers, its efficiency may be worse for much larger integers. Moreover, only 2,048 bits need to be transferred over the network for the protocol based on the Paillier encryption scheme, while at least $512^2 \cdot 4 \cdot 80$ bits have to be transferred for the protocol built upon Yao's garbled circuits[1], which may not be desirable when the network is congested. The example described above clearly emphasizes the importance of having adaptive mechanisms that could dynamically decide which cryptographic primitive to use under different input data size and various resource constraints.

[1]This is because at least 512^2 AND gates, each associated with 4 entries in a garbled truth table, have to be evaluated and we assume that the bit-length of each entry in a garbled truth table is 80.

REFERENCES

[1] Oded Goldreich. *The Foundations of Cryptography - Volume 2, Basic Applications*. Cambridge University Press, 2004.

[2] Rakesh Agrawal, Alexandre V. Evfimievski, and Ramakrishnan Srikant. Information sharing across private databases. In *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data, San Diego, California, USA, June 9-12, 2003*, pages 86–97, 2003.

[3] Oded Goldreich. *The Foundations of Cryptography - Volume 1, Basic Techniques*. Cambridge University Press, 2001.

[4] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *27th Annual Symposium on Foundations of Computer Science, Toronto, Canada, 27-29 October 1986*, pages 162–167, 1986.

[5] Vladimir Kolesnikov, Ahmad-Reza Sadeghi, and Thomas Schneider. Improved garbled circuit building blocks and applications to auctions and computing minima. In *Cryptology and Network Security, 8th International Conference, CANS 2009, Kanazawa, Japan, December 12-14, 2009. Proceedings*, pages 1–20, 2009.

[6] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Advances in Cryptology - EUROCRYPT '99, International Conference on the Theory and Application of Cryptographic Techniques, Prague, Czech Republic, May 2-6, 1999, Proceeding*, pages 223–238, 1999.

[7] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam D. Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of Cryptography, Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006, Proceedings*, pages 265–284, 2006.

[8] Frank McSherry and Kunal Talwar. Mechanism design via differential privacy. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2007), October 20-23, 2007, Providence, RI, USA, Proceedings*, pages 94–103, 2007.

[9] Ali Inan, Murat Kantarcioglu, Gabriel Ghinita, and Elisa Bertino. Private record matching using differential privacy. In *EDBT 2010, 13th International Conference on Extending Database Technology, Lausanne, Switzerland, March 22-26, 2010, Proceedings*, pages 123–134, 2010.

[10] Jianneng Cao, Fang-Yu Rao, Elisa Bertino, and Murat Kantarcioglu. A hybrid private record linkage scheme: Separating differentially private synopses from matching records. In *31st IEEE International Conference on Data Engineering, ICDE 2015, Seoul, South Korea, April 13-17, 2015*, pages 1011–1022, 2015.

[11] Xi He, Ashwin Machanavajjhala, Cheryl J. Flynn, and Divesh Srivastava. Composing differential privacy and secure computation: A case study on scaling private record linkage. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*, pages 1389–1406, 2017.

[12] Ivan P. Fellegi and Alan B. Sunter. A Theory for Record Linkage. *Journal of the American Statistical Association*, 64(328):1183–1210, 1969.

[13] Mehmet Kuzu, Murat Kantarcioglu, Ali Inan, Elisa Bertino, Elizabeth Durham, and Bradley Malin. Efficient privacy-aware record integration. In *Joint 2013 EDBT/ICDT Conferences, EDBT '13 Proceedings, Genoa, Italy, March 18-22, 2013*, pages 167–178, 2013.

[14] Vladimir Kolesnikov and Thomas Schneider. Improved garbled circuit: Free XOR gates and applications. In *Automata, Languages and Programming, 35th International Colloquium, ICALP 2008, Reykjavik, Iceland, July 7-11, 2008, Proceedings, Part II - Track B: Logic, Semantics, and Theory of Programming & Track C: Security and Cryptography Foundations*, pages 486–498, 2008.

[15] Michael Hay, Vibhor Rastogi, Gerome Miklau, and Dan Suciu. Boosting the accuracy of differentially private histograms through consistency. *PVLDB*, 3(1):1021–1032, 2010.

[16] Graham Cormode, Cecilia M. Procopiuc, Divesh Srivastava, Entong Shen, and Ting Yu. Differentially private spatial decompositions. In *IEEE 28th International Conference on Data Engineering (ICDE 2012), Washington, DC, USA (Arlington, Virginia), 1-5 April, 2012*, pages 20–31, 2012.

[17] `http://archive.ics.uci.edu/ml/datasets/Census+Income`, 1996.

[18] Frederick Douglas. `http://hms.isi.jhu.edu/acsc/damgard-jurik/`, 2012.

[19] Ivan Damgård and Mads Jurik. A generalisation, a simplification and some applications of paillier's probabilistic public-key system. In *Public Key Cryptography, 4th International Workshop on Practice and Theory in Public Key Cryptography, PKC 2001, Cheju Island, Korea, February 13-15, 2001, Proceedings*, pages 119–136, 2001.

[20] Yan Huang, David Evans, Jonathan Katz, and Lior Malka. Faster secure two-party computation using garbled circuits. In *20th USENIX Security Symposium, San Francisco, CA, USA, August 8-12, 2011, Proceedings*, 2011.

[21] Rob Hall and Stephen E. Fienberg. Privacy-preserving record linkage. In *Privacy in Statistical Databases - UNESCO Chair in Data Privacy, International Conference, PSD 2010, Corfu, Greece, September 22-24, 2010. Proceedings*, pages 269–283, 2010.

[22] Ali Al-Lawati, Dongwon Lee, and Patrick D. McDaniel. Blocking-aware private record linkage. In *IQIS 2005, International Workshop on Information Quality in Information Systems, 17 June 2005, Baltimore, Maryland, USA (SIGMOD 2005 Workshop)*, pages 59–68, 2005.

[23] Luca Bonomi, Li Xiong, Rui Chen, and Benjamin C. M. Fung. Frequent grams based embedding for privacy preserving record linkage. In *21st ACM International Conference on Information and Knowledge Management, CIKM'12, Maui, HI, USA, October 29 - November 02, 2012*, pages 1597–1601, 2012.

[24] Alexandros Karakasidis and Vassilios S. Verykios. Privacy preserving record linkage using phonetic codes. In *2009 Fourth Balkan Conference in Informatics, BCI 2009, Thessaloniki, Greece, 17-19 September 2009*, pages 101–106, 2009.

[25] Monica Scannapieco, Ilya Figotin, Elisa Bertino, and Ahmed K. Elmagarmid. Privacy preserving schema and data matching. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, Beijing, China, June 12-14, 2007*, pages 653–664, 2007.

[26] Rainer Schnell, Tobias Bachteler, and Jörg Reiher. Privacy-preserving record linkage using bloom filters. *BMC Med. Inf. & Decision Making*, 9:41, 2009.

[27] Mohamed Yakout, Mikhail J. Atallah, and Ahmed K. Elmagarmid. Efficient private record linkage. In *Proceedings of the 25th International Conference on Data Engineering, ICDE 2009, March 29 2009 - April 2 2009, Shanghai, China*, pages 1283–1286, 2009.

[28] Burton H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Commun. ACM*, 13(7):422–426, 1970.

[29] Michael J. Freedman, Kobbi Nissim, and Benny Pinkas. Efficient private matching and set intersection. In *Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings*, pages 1–19, 2004.

[30] Lea Kissner and Dawn Xiaodong Song. Privacy-preserving set operations. In *Advances in Cryptology - CRYPTO 2005: 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005, Proceedings*, pages 241–257, 2005.

[31] Wahbeh H. Qardaji, Weining Yang, and Ninghui Li. Differentially private grids for geospatial data. In *29th IEEE International Conference on Data Engineering, ICDE 2013, Brisbane, Australia, April 8-12, 2013*, pages 757–768, 2013.

[32] Jun Zhang, Xiaokui Xiao, and Xing Xie. Privtree: A differentially private algorithm for hierarchical decompositions. In *Proceedings of the 2016 International Conference on Management of Data, SIGMOD Conference 2016, San Francisco, CA, USA, June 26 - July 01, 2016*, pages 155–170, 2016.

[33] Noman Mohammed, Dima Alhadidi, Benjamin C. M. Fung, and Mourad Debbabi. Secure two-party differentially private data release for vertically partitioned data. *IEEE Trans. Dependable Sec. Comput.*, 11(1):59–71, 2014.

[34] Yuan Hong, Jaideep Vaidya, Haibing Lu, Panagiotis Karras, and Sanjay Goel. Collaborative search log sanitization: Toward differential privacy and boosted utility. *IEEE Trans. Dependable Sec. Comput.*, 12(5):504–518, 2015.

[35] Sahar Mazloom and S. Dov Gordon. Secure computation with differentially private access patterns. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018*, pages 490–507, 2018.

[36] Boaz Barak, Kamalika Chaudhuri, Cynthia Dwork, Satyen Kale, Frank McSherry, and Kunal Talwar. Privacy, accuracy, and consistency too: a holistic solution to contingency table release. In *Proceedings of the Twenty-Sixth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 11-13, 2007, Beijing, China*, pages 273–282, 2007.

[37] Bolin Ding, Marianne Winslett, Jiawei Han, and Zhenhui Li. Differentially private data cubes: optimizing noise sources and consistency. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2011, Athens, Greece, June 12-16, 2011*, pages 217–228, 2011.

[38] Wahbeh H. Qardaji, Weining Yang, and Ninghui Li. Priview: practical differentially private release of marginal contingency tables. In *International Conference on Management of Data, SIGMOD 2014, Snowbird, UT, USA, June 22-27, 2014*, pages 1435–1446, 2014.

[39] Jun Zhang, Graham Cormode, Cecilia M. Procopiuc, Divesh Srivastava, and Xiaokui Xiao. Privbayes: private data release via bayesian networks. In *International Conference on Management of Data, SIGMOD 2014, Snowbird, UT, USA, June 22-27, 2014*, pages 1423–1434, 2014.

[40] Rui Chen, Qian Xiao, Yu Zhang, and Jianliang Xu. Differentially private high-dimensional data publication via sampling-based inference. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Sydney, NSW, Australia, August 10-13, 2015*, pages 129–138, 2015.

[41] Vladimir I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10(8):707–710, 1966.

[42] Jean Bourgain. On lipschitz embedding of finite metric spaces in hilbert space. *Israel Journal of Mathematics*, 52(1-2):46–52, 1985.

[43] Gísli R. Hjaltason and Hanan Samet. Properties of embedding methods for similarity searching in metric spaces. *IEEE Trans. Pattern Anal. Mach. Intell.*, 25(5):530–549, 2003.

[44] Gonzalo Navarro. A guided tour to approximate string matching. *ACM Comput. Surv.*, 33(1):31–88, 2001.

[45] `http://aiweb.cs.washington.edu/research/projects/xmltk/xmldata/www/repository.html`.

[46] Martin Beck and Florian Kerschbaum. Approximate two-party privacy-preserving string matching with linear complexity. In *IEEE International Congress on Big Data, BigData Congress 2013, June 27 2013-July 2, 2013*, pages 31–37, 2013.

[47] Xiao Shaun Wang, Yan Huang, Yongan Zhao, Haixu Tang, XiaoFeng Wang, and Diyue Bu. Efficient genome-wide, privacy-preserving similar patient query based on private edit distance. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, October 12-16, 2015*, pages 492–503, 2015.

[48] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: A review. *ACM Comput. Surv.*, 31(3):264–323, September 1999.

[49] Patrick Pantel and Dekang Lin. Document clustering with committees. In *SIGIR 2002: Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, August 11-15, 2002, Tampere, Finland*, pages 199–206, 2002.

[50] Ryszard S. Michalski and Robert E." Stepp. *Learning from Observation: Conceptual Clustering*, pages 331–363. Springer Berlin Heidelberg, Berlin, Heidelberg, 1983.

[51] Andrea Baraldi and Palma Blonda. A survey of fuzzy clustering algorithms for pattern recognition. I. *IEEE Trans. Systems, Man, and Cybernetics, Part B*, 29(6):778–785, 1999.

[52] Andrea Baraldi and Palma Blonda. A survey of fuzzy clustering algorithms for pattern recognition. II. *IEEE Trans. Systems, Man, and Cybernetics, Part B*, 29(6):786–801, 1999.

[53] Chunhua Su, Jianying Zhou, Feng Bao, Tsuyoshi Takagi, and Kouichi Sakurai. Two-party privacy-preserving agglomerative document clustering. In *Information Security Practice and Experience, Third International Conference, ISPEC 2007, Hong Kong, China, May 7-9, 2007, Proceedings*, pages 193–208, 2007.

[54] Jaideep Vaidya and Chris Clifton. Privacy-preserving k-means clustering over vertically partitioned data. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, August 24 - 27, 2003*, pages 206–215, 2003.

[55] Geetha Jagannathan and Rebecca N. Wright. Privacy-preserving distributed k-means clustering over arbitrarily partitioned data. In *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Chicago, Illinois, USA, August 21-24, 2005*, pages 593–599, 2005.

[56] Paul Bunn and Rafail Ostrovsky. Secure two-party k-means clustering. In *Proceedings of the 2007 ACM Conference on Computer and Communications Security, CCS 2007, Alexandria, Virginia, USA, October 28-31, 2007*, pages 486–497, 2007.

[57] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*, pages 169–178, 2009.

[58] Craig Gentry and Shai Halevi. Implementing gentry's fully-homomorphic encryption scheme. In *Advances in Cryptology - EUROCRYPT 2011 - 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn, Estonia, May 15-19, 2011. Proceedings*, pages 129–148, 2011.

[59] S. Lloyd. Least squares quantization in pcm. *IEEE Trans. Inf. Theor.*, 28(2):129–137, September 2006.

[60] Keinosuke Fukunaga. *Introduction to Statistical Pattern Recognition (2Nd Ed.)*. Academic Press Professional, Inc., San Diego, CA, USA, 1990.

[61] Robert B. Bohn. Us government cloud computing technology roadmap volume i: High-priority requirements to further usg agency cloud computing adoption; and volume ii: Useful information for cloud adopters. `https://www.nist.gov/publications/us-government-cloud-computing-technology-roadmap-volume-i-high-priority-requirements`, November 2011.

[62] Dongxi Liu, Elisa Bertino, and Xun Yi. Privacy of outsourced k-means clustering. In *9th ACM Symposium on Information, Computer and Communications Security, ASIA CCS '14, Kyoto, Japan - June 03 - 06, 2014*, pages 123–134, 2014.

[63] Carmit Hazay, Gert Læssøe Mikkelsen, Tal Rabin, and Tomas Toft. Efficient RSA key generation and threshold paillier in the two-party setting. In *Topics in Cryptology - CT-RSA 2012 - The Cryptographers' Track at the RSA Conference 2012, San Francisco, CA, USA, February 27 - March 2, 2012. Proceedings*, pages 313–331, 2012.

[64] Bharath K. Samanthula, Yousef Elmehdwi, and Wei Jiang. k-nearest neighbor classification over semantically secure encrypted relational data. *IEEE Trans. Knowl. Data Eng.*, 27(5):1261–1273, 2015.

[65] Ian F. Blake and Vladimir Kolesnikov. One-round secure comparison of integers. *J. Mathematical Cryptology*, 3(1):37–68, 2009.

[66] Yousef Elmehdwi, Bharath K. Samanthula, and Wei Jiang. Secure k-nearest neighbor query over encrypted data in outsourced environments. In *IEEE 30th International Conference on Data Engineering, Chicago, ICDE 2014, IL, USA, March 31 - April 4, 2014*, pages 664–675, 2014.

[67] Bharath K. Samanthula, Chun Hu, and Wei Jiang. An efficient and probabilistic secure bit-decomposition. In *8th ACM Symposium on Information, Computer and Communications Security, ASIA CCS '13, Hangzhou, China - May 08 - 10, 2013*, pages 541–546, 2013.

[68] Ivan Damgård, Mads Jurik, and Jesper Buus Nielsen. A generalization of paillier's public-key system with applications to electronic voting. *Int. J. Inf. Sec.*, 9(6):371–385, 2010.

[69] `http://gmplib.org/`.

[70] Muhammad Naeem and Sohail Asghar. Kegg metabolic reaction network data set. `https://archive.ics.uci.edu/ml/datasets/KEGG+Metabolic+Reaction+Network+(Undirected)`, 2011.

[71] Maneesh Upmanyu, Anoop M. Namboodiri, Kannan Srinathan, and C. V. Jawahar. Efficient privacy preserving k-means clustering. In *Intelligence and Security Informatics, Pacific Asia Workshop, PAISI 2010, Hyderabad, India, June 21, 2010. Proceedings*, pages 154–166, 2010.

[72] Keng-Pei Lin. Privacy-preserving kernel k-means outsourcing with randomized kernels. In *13th IEEE International Conference on Data Mining Workshops, ICDM Workshops, TX, USA, December 7-10, 2013*, pages 860–866, 2013.

[73] Georgios Kellaris, George Kollios, Kobbi Nissim, and Adam O'Neill. Accessing data while preserving privacy. *CoRR*, abs/1706.01552, 2017.

[74] Sameer Wagh, Paul Cuff, and Prateek Mittal. Root ORAM: A tunable differentially private oblivious RAM. *CoRR*, abs/1601.03378, 2016.

VITA

Fang-Yu Rao was born in Taipei City, Taiwan. He graduated with a Bachelor degree and a Master degree in the Department of Computer Science and Engineering, both from National Sun Yat-sen University (NSYSU) in Kaohsiung City, Taiwan. He started as a Ph.D. student in the Department of Computer Science at Purdue University in 2008. He worked on topics related to privacy-preserving data analytics under the supervision of Dr. Elisa Bertino. Fang-Yu received his Master degree in Computer Science in December 2018, and his Ph.D. in Computer Science in Spring 2019.