PRIVACY-PRESERVING GENOMIC DATA PUBLISHING VIA DIFFERENTIAL PRIVACY

by

Tanya Khatri

A thesis

submitted in partial fulfillment

of the requirements for the degree of

Master of Science in Computer Science

Boise State University

December 2018

BOISE STATE UNIVERSITY GRADUATE COLLEGE

DEFENSE COMMITTEE AND FINAL READING APPROVALS

of the thesis submitted by

Tanya Khatri

Thesis Title: Privacy-Preserving Genomic Data Publishing Via Differential Privacy

Date of Final Oral Examination: 2 October 2018

The following individuals read and discussed the thesis submitted by student Tanya Khatri, and they evaluated the presentation and response to questions during the final oral examination. They found that the student passed the final oral examination.

Gaby Dagher, Ph.D.	Chair, Supervisory Committee
Jyh-haw Yeh, Ph.D.	Member, Supervisory Committee
Yantian Hou, Ph.D.	Member, Supervisory Committee

The final reading approval of the thesis was granted by Gaby Dagher, Ph.D., Chair of the Supervisory Committee. The thesis was approved by the Graduate College.

dedicated to “my family”

ACKNOWLEDGMENTS

I would like to express my thanks and gratitude to my advisor Dr. Gaby Dagher, for his persistent guidance, encouragement and comprehensive advice throughout the research. His technical expertise and editorial advice was significantly helpful throughout the research and in writing the thesis. I would also like to thank my committee members, Dr. Jyh-haw Yeh and Dr. Yantian Hou for their valuable feedback during the research process. Heartfelt thanks to my husband for being co-operative and understanding and also for his continuous love, support and motivation throughout my graduation process.

Abstract

Privacy-preserving data publishing is a mechanism for sharing data while ensuring the privacy of individuals is preserved in the published data and utility is maintained for data mining and analysis. There is a huge need for sharing genomic data to advance medical and health research. However, since genomic data is highly sensitive and the ultimate identifier, it is a big challenge to publish genomic data while protecting the privacy of individuals in the data.

In this thesis, we address the aforementioned challenge by presenting an approach for privacy-preserving genomic data publishing via differentially-private suffix tree. The proposed algorithm uses a top-down approach and utilizes Laplace mechanism to divide the raw genomic data into disjoint partitions, and then normalize the partitioning structure to ensure consistency and maintain utility. The output of our algorithm is a differentially-private suffix tree, a data structure most suitable for efficient search on genomic data. We experiment on real-life genomic data obtained from the Human Genome Privacy Challenge project, and we show that our approach is efficient, scalable, and achieves high utility with respect to genomic sequence matching count queries.

Contents

List of Tables

List of Figures

LIST OF ABBREVIATIONS

PPDP –	Privacy-Preserving Data Publishing
PPDM –	Privacy-Preserving Data Mining

LIST OF SYMBOLS

$\mathcal{D}$	denotes genomic data
B	denotes total privacy budget
B_l	denotes the privacy budget per level
h	denotes the height of the partitioning tree
$\mathcal{P}$	denotes the partitioning tree
$\hat{\mathcal{P}}$	denotes the normalized partitioning tree
$\mathcal{T}$	denotes the B-differentially-private suffix tree
θ	theta symbol, denotes the specialization threshold
c	denotes the optimal constant used for computing specialization threshold
k	denotes the length of the genomic sequence
n	denotes the number of sequences

Chapter 1

INTRODUCTION

In this revolutionary era of science and technology, it is much easier to access, analyze and interpret genomic data. Genomes are the most vital part of the human, as they include health and other information about the person, as well as their ancestors, siblings and decedents. Since the sharing of genomic data is essential for the advancement of genomic research, it is important to ensure that the sensitive information about individuals in their genomic sequences is protected in any shared data for scientific research. Maintaining the correct trade-off between utility and privacy is particularly challenging for genomic data as each individuals' DNA sequence is unique and therefore, a DNA sample can never be made truly anonymized.

According to the National Human Genome Research Institute (NHGRI) [38]:

> "People have a right to keep their medical information, and that of their dependents, private. Yet medical records are a rich source of research data, and it is in the interest of medical research, and thus everyone's health and well-being, that scientists have access to large numbers of participants and quantities of data. How do we strike the proper balance between scientific progress and patient privacy?"

The Health Insurance Portability and Accountability Act (HIPAA) [1] and Genetic Information Nondiscrimination Act (GINA) [2] are the frameworks provided by the government to achieve the aforementioned delicate balance.

Genome Wide Association Studies (GWAS) investigate genomic and biometric data with the purpose of identifying genetic variations that may be linked to diseases. The goal of GWAS is to produce aggregate statistics that are produced by examining many single nucleotide polymorphism (SNPs) locations from a group of study participants. This can be achieved by calculating *chi-squared* and *p-value* statistics. Since the aggregate statistics are taken from a sample of thousands of individuals, researchers believed that privacy was preserved by using de-identification techniques and a large sample size.

With the emergence of cloud computing, the possibility of large scale distribution of data collection from multiple resources has increased, as has the threat to privacy or information leakage. Recent work has shown, however, that the large volume of data collected from each patient exposes them to privacy breaches, even if only the aggregate statistics are reported. Homer *et al.* [36] show that a participant's information can be inferred from the allele frequencies of a large number of single-nucleotide polymorphisms (SNPs). Given the minor allele frequencies (MAFs) of both a reference population and a test population, the presence of an individual with a known genotype can be inferred using a t-test and a distance metric designed to contrast similarity between an individual and the test population. Table 1.1 shows example statistics for Homer's attack, where given the genome of the victim (set of variants), the size of the mixture and the population allele frequencies, an attacker can re-identify an individual in a case group with a certain disease by calculating the distance measure $D(x) = |x - p| - |x - m|$. If $D > 0$, it means that an individual is most likely to be in the mixture and if $D < 0$, it means that an individual is most likely to be in the reference population. $D = 0$ means equally likely to be in the mixture and in the reference population. Wang *et al.* [87] show that a participants' actual genome can be reconstructed using correlation information about the SNPs. There are many other attacks [32] [31] [71] that could result in breaching the privacy of individuals.

Table 1.1: Homer's Attack : The attacker knows the genome of the victim (set of variants), the size of the mixture he's attacking and the population allele frequencies. [36]

Id	**Allele Frequency**			**Distance Measure**	**Inference**
	Reference Population(p)	Mixture(m)	Person of Interest(x)	$D(x) = \lvert x - p\rvert - \lvert x - m\rvert$	
j	0.25	0.75	1.0	0.50	Most likely to be in the Mixture
j+1	0.25	0.75	0.50	0.00	Equally likely to be in the Mixture and in the Reference Population
j+2	0.25	0.75	0.0	-0.50	Most likely to be in the Reference Population

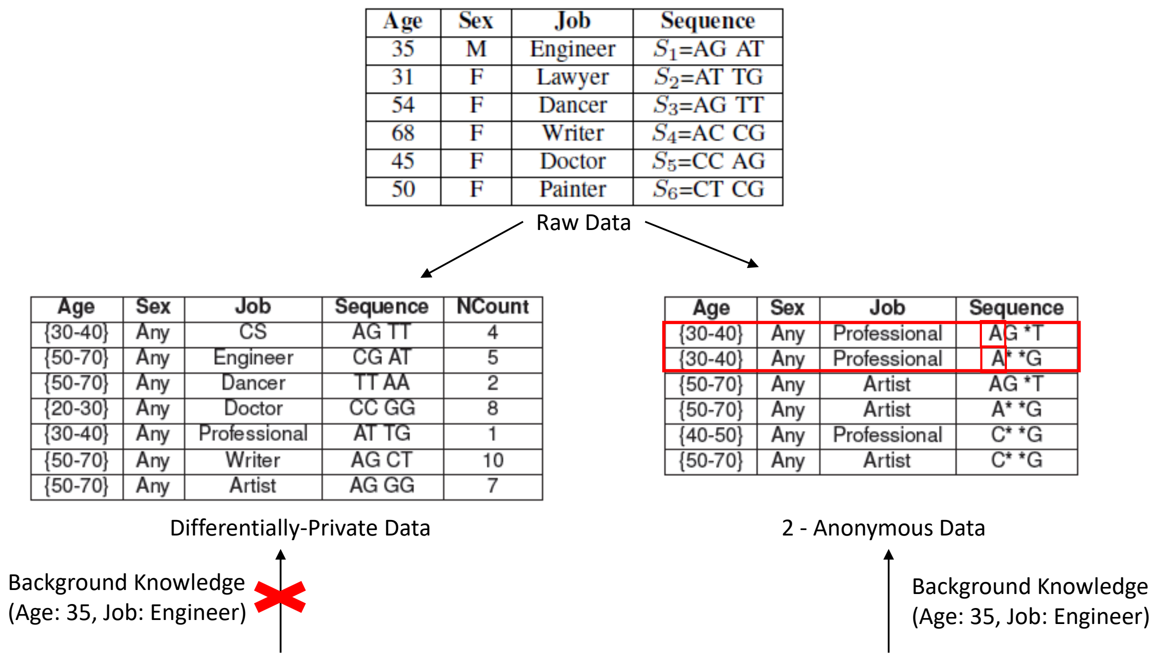
It is a challenge to promote privacy-preserving genomic data sharing for scientific research, given that genomic data is extremely high-dimensional and cannot be revoked. It is a treasure trove of sensitive information and it is the ultimate identifier, as it contains millions of SNPs that are easily identifiable. As shown in [36], not only the genomic data needs to be protected, but the analysis results containing allele frequencies or test statistics as well. Therefore, there is a need to strike a balance between privacy and utility such that the output is privacy-preserving while utility is maintained.

1.1 Motivation

Several methods have been proposed to address the problem of privacy-preserving data publishing. Samarati [74] and Sweeney [77] propose the k-anonymity privacy model, which stipulates that an individual should not be identifiable from a group of less than k individuals based on quasi-identifier attributes QIDs. However, Machanavajjhala *et al.* [55] show that with additional knowledge about the victim, k-anonymous data is vulnerable against background knowledge attacks. Similarly, there are a number of other partition-based privacy models such as $(\alpha - k)$ anonymity [90] [39], t-closeness [53] [76] and $(c - k)$-safety that model the adversary differently and have different assumptions about its background knowledge. The following example illustrates how background knowledge can be used to infer sensitive information from k-anonymous genomic data.

Example 1.1.1. Given raw data as shown in Figure 1.1 containing genomic information about individuals, we anonymize the raw data and generate 2-anonymous data. If the attacker knows that the data contains information about a person who is an engineer and 35 years of age, then the attacker can determine with 100% certainty that the genomic sequence of that person starts with nucleotide *A*. However, a differentially-private version of the raw data neutralizes the linkage attack i.e., if you cannot link when I am not in the database, then obviously you cannot link when I am in the database. An attacker can not infer anything about an individual by looking at differentially-private data, regardless of any background knowledge. □

Therefore, we use differential privacy to privately publish genomic data. There are different ways to publish genomic data as shown in Figure 1.2. Given raw genomic data, a first approach is to apply differential privacy and release the data, where the released data will provide utility and consistency, but will not support an efficient search on genomic data. Therefore, to ensure efficiency, a second approach is to release a suffix tree, an efficient data structure for genomic search, by applying differential privacy on the genomic data, but this tree will result in poor utility due to inconsistent suffixes. So, a third approach is to construct a suffix tree from raw data and then apply differential privacy and release a differentially-private suffix tree. Using this approach, we will be able to achieve efficiency because it will output a suffix tree, however, measuring utility and consistency is unclear. Therefore, we came up with a fourth approach (our approach), that is a hybrid approach. It takes raw genomic data, generates differentially-private suffixes, applies utility constraint to maintain usefulness and then generates a suffix tree. This approach provides utility,



Age	Sex	Job	Sequence
35	M	Engineer	S_1=AG AT
31	F	Lawyer	S_2=AT TG
54	F	Dancer	S_3=AG TT
68	F	Writer	S_4=AC CG
45	F	Doctor	S_5=CC AG
50	F	Painter	S_6=CT CG

Age	Sex	Job	Sequence	NCount
{30-40}	Any	CS	AG TT	4
{50-70}	Any	Engineer	CG AT	5
{50-70}	Any	Dancer	TT AA	2
{20-30}	Any	Doctor	CC GG	8
{30-40}	Any	Professional	AT TG	1
{50-70}	Any	Writer	AG CT	10
{50-70}	Any	Artist	AG GG	7

Age	Sex	Job	Sequence
{30-40}	Any	Professional	AG *T
{30-40}	Any	Professional	A* *G
{50-70}	Any	Artist	AG *T
{50-70}	Any	Artist	A* *G
{40-50}	Any	Professional	C* *G
{50-70}	Any	Artist	C* *G

Figure 1.1: Given raw data that contains age, sex, job and sequence, we anonymize the data and generate two versions, i.e., 2-Anonymous Data and Differentially-Private Data. If an attacker has background knowledge that the data contains information about a person who is an engineer and 35 years of age, then looking at 2-Anonymous Data, the attacker can determine with 100% certainty that the genomic sequence of that person starts with nucleotide A. However, a differentially-private version of the raw data neutralizes the linkage attack and provides privacy guarantees irrespective of an attacker's background knowledge and computational power.

consistency and efficiency by ensuring that the suffixes are consistent and the outputted suffix tree provides differential privacy.

In this thesis, we address the problem of privacy-preserving genomic data publishing by proposing an approach that efficiently perturbs the raw genome data and outputs an anonymized suffix tree that is privacy-preserving to effectively and efficiently support count queries for genomic sequence matching. To generate a privacy-preserving suffix tree, we utilize differential privacy [21], a rigorous privacy model that provides strong privacy guarantees independent of an adversary's background knowledge and computational power.

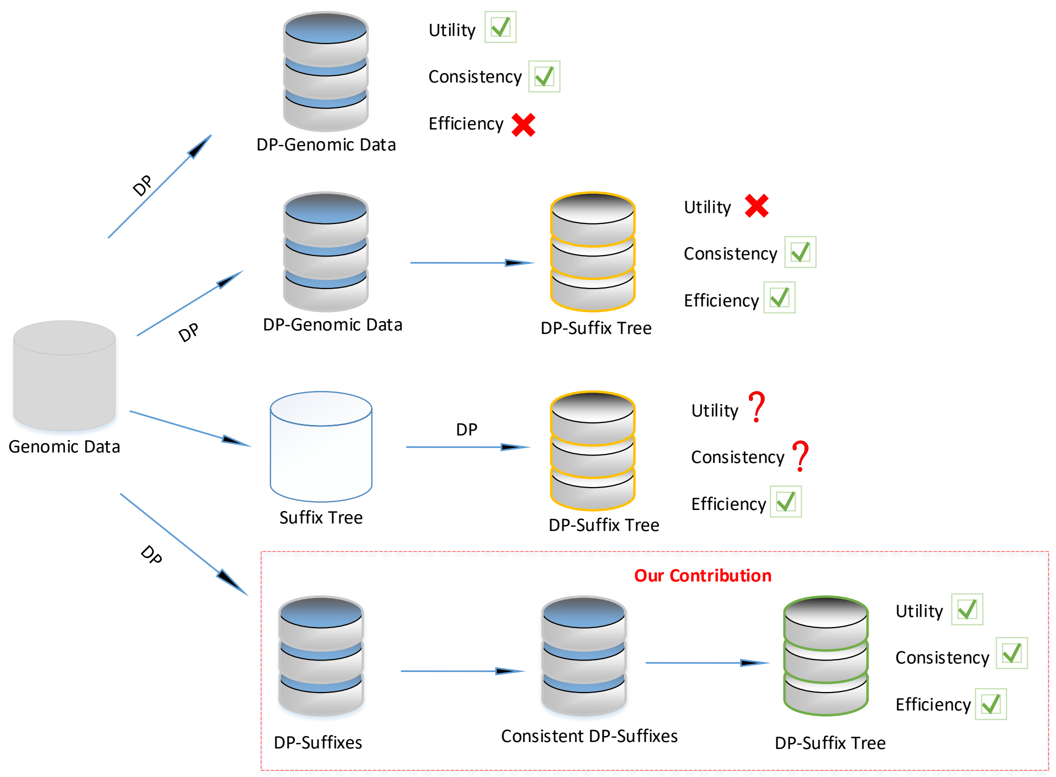


Figure 1.2: Alternative approaches to publish genomic data via differential privacy

Differential privacy is typically achieved through random perturbation, where noise is carefully calibrated to the sensitivity. A differentially-private mechanism ensures that all outputs are insensitive to the individual's data. In other words, an individual's privacy is not at risk because of her participation in the dataset.

Our approach consists of a three-phase algorithm. Given raw genomic data $\mathcal{D}$, we first construct a differential private partitioning tree, then we apply a bottom-up approach to normalize the tree, and finally we apply a top-down approach on the normalized tree to obtain a differentially-private suffix tree.

The contributions of this thesis can be summarized as follows:

- We propose a novel non-interactive approach for anonymizing genomic data and releasing a B-differentially-private suffix tree with an effective utility for genomic sequence matching. We are the first to publish a differentially-private suffix tree for efficient searching on genomic data.

- We propose a top-down partitioning approach using the Laplace mechanism to efficiently process datasets containing a large number of genomic sequences.

- To obtain higher utility, we apply a normalization technique on the partitions to ensure consistency among genomic sequences and their suffixes.

- We implemented our approach and performed extensive experiments on real-life genomic data obtained from the Human Genome Privacy Challenge [3]. The results show that our approach is efficient, scalable, and achieves high utility with respect to count queries.

1.2 Thesis Statement

The objective of this thesis is to answer the following question: **How can a data owner publish genomic data while simultaneously safeguarding the privacy of the data and maintaining its usefulness?**

More specifically, given a genomic data $\mathcal{D}$, where $\mathcal{D}$ contains sequential genomic data sequences $\{S_1, \ldots, S_n\}$ and given a privacy budget B, our objective is to generate a differentially-private suffix tree $\mathcal{T}$ for efficient search that supports data mining and analysis tasks such that:

1. $\mathcal{T}$ satisfies B-differential privacy.

2. $\mathcal{T}$ preserves data utility with respect to count queries.

3. The proposed approach is efficient and scalable.

1.3 Organization of the Thesis

The thesis is organized as follows:

- Chapter 2 discusses the building blocks/preliminaries to establish the background knowledge needed for this thesis work.
- Chapter 3 talks about the related work that has been done in this field.
- Chapter 4 elaborates the proposed approach to address the problem discussed.
- Chapter 5 discusses the privacy and complexity analysis of our proposed approach.
- Chapter 6 describes the experiments and performance evaluation of our proposed solution.
- Chapter 7 concludes our work followed by a discussion about future work.

Chapter 2

BACKGROUND

In this chapter, we introduce and define the building blocks/preliminaries that help establish the foundational knowledge required to better understand the proposed work.

2.1 Genomic Data

The human genome is the complete set of genetic information which is composed of four nucleotide bases: adenine, cytosine, guanine and, thymine, represented by the letters *A,C,G,T* respectively. Single nucleotide polymorphisms, frequently called SNPs (pronounced "snips"), are the most common type of genetic variation among humans. Mostly, these variations are found in the DNA between genes in a single nucleotide that occurs at a specific position in the genome, where each variation is present to some appreciable degree within a population. Each SNP represents a difference in a single DNA building block, called a nucleotide, e.g., SNP may replace the nucleotide cytosine (C) with the nucleotide thymine (T) in a certain stretch of DNA.

Figure 2.1 shows an example of a DNA sequence of three different people. We observe that at most locations, they have exactly the same nucleotide bases but there is a specific location in which there is a variation; this variation is called a SNP.

While SNPs occur throughout a person's DNA, the frequency of occurrence is on average once in 300 nucleotides. They act as biological markers, which help scientists

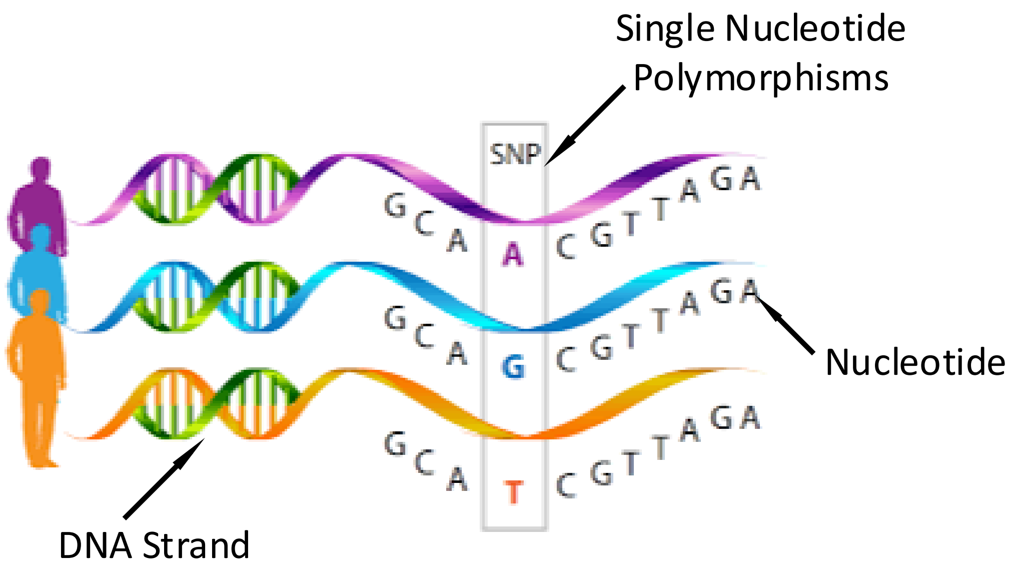


Figure 2.1: Example of genomic data of three different individuals

and researchers to identify and associate genes with diseases. They also help to identify and study particular disease trends among particular age groups. When SNPs occur within a gene or in a regulatory region near a gene, they may play a more direct role in causing certain diseases by affecting the genes function. For example, at a specific base position in the human genome, the C nucleotide may appear in most individuals, but in a minority of individuals, the position is taken by an A. This means that there is a SNP at this specific position, and the two possible nucleotide variations C or A are said to be alleles for this position. SNPs also identify differences in our susceptibility to disease, e.g., sickle-cell anemia, β-thalassemia and cystic fibrosis. The severity of illness and the way our body responds to treatments are also manifestations of genetic variations. For example, a single-base mutation in the APOE (apolipoprotein E) gene is associated with a higher risk for Alzheimer's disease [30]. Since the human genome contains the most vital and sensitive information about the person, their ancestors and family members, it is essential to ensure human genome privacy.

2.2 Differential Privacy

Differential Privacy is a model introduced by Dwork *et al.* [21] for the purpose of preserving data privacy without making assumptions about the attacker's background knowledge. Differential privacy provides a strong guarantee that the presence or absence of an individual will not affect the final output of the query significantly.

2.2.1 ϵ-Differential Privacy

Definition 2.2.1. *ϵ-Differential Privacy [88].* Given any two neighbouring datasets D_1 and D_2 that differ on, at most, one record, a sanitizing mechanism M preserves ϵ-differential privacy if for any output $\hat{D} \in Range(M)$:

$$Pr[M(D_1) = \hat{D}] \leq e^{\epsilon} \times Pr[M(D_2) = \hat{D}]$$

where the probabilities are over the randomness of M. □

To achieve differential privacy, noise must be added to the true output by calibrating the noise to the sensitivity. The sensitivity of a function is a measure of the change in the output when one of the inputs is changed. Calibrating noise to sensitivity was introduced by Dwork *et al.* [23].

Sensitivity

Noise can be added to any dataset for generating a noisy dataset to achieve differential privacy. The amount of noise to be added to achieve ϵ-differential privacy is dependent on the *sensitivity* of the function. Therefore, sensitivity can be informally defined as the maximum change possible by altering a single record (*i.e.* added or removed). In some cases, sensitivity may be represented as **global sensitivity** and therefore shares the same definition as sensitivity.

Definition 2.2.2. *Global Sensitivity [61].* Given a query function $f : D \to \mathbb{R}^d$, the global sensitivity of f is:

$$\Delta(f) = max_{D_1,D_2} \parallel f(D_1) - f(D_2) \parallel_1$$

where D_1 and D_2 are any two neighbouring datasets that differ on, at most, one record. $\square$

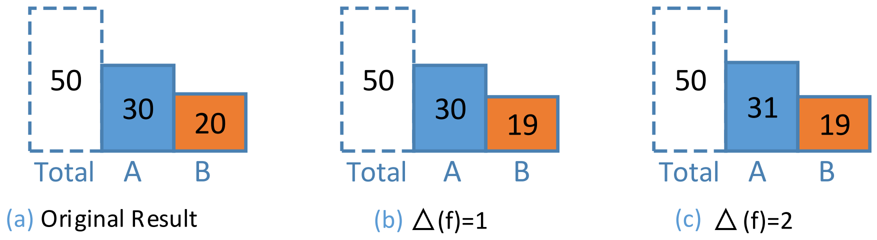


Figure 2.2: (a) represents a histogram depicting data of 50 participants who took part in a survey, the data is distributed into 2 bins *A* and *B*, where bin *A* and bin *B* represents the total number of participants who play basketball and football respectively. Figure 2.2(b) and Figure 2.2(c) represent neighboring datasets of Figure 2.2(a). Now, let's consider a participant from B decides to remove his record from the survey, that is, one value is changed in the histogram as shown in Figure 2.2(b), therefore, the $\Delta(f)$ is 1. However, if he would have changed his position from playing football to basketball, *i.e.* from bin *B* to *A* as shown in Figure 2.2(c), the $\Delta(f)$ would be 2, since 1 value was removed from *B* and 1 value was added to *A* in the histogram.

Depending on the type of data, the sensitivity may vary, which in return could affect the added noise. For example, in tabular data, each row/record represents one individual and each column represents the associated attributes. Therefore, when an individual's record is added to or removed from the dataset, it affects, at most, one row. Hence, by the above definition of sensitivity, the value of $\Delta(f)$ is equal to 1. Whereas, in the case of histogram data, the numerical data is represented graphically (bins/buckets) based on the frequency distribution. Each bin's area represents the frequency of occurrence of a certain participant-group's data (see Figure 2.2).

Laplace Distribution

To achieve differential privacy when using query function f, the principal approach is to perturb the true output of f by adding to it a random noise that is adjusted based on $\Delta(f)$. In the study in [21], the authors propose to generate the noise according to Laplace distribution, $Lap(\lambda)$, where the probability distribution function is

$$Pr(x|\lambda) = \frac{1}{2\lambda} e^{\frac{|x|}{\lambda}}$$

the mean is 0, and the standard deviation is λ which is determined based on the global sensitivity $\Delta(f)$ and the privacy level ϵ.

Theorem 2.2.1. *For any function $f : D \to \mathbb{R}^d$ that maps datasets to reals, the privacy mechanism M: $M(D) = f(D) + Lap(\Delta(f)/\epsilon)$ satisfies ϵ-differential privacy.*

$\square$

Exponential Mechanism

Exponential mechanism is one of the techniques to analyze which outputs will not make sense after noise is added, and was proposed by McSherry and Talwar [58] to select an output from the generated outputs such that the score of the utility function q is considered to make the decision. These score values represent the chances that the generated output will be similar to the optimal output where *q* has low sensitivity; *i.e.* the higher the score value, the more the chances of the generated result being similar to the optimal result, and therefore the more of a chance of it being selected.

The main goal of the Exponential mechanism is mapping *n* inputs from the domain *D* to a range *R*. This mapping may be randomized so that each element of the domain *D* corresponds to the probability distribution over the range *R*. Assuming for this scenario, *q* : (D^n × *R*) $\rightarrow$ *R* is a quality function, such that an instance of a database *d* $\in D^n$, assigns a score value to all outcomes *r* $\in$ *R*.

Definition 2.2.3. *[58]**Exponential Mechanism**: Let the sensitivity of score function q be*

$$S(q) = \max_{r, A\Delta B=1} \| q(A,r) - q(B,r) \|$$

where M is the mechanism for choosing an outcome r $\in$ R such that an instance of a database d $\in D^n$ maintains ϵ-differential privacy.

$$M(d,q) = \{ \text{ return } r \text{ with probability} \propto \exp\left(\frac{\epsilon q(d,r)}{2S(q)}\right) \} \tag{2.1}$$

□

As discussed above, the computed score value denotes the likelihood of a pair (*d,r*) having been chosen, *i.e.*, when the score is higher, it is more likely to be chosen. Exponential mechanism introduces a class of mechanisms that include all differentially-private mechanisms [58]. Therefore, it can be said that for a database and a predefined ϵ value, the quality function *q* generates a probability distribution. The Exponential mechanism samples the output on the output domain over which the probability distribution is generated [61]. The high scoring outcomes are exponentially more likely to be chosen, since the probability function favors them, while ensuring differential privacy. The Exponential mechanism by [58] is also referred to as exponential sampling technique by [45].

Theorem 2.2.2. *For any function having utility score $Utility$, an algorithm that chooses an output with probability directly proportional to $exp\frac{\epsilon}{2S(q)}(Utility)$ satisfies ϵ-differential privacy.* $\square$

Gaussian Mechanism

Gaussian mechanism was proposed in [22] and is similar to the Laplace mechanism but adds independent and identically distributed (i.i.d.) Gaussian noise to achieve (ϵ,δ)-differential privacy, where $\delta > 0$ but typically a smaller ϵ for the same utility. We know that $\mathcal{Q}$-function is defined as $\mathcal{Q}(x) := \frac{1}{\sqrt{2\pi}} \int_x^\infty e^{\frac{-u^2}{2}} du$. We include the following theorem from [67].

Theorem 2.2.3. *Let q: $D \rightarrow \mathbb{R}^k$ be a query and $\epsilon > 0$, $\frac{1}{2} > \delta > 0$. Then the Gaussian mechanism $M_q : D \times \Omega \rightarrow \mathbb{R}^k$ defined by $M_q(d) = q(d) + w$, where $w \sim \mathcal{N}(0, \sigma^2 I_k)$, where $\sigma \geq \frac{\Delta 2q}{2\epsilon} (K + \sqrt{k^2 + 2\epsilon})$ and $K = \mathcal{Q}^{-1}(\delta)$, is ($\epsilon$,$\delta$)-differentially-private.* $\square$

The authors of [22] propose a differentially-private mechanism that adds i.i.d. zero mean Gaussian noise to modify answers to a numeric query. When Gaussian noise is added, it results in δ-approximate ϵ-indistinguishability [21] in the noisy sums, where the value of ϵ is greater than $[log(1/\delta)/R]^{1/2}$. Additionally, Exponential noise ensures ϵ-indistinguishability, since the value of δ is zero [22]. The Exponential noise and Gaussian noise both have their own advantages: where Exponential noise provides a more robust solution, since $\delta = 0$, Gaussian noise provides better accuracy for any ϵ value.

2.2.2 (ϵ, δ)-Differential Privacy

For an input dataset D to a randomized algorithm $\mathcal{K}$, the random variable corresponding to D is $\mathcal{K}\ (D)$. The probability of an event occurring is not similar to that of a more probable event under the distribution $\mathcal{K}(D_1)$ or $\mathcal{K}(D_2)$, because the metric in differential privacy is multiplicative. This necessary condition for differential privacy was relaxed in later research, and (ϵ,δ)-differential privacy represents a more relaxed differential privacy model compared with ϵ-differential privacy, which provides a stronger privacy guarantee. (ϵ,δ)-differential privacy is defined as follows.

Definition 2.2.4. ***(ϵ,δ)-Differential Privacy:*** *A randomized algorithm $\mathcal{K}$ is (ϵ,δ)-differentially-private if for all databases D_1, $D_2 \in (\boldsymbol{D})^n$ such that one individual's record is varied, where S represents all subsets of outputs in the equation*

$$Pr\ [\mathcal{K}(D_1) \in S] \leqslant exp(\epsilon) \times Pr\ [\mathcal{K}(D_2) \in S] + \delta\ , \tag{2.2}$$

when the value of δ = 0, the above equation represents an ϵ-differential privacy guarantee [51]. ☐

2.2.3 Local differential privacy

So far, we have talked about centralized data privacy models, where we have a trusted data administrator who can directly access private data and we have assumed that the adversary only has access to the output of the algorithm. But what if we trust no one to look at our data? Local differential privacy comes into play in such scenarios. It enables an agent to answer questions in a differentially-private manner about their own data, without sharing it with anyone else. Each individual possesses its own data element i.e. a database of size 1, and answers questions about it only in a differentially-private manner [25]. Consider the database $x \in N^{|\mathcal{X}|}$ is a collection of n elements from some domain $\mathcal{X}$, and each $x_i \in x$ is held by an individual. The x_i's are sampled independently from some distribution P_v where $v \in \{0, 1\}$. A statistical privatization mechanism Q_i is a conditional distribution that maps $x_i \in \mathcal{X}$ randomly to $y_i \in \mathcal{Y}$, where $\mathcal{Y}$ is an output possibly larger than $\mathcal{X}$. The y_i's are referred to as the privatized views of x_i's.

Definition 2.2.5. ***ϵ-local differential privacy*** *For a given non-negative ϵ, we say that a mechanism Q is ϵ-locally differentially-private if*

$$\sup_{S \in \sigma(\mathcal{Y}), x, x' \in \mathcal{X}} \frac{(S|x_i = x)}{(S|x_i = x')} \leq e^{\epsilon} \tag{2.3}$$

where $\sigma(\mathcal{Y})$ denotes an appropriate σ-field on $\mathcal{Y}$ [43][20].

□

2.2.4 Composition Properties of Differential Privacy

Any sequence of computations that each provides differential privacy in isolation also provides differential privacy in sequence, which is known as *sequential composition*.

Lemma 1. *Sequential composition [59]. Let each algorithm A_i provide ϵ_i-differential privacy. A sequence of $A_i(\mathcal{D})$ over the dataset $\mathcal{D}$ provides ($\sum_i \epsilon_i$)-differential privacy.* □

If the sequence of computations is conducted on disjoint datasets, the privacy cost does not accumulate but depends on only the worst guarantee of all the compositions. This is known as *parallel composition*.

Lemma 2. *Parallel composition [59]. Let each algorithm A_i provide ϵ_i-differential privacy. A sequence of $A_i(\mathcal{D})$ over a set of disjoint datasets $\mathcal{D}$ provides ϵ_i-differential privacy.*

□

2.3 Suffix Tree

Genomic data can be represented in a more compact way in terms of a suffix tree. Suffix trees are a space efficient data structure to store a string that allows many kinds of queries to be answered quickly. These are very efficient for searching large sequences like a genome [28] [57]. A suffix tree groups sequences with the same suffix into the same branch, such that every path from root to leaf in a tree represents a suffix. Let S denote a string, the length of which is k. Let $S[i, j]$ denote the substring of S from position i to position j. Before constructing the suffix tree, we concatenate a new character, \$ to S. The importance of this character is twofold. First, by adding it to the string, we can avoid that a suffix will be a prefix of another suffix, which is undesirable. Second, the generalization is also made easier by this operation. We always consider a fixed size sequence. A sequence $S' = AT\$$ is a suffix of a sequence $S = AGAT\$$. We formally define a suffix tree below.

Definition 2.3.1. *Suffix Tree:* A suffix tree $\mathcal{T}$ of a genomic data is a rooted, directed tree. The edges of the suffix tree are labeled with nucleotide type A, C, G, T. On a path from the root to the leaf, one can read the suffix of the string and a \$ sign. We call a leaf w reachable from the node v, if there is a directed path from v to w. $\square$

Example 2.3.1. Consider the sequence $S = ACTCAG\$$ in Table 2.1. Suffix Tree : a tree of all possible suffixes of S as shown in Figure 2.3. $\square$

Table 2.1: All possible suffixes of the sequence $ACTCAG\$$

	Suffix
1	ACTCAG$
2	CTCAG$
3	TCAG$
4	CAG$
5	AG$
6	G$
7	$

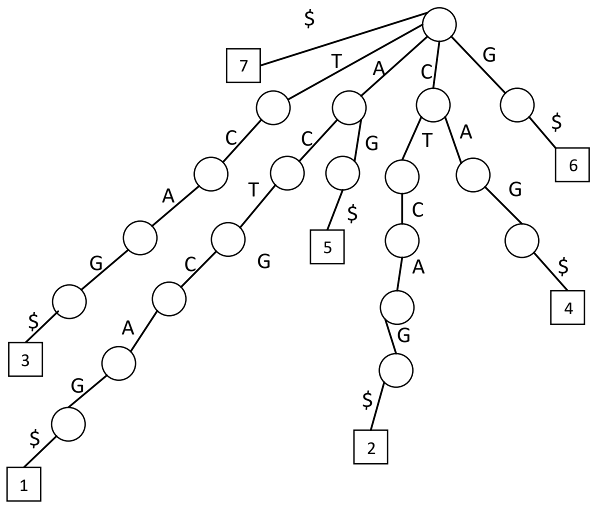


Figure 2.3: The tree of all possible suffixes of the sequence $ACTCAG\$$, where each path from root to leaf represents a unique suffix.

Chapter 3

LITERATURE REVIEW

In this chapter, we elaborate the related work, and condense the comparative information into Table 3.1 alongside our proposed work.

3.1 Privacy-Preserving Data Publishing

This related line of work studies how to transform raw data into a version that is immunized against privacy attacks but that still supports effective data mining tasks. It seeks to publish private data in a manner that maintains as much utility as possible while accomplishing the goal of anonymization. Figure 3.1 shows privacy-preserving data publishing in a non-interactive framework. In the *data collection* phase, the data publisher collects data from record owners/individuals (e.g., Jake, Bob). In the *data publishing* phase, the data publisher utilizes a differentially-private anonymizer to achieve anonymization and publishes the collected data to data recipients, who will then perform data mining tasks on the published dataset.

3.1.1 Genomic Data

Genomic data refers to the genome and DNA data of an organism which is used in bio informatics for collecting, storing and processing the genomes of living things. Although, Genome-Wide Association Studies (GWAS) are used for analysis of sets of DNA sequences to discover and identify the genetic basis of disease, these statistics that are published as the result of GWAS can be used for identification of the participating individuals. This has led to research on publishing GWAS data in a differentially-private manner. The papers [6] [94] [88] [64] [81] [26] [37] list a number of mechanisms to achieve differential privacy on genomic data. Akgün *et al.* [6] have categorized previously identified problems and their respective solutions introduced in research prior to theirs. Some of the problems and their solution techniques discussed by paper [6] are discussed in the following paragraphs in this sub-section.

Wang *et al.* [88] introduce an approach to disseminate differentially-private genomic data using top down specialization. This method assumes a data owner has a data table $\mathcal{D}(A^i, A^{snp})$ where $\mathcal{A}^i$ are explicit identifiers and $\mathcal{A}^{snp}$ a set of SNPs. The algorithm aims to satisfy ϵ-differential privacy while retaining data utility with a high sensitivity and generates an anonymized data table $\hat{D}$ that can be released to the public. Uhler *et al.* [81] introduce methods that focus on releasing differentially-private minor allele frequencies, p-values, and χ^2 statistics for the M most relevant SNPs regardless of arbitrary external information. They also apply penalized logical regression techniques while maintaining differential privacy guarantees to locate genome-wide associations in the data. Finally, for testing the approach, the proposed techniques are compared both on simulation data and on real canine hair length genomic data. This approach is an adaptation of Bhaskar *et al.* [8] to genome wide association studies. Yu *et al.* [94] and Yu & Ji [94] extend the work of Uhler *et al.* [81] by allowing for an arbitrary number of cases and controls and performance of a risk-utility analysis. Risk-utility analysis is used for assessment of performance of the proposed methods. This analysis is performed on real datasets that consist of DNA samples collected by the Wellcome Trust Case Control Consortium. Next, the methods proposed are compared to the differentially-private publishing mechanism proposed by

Johnson and Shmatikov [41]. The work of Li *et al.* [54] introduces the compressive mechanism, which uses a probabilistic compression procedure that generates a synopsis, adds Laplace noise to the compressed data, and then decodes the results. Compared to other synopsis proposals, the compressive sensing mechanism provides more accurate statistical query results while using less noise under certain conditions. Building upon this work, Roozgard *et al.* [72] presents a compressed sensing based, differentially-private genomic data dissemination algorithm that takes sequences of genomic nucleotides from multiple subjects, and transforms the frequencies of SNPs into a sparse vector representation. The Laplace noise is then added to all elements of the sparse vector. Jiang *et al.* [40] suggest a method for releasing the top-K most significant SNPs across the genome when K is small. In the past, various studies like [26] and [37] have led to recent mechanisms for publishing differentially-private data. Naveed *et al.* [64] also surveyed the field and discussed various techniques that have been used in previous studies and the mechanisms proposed for achieving differential privacy on genomic datasets. Although these approaches are

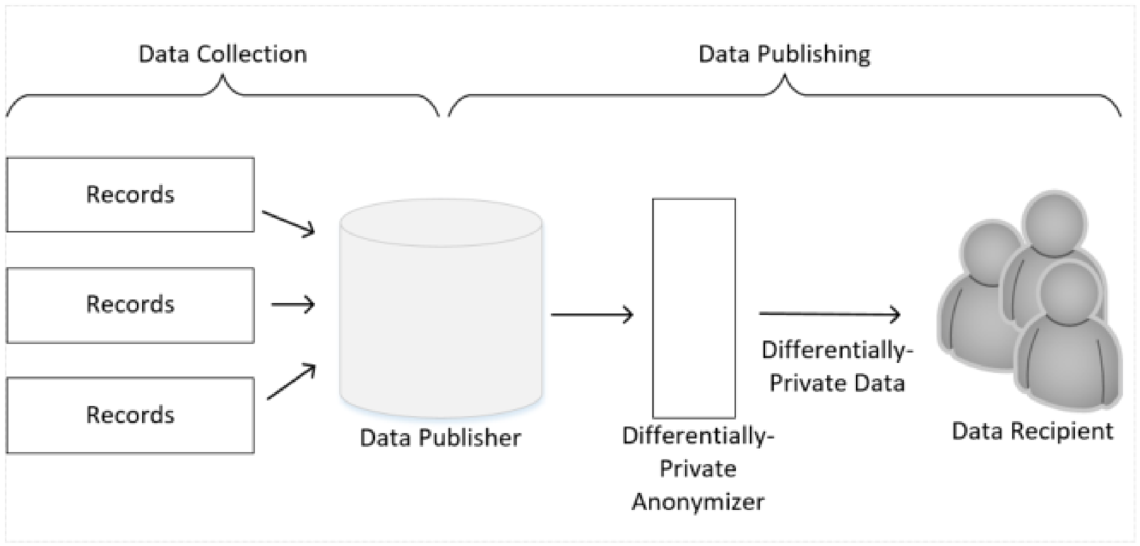


Figure 3.1: Differentially-Private Data Publishing

related to PPDP with respect to genomic data, none of them publishes a suffix tree. Our proposed approach publishes a differentially-private suffix tree, which is the most efficient data structure for genomic data search and at the same time supports effective data mining tasks. A comparative evaluation of our approach with the related work done in this field is represented in the Table 3.1.

3.1.2 Non Genomic Data

Mohammed *et al.* [62] propose a generalization-based sanitization algorithm for relational data with the goal of classification analysis. Chen *et al.* [14] propose a probabilistic top-down partitioning algorithm for set-valued data. Xiao *et al.* [92] propose a wavelet-transformation based approach for relational data to lower the magnitude of noise, rather than adding independent Laplace noise. Several available PPDP techniques, such as k-anonymity [74] [78] [49] [90] [60] [50], l-diversity [97] [55], and t-closeness [53] [76] exist, each having their own merits and based on varying assumptions about the background knowledge of an attacker. Typically, such techniques utilize one or more anonymization techniques such as generalization, bucketization, and slicing. Publishing data with k-anonymous privacy guarantees the record of an individual is indistinguishable from at least (k-1) others' as it anonymizes data by generalizing quasi identifiers. Since k-anonymity assumes that each record represents a distinct individual, it provides little privacy to a group of k records being owned by fewer than k owners. To overcome this issue in k-anonymity, (X,Y)-anonymity [83] was introduced. k-anonymity based privacy models rely on the formation of a group, but if the records in the assigned group consist of sensitive attributes having similar values, the adversary could perform an attribute linkage attack *i.e.* infer an individual's sensitive value based on the values received from the entire group and singling out the individual, thereby eliminating privacy guarantees. To avoid this, privacy-preserving models that could potentially defend against attribute linkage attacks were proposed.

One of these contributions was *l*-diversity, which guarantees privacy by mandating that every quasi-identifier group will have at least *l* sensitive attributes. In papers [55] [47] [48] and [86], techniques were proposed to achieve *l*-diversity and recursive *(c,l)*-diversity (an improvement over *l*-diversity). While *l*-diversity is a significant improvement over *k*-anonymity, it has its own drawbacks–each sensitive attribute taking values uniformly being one of them. To avoid this, other privacy models were proposed to prevent attribute linkage that could be achieved even in *l*-diversity. These include confidence bounding [85] [84], *(X,Y)*-privacy [83], (α,k)-anonymity [90], *(k,e)*-anonymity [96] and *t*-closeness [53] [76]. One of the most noteworthy of them being *t*-closeness, implemented by [53] [76] [11] [70] [52] which provides good privacy guarantee in the published data. In addition to these models, other privacy models were researched to provide privacy guarantees in cases where previous models would fail. Such as a case where an attacker may not know an individual's record in the dataset, but may confidently be able to infer the presence or absence of an individual's record in the published data. A number of other privacy models were implemented to overcome the table linkage attack and to reduce an attacker's probabilistic belief about an individual in the published data. The authors in [13] introduced the *(c,t)*-isolation privacy model, other techniques that provided privacy guarantees such as δ-presence [66] [65], (d,γ)-privacy [68] and distributional privacy [9] were employed.

Graph Data: Graph data is a dataset that employs graph structures where the nodes are connected to each other by edges and therefore possess the property to store and represent data. Based on the type of neighboring graphs to which differential privacy is applied before publishing, edge differential privacy and node differential privacy were introduced by [34]. They indicated that node differential privacy, though it provides a stronger privacy guarantee by aiming to achieve node differential privacy for publishing graph data, it may compromise the utility of the published data. Therefore, numerous research for publishing graph data intend to achieve edge differential privacy instead.

Edge Differential Privacy: Considering a graph G_1, the neighboring possible graphs include the ones in which an arbitrary edge is either added or removed compared to G_1. Achieving differential privacy, in these neighboring graphs is called edge differential privacy.

Definition 3.1.1. *[34] Edge Differential Privacy. Let* G_1 = (V_1,E_1) *and* G_2 = (V_2,E_2) *be a pair of graphs where* V_1 = V_2 *and* E_2 = E_1 - (υ_1,υ_2) *such that* (υ_1,υ_2) *is the edge they differ on. A randomized function* $\mathcal{K}$ *achieves* (ϵ,n)*–edge differential privacy, if for all graphs* G_1 *and* G_2 *differing on at most* $|(\upsilon_1,\upsilon_2)| = n$ *edges and where* S *is all the sets of answers,*

$$Pr\ [\mathcal{K}(G_1) \in S] \leqslant exp(\epsilon) \times Pr\ [\mathcal{K}(G_2) \in S]$$

where the parameter ϵ *represents the privacy budget.* □

A number of researchers have focused their research for publishing graphs via edge differential privacy; however, some research [80] also discusses the addition or removal of *n*-arbitrary edges from the graph G_1 as a representation of edge differential privacy.

Node Differential Privacy: Considering a graph G_1, the neighboring possible graphs include the graph in which a node and all the edges connected to that node are either added or removed compared to G_1. Achieving differential privacy, in these neighboring graphs is called node differential privacy.

Definition 3.1.2. *[34] Node Differential Privacy. Let G_1 = (V_1,E_1) and G_2 = (V_2,E_2) be a pair of graphs where $V_2 = V_1 - \upsilon$ and $E_2 = E_1 - (\upsilon_1,\upsilon_2) \mid \upsilon_1 = \upsilon \vee \upsilon_2 = \upsilon$ such that $\upsilon \in V_1$. A randomized function $\mathcal{K}$ achieves (ϵ,n) - node differential privacy, if for all graphs G_1 and G_2 differing on one node and it's connecting edges and where S is all the sets of answers,*

$$Pr\ [\mathcal{K}(G_1) \in S] \leqslant exp(\epsilon) \times Pr\ [\mathcal{K}(G_2) \in S]$$

where the parameter ϵ is the desired privacy budget. □

Although we have defined node differential privacy, we note that achieving node differential privacy is difficult in cases of high sensitivity, where a particular node has a connection with all other nodes in the graph, thus providing insufficient utility on the published data.

Next, we discuss various approaches for differentially-private data publishing for graph data, which guarantee edge differential privacy [63] [73] [44] [80], node differential privacy [46], or miscellaneous methods [5] [15] [91] to achieve differential privacy for publishing graph or synthetic data. In addition to data publishing, we include approaches for publishing results to queries that have not been clearly stated to be interactive or non-interactive query models, where a non-interactive query model lets the data publishers publish the results to all queries at once.

Mülle *et al.* [63] propose a graph clustering approach that guarantees (ϵ,1)-edge differential privacy. This approach does not deal with the publishing of graphs, but it is applied to publish usable graph clustering results under a differential privacy guarantee. The proposed *PIG* (privacy-integrated graph) consists of two steps: perturbation of the input graph *PIG*$_{pert}$ and a graph clustering algorithm applied to the perturbed graph. The authors introduced a *PIG* clustering approach, which perturbs the input graph by perturbing the adjacency matrix of the graph based on the value of ϵ, and applies a graph clustering approach to the perturbed graph. Perturbation consists of a combination of edge sampling and edge flipping (edge randomization), and the algorithm introduces a privacy parameter s to choose between preservation and randomization, where ($s \in [0, 1]$). Next, the approach employs graph clustering by enacting *PIG* with two graph clustering approaches: SCAN [93] and graph k-medoids [69]

Sala *et al.* [73] introduce a differentially-private graph model *Pygmalion*, to maintain a trade-off between privacy and the similarity to the original graph data while publishing it in a differentially-private manner. The authors state that the addition of significant noise to apply differential privacy might disrupt the graph structure. The partitioning approach discussed in the paper [73] provides a strong privacy guarantee while publishing a less noisy graph. The original graph is converted into $dK - 2$ degree distributions of itself, and differential privacy is applied by adding noise to these $dK - 2$ distributions. The authors develop a degree-based clustering algorithm, which partitions the statistical representation of a graph. By maintaining ϵ-differential privacy within each cluster, ϵ-differential privacy is achieved over the entire dataset. Finally, the added noise is evenly distributed, to reduce the effective errors by applying isotonic regression.

Task and Clifton [80] introduce a new differential privacy guarantee for graph data called *out-link privacy*, which provides a strong privacy guarantee when a small amount of noise is added. Here, an attacker possessing a record will not be able to determine the presence or absence of a person in the survey from which the published graph was produced. The high-degree nodes of the graph are better protected in out-link privacy than in edge privacy, as all relationships cited by a popular person are protected, and even when other nodes may be linked to a node the mutuality of the relationship can be denied. The paper introduces two techniques for implementing out-link privacy, which apply an ego-network analysis [56]. This ego-network analysis works by answering queries non-interactively, which could result in noisy data if edge or node differential privacy were applied. The ego-network analysis approach is employed to analyze social network data based on individuals whose records are present in the social network.

The first algorithm results in a dataset from which out-degree and clustering data can be extracted based on the ego-network. The second algorithm then produces the social cohesion patterns among individuals in the network. The third algorithm privatizes the centrality data, and results in a popularity graph that represents the social circles of influential individuals in the data. The authors apply Laplace noise to the edge weights when an individual adds or removes their records, based on the sensitivity. Finally, when post-processing the data the edges with low weights are eliminated, which results in a weighted popularity graph being published.

Although graph data publishing under node differential privacy guarantee has several issues concerning the utility of published data, Day *et al.* [18] propose approaches for publishing the node degree distribution of a graph using node differential privacy. The two proposed approaches each employ aggregation and a cumulative histogram for publishing the node degree distribution of a graph using graph projection. This technique ensures that the graph is θ-degree bounded, *i.e.*, when the graph is projected, a smaller value of θ indicates that more edges are pruned, while a greater value leads to the addition of more noise. The authors propose a series of algorithms and sub-algorithms for publishing degree histograms under node differential privacy. Of these, the (θ,Ω)-histogram and θ-cumulative histogram approaches are based on sensitivity bounds. For the selection of input parameters for these algorithms, the authors introduce low-sensitivity quality functions. Based on the results obtained by applying the proposed technique to real-world graph datasets, the authors conclude that this approach significantly reduces the error of approximating the degree distribution.

Ahmed *et al.* [5], propose a mechanism based on a random projection approach, which in turn employs random matrix theory to reduce the dimensions of the adjacency matrix, and therefore achieve differential privacy with the addition of less noise. Next, the authors prove that their mechanism is able to achieve differential privacy, and also list the theoretical error bounds for approximating the top k eigenvectors. Finally, the authors test the utility of the published data for two applications that require spectral information of a graph, and compare their proposed mechanism with Wang *et al.* [86], which perturbs the eigenvector of the original data directly by introducing Laplace noise. Through this comparison, the authors conclude that the performance of their mechanism is viable.

Chen *et al.* [15] claim that previous research indicated differential privacy is vulnerable to data correlation, and therefore it cannot be applied to network data that may be correlated. They claim that introducing a parameter that measures the extent of correlation could help to provide differential privacy guarantees in correlated network data. The first step is to generate a private vertical labeling for a given network dataset, to make the corresponding adjacency matrix generate clusters. These dense clusters of the adjacency matrix are identified in the next step using partitioning. Finally, the Exponential mechanism is employed to generate the noisy adjacency matrix.

Xiao *et al.* [91] introduce an approach for releasing network data from information networks in a differentially-private manner. The proposed approach is based on the observation that instead of considering the edges directly the connection probabilities between vertices are estimated, then the noise that needs to be added by differential privacy is significantly reduced. A statistical *hierarchal random graph* (HRG) [17] model is employed in the proposed model to infer the structure of the network. The proposed mechanism samples possible HRG structures using MCMC (Markov chain Monte Carlo) to guarantee differential privacy. The authors theoretically proved that the sensitivity is lower when an inference is performed on the network data published by the proposed mechanism.

However, these approaches cannot be directly applied to genomic data, as the genomic data is highly sensitive and hence requires careful processing. We use differential privacy [21] [23] [24] [16] [14] [75] [92] to address privacy-preserving genomic data publishing.

3.2 Privacy-Preserving Data Mining

In this related line of work, the raw data is first anonymized while maintaining an effective level of data utility, and then released for the purpose of data mining and analysis. The data owners jointly compute a data mining function on their private data, and only learn the correct output and nothing else. Figure 3.2 shows privacy-preserving data mining in an interactive framework. In the *data collection* phase, the data owner collects data from record owners/individuals (e.g., Jake, Bob). In the *data mining* phase, the data owner answers queries posed by data recipients and returns differentially-private results.

3.2.1 Genomic Data

There has been ample ongoing research to provide access to published/unpublished datasets while maintaining differential privacy. Other than publishing the data in a differentially-private manner, a data miner/analyst could pose queries to the data owner interactively and non-interactively. While answering non-interactive queries, the system is aware of all the queries that will be posed by the data miner in advance and it can take appropriate measures to make the data private. But in the case of interactive queries, the system would respond to the ad-hoc queries without any knowledge of the queries or any insight into the future. *Privacy-preserving query processing* is the task wherein, the queries posed over the statistical data are answered by injecting random noise to each of the responses to guarantee the privacy of an individual. This randomness hides the presence or absence of an individual in the data, while maximizing the accuracy of the responses to the posed queries. The techniques to achieve *privacy-preserving data mining (PPDM)*, *privacy-preserving data analysis* and *privacy-preserving data publishing* (discussed in this thesis) under differential privacy have been widely studied. Privacy-preserving data mining is the task of mining information from a dataset wherein the data owner is responsible for maintaining the privacy guarantees in the results sent to the data miner.

Atallah *et al.* [7] introduce a privacy preserving sequence comparison algorithm by modifying the edit-distance protocol. Szajda *et al.* [79] propose a practical data privacy scheme based on modifying the Smith Waterman sequence comparison algorithm. Johnson & Shmatikov [41] propose an algorithm to perform privacy preserving computation of the location of SNPs and p-value associated with a disease, and measures of correlations between SNPs. De Cristofaro *et al.* [19] present a private substring matching protocol to conduct a test in which no other information is learned by the conducting parties.

3.2.2 Non Genomic Data

Agrawal & Srikant [4] develop the idea of building accurate models about aggregated data without access to precise information in individual data records. The major approaches of PPDM include perturbation, anonymization, and crytptographical techniques. Hardt & Rothblum [33] introduce a differentially-private multiplicative weights mechanism application to privacy preserving data analysis. Important privacy preserving statistical analysis methods such as logistic regression have been introduced by following

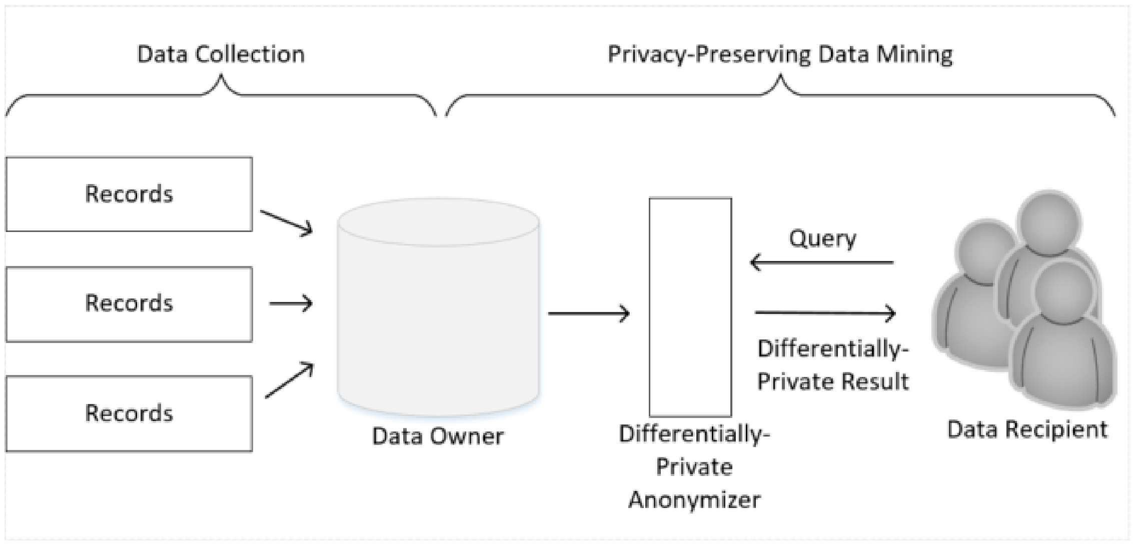

Data Collection
Privacy-Preserving Data Mining
Records
Records
Records
Data Owner
Differentially-Private Anonymizer
Query
Differentially-Private Result
Data Recipient

Figure 3.2: Privacy-Preserving Data Mining

papers [12] [94] [81]. There is much overlap in the field of PPDP and PPDM and many of the techniques can be applied to both purposes.

Table 3.1 shows the comparative evaluation of our approach with the related work done in this field. Data publishing to interested data recipients is accomplished via interactive or non-interactive frameworks. In an interactive framework, the data miners ask the query to which the data publisher responds by returning the differential private result. A certain budget is consumed for each query asked by the data miners. Therefore, data miners can only ask queries based on the budget they possess (Figure 3.2). In a non-interactive framework, the data publisher publishes the anonymized dataset and the miners can query the dataset to obtain differentially-private results. Our work is focused on privacy-preserving data publishing.

Table 3.1: Comparative evaluation of main features in related approaches including our proposed approach (properties in columns are positioned as beneficial with fulfillment denoted by ●)

Approach	**Privacy Model**		**Domain**		**Data Release**			
	Differential Privacy	**Syntactic Privacy**	**PPDP**	**PPDM**	**Suffix Tree**	**Hierarchical**	**Sparse Vector**	**Others**
Wang *et al.* [88]	●		●			●		
Roozgard*et al.* [72]	●		●				●	
Uhler *et al.* [81]	●		●					●
Yu *et al.* [95]	●		●					●
Jiang *et al.* [40]	●		●					●
Atallah *et al.* [7]		●		●				●
Szajda *et al.* [79]		●		●				●
Johnson and Shmatikov [42]	●			●				●
Cristofaro *et al.* [19]		●		●				●
Our proposed solution [Chapter 4]	●		●		●			

Chapter 4

PROPOSED ALGORITHM

Let $\mathcal{D}$ is the genomic data that contains n sequences $\{S_1, \ldots, S_n\}$, where each sequence contains a number of SNPs, for example, $S_1 = AGAT$. To achieve differential privacy, the principal approach is to perturb the true output by adding a random noise to it. To generate the noise according to Laplace distribution we need to determine the privacy budget B beforehand. The suffix tree is one of the most important and widely used data structures in bio-informatics and comparative genomics. It is a special data structure with a wide range of applications, including exact matching problems, substring problems, data compression and circular strings. A suffix tree is crucially important in sequencing and investigating DNA, such as looking for the longest common substring of two DNA sequences and finding exact and inexact matchings of a sample in a long sequence. In computational biology, molecular biology and bio-informatics these problems are crucially important. In our approach, we aim to generate a differentially-private suffix tree $\mathcal{T}$ that will support count queries. Count queries, as a general data analysis task, are the building block of many data mining tasks. We denote by *user count query* any data mining's count query that attempts to answer the following question: how many times a specific pattern u occurs in the data, i.e., suffix-tree $\mathcal{T}$?

In this chapter, we first present an overview of our proposed privacy-preserving approach for publishing a differentially-private genomic suffix tree. We then elaborate the key steps in each algorithm. The objective of our solution is to publish a suffix tree from raw genomic data using differential privacy while maintaining utility. Given a genomic data $\mathcal{D}$ containing sequential genomic data sequences $\{S_1, \ldots, S_n\}$ and given a privacy budget B and height h, our approach constructs a partitioning tree $\mathcal{P}$ with height h, normalizes it using a bottom-up approach, and then generates a differentially-private suffix tree $\mathcal{T}$. Our solution consists of four algorithms:

Algorithm 1 - Differentially-Private Genomic Data Publishing: It is the main algorithm that calls the algorithms 2, 3 and 4 to construct a differentially-private suffix tree from the genomic data $\mathcal{D}$. To ensure privacy and an efficient search for genomic data, a differentially-private suffix tree using Laplace mechanism is constructed over the raw genomic data.

Algorithm 2 - Differentially-Private Partitioning Tree Generation: This algorithm efficiently generates suffixes of raw genomic data sequences by constructing a differentially-private partitioning tree.

Algorithm 3 - Bottom-Up Normalization: To ensure the consistency and utility of the output genomic data, a bottom-up approach is used to normalize the partitioning tree based on a proposed utility constraint.

Algorithm 4 - Suffix Tree Generation: This algorithm uses a top-down approach to generate a B-differentially-private suffix tree from the normalized partitioning tree to support count queries for genomic sequence matching.

4.1 Differentially-Private Genomic Data Publishing

This main algorithm takes as input a raw genomic data $\mathcal{D}$, which contains sequences $\{S_1, \ldots, S_n\}$ such that the length of i-th sequence $|S_i| = k_i$, a privacy budget B and a user-specified height h, and returns a suffix tree $\mathcal{T}$ satisfying B-differential privacy, as shown in Algorithm 1. In Step 1, it calls Algorithm 2 to construct a noisy partitioning tree $\mathcal{P}$ using Laplace noise and the specialization threshold θ. In Step 2, Algorithm 3 is called, which takes as input a differentially-private partitioning tree $\mathcal{P}$, and normalizes $\mathcal{P}$ based on the utility constraint to maintain the usefulness of the outputted normalized partitioning tree $\hat{\mathcal{P}}$. In Step 3, Algorithm 4 is executed to construct a differentially-private suffix tree $\mathcal{T}$ based on the differentially-private suffixes obtained from the normalized partitioning tree $\hat{\mathcal{P}}$. In Step 4, the algorithm outputs a differentially-private suffix tree $\mathcal{T}$.

Algorithm 1 Differentially-Private Genomic Data Publishing

```
Differentially-Private Genomic Data Publishing Algorithm

Input: Genomic Data $\mathcal{D} = \{S_1, \ldots, S_n\}$
Input: Privacy Budget $B$
Input: Height of the tree $h$
Output: Differentially-private suffix tree $\mathcal{T}$

  1. Execute Algorithm 2 to construct a partitioning tree $\mathcal{P}$ based on $\mathcal{D}$ and $B$.
  2. Execute Algorithm 3 to normalize $\mathcal{P}$ according to the utility constraint 4.3.1
  3. Execute Algorithm 4 to construct a differentially-private suffix tree $\mathcal{T}$ from $\hat{\mathcal{P}}$.
  4. Return suffix tree $\mathcal{T}$.
```

4.2 Partitioning Tree Generation

In this phase, our approach is to generate differentially-private suffixes to ensure privacy-preserving data publishing. Our strategy for generating suffixes in a differentially-private manner is to use a top-down approach based on constructing disjoint partitions for multiple levels using Laplace mechanism and the specialization threshold θ. Therefore, we construct a partitioning tree $\mathcal{P}$ by recursively grouping sequences in $\mathcal{D}$ into disjoint sub-datasets based on their suffixes. Given a raw genomic data $\mathcal{D} = \{S_1, \ldots, S_n\}$, privacy budget B and user specified height *h*, the algorithm generates a differentially-private partitioning tree $\mathcal{P}$. Each partition contains four values: the nucleotide type, the suffix, sequences containing the suffix and the noisy count associated with it as described in Definition 4.2.1. In Step 1, we compute the specialization threshold $\theta = c{*}2\sqrt{2h/B_l}$ (two times the standard deviation of noise) [14], where *c* is a constant that will be determined through experiments, h is the height of the tree and B_l is the privacy budget per level.

Step 2 creates a virtual root partition *r*, where a partition is a data structure formally defined as follows.

Definition 4.2.1. *Partition.* A partition is a tuple with four values [*Nuc, UNuc, Seqs, NCount*], where:

- Nucleotide (*Nuc*) is a type of bases - adenine, guanine, thymine, and cytosine *A, G, T, C* in a strand of DNA.
- *UNuc* keeps track of the suffixes of *Seqs* = $\{S_1, \ldots, S_n\}$ for each child partition.
- *Seqs* = $\{S_1, \ldots, S_n\}$ is the set of sequences that ends with the suffix *UNuc*.
- *NCount* is the noisy count which is the summation of count of sequences *Seqs* and Laplace noise. □

All sequences in $\mathcal{D}$ are initially assigned to $r.Seqs$, $r.UNuc$ is set to *Any*, and *r.UNuc* and *r.NCount* are set to *Null*. In Step 3, for each nucleotide type *A, G, T* and *C*, we create child partition v for r. For each child partition v, we assign $v.Nuc$ as the nucleotide type and $v.UNuc$ as $v.Nuc \cup Parent(v).UNuc$. For each sequence, if $v.UNuc$ is a suffix of

S, then we assign it to $v.Seqs$. The variance of the Laplace noise is $2 * \Delta f/(B_l - \sqrt{B_l})$, where Δf is the sensitivity and $(B_l - \sqrt{B_l})$ is the privacy budget used for partitioning. The sensitivity Δf in our algorithm is upper bounded by *h* i.e., the height of the tree. The noisy count associated with each partition is the sum of the true count and the Laplace noise. To build $\mathcal{P}$, we use a uniform budget allocation scheme. We divide the total privacy budget B equally, i.e., the privacy budget used per level for constructing $\mathcal{P}$ is $B_l = B/h$. One important observation is that all partitions at the same level contain disjoint sequence sets, and therefore the privacy budget allocated to a level can be used in full for each partition in it. In Step 5, for each child partition v created in Step 3, we continue to create further child partitions if $v.Ncount \geq \theta$ for that partition and the height $h > 0$. Step 6 outputs a differentially-private partitioning tree $\mathcal{P}$.

Example 4.2.1. Consider the sequences $\{S_1, ..., S_6\}$ shown in Figure 1.1 as the input sequences. Initially the algorithm creates one root partition *r*, where $r.Nuc$ is set to Any, $r.UNuc$ is set to $null$, $r.Seqs$ contains all the input sequences $S_1, S_2, S_3, S_4, S_5, S_6$ and $r.NCount$ is set to $null$. At level 1, the algorithm creates four new child partitions A, G, T and C of r, as shown in Figure 4.1. The sequences with corresponding suffixes will be assigned to the respective partitions. As none of the sequences has A or C as the suffix, therefore, the partitions with suffixes A and C will be assigned as $null$ in $Seqs$, the partition with suffix G will contain S_2, S_4, S_5, S_6 and the partition with suffix T will contain S_1, S_3. For level 1, $\theta = 35$ and $NCount$ for the partition with suffix G is $40 \geq 35$, therefore we will further create the child partitions for G. Similarly, depending on the noisy count of each partition, we decide whether to create further partitions or not until $h > 0$. $\square$

Privacy Budget Allocation: B-differential privacy can be achieved by applying a differentially-private mechanism, commonly Laplace mechanism [23], that consumes a

Algorithm 2 Differentially-Private Partitioning Tree Generation

```
Differentially-Private Partitioning Tree Generation Algorithm

Input: Genomic Data $\mathcal{D} = \{S_1, \ldots, S_n\}$
Input: Privacy Budget $B$
Input: Height of the tree $h$
Output: Differentially-private partitioning tree $\mathcal{P}$

1. Compute the specialization threshold $\theta = c * 2\sqrt{2h/B_l}$, where $h$ is the height of the tree.
2. Construct a partitioning tree $\mathcal{P}$ with a virtual root partition r:
    (a) Assign all sequences in $\mathcal{D}$ to $r$: $r.Seqs = \{S_1, \ldots, S_n\}$
    (b) Set nucleotide $r.Nuc$ to Any.
    (c) Set $r.UNuc$ and noisy count $r.NCount$ to Null.
3. For each nucleotide type $A, C, G$ and $T$, create a child partition $v$:
    (a) Set $v.Nuc$ to the nucleotide type.
    (b) Set $v.UNuc$ to $v.Nuc \cup Parent(v).UNuc$.
    (c) For each sequence $S$ in $Parent(v).Seqs$, assign $S$ to $v.Seqs$ iff $v.UNuc$ is a suffix of $S$.
    (d) Generate Laplace noise $L_{noise} = Lap(2 \times h/(B_l - \sqrt{B_l}))$.
    (e) Set $v.NCount$ to $|v.Seqs| + L_{noise}$, where $|v.Seqs|$ is the number of sequences assigned to partition $v$.
4. $h \leftarrow h - 1$.
5. While $h > 0$, for each child partition $v$ created in Step 3, if $v.NCount \geq \theta$, then repeat Steps 3 and 4.
6. Return $\mathcal{P}$.
```

privacy budget B and calibrates noise according to the global sensitivity $\Delta(f)$ of a function f. To address the existing utility and privacy trade-off, a geometric mechanism [27] was proposed which is a factor of an optimal mechanism $\mathcal{M}_u$ that is user-independent for every user u [10]. As in our approach, we will be normalizing the generated partitioning tree $\mathcal{P}$ so that leaf nodes would be least noisy, therefore, to construct partitions at each level, we employ a uniform budget allocation scheme. That is, the privacy budget allocated to each level l is $B_l = B/h$, which is used for constructing partitions at level l in $\mathcal{P}$, as well as to compute the specialization threshold θ.

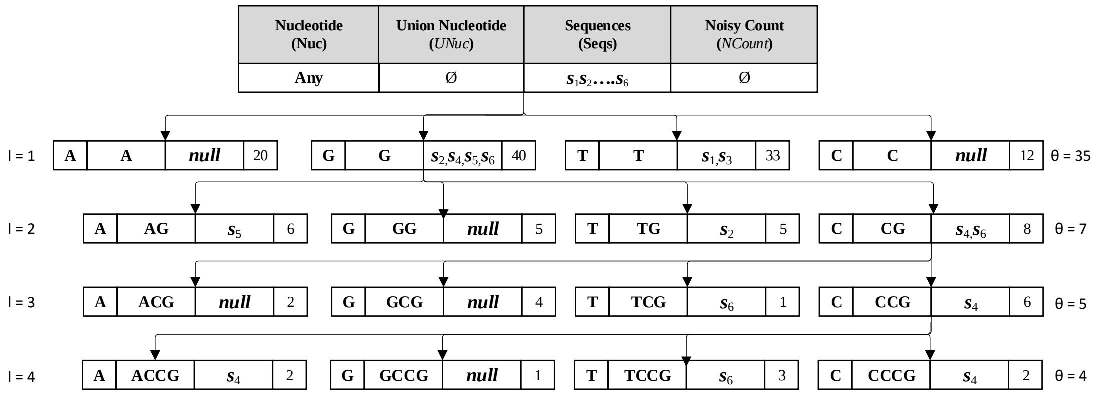


Figure 4.1: Partitioning Tree: At level 1, the algorithm creates four new partitions A, G, T and C of the root partition. The partition with suffix G contains S_2, S_4, S_5, S_6 and the partition with suffix T contains S_1, S_3. At level 1, since θ is 35, and $NCount$ for partition with suffix G is 40, which is $\geq$ 35, therefore we create the child partition for partition with suffix G. Similarly, we create child partitions until the height $h > 0$.

Sensitivity Analysis: The sensitivity of a function is defined as the maximum difference in the function output when one single element in the function domain is modified. If sensitivity is low, that means, the presence or absence of an individual will change the outcome very little, so less noise needs to be added in order to ensure privacy and vice-versa.

Lemma 3. *The sensitivity of our algorithm is upper bounded by h.*

Proof: In the output suffix tree, if a sequence is modified, which means we add, delete or replace one or more nucleotide; then the maximum number of suffixes in a suffix tree that will be affected is at most k, where k is the length of the sequence. However, the maximum length of a sequence that we will be inserting in a final suffix tree based on our algorithm is h, where h is the height of the tree. Therefore, the overall sensitivity of our algorithm is upper bounded by h. □

4.3 Bottom-Up Normalization

To ensure the utility (usefulness) and consistency of the data, a bottom-up approach is proposed to normalize the partitioning tree based on the following utility constraint.

Definition 4.3.1. *Utility Constraint.* The noisy count $NCount$ of any node v in the partitioning tree $\mathcal{P}$ should be greater or equal to the total sum of its children's noisy counts. That is:

$\forall v \in \mathcal{P},\ v.NCount \geq \sum_{u \in child(v)} u.NCount$ □

Algorithm 3 takes the differentially-private partitioning tree $\mathcal{P}$ as an input and updates the noisy count $NCount$ of the partitions from the leaf to the root to generate a normalized partitioning tree $\hat{\mathcal{P}}$. Following the bottom-up approach, we start with level $l = h - 1$, where h is the height of the differentially-private partitioning tree $\mathcal{P}$. The algorithm ensures that for each non-leaf node in $\mathcal{P}$, the utility constraint holds true.

If the utility constraint described in Definition 4.3.1 is not satisfied, that is, the noisy count of the parent is less than the sum of the noisy count of its children, then the parent's noisy count is updated to be the sum of the noisy count of all its children, as described in Step 2. We keep normalizing based on the utility constraint until we reach the root partition

Algorithm 3 Bottom-Up Normalization

```
Bottom-Up Normalization Algorithm

Input: Differentially-private partitioning tree $\mathcal{P}$
Output: Normalized partitioning tree $\hat{\mathcal{P}}$

Apply a bottom-up approach on $\mathcal{P}$ to obtain a normalize partitioning tree $\hat{\mathcal{P}}$:

    1. $l \leftarrow h - 1$, where $h$ is the height of $\mathcal{P}$.
    2. For each non-leaf node $v$ in $\mathcal{P}$ at level $l$, if the utility constraint 4.3.1 is not
       satisfied, then update the noisy count of $v$:
       $v.NCount \leftarrow \sum_{u \in child(v)} u.NCount$.
    3. $l \leftarrow l - 1$.
    4. While $l > 0$, repeat Steps 2 and 3.
    5. Return $\hat{\mathcal{P}}$.
```

of the tree. In Step 5, a differentially-private normalized partitioning tree $\hat{\mathcal{P}}$ is produced as an output.

Example 4.3.1. Consider the partitioning tree $\mathcal{P}$ in Figure 4.1. At level 3, the partition with suffix CCG has $NCount = 6$ and the sum of the noisy count of its children is 8. As this violates the utility constraint, we update $NCount$ of the partition with suffix CCG to $8 = 2 + 1 + 3 + 2$ and similarly, we update $NCount$ of the partition with suffix CG to $15 = 2 + 4 + 1 + 8$ as shown in Figure 4.2. We continue to normalize further till we reach the root partition. $\square$

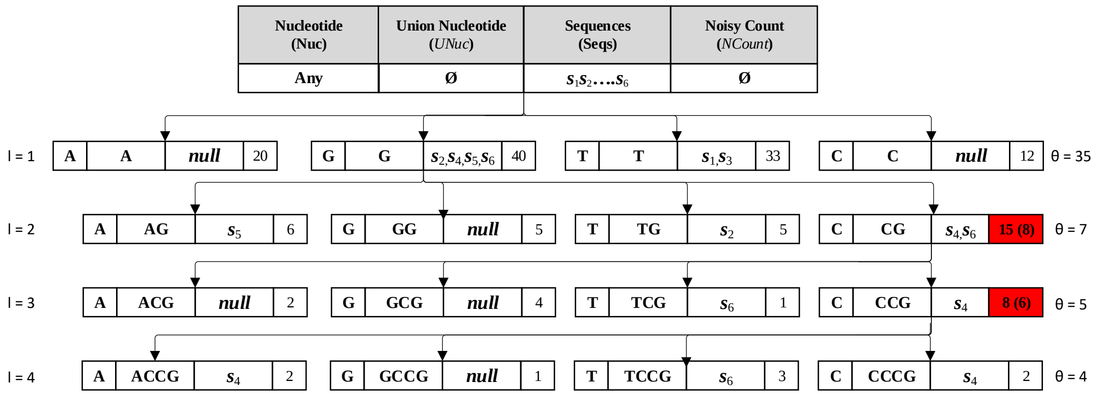


Figure 4.2: Normalized Partitioning Tree: At level 3 in Figure 4.1, the partition with suffix CCG has $NCount = 6$ and the sum of the noisy count of its children is 8. As this violates the utility constraint, we update $NCount$ of the partition with suffix CCG to $8 = 2 + 1 + 3 + 2$ and similarly, we update $NCount$ of the partition with suffix CG to $15 = 2 + 4 + 1 + 8$ as shown in Figure 4.2. We continue to normalize further following bottom-up approach until level l> 0.

4.4 Suffix Tree Generation

This algorithm takes as an input the normalized partitioning tree $\hat{\mathcal{P}}$, and based on a top-down approach it outputs the B-differentially-private suffix tree $\mathcal{T}$. In the normalized partitioning tree, $U.Nuc$ represents the suffix and $NCount$ represents the normalized noisy count. We traverse the normalized partitioning tree level-wise and create a branch in the suffix tree for each suffix present in $\hat{\mathcal{P}}$. Starting with level 1, for each partition v in $\hat{\mathcal{P}}$ as depicted in Figure 4.2, we use the suffix present in $v.UNuc$ and its corresponding $v.NCount$ to create a tree branch in the suffix tree $\mathcal{T}$, as shown in Figure 4.3, and add the associated noisy count at the end. For each added branch, we append the dollar sign \$ equal to the noisy count $v.NCount$. Every path from the root to the square node represents a suffix. Because the tree is normalized, there is no violation of utility constraint when we take the parent suffix and add it to the suffix tree i.e., following a top-down approach. Since, the noisy count of the parent partition is always greater than or equal to the noisy count of its

Algorithm 4 Suffix Tree Generation

```
Suffix Tree Generation Algorithm

Input: Normalized partitioning tree $\hat{\mathcal{P}}$
Output: Differentially-private suffix tree $\mathcal{T}$

Apply a top-down approach on $\hat{\mathcal{P}}$ to obtain a differentially-private suffix tree $\mathcal{T}$:

  1. Set the initial level: $l \leftarrow 1$.
  2. For each node $v$ in $\hat{\mathcal{P}}$ at level $l$:
      (a) Use $v.UNuc$ sequence to create a tree branch in $\mathcal{T}$.
      (b) Append $v.NCount$ dollar signs $ to the added branch.
  3. $l \leftarrow l + 1$.
  4. While $l \leq h$, where $h$ is the height of $\hat{\mathcal{P}}$, then repeat Steps 2 and 3.
  5. Return $\mathcal{T}$.
```

children, the suffixes maintain consistency. Step 5 outputs a differentially-private suffix tree $\mathcal{T}$ that supports count queries for genomic sequence matching and maintains data utility.

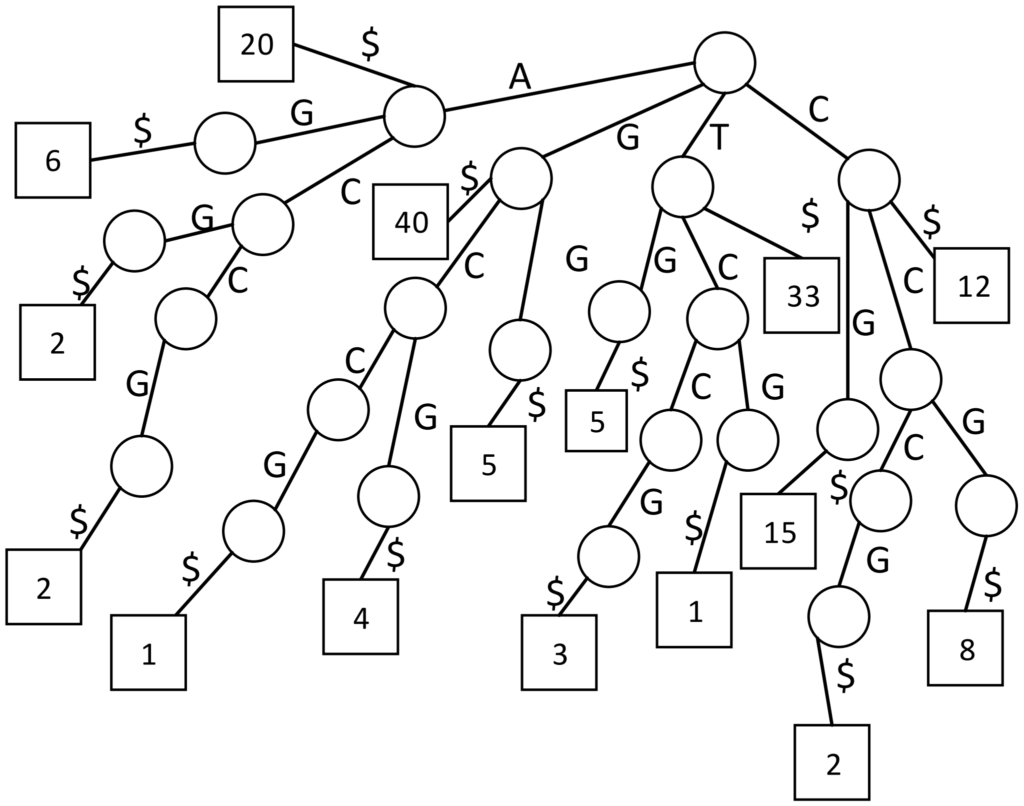


Figure 4.3: Differentially-Private Suffix Tree: It is generated from normalized partitioning tree $\hat{\mathcal{P}}$ shown in Figure 4.2

Chapter 5

ALGORITHM ANALYSIS

In this chapter, we perform the privacy and complexity analysis of our proposed approach.

5.1 Privacy Analysis

We use the composition properties of differential privacy to guarantee that the proposed algorithm satisfies B-differential privacy as a whole. Any sequence of computations that each provides differential privacy in isolation also provides differential privacy in sequence, which is known as *sequential composition*.

Lemma 4. *Sequential composition [59]. Let each A_i provide B_i-differential privacy. A sequence of $A_i(\mathcal{D})$ over the dataset $\mathcal{D}$ provides ($\sum_i B_i$)-differential privacy.* $\square$

However, if the sequence of computations is conducted on disjoint datasets, the privacy cost does not accumulate but depends on only the worst guarantee of all the compositions. This is known as *parallel composition*.

Lemma 5. *Parallel composition [59]. Let each A_i provide B_i-differential privacy. A sequence of $A_i(\mathcal{D})$ over a set of disjoint datasets $\mathcal{D}$ provides B_i-differential privacy.* $\square$

Proposition 5.1.1. *Given a privacy budget B, our algorithm generates a B-differentially-private genomic suffix tree.*

Proof: Our proposed algorithm consists of three phases - *partitioning tree generation*, *bottom-up normalization* and *suffix tree generation*. We have a fanout $f = 4$. According to the Lemma 5, we can use the same privacy budget for each partition at the same level. Therefore, if we can prove that the summation of the privacy budget used in each partitioning level is less than or equal to B, we get the conclusion that our approach satisfies B-differential privacy.

In the partitioning tree generation phase of our algorithm, the allocation of the privacy budget per level is $B_l = B/h$, where h is the height of the partitioning tree $\mathcal{P}$. The privacy budget needed for computing the specialization threshold θ is $\sqrt{B_l}$. As all the partitions on the same level in the partitioning contain disjoint sets of sequences, according to the Lemma 4, the total privacy budget for each partition to build the noisy partitioning tree $\mathcal{P}$ is $B_l - \sqrt{B_l}$. The only time we refer to the original data is when we compute the total count of the sequences in each partition, and as mentioned above, for each partition we compute $B_l - \sqrt{B_l}$ privacy budget. For the bottom-up normalization and the suffix tree generation phase, no privacy budget is consumed as we are not utilizing the original data. Therefore, the total privacy budget consumed can be formulated as:

$$\sum_{l=1}^{h} \underbrace{(B_l - \sqrt{B_l})}_{\text{partition}} + \underbrace{\sqrt{B_l}}_{\text{threshold}} = h \times B_l \leq B$$

As proven by Hay *et al.* [35], a post-processing of differentially-private results remains differential private. Therefore, our approach satisfies B-differential privacy. □

5.2 Complexity Analysis

Proposition 5.2.1. ***(Complexity).*** *The overall complexity of our proposed approach in the average case is* $\mathcal{O}(h^2 \times n)$.

Proof. We can determine the time complexity of the proposed approach in terms of three phases: Partitioning tree generation, Normalization and Suffix tree generation.

Partitioning tree generation phase. In the partitioning tree generation phase, the runtime complexity is: the cost of generating each partition, times the number of partitions in the tree. For each level l of the partitioning tree, the maximum number of possible partitions in the worst case is 4^l, and the total number is $\sum_{l=1}^{h} 4^l$. However, the use of the specialization threshold θ restricts the exponential growth of the partitions; θ fluctuates according to the Laplace noise distribution, which balances the number of partitions [25]. Therefore, in the average case, the number of partitions per level is $P_l \approx 4 \times l \ll 4^l$, and the total number of partitions is:

$$\sum_{l=1}^{h} 4 \times l = 2 \times h^2$$

Given that we process n sequences per level, the total cost of constructing a partitioning tree in an average case is $\mathcal{O}(n \times h^2)$, where, n is the number of sequences and h is the height of the partitioning tree.

Normalization phase. In the normalization phase, we traverse the partitioning tree once. Therefore, the total cost to construct a normalized partitioning tree in an average case is: $\sum_{l=1}^{h} 4 \times l = 2 \times h^2 = \mathcal{O}(h^2)$.

Suffix tree generation phase. In the suffix tree generation phase, the runtime complexity is: the time taken to insert a full sequence, times the number of sequences generated. As stated by [82] [89] [29], a string of length *h* can be inserted in $\mathcal{O}(h)$. In the average case, the number of full sequences to be generated from the partitioning tree is $\mathcal{O}(h)$, where these full sequences will account for all the trimmed sequences (short inbuilt suffixes). Therefore, the average computational cost to build a suffix tree is $\mathcal{O}(h \times h) = \mathcal{O}(h^2)$.

Therefore, the overall complexity of our proposed approach in the average case is: $\mathcal{O}(h^2 \times n + h^2 + h^2) = \mathcal{O}(h^2 \times n)$. $\square$

Given that $h \ll n$, we show in our experiment that our algorithm scales linearly w.r.t. a linear increase of n.

Chapter 6

EXPERIMENTAL EVALUATION

In this chapter, we evaluate the performance of our algorithm. First, we discuss the implementation details, and then we present the experimental results that include determining the optimal value of constant c for computing θ, scalability of query processing with respect to the number of input sequences, efficiency with respect to the privacy budget B and the utility with respect to the percentage of the query length. We also perform a comparative evaluation of our proposed algorithm with differentially-private genome data dissemination through top-down specialization algorithm [88]. We implemented our algorithm in Java, and our experiments were conducted on a machine equipped with an Intel(R) Core(TM) i7-6700 CPU @ 3.40GHz processor and 32.0 GB RAM, running Windows 10 64-bit operating system.

6.1 Datasets

The humane SNPs were obtained from the Human Genome Privacy Challenge [3]. The chr2 and chr10 datasets contain 311 SNPs and 610 SNPs respectively. The length of the sequences is 200. The details of the datasets is provided in Table 6.1.

Table 6.1: Properties of Experimental Datasets

Dataset	**# of SNP Sequences**	**Length of Sequence**
chr2	311 (29504091-30044866)	200
chr10	610 (55127312-56292137)	200

6.2 Determining Optimal Value of Constant c

Recall that $\theta = c * 2\sqrt{2h/B_l}$. To determine the optimal value of constant c for the specialization threshold θ, we randomly generate queries of varying length from the suffix tree and experiment the data utility for different c values. We started experimenting with a $c > 0.25$, and noticed that θ was too big and the condition to partition further was never satisfied. Therefore, we experimented on c values from the range [0, 0.25]. As shown in Figure 6.1, the best results in terms of average relative error is for $c = 0.15$.

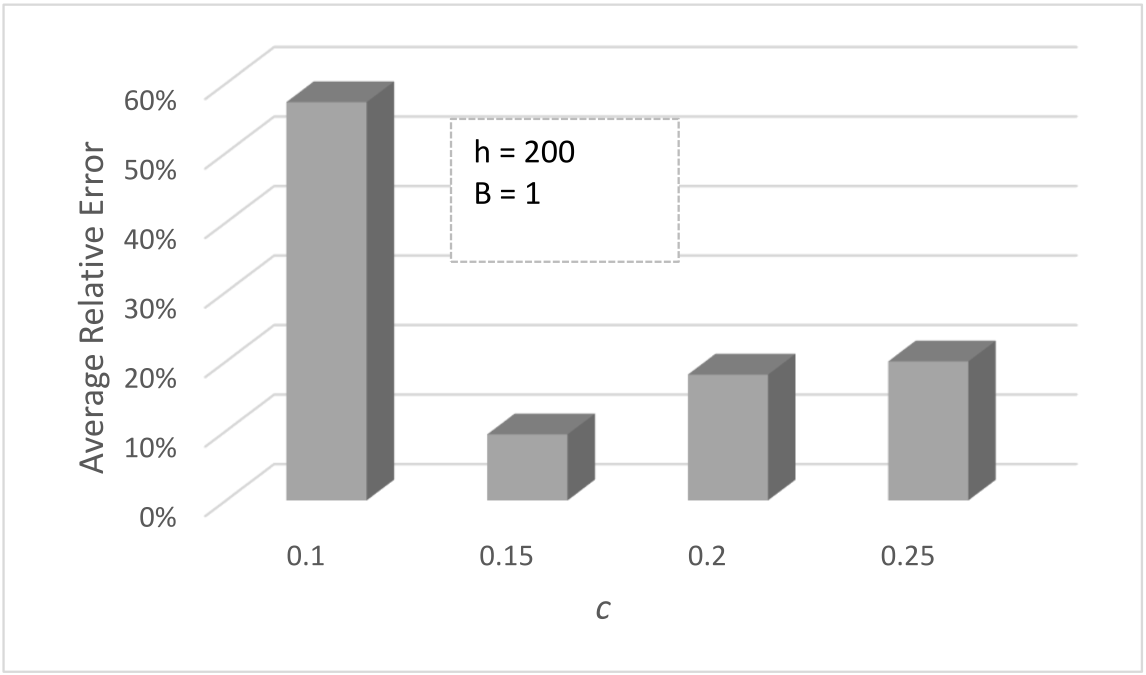


Figure 6.1: Determining the optimal value of c for computing the threshold θ

6.3 Scalability

The objective is to measure the runtime of each construction phase in order to ensure its capability to scale up in terms of the number of sequences. We measure the runtime of our proposed algorithm with respect to the linear increase in the number of sequences. We set the privacy budget B to 1, the height of the tree to 200 and the value of c as 0.15. Figure 6.2 illustrates the runtime of our algorithm with respect to a linear increase in the number of sequences in the randomized dataset (200k, 400k, 600k, 800k and 1000k records). The x-axis represents the number of records and the y-axis shows the runtime in minutes. We observe that the growth in total runtime is linear when the data records increase linearly. We also observe that the partitioning tree construction phase is the most dominant phase in the algorithm. However, all the three phases scale linearly with respect to the number of sequences.

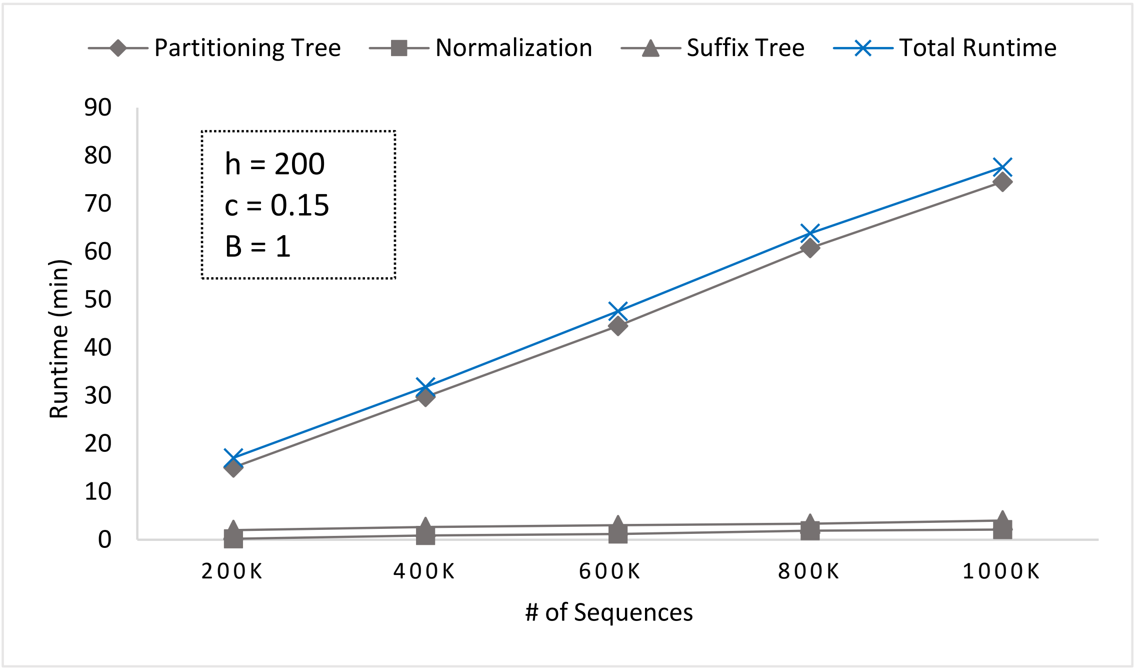


Figure 6.2: Scalability w.r.t the number of sequences

6.4 Efficiency

We evaluate the efficiency of our proposed approach on the chr2 and chr10 dataset with respect to the privacy budget B varying from 0.5 to 1.25 at an interval of 0.25. The x-axis represents the privacy budget B and the y-axis represents the runtime in milliseconds as shown in Figure 6.3. We observe that for the chr2 dataset the runtime is between 200-300 milliseconds and for the chr10 dataset the runtime is between 600-700 milliseconds. However, the runtime is consistent with the increase in B for both the datasets.

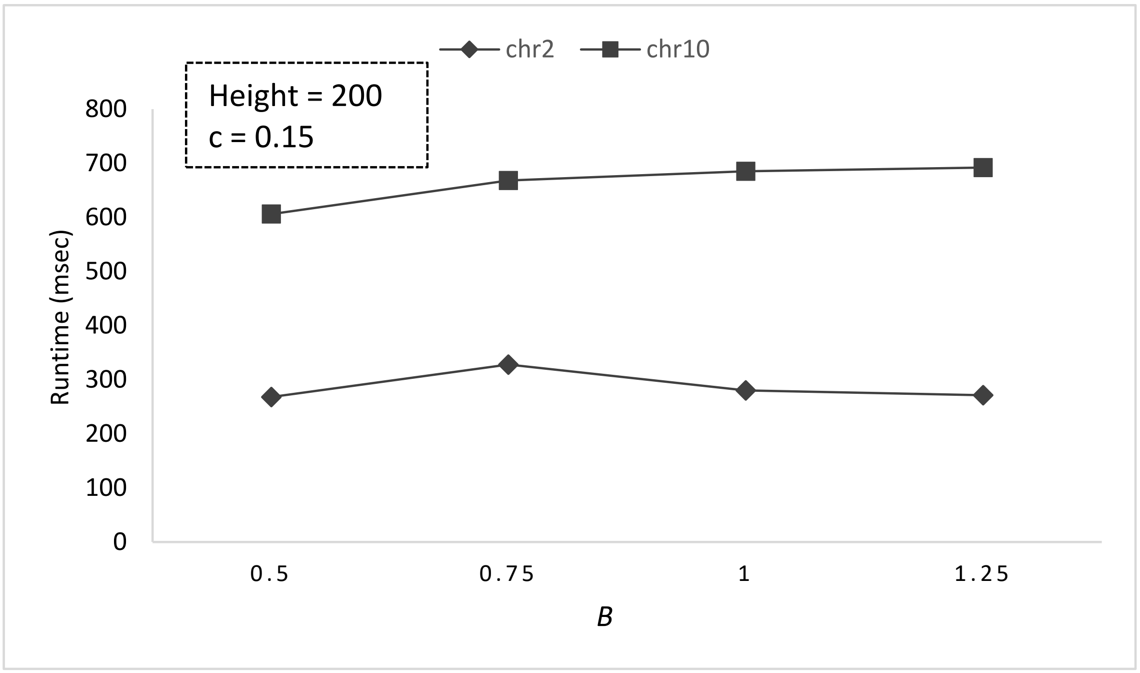


Figure 6.3: Efficiency w.r.t the privacy budget B

6.5 Utility

We measure the utility of a count query Q over the sanitized dataset $\tilde{\mathcal{D}}$ by its *relative error* with respect to the actual result over the raw dataset $\mathcal{D}$. The relative error is computed as:

$$\textit{Relative Error} = \frac{|Q(\tilde{\mathcal{D}}) - Q(\mathcal{D})|}{max(Q(\mathcal{D}), s)} \tag{6.1}$$

where s is a sanity bound that is set to 1.

We randomly generate 500 counting queries with varying length of sequences. We divide the query set into five subsets such that the query length of the i-th subset is uniformly distributed in $[1, \frac{i}{5}\sqrt{k}]$ and each query is drawn randomly, where $k = 200$ is the length of the sequence . All relative errors will be computed based on the average of 10 runs.

The experimental results are generated using both datasets: chr2 and chr10. We use 500 random queries of length $l \in [1, 14]$, since $i = 5$ in our experiments. The first subset for 20% of query length is $[1, \frac{1}{5}\sqrt{200}]$, which is equal to [1,3]. Similarly, the five uniformly distributed subsets for 20%, 40%, 60%, 80% and 100% of query length are $[1, 3], [1, 6], [1, 8], [1, 11], [1, 14]$ respectively. We select five random queries for each sampling and the relative error is the average of 10 runs.

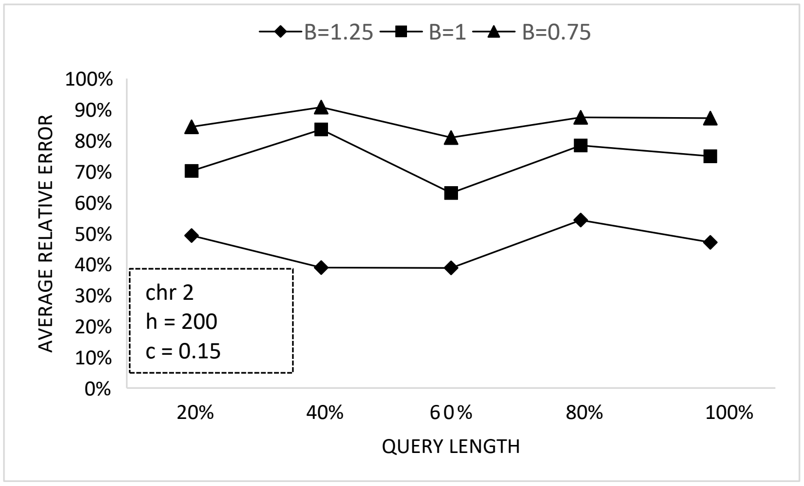
Figures 6.4 and 6.5 depict the average relative error as the value of the privacy budget B grows from 0.75 to 1.25 at an interval 0.25, and height is set to 100, 150 and 200 with $c = 0.15$. The x-axis represents the maximum query length of each subset in terms of the percentage of$\sqrt{k}$ and the y-axis represents the average relative error. We observe that the average relative error decreases with the increase in B with respect to both the datasets, and the best data utility is obtained for the privacy budget $B = 1.25$. This observation remains true irrespective of the height h of the tree, as shown in both the figures for $h \in \{100, 150, 200\}$. This is due to the fact that as we increase the privacy budget, we are decreasing the noise that is being added and hence the data utility is increased. Also, we observe that in the chr2 dataset (the smaller dataset), the average relative error is almost constant for each B and h, regardless of the query length. On the other hand, in the chr10 dataset, given B and h, the average relative error is variable depending on the query length. This is due to the fact that in the larger dataset the size of the data grows, and more noise is added to ensure privacy since the noisy count of more partitions will surpass the threshold θ. Therefore, we conclude that as the data grows, the utility increases but it becomes sensitive to the query length.

We also compare the performance of our proposed algorithm with the research work done by Wang *et al.* [88] for the differentially-private genome data dissemination through top-down specialization. We use the same dataset as in [88], and make the experimental settings as close as possible. We set the privacy budget $B = 1$, and determine the accuracy of our algorithm as follows:

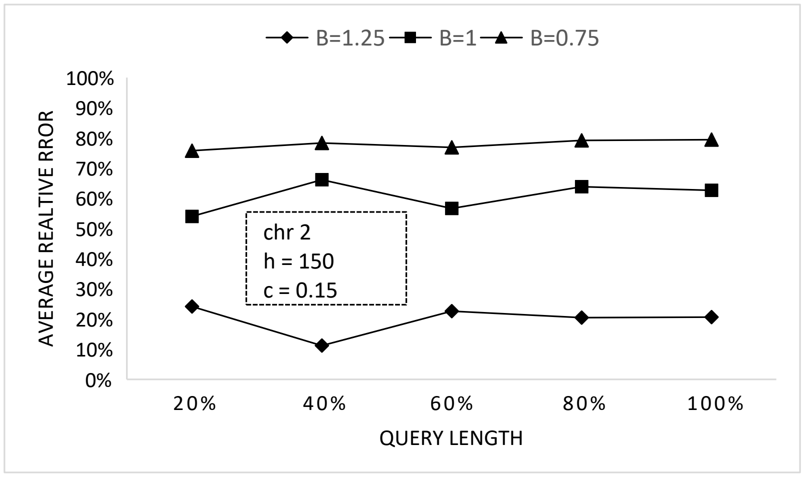
$$\textit{Accuracy} = 1 - \textit{Average Relative Error} \tag{6.2}$$

where the relative error is computed using Equation 6.1.

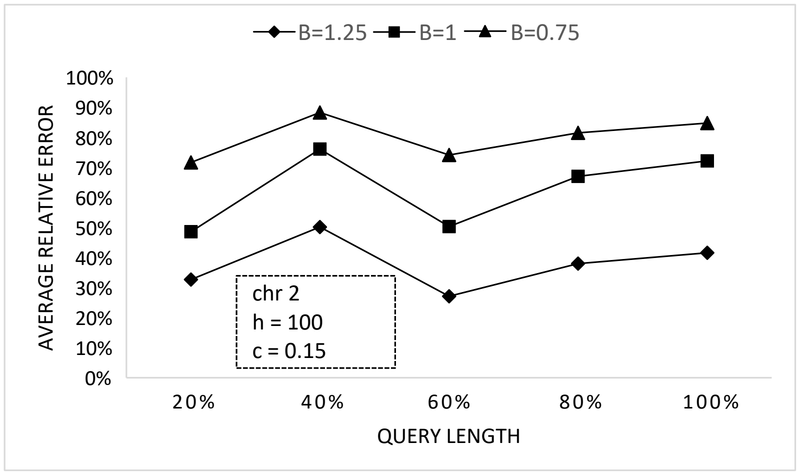
Figure 6.6 illustrates that for both the datasets chr2 and ch10, our proposed algorithm provides higher accuracy than [88]. That is, for $B = 1$, both the algorithms achieve better accuracy on the larger dataset (chr10), however our solution achieves a higher accuracy of up to 70%.



(a)

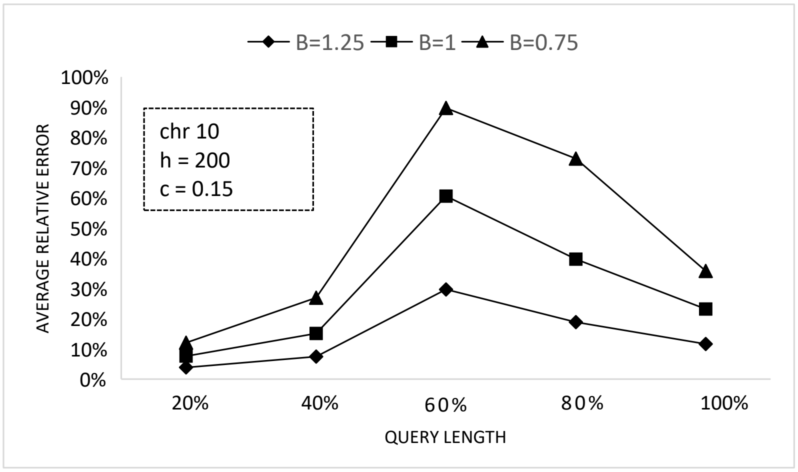


(b)

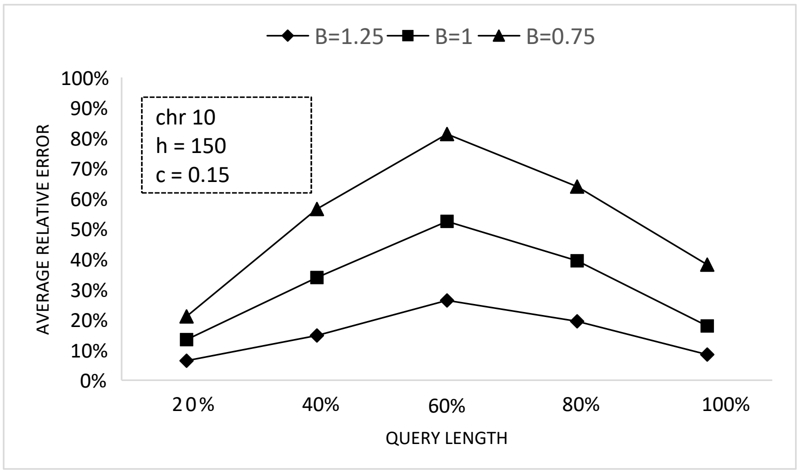


(c)

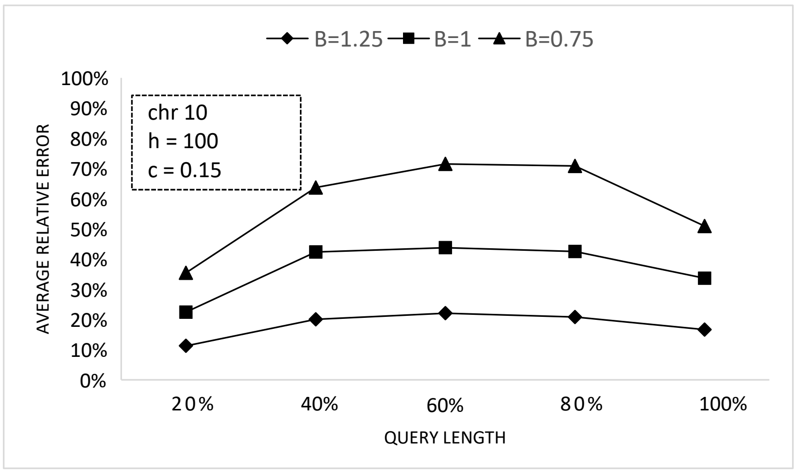
Figure 6.4: Average relative error with respect to the query length percentage for the chr2 dataset



(a)



(b)



(c)

Figure 6.5: Average relative error with respect to the query length percentage for the chr10 dataset

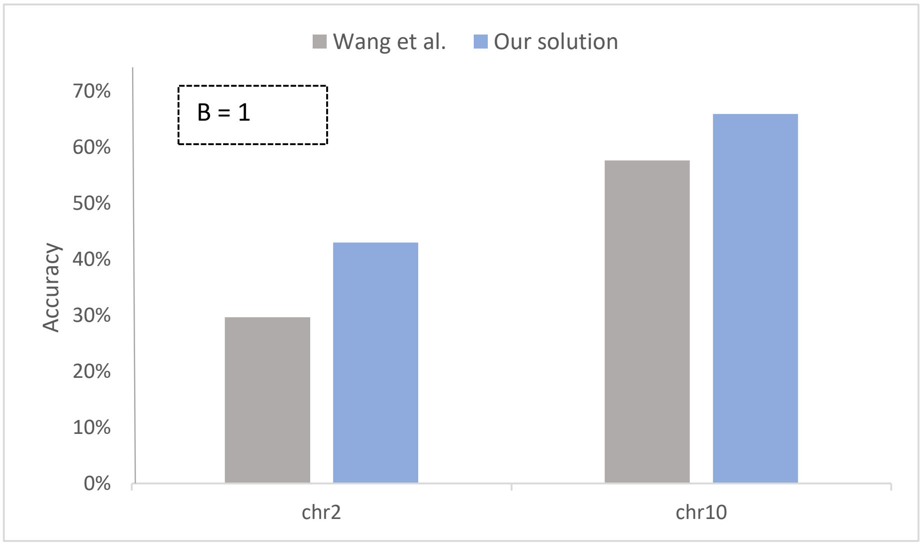


Figure 6.6: Comparative evaluation of our proposed algorithm with Wang *et al.* [88]

Chapter 7

CONCLUSION AND FUTURE WORK

7.1 Summary

In this thesis, we did extensive research on differentially-private data publishing and propose an approach for privacy-preserving genomic data publishing via differentially-private suffix tree that maintains effective level of data utility for data mining and is scalable and efficient. We observe that it is feasible to design an algorithm for differentially-private genomic data publishing that generates high utility. In the experimental evaluation of our approach, the best data utility is obtained for $B = 1.25$ for both the datasets irrespective of h. As the data grows, the utility is higher but it becomes more sensitive to the query length.

In Chapter 2, we introduce and discuss the terms and definitions that are significant to our thesis work. We explain genomic data, differential privacy and the mechanisms to achieve differential privacy i.e., Laplace, Exponential and Gaussian. We also discuss the relaxations of differential privacy, local differential privacy and the composition properties of differential privacy.

In Chapter 3, we elaborate the related work that has been done in the field of our thesis research. We discuss privacy-preserving data publishing and privacy-preserving data mining with respect to genomic and non-genomic data. We also did a comparative evaluation of the PPDP and PPDM techniques related to genomic data.

In Chapter 4, we propose our solution for privately publishing genomic data. We propose a three-phase approach that includes differentially-private partitioning tree generation, bottom-up normalization and differentially-private suffix tree generation. The proposed solution is scalable, efficient and preserves high utility for the data mining tasks.

In Chapter 5, we perform the privacy and complexity analysis of our approach. We show that our approach preserves privacy and is scalable even with respect to the large datasets.

In Chapter 6, we evaluate the performance of our algorithm. We performed the experiments to determine the scalability, efficiency and utility of our solution. The experiments demonstrate that our approach is scalable, efficient and maintains high utility with respect to count queries.

In a nutshell, the main contribution of this thesis is to propose an approach for privacy-preserving genomic data publishing via differentially-private suffix tree, which satisfies B-differential privacy, preserves data utility and is efficient and scalable.

7.2 Looking Ahead

As for future work, it would be interesting to explore how to determine the optimal value of tree height h in a differentially-private manner instead of having it as a user input, given that it determines the depth of the partitioning tree $\mathcal{P}$ and the length of the suffixes that will eventually be inserted in the suffix tree. We also look forward to exploring how to support different types of count queries for searching on the genomic data, as well as other types of queries, such as fuzzy search matching queries and frequent sequential pattern queries.

Bibliography

[1] Health insurance portability and accountability act (hipaa), 1996.

[2] Genetic information nondiscrimination act (gina), 2008.

[3] Human genome privacy protection challenge, 2014.

[4] Rakesh Agrawal and Ramakrishnan Srikant. Privacy-preserving data mining. In *Proceedings of ACM SIGMOD Record*, pages 439–450, 2000.

[5] Faraz Ahmed, Rong Jin, and Alex X. Liu. A random matrix approach to differential privacy and structure preserved social network graph publishing. *CoRR*, 2013.

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

[7] Mikhail J Atallah, Florian Kerschbaum, and Wenliang Du. Secure and private sequence comparisons. In *Proceedings of the 2003 ACM workshop on Privacy in the electronic society*, pages 39–44. ACM, 2003.

[8] Raghav Bhaskar, Srivatsan Laxman, Adam Smith, and Abhradeep Thakurta. Discovering frequent patterns in sensitive data. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 503–512. ACM, 2010.

[9] Avrim Blum, Katrina Ligett, and Aaron Roth. A learning theory approach to non-interactive database privacy. In *Proceedings of ACM Symposium on Theory of Computing (STOC)*, pages 609–618, 2008.

[10] Luca Bonomi and Li Xiong. A two-phase algorithm for mining sequential patterns with differential privacy. In *Proceedings of the 22Nd ACM CIKM*, pages 269–278, 2013.

[11] Jianneng Cao, Panagiotis Karras, Panos Kalnis, and Kian-Lee Tan. Sabre: A sensitive attribute bucketization and redistribution framework for t-closeness. *The VLDB Journal*, pages 59–81, 2011.

[12] Kamalika Chaudhuri and Claire Monteleoni. Privacy-preserving logistic regression. In *Advances in Neural Information Processing Systems*, pages 289–296, 2009.

[13] Shuchi Chawla, Cynthia Dwork, Frank McSherry, and Kunal Talwar. On privacy-preserving histograms. *CoRR*, 2012.

[14] Rui Chen, Benjamin C.M. Fung, Bipin C. Desai, and Nériah M. Sossou. Differentially private transit data publication: A case study on the montreal transportation system. In *Proceedings of the 18th ACM SIGKDD International Conference on KDD*, pages 213–221, 2012.

[15] Rui Chen, Benjamin CM Fung, S Yu Philip, and Bipin C Desai. Correlated network data publication via differential privacy. *The VLDB Journal*, pages 653–676, 2014.

[16] Rui Chen, Noman Mohammed, Benjamin CM Fung, Bipin C Desai, and Li Xiong. Publishing set-valued data via differential privacy. *Proceedings of the VLDB Endowment*, pages 1087–1098, 2011.

[17] Aaron Clauset, Cristopher Moore, and Mark EJ Newman. Hierarchical structure and the prediction of missing links in networks. *Nature*, pages 98–101, 2008.

[18] Wei-Yen Day, Ninghui Li, and Min Lyu. Publishing graph degree distribution with node differential privacy. In *Proceedings of International Conference on Management of Data(SIGMOD)*, pages 123–138, 2016.

[19] Emiliano De Cristofaro, Sky Faber, and Gene Tsudik. Secure genomic testing with size-and position-hiding private substring matching. In *Proceedings of the 12th ACM workshop on privacy in the electronic society*, pages 107–118, 2013.

[20] J. C. Duchi, M. I. Jordan, and M. J. Wainwright. Local privacy and statistical minimax rates. In *2013 IEEE 54th Annual Symposium on FOCS*, pages 429–438, 2014.

[21] Cynthia Dwork. Differential privacy. In *ICALP*, pages 1–12, 2006.

[22] Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. Our data, ourselves: Privacy via distributed noise generation. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 486–503, 2006.

[23] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Proceedings of the Third Conference on Theory of Cryptography*, TCC'06, pages 265–284, 2006.

[24] Cynthia Dwork, Moni Naor, Omer Reingold, Guy N Rothblum, and Salil Vadhan. On the complexity of differentially private data release: efficient algorithms and hardness results. In *Proceedings of the ACM symposium on Theory of computing*, pages 381–390, 2009.

[25] Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.*, pages 211–407, 2014.

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

[27] Arpita Ghosh, Tim Roughgarden, and Mukund Sundararajan. Universally utility-maximizing privacy mechanisms. *SIAM Journal on Computing*, 41(6):1673–1693, 2012.

[28] R. Giegerich and S. Kurtz. Algorithmica, 1997.

[29] R. Giegerich and S. Kurtz. From ukkonen to mccreight and weiner: A unifying view of linear-time suffix tree construction. *Algorithmica*, pages 331–353, 1997.

[30] Martin Gollery. Bioinformatics: Sequence and genome analysis, 2nd ed. david w. mount. cold spring harbor. *Clinical Chemistry*, pages 2219–2219, 2005.

[31] Michael T. Goodrich. The mastermind attack on genomic data. 2009.

[32] Melissa Gymrek, Amy L. McGuire, David Golan, Eran Halperin, and Yaniv Erlich. Identifying personal genomes by surname inference. *Science*, pages 321–324, 2013.

[33] Moritz Hardt and Guy N Rothblum. A multiplicative weights mechanism for privacy-preserving data analysis. In *FOCS, 51st Annual IEEE Symposium*, pages 61–70, 2010.

[34] Michael Hay, Chao Li, Gerome Miklau, and David Jensen. Accurate estimation of the degree distribution of private networks. In *Proceedings of IEEE International Conference on Data Mining(ICDM)*, pages 169–178, 2009.

[35] Michael Hay, Vibhor Rastogi, Gerome Miklau, and Dan Suciu. Boosting the accuracy of differentially private histograms through consistency. *Proc. VLDB Endow.*, pages 1021–1032, 2010.

[36] 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. 2006.

[37] Mr Zhicong Huang. Privacy preserving algorithms for genomic data.

[38] National Human Genome Research Institute. https://www.genome.gov/, 2015.

[39] H. Jian-min, Y. Hui-qun, Y. Juan, and C. Ting-ting. A complete (alpha,k)-anonymity model for sensitive values individuation preservation. In *2008 International Symposium on Electronic Commerce and Security*, pages 318–323, 2008.

[40] Xiaoqian Jiang, Yongan Zhao, Xiaofeng Wang, Bradley Malin, Shuang Wang, Lucila Ohno-Machado, and Haixu Tang. A community assessment of privacy preserving techniques for human genomes. *BMC medical informatics and decision making*, 14:S1, 2014.

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

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

[43] Peter Kairouz, Sewoong Oh, and Pramod Viswanath. Extremal mechanisms for local differential privacy. *J. Mach. Learn. Res.*, pages 492–542, 2016.

[44] Vishesh Karwa, Pavel N Krivitsky, and Aleksandra B Slavković. Sharing social network data: Differentially private estimation of exponential-family random graph models. *arXiv preprint arXiv:1511.02930*, 2015.

[45] Shiva Prasad Kasiviswanathan, Homin K. Lee, Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. What can we learn privately? *SIAM J. Comput.*, pages 793–826, 2011.

[46] ShivaPrasad Kasiviswanathan, Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. *Analyzing Graphs with Node Differential Privacy*, pages 457–476. 2013.

[47] Michael Kern. Anonymity: A formalization of privacy-l-diversity. In *Proceeding Seminar Future Internet (FI), IITM and ACN*, 2013.

[48] P Mayil Vel Kumar and M Karthikeyan. L-diversity on k-anonymity with external database for improving privacy preserving data publishing. *International Journal of Computer Applications*, 2012.

[49] Kristen LeFevre, David J DeWitt, and Raghu Ramakrishnan. Incognito: Efficient full-domain k-anonymity. In *Proceedings of the 2005 ACM SIGMOD*, pages 49–60, 2005.

[50] Kristen LeFevre, David J DeWitt, and Raghu Ramakrishnan. Mondrian multidimensional k-anonymity. In *Data Engineering, 2006. ICDE'06.*, pages 25–25. IEEE, 2006.

[51] David Leoni. Non-interactive differential privacy: A survey. In *Proceedings of the First International Workshop on Open Data(WOD)*, pages 40–52, 2012.

[52] N. Li, T. Li, and S. Venkatasubramanian. Closeness: A new privacy measure for data publishing. *IEEE Transactions on Knowledge and Data Engineering*, pages 943–956, 2010.

[53] Ninghui Li, Tiancheng Li, and Suresh Venkatasubramanian. t-closeness: Privacy beyond k-anonymity and l-diversity. In *Data Engineering, ICDE 2007. IEEE*, pages 106–115, 2007.

[54] Yang D Li, Zhenjie Zhang, Marianne Winslett, and Yin Yang. Compressive mechanism: Utilizing sparse representation in differential privacy. In *Proceedings of the 10th annual ACM workshop on Privacy in the electronic society*, pages 177–182, 2011.

[55] Ashwin Machanavajjhala, Daniel Kifer, Johannes Gehrke, and Muthuramakrishnan Venkitasubramaniam. L-diversity: Privacy beyond k-anonymity. *ACM Trans. Knowl. Discov. Data*, 2007.

[56] Alexandra Marin and Barry Wellman. Social network analysis: An introduction. *The SAGE handbook of social network analysis*, pages 11–25, 2011.

[57] Edward M. McCreight. A space-economical suffix tree construction algorithm. *ACM*, pages 262–272, 1976.

[58] Frank McSherry and Kunal Talwar. Mechanism design via differential privacy. In *Proceedings of IEEE Symposium on FOCS*, pages 94–103, 2007.

[59] Frank D. McSherry. Privacy integrated queries: An extensible platform for privacy-preserving data analysis. In *Proceedings of the SIGMOD '09*, pages 19–30, 2009.

[60] Adam Meyerson and Ryan Williams. On the complexity of optimal k-anonymity. In *Proceedings of the twenty-third ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 223–228, 2004.

[61] Noman Mohammed, Rui Chen, Benjamin C.M. Fung, and Philip S. Yu. Differentially private data release for data mining. In *Proceedings of the 17th ACM SIGKDD on KDD*, pages 493–501, 2011.

[62] Noman Mohammed, B. C. M. Fung, P. C. K. Hung, and C-k Lee. Anonymizing healthcare data: a case study on the blood transfusion service. In *SIGKDD*, pages 1285–1294, 2009.

[63] Yvonne Mülle, Chris Clifton, and Klemens Böhm. Privacy-integrated graph clustering through differential privacy. In *EDBT*, 2015.

[64] 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 Comput. Surv.*, pages 6:1–6:44, 2015.

[65] M. E. Nergiz and C. Clifton. δ-presence without complete world knowledge. *IEEE Transactions on Knowledge and Data Engineering*, pages 868–883, 2010.

[66] Mehmet Ercan Nergiz, Maurizio Atzori, and Chris Clifton. Hiding the presence of individuals from shared databases. In *Proceedings of ACM SIGMOD International Conference on Management of Data*, pages 665–676, 2007.

[67] Jerome Le Ny and George Pappas. Differentially private filtering. *IEEE Transactions on Automatic Control*, pages 341 – 354, 2013.

[68] Vibhor Rastogi, Dan Suciu, and Sungho Hong. The boundary between privacy and utility in data publishing. In *Proceedings of International Conference on Very Large Data Bases(VLDB)*, pages 531–542, 2007.

[69] Matthew J Rattigan, Marc Maier, and David Jensen. Graph clustering with network structure indices. In *Proceedings of international conference on Machine learning*, pages 783–790, 2007.

[70] D. Rebollo-Monedero, J. Forne, and J. Domingo-Ferrer. From t-closeness-like privacy to postrandomization via information theory. *IEEE Transactions on Knowledge and Data Engineering*, pages 1623–1636, 2010.

[71] Laura Rodriguez, Lisa D Brooks, Judith H Greenberg, and Eric D Green. The complexities of genomic identifiability. pages 275–276, 2013.

[72] Aminmohammad Roozgard, Nafise Barzigar, Pramode K Verma, and Samuel Cheng. Genomic data privacy protection using compressed sensing. *Transactions on Data Privacy*, 9:1–13, 2016.

[73] Alessandra Sala, Xiaohan Zhao, Christo Wilson, Haitao Zheng, and Ben Y Zhao. Sharing graphs using differentially private graph models. In *Proceedings of ACM SIGCOMM conference on Internet measurement conference*, pages 81–98, 2011.

[74] P. Samarati. Protecting respondents' identities in microdata release. *TKDE*, pages 1010–1027, 2001.

[75] J. Soria-Comas and J. Domingo-Ferrert. Differential privacy via t-closeness in data publishing. In *2013 Eleventh Annual Conference on Privacy, Security and Trust*, pages 27–35, 2013.

[76] Jordi Soria-Comas, Josep Domingo-Ferrer, David Sánchez, and Sergio Martínez. t-closeness through microaggregation: Strict privacy with enhanced utility preservation. *IEEE Transactions on Knowledge and Data Engineering*, pages 3098–3110, 2015.

[77] Latanya Sweeney. Achieving k-anonymity privacy protection using generalization and suppression. *IJUFKS*, pages 571–588, 2002.

[78] Latanya Sweeney. K-anonymity: A model for protecting privacy. *IJUFKS*, pages 557–570, 2002.

[79] Doug Szajda, Michael Pohl, Jason Owen, Barry G Lawson, and Virginia Richmond. Toward a practical data privacy scheme for a distributed implementation of the smith-waterman genome sequence comparison algorithm. In *NDSS*, 2006.

[80] Christine Task and Chris Clifton. A guide to differential privacy theory in social network analysis. In *Proceedings of International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2012)*, pages 411–417, 2012.

[81] Caroline Uhlerop, Aleksandra Slavković, and Stephen E Fienberg. Privacy-preserving data sharing for genome-wide association studies. *The Journal of privacy and confidentiality*, page 137, 2013.

[82] E. Ukkonen. On-line construction of suffix trees. *Algorithmica*, pages 249–260, 1995.

[83] Ke Wang and Benjamin C. M. Fung. Anonymizing sequential releases. In *Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining(KDD)*, pages 414–423, 2006.

[84] Ke Wang, Benjamin C. M. Fung, and Philip S. Yu. Template-based privacy preservation in classification problems. In *Proceedings of IEEE International Conference on Data Mining(ICDM)*, pages 466–473, 2005.

[85] Ke Wang, Benjamin C. M. Fung, and Philip S. Yu. Handicapping attacker's confidence: An alternative to k-anonymization. *Knowl. Inf. Syst.*, pages 345–368, 2007.

[86] Pingshui Wang and Jiandong Wang. L-diversity algorithm for incremental data release. *Applied Mathematics & Information Sciences*, page 2055, 2013.

[87] 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. 2009.

[88] Shuang Wang, Noman Mohammed, and Rui Chen. Differentially private genome data dissemination through top-down specialization. *BMC medical informatics and decision making*, page S2, 2014.

[89] Peter Weiner. Linear pattern matching algorithms. SWAT '73, pages 1–11, 1973.

[90] Raymond Chi-Wing Wong, Jiuyong Li, Ada Wai-Chee Fu, and Ke Wang. (α, k)-anonymity: an enhanced k-anonymity model for privacy preserving data publishing. In *Proceedings of the 12th ACM SIGKDD on KDD*, pages 754–759, 2006.

[91] Qian Xiao, Rui Chen, and Kian-Lee Tan. Differentially private network data release via structural inference. In *Proceedings of ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 911–920, 2014.

[92] Xiaokui Xiao, Guozhang Wang, and Johannes Gehrke. Differential privacy via wavelet transforms. *IEEE Transactions on Knowledge and Data Engineering*, pages 1200–1214, 2011.

[93] Xiaowei Xu, Nurcan Yuruk, Zhidan Feng, and Thomas AJ Schweiger. Scan: a structural clustering algorithm for networks. In *Proceedings of the ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 824–833, 2007.

[94] Fei Yu, Stephen E Fienberg, Aleksandra B Slavković, and Caroline Uhler. Scalable privacy-preserving data sharing methodology for genome-wide association studies. *Journal of biomedical informatics*, pages 133–141, 2014.

[95] Fei Yu and Zhanglong Ji. Scalable privacy-preserving data sharing methodology for genome-wide association studies: an application to idash healthcare privacy protection challenge. *BMC medical informatics and decision making*, 14, 2014.

[96] Q. Zhang, N. Koudas, D. Srivastava, and T. Yu. Aggregate query answering on anonymized tables. In *IEEE International Conference on Data Engineering*, pages 116–125, 2007.

[97] Bin Zhou and Jian Pei. The k-anonymity and l-diversity approaches for privacy preservation in social networks against neighborhood attacks. *Knowledge and Information Systems*, pages 47–77, 2011.