Genomic Data Security and Privacy

by

Md Momin Al Aziz

A thesis submitted to
The Faculty of Graduate Studies of
The University of Manitoba
in partial fulfillment of the requirements
of the degree of

Master of Science

Department of Computer Science
The University of Manitoba
Winnipeg, Manitoba, Canada
May 2017

Thesis advisor

Noman Mohammed

Author

Md Momin Al Aziz

Genomic Data Security and Privacy

Abstract

Genomic data holds salient information about the characteristics of a living organism. Throughout the last decade, pinnacle developments have given us more accurate and inexpensive methods to retrieve genome sequences of human beings. However, with the advancement of genomic research, there is a growing privacy concern regarding the collection, storage, and analysis of such sensitive data. Recent research results show that given some background information, it is possible for an adversary to re-identify an individual from any genomic dataset. In this thesis, we examine various data sharing models and study the potential privacy attacks in different real-life data sharing scenarios. We then propose appropriate privacy-preserving solutions using cryptographic and statistical techniques. Experimental results show that our proposed solutions are scalable and can guarantee both utility and privacy of the genomic data.

Contents

List of Figures

List of Tables

Acknowledgments

I would like to express my gratitude to Almighty God for granting me the opportunity to write this thesis.

I would like to thank my supervisor, Dr Noman Mohammed for giving me the opportunity to work under his supervision. I am very grateful for his guidance, suggestions, and feedback throughout the preparation of this thesis. I am thankful to the other members of my committee, Drs. Neil D. B. Bruce, Olivier Tremblay-Savard and Shuang Wang for the comment and advice they provided.

I am also grateful to all my collaborators, Drs. Dima Alhadidi, Xiaoqian Jiang, Pourang Irani, Reza Ghasemi (in no particular order) for their guidance. I am also thankful to my colleagues Md Waliullah, Zahidul Hasan, Nazmus Sadat, Kazi Wasif Ahmed and Safiur Rahman Mahdi for their valuable consults and time. I am equally grateful to all the benevolent faculties of UofManitoba Computer Science and Bangladesh University of Engineering and Technology who shaped my research and academic standing.

Finally, I would like to thank *my wife and family* for their countless sacrifices and inspirations throughout the years.

Dedicated to all the sacrifices ...

Chapter 1

Introduction

1.1 Motivation

Seminal advancement in genomic data generation (as shown in Figure 1.1) over the past decade has impacted health science and related scientific studies. The genesis in data accumulation has made the scientific studies on multiple genre of medical genomics more realistic [1]. Throughout the world, different and larger genomic datasets help the researchers to understand the relation between our genomic codes and our health. Researchers can now analyze susceptibility to specific disease or formulate personalized medicine based on the patient's genomic sequence [4; 5]. Therefore, the continuation and advancement of medical research heavily relies on the availability of these datasets and their ceaseless growth.

Genomic data is highly sensitive as it indicates the current and future susceptibility to specific disease for an individual and sometimes for his relatives as well. Therefore, genomic sequences impose a greater privacy risk (see Table 1.1). In addition, genomic databases are often owned by different organiza-

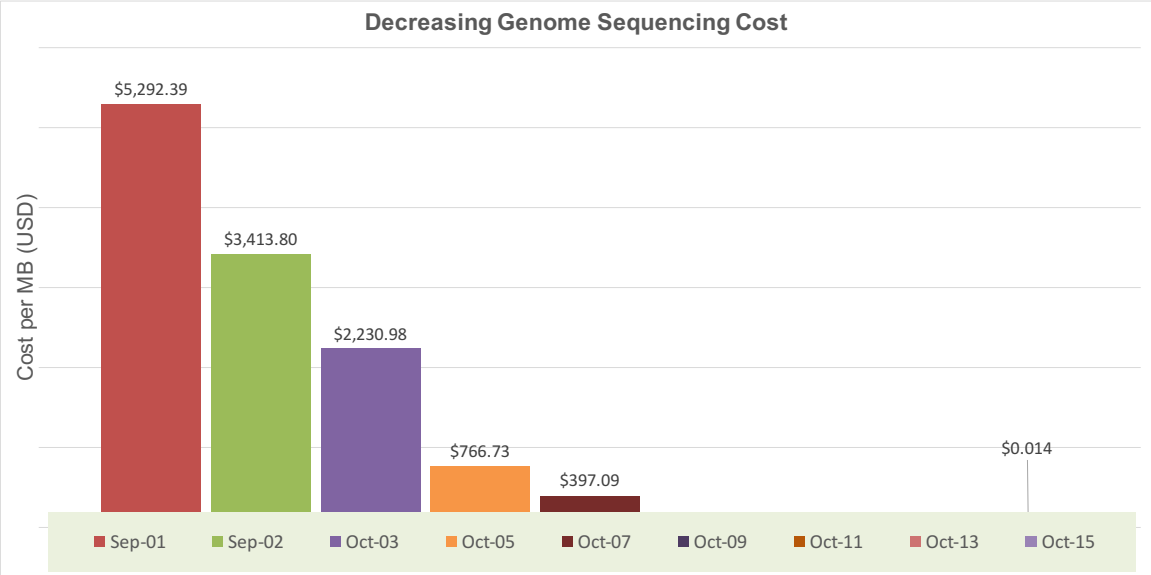


Figure 1.1: Decreasing cost for per MB of genomic data from 2001 to 2015 [1]

tions making it not available for public usage. Moreover, these databases require a high-performance computation and a large amount of storage capacity to be processed. New paradigm such as cloud computing enables organizations to execute large-scale computations for a lower price. This makes Infrastructure-as-a-Service (IaaS) and Storage-as-a-Service (SaaS) compelling services for genomic data researchers. However, despite the many benefits of the cloud computing architecture, health care institutes are reluctant to adopt the IaaS and SaaS models because they require outsourcing the data or computation to an untrusted cloud service provider which might result in a security or privacy violation. Numerous attack models have been developed to infer information from this shared data and those can be classified into two major areas [6; 7; 8]: *a*) re-identification attack from background knowledge or other online data and *b*) inference about physical attributes (phenotype) or disease association . Prominent privacy attacks targeting genomic data are summarized and chronologically shown and in Table 1.1.

An intuitive and popular approach to mitigate this problem is to enforce privacy policies regarding sharing the data. This strategy is much popular as different laws and regulations are followed in different institutions around the globe which regulate the sharing and disclosure of these sensitive data. Though these policies are protecting the privacy of the participating individuals, they have some serious demerits. For example, the time needed by a governing body to review researchers' applications requesting the access of datasets is lengthy. This affects the timely outcome of the research and demotivates researchers to pursue such studies. It also restricts cross border data sharing as these databases are often regulated to be kept inside a certain country or a province.

Cryptography is a mature area of research and the use of different encryption techniques can ensure individual privacy and the security of the data even on an untrusted environment. There have been some seminal developments in multiple crypto primitives in recent years which allow researchers to carry out computations on encrypted data, maintaining integrity of different inputs from different data owners and generating privacy preserving outputs. However these techniques come with a cost in prolonged execution time in some cases. Hence, developing a secure system using various cryptographic techniques that guarantees both privacy and utility of genomic data is an important research problem.

1.2 Contributions

This thesis addresses four research problems about secure and private genomic data management. Following we detail the technical contributions of this thesis.

Table 1.1: Privacy attacks proposed on genomic data

Author	**Year**	**Summary**
Sweeney [9]	2000	Identifying 87% of US citizens with ZIP code, gender, date of birth
Gottlib [10]	2001	Finding employees who are susceptible to genetic diseases depending on genomic data
Lin *et al.* [11]	2004	Identifying a person by only 75 independent SNPs
Homer *et al.* [12]	2008	Telling if a user is present from a DNA mixture
Kong *et al.* [13]	2008	Inferring genotype of offspring using parents' gene
Wang *et al.* [14]	2009	Using some hundred SNPs to re-identify one person from a GWAS study (an extension of [12])
Naik [15]	2009	Finding biological father with open access genomic
Goodrich[16]	2009	Revealing information about the full identity of a genomic query sequence after encryption
Humbert *et al.* [17]	2013	Inferring close relatives' genomes using statistical inference
Gymrek *et al.* [18]	2013	Identifying personal genomes from surnames with Y-chromosome
Fredrikson *et al.* [19]	2014	Predicting genetic markers using machine learning models on differentially private data
Shringarpure *et al.* [3]	2015	Identifying participants from a genomic beacon service with 1000 queries

Secure and Efficient Multiparty Computation on Genomic Data

Large scale biomedical research projects involve analysis of huge amount of genomic data which is owned by different data owners. The collection and storing of genomic data is sometimes beyond the capability of a sole organization. Genomic data sharing is a feasible solution to overcome this problem. This scenario can

be generalized into the problem of aggregating distributed data among multiple databases that are owned by different data owners. However, it is important to guarantee that an adversary cannot learn anything about the data or the individual contribution of each party towards the final output of the computation. In this work, we propose a practical solution for secure sharing and computation of genomic data. We adopt the Paillier cryptosystem and the order preserving encryption to securely execute the count query and the ranked query. Experimental results demonstrate that the computation time is realistic enough to make our system adoptable in the real world. The results of this chapter are published in the ACM International Database Engineering Applications Symposium (IDEAS 2016) [20].

Aftermath of Bustamante Attack on Genomic Beacon Service

With the enormous need for federated eco-system for holding global genomic and clinical data, Global Alliance for Genomic and Health (GA4GH) has created an international website called beacon service which allows a researcher to find out whether a specific dataset can be utilized to his or her research beforehand [21]. This simple webservice is quite useful as it allows queries like whether a certain position of a target chromosome has a specific nucleotide. However, the increased integration of the individuals' genomic data into clinical practice and research has raised serious privacy concerns. Though the answer of such queries are yes or no in Beacon network, it results in a serious privacy implication as demonstrated in a recent work from Shringarpure and Bustamante [3]. In their attack model [3], the authors demonstrated that with a limited number of queries, presence of an individual in any dataset can be determined. The main contribution of this work is the

proposal of two simple lightweight solutions that will make the attack very difficult to achieve while maintaining the fundamental motivation of beacon database network. We also experimentally evaluated our solution on the original attack model [3] to better understand the privacy and utility tradeoffs provided by these two methods. The results of this chapter are accepted in BMC Medical Genomics [22].

Privacy Preserving Technique for Outsourcing Genomic Databases

Applications of genomic studies are spreading rapidly in many domains of science and technology such as healthcare, biomedical research, direct-to-consumer services, and forensics. However, there are a number of obstacles that make it hard to access and process a big genomic database for these applications. First, genomic sequences require a high-performance computation and a large amount of storage capacity to be processed. Second, genomic databases are often owned by different organizations and thus not available for public usage. Cloud computing paradigm can be leveraged to facilitate the creation and sharing of big genomic databases for these applications. Genomic data owners can outsource their databases in a centralized cloud server to ease the access of their databases. However, data owners are reluctant to adopt this model, as it requires outsourcing the data to an untrusted cloud service provider that may cause data breaches. In this work, we propose a privacy-preserving model for outsourcing genomic data to a cloud. The proposed model enables query processing while providing privacy protection of genomic databases. Privacy of the individuals is guaranteed by permuting and adding fake genomic records in the database. These techniques allow cloud to securely and efficiently evaluate *count* and *top-k* queries .The results of this chapter

are accepted in IEEE Journal of Biomedical and Health Informatics (JBHI) [23].

Privacy Preserving Approximation of Edit Distance on Genomic Data

Edit distance is a well established metric to quantify how dissimilar two strings are by counting the minimum number of operations required to transform one string into the other. It is utilized in the domain of human genomic sequence similarity as it captures the requirements and leads to a better diagnosis of diseases. However, in addition to the computational complexity due to the large genomic sequence length, the privacy of these sequences are highly important. As these genomic sequences are unique and can re-identify an individual, these cannot be shared in a plaintext. In this work, we propose two different approximation methods to securely compute the edit distance among genomic sequences. We use shingling, private set intersection methods, the banded alignment algorithm, and garbled circuits to implement these methods. We experimentally evaluate these methods and discuss both advantages and limitations. The results of this chapter are accepted in BMC Medical Genomics [24].

1.3 Organization

This thesis is organized as follows:

- Chapter 2 explores some of the necessary background which are utilized by the different methods proposed in this thesis.
- Chapter 3 shows how to securely execute count and ranked query on a federated architecture.

- Chapter 4 discusses a privacy attack on a genomic beacon webservice and analyzes some solutions.
- Chapter 5 discusses methods for genomic data storage and computation on the centralized architecture.
- Chapter 6 presents two different approximation techniques for edit distance on genomic data.
- Chapter 7 concludes the thesis.

Chapter 2

Privacy Preserving Techniques

In this chapter, we summarize the necessary privacy preserving techniques that can be utilized to achieve privacy-preserving genomic data sharing and computation. Different seminal contributions are also shown chronologically in Figure 2.1.

2.1 Differential Privacy

Differential privacy is a notion in data security to preserve the privacy of a query result. This was first introduced by Dwork et al. [25] in 2006 where they proposed a *privacy preserving statistical database system*. Since then, it has become a gold standard of privacy [26] as it can offer theoretical bound on privacy leakage.

Definition 2.1.1. A randomized algorithm $\mathcal{A}$ is differentially private if for all data sets $\mathcal{DB}$ and $\mathcal{DB}'$ where their symmetric difference contains at most one record (i.e., $|\mathcal{DB}\Delta\mathcal{DB}'| \leq 1$), and for all possible sanitized databases $\hat{\mathcal{DB}}$,

$$\mathcal{P}[\mathcal{A}(\mathcal{DB}) = \hat{\mathcal{DB}}] \leq e^{\epsilon} * \mathcal{P}[\mathcal{A}(\mathcal{DB}') = \hat{\mathcal{DB}}]$$

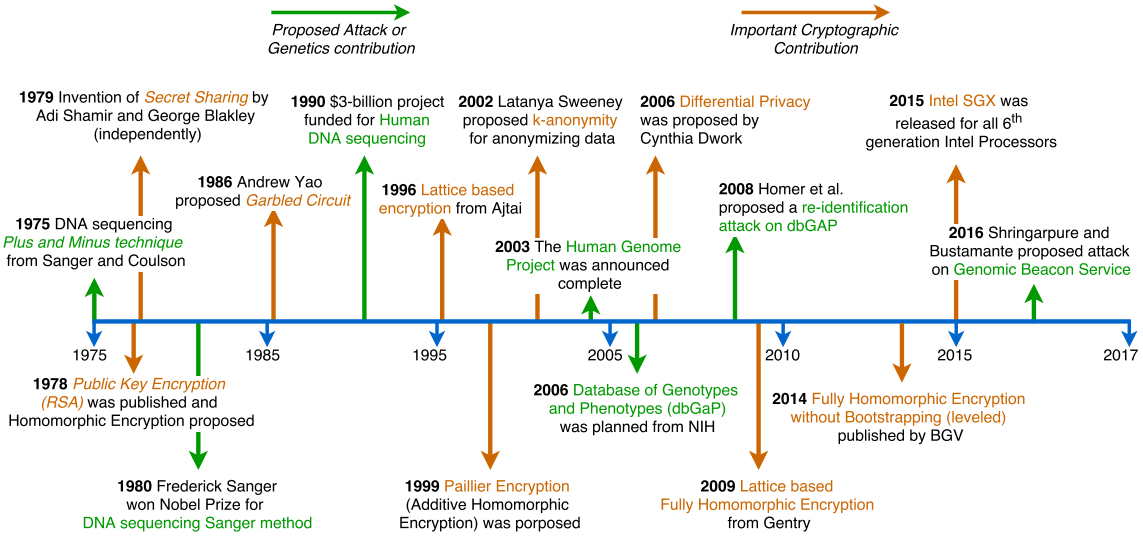


Figure 2.1: Timeline for genomic data evolution and development of multiple cryptographic techniques

Here, ϵ is that privacy budget which regulates the added noise in the output.

Example. *Consider a dataset about lung cancer patients containing their smoking behavior. This information is sensitive to the participants as leakage to this information might affect their health insurance. Nonetheless in any user study, participants have their rights to keep their data private. They only opt for sharing data for better healthcare, not to publicize their health issues. Differential privacy on such data sharing guarantees that no individual can be re-identified from the published data.*

This rigorous notion of privacy can be a potential solution for genomic data privacy as it assures that the same conclusion can be reached even if any participant opts out of the dataset. In other words, differential privacy ensures that the computation is insensitive to one particular record. Another important aspect of differential privacy is its theoretical guarantee on privacy which cannot be achievable by prior privacy preserving techniques (i.e., anonymization).

Available Implementation

One of the most popular tools available in differential privacy was proposed in 2009 by Frank D. Mcsherry [27]. The toolkit was named Privacy Integrated Queries (PINQ) based on the Microsoft's C# Language-Integrated Query (LINQ) feature. PINQ works as a layer in front of the database working on the SQL queries being performed on the data. For example, with any count queries performed on the data, PINQ will return the calibrated noisy or the differentially private output.

2.2 Homomorphic Encryption

Homomorphic encryption (HE) allows one to compute any function over encrypted data without the help of the decryption key. Though, encryption and computation might sound antithetical and much futuristic, it is one the active area of research in cryptology. The scheme was defined soon after RSA in 1978 [28] but was in theory for 30 years mostly. The scheme in a nutshell is: if $c_1 = \xi(m_1)$ and $c_2 = \xi(m_2)$ (where m_1 and m_2 are the plaintexts, c_1 and c_2 are the ciphertexts and ξ is any randomized encryption algorithm), we can compute any function on c_1 and c_2 and get the same result as if we were computing with m_1 and m_2.

Currently there are multiple HE techniques available which often can be categorized according to their functionality. A Fully HE (FHE) can compute any function under encrypted inputs, while additive or multiplicative HE schemes only support additions or multiplications, respectively. For example, *Paillier cryptosystem*, which is a popular additive homomorphic encryption scheme [29], can do linear operations on ciphertexts. With Paillier cryptosystem we can compute the

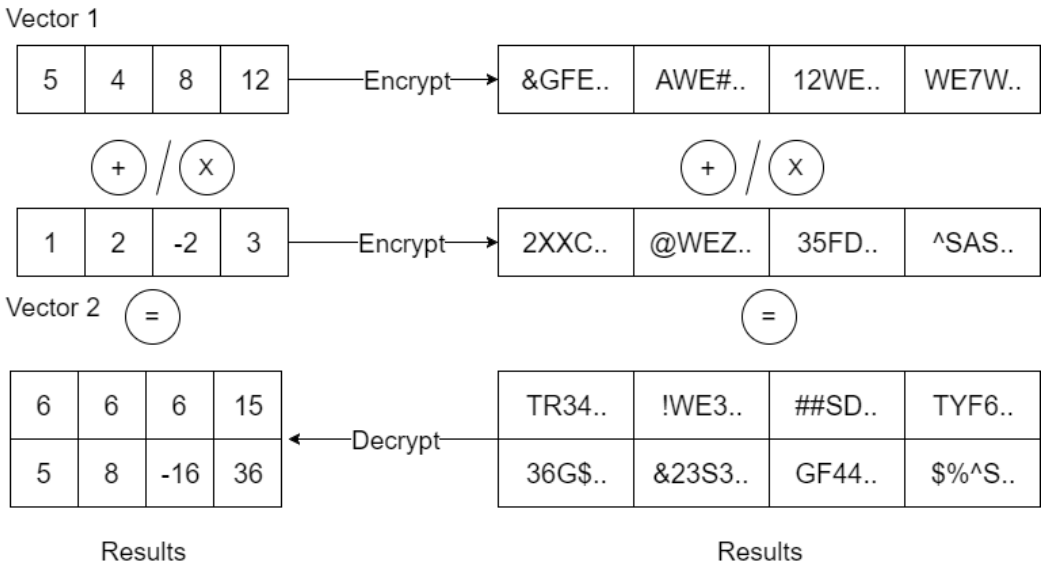


Figure 2.2: Example of fully homomorphic operations on encrypted values

following specific functions (n being the public key):

$$\xi(m_1 + m_2) = (c_1 * c_2)\, mod\, n^2$$

$$\xi(m_1 * k) = c_1^k\, mod\, n^2$$

The first equation allows us to compute the addition of two encrypted values while the second one is a constant multiplication over the ciphertext. In addition, there are some other encryption schemes such as leveled HE [30; 31] that allows computation of encrypted inputs up to a certain limit. Examples of homomorphic addition and multiplication are shown in Figure 2.2.

Available Implementation

There are two major implementations available for FHE: *a*) SEAL (`http://sealcrypto.codeplex.com`) and *b*) HElib (`https://github.com/shaih/HElib`). In Table 2.1, we point some of the differences of both implementations. There are other implemen-

tations available such as Krypto based on Multivariate Quadratic FHE (`https://github.com/kryptnostic/krypto`) and FHEW library from Leo Ducas and Daniele Micciancio [32]. However, these implementations are not popular.

Table 2.1: Comparison of two popular FHE implementations

Feature	**HElib**	**SEAL**
Crypto scheme	[33]	[34]
Language	C++	C++
External library dependency	NTL & GMP	✗
Relinearization	✓	✓
Floating point support	✗	✓
CUDA support	✓	✗
Wrapper available	Python [1]	C#

2.3 Garbled Circuit

A Garbled Circuit (GC) is a constant round protocol which allows any function to be securely computed between multiple parties. This concept was defined in 1982 by Yao [35] to solve "The Millionaire Problem". After much optimization through the years [36], many implementations are currently available like ObliVM [37], FastGC [38], and TinyGC [39].

Example. *The millionaire problem is useful to explain why GC is important in secure multiparty computation. Suppose two individuals want to know who is richer but do not want to reveal their total wealth. They can initiate a GC between them and*

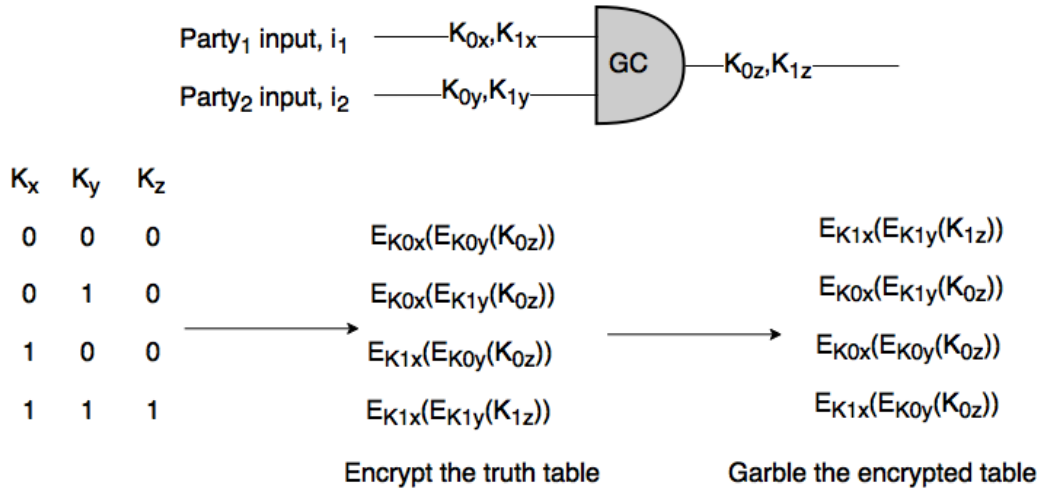


Figure 2.3: Garbled Circuit Explanation

the result will only reveal the richer party and nothing else.

In GC, one party generates a circuit for the computations and the required keys whereas the other party evaluates it. Figure 2.3 shows an example of a garbled circuit for an AND gate. The generator picks two random keys for each wire such that one key corresponds to 0 and the other corresponds to 1 resulting 6 keys in total for each gate. After that the generator encrypts each row of the truth table by encrypting the output-wire key with the corresponding pair of input-wire keys. Then it randomly garbles the table, and sends it to the other party (evaluator) along with the key corresponding to its input bit ($E_{k_{0x}}$ if the input is 0 or $E_{k_{1x}}$ if the input is 1). The evaluator evaluates the circuit by performing an oblivious transfer [40] to get the key that corresponds to its input bit and then decrypts exactly one of the output-wire keys. The evaluator sends the generator the key for the final output wire and the generator informs the evaluator if it corresponds to 0 or 1. Oblivious transfer is another cryptographic protocol where the generator puts $E_{k_{0y}}$ or $E_{k_{1y}}$ as inputs and the evaluator picks its input from it. This protocol

ensures that the evaluator does not learn the other input and the generator is unaware of evaluator's pick. The beauty of the GC protocol is that only one row of the encrypted table will be decrypted by the evaluator to a proper value (with two keys).

Available Implementation

As mentioned above there are multiple efficient implementations available for Garbled Circuits. These implementations are extensible and any computation can be done securely using these tools. Some of the notable tools regarding GC are—FastGC [38], FlexSC [37] and TinyGC [39]. To the best of our knowledge, TinyGC [39] is the recent available implementation of GC (2015).

Chapter 3

Secure and Efficient Multiparty Computation on Genomic Data

3.1 Introduction

Today genomic data is widely used by different research organizations for analysis to uncover information useful for different aspects of human life. For the purpose of future analysis, data is collected, processed, stored and then computations are done on data. Researchers are interested in executing aggregate queries over genomic data (e.g., sum and count). The query results can be used for data mining purposes to come up with new pieces of information about diseases. The accuracy of this computation is largely dependent on the amount of data available for analysis. As more data yields more accurate results, organizations tend to collect as much data as possible prior to performing the computational tasks. A single organization often does not possess the amount of genomic data adequate enough to make a certain decision. Moreover, the collection, processing and storing of

data is very time consuming. As a result disparate organizations tend to share data among each others.

Sharing genomic data possesses severe security threats. Numerous attack models have been developed to infer information from this shared data and those can be explained and classified into three major areas [6; 8; 41] —*a*) re-identification attack from relational data, *b*) inference about phenotype and kinship and *c*) legal, forensics and other attacks. Different privacy policies have been developed and a number of methodologies have been adopted to thwart these attacks.

Cryptographic techniques are adopted to ensure the security of shared data as those allow some functions to be computed on data without revealing anything about data from different parties [42]. Encryption of the data before sharing and then conducting the required computations on the encrypted data is a popular solution [43]. However, genomic data tends to be very large in size and performing computations on these large encrypted data sets is very slow and much more complex than performing computation on the plaintext.

In this chapter we propose a method which can solve the problem of sharing and conducting computations on genomic data in a secure and a time-efficient way. In our model data itself resides in the premises of data owners in plaintext format. Data owners have their own database systems and with proper authentication any researcher can execute queries on them. As there are multiple data owners involved, getting access control from all of them is difficult for the researcher who wants to perform a query on the integrated data. Our proposed system allows any researcher to execute queries on the integrated data using a simple web architecture or Application Programming Interface (API). The computation required to answer a query is performed independently on each data owner's

premise. The individual output of each party is then encrypted and sent to a central server that obliviously performs further computations to produce the final output. This output, which is sent to the researcher from the central server, is the final result of the query. As most of the computation is done on the plaintext and only individual encrypted contributions are used to produce the final result, the overall computation process is very fast than the computation done on encrypted data.

The system overview and related parties are explained in section (3.2). In the following, we summarize our major contributions in this work:

1. We securely execute count and ranked queries (Top-N or Bottom-N) on data distributed and owned by different owners. This is the first work to address ranked query problem for distributed data.

2. Each data owners does not know anything about the contributions of the other data owners. Moreover, the final result is revealed to researchers without disclosing the contribution of each data owner to the researcher.

3. We implement an Application Programming Interface (API) where researchers can execute queries with desired inputs and plug it into their programs for further analysis.

4. We conduct multiple experiments in different realistic settings. The time overhead for secure computations are much closer to their plaintext counterparts (~ 1 second).

The rest of the chapter is organized as follows. Section 3.2 describes the system architecture. The required background information is presented in Section 6.3.

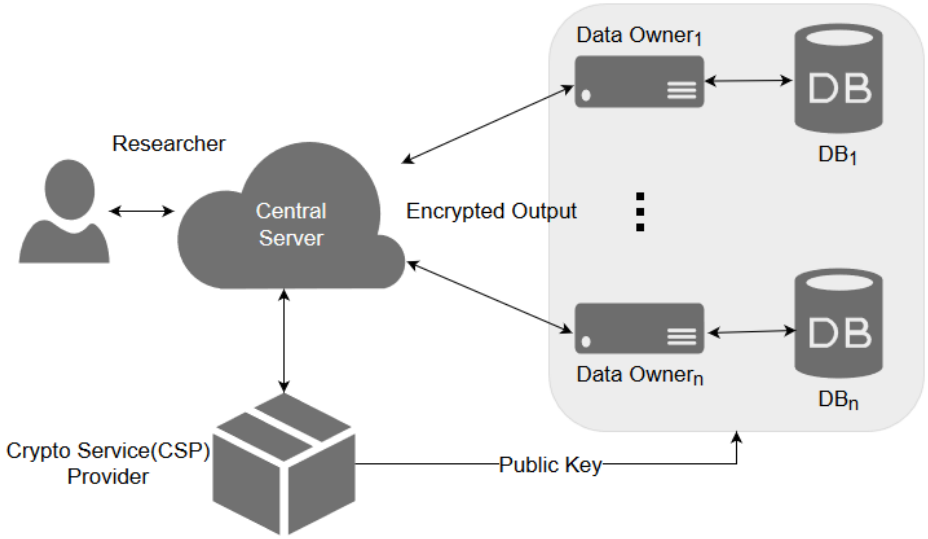


Figure 3.1: Proposed Architecture of the System

Section 3.4 details the approach. Practical results and discussions are presented in Section 6.5 while Section 6.7 discusses related work.

3.2 System Overview

In this section we detail the system architecture followed by the threat model.

3.2.1 Architecture and Entities

The proposed system as shown in Figure 3.1 has four main entities:

- *Researchers:* They might be an individual or an organization who wants to execute queries over databases. Researchers communicate their queries to the central server.
- *Data Owners:* They possess databases upon which queries are performed. Data owners might be clinics or hospitals who want to share their genomic

databases. The proposed model supports any number of data owners. When a query is forwarded to a data owner, it executes the query locally and then encrypts the local result. Afterwards, it sends the encrypted result to the central server for further computation.

- *Crypto Service Provider (CSP):* It manages the keys that are used for encryption and decryption in different stages of our system. Each data owner receives a key from the *CSP* and uses it to encrypt its local result of the query. Finally, *CSP* is responsible for decrypting the final result which is sent to the researcher.

- *Central Server:* The central server communicates with all the other entities. It receives queries from the researcher, forwards them to all data owners and collects individual encrypted results from data owners. Individual encrypted results from data owners are obliviously combined by the central server to produce the final result. The central server also decrypts the final result with the help of *CSP* before sending it to the researcher.

3.2.2 Threat Model

Our goal is to ensure that the data owners, the central server, and the crypto service provider (CSP) learn nothing more about local databases (owned by data owners) beyond what is released by the final query result. Note that we do not provide any privacy guarantee on the query result itself. Hence, it might be possible for a researcher to derive private information of an individual using only query results. Such problem can be avoided by adding noise to the query results and it has been extensively studied in the literature [25].

In this chapter, we adopt the semi-honest adversary model (also known as honest-but-curious). In semi-honest adversary model, adversaries follow the protocol but may attempt to derive additional information from the received messages. This is a common security definition and it is realistic in our problem scenario since different organizations are collaborating to securely share their data for scientific and social benefits. Thus, neither the data owners, the central server, nor the CSP has any motivation to behave maliciously in the hope of producing incorrect output.

In addition, we have the following assumptions regarding our proposed model.

- CSP does not collude with the central server or data owners to learn information about local databases.
- Data owners do not collude among themselves to violate the security of an individual data owner.

In section 3.4.2, we present a validation protocol that can be used by the CSP to validate the computation of the central server. This protocol enables the CSP to identify any incorrect query result.

3.3 Background

In this section we briefly review the concept of ranked query, homomorphic encryption and order preserving encryption.

3.3.1 Ranked Query

One of the most promising and attractive examples of the use of genomic data is the *Similar Patient Query* (SPQ) operation. The purpose of this operation is to

understand the association between the disease and genetic variation – thus increasing the accuracy of medical decisions. *Edit distance* is a proven and most frequent used metric for this SPQ operation and it is extensively adopted in literature [44]. The *edit distance* between two sequences A and B is defined as the minimum number of edits (insertion, deletion or substitution of a single character) required to change A into B. Suppose,

$$A = TTGCCTGAG \text{ and } B = ATTCAG$$

Then, the minimum edits required to convert A to B are — substitute T at position 1 with A; delete G, C and C from position 3, 4 and 5; replace G at position 7 with C. So the edit distance between A and B is 5 as the total number of required edits are 5. Through our proposed *Secure Ranked Query* (3.4.3), any researcher can get the ranked results and their corresponding distances based on the sequence given in the query. Based on the results returned by data owners, the central server ranks the query and returns the top-n number of results where n is explicitly mentioned in the query. The distance measure we consider is secured with order preserving encryption described next.

3.3.2 Order Preserving Encryption

Order Preserving Encryption (OPE) is a method which preserves the order of the plaintext in the ciphertext. Below we give an example of OPE where the orders of the plaintexts are maintained while they are encrypted.

Plaintext	1	2	2	3	4	5
OPE Ciphertext	15	24	24	30	49	52
Frequency Hiding OPE	10	24	28	32	59	72

There are different OPE schemes proposed in the literature. Some schemes do not hide the frequency of a plaintext, while others provide more security by hiding the frequency of a plaintext as shown in the example above. Although frequency hiding OPE schemes [45; 46; 47] provide more security, these schemes fall short for our application purpose (discussed in Section 3.4.3).

In this work, we adopt the OPE scheme proposed by Liu and Wang [48] to perform the secure ranked query operation. There are two constants a, b for any plaintext p_1, p_2 and the encryption of these plaintexts are:

$$\xi(p_1) = a * p_1 + b + noise_1 \tag{3.1}$$

$$\xi(p_2) = a * p_2 + b + noise_2 \tag{3.2}$$

$$where\ 0 \leq noise_1, noise_2 < a$$

Though a and b are constants, due to the addition of the random noise, this scheme provides enough security for our scheme. The security discussions are detailed in Section 3.4.5.

3.4 Secure Computation over Genomic Data

Say there are total n number of hospitals ($H_1, H_2, ..., H_n$) in our system which represent data owners. For a single query provided by a researcher, each hospital will have a single output — H_1 will have output x_1, H_2 will have output x_2 and similarly H_n will have output x_n. In total, there will be n outputs. CSP generates the required public and private keys for the encryption. Data owners get the public keys from CSP. The data owners' outputs ($x_1, x_2, \ldots x_n$) the encrypted values using the public keys provided by the CSP and send to the central server for merging. The central server then runs the desired addition with the Paillier cryptosystem. The

Table 3.1: Data representation in each party

Data Owner		**Sequence**				
#	*Case*	*rs4426491*	*rs4305230*	*rs4630725*		**Cancer**
Data Owner 1	1	CC	CT	GG	...	Negative
	2	CT	CT	AG	...	Negative
	3	CC	CT	GG	...	Negative
Data Owner 2	1	CC	CT	GG	...	Negative
	2	CT	CC	GG	...	Positive
	3	CC	CT	GG	...	Positive
Data Owner 3	1	CT	CC	AG	...	Positive
	2	CT	CT	AG	...	Negative
	3	TT	CC	GG	...	Positive
Data Owner 4	1	TT	CC	AA	...	Positive
	2	CC	CC	GG	...	Positive
	3	CC	CT	GG	...	Positive

sequence diagram of the system is shown in Figure 3.2. The sequence diagram is detailed in the Basic Protocol (3.4.1) which is essential for all the computations. The validation Protocol 2 (3.4.2) is about verifying the results. Ranked query execution is described in 3.4.3.

3.4.1 Basic Protocol

This protocol defines the main mechanism of the framework which ensures the individual contributions of the data owners are secure. For instance, the re-

Protocol 1: Query execution protocol

Data: A query from the researcher

Result: The query result

1 The central server accepts a query from a researcher and sends it to CSP for initialization;

2 *CSP* generates a *secret key* and a *public key* pair and sends the *public key* to the active data owners and the *Central Server* for further computation;

3 After receiving the public key the server sends the query to the individual data owners who will participate in this query;

4 The data owners execute the query on their data and send the encrypted results to the central server;

5 The central server adds the values from the individual parties and sends the encrypted result to CSP for validation and decryption;

6 CSP decrypts the submitted value (as it is the only entity holding the secret key) by the central server and validates it (Section 3.4.2);

7 This result is submitted to the researcher as a final output;

searcher wants to know how many patients have *CC* in their SNP 'rs4426491', *GG* in their SNP 'rs4630725' and cancer diagnosis *Positive* in Table 3.1 (i.e., integrated database from 4 data owners).

If CSP, the central server and data owners follow the protocol mentioned in 1, the individual result for this count operation at data owner 1 will be 0, at data owner 2 will be 1, at data owner 3 will be 0 and at data owner 4 will be 2. Each data owner will encrypt its result using the public key provided by CSP and send the

encrypted result to the central server. The central server will aggregate the local results to produce the final encrypted result $\xi(3)$. After decrypting the encrypted result with the help of CSP, the central server will forward the final result to the researcher.

3.4.2 Validation Protocol

This protocol validates the computations made in the central server. Two big prime numbers (assuming they are bigger than the results) are sent along with the keys by CSP. One of these numbers (x_1) is multiplied by the individual result and the other one (x_2) is added. For example if there are 2 ($n = 2$) parties then CSP will send $\{x_1, x_2\}$, $\{x_1, x_3\}$ to those individual parties. The individual parties now have z_1 and z_2 as outputs (inputs y_1, y_2) of the query which are modified using those primes.

$$c_1 = z_1' = \xi(y_1 * x_1 + x_2) \tag{3.3}$$

$$c_2 = z_2' = \xi(y_2 * x_1 + x_3) \tag{3.4}$$

The new outputs that the parties generate and send to the server is shown in Equation 3.3 and Equation 3.4. After decrypting, CSP subtracts the values x_2 and x_3, divides the result with $n * x_1$ to get the original result and sends it back to the server. If the result is not divisible with $n * p_1$ then there might be a miscalculation on the server. This method ensures that the final output is valid which is important as many data owners are involved in a single query over the network.

3.4.3 Ranked Query

Suppose, we have a list of unique genomic sequences of patients from different hospitals or data owners (Table 3.1). The researcher wants to execute a *Similar Patient Query* [44] operation on every dataset based on the value of *Edit Distance* with respect to a given query sequence. He might want the Top-5 sequences for a particular associated disease. The query might be like the following —

```
SELECT * FROM Sequences
  WHERE rs4426491='CC' AND rs4305230='CT' AND rs4630725='GG'
  LIMIT 5
```

Each data owner has a table named 'Sequences'. Genomic sequences from all of the data owners are shown together in Table 3.1 for a better understanding. Each data owner calculates the individual distances (edit distance) of all the genomic data rows with the query sequence. Then it sends only k encrypted results using OPE to the central server. The number k is specified in the query by the LIMIT command.

This encryption, which is performed at each of the data owner's end, is done by using the OPE described in Section 3.3.2. Consider two data owners with different distance sets $\{10, 11, 13, 14, 15\}$ and $\{11, 12, 13, 14, 15\}$. We can add a random noise while encrypting them and it will hide the frequency of the original distance. But the original order of the distances will be perturbed as most of the distance values are relatively close to each other. Hence, wrong order of the ranked data will cause an erroneous output.

We followed Liu and Wang's OPE scheme [48] (preceded by [49]) where *noises are equal to the individual distance measure*. This protocol allows the researcher to get his desired Top-k results for the given query. The server is able to know only the relative distances. The keys (Equation 3.1, 3.2) provided by CSP are changed

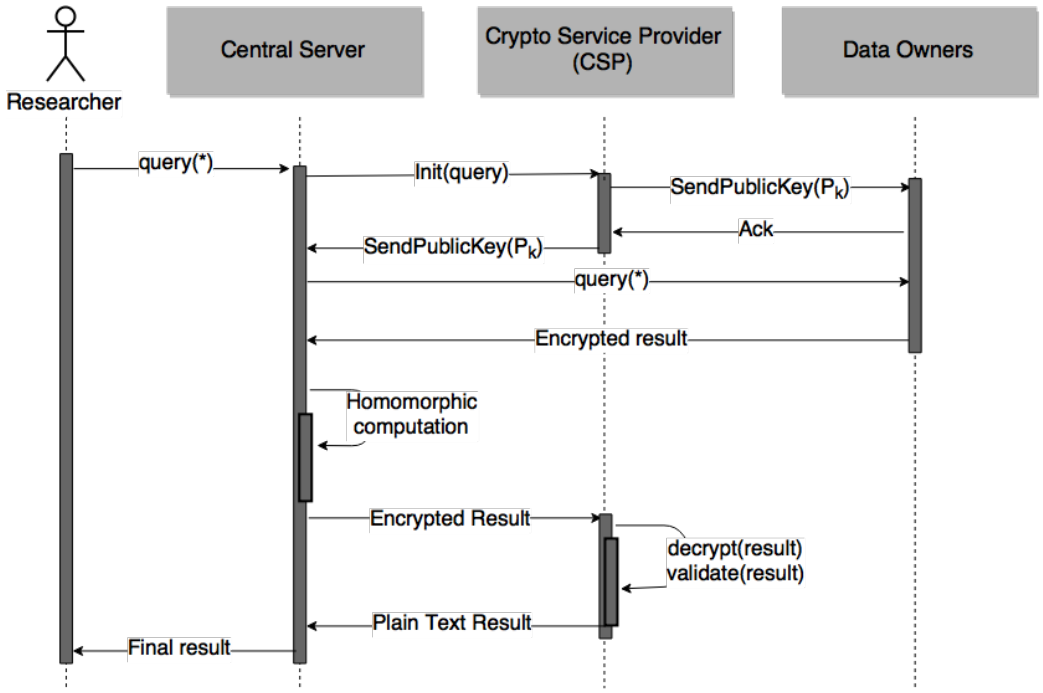


Figure 3.2: Sequence diagram of the approach

for every new query. This ensures stronger security as every edit distance query will have different order range according to the keys. The security of this scheme is discussed in detail in 3.4.5.

3.4.4 Offline CSP

If we want to make the CSP less active on the computation and become offline after the key generation and propagation, the researcher should be responsible for the decryption. In this context, CSP becomes inactive after initializing the system and gives the secret key to the researcher. With secret key, the researcher can retrieve the final result by decrypting (previously done by CSP). The only difference from the previous protocols is the key generation is done on CSP and afterwards the researcher does the rest.

3.4.5 Security Discussion

For the count query we use Paillier Encryption [50] which is an established cryptosystem. This cryptosystem has been extensively used in the literature as a probabilistic additive homomorphic encryption scheme and it provides IND-CPA (indistinguishability under chosen plain text) security guarantee [51; 52].

For the rank query, we use an OPE scheme that does not satisfy formal definition of IND-OCPA (indistinguishability under ordered chosen-plaintext attack) security[53]. Recent research demonstrates that it is possible to determine the constants a and b of the encryption scheme (see Equations 3.1 and 3.2) [47; 54]. For example, if an adversary takes $p_1 =$ any integer and $p_2 = 0$ then the adversary can compute $c_1 - c_2 = a * p_1 + noise_1 - noise_2$. As $0 \leq noise_1, noise_2 < a$, the adversary can learn in which interval a belongs (i.e., $\frac{c_1-c_2}{p_1+1} \leq a \leq \frac{c_1-c_2}{p_1-1}$). Given a, the adversary can then obtain b in some iterations and can eventually decrypt all the values.

However, this attack is not applicable for our proposed scenario as the adversary (i.e., central server) does not provide any inputs (p_1 and p_2) of the OPE scheme and therefore unable to determine the constants a and b. Also each query result will have a different a and b from the CSP along with the noise which is the original distance. The central server only compares different order preserved encrypted values coming from different data owners and publishes the final ranked result. Hence, it is secure to employ the OPE scheme of Liu and Wang [48] for our application scenario.

Table 3.2: Server locations and average latency

Server Location	**IP Address**	**Latency(ms)**
Singapore	52.77.210.123	244
Brazil	54.94.175.199	186
Ireland	52.48.14.85	110
USA (Oregon)	52.32.83.223	37
Canada (Manitoba)	130.179.30.133	1

3.5 Results and Discussions

We considered data owners and researchers on different locations in our experiments so that we can measure the performance of the system in a real world scenario. We fixed the position of CSP and the central server in two different locations and used *Amazon EC2 cloud servers* with the same configuration for all the data owners and CSP. The full system is connected to a cloud server in Canada (Manitoba) which is also the endpoint for the API for researchers. For genomic dataset we used *IDASH 2015 SNP dataset* [55] which were distributed among the data owners. Additionally, for better measurement of Ranked Query, we used a larger dataset 'Bag of Words' [56] ($96,618$ records) distributed between the data owners so that every party owns a different set of words. The code is shared on github (`https://github.com/mominbuet/smc_genome_clients.git`).

We used different combinations of active data owners which are described in Table 3.3. For example *Experiment 1* shows that we have data owners in Canada and Ireland where *Experiment 5* has 5 different data owners on 5 different locations in

Table 3.3: Experimental setup

Exp. #	**Actors**	**Canada**	**USA**	**Ireland**	**Singapore**	**Brazil**
Exp. 1	Hospital	✓	✗	✓	✗	✗
	Researcher	✗	✓	✗	✗	✗
	CSP	✗	✓	✗	✗	✗
Exp. 2	Hospital	✗	✗	✓	✓	✓
	Researcher	✓	✗	✗	✗	✗
	CSP	✗	✓	✗	✗	✗
Exp. 3	Hospital	✓	✗	✓	✓	✗
	Researcher	✓	✗	✗	✗	✗
	CSP	✗	✓	✗	✗	✗
Exp. 4	Hospital	✓	✗	✓	✓	✓
	Researcher	✓	✗	✗	✗	✗
	CSP	✗	✓	✗	✗	✗
Exp. 5	Hospital	✓	✓	✓	✓	✓
	Researcher	✓	✗	✗	✗	✗
	CSP	✗	✓	✗	✗	✗

the world. Table 3.2 shows the 5 server locations and their latency with respect to the central server (which hosts the API) in Manitoba.

Secure count query, ranked query and their plaintext counterparts were executed under these settings. Fig. 3.3 shows the runtime of 10 iterations of these individual protocols. The difference of time for secure and regular query operations are

	Experiment 1	Experiment 2	Experiment 3	Experiment 4	Experiment 5
Secure Count	5148	4778	4820	4862	4799
Regular Count	4962	4291	4411	4371	4282
Secure Ranked Query	8582	10795	10822	10334	13300
Regular Ranked Query	8038	10297	10300	9853	11324

Figure 3.3: Experiment results for different settings in milliseconds (ms)

negligible. Our proposed framework can not only perform the query operations in a secured way but also the time required to execute those secure query operations are almost as the same as performing those operations on plaintext. Figure 3.3 also shows that adding more data owners affects all the protocols (regular and secure) which is completely logical as it will endorse more network communications. But interestingly increasing data owners in different regions does not effect the secure protocols that much as the network communication were almost the same as the plaintext ones. The only overhead the secure method had is decrypting the final result by CSP. This way we achieved faster, secure protocols of the query build like an API which is pluggable to any program.

3.6 Related Work

We are only securing the computations in the cloud but not the final output of the query. To be specific the individual contribution provided by each owner is hidden from the central server but the computations are still done with the encrypted values. There were previous attempts to address a similar problem but no solutions or attempts are available for Secure Multiparty Ranked Query. There are some approaches which relates to our problem architecture but they either rely on homomorphic encryption or Yao's garbled circuit [35].

There is a recent work *SecureMA [57]*, which has similar architecture but proposed a secure computation for a single function. The proposed method considered the secure count but it introduced some approximation over the final result to maintain the security of the protocol. This protocol extensively uses garbled circuit which is the major reason behind its security but also increases bandwidth cost (because of circuits) and runtime along the way.

There are also many use of Fully Homomorphic Encryption (FHE) or Some-What Homomorphic Encryption (SWHE) in genomic data [58; 59]. One example would be *Healer [60]* which is architecturally similar to ours and provides *Secure Logistic Regression* in genomic data. Foresee [61] is also similar and proposes secure chi-square statistics on genomic data. These approaches are computationally secure but the overhead involved makes it impractical to use in real world application scenario. Also these methods did not address Secure Ranked Query problem.

3.7 Future Work

As a future work, we are thinking to extend the approach such that it can securely and privately execute any required statistical queries [62]. Privacy of the output can also be ensured by differentially private noise which can added to the final result to protect the identity of the patients. Additionally, we may define another approach to directly compute the ranked query on encrypted data.

Chapter 4

Aftermath of Bustamante Attack on Genomic Beacon Service

4.1 Introduction

Recent improvements on genomic data sharing efforts have led researchers and clinicians gaining access and make comparisons across data from millions of individuals. Such development made it easier for genetic variant interpretation and in some cases treatment of rare diseases such as some special cancer types [21]. Majority of the big organisations i.e., Broad institute in the U.S., BGI in china, Wellcome Trust Sanger in the UK etc. have an interest of making DNA data easier to access in order for their researchers to treat patients one on one. However, after twelve years of completion of the Human Genome project, the tremendous growth of genomic data has exceeded the containers built to hold such data. Genomic and clinical data are generally still collected in either by disease, institution or by country. More importantly, current data sharing privacy requirements do not nec-

essarily protect individuals' identity within and across institutions and countries. Furthermore, data is often stored in incompatible file formats and there are no standardized tools and analytical methods are in place [21; 63; 20].

With such tremendous needs for global genomic and clinical data repository systems, Global Alliance for Genomic and Health (GA4GH) has created a federated data ecosystems called Beacon data network, a way for searching genomic data as simple as World Wide Web. Since the project's launch in the middle of 2015, the beacon network has currently *23 different organizations* covering over *250 genomic datasets*. The data sets served through beacons can be queried individually or in aggregate via the Beacon Network, a federated search engine (`www.beacon-network.org`) [21]. Thus, the Beacon Project aims to simplify data sharing through a web service ('beacon') that provides only allele-presence information. Users can query institutional beacons for information about genomic data available at the institution. For example, an individual could ask the beacon web server about a genome that has a specific nucleotide and the beacon would response either 'yes' or 'no' [64]. By providing only allele-presence information, beacons were assumed safe from attacks that require allele frequencies.

Although the beacon network has set up to share data and protect patient privacy simultaneously, it could potentially leak phenotypic and membership information of an individual [64]. There is currently no cap on the number of queries a user can make in the Beacon database publicly. Recently, Shringarpure and Bustamante showed that anonymous patients whose DNA data is shared via beacon network can be re-identified [3]. If an attacker has access to victim's DNA, s/he can query different beacons to see whether the victim is in the dataset. They further demonstrated that it is possible to infer whether or not the victim is af-

fected by a certain condition or disease [3]. Therefore, the anonymous beacons are inherently insecure and are open to re-identification attacks.

Very recently, some solutions [65; 66] have been proposed based on different policies around the access of the beacon service. However, these solutions will disrupt the quintessential feature of the proposed beacon service: that is to provide faster access to genomic data and to give open access to the research community. Different access controls are highly necessary for human genomic data access where phenotypic or sensitive information about the disease is disclosed. However, the beacon service only provides us aggregate results of yes or no leading the researcher to a decision regarding the dataset's relatedness to his or her research. Therefore, we propose two solutions based on privacy-preserving techniques, which fit well with the beacon service and mitigate the possibility of identifying an individual from the dataset.

In this article, we explain the 'Bustamante Attack' [3] on genomic beacon services and propose two privacy preserving solutions. The contributions of this article can be summarized in two folds: *a*) understanding the statistical formulations and soundness of the attack, *b*) analyze lightweight privacy preserving solutions to mitigate the attack.

The main contributions of our work are as follows:

- We present the statistical and the mathematical model of the attack in a simplified form. This helps us to analyze different and more realistic parameters on the original attack framework to exploit some weakness and justify our solutions accordingly.

- We show the required steps for any data owner to calculate the risk involved

in sharing their genomic data in a beacon service.

- We propose two easy to implement and lightweight privacy preserving solutions which ensure the applicability of the beacon service as well as the privacy of the participants.
- We provide extensive experiments over synthetic data (according to [3]) to show the privacy-utility evaluation of our proposed methods which will help the development of different privacy preserving techniques on such attack model later on.

To the best of our knowledge, this is the first literature where the attack is explained broadly and some privacy preserving solutions are evaluated. Our implementations are readily available on github [67] which was pipelined in the original attack simulation.

The remaining of the sections are organised as follows. We start with the brief description of beacon service in Section 4.2. This is followed by Bustamante attack in Section 4.3. Then, we further explain our proposed privacy-preserving lightweight solutions in Section 4.4. The evaluation of our proposed methods is presented in Section 4.5. Finally the discussion and related work has been made in Section 4.6 and 4.7 respectively.

4.2 Beacon service for genomic data

A beacon is an online web search engine developed by the Global Alliance for Genomic and Health (GA4GH), which provides a way for genomic data owners and research institutes to easily share genomic data while maintaining patients privacy. It is a genetic mutation sharing platform that allows any user to query an

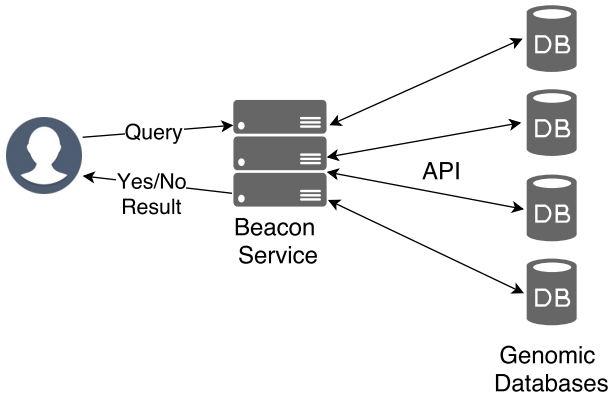


Figure 4.1: Beacon architecture where researcher and data owners are connected to the central beacon service

institution's databases to determine whether these databases contain a genetic variant of interest while keeping all other sequence data obscured. A query in this search engine is defined by three parameters: chromosome number, position in that chromosome, and target nucleotide (A/T/G/C). A beacon query answer is either true or false, denoting the presence of that nucleotide in that specific position and target chromosome. In other words, it will only answer yes/no for the questions like: *Do you have any genomes with an 'A/T/G/C' at some position 'Y', on specific chromosome 'Z'*. This allows a researcher to target some specific dataset, which is relevant to his or her research. This service also helps a clinician to check whether a mutation found in one of her patients is also present in others without actually having access to their genomes [68].

Beacons are easy-to-implement techniques for several large-scale organisations when it comes to sharing genomic data. It also saves researchers a tremendous amount of time for tracking down useful data for their work as well [69; 70]. Unlike large centralized data repositories, a beacon network is distributed across many databases around the world and is virtually connected through software

interfaces allowing continuous authorised access. This federated data ecosystem allows each organisation to control their legal data within their jurisdiction [21]. The shared Genomics API in the beacon framework makes it easy to query all at once and ensures that GA4GH team can quickly add new beacons to the network.

4.3 Bustamante attack on beacon service

A recent study done by Shringarpure and Bustamante [3], developed a likelihood-ratio test that uses only allele presence information to predict if the genome of an individual is present or not in the beacon database.This study suggested that beacons are susceptible to re-identification attacks and thus can be subjugated to invade genetic privacy. Since a beacon database includes data with known phenotypes information such as cancer, autism or other diseases, this re-identification also potentially disclose phenotypic information about an individual whose genomic data is present in the beacon [71]. Through simulations, they demonstrated that by making just *5,000 queries*, it was possible to identify someone and *even their relatives* in a beacon consisting *1,000 individuals*. They found that re-identification of an individual is possible even with the sequencing errors and variant-calling differences. They also demonstrated that a beacon constructed with 65 European individuals from the 1000 genome projects, it is possible to detect membership in the beacon with just 250 SNPs [3].

In this section, we briefly introduce the Bustamante attack and analyze its statistical methods. The goal of this attack is to know whether a genomic sequence g belongs to a specific database with the help of the beacon service. To answer this question they considered two hypotheses:

1. Null hypothesis H_0: the query individual is not in the beacon service
2. Alternative hypothesis H_1: the query individual is in the beacon service.

To determine the correct one, the adversary is allowed to query the beacon service with unlimited amount of queries. The adversary queries specific locations where the query individual has alternative allele to see whether the beacon server also contains an individual with the same allele values. Therefore, the responses of the beacon service are a sequence $x_1, \ldots, x_n$ of yes or no. If we represent yes and no by '1' and '0' respectively, the the answer sequence, R will be a binary vector. For example, if the query individual is in the database, we will get yes (or 1) in each query. However if there are some genome sequencing errors, we might get some wrong answers as well. This error is denoted by δ and also considered by the attack [3].

There is also another considerable case where multiple individuals have the same allele in the database. This is why the attacker needs to leverage the likelihood ratio of both the assumptions whether the the user is in the dataset or not. For a database of N genomes, the log of this likelihood ratio can be computed for the response series R regarding the hypotheses H_i as follows:

$$L_{H_i}(R) = \sum_{i=1}^{n} x_i \log P(x_i = 1|H_i) + (1 - x_i) \log P(x_i = 0|H_i)$$

where, n is the number of queries and x_i is the result from the beacon. $x_i = 1$ denotes the query is present in the database which can come either from the target genome or any of the other $N - 1$ genomes. x_i is only 0 when the query is not present in any of the N genomes.

In article [3], the authors using some simplifying assumptions proved that if the query individual is in the beacon database, $R = x_1, \ldots, x_n$ follows a $Binomial(n, 1-$

D_N) distribution, otherwise R has a $Binomial(n, 1 - \delta D_{N-1})$ distribution where δ is the sequencing error considered. Therefore, the hypothesis can be rewritten as follows:

1. Null hypothesis H_0: $\theta = \theta_0 = n(1 - D_N)$
2. Alternative hypothesis H_1: $\theta = \theta_1 = n(1 - \delta D_{N-1})$

Therefore, we have:

$$L_{H_0}(R) = \sum_{i=1}^{n} x_i \log(1 - D_N) + (1 - x_i) \log(D_N) \tag{4.1}$$

and for alternative hypothesis,

$$L_{H_1}(R) = \sum_{i=1}^{n} x_i \log(1 - \delta D_{N-1}) + (1 - x_i) \log(\delta D_{N-1}) \tag{4.2}$$

where D_{N-1} is the probability that other $N - 1$ individuals (all individual except the query individual) have not the specified allele in the determined location.

Basically, the $L_{H_i}(R)$ will maximize if the H_i hypothesis is correct. Therefore, we compute $\Lambda = L_{H_0}(R) - L_{H_1}(R)$ and the Λ will declare which hypothesis is true.

The log of the likelihood-ratio statistics can be rewritten from equation 4.1, 4.2 as,

$$\begin{aligned}
\Lambda &= L_{H_0}(R) - L_{H_1}(R) \\
&= n \log(\frac{D_N}{\delta D_{N-1}}) + \log(\frac{\delta D_{N-1}(1 - D_N)}{D_N(1 - \delta D_{N-1})}) \sum_{i=1}^{n} x_i \\
&= nB + C \sum_{i=1}^{n} x_i
\end{aligned} \tag{4.3}$$

In any distribution, a threshold t can be fixed where the null hypothesis will be rejected if $\Lambda < t$ and accepted otherwise. The attacker need to decide an appropriate threshold for a specific beacon dataset before launching the attack. Suppose

a false positive error α is given. Regarding this value and the beacon statistical properties, the threshold t_α is determined such that $Pr(\Lambda < t_\alpha | H_0) = \alpha$. From equation 4.3,

$$\begin{gathered} Pr(nB + C\sum_{i=1}^{n} x_i < t_\alpha | H_0) < \alpha \\ Pr(\sum_{i=1}^{n} x_i > \frac{t_\alpha - nB}{C} | H_0) < \alpha \ (C \ is \ negative) \\ Pr(\sum_{i=1}^{n} x_i > t'_\alpha | H_0) < \alpha \end{gathered} \tag{4.4}$$

In the attack instead of calculating Λ and comparing it to the threshold t_α, $\sum_{i=1}^{n} x_i$ is computed and compared with t'_α to make the decision. This threshold t'_α is used to decide whether the null or the alternative hypothesis is correct. In other words whether the individual is present in the beacon database or not will be dictated by this t'_α. To calculate this, the adversary sums the responses from the beacon x_i and retrieves $\sum x_i$. The null hypothesis is rejected simply if $\sum x_i > t'_\alpha$ which leads to a conclusion that the query individual is present in the beacon and the attack is successful.

To calculate the D_N, the authors assumed that the adversary has an idea about the distribution of the allele frequencies on those query positions. Specifically, alternate allele frequencies, f for all SNPs observed in the population are claimed to be distributed as a β distribution according to [3]. Here, $f \sim \beta(a', b')$, where $a = a' + 1$ and $b = b' + 1$, and (a', b') can be precomputed from the genomic dataset in which the beacon service is running. Thus, the adversary needs $n \sim N^{a'+1}$ queries to make his or her decision whether the target individual is present in the database. The value D_N can be approximated as,

$$D_N \approx \frac{\Gamma(a+b)}{\Gamma(b)(2N+a+b)^a} \tag{4.5}$$

Table 4.1: Evaluating the threshold $t'_{0.05}$ for three different databases with $\delta = 10^{-3}$ mismatch error.

Beacon Database	# of Records, N	a'	b'	D_N	t'_α
1k Genomes Phase1	1092	0.0735	1.0096	5.59497e-04	1826
1k Genomes Phase1 Affymetrix	1074	0.6483	1.2876	1.53524e-05	99084
GoNL [72]	498	0.1131	0.8574	0.000941	1005
SSMP [73]	100	0.1848	0.8500	0.00403	234
Simulation	2000	0.1178793	1.1188360	0.0002237	4900

To see the details of deriving and proving the above formula see [3]. We will need this t'_α and D_N for further analysis in the upcoming section as these parameters dictate the attack.

4.4 Privacy preserving solutions

In this section, we provide an analysis of equations 4.4 and 4.5 to calculate t'_α before describing our privacy preserving solutions and experimental results. This analysis allows the beacon service providers and data owners to calculate the risk involved while sharing their beacon data.

4.4.1 Risk analysis of a beacon service

In this section, we evaluated t'_α in greater depth for analyzing the risk involved for a specific beacon dataset. Given N samples and n number of queries, this analysis will help us to determine the number of correct answers that can be returned without identifying the victim.

In other words, as t'_α directly effects the decision boundary of the correctness of null or alternative hypothesis, its better to theoretically ratify its value on a specific setting. We simulated this on some real life human genomic databases like 1000 Genomes Project, SSMP [73] and GoNL [72] for better understanding.

According to central limit theorem the value $R_e = \frac{1}{n}\sum_{i=1}^{n} x_i$ follows normal distribution $\mathcal{N}(1 - D_N,\ \frac{D_N(1-D_N)}{n})$. The threshold t'_α can then be calculated from equation 4.4 as follows:

$$
\begin{aligned}
& Pr(\textstyle\sum_{i=1}^{n} x_i > t'_\alpha | H_0) = \alpha \\
\Rightarrow\ & Pr(\frac{1}{n}\textstyle\sum_{i=1}^{n} x_i > \frac{t'_\alpha}{n} | H_0) = \alpha \\
\Rightarrow\ & Pr(R_e > t''_\alpha | H_0) = \alpha
\end{aligned}
$$

where $t''_\alpha = \frac{t'_\alpha}{n}$. We know that R_e follows the $\mathcal{N}(1 - D_N, \frac{D_N(1-D_N)}{n})$ distribution where $\theta_0 = 1 - D_N$ and variance $\sigma_0^2 = \frac{D_N(1-D_N)}{n}$. Therefore, $\frac{Re-\theta_0}{\sigma_0}$ has standard normal distribution, $\mathcal{N}(0, 1)$. Suppose we want to have a $\alpha = 0.05$ false positive probability then,

$$
Pr(\frac{Re - \theta_0}{\sigma_0} > \frac{t''_\alpha - \theta_0}{\sigma_0}) = 0.05 \tag{4.6}
$$

According to the normal cumulative table and given the fact that $\frac{Re-\theta_0}{\sigma_0}$ follows

standard normal distribution, we have,

$$\frac{t''_\alpha - \theta_0}{\sigma_0} = 1.65 \Rightarrow t''_\alpha = 1.65\sigma_0 + \theta_0$$
$$\Rightarrow t'_\alpha = n(1.65\sigma_0 + \theta_0) = n(1.65\frac{\sqrt{D_N(1-D_N)}}{\sqrt{n}} + \theta_0)$$

Table 4.1 shows the computation of the t'_α for $\alpha = 0.05$ in different beacon databases constructed from real life genomic datasets according to [3].

The threshold t'_α indicates the number of *yes* that an adversary requires to conclude that the query individual is within the dataset. For example in table 4.1, the experimental results show that for '1k Genomes Phase 1' dataset with 1092 individuals, the adversary needs 1826 *yes* answers of queries to infer that the victim is present in the dataset. Any quantity less than 1826 yes answers (with mismatch rate $\delta = 10^{-3}$) will conclude that the individual is not present.

The relationship between null and alternative hypothesis along with the threshold are showed in Figure 4.2. In Figure 4.2, the black and the green lines represent the outputs distributions. If the null hypothesis is true, then the outputs follows black line and the outputs follows the green line if the alternative hypothesis holds. Three other real world datasets were tested and depicted in the Appendix (Figure A.1).

However, regardless of the risk analysis of a specific dataset with the outlined equations, the beacon service still needs privacy preserving mechanisms. The necessity of such methods are amplified due to the fact that these beacons are designed to support thousands of public queries. Though the simplified equations above can enlighten a data owner about the sensitivity of the underlying data, the data owner still needs some methods, which we will present in the next section, to protect the privacy of the individuals.

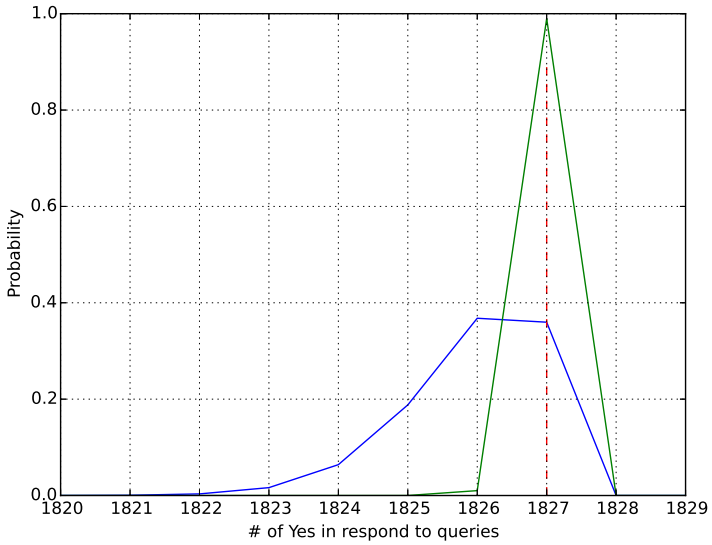


Figure 4.2: Analysis of the original attack algorithm on real world dataset where the green line represents the responses of the beacon service when the query individual is in the in beacon database while the black one represents them not being there. The red line denotes the t'_α.

4.4.2 Proposed methods

In this section, we propose two privacy preserving methods which are similar in nature. However the probability of different outputs from these two methods are different. The methods are:

1. Method 1: Eliminating random positions
2. Method 2: Biased randomized response

Both of these methods introduce inaccurate results to hide the presence of any individual in a beacon web service. Due to this inaccuracy, these methods decrease some utility of the beacon service as the underlying data will be perturbed.

Algorithm 2: Eliminating Random Positions

Data: $query$, $dataset$, $bias\, b$

Result: Yes/No

```
1 random_value ← RandomValue(0,1);
2 true_result ← CheckPresence(dataset,query);
3 if random_value ≤ b/100 then
4 |   return true_result;
5 else
6 |   return ! true_result;                    // wrong answer
```

Hence, we need to devise methods that give incorrect answers (false positives or negatives) as less as possible. A false positive is answering *yes* when the query result is *no*. Similarly, a false negative is answering *no* when the query result is *yes*. In Section 4.5, we experimentally evaluate these two methods in terms of data privacy and utility.

Eliminating random positions

In this method, the data owners apply algorithm 2 to output their data to the beacon service provider. The output of algorithm 2 will infuse inaccuracies in the result from the original data with the help of a driving factor *bias*. For example, if data owner has a dataset $\mathcal{D}$ then this method will transform it to a $\mathcal{D}'$ where there will be some false positives and negatives with respect to bias, b.

In algorithm 2, for higher bias value, we will get higher accuracy and for lower bias value, we will get lower accuracy. For example, if the bias value is 50, we will obtain answers with a probability of 50% false positives and negatives. We further

analyzed different bias values for this algorithm in Section 4.5.

Biased randomized response

Randomized response [74] was proposed in 1965 by Warner as a statistical tool to remove potential bias and add a probabilistic noise to the answers. For example, data owners transform $\mathcal{D} \rightarrow \mathcal{D}'$ with respect to certain probability. In the original method, the person who has been asked a private question flips a coin. If it is tail then s/he answers truthfully. Otherwise, for head, s/he flips the coin again and responds truthfully for tail and provides opposite answer for head.

In a beacon service, we incorporated this method as beacon queries are considered private and their answers are in binary (yes or no) form. For example, a typical query inquires about the presence of a major and minor allele in a specific position of a chromosome. Algorithm 3 can transform the raw data according to the randomized response method and this transformed data can be used further to answer queries.

However, answering queries in this fashion will induce some error and the utility of the beacon services will be questioned. Thus, we experimented on a biased randomized response where this $1/2$ probability is modified for better utility or true results in algorithm 3.

In Algorithm 3, we changed the dichotomous behaviour of general randomized response with a control variable named bias. Similar to Algorithm 2, a higher bias will give more accurate results and will provide less privacy on the data. We showed the analysis for different bias values in Section 4.5.

However, there is a similarity between both the algorithms. Algorithm 2 returns true answer with probability b given $b \in [0, 1]$, while Algorithm 3 returns true

Algorithm 3: Biased Randomized Response

Data: $query$, $dataset$, $bias\, b$

Result: Yes/No

```
1  random_value ← RandomValue(0,1);
2  true_result ← CheckPresence(dataset,query);
3  if random_value ≤ b/100 then
4      return true_result;
5  else
6      random_value ← RandomValue(0,1);
7      if random_value ≤ b then
8          return true_result;
9      else
10         return ! true_result;                    // wrong answer
```

answer with probability $1-(1-b)^2$. Therefore, Algorithm 3 with bias b_2 will be same as Algorithm 2 having bias $b_1 = 2b_2 - b_2^2$ where $b_1, b_2 \in [0,1]$.

Theorem 1. *1 The response from Algorithm 3 is* $|ln(\frac{1}{(1-b)^2} - 1)|$ *differentially private.*

Proof. Lets fix a respondent and a randomized device (i.e., coin flip) with bias b and range $[0,1]$. For a 'Yes' answer from this respondent, we get

$$P(Response = Yes|Truth = Yes) = b + (1-b)b$$

$$P(Response = Yes|Truth = No) = (1-b)^2$$

Thus for 'Yes' answer we have,

$$\frac{P(Response = Yes|Truth = Yes)}{P(Response = Yes|Truth = No)}$$
$$= \frac{b + (1-b)b}{(1-b)^2} = \frac{1-(1-b)^2}{(1-b)^2} = \frac{1}{(1-b)^2} - 1$$

Similarly for a 'No' answer we have,

$$\frac{P(Response = No|Truth = No)}{P(Response = No|Truth = Yes)} = \frac{1}{(1-b)^2} - 1$$

Since, both the probabilities are bounded by $\frac{1}{(1-b)^2} - 1$, Algorithm 3 satisfies $|ln(\frac{1}{(1-b)^2} - 1)|$ differentially privacy. ■

For example, if the randomized device is a regular coin then we have bias $b = \frac{1}{2}$ in range $[0, 1]$. Thus the mechanism would be $|ln(\frac{1}{(1-\frac{1}{2})^2} - 1)| = |ln(3)|$ differentially private [75].

4.5 Experimental Analysis

In this section, we evaluated both the methods according to the proposed attack [3]. We used similar experimental setup according to the original paper to directly benchmark our solution to the attack scenario. We also changed some of their population size and other parameters in order to do further analysis.

4.5.1 Original results

The original simulation [3] were experimented on a sample of $1,000$ individuals containing $500,000$ SNPs which we doubled to $1,000,000$ SNPs. Alternate allele frequencies of these SNPs were sampled from binomial distribution for a standard neutral model under the assumption of a population size of $20,000$ individuals.

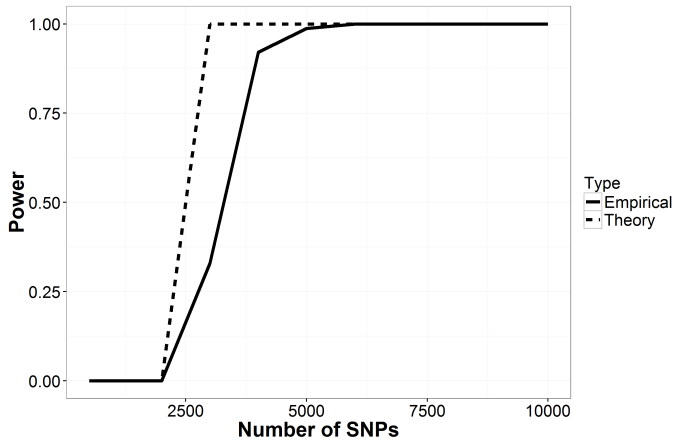


Figure 4.3: Power(LRT) of re-identification attacks of individuals on beacons constructed with 1,000 individuals on our experimental setting without any privacy preserving mechanism

Then the query beacon was constructed having $1,000$ individuals (from $20,000$). We also considered larger beacon sizes with $1200, 1500, 2000$ individuals. Then the log likelihood ratio tests (LRT) to confirm the hypothesis were done assuming,

- 400 individuals from the beacon
- 400 individuals not from the beacon

The comparison between both setups are also shown in Table 4.2.

The outcome of the attack in our setting is depicted in Figure 4.3 where the power of the log-likelihood ratio tests (LRT) are on Y axis while the X axis shows the number of SNPs queried by any adversary. The figure demonstrates that the proposed attack has more than 95% power to detect whether an individual is present in the beacon of $1,000$ individuals with just $5,000$ SNP queries. This result also supports the claim of the original paper [3].

As Equation 4.5 shows the dependency between the LRT outputs and the

Table 4.2: Parameter consideration in our experiment and the original paper [3] (1k=1000)

Parameter Name	Original paper [3]	Our setup
Population Size	10k	20k
SNPs considered	500k	1,000k
Beacon Size	1k	1k, 1.2k, 1.5k, 2k

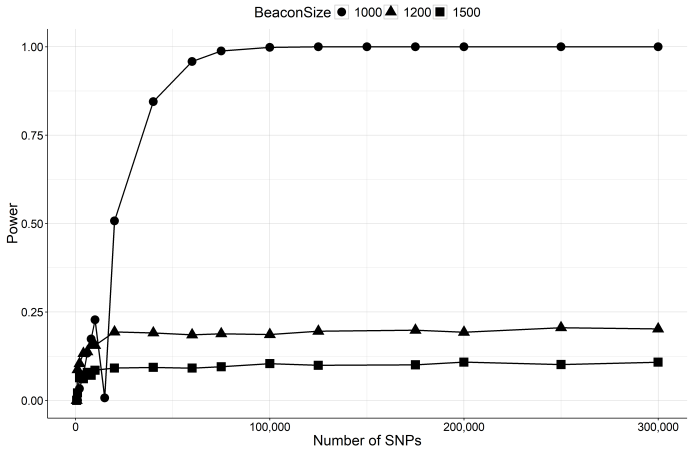


Figure 4.4: Power(LRT) of re-identification attacks on beacons constructed with different numbers of individuals. We show the results of the attack for 3 different beacon database size ($1000, 1200, 1500$)

beacon size (number of individuals, N), we further analyzed the attack for a different number of N. We show the power of the attack for different beacon size $N = \{1000, 1500, 2000\}$ in Figure 4.4.

In the Appendix A, we include the analysis for different genome sequencing error rates and re-identifying the relatives of those 400 individuals. For example, the relatedness (ϕ) can be defined as twins, parent-offspring, siblings, cousins

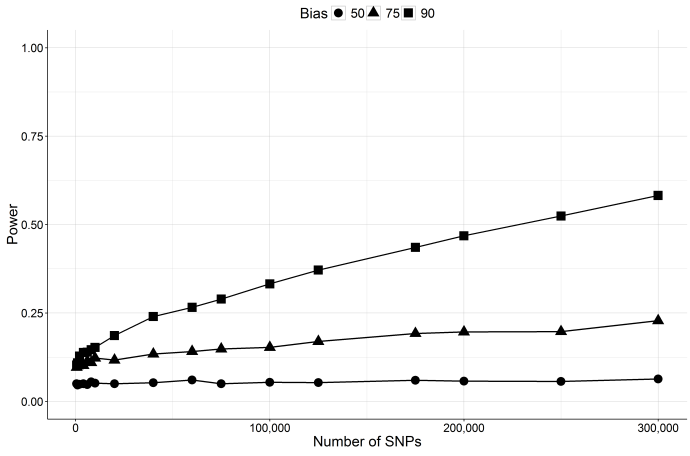


Figure 4.5: Effect of method 1 on the power of the attack on beacon database considering different bias

etc., where $\phi = \{1, 0.5, 0.25, 0.125\}$. As twins share the same genomic sequence, the LRT tests should be similar and conclusive after $5,000$ queries.

4.5.2 Our results

According to the test framework, we evaluated our proposed methods. We employed our privacy preserving mechanisms to perturb the original answer and then evaluated the performance of our techniques.

Figure 4.5 and 4.6, show the results of Algorithm 2 and 3. As expected from the privacy-utility relation, we see that more accurate answer results in less privacy as the LRT powers keep rising for bias 90 (90% accuracy) after 300,000 queries. That is even with only 10% errors, the adversary needs more than 300,000 queries to determine the presence of an individual in the beacon database.

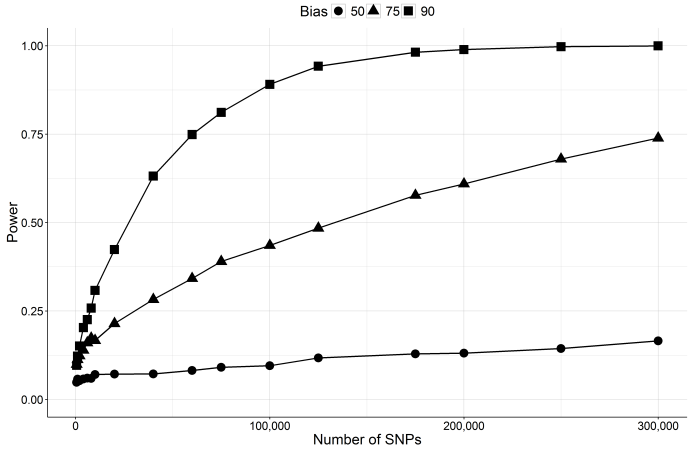


Figure 4.6: Effect of method 2 on the power of the attack on beacon database considering different bias

4.5.3 Accuracy Analysis

As mentioned previously, there is a need for a method which will induce errors to provide the privacy of the individuals' present in a beacon service. Both of our methods add random errors to the beacon database where these errors can be defined as false positives and negatives. In this context, *false positives* are those where the beacon service answered yes regardless of the fact that there was no existence of that data. *False negatives* are those where the beacon answered false to a true answer. Accuracy is defined as,

$$accuracy = \frac{N_{TP} + N_{TN}}{N_{TP} + N_{TN} + N_{FP} + N_{FN}}$$

Figure 4.7 shows the calculation of the accuracy for both of our methods. It is clear from the figure that both methods with a higher bias provide more accurate result. This allows the corresponding data owner to decide the amount of utility they want to provide with respect to the privacy of the individuals.

We also show multiple levels of privacy achieved for different accuracy of the

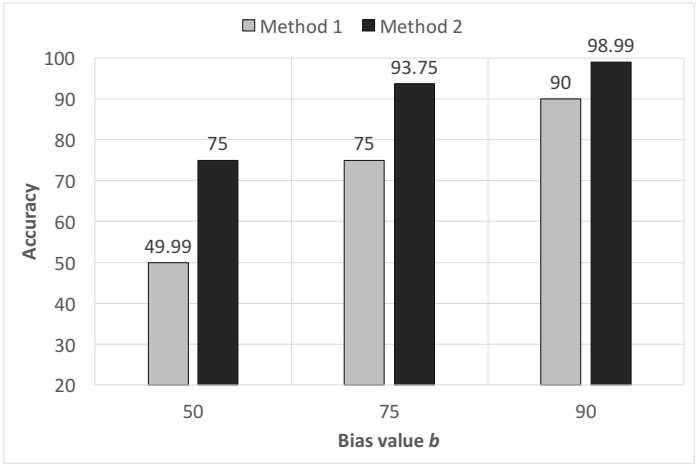


Figure 4.7: Accuracy of both methods with different bias (50, 75, 90) consideration

beacon data. Figure 4.8 shows the different LRT powers for different accuracy of the data in 300,000 queries. As higher LRT powers defines better assumptions from the adversary, we can model it as the privacy loss where utility can be defined as the accuracy aforementioned. It is noteworthy that, higher utility results in higher privacy loss as we can see with 98% accuracy we have LRT power as 1 where 75% accuracy has 0.22.

4.6 Discussion

In this section, we discuss few issues regarding the original attack and the applicability of our solutions.

4.6.1 Different bias on tiered access control

One clear indication from GA4GH and the research community on this privacy issue of genomic beacon service is implementing an access control over this sensitive information [21; 66]. Multiple layers of access control have been proposed

where a different level of users will have different privileges over the beacon service. This kind of hierarchy in accessing a service is often named as 'tiered access control'. The applicability of this model in beacon service is already proposed in a recent study [65].

Our solution methods fit the tiered access control as we have different levels of privacy guarantee for different bias value. Higher bias leading to higher accuracy might be granted to a more trusted user where a public user might only get the lowest utility with high privacy over the beacon data. This will ensure the utility that the beacon promises while not revealing the presence of an individual.

4.6.2 Statistical inference attack

There are two different ways we can incorporate our methods on a beacon service. First, by using them while answering queries in real time and secondly, using them to preprocess the database beforehand to answer queries. In our analysis, we use the algorithms to preprocess the database due to the statistical inference attack. If we use the algorithms in real time, then an adversary might average the outputs of a specific query and obtain the original output.

4.6.3 Different allele frequency assumptions

The original attack scenario assumes that the allele frequency of the dataset follows a beta distribution [3]. However, in real life, an adversary can find the specific allele frequency of any position from a public database. This enables the adversary to launch more powerful attacks against the beacon service. It is noteworthy that iDASH 2016 competition [76] presented the problem under this formulation [77; 78].

However, in this work we also assume that the adversary has limited background knowledge and s/he does not have access to specific frequencies of each position. More rigorous privacy guarantee like *differential privacy* [75] can be provided against a stronger adversary.

4.7 Related Work

Genomic privacy has recently gained significant concern among the general public and research community. De-identification is a common practice in research and clinical practice to protect genomic privacy and confidentiality of the participants. Normally, privacy is achieved by anonymizing a person's identity while sharing genomic related data. Since the de-identified genomic data are typically published with additional metadata and anonymized quasi-identifiers information, these pieces of information can be used to re-identify an unknown genome and thus disclosing the identity of the participant. Significant research has been done so far in this area. Below are some of the recent works related to re-identification attacks in genomic and health-related data.

In the recent study, Sweeney et. al [79] showed that participants in the Personal Genome Project (PGP) can be easily identified based on their demographics without even using any genomic information. They also stressed that 84% to 97% of the participants are correctly identified by linking the demographics to publicly available records such voter list and the name hidden in the attached documents.

Gymrek et al. [80], showed that a person's identity can be exposed via surname inference by profiling short tandem repeat on the Y-chromosome and querying recreational genomic genealogy databases. In their study, they showed that by

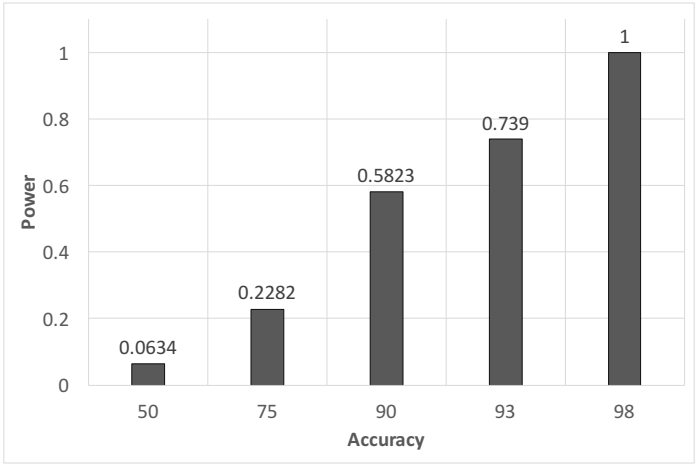


Figure 4.8: Privacy-Utility curve for different accuracy on X-axis and their corresponding LRT powers for 300,000 queries

scanning two largest Y-chromosome genealogical websites, 10-14% US white male individuals are subject to surname inference attack. Moreover, when the attacker gains access to that target DNA sample, they can simply search available genomic databases with sensitive attributes (e.g., drug abuse). Hence, the person's identity with attributes can be easily found.

In recent study [81], Gitschier showed that a surname of an individual participating in HapMap database can be inferred by the combination of information from genealogical registries and a haplotype analysis of the Y-chromosome collected for the HapMap Project. In [82], the authors presented an attack that involves the association of DNA sequences to personal names, through diagnosis codes.

Zhou et al. [83] studied the privacy risks of releasing aggregate genomic data and showed that individuals participating in such research study can be easily identified and for some cases, their DNA sequences can be fully recovered. They have proposed a risk-scale system to classify aggregate data and a guide for their release.

Homer et al. [84] proved it is possible to detect the presence of an individual in a complex genomic DNA mixture even when the mixture contains only trace quantities of his or her DNA.They showed that an individual participating publicly released Genome Wide Association Study (GWAS) can be easily identified by his/her known genotypes and analysing the allele frequencies of a large number of SNPs.

Wang et al. [85] showed a higher risk that individuals can actually be identified from a relatively small set of statistics such as those routinely published in GWAS papers. Their first attack is the extension of homer's attack and showed that the presence of an individual in the case group can be determined based upon the pairwise correlation among as few as a couple of hundred SNPs. The second attack can lead to a complete disclosure of hundreds of the participants' SNPs, by analyzing the information derived from the published statistics.

In another study, Malin and Sweeney in [86] introduced re-Identification of Data in Trails (REIDIT) algorithms which link individuals genomic data to the publicly available records. They showed that it is possible to identify a person by looking at the unique features in patient-location visit in a distributed healthcare environment.

Other than these, there are multiple surveys available which summarizes and demonstrates some other attacks [7; 87].

4.8 Future Work

Recently, a new attack model has been proposed for the beacon web service [77] which takes individual allele frequency into consideration as mentioned in Section

4.6.3. Hence, we need to study that attack and test our solutions for that adversary model. However, the new attack [77] is much stronger as it considers individual frequency rather than a global distribution.

Chapter 5

Privacy Preserving Techniques for Outsourcing Genomic Databases

5.1 Introduction

The recent advancement in genomic data generation and availability has impacted related scientific studies. These large genomic datasets help us to understand the relation between some diseases and our genes. Researchers can now analyze susceptibility to specific diseases or even make personalized medicine based on the patients genomic sequence [88; 4]. Therefore, the continuation and advancement of medical research heavily depend on the availability of large genomic datasets.

There is an urgent need for a unified genomic data repository [89]. The current practice is to keep genomic records within organizations (hospitals, government agencies, etc.) and share them with specific researchers on request. It allows re-

searchers to gain access to genome sequences and conduct their studies. However, one critical flaw in this approach is each organization has to take responsibility of gathering, storing and sharing data independently. It instigates concerns among privacy and security specialists because these duties are time consuming and sometimes well beyond the abilities and expertise of small organizations. Also there are issues of sharing data beyond borders or even a single province of a country [90]. A possible solution that addresses these concerns is establishing a centralized data repository to store genomic data and securely share this information among researchers. Outsourcing genomic data in a unified repository will accelerate research and reduce required time and cost for research studies. Therefore, many projects have started to collect and centralize genome sequences and other sensitive attributes of individuals in a unified repository [91].

A recent survey documented 14 different techniques for breaching genomic data privacy, and classified these techniques under three categories: identity disclosure attacks, attribute disclosure attacks, and completion attacks [8]. These techniques question the efficacy of the protection mechanisms that are currently in practice for genomic data sharing. This reality demands new privacy preserving framework that can provide data privacy of genomic databases and prevent inference attacks against outsourced data.

The objectives of this work is to present a privacy-preserving framework to store and execute queries on genomic data. In particular, the framework should ensure the privacy of individuals (e.g., genome sequence or sensitive attributes) and the query execution mechanism should be scalable to large databases.

Current Techniques

One of the earlier attempts to securely outsource genomic data in a cloud is a cryptographic model proposed by Kantarcioglu et al. [92]. Their proposed model encrypts every record by a homomorphic encryption scheme. Although encrypting all data prevents malicious observers from gaining any information and provides much security, their method has two shortcomings. First, the query execution time for their method is quite large. As mentioned in their paper, it takes around 30 mins to execute a count query over 40 SNPs in a database of 5000 records. Therefore, their method may not be scalable for big databases containing millions of records. Second, due to the use of homomorphic encryption scheme, which has a large ciphertext expansion factor, the encrypted database requires large storage space. Canim et al. later presented a cryptographic model to improve the efficiency of the previous scheme [93]. However, the proposed model requires a secure co-processor, which is not always possible to ensure in real-life scenarios. In practice, we outsource our data to cloud servers (e.g. Amazon EC2) where it is not easy to ensure the availability of a secure hardware. These are the only two solutions that are closely related to the problem of secure query execution over outsourced genomic data.

There are a number of other cryptographic solutions proposed for genomic data privacy over the past few years [20; 59; 94; 95]. These methods, although related, do not address the problem discussed in this work.

Contributions

In this work, we propose a private and efficient model that addresses the challenges of outsourcing genomic data for query execution. Our proposed model sup-

ports two types of queries: count query and top-k. In a count query, researchers indicate some positions of SNPs along with their values. The query result returns the number of individuals matching this query predicate. Top-k query determines k records in a database that are most similar to a given reference sequence. Other complex functions can be computed using these primitive queries.

The privacy of genomic databases in an outsourced model can be guaranteed by encrypting each record with a semantically secure encryption scheme (as done by the existing technique). Encrypted data provide protection against internal (i.e., cloud service provider) or external (i.e., adversaries exploiting system vulnerabilities) parties as the decryption key is only known to the data owner. However, the main shortcoming of this approach is efficiency. We propose a private and efficient model for data outsourcing *without encrypting the whole database*. In designing our solution, we attempt to trade off between the privacy, efficiency, and storage. Our proposed framework is very efficient for small number of phenotypes compared to other crypto-based solutions (see supporting documents for more discussion).

The contributions of this work are summarized below:

- Privacy: It guarantees privacy of participants' genome sequences and their sensitive attribute (e.g., disease).
- Type of Queries: The proposed model can privately execute count and top-k queries.
- Efficiency: The proposed model has been implemented and tested with both real-life and synthetic datasets. Experimental results demonstrate that a count and a top-k query over 40 SNPs in a database of 20,000 records takes

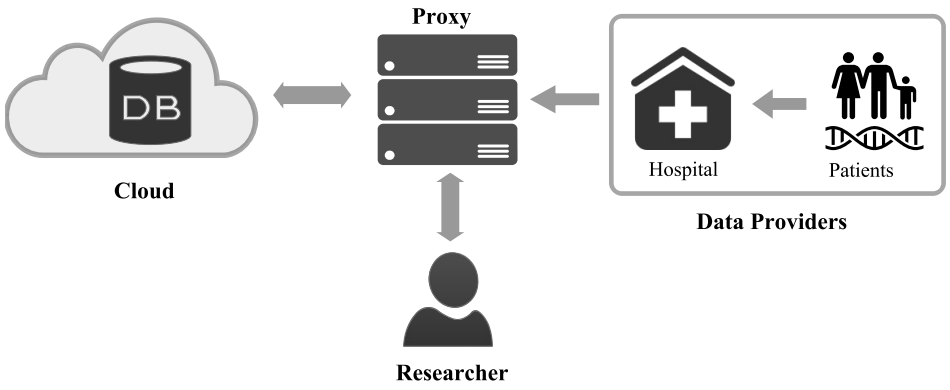


Figure 5.1: Main architecture of the proposed scheme is consisted of four entities: researchers, cloud, proxy and data providers

around 100 and 150 seconds, respectively.

The rest of the chapter is organised as follows. Section 5.2 provides an overview of the proposed method. Section 5.3 reviews genomic dataset, and security requirement of the framework. In Section 5.4, we introduce the concept of valid permutation, which is used in our method to protect the privacy of individuals. We describe our proposed model in Section 5.5. Section 5.6 analyzes how the proposed model preserves privacy of individuals. The experimental results are presented in Section 5.7. We finally review previous works in Appendix A.

5.2 Overview Of Presented scheme

The proposed model includes four major parties: proxy, cloud, researchers and data providers. The architecture of the proposed model is shown in Figure 5.1.

- Data Providers: Data providers (e.g., hospitals) possess raw genomic databases and provide their data to proxy. Proxy after preprocessing stores the data in

the cloud. Once stored in the cloud, no further interaction is needed between data providers and the cloud to execute any query.

- Proxy: Proxy is a trusted party which plays an important role in the proposed model. It preprocesses raw data by encrypting, permuting and adding fake genomic sequences. After preprocessing, it stores the modified version in the cloud. Proxy also translates queries received from researchers to facilitate query execution by the cloud. Finally, proxy receives encrypted results from the cloud and it returns query results to researchers after decryption.

- Cloud Server: Any commercial cloud service provider can play this role as they offer a cheap and highly scalable solution for any organization. Cloud receives translated queries from the proxy and it executes these queries on the stored data. The query execution process does not leak any information to cloud since the whole computation is done on modified data. In this discussion, DB refers to the database inside the cloud.

- Researchers: Researchers might be an individual or an organization who want to execute queries over genomic databases. Researchers submit their queries to the proxy.

The proposed framework is practical for real-life application scenarios. We assume that the cloud is an untrusted entity as it is operated by third party organizations (e.g., Amazon, Microsoft). In addition, there are different privacy regulations (e.g., HIPAA, CLIA) which do not allow the usage of public cloud without ensuring privacy protection [96]. For example, NIH allows researchers to use public cloud; however, it is the responsibility of the researchers to ensure data security and privacy [97]. On the other hand, the proxy is unconditionally trusted in our

Table 5.1: Raw genomic data

	Genomic Sequence					Phenotype
Rec #	POS_1	POS_2	POS_3	POS_4	POS_5	Cancer
1	A	T	T	G	C	No
2	T	A	T	G	C	Yes
3	A	T	G	T	C	No
4	A	T	T	T	G	No
5	T	A	G	T	C	No
6	T	A	T	T	G	No
Freq.	T=0.5 A=0.5	T=0.5 A=0.5	T=0.67 G=0.33	T=0.67 G=0.33	C=0.67 G=0.33	-

framework. This assumption is not far-fetched because there are government organizations (e.g., National Institutes of Health (NIH) in US) that are trusted and currently mediate data sharing requests by researchers. These government organizations provide access controls and other services. Hence, these organizations can play the role of proxy and connect researchers to the cloud.

5.3 Preliminary

In this section, we present the format of genomic dataset and privacy requirement of our model.

5.3.1 Data

Table 5.1 shows an example of raw genomic dataset (ignore the frequency rows for now). We can categorize the attributes of each record, Rec_i, into two categories: *a*) genomic sequence and *b*) phenotype. A genomic sequence is consisted of four letters A, T, G and C. Genome sequence is sensitive because it contains information regarding physical characteristics of an individual [98]. We store this sequence according to their positions. Phenotype attribute contains an individual's observable physical trait. In this method, we assume that the phenotype attribute refers one disease affection status.

5.3.2 Privacy Requirements

A genomic database consists of genomic sequences and disease status of patients. Recent research results show that given some background information about an individual, an adversary can identify or learn sensitive information about the victim from the dataset [18; 12]. If an adversary is able to link a record to an individual then she will also learn the sensitive attribute (the value of the attribute Cancer), since the relation between genomic sequence and phenotypes are not eliminated. Therefore, a model provides privacy protection if it prevents revealing genome sequence and also thwarts inference attack against victim's sensitive value.

Definition 5.3.1. We consider a genomic dataset (DB) secure if,

- It does not reveal the original genome sequence.
- Given a victim's genome sequence g_{victim}, an adversary $\mathcal{A}$ after observing the DB cannot guess better than random ($1/2$) the value of the sensitive attribute.

In this work, we assume that the cloud is *semi-honest* and *non-colluding* with other parties. A party is semi-honest when it follows the protocol. However, it might be curious to gain more information from the messages of the protocol execution. Non-colluding means a party does not collude with others to derive more information. Please see [99] for a formal definition regarding semi-honest adversary model.

5.4 Valid Permutations

In this work, we use a permutation technique over genome sequences to prevent the cloud from getting meaningful information about any genome sequences. Any permutation over a set with n elements can be represented as follows,

$$(i_1, i_2)(i_3, i_4) \cdots (i_{n-1}, i_n),\ i_j \in \{1, 2, \ldots, n\} \tag{5.1}$$

We cannot use any permutation because of the available background knowledge regarding genome sequences. An adversary can use the background knowledge about a specific position and its allele frequency to undo a permutation. Further discussion on this is available in the supporting document.

For example, if the proxy applies $(1, 4)$ permutation to Table 5.1, background knowledge of allele frequency will reveal this permutation due to different frequencies. However, the permutation $(1, 2)$ is a valid permutation because both the first and the second column have same nucleotide set (i.e., 'A' and 'T') and same nucleotide-frequency ('A' and 'T' appear with half probability).

Definition 5.4.1. A permutation over genomic sequences is called valid if it swaps columns with same nucleotide set having same frequency. The set of all of the valid permutation is denoted by $\mathcal{VP}$.

If valid permutations are applied, the permuted genome sequences will inherit statistical features of the original data. Therefore, no adversary can infer information about the permutation. Obviously, $\mathcal{VP}$ is strongly related to background knowledge about genomic sequences. For example, in Table 5.1, the first and the second column can be swapped. Similarly, third and fourth columns can be swapped. Therefore, we have three valid permutations that can be applied $\mathcal{VP} = \{(1,2),(3,4),(1,2)(3,4)\}$ for Table 5.1.

5.5 Proposed Model

In this section, we present our proposed model. The dataset is first preprocessed by the proxy and then stored at the cloud. The cloud uses the stored database to execute queries.

Data Preparation

Proxy receives genomic dataset (i.e., Table 5.1) from the data provider. Data preparation phase has two steps: *Valid Permutation* and *Adding Fake Records*.

Permutation

Proxy permutates genome sequences according a valid permutation π ($\pi \in \mathcal{VP}$) as described in Section 5.4. Proxy then generates a public-private key pair for the homomorphic encryption $E(.)$ and computes $(E(1), \pi(g), A_g)$ for each record where A_g is the sensitive attribute like disease association. Suppose the chosen valid permutation is $(1,2)$, then we get Table 5.2 from Table 5.1.

Table 5.2: Permuted genomic data by valid permutation (1,2) from Table 5.1

Rec #	Tag	Sequence					Cancer
1	E(1)	T	A	T	G	C	No
2	E(1)	A	T	T	G	C	Yes
3	E(1)	T	A	G	T	C	No
4	E(1)	T	A	T	T	G	No
5	E(1)	A	T	G	T	C	No
6	E(1)	A	T	T	T	G	No

Adding Fake Records

The second privacy requirement is to remove inference attack on sensitive attribute (e.g., disease) using the knowledge of the genomic sequence of a victim. We add fake genomic sequences to permuted sequences to achieve this objective.

Proxy adds fake records to original dataset to disassociate the relationship between genomic sequence and sensitive attribute. For any $g \in DB$, the proxy adds $(E(0), g, YES)$ or $(E(0), g, NO)$ to the dataset. The encryption of the tag 0, denoted by $(E(0)$, indicates that this record is a fake record. All records are shuffled randomly before sending data to the cloud to hide fake records from original ones. The modified dataset after adding fake records as shown in Table 5.3 which will be shuffled and then sent to cloud for query execution.

Query Execution

The cloud is able to execute count and top-*k* queries on the stored database. Following, we elaborate the procedure for query execution.

Count Query

1. Researchers submit a count query as follows:

```
SELECT * FROM Sequences WHERE POS_1='A' AND POS_2='T'
AND CANCER='YES'
```

2. Proxy translates this query to

```
SELECT * FROM Sequences WHERE POS_1='T' AND POS_2='A'
AND CANCER='YES'
```

   where $\pi(1) = 2$ and $\pi(2) = 1$.

3. After getting translated query, the cloud computes the output as follows:

   (a) $ANS = E(0)$.

   (b) For $(E(\theta_g), g, A_g) \in DB$ if g and A_g satisfies the translated query then $ANS = ANS + E(\theta_g)$, where $\theta_g \in \{0, 1\}$.

4. Cloud sends ANS to Proxy.

5. Proxy decrypts ANS and delivers the answer to the researcher.

The cloud gains no information about the output of the query as it is encrypted by a semantically secure encryption system.

Example. After receiving the translated query (POS_1 = T and POS_2 = A and CANCER = YES), the cloud initializes $ANS = E(0)$. In Table 5.3, record number 2, 6 and 8 match the query predicate. Cloud adds their tags to $ANS = ANS + E(0) + E(0) + E(0) = E(0)$ and gives ANS to the proxy. Proxy after decryption obtains the answer (which is zero) and finally sends it back to researcher.

Table 5.3: Dataset after adding fake records (indicated by white rows) which is shuffled and sent to the cloud for storage and computation

Rec #	Tag	Permuted Genomic Sequence					Cancer
1	E(1)	T	A	T	G	C	No
2	E(0)	T	A	T	G	C	Yes
3	E(1)	A	T	T	G	C	Yes
4	E(0)	A	T	T	G	C	No
5	E(1)	T	A	G	T	C	No
6	E(0)	T	A	G	T	C	Yes
7	E(1)	T	A	T	T	G	No
8	E(0)	T	A	T	T	G	Yes
9	E(1)	A	T	G	T	C	No
10	E(0)	A	T	G	T	C	Yes
11	E(1)	A	T	T	T	G	No
12	E(0)	A	T	T	T	G	Yes

Top-k Query

1. Researchers submit a top-k query as follows:

```
SELECT * FROM Sequences WHERE POS_1='A' AND POS_2='T'
LIMIT k
```

   where k is the number of records.

2. Proxy translates the query to

```
SELECT * FROM Sequences WHERE POS_1='T' AND POS_2='A'
LIMIT n
```

where $\pi(1) = 2$, $\pi(2) = 1$ and n is security parameter and have no relation with k.

3. Cloud finds n nearest records and sends to the proxy.

4. Proxy decrypts tags of these records to verify if there are k original records. If there are less than k original records, proxy sends a message to the cloud to send the next n nearest records.

5. Proxy repeats this procedure until it receives k nearest original records.

6. After getting k records, proxy sends the records to the researcher.

The variable n is introduced to hide the variable k (required limit) from the cloud. In each iteration, the cloud sends n records containing both fake and real records. Therefore, the cloud does not know exactly what is the real value of k.

Example. Suppose in a top-k query (POS_1=T and POS_2=A), the values for the parameters k and n are 5 and 3, respectively. Hence, the cloud finds 3 nearest records that are sent back to proxy. These records are # 1, 2 and 5 in Table 5.3. After decryption of the tag, proxy obtains two ($1, 5$) original record. As its a top-5 query, it requests for the next three nearest records which are # 6, 7 and 8. After proxy decrypts these values, it gets only one original record. Therefore, the proxy asks for 3 more records which the cloud does not not have. At this stage, proxy obtains k (or less) number of original records according to the query. Finally, it sends 3 records # 1, 5 and 7 to the researcher.

5.6 Privacy Discussion

In this section, we discuss the privacy analysis of the proposed method. We provide a sketch of how this model thwarts privacy attacks and present the formal proofs in the Appendix B.4 and B.5.

5.6.1 Preventing Cloud to Retrieve Genomic Sequences

We mentioned two major privacy concerns. The first one is that the genomic sequence should not be revealed to the cloud. To achieve this requirement, proxy applies a valid permutation on genome sequences. The following theorem shows why applying a valid permutation prevents the cloud from gaining original genome sequences.

Theorem 2. *Given a permuted genomic database DB and for any two possible valid permutation keys $\pi_1, \pi_2 \in \mathcal{VP}$, we get,*

$$Pr(\pi_{key} = \pi_1 | DB) = Pr(\pi_{key} = \pi_2 | DB)$$

where π_{key} is the chosen key for this permuted database DB.

The proof of this Theorem in the Appendix B.4. Therefore, permutation is irreversible and it is impossible for an observer to rearrange the columns and retrieve the original genome sequence.

5.6.2 Removing Disease Association

The second security requirement is to eliminate the association between a genomic sequence and the corresponding sensitive attribute (e.g., disease). We show that cloud cannot exploit relation between genomic sequences and sensitive attribute.

Theorem 3. *Suppose a genomic sequence g is given. An adversary $\mathcal{A}$ after observing DB cannot guess better than half the sensitive value of g.*

The formal proof is provided in the Appendix B.5. Suppose, Alice's (victim) genomic sequence g_A is known to adversary (i.e., cloud). This does not help the adversary to gain knowledge regarding the sensitive attribute of Alice. This is because according to the Theorem 1, the adversary cannot gain information about the permutation. Moreover, the proxy disassociates the relationship between genomic sequence and sensitive attribute by adding fake sequences. Thus, an adversary finds that half of possible candidate records for Alice genomic sequence have cancer while the other half do not. Therefore, observing even the full database does not help the adversary to obtain Alice's disease *better than a random guess*.

5.6.3 Limitation

However, this framework is still vulnerable to Homer et al.'s attack [12] as we preserve the frequency of individual position. As we use commercial cloud platforms for data storage or computation, the risk of re-identification is not totally mitigated. In other words, any malicious cloud can re-identify specific person (if it has the original sequence) but still cannot guess better than random about the disease association. Since we do not encrypt the whole sequence for efficiency, this vulnerability remains for the malicious cloud (although we assume the cloud to be semi-honest in section 5.2).

We assume for Theorem 1 that the polymorphic sites are independent. In real life, certain SNPs are correlated and might be utilized together to infer the valid permutation key. However, the risk from linkage disequilibrium between two different sites is hard to quantify in general and permutation on the original

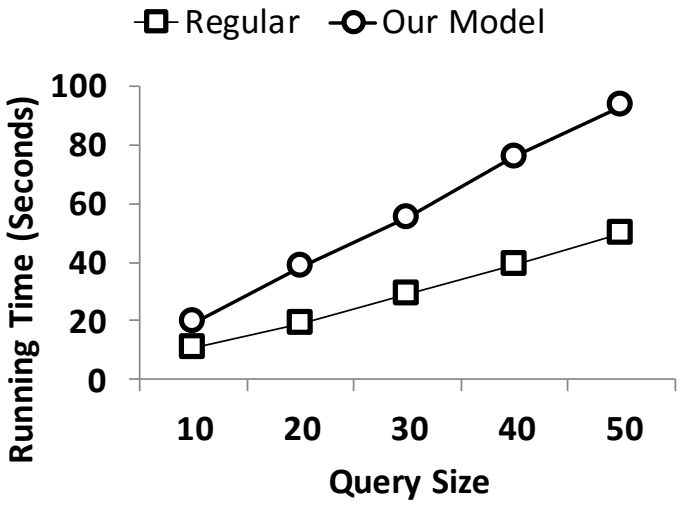


Figure 5.2: Running time (in seconds) of count query over real data with different query sizes

sequence impacts this as well. Thus, it is not clear how an adversary can use this information to undo a permuted database.

5.7 Implementation

In this section, we experimentally evaluate the performance of the proposed model. To simulate the real-life scenario, we placed the proxy and the cloud in two different machines located in two different geographic regions. For the cloud, we use Amazon EC2 in Oregon U.S. with t2.micro configuration. The proxy was on a separate machine (Ram 8GB, Intel i5 3.3 Ghz) located in Winnipeg, Canada. The source code is available publicly at `https://github.com/mominbuet/PermutationGenomicData.git`.

We consider the following aspects in order to assess the efficiency of the our

proposed method.

- *Query time:* Time needed to answer any count or ranked query.
- *Insertion time:* Time needed to process genomic database by the proxy.
- *Space complexity:* Storage space needed to store the database (both original and fake records) in the cloud.

We also compare our proposed method with a non-secure (i.e., no permutation and addition of fake records) scheme and the method proposed by Kantarcioglu et al. [92]. Their model is objective-wise similar to ours as it answers count query but uses homomorphic encryption to encrypt the entire database.

We use both real-life and synthetic data sets for evaluating our model. The real-life data is taken from iDash competition 2015 [55] where there are 311 different SNPs of 400 different participants divided into case and control groups. We use the case-control group separation to denote the value of the sensitive attribute *Cancer*. For synthetic data, we took the allele frequency of CHB, CHS, JPT and MXL population from *1000genomes* dataset (August 2010 Release) and generated *60,000* data rows according to that frequency.

We evaluate our method using the real-life data in Section 5.7.2. The comparison is done for both secure and non-secure versions. For non-secure version, we do computations on plaintext to get the run time for operations. We call this model as *regular model*. We then perform the same operations for the proposed model and report their run times. For synthetic dataset, we use 5K, 10K, 15K and 20K records to compare the model with the regular and the Kantarcioglu et al.'s model [92] (5.7.3).

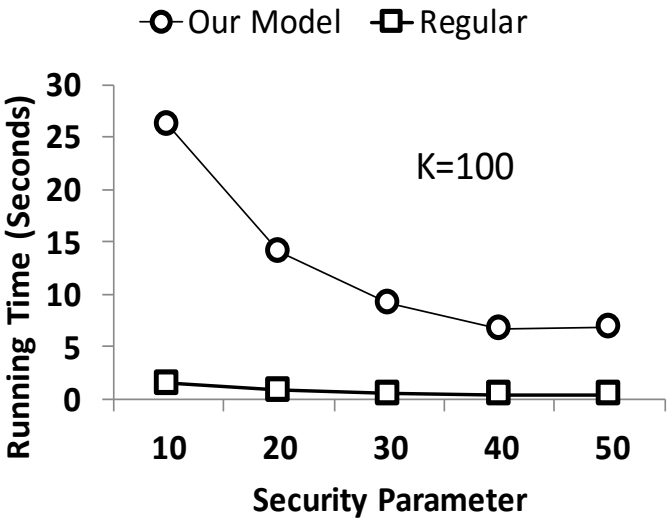


Figure 5.3: Running time (in seconds) of top-k query over real data with different security parameters where $k = 100$

5.7.1 Insertion Time and Required Space

Two important factors in efficiency are insertion time and required space for the data. The time required for processing the raw data (provided by the data owners) and storing them in the cloud is called insertion time. Table 5.4 shows insertion time and required space for different models.

Unfortunately, the insertion time is not reported in Kantarcioglu et al's paper [92]. However, their method offers stronger security by encrypting the whole database and the expanding factor is bigger due to the encryption scheme (the size of each ciphertext is 1024 bits). Therefore, we can conclude that it has significantly higher insertion time than our scheme. Our model's insertion time is almost *four times* greater than that of regular model. Moreover, Table 5.4 shows that the proposed model requires double the space of the regular model because of fake genomic sequences.

Table 5.4: Insertion Time and Required Space for different data size

Model	Data Size	Insertion min.	Expanding Factor
Regular	400	2	1
	5K	70	
	10K	135	
	15K	213	
	20K	270	
Kantarcioglu [92]	Any	–	1024
Our Model	400	8.7	2
	5K	257	
	10K	511	
	15K	763	
	20K	1102	

5.7.2 Query time for real-life data

The run times for count and top-k queries over the real-life data are shown in Figures 5.2 and 5.3.

According to the balancing technique, the size of database doubles due to the addition of fake records. Therefore, the required time for query execution also doubles for the additional records. In addition, the proxy needs to do small amount of work to get the final result. Figure 5.2 shows that our proposed model pays a small penalty in terms of execution time to achieve security.

Figure 5.3 shows the query time for top-k query. With bigger security parameter n, it takes less time to answer queries. This happens because the proxy asks

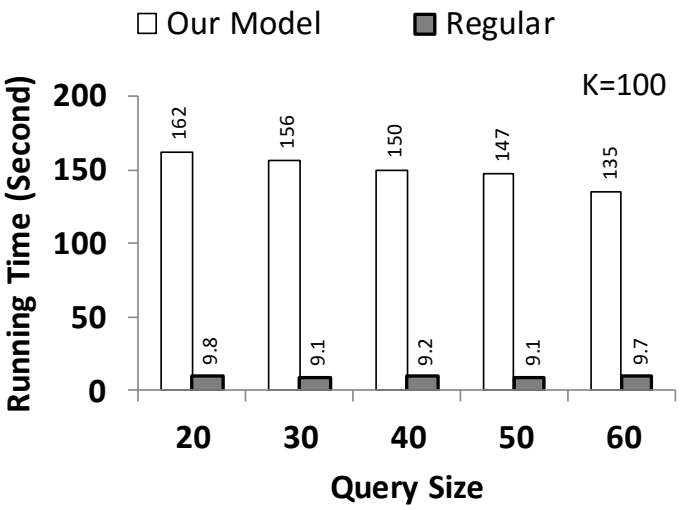


Figure 5.4: Implementing model over a synthetic database with 20k records and $K = 100$. The X-axis shows different security parameter for top-*k* query.

for more records in each time resulting less number of iterations with the cloud.

5.7.3 Query time for synthetic data

In this section we present our experimental results for 5K, 10K, 15K and 20K records. Figure 5.5 shows the results for count query on different data sizes.

Experimental results show a big difference in running time of our model compared to the existing method (logarithmic scale). This is because the existing technique encrypts the whole database using homomorphic encryption while we only encrypt the tag (original/fake). Homomorphic encryption imposes heavy computational cost on the cloud effecting both running times and storage requirement.

Second type of query is top-k query. To the best of our knowledge, the problem of secure top-k query on outsourced databases has not yet addressed. Recent work addresses the problem of secure edit distance [44] which is related our problem; however, they have a different architecture and do not provide a benchmark for

genomic data.

We also evaluated the scalability of the proposed method by varying the number of SNPs. The query time does not depend on the number of SNPs. Due to the space constrain, we report these results in the Appendix B.1.

5.8 Related Works

There are two main approaches in ensuring genomic data privacy [6]: Non-cryptographic and cryptographic approach. In this section, we provide a brief overview of existing techniques adopting these two approaches. For more discussion, we refer interested readers to recent surveys [6; 8; 41].

5.8.1 Cryptographic Approach

Cryptographic approaches perform computation on encrypted data to ensure the privacy of individuals. Many existing techniques adopt a semi-honest server (e.g., cloud) to store and perform computation over genomic data.

In [92], the authors presented a cryptographic approach to outsource genomic sequences in a cloud server. They encode genomic sequences and then encrypt by a homomorphic encryption.

Canim et al. [93] leveraged a trusted hardware inside untrusted cloud to ensure privacy. This secure hardware helps server to execute queries independently. Instead of homomorphic encryption, the authors uses symmetric cryptosystem. However, both of these techniques can only process count queries; whereas, the proposed method presented in the work can execute count as well as top-k queries.

Some recent works from Lauter et al. [58] shows some secure versions of

statistical algorithms used in genomic studies like Hardy-Weinberg Equilibrium, Pearson Goodness-Of-Fit Test, Linkage Disequilibrium etc. Though their targeted genomic dataset is similar to ours but their solution relies on the rigorous security definition of homomorphic encryption. This involves modification of the algorithms and also introduces a tradeoff between accuracy and security.

A new notion of 'Similar Patient Queries' was introduced by Wang et al. [44] which showed the importance of secure ranked query. The main contribution of this work was an approximation of Edit Distance which can be securely computed between two parties. With this distance (or string difference) you can rank the sequences of different patients and get similar patients like the searched one. But the level of approximation used to rank the results has errors which we do not have.

There are a number of other cryptographic solutions proposed for genomic data privacy [20; 59; 94; 57; 95; 100; 101; 102; 103]. These methods, although related, do not address the problem discussed in this chapter. These techniques use either homomorphic encryption [104; 105], Yao's garbled circuit [35] or both as the underlying secure computation primitive. In this method, we use homomorphic encryption minimally to devise a private and efficient framework for genomic data outsourcing.

5.8.2 Non-cryptographic Approach

Non-cryptographic approaches adopt various sanitization techniques to ensure the privacy of genomic data. Privacy preserving data publishing (PPDP) is a well studied domain and has been researched extensively for various types of data [106]. These techniques study how to transform raw data into a version that is immu-

nized against privacy attacks but that still preserves useful information for data analysis. Existing techniques first sanitize raw data and then release the sanitized data for public use. Once shared, the data owner has no further control over the shared data.

Existing techniques are primarily based on two major privacy models: k-anonymity [107] and ϵ-differential privacy [108]. In spite of its wide applicability in the healthcare domain [109; 110], recent research results indicate that k-anonymity based techniques are vulnerable to an adversary's background knowledge [111; 112; 113; 114]. This has stimulated a discussion in the research community in favor of the ϵ-differential privacy model, which provides provable privacy guarantees independent of an adversary's background knowledge. However, it is not well understood yet whether differential privacy is the right privacy model for biomedical data as it fails to provide adequate data utility [8].

To satisfy a specific privacy model, while many anonymization techniques have been proposed for various type of data (i.e., relational [115], set-valued [116], spatio-temporal data [117]), the problem of genomic data anonymization has been little studied. Recent methods [118; 119] propose to generalize genomic data according to k-anonymity privacy model. However, these techniques do not provide comprehensive privacy protection. Malin [120] presented how to anonymize genome sequences without health data, and Loukides et al. [118] and Heatherly et al. [119] proposed to anonymize only health data (i.e., no protection for genome sequences). In addition, all these methods adopt k-anonymity as the underlying privacy model. Therefore these are vulnerable to the recently discovered privacy attacks [111; 112; 113; 114]. More recently, differentially private mechanisms [121; 122] have been proposed for genomic data. However, these techniques only

release some aggregate information (e.g., minor allele frequencies).

One of the limitations of the non-cryptographic approach is that there is a trade off between privacy and utility. All the proposed methods compromise significant amount of utility while protecting privacy. For example, differentially private mechanism may provide wrong information due to noise addition. Therefore, cryptographic approach has recently received much attention as an alternative approach to protect genomic data privacy.

5.9 Future Work

One of the limitations of the proposed method is that it does not consider multiple phenotypes. Though this method scales well to the number of records, it only provides security guarantee and efficiency for one phenotype (disease information). Extending this framework to support arbitrary number of diseases is an important future work.

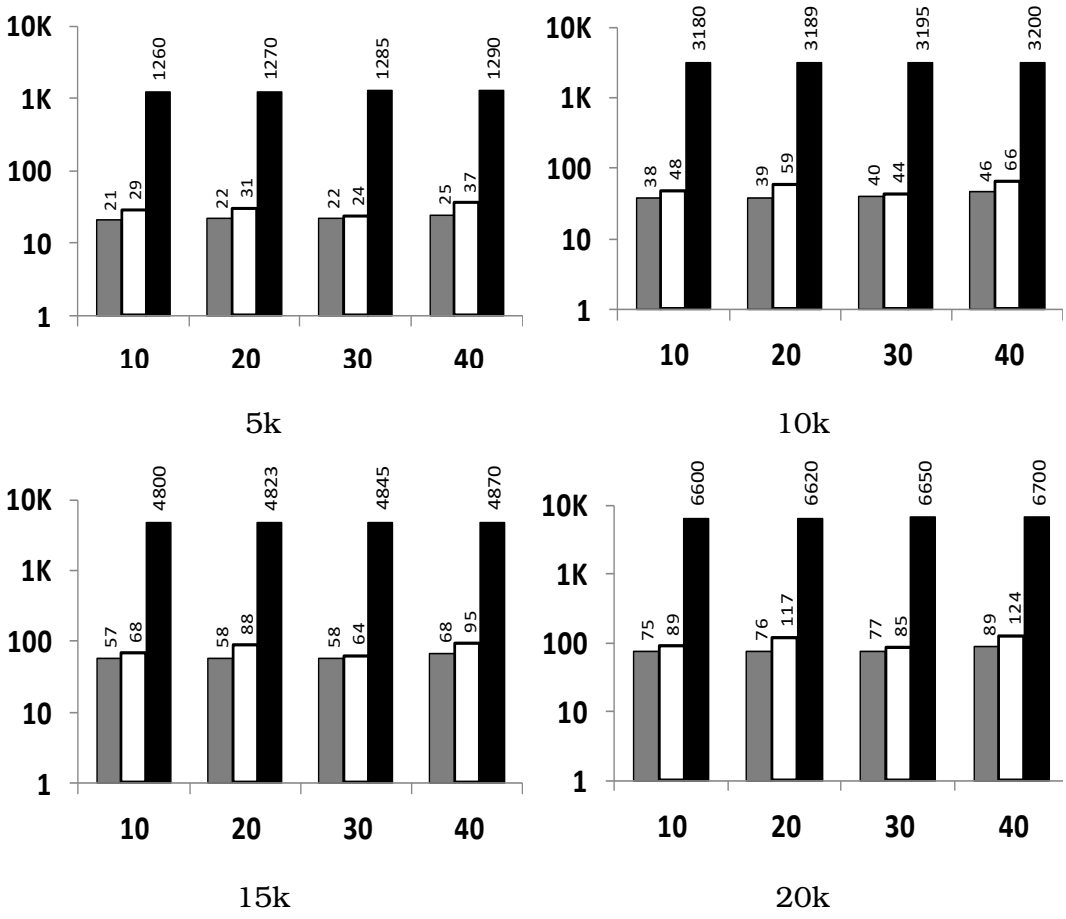


Figure 5.5: These tables show the results of implementation the model over four synthetic databases with 5k, 10k, 15k and 20k records. X-axis shows the query size for regular, our method & [92]'s method respectively and the Y-axis shows running time for addressing any query (logarithmic scale). These figures demonstrate that running time for count query is more or less twice the required time in regular model while the tables shows the difference in answering queries from Kantarcioglu *et al*'s work [92]

Chapter 6

Secure Approximation of Edit Distance on Genomic Data

6.1 Introduction

Similar Patients Query (SPQ) [2] is used to identify similar patients from a large number of medical sources. The similarity is measured based on the sequenced genomes of patients. Nowadays sequencing and interpreting genomic information is cheaper and easier than ever. However, executing SPQs has been seen as a double-edge sword. The results of executing SPQs will lead to a better diagnosis of diseases and early detection of certain diseases. On the other hand, executing SPQs raises some security and privacy concerns. DNA sequences include health and other information about patients and their families. The disclosure of such genomic sequences could harm patients from different perspectives such as affecting the employment and the education opportunities. What makes things more serious, are some federal laws to address privacy issues such as the Health Insurance

Portability and Accountability Act (HIPAA) [123]. HIPAA is the United States' legislation that provides data privacy and security provisions for safeguarding medical information. Accordingly, there is a desideratum to privately execute SPQs over genomic data.

Edit distance or Leveshtein Distance [124], which has been a popular metric of string similarity, can be defined as the minimum number of operations (insertions, deletions and substitutions) required to convert one string to another. This metric is widely used in different problems for its superior utility and accuracy over other string distance metrics such as hamming distance and JaroWinkler distance [125]. For human genomic data, edit distance seems to capture the requirement as we can find similar patients [2] based on genomic information. However, this superiority comes with a cost as edit distance is a quadratic time algorithm. That is, given two strings with n lengths, it requires $O(n^2)$ operations to compute the edit distance; this is not acceptable for long string sequences. For this reason, edit distance problem has been studied over the years by the theoretical computer science community in order to find a better alternative, a faster algorithm [126; 127], or an approximation algorithm. Particularly, in human genomic data where we have billions of base pairs and genomic sequences are constructed with nucleotides (*A, T, G, C*), this algorithm falls short as most datasets contain millions of records. For this reason, other algorithms of string similarity to deal with genomic data have been proposed [128; 129]. These algorithms have been mainly diverged into two directions, either designing faster algorithms by bounding the algorithm or resorting to an approximation which is the approach that we adopt in this work.

Privacy and time efficiency should be considered while computing the edit distance over human genomic data to find similar patients. Data owners are

not wiling to share their genomic data in plaintext to researchers to avoid re-identification of patients [12; 130] and legal consequences [123]. Proper authentication and access control over these high volume of sensitive genomic data are ensured with time costly verification methods which often results in delays by several months [97].

In this work, we propose a framework which captures these requirements by preserving the privacy of the query issued by a researcher and the genomic data owned by a data owner in a time efficient manner. In other words, our framework allows efficient approximation algorithm of string similarity over genomic data where the data owner cannot see the researcher's query and the researcher cannot access the genomic data of the data owner. The proposed framework consists of two algorithms of approximating the edit distance over genomic data. The first one resorts to the concept of shingles [131] supported by private set intersection techniques [132]. The second one depends on the banded alignment [133; 134] implemented using garbled circuits [35; 38]. The contributions of this article can be summarized as follows:

- We propose an approximation of the edit distance based on shingles [131] and the Permutation-based Hashing Set Intersection (Phasing) [132]. A k-shingle for a genomic sequence can be defined as any substring of length k that can be found within the sequence. Shingles are generated for the sequences of the data owner and the sequence of the researcher. Phasing is then used to privately intersect the shingles of the researcher and the shingles of the data owner such that the query and the genomic data are obscured from the data owner and the researcher, respectively.

- We propose another algorithm of approximating the edit distance the preserves the privacy of the query and the genomic data using the banded alignment and garbled circuits. The banded alignment approximates the edit distance by reducing the number of the needed comparisons. To privately execute the banded edit distance, we resort to garbled circuits.

- We experimentally show that the first approximation algorithm is time-efficient whereas the second one is more accurate using different datasets. We also show that the first approximation can be applied before the second one because they have complimentary properties. Moreover, we compare these approximations with state-of-the-art techniques [2]. Experimental results show that our proposed algorithms outperform existing techniques both in terms of efficiency and accuracy.

The rest of the chapter is organized as follows. Section 6.2 describes the research problem. The required background information is presented in Section 6.3. Section 6.4 details the proposed approximation algorithms. Practical results and discussions are presented in Section 6.5 and Section 6.6, respectively while Section 6.7 discusses related work.

6.2 Problem Definition

Similar Patients Query [2; 23] mainly uses edit distance as a metric to measure the similarity between different genomic sequences. It allows researchers or health care professionals to retrieve similar genomic sequences based on a query sequence. For example, a new patient gets admitted and the physician is seeking for previous patients with similar genomic sequences. The history of previous pa-

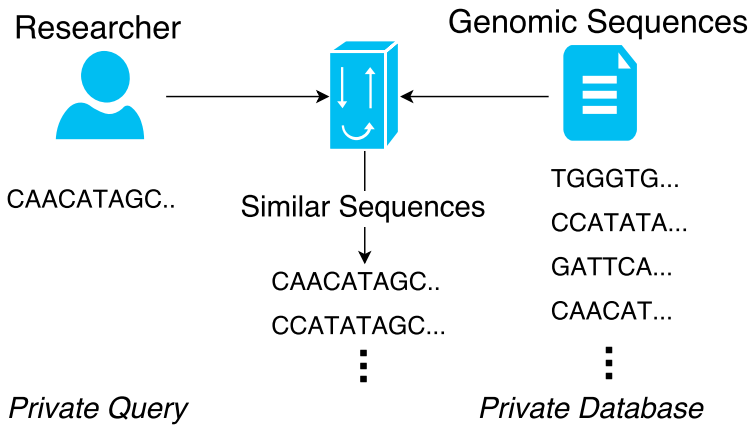


Figure 6.1: Problem architecture

tients will help the physician to come up with a definitive diagnosis in a timely manner.

The architecture of the proposed framework is shown in Figure 6.1. It consists of two entities: a researcher and a data owner (i.e., hospital or genomic data storage). The researcher is working on a new disease but cannot reveal the subject's sequence to the data owner and the data owner does want to disclose its genomic data to the researcher. Both parties need a protocol to interact with each other to determine similar sequences without disclosing the genomic sequences of their patients. The number of sequences revealed to the researcher through the private mechanism is predefined.

More formally, given a dataset of genomic sequences $GS = s_1, s_2, \ldots s_n$ owned by a data owner and a genomic sequence s_q provided by the researcher as a query predicate, the problem of similar patients query (SPQ) is to retrieve the top-k similar patients from GS, where the k sequences are determined according to the query sequence s_q and a similarity metric (i.e., edit distance). The retrieval should be conducted in a way such that the data owner cannot see s_q and the researcher

cannot access any sequence in S other than the final output (i.e., top-k sequences).

6.3 Preliminaries

In this section, we present an overview of the building blocks that are utilized in the proposed solution.

Edit Distance

A word over the finite alphabet Σ is a sequence $a_i, \ldots, a_n$ of symbols where $a_i \in \Sigma$ for $i = 1, \ldots, n$. The empty word is denoted by ϵ. An edit operation is a pair (a_i, b_i) with $a_i, b_i \in \Sigma \cup \{\epsilon\}$ and $a_i b_i \neq \epsilon$. The edit operation (a_i, b_i) is called an insertion if $a_i = \epsilon$, a deletion if $b_i = \epsilon$, and a substitution if $a_i \neq \epsilon \neq b_i$. An edit operation is a basic step in transforming a word into another word. The meaning of the operations (ϵ, b_i), (a_i, ϵ), and (a_i, b_i) is to insert b_i, to delete a_i, and to substitute a_i by b_i, respectively. A cost $c(a_i \rightarrow b_i)$ is assigned to each edit operation (a_i, b_i). It is generally assumed that $c(a_i \rightarrow b_i) = 1$ and $c(a_i \rightarrow b_i) = 0$ for $a_i \neq b_i$ and $a_i = b_i$, respectively. An edit sequence S is a sequence of edit operations, $S = ((a_1, b_1), \ldots, (a_n, b_n)), n \geq 1$. The cost of an edit sequence S is defined as $C(S) = \sum_{i=1}^{n} c(a_i, b_i)$

Definition 6.3.1. The edit distance $d(X, Y)$ between two words X, Y is defined as the minimum cost taken over all edit sequences that transform X into Y. That is $d(X, Y) = min\{C(s) | s$ is a sequence of edit operations transforming X into $Y\}$.

For example, Let us assume that X and Y are genomic sequences such that $X = ATGC$ and $Y = ATGG$. It takes one operation to convert X to Y. In other words, the edit distance is one. Wagner Fischer's algorithm to compute the edit distance is shown in Algorithm 4 [135].

Algorithm 4: Edit distance (Wagner Fischer's Algorithm [135])

Data: Sequence X and sequence Y

Result: Edit distance $d(X, Y)$ between two sequences X, Y

```
1  m ← length(X);
2  n ← length(Y);
3  set each element in d to zero;
4  for i ← 1 to m do
5      d[i, 0] ← i;
6  for j ← 1 to n do
7      d[0, j] ← j;
8  for j ← 1 to n do
9      for i ← 1 to m do
10         if X[i] = Y[j] then d[i, j] ← d[i − 1, j − 1];
11         else d[i, j] ←
12         Minimum (d[i − 1, j] + 1, d[i, j − 1]) + 1, d[i − 1, j − 1] + 1);
13 return d[m, n];
```

Threat Model

We adopt the semi-honest model where both parties follow the protocol but may try to deduce additional information from the received messages. A protocol is private in a semi-honest environment if the view of each party during the execution of the protocol can be effectively simulated by a probabilistic polynomial-time algorithm knowing only the input and the output of that party [136]. Many protocols involve the composition of privacy-preserving subprotocols in which all intermediate out-

puts from one subprotocol are inputs to the next subprotocol. These intermediate outputs are either simulated given the final output and the local input for each party or computed as random shares. Using the composition theorem [136], it can be shown that if each subprotocol is privacy-preserving, then the resulting composition is also privacy-preserving. The semi-honest model is a realistic adversary model in the context of this work where a level of trust among the parties can be ensured through a mutual legal agreement.

6.4 Edit Distance Approximations over Genomic Data

In this section we discuss two different techniques to approximate the edit distance over genomic data. These approximation algorithms are detailed in the following subsections.

6.4.1 Shingles with Private Set Intersection

The first approximation algorithm consists of two steps. The first step depends on the concept of shingles [131; 137] whereas the second one depends on the Private Set Intersection (PSI) [132]. These two steps are summarized in Algorithm 5.

Shingles. Shingling [131; 138] is a technique used to identify lexically similar documents in data mining. For any string S, a w-shingle is a set where each item is a substring of length w. These items can be unique or might appear more than once (bag technique). In this work, we only consider the unique property of the shingles.

Example. *Consider one genomic sequence 'CAACATAGCAAC' and* $w = 4$*, then the set of 4-shingles will be* $\{$*CAAC, AACA, ACAT, CATA, ATAG, TAGC, AGCA, GCAA*$\}$*.*

Algorithm 5: Shingling and PSI approximation

Data: Researcher's private query genomic sequence and data owner's dataset of genomic sequences which cannot be shared publicly

Result: Top-k similar genomic sequences by approximating the edit distance

1 The researcher creates w-shingles of the query sequence and notifies the data owner;

2 The data owner creates w-shingles for all the sequences in the dataset;

3 The data owner and researcher engage in a Private Set Intersection (PSI) [132] protocol to determine common shingles without revealing their data to the other party;

4 The data owner gets the final intersection result;

5 The data owner orders the records which have a higher number of matches and the top-k records are sent to the researcher;

Notice that '$CAAC$' appears twice in the sequence but only considered once when constructing the shingles. To the best of our knowledge, this is the first time this concept is used in privacy preserving computation of genomic data. It is particularly helpful for genomic sequences as we have only four nucleotides (A, T, G, C) to consider. In this step, the data owner and the researcher generate the w-shingles for the genomic sequences in the dataset and the genomic sequence in the query, respectively.

Private Set Intersection (PSI) It is a useful technique and is used in many real applications [132]. It addresses the problem of two parties who do not want to share their data but want to discover the common items between them. Formally,

Definition 6.4.1. Consider two different parties having two different sets A and B respectively. The output of a private set intersection only reveals the set $A \cap B = \{x : x \in A \wedge x \in B\}$ while A and B are kept private from each party.

In this step, we adopt state of the art Permutation-based Hashing Set Intersection (Phasing) [132] to privately intersect the shingles of the researcher and the shingles of the data owner generated in the first step. The data owner does not share its data or see the query sequence from the researcher. The data owner gets the result of the intersected shingles and orders the records according to the number of matches with the intersection result. For example, if record 1 has 10 shingles in the intersection set whereas record 2 has 9 singles, then record 1 is more similar to the query sequence than record 2. The data owner picks out the top-k and sends them to the researcher. The process is stated in Algorithm 5.

6.4.2 Banded Alignment Using Garbled Circuits

The second approximation algorithm depends on two concepts: the banded alignment [133] to compute the edit distance and garbled circuits [35; 38] to compute the banded edit distance in a privacy-preserving setting. The original Wagner Fischer's algorithm detailed in Algorithm 4 has an average case running time of $O(nm)$ where n is the number of sequences and m is the length of a genomic sequence. Since genomic sequences are generally long, running time $O(nm)$ is not scalable for human genomes [2]. We adopt in this step a banded alignment [133] to reduce the runtime from $O(nm)$ to $O(nb)$ where b is a constant (band length). As outlined in Algorithm 6, we only compare each nucleotide from sequence A with a certain region b in the second sequence. Algorithm 4 has to calculate through both of the whole sequences to find its score.

Algorithm 6: Banded edit distance

Data: Sequence X, sequence Y, and band length b

Result: b-banded alignment distance $d'(X, Y)$ between two sequences X, Y [133]

```
1  m ← length(X);
2  n ← length(Y);
3  set each element in d' to zero;
4  for i ← 1 to m do
5      d'[i, 0] ← i;
6  for j ← 1 to n do
7      d'[0, j] ← j;
8  for j ← 1 to n do
9      if j − b < 1 then lowest ← 1;
10     else lowest ← j − b;
11     if j + b > m then highest ← m;
12     else highest ← j + b;
13     for i ← lowest to highest do
14         if X[i] = Y[j] then d'[i, j] ← d'[i − 1, j − 1];
15         else d'[i, j] ← Minimum
               (d'[i − 1, j] + 1, d'[i, j − 1]) + 1, d'[i − 1, j − 1] + 1);
16 return d'[m, n];
```

To execute the banded edit distance detailed in Algorithm 6 in a private setting, we resort to garbled circuits [35; 37]. Due to privacy constraints, it is unwise to compare nucleotides at different positions using garbled circuits. The researcher

can exhaustively find out the corresponding value in any given respectable position. This is why the banded edit distance is implemented using a garbled circuit where the final output is the edit distance between the sequences (see Section 6.6 for more discussion).

Garbled circuits are expensive time-wise especially if the data owner owns a large number of records. To overcome this deficiency, we apply the banded edit distance using garbled circuits after shingling and PSI as both approximation algorithms have complimentary properties.

6.4.3 Joined Approach

In the joined approach, we use the first approximation (shingling and PSI) to reduce the search space for the second approximation as the first one is computationally much faster than the second one. Here the first method decreases the number of records that are used as an input to the second approximation by retrieving the top-t for a top-k query ($t > k$) as detailed in Figure 6.2. The relation between t (the top-t records resulted from the first approximation) and k (the top-k that should result from the second approximation) is $t = ck$ where c is a constant. The value of the constant c depends on the value of k, sequence length and dataset size. For example, if the dataset of the data owner contains 2000 records and the researcher is interested in the top-10 similar sequences, the first approximation reduces the number of records to 50 (c=5 and $t = 50$) and then we will use these 50 records as an input to the second approximation to end up with the top-10 similar records (k=10). If the value of k is changed to 20, t becomes 100 ($t = 5 \times 20$) records.

Notice that if the number of the records in the dataset of the data owner is not large or the sequence lengths are small, then there is no need for the first ap-

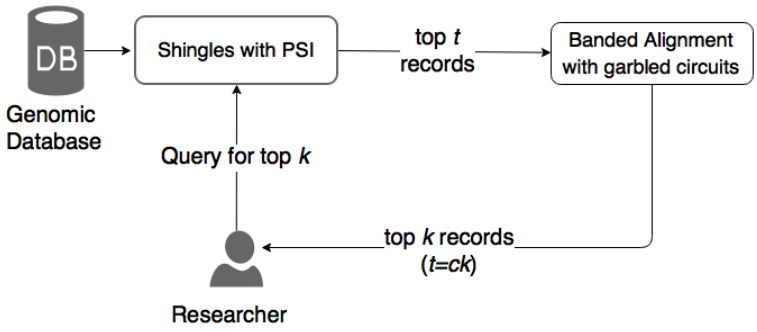


Figure 6.2: Execution order (second approximation)

proximation algorithm to decrease the number of the records as banded alignment over garbled circuit will be efficient enough. We further show these relations on Section 6.5. To get the top-k after that, Algorithm 6 depends on the garbled circuit to compare two sequences. After that, the data owner orders the records of the dataset according to the values of the edit distance and sends the top-k records to the researcher.

6.5 Experimental Results

In this section, we analyze the performance of the proposed approximation algorithms for privacy-preserving genomic data similarity problem. To simulate the real-life scenario, we placed the data owner and the researcher in a virtual machine with 4GB RAM. The reason behind this choice is that we are interested in computing the required time to securely execute both approximations with fixed network latency (5-10ms). The source code is available publicly at GitHub [139] for interested readers. We consider the following aspects in order to assess the efficiency of the proposed approximation algorithms.

- *Space complexity for shingles:* storage space needed to store the shingle dataset.

- *Accuracy analysis:* performance of the approximation algorithms measured against the original edit distance algorithm.
- *Runtime analysis:* time needed for preprocessing and to answer the researcher's query.
- *Benchmarking:* accuracy and time comparison with a state of the art technique [2].

We used both real-life and synthetic datasets for evaluating our model. The real-life dataset is taken from the recent iDASH competition 2016 [76] where there were approximately 3000-4000 different SNPs from 500 different individuals. For better analysis, we generated synthetic data by accumulating the allele frequency of CHB, CHS, JPT and MXL populations from *1000genomes* dataset (August 2010 Release) [140] and generated 2000 genomic sequences with around 9000 SNPs each. Corresponding details about the datasets are presented in Table 6.1. The query sequence length for Database 1 is (3465) and specified by the iDASH competition 2016 [76]. For Database 2, the query sequence length is (9000-10,000). Actually, the 50 query sequences were generated while generating Dataset 2. In other words, we generated 2050 sequences such that 50 were assigned for the query and the rest constructed the dataset. We will call the real-life dataset taken from iDASH2016 and the synthetic dataset generated from *1000genomes* Dataset1 and Dataset2, respectively throughout the rest of the discussion.

6.5.1 Space Complexity for Shingles

As transforming a genomic dataset to a shingle dataset will be space exhaustive, we need to analyze the space requirements for different shingle sizes. We only con-

Table 6.1: Dataset consideration

Parameters	**Dataset 1**	**Dataset 2**
Number of records (n)	500	2000
Sequence length (l)	3400-3500	9000-10000
Number of queries	1	50
Query length	3461	9000-10000
Data size (MB)	1.65	17.2
Data source	iDash 2016 [76]	Generated

sider unique strings when transforming the original genomic dataset to shingles dataset. For example, if there are n genomic sequences each with l length, then the size of the dataset is $n \times l$. If we consider fixed size w-shingles (i.e., $w = 5$) then we need to construct a $r \times w$ dataset where there are r unique shingles each with w length. For example, if $w = 2$, we have only 4^2 possible shingles ($AA, AT, AC, AG, \ldots$) since sequences are constructed with 4 nucleotides (*A, T, G, C*). This converts a $n \times l$ genomic dataset to a 16×2 shingle dataset. However this transformation will be expensive for large values of w.

Theoretically, the number of unique shingles (4^w) should grow exponentially as the size of w increases. Nevertheless, due to the high repetitions in genomic sequences, this quantity is much smaller for practical application scenarios. As shown in Table 6.2, if w =10 then we have $354,457$ shingles from a dataset of 3,000 records (9,000-10,000 length) where theoretically we should have $4^{10} = 1,048,576$. This results in a shingle dataset of 4.05 MB whereas the size of the original dataset with sequences was 17.718 MB. Larger values of w increases the size of the shingle

Table 6.2: Relationship between the shingle dataset size and the number of unique shingles for different shingle size (w)

Shingle Size w	**Unique Shingles**	**Shingle dataset Size** (MB)
5	1024	0.007
10	354,457	4.05
15	1,383,525	22.4
20	2,927,918	61.4

dataset as shown in Table 6.2.

6.5.2 Accuracy Analysis

As we are proposing two approximation algorithms, we analyze their accuracy separately and jointly. Here the accuracy is defined as,

$$accuracy = \frac{\text{\# of match in a top-k query}}{\text{\# of positives from edit distance}} = \frac{N_{TP}}{N_P} = \frac{N_{TP}}{N_{TP} + N_{FN}}$$

where TP, FN, P are true positives, false negatives and positives, respectively. In general, this accuracy (also known as true positive rate, sensitivity or recall) denotes how many records are positives for both the approximation algorithms and the original edit distance algorithm. For example in a top-*3* query, we have records $\{1000, 1010, 505, 1101\}$ as an output from the edit distance algorithm where the records 505 and 1101 have the same distance and ranked 3rd. Similarly, from our approximation algorithm, we have the rank as $\{1000, 1010, 505, 202\}$ which will

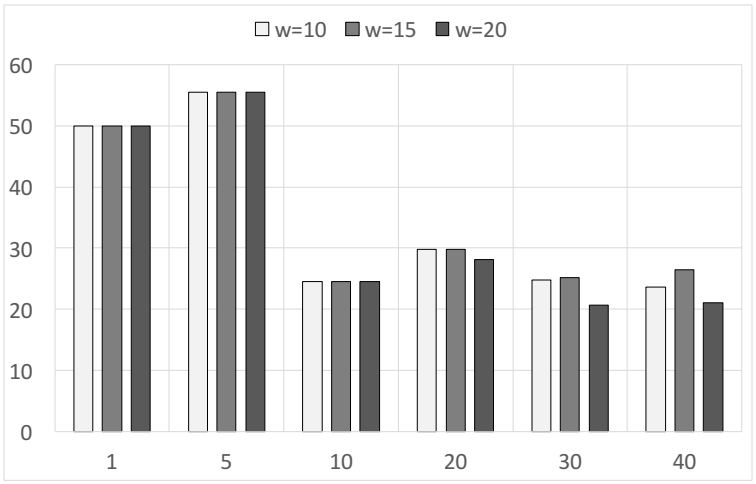


Figure 6.3: Accuracy of shingling and PSI approximation using Dataset 1. X-axis shows different k values (top-k) and Y-axis shows the accuracy for different w values

lead the accuracy to be $\frac{3}{4} = 75\%$. Some further analysis and explanation are available in the Appendix C as well.

The first approximation algorithm using shingles and PSI is much accurate when the dataset is small (i.e., Dataset 1). While for larger datasets, this method falls short and we need the banded alignment algorithm to obtain good accuracy. They can also be used in conjunction or jointly. In Figure 6.3, we depict the accuracy of the shingling and PSI approximation using Dataset 1. The dataset had 500 records and short sequences (around 3000 each). The performance of the proposed approximation (shingles and PSI) algorithm is measured against the original edit distance algorithm. The optimal value of w is ($log_{|\Sigma|}l$) [134] where l is the sequence length and $|\Sigma| = 4$ for genomic sequences. However, as lower w values resulted in higher false positives, those are not showed here for brevity. However, more graphs are shown in the Appendix C regarding this issue.

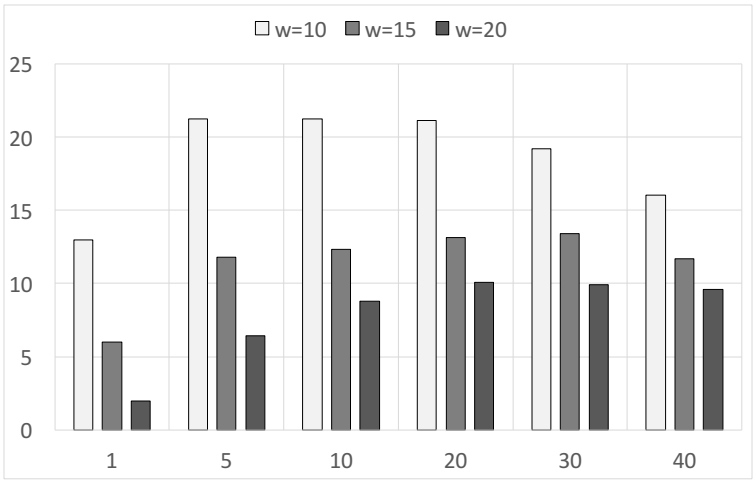


Figure 6.4: Accuracy of shingling and PSI approximation using Dataset 2. X-axis shows different k values (top-k) and Y-axis shows the accuracy for different w values

Aforementioned, this method has some shortcomings when dealing with Dataset 2 where we have longer sequences. In Figure 6.4 we show this deficiency as the accuracy ranges in $2 - 13\%$ for top-1 queries. This is due to the higher sequence lengths and numbers as shingle matches cannot efficiently represent the original edit distance.

Due to this deficiency from shingles and PSI approximation algorithm, we switch the other technique to approximate edit distance which is more accurate for longer sequences. The accuracy of our banded alignment is showed in Figure 6.5. The accuracy of this method is impeccable due to the resemblance with the original edit distance and lower dimension of data. However there is a certain cost involved in executing the banded alignment over a garbled circuit (to provide security) which results in longer run times. We further elaborate this notion in the upcoming run time analysis section.

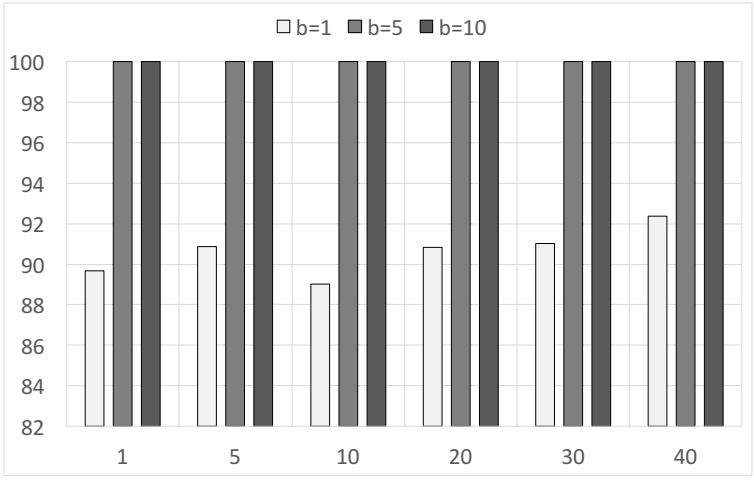


Figure 6.5: Accuracy of the banded alignment using Dataset 2. X-axis shows different k values (top-k) and Y-axis shows the accuracy for different band values b values

Figure 6.6 shows the accuracy for both approximation algorithms joined according to Figure 6.2. Here, we use the top-t outputs from the first approximation algorithm as an input to the banded alignment to end up with the target top-k results. It is clear from Figure 6.6 that larger values of t returns better accuracy. On the other hand, larger values of t has a negative impact on the running time as demonstrated in the subsequent section. The accuracy in Figure 6.5 is certainly better than the accuracy in Figure 6.6. However, we proposed the combining of both approximations as shown in Figure 6.2 because this reduces the execution time. We have further discussed these issues in the Appendix C.1.

6.5.3 Runtime Analysis

We show in Table 6.3 a summary of the running time of both private approximation algorithms along with other insecure techniques. We also give the time

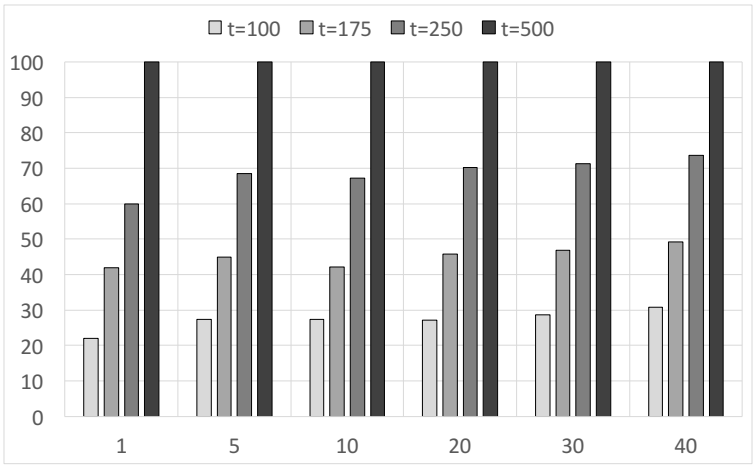


Figure 6.6: Accuracy of the banded alignment after shingles and PSI method using Dataset 2. X-axis shows different k values (top-k) and Y-axis shows the accuracy for different t values

required for the state of the art work conducted by Wang *et al.* [2] as it provides a solid benchmark for assessing runtime. The benchmarking is done using Dataset 2. The first approximation (shingling and PSI) is the fastest. The banded alignment takes longer since it depends on the sequence length of the query due to the runtime $O(nb)$.

This concern can be further elaborated in Figure 6.7 where we show the run time for both approximation algorithms with different band sizes and c. The noticeable aspect of Figure 6.7 is that the running time of the joined approach has a linear relationship with the value of k. If k is increased, t (the input of the banded alignment ($t = ck$)) will be increased and accordingly the running time will be increased. This is the primary reason behind using the shingle approach before the banded alignment in the joined approach as it reduces the search space in a constant time for the banded alignment for a large dataset. Also, it is clear that the preprocessing time is a one time cost and depends on the genomic database size

Table 6.3: Running time analysis (top-10 queries with $k = 10$, $c = 5$ ($t = ck$), $w = 10$, and $b = 5$)

Dataset	**Method**	**Preprocessing Time (s)**	**Query Time (s)**
Dataset 1	Plain Edit Distance	0	23
Dataset 1	Shingles with PSI	18	5
Dataset 1	Protocol 1 [2]	5.7	585
Dataset 1	Protocol 2 [2]	5.7	511
Dataset 2	Plain Edit distance	0	930
Dataset 2	Protocol 1 [2]	61	3049
Dataset 2	Protocol 2 [2]	61	2800
Dataset 2	Shingles with PSI	181	108
Dataset 2	Shingles with PSI + banded alignment	181	730

which we can neglect. However, the banded alignment over garbled circuit can be much faster under some security assumptions which we explain in 6.6.2. The run time for Dataset 1 is provided in the Appendix C.11.

6.5.4 Benchmarking

In Figure 6.8 and Figure 6.9 we show the performance of the state of the art approximation algorithm proposed by Wang *et al.* [2] using Dataset 1 and Dataset 2, respectively. Both protocols presented in the paper of Wang *et al.* have high accuracy using Dataset 1 for top-$\{1, 5, 10, 20, 30, 40\}$ queries which resemble our

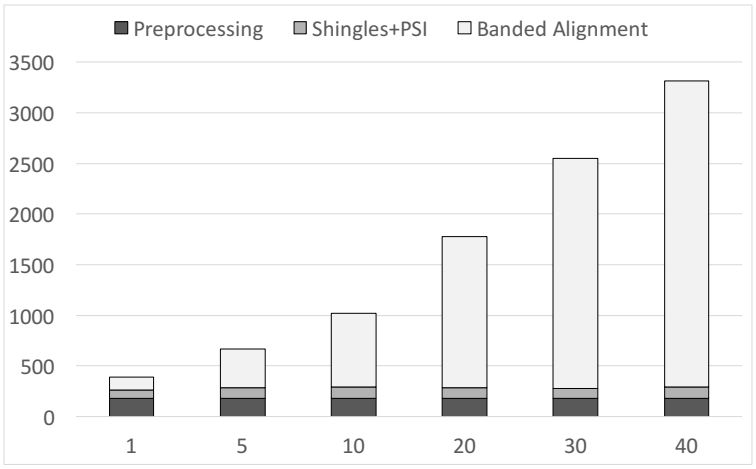


Figure 6.7: Run time analysis using Dataset 2 X-axis shows different k values (top-k) and Y-axis shows the run time (in seconds) for different approximations where $b = 5$ and $c = 5$

PSI and shingle based approach. It is noteworthy that we take much less time to achieve similar accuracy ($24s$ vs $585s$). However, this accuracy drops for longer sequences as shown in Figure 6.9 for Dataset 2. This clearly shows the benefit of our second approximation algorithm using the banded alignment technique. Thus, our joined approach achieves a good balance between accuracy and runtime.

6.6 Security Discussions

In this section, we elaborate some of our design choices and discuss the limitations of the proposed methods.

6.6.1 Security of Private Set Intersection Methods

In addition to Phasing algorithm [132], there are a number of other private set intersection techniques [141]. Among these, we experimentally evaluated the basic

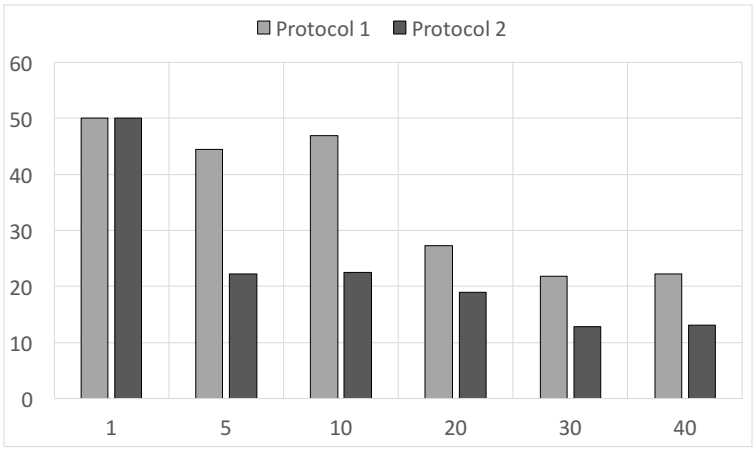


Figure 6.8: Accuracy of Protocol1 and Protocol2 [2] using Dataset 1. X-axis shows different k values (top-k) and Y-axis shows the accuracy for both protocols

hashing based method [142], Diffie-Hellman based protocol [143] and permutation based hashing method [132]. We found that only the hashing based method has a better performance than the Phasing algorithm. We did not opt for the hashing method [132] because an active adversary can run a brute force algorithm on a specific shingle size (w). This will eventually reveal the query sequence (or genomic data) as the data owner (or the researcher) can reconstruct the sequence from shingles. Therefore, we use the Phasing algorithm [132] where such attack is not possible.

6.6.2 Banded Alignment in Garbled Circuit

In the banded alignment, we implemented the whole algorithm using a garbled circuit (GC). This design choice is due to the leakage consideration of individual position comparisons of the edit distance algorithm. In the original edit distance algorithm, characters are matched one at a time at different positions of the sequences of the query and the dataset. If a researcher is allowed to query the

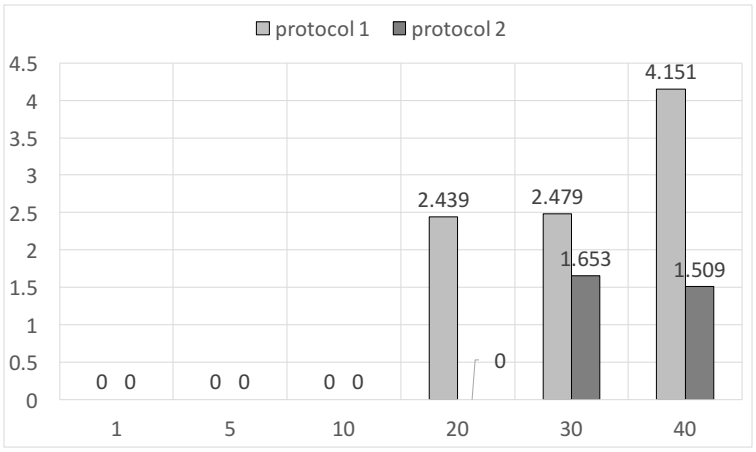


Figure 6.9: Accuracy of Protocol1 and Protocol2 [2] using Dataset 2

genomic dataset and individual comparisons are done using a GC, then s/he can exhaustively find out the corresponding value in any given respectable position as there are only 4 possible values (A, T, G, C). However, in our method, the whole iteration of the computation is done inside a garbled circuit and it outputs only the final result of the edit distance. Thus, our banded alignment protocol allows a researcher and a data owner to obliviously calculate the distance between two strings without leaking any further information.

6.6.3 Joined Approximation

The output of the joined approximation is the top-k sequences given a target query and these k sequences are public and the output of the protocol. In the joined approximation, the data owner knows the t records which are the output of the shingling and PSI. As these t records are not revealed to the researcher, it does not violate the security requirement. Also, it does not reveal any additional information to the data owner as t sequences are more general than the final k sequences.

6.7 Related work

One of the primary works in the domain of privacy preserving genomic sequence similarity is conducted by Jha *et al.* [144]. In their paper, they showed three different protocols which can replicate the original edit distance algorithm over a garbled circuit. However, due to the performance of the garbled circuit available that time, it took around 40 seconds for computing the edit distance between two sequences where the length of each one of them is 25. After the proposal of the fully homomorphic encryption (FHE) by Gentry [145], edit distance was also proposed to be homomorphically computed via lattice encryption by Cheon *et al.* [101]. However, due to the current state of FHE, the scheme is still inefficient as it takes 16.4 seconds to compute a 8×8 block of dynamic programming. As the crypto behind the FHE advances and improves, we might see a better usage of this in the future.

The closest research to ours is conducted by Wang *et al.* [2]. The research addressed the problem of approximating the original edit distance in a realistic setting. The method used a public reference genomic sequence to compute an approximation of the edit distance between two strings. However, the selection of a public reference leaks some information about the underlying data distribution. Moreover, it affects the accuracy as the computation is done according to a reference. In a more recent work, Shimizu *et al.* [146] proposed the usage of Burrows-Wheeler transformation for finding target queries on a genomic dataset. The problem addressed in this paper [146] is different, although closely related, as it does not answer secure string similarity for genomic data. There are also some other related studies such as [147; 148; 149] which address approximating

Table 6.4: Chronological development of different methods.

Authors	**Year**	**Data ($n \times m$)**	**Time (s)**	**Principal Method**
Jha *et al.* [144]	2008	25×25	< 40	Smith-Waterman technique
Wang *et al.* [149]	2009	400×400	28.5	Custom protocols
Wang *et al.* [2]	2015	2000×9000	2800	Private set difference with a reference sequence
Cheon *et al.* [101]	2015	8×8	16.4	Homomorphic encryption
Shimzu *et al.* [146]	2016	2184 genomes	4-10	Burrows-Wheeler transform

or securely computing edit distance. Some of these are summarized in Table 6.4.

6.8 Future Work

One of the interesting research question prevailed from this work is the length of the genomic data in approximating edit distance. Though theoretically reducing the run time of edit distance is still under the lens from researchers [127], the secure approximation of this similarity algorithm can utilize the data distribution as shown in the aforementioned methods. Also as the next generation sequencing data [150] are already in motion increasing the size of genomic data each day, we need faster adhoc methods for securely computing similarity representing edit distance.

Chapter 7

Conclusion

In this thesis, we have proposed different privacy-preserving techniques to be applied on genomic data in the setting of distributed data storage or federated model (Chapter 3 and 4). We also addressed two problems where data are stored in a central repository and specific computations are allowed (i.e., count query, edit distance) on Chapter 5 and 6.

7.1 Summary

Firstly, we proposed an approach to securely compute the count and the ranked query over genomic data in an efficient way. The major contributions of this work are the data being owned by different data owners and still providing an efficient computation under a security guarantee. The computation time of the secure computation is much closer to the time of the corresponding regular computation over the plaintext and the final output does not reveal the individual contributions from the data owners.

Additionally, we analyzed Bustamante attack and provided a method to calcu-

late the risk involved in sharing the genomic data. Even in this architecture, the genomic data are owned by different data owners and they only provide binary results. We proposed two simple privacy preserving solutions: eliminating random positions and biased randomized responses which mitigate the re-identification attack. Our lightweight privacy preserving solutions ensure a good trade-off between data privacy and utility. Experimental results demonstrate that given a higher bias, both solutions are able to provide high data utility to a researcher or user.

In the second part of the thesis, we have also proposed a model for outsourcing genomic data to a public cloud or a central repository. The proposed model ensures the security of outsourced data by permutating and adding fake records. Here two kinds of queries (count query and top-k) are considered on the outsourced data which can be extended to more complex queries. We have demonstrated that storing genomic data using the proposed method does not reveal the identity of an individual and mitigates the risk of the association between genome sequence and sensitive data. We conducted extensive experiments and implemented the model using a public cloud (Amazon EC2). Experimental results on real-life and synthetic data demonstrate that the proposed model outperforms the existing technique in terms of query execution time and its performance is close to the insecure method.

Finally, we studied the problem of approximate edit distance and secure execution methods on genomic data. Securely computing edit distance between human genomes have become very important in medical and public health domains. We proposed novel techniques to privately approximate the edit distance on human genomes. We implemented these techniques and experimental results show that the proposed methods are accurate and time-efficient, and performs better than

existing methods.

To summarize, the main contributions of this thesis is the efficient, secure and privacy preserving computation of genomic data considering different functions and frameworks.

7.2 Looking Ahead

Even to this day, there are no single solution which can be availed for every privacy issue of genomic data. Future directions appear in different privacy techniques or models on next generation genome sequencing data [150] which comes with higher storage and computation time. Regardless of all these technical answers available (and to appear) on the questions of privacy and security of genomic data, there is an acute demand for an update in the current policies around genomic data usage. As these newer privacy preserving techniques can only be enforced by laws and policies, these should be considered to facilitate the sharing of data for genomic research.

Appendix A

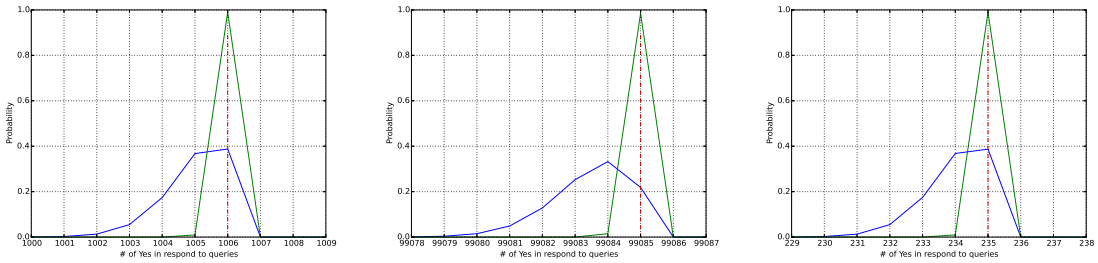
Appendix: Aftermath of Bustamante Attack on Genomic Beacon Service

A.1 Risk Analysis on other datasets

The null hypothesis vs alternative hypothesis for three different datasets (SSMP [73], GoNL [72], 1k Genomes Phase 1 Affymetrix) are given in Figure A.1.

A.2 Further Experiments

In this section we show further experiments on some of the other parameters of the attack which we did not provide on the main paper.



GoNL 1k Genomes SSMP

Figure A.1: The green line represents the distribution of the query individual being in beacon database while the blue one represents them not being there. The red line denotes the t'_α.

A.2.1 Effect of genome sequencing error

The effect of genome sequencing errors effect the attack which is depicted in Figure A.2. This effect was not present on the original paper [3] which we show here. According to them, we also considered three different sequencing errors ($1e-06, 0.001, 0.01$) where $1e-06$ is the ideal case with very less errors. Further explanations are given at Table S2 [3].

A.2.2 ROC curve for the LRT

The receiver operating characteristic (ROC) curve (False Positive Power vs LRT Power) are shown in Figure A.5. The false positive rates are different assumptions from the adversary. As its depicted that for different mismatch rate, δ and false positive rates (0.05 to 0.5) the LRT powers changes. Our proposed method 2 with bias 90 restricts the LRT power to be bounded and close to 0 while the original attack the individual is identified (power 1) in just after 5000 queries.

A.2.3 False Positive Rate vs LRT Power

In Figure A.3 we show the effect of our method and different false positive assumption from the adversary. For example in Figure A.3A LRT powers on Y-axis are plotted with respect to different false positive rates ($0.005, 0.01, 0.025, 0.05$). As its clear that if the adversary assumed 2.5% false positive results or more from the beacon, still it can identify individuals as the LRT powers are 1. While in our proposed methods even with assumptions of 50% false positives, the adversary is inconclusive about the presence. It is noteworthy that Figure A.3A shows this effect for 5,000 queries where our method (randomized response with 90 bias, 98% accuracy) achieves that on 125,000 queries.

A.2.4 Relative re-identification

Figure A.4A shows the identification of the relatives in the attack scenario [3] for different number of queries. It also contains the effect of our proposed solution (Randomized response with bias 90). The underlying dataset for this experiment has the original parameters of [3].

It is clear from the figure A.4B that identifying relatives of an individual is inconclusive even after 100,000 queries from the adversary where s/he can have similar LRT powers after 10,000 queries.

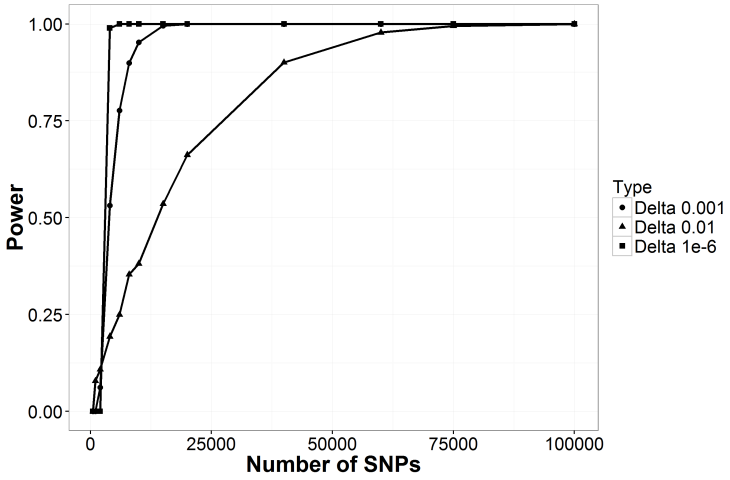
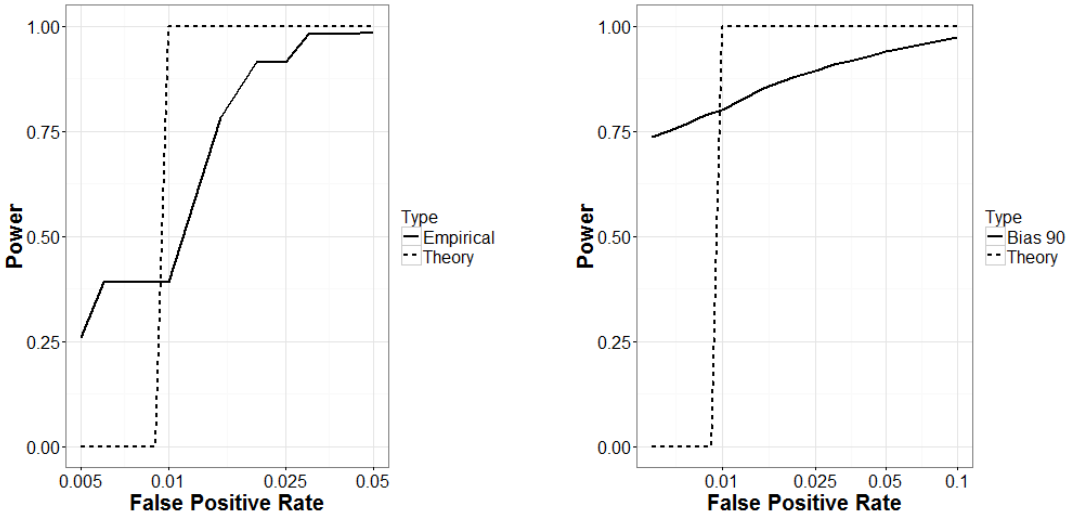
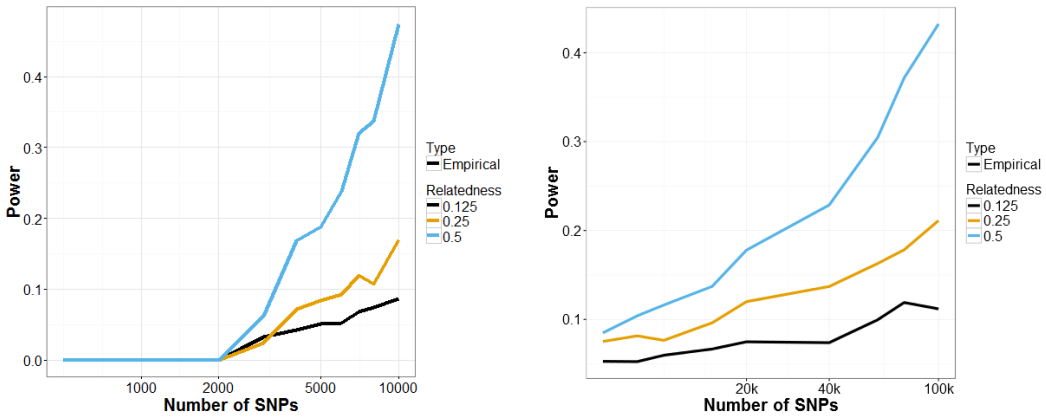


Figure A.2: Power(LRT) of re-identification attacks of individuals on beacons constructed with 1,000 individuals and different sequencing error rate ($\delta = \{1e-06, 0.001, 0.01\}$) assumptions



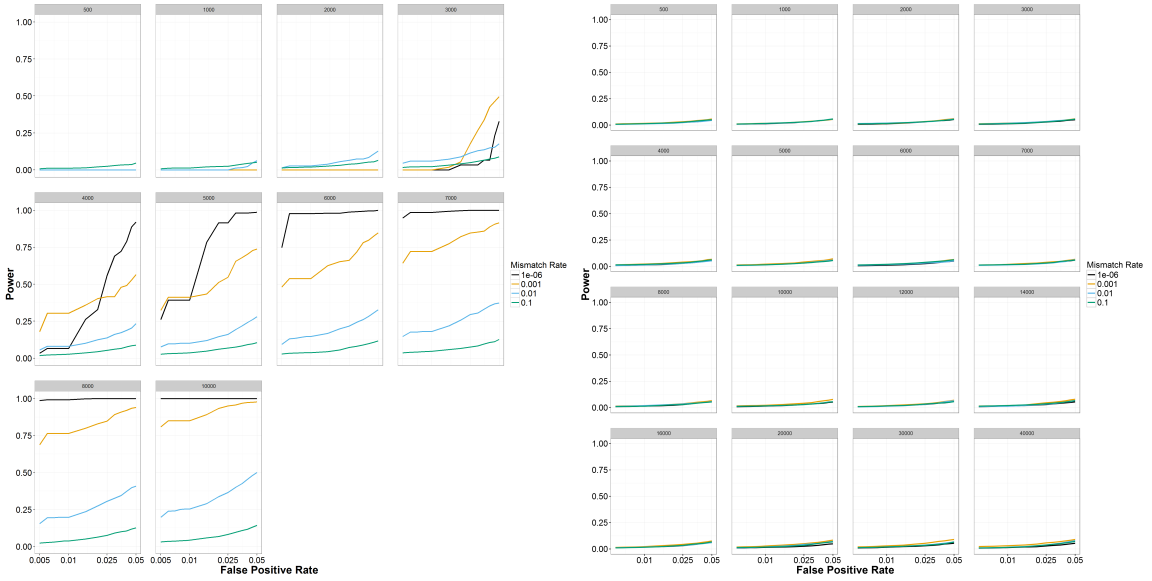
Original results after 5k queries Randomized response after 125k queries

Figure A.3: False Positive Rate vs LRT Power shows the difference made by our methods on the LRT power on different false positive rates



Original results Randomized response

Figure A.4: Effect of randomized response (with bias 90) on identifying relatives in beacon service where X-axis denotes the number of queries required to achieve the Power which is showed in Y-Axis



Original Results Randomized Response

Figure A.5: ROC curve of the Log Likelihood Ratio test (LRT) for different error rates. In the original attack LRT powers vary with the increasing errors while in our results its mostly static and close to 0. Different panels show different number of SNPs queried from 500 to 40,000 and x-axis is in logarithmic scale.

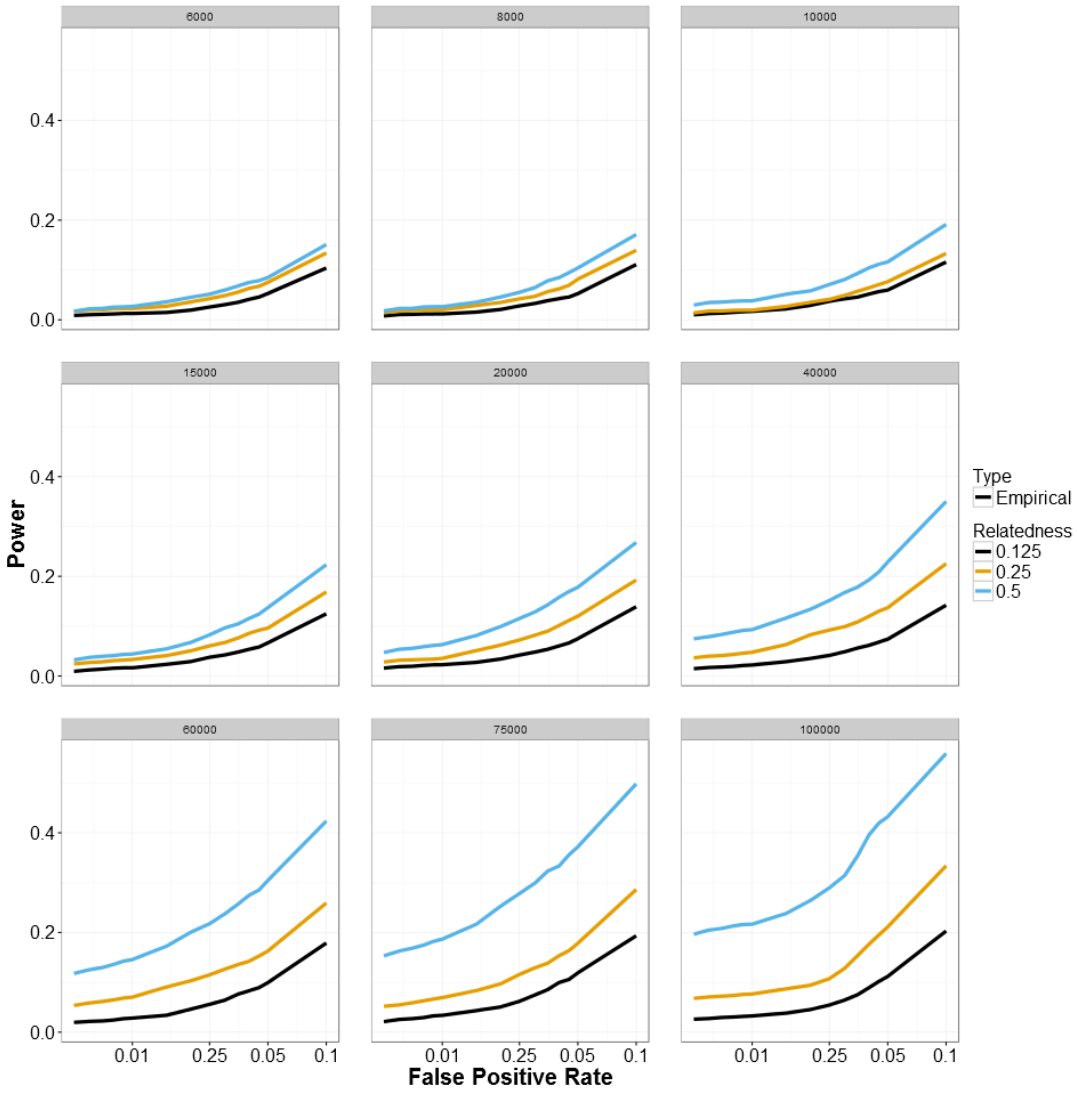


Figure A.6: Effect of randomized response (with bias 90) on identifying relatives with different false positive assumptions (X-axis) from adversary in beacon service. Y-Axis denotes the LRT powers

Appendix B

Appendix: Private and Efficient Query Processing on Outsourced Genomic Databases

B.1 More on valid permutation

We define permutation according to abstract algebra as, 'any permutation is a product of transpositions' [151]. Transpositions (pairwise permutation) are a kind of specific permutation in which only two positions are permuted. For example, suppose we have three columns a, b and c which have same letter frequency. In the following table, all permutations in product of transpositions are shown.

Notice that to achieve $(b\ c\ a)$, $(1,2)(2,3)$ means first swap the first column with the second which results in $(b\ a\ c)$ then swap second and third column which causes $(b\ c\ a)$ position of columns.

The valid permutation swaps columns with same nucleotide set having same

Possible Column Positions	Permutation in Transposition Form
a b c	Requires Nothing
a c b	(2,3)
b a c	(1,2)
c b a	(1,3)
b c a	(1,2) (2,3)
...	...

frequency. This helps to thwart attacks based on adversary's background knowledge. However, if we swap two columns with different frequencies then an adversary might be able to undo the permutation due to the available statistical information regarding SNPs (e.g., public information available from 1000 genomes project).

For example, in Table 1 (from the main paper), if we swap column 1 and 3 then an adversary may infer these two columns because they have different frequencies and their frequencies are available in 1000 genomes project.

B.2 Storage requirement for multiple phenotypes

Genomic SNP data sets usually contain thousands of SNPs in each record and these SNPs are associated with multiple diseases. Our proposed method uses one disease which could be useful as a proof of concept for multiple disease types. However, the addition of fake records (as proposed) will lead to an exponential increase in the size of the database as more disease types are added. In particular, for n phenotypes, $(2^n - 1)$ additional fake records need to be added to provide the

privacy guarantee.

There is a direct relationship between the storage requirement (i.e., addition of fake records) and privacy guarantee. Strong privacy requirement requires larger storage space. For example, existing crypto-based solution has an expansion of at least 1024 times. Compared to existing solutions, our solution is still more efficient for less than 10 phenotypes (i.e., $2^{10} = 1024$). However, the storage requirement (to provide stronger privacy) grows exponentially as the number of phenotypes increases. This concern can be easily addressed if we relax our privacy guarantee. Currently, our solution provides a strong privacy guarantee: 'an adversary cannot guess better than random the value of the sensitive attribute'.

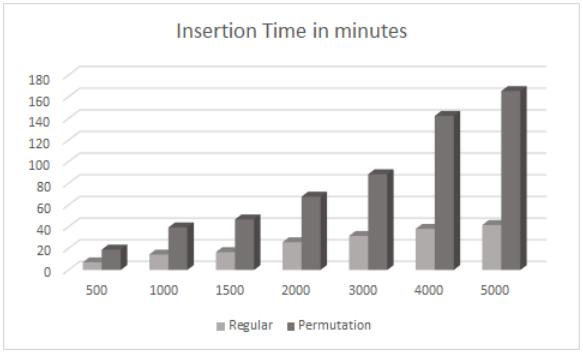
If we relax this requirement, and adopt other standard privacy notions, such as l-diversity [111], then the proposed method does not need to add exponential number of fake records for multiple diseases. Informally, l-diversity requires that there are at least l distinct sensitive values for each genomic sequence.

Moreover, we would like to stress that our proposed framework is still very efficient for small number of phenotypes compared to other crypto-based solutions (expansion factor is 1024 times). In real-life applications, there are many genomic data repositories that contain datasets with few diseases such as CGHub[1], EGA[2], COSMIC [3], and CPRG. Hence, the proposed model will facilitate privacy-preserving genomic analysis for these datasets. For example, doctors are interested in mining the potential biomarkers for Kawasaki disease [152] using datasets that has only one phenotype. Our proposed method offers a solution to analyze the data using the cloud, which would be difficult other wise due to the privacy concerns.

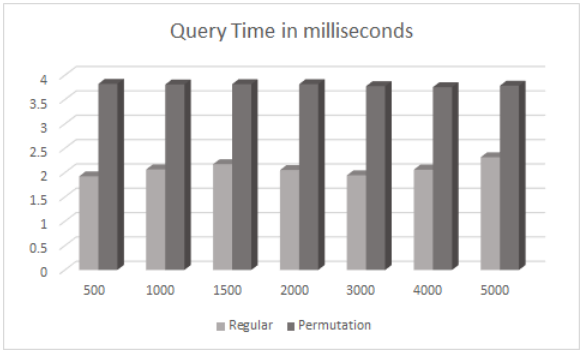
[1] cghub.ucsc.edu
[2] www.ebi.ac.uk/ega/home
[3] cancer.sanger.ac.uk/cosmic

B.3 Variable size SNPs analysis

SNP datasets usually contain hundreds to thousands of SNPs and the proposed method should be scalable as the number of SNPs increases. Following we report the insertion and count query time by varying the the number of SNPs while keeping the number of rows fixed (1000 rows).



Insertion time



Count Query time (logarithmic)

Figure B.1: Insertion and count query time based on different size of SNPs for $1,000$ records

Figure B.1 shows that as the SNPs size increases, the insertion time also increases for both regular and permutated genomic data. It is because more data causes more network communication and processing time of the proxy. However, for the count query, the query execution time does not vary that much. It is because the runtime for executing a query is a function of the number of predicates in the query (query size) and the number of records in the database. The number of SNPs in the genomic sequences does not play an important role because database only looks at the determined locations in the query to find the matches and ignores all other locations. Thus the time required is nearly constant and only depends on network latency.

B.4 Proof of Theorem 1

We now provide the proof for Theorem 1 and 2. The security definitions were

Definition B.4.1. We consider a genomic dataset (DB) secure if,

- It does not reveal the original genome sequence.
- Given a victim's genome sequence g_{victim}, an adversary $\mathcal{A}$ after observing the DB cannot guess better than random ($1/2$) the value of the sensitive attribute.

$$|Pr(\mathcal{A}\ guess\ g_{victim}\ disease|\mathcal{A}\ observed\ DB) - \leq 1/2| \leq \epsilon$$

where ϵ is a negligible function of the security parameter.

Theorem 4. *Given a permuted genomic database DB and for any two possible valid permutation keys π_1, $\pi_2 \in \mathcal{VP}$, we get,*

$$Pr(\pi_{key} = \pi_1|DB) = Pr(\pi_{key} = \pi_2|DB)$$

where π_{key} is the chosen key for this permuted database DB.

Proof. We assume that the columns of genome sequences are independently distributed [3]. Therefore, the only background knowledge that an adversary have is columns' statistical features (e.g., first column has 'A' and 'T' with half probability of appearance for each). If we swap two columns in the dataset with same statistical properties and show the result to an adversary, it cannot determine the permuted columns because in this case, the permuted table statistical features completely conforms the adversary's background knowledge. Therefore, observing the permuted database does not change the distribution of choosing the permuted key in view of the adversary. Adding this fact to the fact that the distribution

of choosing permutation follows uniform distribution (see Section 5.3 of paper) hence,

$$Pr(\pi_{key} = \pi_1 | O.DB) = Pr(\pi_{key} = \pi_2 | O.DB)$$

■

It is because undoing a valid permutation is not possible using only the frequency analysis. For example, suppose the first column is consisted of 'A' and 'T' with equal probability. Hence, all other columns with the same frequency can be swapped with the first column and the adversary is unable to determine the right column for swapping. However, with better statistical knowledge such as Linage Disequilibrium between two positions, it might be possible to infer the permutation key which is beyond the scope of this article.

B.5 Proof of Theorem 2

Second aim in security of outsourcing genomic data models is removing any association between DNAs and cancer status. We will demonstrate that observing a DB does not deviate guessing of a patient by any adversary.

Theorem 5. *Suppose a genomic sequence g is given. Any adversary $\mathcal{A}$ after observing DB cannot guess better than a random guess about g's disease.*

Proof. Suppose π_{key} is the chosen random valid permutation. When we give the information that g is within the database it restricts the possible valid permutations to an subset $I = \{\pi \in \mathcal{VP} |\ \pi(g) \in\ DB\}$. We have,

$$Pr(g\ has\ cancer) = Pr(\pi_{key}(g)\ has\ cancer | O.DB)$$

According to Bayes' law we have,

$$I_{yes} = \{\pi \in I | \pi(g)\ in\ DB\ has\ Cancer\}$$

$$I_{no} = \{\pi \in I | \pi(g)\ in\ DB\ has\ not\ Cancer\}$$

where $I = I_{yes} \cup I_{no}$. If $\pi \in I_{yes}$ then $Pr(\pi(g)\ has\ cancer) = 1$. Similarly, if $\pi \in I_{no}$ then $Pr(\pi(g)\ has\ cancer) = 0$. Therefore,

$$\sum_{\pi \in I} Pr(\pi(g)\ has\ cancer).Pr(\pi_{key} = \pi | O.DB) =$$

$$\sum_{\pi \in I_{yes} \cup I_{no}} Pr(\pi(g)\ has\ cancer).Pr(\pi_{key} = \pi | O.DB) =$$

$$\sum_{\pi \in I_{yes}} Pr(\pi_{key} = \pi | O.DB)$$

At the end we only need to prove that $\sum_{\pi \in I_{yes}} Pr(\pi_{key} = \pi | O.DB)$ is equal to $1/2$. According to Theorem 1, in view of the adversary the probability of choosing permutation key is uniformly distributed over the possible valid permutation, say I, set hence,

$$\begin{aligned} \forall \pi \in I\ Pr(\pi_{key} = \pi | O.DB) = \frac{1}{|I|} &= \frac{1}{|I_{yes} \cup I_{no}|} \\ &= \frac{1}{|I_{yes}| + |I_{no}|} \end{aligned}$$

Therefore,

$$\begin{aligned} \sum_{\pi \in I_{yes}} Pr(\pi_{key} = \pi | O.DB) &= \sum_{\pi \in I_{yes}} \frac{1}{|I_{yes}| + |I_{no}|} \\ &= \frac{|I_{yes}|}{|I_{yes}| + |I_{no}|} \end{aligned}$$

In addition, according to the model that family members are balanced therefore $|I_{yes}| = |I_{no}|$ and consequently $\frac{|I_{yes}|}{|I_{yes}| + |I_{no}|} = \frac{1}{2}$. ■

Appendix C

Appendix: Secure Approximation of Edit Distance on Genomic Data

C.1 Tradeoff in Joined Approach

There is a tradeoff between accuracy and execution time while considering the joined approximation approach. Shingle-based approximation is computationally inexpensive while the banded alignment using Garbled Circuit is more accurate. This is why the parameter t is utilized to bound this tradeoff as higher values of t will result in more accurate but slower execution. However for smaller values of t after the first approximation, the search is only limited to the top-t using the banded alignment. The records, which are similar but do not show up in the top-t, are not considered and cause the false negative results.

Further Accuracy Analysis

Some of the analysis on different parameters such as band size, shingle length, different values for c are provided here as well as their effect on different metric of accuracy. For further analysis about the accuracy of the proposed approximations, we describe some notations used in the context of similarity search:

- True Positive (TP): records which are similar and also returned as similar by the approximation algorithm.
- False Positive (FP): records which are not similar but returned as similar by the approximation algorithm.
- True Negative (TN): records which are not similar and also are not returned as similar by the approximation algorithm.
- False Negative (FN): records which are similar but are not returned as similar by the approximation algorithm.

We will also need some metrics such as:

$$Recall = \frac{N_{TP}}{N_{TP} + N_{FN}}$$
$$Precision = \frac{N_{TP}}{N_{TP} + N_{FP}}$$
$$Fallout = \frac{N_{FP}}{N_{FP} + N_{TN}}$$

As we showed the recall values or true positive rates for propose algorithms along with the benchmarking, here we are more interested in the precision and the false positive rate (fallout rate).

C.2 Recall Analysis

As we provided some Figures about the recall (accuracy) or True Positive Rates (TPR) in the main paper, here we show some more which may appear missing in the main paper. The true positive rates for the banded alignment and the joined approximation on Dataset 1 would be the point of interest here. In Figure C.1 and Figure C.2 we show the high true positive rates for the banded alignment and the joined approximation (banded alignment after PSI-Shingle approach) using Dataset 1, respectively. Figure C.11 shows the true positive rates of shingling and PSI approximation using Dataset 1 for different low w values ($w = \{4, 5, 6\}$).

C.3 Precision Analysis

Precision denotes that a sequence which is returned as an output from the approximation algorithm is actually a similar one. In other words, higher precision will indicate better performance as this is one of the general requirement of any algorithm. We show the precision of the PSI-Shingle approach using Database 1 and Database 2 in Figure C.3 and Figure C.4, respectively. The PSI-Shingle approach has less precision using Dataset 2; however, it outperforms Wang *et al.*'s Protocol1. In Figure C.5, we show the precision of the banded alignment using Dataset 2. From the Figures, we can see that the PSI-Shingle and the banded alignment perform well using Dataset 1 and Dataset 2, respectively.

C.4 False Positive Rate or Fallout Analysis

In a ranked query, false positives are those records which do not belong to the top-k results but appear as a result. For example, suppose a genomic sequence is not ranked among the top-10 nearest sequences according to the edit distance metric. However, our approximation methods output that record in the top-10 allowing this to be treated as a false positive. This analysis is much useful for understanding the real accuracy of any approximation method as it reveals the number of wrong answers that are included in the result of an approximation.

In Figure C.6 we show the false positive rate ($\frac{N_{FP}}{N_{FP+TN}}$) of the first approximation using Dataset 1 where N_{FP} represents the number of false positives and N_{TP} represents the number of true positives. It shows that for a small value of w (i.e., $w = 5$), there are many unwanted false positives. For $w = \{10, 15, 20\}$ the false positives are rather less in quantity which is desirable. Figure C.7 shows the perfect FPR which is expected for the banded alignment using Dataset 2. For $b = \{1, 5, 10\}$ the FPR is 0 for any top-k query. Figure C.8 shows the FPR for the joined approximation. Figure C.12 shows the fallout of shingling and PSI approximation using Dataset 1 for different low w values ($w = \{4, 5, 6\}$).

C.5 Run Time Analysis

We analyze the running time for Dataset 1 in Figure C.9. Here the effect of using the banded alignment can be further explained. As we picked $c = 5$, a top-5 query ($k = 5$) will require the top-25 records ($t = c \times k = 5 \times 5$) from the PSI-Shingle approach. These (25) records will be used as an input to the banded alignment. This is why higher values of k increases the run time. For example, a top-40 query

in this case results in 400 banded alignment executions over garbled circuit which take around 2436 seconds.

We also provide the time analysis for different values of the band size (b). large values of b will slow down the banded alignment over GC because large values of b will result in more computations. Figure C.10 shows the query time for $b = 20$ where it takes 4199 seconds to do a top-40 query. Also it is noteworthy that for $b = 5$, it takes 2436 seconds. We do not provide the time analysis for a fixed value of t in joined approximation because the run time will be constant in this case.

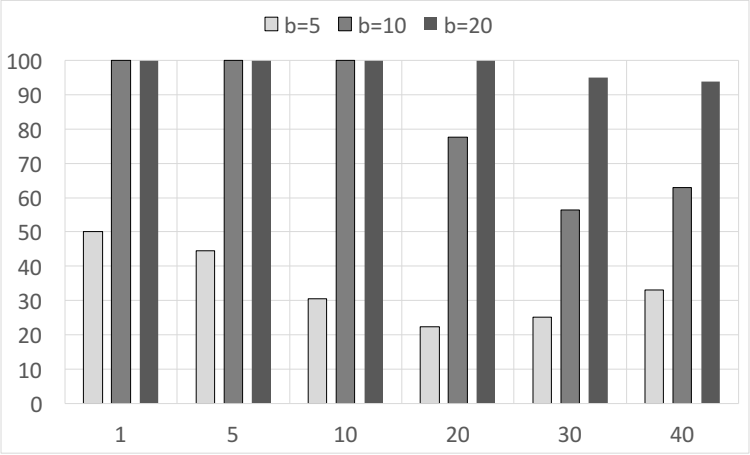


Figure C.1: TPR (banded alignment) using Dataset 1. X-axis shows different k values (top-k) and Y-axis shows the rate for different b values

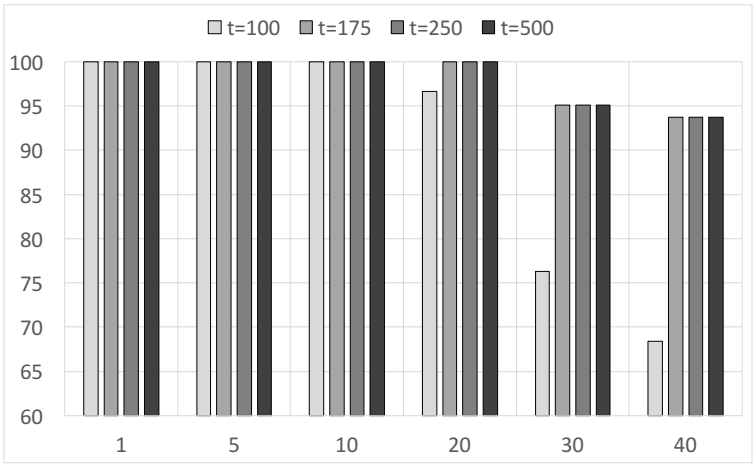


Figure C.2: TPR (banded alignment after shingling and PSI method) using Dataset 1. X-axis shows different k values (top-k) and Y-axis shows the rate for different t values

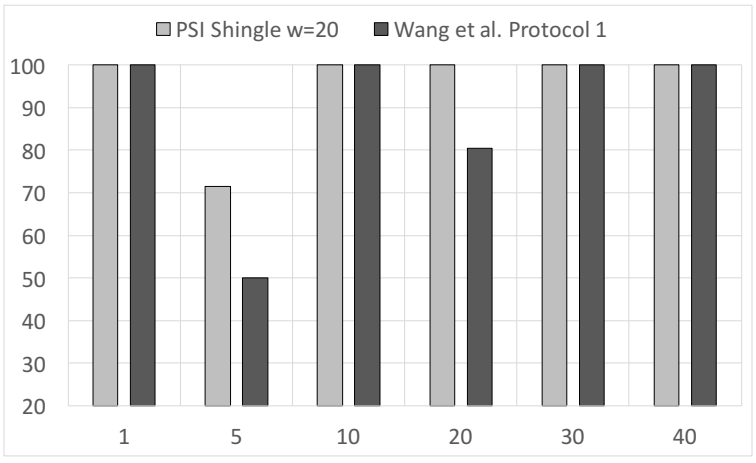


Figure C.3: Precision of shingling and PSI approximation using Dataset 1. X-axis shows different k values (top-k) and Y-axis shows the rate for different approaches

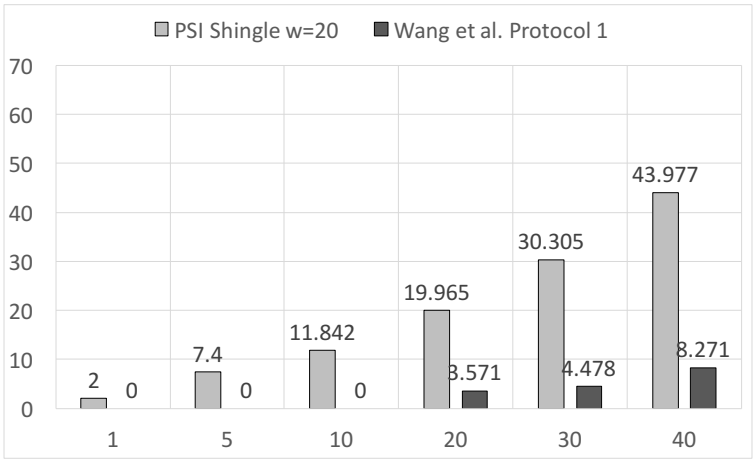


Figure C.4: Precision of shingling and PSI approximation using Dataset 2. X-axis shows different k values (top-k) and Y-axis shows the rate for different approaches

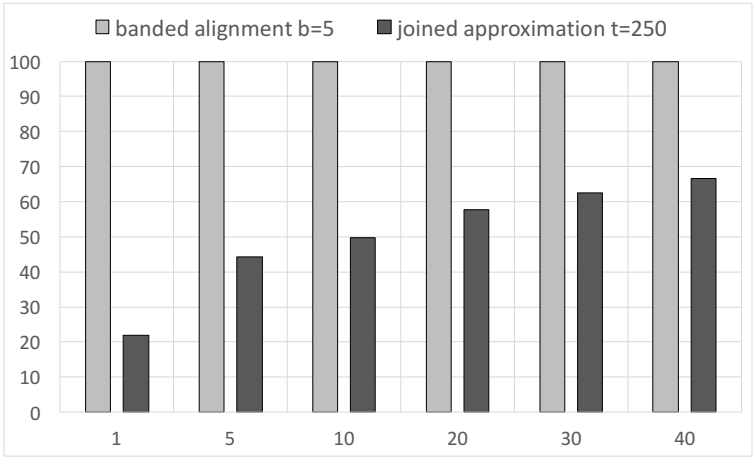


Figure C.5: Precision of banded alignment using Dataset 2. X-axis shows different k values (top-k) and Y-axis shows the rate for different approaches

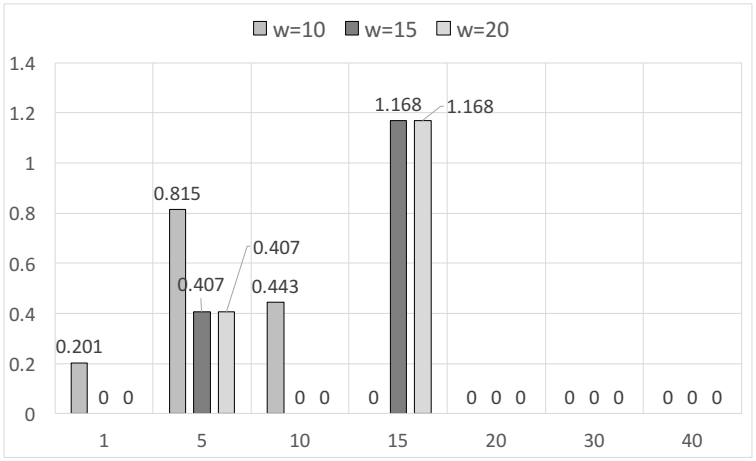


Figure C.6: False positive rate of shingling and PSI approximation using Dataset 1. X-axis shows different k values (top-k) and Y-axis shows the ratio for different w values

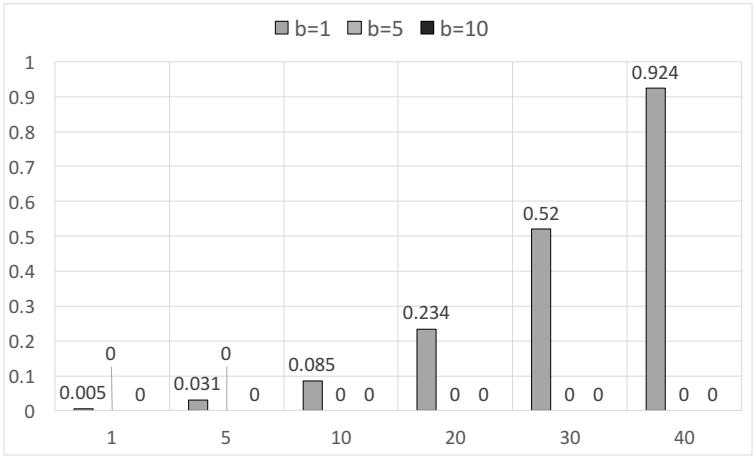


Figure C.7: False positive rate of the banded alignment using Dataset 2. X-axis shows different k values (top-k) and Y-axis shows the ratio for different t values

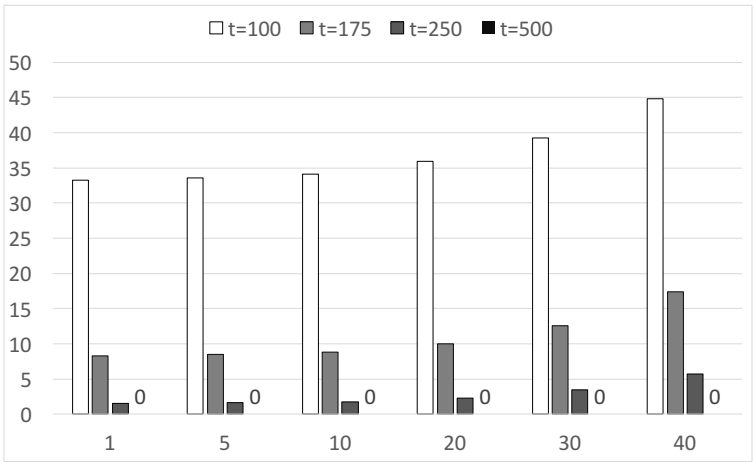


Figure C.8: False positive rate of the banded alignment after shingles and PSI method using Dataset 2. X-axis shows different k values (top-k) and Y-axis shows the ratio for different t values

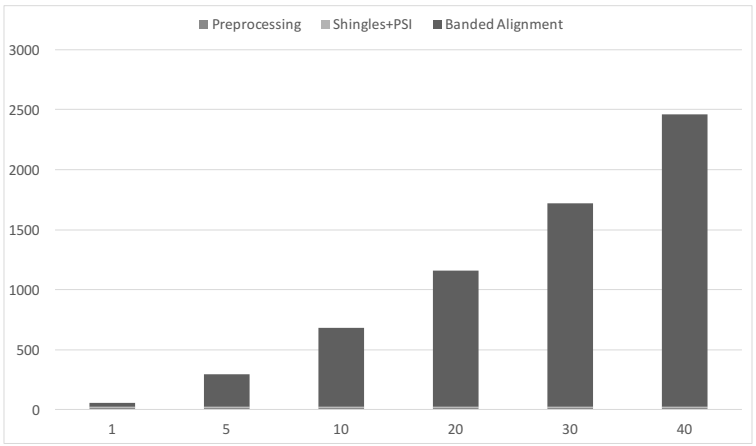


Figure C.9: Run time analysis using Dataset 1 X-axis shows different k values (top-k) and Y-axis shows the run time (in seconds) for different approximations where $b = 5, c = 10, t = ck$

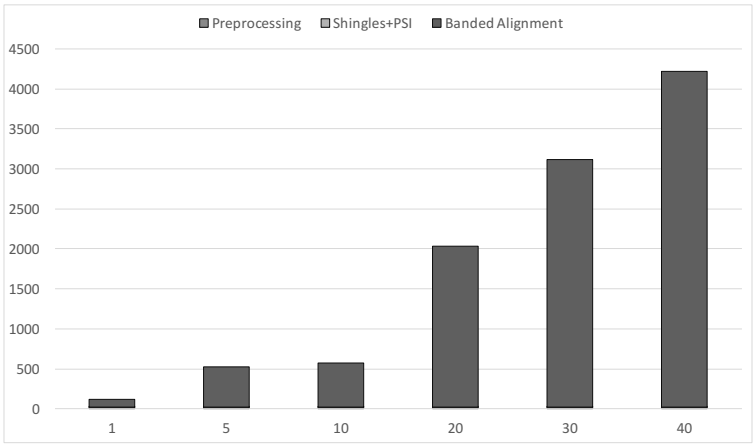


Figure C.10: Run time analysis using Dataset 1. X-axis shows different k values (top-k) and Y-axis shows the run time (in seconds) for different approximations where $b = 20, c = 5, t = ck$

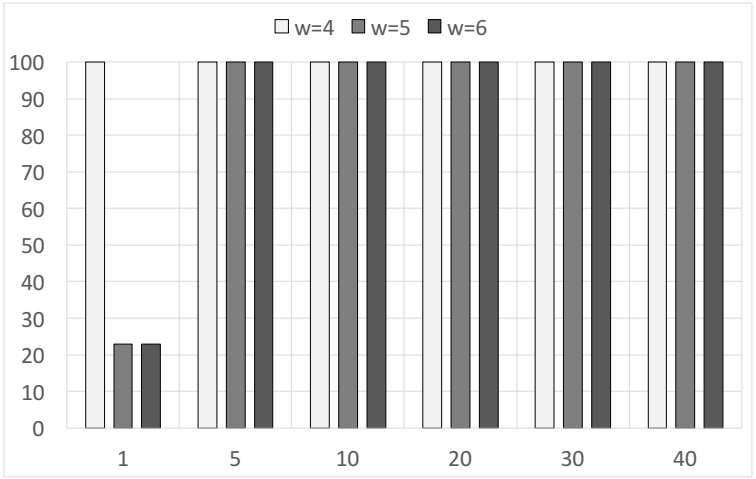


Figure C.11: True positive rate of shingling and PSI approximation using Dataset 1. X-axis shows different k values (top-k) and Y-axis shows the ratio for $w = \{4, 5, 6\}$ values

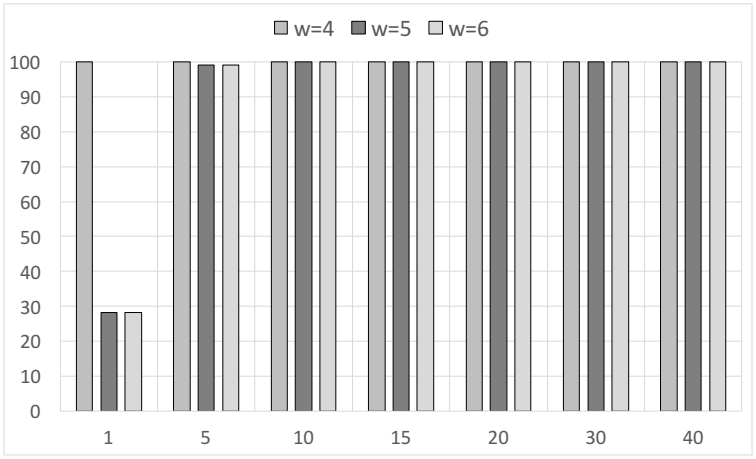


Figure C.12: False positive rate of shingling and PSI approximation using Dataset 1. X-axis shows different k values (top-k) and Y-axis shows the ratio for $w = \{4, 5, 6\}$ values

Bibliography

[1] Dna sequencing costs. http://www.genome.gov/sequencingcosts/. Accessed: 2016-01-11.

[2] X. Sh. Wang, Y. Huang, Y. Zhao, H. Tang, X. Wang, and D. 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*, CCS '15, pages 492–503. ACM, 2015.

[3] Suyash S Shringarpure and Carlos D Bustamante. Privacy risks from genomic data-sharing beacons. *The American Journal of Human Genetics*, 97(5):631–646, 2015.

[4] Susan Brown Trinidad, Stephanie M Fullerton, Julie M Bares, Gail P Jarvik, Eric B Larson, and Wylie Burke. Genomic research and wide data sharing: views of prospective participants. *Genetics in Medicine*, 12(8):486–495, 2010.

[5] Barry Barnes and John Dupré. *Genomes and what to make of them*. University of Chicago Press, 2009.

[6] Muhammad Naveed, Erman Ayday, Ellen W Clayton, Jacques Fellay, Carl A

Gunter, Jean-Pierre Hubaux, Bradley A Malin, and XiaoFeng Wang. Privacy in the genomic era. *ACM Computing Surveys (CSUR)*, 48(1):6, 2015.

[7] Shuang Wang, Xiaoqian Jiang, Siddharth Singh, Rebecca Marmor, Luca Bonomi, Dov Fox, Michelle Dow, and Lucila Ohno-Machado. Genome privacy: challenges, technical approaches to mitigate risk, and ethical considerations in the united states. *Annals of the New York Academy of Sciences*, 1387(1):73–83, 2017.

[8] Yaniv Erlich and Arvind Narayanan. Routes for breaching and protecting genetic privacy. *Nature Reviews Genetics*, 15(6):409–421, 2014.

[9] Latanya Sweeney. Simple demographics often identify people uniquely. *Health (San Francisco)*, 671:1–34, 2000.

[10] Scott Gottlieb. Us employer agrees to stop genetic testing. *BMJ: British Medical Journal*, 322(7284):449, 2001.

[11] Zhen Lin, Art B Owen, and Russ B Altman. Genomic research and human subject privacy. *SCIENCE-NEW YORK THEN WASHINGTON-.*, pages 183–183, 2004.

[12] Nils Homer, Szabolcs Szelinger, Margot Redman, David Duggan, Waibhav Tembe, Jill Muehling, John V Pearson, Dietrich A Stephan, Stanley F Nelson, and David W Craig. Resolving individuals contributing trace amounts of dna to highly complex mixtures using high-density snp genotyping microarrays. *PLoS Genet*, 4(8):e1000167, 2008.

[13] Augustine Kong, Gisli Masson, Michael L Frigge, Arnaldur Gylfason, Pasha Zusmanovich, Gudmar Thorleifsson, Pall I Olason, Andres Ingason, Stacy

Steinberg, Thorunn Rafnar, et al. Detection of sharing by descent, long-range phasing and haplotype imputation. *Nature genetics*, 40(9):1068–1075, 2008.

[14] Rui Wang, Yong Fuga Li, XiaoFeng Wang, Haixu Tang, and Xiaoyong Zhou. Learning your identity and disease from research papers: information leaks in genome wide association study. In *Proceedings of the 16th ACM conference on Computer and communications security*, pages 534–544. ACM, 2009.

[15] Gautam Naik. Family secrets: An adopted man's 26-year quest for his father, 2009.

[16] Michael T Goodrich. The mastermind attack on genomic data. In *Security and Privacy, 2009 30th IEEE Symposium on*, pages 204–218. IEEE, 2009.

[17] Mathias Humbert, Erman Ayday, Jean-Pierre Hubaux, and Amalio Telenti. Addressing the concerns of the lacks family: quantification of kin genomic privacy. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 1141–1152. ACM, 2013.

[18] Melissa Gymrek, Amy L McGuire, David Golan, Eran Halperin, and Yaniv Erlich. Identifying personal genomes by surname inference. *Science*, 339(6117):321–324, 2013.

[19] Matthew Fredrikson, Eric Lantz, Somesh Jha, Simon Lin, David Page, and Thomas Ristenpart. Privacy in pharmacogenetics: An end-to-end case study of personalized warfarin dosing. In *23rd USENIX Security Symposium (USENIX Security 14)*, pages 17–32, 2014.

[20] Md Momin Al Aziz, Mohammad Z. Hasan, Noman Mohammed, and Dima Alhadidi. Secure and efficient multiparty computation on genomic data. In *Proceedings of the 20th International Database Engineering & Applications Symposium*, pages 278–283. ACM, 2016.

[21] GA4GH. A federated ecosystem for sharing genomic, clinical data. *Science*, 352(6291):1278–1280, 2016.

[22] Md Momin Al Aziz, Reza Ghasemi, Md Waliullah, and Noman Mohammed. Aftermath of bustamante attack on genomic beacon service. *BMC medical genomics (to appear)*, 2017.

[23] Reza Ghasemi, Md Momin Al Aziz, Noman Mohammed, Massoud Hadian Dehkordi, and Xiaoqian Jiang. Private and efficient query processing on outsourced genomic databases. *IEEE Journal of Biomedical and Health Informatics*, 2016.

[24] Md Momin Al Aziz, Dima Alhadidi, and Noman Mohammed. Secure approximation of edit distance on genomic data. *BMC medical genomics (to appear)*, 2017.

[25] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography*, pages 265–284. Springer, 2006.

[26] J. Hsu, M. Gaboardi, A. Haeberlen, S. Khanna, A. Narayan, B. C. Pierce, and A. Roth. Differential privacy: An economic method for choosing epsilon. In *2014 IEEE 27th Computer Security Foundations Symposium*, pages 398–410, 2014.

[27] Frank D McSherry. Privacy integrated queries: an extensible platform for privacy-preserving data analysis. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*, pages 19–30. ACM, 2009.

[28] Ronald L Rivest, Len Adleman, and Michael L Dertouzos. On data banks and privacy homomorphisms. *Foundations of secure computation*, 4(11):169–180, 1978.

[29] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *International Conference on the Theory and Application of Cryptographic Techniques (EUROCRYPT)*, pages 223–238, 1999.

[30] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) lwe. *SIAM Journal on Computing*, 43(2):831–871, 2014.

[31] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. *ACM Transactions on Computation Theory (TOCT)*, 6(3):13, 2014.

[32] LÃľo Ducas and Daniele Micciancio. Fhew: Bootstrapping homomorphic encryption in less than a second. Cryptology ePrint Archive, Report 2014/816, 2014. `http://eprint.iacr.org/2014/816`.

[33] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, pages 309–325. ACM, 2012.

[34] Junfeng Fan and Frederik Vercauteren. Somewhat practical fully homomorphic encryption. *IACR Cryptology ePrint Archive*, 2012:144, 2012.

[35] Andrew Chi-Chih Yao. Protocols for secure computations. In *FOCS*, volume 82, pages 160–164, 1982.

[36] Yehuda Lindell and Benny Pinkas. A proof of security of yao's protocol for two-party computation. *Journal of Cryptology*, 22(2):161–188, 2009.

[37] Xiao Wang, Hubert Chan, and Elaine Shi. Circuit oram: On tightness of the goldreich-ostrovsky lower bound. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 850–861. ACM, 2015.

[38] Yan Huang, David Evans, Jonathan Katz, and Lior Malka. Faster secure two-party computation using garbled circuits. In *USENIX Security Symposium*, volume 201, 2011.

[39] E. M. Songhori, S. U. Hussain, A. R. Sadeghi, T. Schneider, and F. Koushanfar. Tinygarble: Highly compressed and scalable sequential garbled circuits. In *2015 IEEE Symposium on Security and Privacy*, pages 411–428, 2015.

[40] M.l O. Rabin. How to exchange secrets with oblivious transfer, 2005.

[41] Mete Akgün, A Osman Bayrak, Bugra Ozer, and M Şamil Sağıroğlu. Privacy preserving processing of genomic data: A survey. *Journal of biomedical informatics*, 56:103–111, 2015.

[42] Yaniv Erlich, James B. Williams, David Glazer, Kenneth Yocum, Nita Farahany, Maynard Olson, Arvind Narayanan, Lincoln D. Stein, Jan A.

Witkowski, and Robert C. Kain. Redefining genomic privacy: Trust and empowerment. 12(11):e1001983, 11 2014.

[43] S. Barouti, D. Alhadidi, and M. Debbabi. Symmetrically-private database search in cloud computing. In *2013 IEEE 5th International Conference on Cloud Computing Technology and Science (CloudCom)*, volume 1, pages 671–678, Dec 2013.

[44] 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 ACM Conference on Computer and Communications Security*, pages 492–503. ACM, 2015.

[45] Florian Kerschbaum. Frequency-hiding order-preserving encryption. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 656–667. ACM, 2015.

[46] Dan Boneh, Kevin Lewi, Mariana Raykova, Amit Sahai, Mark Zhandry, and Joe Zimmerman. Semantically secure order-revealing encryption: Multi-input functional encryption without obfuscation. In *Advances in Cryptology-EUROCRYPT 2015*, pages 563–594. Springer, 2015.

[47] Raluca A Popa, Frank H Li, and Nickolai Zeldovich. An ideal-security protocol for order-preserving encoding. In *Security and Privacy (SP), 2013 IEEE Symposium on*, pages 463–477. IEEE, 2013.

[48] Dongxi Liu and Shenlu Wang. Programmable order-preserving secure index for encrypted database query. In *IEEE 5th International Conference on Cloud Computing (CLOUD)*, pages 502–509. IEEE, 2012.

[49] Dongxi Liu and Shenlu Wang. Nonlinear order preserving index for encrypted database query in service cloud environments. *Concurrency and Computation: Practice and Experience*, 25(13):1967–1984, 2013.

[50] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Advances in cryptology,EUROCRYPTâĂŹ99*, pages 223–238. Springer, 1999.

[51] Dario Catalano, Rosario Gennaro, and Nick Howgrave-Graham. The bit security of paillier's encryption scheme and its applications. In *Advances in CryptologyâĂŤEUROCRYPT 2001*, pages 229–243. Springer, 2001.

[52] Moni Naor and Moti Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *Proceedings of the twenty-second annual ACM symposium on Theory of computing*, pages 427–437. ACM, 1990.

[53] Alexandra Boldyreva, Nathan Chenette, Younho Lee, and Adam O'neill. Order-preserving symmetric encryption. In *Advances in Cryptology-EUROCRYPT 2009*, pages 224–241. Springer, 2009.

[54] Florian Kerschbaum and Axel Schröpfer. Optimal average-complexity ideal-security order-preserving encryption. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 275–286. ACM, 2014.

[55] Idash-privacy and security workshop on genomic data. `http://www.humangenomeprivacy.org/2015/competition-tasks.html`, 2015.

[56] M. Lichman. UCI machine learning repository, 2013.

[57] Wei Xie, Murat Kantarcioglu, William S Bush, Dana Crawford, Joshua C Denny, Raymond Heatherly, and Bradley A Malin. Securema: protecting participant privacy in genetic association meta-analysis. *Bioinformatics 30(23): 3334–3341*, 2014.

[58] Kristin Lauter, Adriana López-Alt, and Michael Naehrig. Private computation on encrypted genomic data. In *Progress in Cryptology-LATINCRYPT 2014*, pages 3–27. Springer, 2014.

[59] Joppe W Bos, Kristin Lauter, and Michael Naehrig. Private predictive analysis on encrypted medical data. *Journal of biomedical informatics*, 50:234–243, 2014.

[60] Shuang Wang, Yuchen Zhang, Wenrui Dai, Kristin Lauter, Miran Kim, Yuzhe Tang, Hongkai Xiong, and Xiaoqian Jiang. Healer: Homomorphic computation of exact logistic regression for secure rare disease variants analysis in gwas. *Bioinformatics*, 32(2):211–218, 2016.

[61] Yuchen Zhang, Wenrui Dai, Xiaoqian Jiang, Hongkai Xiong, and Shuang Wang. Foresee: Fully outsourced secure genome study based on homomorphic encryption. *BMC medical informatics and decision making*, 15(Suppl 5):S5, 2015.

[62] Md Nazmus Sadat, Md Momin Al Aziz, Noman Mohammed, Feng Chen, Shuang Wang, and Xiaoqian Jiang. Safety: Secure gwas in federated environment through a hybrid solution with intel sgx and homomorphic encryption. *arXiv preprint arXiv:1703.02577*, 2017.

[63] Aaron Krol. Beacon Projects Cracks the Door for Genomic Data Sharing. http://goo.gl/3fo1VU, 2015. [Online; accessed 15-August-2016].

[64] Privacy Risks in Genome Sharing Network. http://goo.gl/rQqHkz, 2015. [Online; accessed 19-August-2016].

[65] Stephanie OM Dyke, Edward S Dove, and Bartha M Knoppers. Sharing health-related data: a privacy test? *NPJ genomic medicine*, 1(1):16024–1, 2016.

[66] Beacon Project Mitigates Privacy Risks While Maximizing Value of Responsible Data Sharing. goo.gl/1Uy0d1, 2015. [Online; accessed 1-November-2016].

[67] Genomic beacon privacy shringarpure. https://github.com/mominbuet/GenomicBeaconPrivacyShringarpure. Online; accessed 14 January 2017.

[68] Ruth Williams. Toward Protecting Participants Privacy. http://goo.gl/vwk7ok, 2015. [Online; accessed 20-August-2016].

[69] Beacon Network. https://www.beacon-network.org/#/about, 2015. [Online; accessed 14-August-2016].

[70] Beacon FAQs. https://genomicsandhealth.org/files/public/Beacon-FAQ.pdf, 2015. [Online; accessed 14-August-2016].

[71] Xinghua Shi and Xintao Wu. An overview of human genetic privacy. *Annals of the New York Academy of Sciences*, 1387(1):61–72, 2017.

[72] The Genome of the Netherlands Consortium. Whole-genome sequence varia-

tion, population structure and demographic history of the dutch population. *Nature Genetics*, 46(8):818–825, 2014.

[73] Lai-Ping Wong, Rick Twee-Hee Ong, Wan-Ting Poh, Xuanyao Liu, Peng Chen, Ruoying Li, Kevin Koi-Yau Lam, Nisha Esakimuthu Pillai, Kar-Seng Sim, Haiyan Xu, et al. Deep whole-genome sequencing of 100 southeast asian malays. *The American Journal of Human Genetics*, 92(1):52–66, 2013.

[74] Stanley L Warner. Randomized response: A survey technique for eliminating evasive answer bias. *Journal of the American Statistical Association*, 60(309):63–69, 1965.

[75] Cynthia Dwork, Aaron Roth, et al. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science*, 9(3-4):211–407, 2014.

[76] idash privacy security workshop 2016. `http://www.humangenomeprivacy.org/2016/competition-tasks.html`. Online; accessed 23 December 2016.

[77] Jean Louis Raisaro, Florian Tramer, Zhanglong Ji, Diyue Bu, Yongan Zhao, Knox Carey, David Lloyd, Heidi Sofia, Dixie Baker, Paul Flicek, et al. Addressing beacon re-identification attacks: Quantification and mitigation of privacy risks. Technical report, 2016.

[78] Genomic beacon privacy idash 2016. `https://github.com/mominbuet/GenomicBeaconPrivacyIDASH`. Online; accessed 14 January 2017.

[79] Latanya Sweeney, Akua Abu, and Julia Winn. Identifying participants in the personal genome project by name. *Available at SSRN 2257732*, 2013.

[80] Melissa Gymrek, Amy L McGuire, David Golan, Eran Halperin, and Yaniv Erlich. Identifying personal genomes by surname inference. *Science*, 339(6117):321–324, 2013.

[81] Jane Gitschier. Inferential genotyping of y chromosomes in latter-day saints founders and comparison to utah samples in the hapmap project. *The American Journal of Human Genetics*, 84(2):251–258, 2009.

[82] Grigorios Loukides, Aris Gkoulalas-Divanis, and Bradley Malin. Anonymization of electronic medical records for validating genome-wide association studies. *Proceedings of the National Academy of Sciences*, 107(17):7898–7903, 2010.

[83] Xiaoyong Zhou, Bo Peng, Yong Fuga Li, Yangyi Chen, Haixu Tang, and XiaoFeng Wang. To release or not to release: evaluating information leaks in aggregate human-genome data. In *European Symposium on Research in Computer Security*, pages 607–627. Springer, 2011.

[84] Nils Homer, Szabolcs Szelinger, Margot Redman, David Duggan, Waibhav Tembe, Jill Muehling, John V Pearson, Dietrich A Stephan, Stanley F Nelson, and David W Craig. Resolving individuals contributing trace amounts of dna to highly complex mixtures using high-density snp genotyping microarrays. *PLoS Genet*, 4(8):e1000167, 2008.

[85] Rui Wang, Yong Fuga Li, XiaoFeng Wang, Haixu Tang, and Xiaoyong Zhou. Learning your identity and disease from research papers: information leaks in genome wide association study. In *Proceedings of the 16th ACM conference on Computer and communications security*, pages 534–544. ACM, 2009.

[86] Bradley Malin and Latanya Sweeney. How (not) to protect genomic data privacy in a distributed network: using trail re-identification to evaluate and design anonymity protection systems. *Journal of biomedical informatics*, 37(3):179–192, 2004.

[87] Muhammad Naveed, Erman Ayday, Ellen W Clayton, Jacques Fellay, Carl A Gunter, Jean-Pierre Hubaux, Bradley A Malin, and XiaoFeng Wang. Privacy and security in the genomic era. *arXiv preprint arXiv:1405.1891*, 2014.

[88] Jeffrey C Barrett, Sarah Hansoul, Dan L Nicolae, Judy H Cho, Richard H Duerr, John D Rioux, Steven R Brant, Mark S Silverberg, Kent D Taylor, M Michael Barmada, et al. Genome-wide association defines more than thirty distinct susceptibility loci for crohn's disease. *Nature genetics*, 40(8):955, 2008.

[89] Geoffrey S Ginsburg, Thomas W Burke, and Phillip Febbo. Centralized biorepositories for genetic and genomic. *JAMA*, 299(11):1359–1361, 2008.

[90] Dina N Paltoo, Laura Lyman Rodriguez, Michael Feolo, Elizabeth Gillanders, Erin M Ramos, Joni Rutter, Stephen Sherry, Vivian Ota Wang, Alice Bailey, Rebecca Baker, et al. Data use under the nih gwas data sharing policy and future directions. *Nature genetics*, 46(9):934, 2014.

[91] 1000 Genomes Project Consortium et al. A map of human genome variation from population-scale sequencing. *Nature*, 467(7319):1061–1073, 2010.

[92] Murat Kantarcioglu, Wei Jiang, Ying Liu, and Bradley Malin. A cryptographic approach to securely share and query genomic sequences. *IEEE*

Transactions on Information Technology in Biomedicine, 12(5):606–617, 2008.

[93] Mustafa Canim, Murat Kantarcioglu, and Bradley Malin. Secure management of biomedical data with cryptographic hardware. *IEEE Transactions on Information Technology in Biomedicine*, 16(1):166–175, 2012.

[94] Kana Shimizu, Koji Nuida, and Gunnar Rätsch. Efficient privacy-preserving string search and an application in genomics. *Bioinformatics, 32(11): 1652–1661*, 2016.

[95] Justin Wagner, Joseph N. Paulson, Xiao Wang, Bobby Bhattacharjee, and Hèctor Corrada Bravo. Privacy-preserving microbiome analysis using secure computation. *Bioinformatics*, 2016.

[96] Scott D Kahn et al. On the future of genomic data. *Science(Washington)*, 331(6018):728–729, 2011.

[97] Jorge L Contreras. Nih's genomic data sharing policy: timing and tradeoffs. *Trends in Genetics*, 31(2):55–57, 2015.

[98] David W Mount and David W Mount. *Bioinformatics: sequence and genome analysis*, volume 2. Cold spring harbor laboratory press New York:, 2001.

[99] Oded Goldreich. *Foundations of cryptography: volume 2, basic applications*. Cambridge university press, 2009.

[100] Shuang Wang, Yuchen Zhang, Wenrui Dai, Kristin Lauter, Miran Kim, Yuzhe Tang, Hongkai Xiong, and Xiaoqian Jiang. Healer: Homomorphic computation of exact logistic regression for secure rare disease variants analysis in gwas. *Bioinformatics, 32(2):211–8*, 2015.

[101] Jung Hee Cheon, Miran Kim, and Kristin Lauter. Homomorphic computation of edit distance. In *Financial Cryptography and Data Security*, pages 194–212. Springer, 2015.

[102] Miran Kim and Kristin Lauter. Private genome analysis through homomorphic encryption. *BMC medical informatics and decision making*, 15(Suppl 5):S3, 2015.

[103] Mahsa Shabani, Stephanie OM Dyke, Yann Joly, and Pascal Borry. Controlled access under review: Improving the governance of genomic data access. *PLOS Biol*, 13(12):e1002339, 2015.

[104] Shai Halevi and Victor Shoup. Algorithms in helib. In *Advances in Cryptology–CRYPTO 2014*, pages 554–571. Springer, 2014.

[105] Nathan Dowlin, Ran Gilad-Bachrach, Kim Laine, Kristin Lauter, Michael Naehrig, and John Wernsing. Manual for using homomorphic encryption for bioinformatics. 2015.

[106] Benjamin Fung, Ke Wang, Rui Chen, and Philip S Yu. Privacy-preserving data publishing: A survey of recent developments. *ACM Computing Surveys (CSUR)*, 42(4):14, 2010.

[107] Latanya Sweeney. k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(05):557–570, 2002.

[108] C. Dwork. Differential privacy. In *Proceedings of the 33rd International Conference on Automata, Languages and Programming (ICALP)*, pages 1–12. 2006.

[109] Khaled El Emam, Fida Kamal Dankar, Romeo Issa, Elizabeth Jonker, Daniel Amyot, Elise Cogo, Jean-Pierre Corriveau, Mark Walker, Sadrul Chowdhury, Regis Vaillancourt, et al. A globally optimal k-anonymity method for the de-identification of health data. *Journal of the American Medical Informatics Association*, 16(5):670–682, 2009.

[110] Khaled El Emam. Methods for the de-identification of electronic health records for genomic research. *Genome medicine*, 3(4):25, 2011.

[111] Ashwin Machanavajjhala, Daniel Kifer, Johannes Gehrke, and Muthuramakrishnan Venkitasubramaniam. l-diversity: Privacy beyond k-anonymity. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 1(1):3, 2007.

[112] Ninghui Li, Tiancheng Li, and Suresh Venkatasubramanian. Closeness: A new privacy measure for data publishing. *IEEE Transactions on Knowledge and Data Engineering*, 22(7):943–956, 2010.

[113] Lei Zhang, Sushil Jajodia, and Alexander Brodsky. Information disclosure under realistic assumptions: Privacy versus optimality. In *Proceedings of the 14th ACM conference on Computer and communications security*, pages 573–583. ACM, 2007.

[114] Srivatsava Ranjit Ganta, Shiva Prasad Kasiviswanathan, and Adam Smith. Composition attacks and auxiliary information in data privacy. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 265–273. ACM, 2008.

[115] Benjamin Fung, Ke Wang, and Philip S Yu. Anonymizing classification data

for privacy preservation. *IEEE Transactions on Knowledge and Data Engineering*, 19(5):711–725, 2007.

[116] Manolis Terrovitis, Nikos Mamoulis, and Panos Kalnis. Local and global recoding methods for anonymizing set-valued data. *Journal on Very Large Data Bases (VLDB)*, 20(1):83–106, 2011.

[117] Liyue Fan, Li Xiong, and Vaidy Sunderam. Differentially private multi-dimensional time series release for traffic monitoring. In *Data and Applications Security and Privacy XXVII*, pages 33–48. Springer, 2013.

[118] Grigorios Loukides, Aris Gkoulalas-Divanis, and Bradley Malin. Anonymization of electronic medical records for validating genome-wide association studies. *Proceedings of the National Academy of Sciences*, 107(17):7898–7903, 2010.

[119] Raymond D Heatherly, Grigorios Loukides, Joshua C Denny, Jonathan L Haines, Dan M Roden, and Bradley A Malin. Enabling genomic-phenomic association discovery without sacrificing anonymity. *PloS one*, 8(2):e53875, 2013.

[120] Bradley Malin. *Protecting DNA sequence anonymity with generalization lattices*. Carnegie Mellon University, School of Computer Science [Institute for Software Research International], 2004.

[121] Stephen E Fienberg, Aleksandra Slavkovic, and Caroline Uhler. Privacy preserving gwas data sharing. In *Data Mining Workshops (ICDMW), 2011 IEEE 11th International Conference on*, pages 628–635. IEEE, 2011.

[122] Aaron Johnson and Vitaly Shmatikov. Privacy-preserving data exploration in genome-wide association studies. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1079–1087. ACM, 2013.

[123] Centers for Medicare & Medicaid Services. Are you a covered entity? `https://goo.gl/sdkm13`. Online; accessed 6 December 2016.

[124] Vladimir I Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. In *Soviet physics doklady*, volume 10, page 707, 1966.

[125] William Cohen, Pradeep Ravikumar, and Stephen Fienberg. A comparison of string metrics for matching names and records. In *Kdd workshop on data cleaning and object consolidation*, volume 3, pages 73–78, 2003.

[126] Alexandr Andoni and Krzysztof Onak. Approximating edit distance in near-linear time. *SIAM Journal on Computing*, 41(6):1635–1648, 2012.

[127] Arturs Backurs and Piotr Indyk. Edit distance cannot be computed in strongly subquadratic time (unless seth is false). In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing*, pages 51–58. ACM, 2015.

[128] Nick Koudas, Amit Marathe, and Divesh Srivastava. Flexible string matching against large databases in practice. In *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*, pages 1078–1086. VLDB Endowment, 2004.

[129] Ziv Bar-Yossef, TS Jayram, Robert Krauthgamer, and Ravi Kumar. Approx-

imating edit distance efficiently. In *Foundations of Computer Science, 2004. Proceedings. 45th Annual IEEE Symposium on*, pages 550–559. IEEE, 2004.

[130] Noralane M Lindor, Kiley J Johnson, Jennifer B McCormick, Eric W Klee, Matthew J Ferber, and Gianrico Farrugia. Preserving personal autonomy in a genomic testing era. *Genetics in Medicine*, 15(5):408–409, 2013.

[131] Jure Leskovec, Anand Rajaraman, and Jeffrey David Ullman. *Mining of massive datasets*. Cambridge University Press, 2014.

[132] Benny Pinkas, Thomas Schneider, Gil Segev, and Michael Zohner. Phasing: Private set intersection using permutation-based hashing. In *24th USENIX Security Symposium (USENIX Security 15)*, pages 515–530, 2015.

[133] James W Fickett. Fast optimal alignment. *Nucleic acids research*, 12(1Part1):175–179, 1984.

[134] Luis Gravano, Panagiotis G Ipeirotis, Hosagrahar Visvesvaraya Jagadish, Nick Koudas, Shanmugauelayut Muthukrishnan, Divesh Srivastava, et al. Approximate string joins in a database (almost) for free. In *VLDB*, volume 1, pages 491–500, 2001.

[135] R. A. Wagner and M. J. Fischer. The string-to-string correction problem. *Journal of the ACM*, 21(1):168–173, 1974.

[136] Oded Goldreich. *Foundations of Cryptography: Volume 2, Basic Applications*. Cambridge University Press, New York, NY, USA, 2004.

[137] Esko Ukkonen. Approximate string-matching with q-grams and maximal matches. *Theoretical computer science*, 92(1):191–211, 1992.

[138] Grzegorz Kondrak. N-gram similarity and distance. In *International Symposium on String Processing and Information Retrieval*, pages 115–126. Springer, 2005.

[139] Secure approximate edit distance. `https://github.com/mominbuet/SecureApproxEditDistance`. Online; accessed 14 January 2017.

[140] 1000 genomes dataset phase 1. `ftp://ftp.1000genomes.ebi.ac.uk/vol1/ftp/phase1/analysis_results/integrated_call_sets/`. Online; accessed 23 December 2016.

[141] Encryptogroup. `https://github.com/encryptogroup/PSI`. Online; accessed 23 December 2016.

[142] Benny Pinkas, Thomas Schneider, and Michael Zohner. Faster private set intersection based on ot extension. In *23rd USENIX Security Symposium (USENIX Security 14)*, pages 797–812, 2014.

[143] Catherine Meadows. A more efficient cryptographic matchmaking protocol for use in the absence of a continuously available third party. In *Security and Privacy, 1986 IEEE Symposium on*, pages 134–134. IEEE, 1986.

[144] Somesh Jha, Louis Kruger, and Vitaly Shmatikov. Towards practical privacy for genomic computation. In *2008 IEEE Symposium on Security and Privacy (sp 2008)*, pages 216–230. IEEE, 2008.

[145] Craig Gentry and Shai Halevi. Implementing gentryâĂŹs fully-homomorphic encryption scheme. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 129–148. Springer, 2011.

[146] Kana Shimizu, Koji Nuida, and Gunnar RÃďtsch. Efficient privacy-preserving string search and an application in genomics. *Bioinformatics*, 32(11):1652–1661, 2016.

[147] Marina Blanton and Mehrdad Aliasgari. Secure outsourcing of dna searching via finite automata. In *IFIP Annual Conference on Data and Applications Security and Privacy*, pages 49–64. Springer, 2010.

[148] Pierre Baldi, Roberta Baronio, Emiliano De Cristofaro, Paolo Gasti, and Gene Tsudik. Countering gattaca: efficient and secure testing of fully-sequenced human genomes. In *Proceedings of the 18th ACM conference on Computer and communications security*, pages 691–702. ACM, 2011.

[149] Rui Wang, XiaoFeng Wang, Zhou Li, Haixu Tang, Michael K Reiter, and Zheng Dong. Privacy-preserving genomic computation through program specialization. In *Proceedings of the 16th ACM conference on Computer and communications security*, pages 338–347. ACM, 2009.

[150] Jorge S Reis-Filho. Next-generation sequencing. *Breast Cancer Research*, 11(3):S12, 2009.

[151] József Dénes. The representation of a permutation as the product of a minimal number of transpositions and its connection with the theory of graphs. *Publ. Math. Inst. Hungar. Acad. Sci*, 4:63–70, 1959.

[152] Chiea Chuen Khor, Sonia Davila, Willemijn B Breunis, Yi-Ching Lee, Chisato Shimizu, Victoria J Wright, Rae SM Yeung, Dennis EK Tan, Kar Seng Sim, Wang, and Jie Jin. Genome-wide association study identifies fcgr2a as

a susceptibility locus for kawasaki disease. *Nature genetics*, 43(12):1241–1246, 2011.