PRIVACY-PRESERVING RARE DISEASE ANALYSIS WITH FULLY HOMOMORPHIC ENCRYPTION

A Thesis Submitted to
the Graduate School of Engineering and Sciences of
İzmir Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of

MASTER OF SCIENCE

in Computer Engineering

by
Güliz AKKAYA

July 2023
İZMİR

We approve the thesis of **Güliz AKKAYA**

Examining Committee Members:

Asst. Prof. Dr. Nesli ERDOĞMUŞ
Department of Computer Engineering, Izmir Institute of Technology

---

Prof. Dr. Cüneyt BAZLAMAÇCI
Department of Computer Engineering, Izmir Institute of Technology

---

Prof. Dr. Mehmet Ufuk ÇAĞLAYAN
Department of Computer Engineering, Yaşar University

12 July 2023

Asst. Prof. Dr. Nesli ERDOĞMUŞ
Supervisor, Department of Computer Engineering
Izmir Institute of Technology

Dr. Mete AKGÜN
Co-Supervisor, Department of Computer Science
University of Tübingen

Prof. Dr. Cüneyt BAZLAMAÇCI
Head of the Department of Computer Engineering

Prof. Dr. Mehtap EANES
Dean of the Graduate School of Engineering and Sciences

ACKNOWLEDGMENTS

Firstly, I would like to express my sincere gratitude to my advisor Dr. Mete Akgün for his guidance and continuous support for this thesis study.

I would also like to thank my advisor Asst. Prof. Nesli Erdoğmuş for her valuable support during this thesis study.

Finally, I would like to thank my family, my mother, my father, and my brother for their love and support throughout my life.

ABSTRACT

PRIVACY-PRESERVING RARE DISEASE ANALYSIS WITH FULLY HOMOMORPHIC ENCRYPTION

Rare diseases severely affect many people across the world at the present time. Researchers conduct studies to understand the reasons behind rare diseases and as a result of this research, diagnosis, and treatment methods are developed. Rare disease analysis is performed to specify the disease-causing variants on the genome data of patients. The researchers need access to as much genome data as possible to find causing variants of rare diseases. On the other hand, the genome data of patients should be protected because it can be used to detect the identity of individuals. The researchers are not able to share the genome data of patients easily because of regulations such as General Data Protection Regulation (GDPR). For this reason, rare disease analysis should be performed in a secure way that protects the privacy of patients while enabling the collaboration of multiple medical institutions. In this context, a privacy-preserving collaborative system for rare disease analysis should be provided. This thesis study focuses on the utilization of fully homomorphic encryption, a method that enables unlimited number of operations to be performed on encrypted data, for privacy-preserving collaborative rare disease analysis. Two different methods, the boolean circuit method, and the integer arithmetic method, are implemented to perform rare disease analysis on the encrypted genome data to find disease-causing variants, and various experiments are performed to assess the efficiency of the proposed methods.

ÖZET

TAM HOMOMORFİK ŞİFRELEME İLE GİZLİLİĞİ KORUYAN NADİR HASTALIK ANALİZİ

Günümüzde nadir hastalıklar dünya genelinde birçok insanı ciddi şekilde etkilemektedir. Araştırmacılar, nadir hastalıkların arkasındaki nedenleri anlamak için çalışmalar yürütür ve bu araştırmalar sonucunda teşhis ve tedavi yöntemleri geliştirilir. Nadir hastalık analizi hastaların genom verileri üzerinde hastalığa neden olan varyantların belirlenmesiyle gerçekleştirilir. Araştırmacıların, nadir hastalıklara neden olan varyantları bulabilmeleri için olabildiğince çok genom verisine erişmesi gerekir. Buna karşılık, hastaların genom verileri bireylerin kimliğinin tespit edilmesinde kullanılabileceği için korunmalıdır. Araştırmacılar, Genel Veri Koruma Tüzüğü (GDPR) gibi düzenlemeler nedeniyle hastaların genom verilerini kolayca paylaşamamaktadır. Bu nedenle, nadir hastalık analizinin birden fazla sağlık kuruluşunun işbirliğine olanak sağlarken hastaların gizliliğini de koruyan güvenli bir şekilde yapılması gerekmektedir. Bu kapsamda nadir hastalık analizi için gizliliği koruyan ortak çalışmaya dayalı bir sistem sunulmalıdır. Bu tez çalışması, gizliliği koruyan ortak çalışmaya dayalı nadir hastalık analizi için, şifrelenmiş veriler üzerinde sınırsız sayıda işlemin gerçekleştirilmesine olanak sağlayan bir yöntem olan tam homomorfik şifrelemenin kullanımına odaklanmaktadır. Hastalığa neden olan varyantları belirleyerek şifrelenmiş genom verileri üzerinde nadir hastalık analizi yapmak için boolean devre yöntemi ve tamsayı aritmetik yöntemi olmak üzere iki farklı yöntem uygulanmıştır, ve önerilen yöntemlerin verimliliğini değerlendirmek için çeşitli deneyler gerçekleştirilmiştir.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

The significance of data has increased day by day in recent years. In many fields such as health, finance, and education, a multitude of solutions that utilize data analysis are proposed by domain experts for various problems that they face in these fields [1–6], and these solutions provide great convenience for people. The amount of data used to develop autonomous data analysis systems affects their performance. As the amount of data used increases, more accurate and meaningful results are obtained from the data analysis process. For this reason, data sharing among experts is essential to construct more efficient systems. Although sharing available data is advantageous, it may also cause serious privacy issues. Privacy has become a major concern due to easy access to data in today's world in many fields [7–9]. For instance, many problems such as identity theft, misuse of personal information, and financial loss have arisen due to unauthorized access to data. In this respect, these threats should be considered while sharing data, and appropriate methods should be employed to protect the privacy of individuals.

Among many areas where data is analyzed to solve various problems, health is a rather important field. The data of patients are required to be shared among researchers, and researchers analyze these data to diagnose and treat diseases [10–12]. As a result of these analyses, many effective treatments can be developed. However, medical data contain the sensitive information of patients. Unauthorized access to these data may cause a leak of personal information of patients, and threaten their privacy [13]. Therefore, privacy-preserving solutions should be provided to protect the sensitive data of patients.

Genome data is rather sensitive data that is frequently used in medical research. It contains the complete set of genetic information in an organism. Accessing genome data is a critical utility of advancing medical research, as it provides significant insights into various processes. Researchers require access to patients' genome data to accurately diagnose and provide appropriate treatments for diseases. Rare disease analysis is an important research area in which genome data is utilized. By analyzing genome data, researchers gain a better understanding of disease-causing mutations, which in turn facilitates the diagnosis and treatment of rare diseases [14, 15]. In rare disease research, the whole genome data of patients are analyzed to detect the disease-causing variants. For such analysis, genome sequencing which is a laboratory procedure is utilized as a research tool to determine the entire genetic structure of patients. The changes in areas of the genome are observed using

genome sequencing [16], and this helps to find disease-causing mutations in the whole genome data for rare disease analysis.

The genome data of individuals must be accessible to find patterns and understand the disease-causing genes by researchers. Although sharing genome data with researchers is a requirement for such analysis, this causes severe privacy issues because the genome data contains an individual's complete set of genetic information [17, 18]. Since genome data is the ultimate identifier to reveal an individual's identity, genome data should not be shared publicly. Anyone who has this sensitive data can determine the patient's identity easily, and this creates a risk to the patient's privacy. For this reason, privacy-preserving methods should be utilized to protect the privacy of rare disease patients while sharing and processing sensitive data of patients.

In this study, a collaborative and privacy-preserving method is proposed for rare disease analysis that protects patients' genome data by leveraging the fully homomorphic encryption method (FHE). Fully homomorphic encryption is a powerful technique that enables unlimited operations to be performed on the encrypted data without the need for decryption. Rare disease analysis involves identifying disease-causing variants within the complete genome data based on specific inheritance models. This requires researchers to have access to patients' genome data to identify mutations responsible for rare diseases. However, the sharing of complete genome data can compromise patient identity. To address this concern, the FHE method is employed for rare disease analysis. The process involves filtering mutations associated with rare diseases on the encrypted genome data using FHE. Only the necessary variant data, obtained from the filtering process, is shared with researchers. Consequently, the privacy protection of rare disease patients is ensured throughout the collaborative analysis.

1.1. Motivation

In today's world, numerous individuals are affected by various diseases, some of which have a profound negative impact on their lives. To enhance the quality of life for these patients, it is crucial to develop effective treatments. In the medical field, researchers play an important role in developing treatments by analyzing disease-related data and uncovering the underlying causes. Through this analysis, they can gain valuable insight and identify suitable treatments.

Rare diseases are diseases that affect a small number of people in the general population (less than 1 in 2,000 people) [19]. Although rare diseases affect a few people in the total population, there is a vast amount of rare disease patients in the world, and

the consequences of these diseases may be extremely severe for patients. At present, there are more than 7,000 identified rare diseases, and the estimated number of rare disease patients worldwide is 350 million [20]. Cystic fibrosis, muscular dystrophy, and myocarditis are just a few examples of these diseases that seriously affect the lives of many patients. For instance, cystic fibrosis patients experience various troublesome symptoms or life-threatening complications such as muscle and joint pains, gastrointestinal problems, cancer, and heart failure [21]. As another example, many pediatric cancer types such as pediatric non-small cell lung cancer and pediatric t-cell leukemia affect many children and threaten their lives. Considering that there are a large number of rare disease patients and these diseases seriously hinder the quality of patients' life, it is significant to provide treatments for them.

In a typical rare disease analysis scenario, researchers analyze the genome data of rare disease patients to diagnose rare diseases and provide appropriate treatment methods. The researchers need to have access to the genome data of patients. Thus, they can specify disease-causing mutations by analyzing whole genome data. Besides, researchers should access the genome data from as many patients as possible to detect the disease-causing variants because rare disease patients with the same phenotype are rarely found in the population. On the other hand, the genome data of patients can not be publicly available, since it reveals private information. Furthermore, sharing the genome data of patients is regulated by law, and access to the genome data of patients is not straightforward. General Data Protection Regulation (GDPR) [22] protects the genome data of patients by data sharing rules. For these reasons, researchers can not easily access that data, and this makes research on rare disease diagnosis and treatment very difficult. In this context, an appropriate solution that ensures the privacy of patients' genome data while enabling collaborative research for rare diseases is necessary. This thesis proposes a method that enables rare disease analysis in a way that prioritizes privacy and collaboration. The analysis is conducted using fully homomorphic encryption, which establishes the protection of patients' privacy. At the same time, it allows researchers to access genome data from various medical institutions, enabling a collaborative environment for rare disease analysis.

1.2. Aim and Objectives

In this thesis, the proposed solution aims to perform collaborative rare disease analysis with the fully homomorphic encryption method in a privacy-preserving manner that protects the patients' sensitive data. It recognizes the importance of collaboration

among medical institutions in advancing rare disease research. By leveraging the fully homomorphic encryption method, the solution enables efficient and secure sharing of genome data from multiple medical institutions for a collaborative effort. This approach allows researchers to collectively analyze the data, identify disease-causing variants, and gain insights into rare diseases while safeguarding the privacy of individual patients. This facilitates the seamless exchange of knowledge and expertise among medical institutions, fostering a collaborative environment for rare disease research.

1.3. Outline of Thesis

The contents of the next chapters can be explained as follows. In Chapter 2, studies related to privacy-preserving rare disease analysis are listed. Next, the main concepts of rare disease analysis and fully homomorphic encryption method are discussed in Chapter 3. The details of rare disease analysis, inheritance models, and Variant Call Format (VCF) are mentioned in Section 3.1. The evaluation of homomorphic encryption and fully homomorphic encryption schemes are detailed in Section 3.2. In Chapter 4, the methods used for privacy-preserving rare disease analysis are presented. The method for representing the genome data is described in Section 4.1. The details of two different methods, namely the boolean circuit method and the integer arithmetic method are given in Section 4.2. Experimental results obtained with these methods are discussed in Chapter 5. Lastly, the research findings and future work are elaborated in Chapter 6.

CHAPTER 2

RELATED WORK

In this chapter, existing studies related to privacy-preserving rare disease analysis are detailed and the methods proposed in these studies are discussed.

A method that enables private computation on encrypted genomic data is proposed by Lauter et al. (2014) [23]. In that study, standard genomic algorithms used for genome-wide association studies (GWAS) are utilized to analyze the encrypted genome data using homomorphic encryption. Statistical algorithms such as the Pearson Goodness-of-Fit, the D' and r^2 measures of linkage disequilibrium, the Estimation Maximization (EM), and the Cochran-Armitage Test for Trend (CATT) are used to find the associations to diseases in the genome data.

In another study [24], a framework is proposed by Wang et al. (2015) using the homomorphic encryption method to realize privacy-preserving rare variants analysis with a small sample size. Analysis methods such as secure rejection sampling, and secure integer comparison are proposed. Secure integer comparison is utilized to compute a homomorphic exact logistic regression model. The P-value of exact logistic regression parameters on encrypted data is estimated using the proposed framework which additionally provides secure outsourcing.

Another framework is presented by Zhang et al. (2015) that fully outsources GWAS securely using homomorphic encryption [25]. It allows to perform secure divisions on encrypted data. Two different protocols that provide secure errorless division and secure approximation division are presented in the study. The proposed framework is evaluated using chi-square statistic computation.

Secret sharing-based techniques are proposed by Zhang et al. (2015) to provide secure computation on real-life genomic data [26]. In that study, secure distributed genome analysis for GWAS and sequence comparison computation are provided. Minor allele frequency, chi-squared statistics computation, and distance computation between two genomic sequences are performed securely using the proposed techniques.

A hardware-based framework is proposed by Chen et al. (2017) to perform rare disease analysis with privacy protection [27]. The proposed framework provides secure transmission and analyses of sensitive genome data. The Software Guard Extensions (SGX) is used to perform trustworthy computations on genome data of rare disease patients. The proposed solution is evaluated using family-based transmission tests as a

real-life example. The transmission disequilibrium test (TDT) [28] which is a family-based association test for linkage disequilibrium is performed. In that study, the aim is to provide international collaboration for rare disease analysis, which is realized by jointly analyzing the data hosted on three different continents. The cohorts associated with children from the U.S., the UK, and Singapore are analyzed for Kawasaki disease which is a rare disease with the proposed framework.

In another study [29], Wang et al. (2017) propose a solution to protect the privacy of families when using the TDT in genome-wide association studies by utilizing the differential privacy (DP) method. The differentially private mechanisms for TDT based on test statistics, P-values, and the shortest Hamming distance (SHD) scores are developed to perform genome analysis in a privacy-preserving manner, and the performance results of the DP mechanisms for TDT are compared.

In [30], Jagadeesh et al. (2017) propose a privacy-preserving solution using secure multi-party computation (MPC) for genomic diagnosis without revealing patients' genome data. Secure MPC is a cryptographic technique that provides multiple parties to compute a function jointly without revealing their private inputs to each other [31]. In that study, three different boolean operations are defined for filtering disease-causing variants. Firstly, the INTERSECTION operation finds rare functional variants that are shared by two different parties. Secondly, the SETDIFF operation is used to find rare functional variants seen in all affected individuals but not seen in unaffected individuals. Thirdly, the MAX operation provides to find a gene that contains rare functional mutations in a large number of affected cases. The defined boolean operations are utilized to provide privacy-preserving disease diagnosis in various real scenarios such as small patient cohorts, trio analysis, and two-hospital collaboration.

Last but not least, a privacy-preserving solution is proposed for identifying disease-causing mutations with privacy protection by Akgün et al. (2020) using the secure MPC method [32]. The rare disease analysis is performed on large-scale genome data considering various inheritance models such as recessive, dominant, and compound heterozygous. The secure protocols based on the combination of arithmetic and boolean sharing are proposed. In addition, privacy-preserving cross-institutional collaborations are provided for rare disease analysis with the proposed solution.

CHAPTER 3

BACKGROUND

This chapter discusses some basic concepts about rare disease analysis and fully homomorphic encryption. Firstly, the rare disease analysis process is explained in Section 3.1. Next, three inheritance models that are considered for rare disease analysis are mentioned in three consecutive sections: the recessive inheritance model, the dominant inheritance model, and the de novo inheritance model. Furthermore, the details of the Variant Call Format (VCF) are expressed. In Section 3.2, the homomorphic encryption method and its types as partially homomorphic encryption, somewhat homomorphic encryption, and fully homomorphic encryption are explained in detail.

3.1. Rare Disease Analysis

Rare disease analysis is performed by researchers by examining the genome data of patients. Researchers use genome sequencing as a research tool to analyze the variants in the genome data. The genome data of an individual contain millions of variant information and the researchers need to expose the disease-causing variants among a huge amount of genome data to understand the reasons for rare diseases. The Variant Call Format (VCF) files that store the patients' genome sequencing are analyzed by filtering variants to specify the disease-causing mutations under specific inheritance models such as recessive and dominant inheritance models, and family information. As the result of this filtering process, the variants of interest are found, and the research is focused on filtered variants to diagnose and treat rare diseases. Filtering the disease-causing variants in a VCF file considering the given inheritance model and family information is realized with the following steps:

- **Inheritance Model and Family Structure**
  Understanding of the inheritance model of interest and the family structure in the VCF file plays an essential role in rare disease analysis. This provides to identify the inheritance pattern of disease-causing variants in the family.

- **Identification of Family Members and Relationships**
  The family members such as the mother, and father, affected individuals, and unaffected individuals in the VCF file and the relationships between them are identified. Specific labels or IDs are assigned to family members for identification.

- **Quality Control Filters**
  Low-quality variant samples are removed from the data using standard quality control filters such as genotype quality scores, allele balance checks, and read depth thresholds.

- **Inheritance Model-Specific Filters**
  The variants that comply with the expected inheritance pattern are detected using the inheritance model-specific filters.

  - **Autosomal Recessive:** The homozygous variants that are present in affected individuals and are not present or are present in heterozygous form in unaffected individuals are filtered.
  - **Autosomal Dominant:** The heterozygous variants that are present in affected individuals and are not present in unaffected individuals are filtered.
  - **De Novo Mutations:** The variants that are present in all affected individuals but are not present in other family members are filtered.

- **Minor Allele Frequency Filtering**
  The variants with high minor allele frequencies in the population are filtered using a threshold and the variants that are more probably associated with rare disease are specified.

- **Functional Impact Prediction**
  The variants that are present in the functional genome regions, such as exons or splice sites are possible disease-causing variants. Functional annotation tools and databases are utilized to predict the possible effect of variants on gene function. Thus, potentially disease-causing variants are prioritized.

- **Co-Segregation Analysis**
  Co-segregation analysis is performed to verify the segregation of variants with the disease phenotype. This analysis includes examining whether the variants pass down with the disease phenotype across multiple affected and unaffected family members.

- **Interpretation and Validation**
  The filtered variants are interpreted considering the inheritance model used and the

rare disease of interest. The most promising variants are validated using experimental techniques or validation datasets.

It is essential to protect the sensitive genome data of patients in the rare disease analysis process. The VCF files contain the variant calls and genotypes data which are considered sensitive personal information. The sensitive data fields in a VCF file are given in detail in the following:

- **Individual Genotype Data:** The genotypes are considered extremely confidential data. The genetic data, such as the alleles inherited at specific variant positions are represented in genotypes. This data has the potential to reveal sensitive information such as an individual's disease status and genetic traits. To eliminate this risk, the genotype data should be encrypted while sharing and storing.
- **Sample IDs and Names:** The identifiers such as IDs or names in the VCF files should be encrypted to prevent the identification of individuals.
- **Clinical Information:** Any clinical data such as disease status and medical conditions in VCF files should be protected to ensure privacy.
- **Family Relationships:** Preserving the privacy of family relationships in a VCF file is essential to protect the identities of individuals.
- **Population Information:** VCF files can contain information from particular populations or ethnic groups, this information should be protected to prevent any possible implications regarding ethnicity or ancestry.

As mentioned above the genome data of rare disease patients should not be shared publicly because of privacy concerns. Besides, it is not allowed to share the sensitive genome data of rare disease patients under some regulations. For these reasons, researchers do not have access to sufficient patient data for rare disease analysis. The solution is for the genome data of different patients to be accessible by the researchers using some privacy-preserving methods because data size and variability have a significant effect on the effectiveness of the rare disease analysis in discovering successful treatments for patients.

3.1.1. Inheritance Models

The rare disease analysis is realized by prioritizing the disease-causing variants in the whole genome data considering specific inheritance models. It is essential to utilize

inheritance models for rare disease analysis since the disease-causing variants are inherited from parents. The inheritance models used for privacy-preserving rare disease analysis in this study are explained in the following subsections.

3.1.1.1. Recessive Inheritance Model

In the recessive inheritance model, the parents of the affected individual are both heterozygous carriers and the affected individual is homozygous for the disease-causing variant. An example condition for the recessive inheritance model is shown in Figure 3.1. In this example, the parents are both carriers, and the children are carrier, unaffected and affected by disease respectively. The variants that comply with the recessive inheritance model are filtered when a query is made by the researcher and the filtered variants are analyzed to interpret rare diseases.

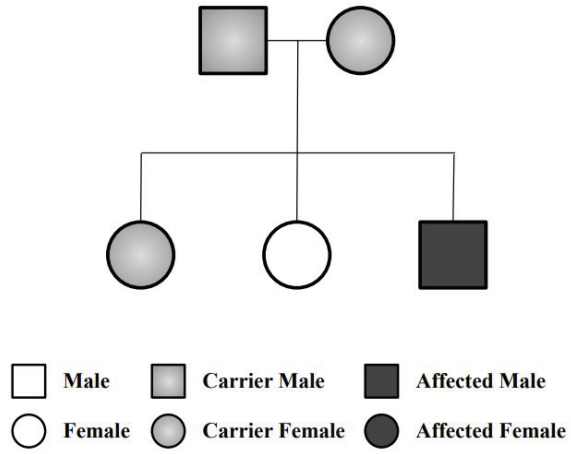

Figure 3.1. Sample Recessive Inheritance Condition

3.1.1.2. Dominant Inheritance Model

In the dominant inheritance model, the affected individual is heterozygous for the disease-causing variant and has a single copy of the disease-causing variant that is passed down from one parent. An example condition for the dominant inheritance model is

shown in Figure 3.2. In this example, one of the parents is affected by the disease, and the other is unaffected. Besides, one of the children is unaffected and the others are affected. When a query is made by the researcher, disease-causing variants matching the dominant inheritance model are filtered and the filtered variants are used for rare disease analysis.

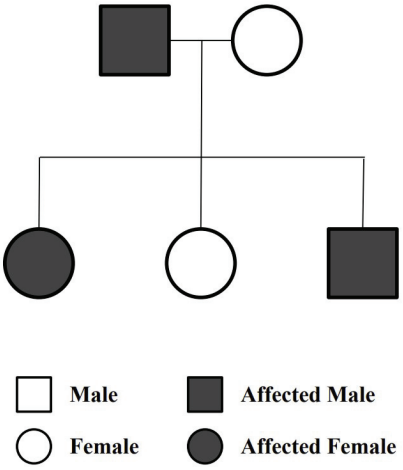

Figure 3.2. Sample Dominant Inheritance Condition

3.1.1.3. De Novo Inheritance Model

In the De Novo inheritance model, all individuals affected by the rare disease have the same type of disease-causing mutation and this mutation is not present in other family members. The disease-causing variants that comply with this case are filtered for this inheritance model.

3.1.2. Variant Call Format

The Variant Call Format [33] is a generic format for representing and storing human genome sequence variation data. The genome variation data of rare disease patients are stored in VCF files and researchers analyze the disease-causing variants using the information in these files. A VCF file has two sections. The header section contains

the descriptions of tags and annotations used in the data section. It has a field definition line that contains the names of the data section columns such as chromosome, position, identifier, reference allele, alternate non-reference alleles, format, and an arbitrary number of sample IDs. The data section, on the other hand, includes the corresponding data to the columns listed in the header section. A sample VCF file is shown in Figure 3.3.

```
##fileformat=VCFv4.1
##fileDate=20110413
##source=VCFtools
##reference=file:///refs/human_NCBI36.fasta
##contig=<ID=1,length=249250621,md5=1b22b98cdeb4a9304cb5d48026a85128,species="Homo Sapiens">
##contig=<ID=X,length=155270560,md5=7e0e2e580297b7764e31dbc80c2540dd,species="Homo Sapiens">
##INFO=<ID=AA,Number=1,Type=String,Description="Ancestral Allele">
##INFO=<ID=H2,Number=0,Type=Flag,Description="HapMap2 membership">
##FORMAT=<ID=GT,Number=1,Type=String,Description="Genotype">
##FORMAT=<ID=GQ,Number=1,Type=Integer,Description="Genotype Quality">
##FORMAT=<ID=DP,Number=1,Type=Integer,Description="Read Depth">
##ALT=<ID=DEL,Description="Deletion">
##INFO=<ID=SVTYPE,Number=1,Type=String,Description="Type of structural variant">
##INFO=<ID=END,Number=1,Type=Integer,Description="End position of the variant">
#CHROM POS ID     REF  ALT    QUAL FILTER  INFO                FORMAT     SAMPLE1  SAMPLE2
1        1  .     ACG  A,AT    40  PASS    .                   GT:DP      1/1:13   2/2:29
1        2  .     C    T,CT    .   PASS    H2;AA=T             GT         0|1      2/2
1        5  rs12  A    G       67  PASS    .                   GT:DP      1|0:16   2/2:20
X      100  .     T    <DEL>   .   PASS    SVTYPE=DEL;END=299  GT:GQ:DP   1:12:.   0/0:20:36
```

Figure 3.3. Sample Variant Call Format (VCF) File
(Source: The Variant Call Format and VCFtools [33])

In the sample data column, variant data of different patients are represented as homozygous or heterozygous using numerical representations. Another example that contains variant data of 3 different samples is shown in Figure 3.4.

```
FORMAT        NA00001          NA00002          NA00003
GT:GQ:DP:HQ   0|0:48:1:51,51   1|0:48:8:51,51   1/1:43:5:.,.
GT:GQ:DP:HQ   0|0:49:3:58,50   0|1:3:5:65,3     0/0:41:3:.,.
GT:GQ:DP:HQ   1|2:21:6:23,27   2|1:2:0:18,2     2/2:35:4:.,.
GT:GQ:DP:HQ   0|0:54:.:56,60   0|0:48:4:51,51   0/0:61:2:.,.
GT:GQ:DP      0/1:.:4          0/2:17:2         1/1:40:3
```

Figure 3.4. Sample Variant Data in a VCF file
(Source: The Variant Call Format Specification [34])

For this VCF file, the variant data at the first position are explained in Table 3.1. Sample data 1 (NA00001) is classified as a homozygous reference allele, sample data 2 (NA00002) is classified as a heterozygous alternate allele, and sample data 3 (NA00003) is classified as a homozygous alternate allele.

Table 3.1. The Variant Data and Descriptions

Sample Variant Data	Description
0|0:48:1:51,51	Homozygous Reference Allele
1|0:48:8:51,51	Heterozygous Alternate Allele
1/1:43:5:.,.	Homozygous Alternate Allele

3.2. Homomorphic Encryption

Homomorphic Encryption (HE) is a cryptographic technique that allows to perform computations on the encrypted data without requiring access to the decryption key. Rivest et al. [35] used the term "homomorphism" for the first time in the cryptography field to refer to a method that performs operations on the encrypted data without any decryption. Encryption schemes that are homomorphic over given operations should follow the following rules [36]:

- An encryption scheme is defined as homomorphic over the addition and multiplication operations if the following equations are supported, where E is the encryption algorithm, and M is the set of all possible messages:

$$(1) \quad E(m_1) + E(m_2) = E(m_1 + m_2),\ \forall m_1, m_2 \in M$$

$$(2) \quad E(m_1) * E(m_2) = E(m_1 * m_2),\ \forall m_1, m_2 \in M$$

HE schemes mainly perform four operations: Key generation, encryption, evaluation, and decryption. Firstly, the key generation operation generates a pair of secret and public keys for the asymmetric HE or a single key for symmetric HE. In the next step, the encryption operation generates a ciphertext from the plaintext using a specific encryption algorithm. Some level of noise is added to the plaintext to create a ciphertext. The noise term can be described as the error that is added to the ciphertext to achieve the required security level with the encryption operation. Successively, the evaluation is performed by applying computations on the ciphertexts. This operation takes ciphertexts as input and generates an evaluated ciphertext as output by performing a set of operations of a function. Lastly, the evaluated ciphertext is decrypted using the decryption key and the result is obtained in the decryption step.

Different evaluation steps of homomorphic encryption schemes lead to 3 types

of HE: Partially Homomorphic Encryption (PHE) schemes, Somewhat Homomorphic Encryption (SWHE) schemes, and Fully Homomorphic Encryption (FHE) schemes [36]. The details of the HE scheme types are discussed in the following sections.

3.2.1. Partially Homomorphic Encryption

Partially Homomorphic Encryption supports only one type of operation, addition or multiplication, on the encrypted data. For example, Rivest-Shamir-Adleman (RSA) scheme is one of the earliest PHE schemes. RSA [35] is based on the factoring problem of the product of two large prime numbers and only supports the multiplication operation. Another prominent example of PHE schemes is the Paillier scheme. Paillier scheme [37] is a probabilistic encryption scheme based on the composite residuosity problem. Paillier scheme only supports the addition operation on the encrypted data without revealing the values.

3.2.2. Somewhat Homomorphic Encryption

Somewhat Homomorphic Encryption supports addition and multiplication operations on the encrypted data with a limited number of times. The reason for the limited number of operations in the SWHE schemes can be explained as follows. The encryption process introduces some level of noise into the ciphertexts and the noise level increases with every performed operation on a ciphertext. After a certain threshold, the noise level becomes too large and the decryption operation of the ciphertext does not result in correct values. Besides, the size of the ciphertext increases with each operation and this limits the number of operations because of efficiency problems in computations. Yao's Garbled Circuit [38] is one of the earliest examples of SWHE schemes and is based on millionaires' problem. It supports addition and multiplication operations. However, the size of the ciphertext increases at least linearly with the computation of every gate in the circuit, and this results in computational overhead and limits the number of operations over the ciphertexts. Boneh-Goh-Nissim (BGN) scheme [39] is another important example of SWHE schemes and is based on the subgroup decision problem. BGN scheme supports an unlimited number of addition operations and one multiplication operation. Furthermore, the size of the ciphertext stays constant in the BGN scheme. This property is critical and utilized to obtain FHE schemes.

3.2.3. Fully Homomorphic Encryption

Fully Homomorphic Encryption supports unlimited number of addition and multiplication operations over the encrypted data. As mentioned in the previous section, the noise level of a ciphertext increases with each addition or multiplication operation. The multiplication operation increases the noise level in a ciphertext more than the addition operation because of its computational complexity. After a certain threshold, the decryption function can not recover the message correctly because of the high level of noise in the ciphertext. For this reason, the noise level must be reduced to successfully decrypt the ciphertext. The first example of FHE schemes is proposed by Gentry in 2009 [40]. The scheme is an ideal lattice-based FHE scheme. Gentry presents squashing and bootstrapping methods as solutions to provide an unlimited number of operations on ciphertexts. The squashing method is utilized to decrease the decryption complexity. The bootstrapping method produces a ciphertext with a lower level of noise from the noisy ciphertext. Thus, more computations can be performed on the obtained ciphertext after the bootstrapping operation. Some of the important FHE schemes and their details are explained in the following section.

3.2.3.1. Fully Homomorphic Encryption Schemes

Fully Homomorphic Encryption schemes can be categorized into four different branches: the ideal lattice-based schemes, the schemes over integers on the approximate greatest common divisor (AGCD) problem, the learning with errors (LWE) problem, and its ring version (RLWE) based schemes, and NTRU based schemes [41].

The ideal lattice-based schemes rely on a mathematical construction called the lattice. In mathematics, a lattice is an integer linear combination of a set of linearly independent vectors. Lattice-based cryptography is considered safe against quantum computers, and the security of the lattice-based schemes is based on problems such as the Shortest Vector Problem and its variants [42]. The Shortest Vector Problem is focused on finding the shortest nonzero vector in a given lattice. However, as the ideal lattice-based schemes are based on mathematical constructions that are difficult to implement efficiently, the computational cost is an issue for these schemes. The first FHE scheme proposed by Gentry is an example of ideal lattice-based schemes. As mentioned before, the bootstrapping technique is proposed to decrease the error level of ciphertexts by Gentry. The bootstrapping method essentially employs a re-encryption process of the ciphertext

with a smaller level of error. Although bootstrapping method allows performing unlimited number of operations on a ciphertext, this method should be used only when it is necessary because of its high computational complexity.

The schemes over integers based on the AGCD problem is a less complex FHE scheme type than the ideal lattice-based FHE schemes. The security of the schemes over integers relies on the AGCD problem, which tries to find a common near divisor of the given set of integers. The Dijk-Gentry-Halevi-Vaikuntanathan (DGHV) scheme [43] proposed by Van Dijk et al. is an example of the schemes over integers on the AGCD problem. The DGHV scheme has high computational complexity, and its public key is large. For this reason, it has a low-efficiency level. After DGHV, some optimizations that improve efficiency are also proposed. For instance, the optimization proposed by Coron et al. reduces the size of the public key to lower the complexity level of this scheme [44].

Another type of FHE schemes is the schemes based on learning with errors (LWE) and ring learning with errors (RLWE) problems. LWE problem is focused on finding a vector from a list of approximate random equations on this vector. Two FHE schemes based on the LWE and RLWE problems [45] [46] are proposed by Brakerski and Vaikuntanathan utilizing the bootstrapping method and circular security assumption. Furthermore, re-linearization and dimension-modulus reduction are proposed as two novel methods in this study. A leveled fully homomorphic encryption scheme called Brakerski-Gentry-Vaikuntanathan (BGV) is proposed by Brakerski et al., which avoids the bootstrapping method that has high computational complexity [47]. The term "leveled" refers to a predefined level of complexity, and only a set of allowed operations by the defined complexity level can be performed. The purpose of the predefined complexity level is to increase efficiency while performing operations on the encrypted data. In addition, the batching and the modulus switching methods are introduced with the BGV scheme in this study. In [48], the Brakerski-Fan-Vercauteren (BFV) scheme that allows performing modular arithmetic operations on the encrypted integer data is proposed by Fan and Vercauteren. BFV scheme is also a leveled fully homomorphic encryption scheme and the security of the scheme relies on the RLWE problem. An optimization of the BFV scheme that uses the residue number system (RNS) is introduced by Bajard et al. [49]. In [50], another leveled homomorphic encryption scheme is proposed by Cheon et al. that performs approximate arithmetic operations, and an open-source library that implements the scheme is presented in this study. The RLWE-based Chillotti-Gama-Georgieva-Izabachène (CGGI) scheme that allows performing logical operations such as AND, OR, and XOR on the encrypted data is proposed by Chillotti et al. [51].

Lastly, the NTRU scheme is an encryption scheme based on lattice problems introduced by Hoffstein et al. in 1998 [52]. An NTRU-based FHE scheme called the Lopez-Tromer-Vaikuntanathan (LTV) scheme is proposed by Lopez-Alt et al. [53]. It utilizes bootstrapping and modulus switching methods, and employs the circular security

concept for the security of the scheme. The study also introduces the multikey FHE as a new concept that allows performing computations on ciphertexts encrypted under different keys. The parameters of the NTRU-based schemes should be selected carefully because it is proven that they are vulnerable to attacks. The size of parameters should be large enough to obtain a secure NTRU-based scheme. However, this decreases efficiency.

The FHE schemes utilized in the methodology of this thesis, namely CGGI and BFV are explained in more detail in the following subsections.

3.2.3.1.1. CGGI Scheme

The key generation, encryption, evaluation, and decryption algorithms of the CGGI scheme are explained in the following [41].

Let $R = \mathbb{Z}[x]/\langle x^d + 1\rangle$ be the ring of polynomials with integer coefficients modulo of the polynomial $x^d + 1$, where d is a power of 2, $T = \mathbb{R}[x]/\langle x^d + 1\rangle \bmod 1$ and $R_2 = \mathbb{F}_2[x]/\langle x^d + 1\rangle$. Each element in R_2 is a polynomial in R with binary coefficients.

- **Key Generation**
  For the key generation, the security parameter λ is given as input and the small secret key $\mathsf{s} \in R_2^n$ is generated as output.

- **Encryption**
  For the encryption, the secret key $\mathsf{s} \in R_2^n$, the error parameter α, and the message $m \in T$ are given as input. Next, a uniformly random mask $\mathsf{a} \in T^n$ and a small error $e \in \chi$ are chosen, where χ is a B-bounded distribution. As the output, the ciphertext $\mathsf{c} = (\mathsf{a}, \mathsf{s} \cdot \mathsf{a} + m + e) \in T^n \times T$ is generated.

- **Evaluation**

  - **Homomorphic Linear Combinations of Ciphertexts**
    $\mathsf{c}_1, ..., \mathsf{c}_p$ are independent ciphertexts under the same key s, and $f_1, ..., f_p$ are integer polynomials in R. The ciphertext c is considered as $\mathsf{c} = \sum_{i=1}^{p} f_i \cdot \mathsf{c_i}$, and

$$\mathsf{Dec_s(c)} = \sum_{i=1}^{p} f_i \cdot \mathsf{Dec_s(c_i)}.$$

- **Decryption**
  For the decryption, the secret key $\mathsf{s} \in R_2^n$ and the ciphertext $\mathsf{c} \in T_n + 1$ are given as input. Next, the secret linear κ-Lipschitz function φ_s of the ciphertext c is

computed. The phase $\varphi_s : T^n \times T \rightarrow T$ is such that $\varphi_s(\mathsf{a}, b) = b - \mathsf{s} \cdot \mathsf{a}$. This function is parametrized by the secret key $\mathsf{s} \in R_2^n$. The value of the phase $\varphi_s(\mathsf{c})$ is approximate to the actual message and rounded to the nearest point in the message space $M \subset T$.

$$\varphi_s(\mathsf{c}) = \mathsf{s} \cdot \mathsf{a} + m + e - \mathsf{s} \cdot \mathsf{a} = m + e$$

3.2.3.1.2. BFV Scheme

The key generation, encryption, evaluation, and decryption algorithms of the BFV scheme are explained in the following [41].

Let $R = \mathbb{Z}[x]/\langle x^d + 1\rangle$ be the ring of polynomials with integer coefficients modulo of the polynomial $x^d + 1$, where d is a power of 2. Let q and p be positive integers and let $\triangle = [q/p]$ and $r_t(q) = q \bmod p$. For any natural number t, $R_t = \mathbb{Z}_t[x]/\langle x^d + 1\rangle$. Let χ be a B-bounded probability distribution over R_q.

- **Key Generation**
  For the key generation, the security parameter λ is given as input and the small secret key $\mathsf{sk} = s \in \chi$ is generated as output. Besides, $a \in R_q$ is generated uniformly at random and $-(a \cdot s + e) \bmod q$ is computed, where $e \in \chi$ is a small random error. The public key $\mathsf{pk} = (p_0, p_1) = (-(a \cdot s + e) \bmod q, a)$ is generated as output.

- **Encryption**
  For the encryption, the message $m \in R_q$ and the $\mathsf{pk} \in R_q^2$ are given as input. The values $u, e_1, e_2 \in \chi$ are chosen at random. The ciphertext $\mathsf{c} = (c_0, c_1)$ is generated as output, where $c_0 = (p_0 \cdot u + e_1 + \triangle m) \bmod q$ and $c_1 = (p_1 \cdot u + e_2) \bmod q$.

- **Evaluation**
  The ciphertext c is assumed as a polynomial evaluated in s instead of a vector with two components.

  - **Homomorphic Addition**
    The addition of two ciphertexts is calculated as

    $$c_1(s) + c_2(s) \bmod q.$$

  - **Homomorphic Multiplication**
    The multiplication of two ciphertexts is calculated as

    $$c_1(s) \cdot c_2(s) = \alpha_0 + \alpha_1 \cdot s + \alpha_2 \cdot s^2.$$

- **Decryption**
  For the decryption, the secret key $s \in R_q$ and the ciphertext $\mathsf{c} \in R_q^2$ are given as input. $[(p \cdot (c_0 + c_1 \cdot s) \bmod q/q] \bmod p$ is computed and message m is generated as output.

CHAPTER 4

METHODOLOGY

In this chapter, the methods that are used to perform privacy-preserving rare disease analysis with fully homomorphic encryption are described. In Section 4.1., the method for the representation of the rare disease patients' genome data is explained. Next, the privacy-preserving rare disease analysis is detailed in Section 4.2. In addition, the details of the methods, namely as boolean circuit method and integer arithmetic method are given in two different subsections.

4.1. Genome Data Representation

The genotype data that contains the sample variant information of different individuals is stored in the Variant Call Format (VCF) files. The types of variants are classified as homozygous variants or heterozygous variants in VCF files. As the first step, the genotype data should be represented in a numerical way to utilize the fully homomorphic encryption method. The variant data of patients are converted to numerical data form that contains only the values of 1 and 0 as follows: 10 for the homozygous variant, 01 for the heterozygous variant, and 00 for the no disease-causing variant case. Some examples of genotype data representation are shown in Table 4.1.

After this process, the transformed genotype data is encrypted to be shared for rare disease research.

4.2. Privacy-Preserving Collaborative Rare Disease Analysis

As mentioned in the previous sections, researchers can not access the genome data of patients easily because of privacy concerns. Although they may have access to the genome data of a small number of rare disease patients, it is usually not enough to analyze rare diseases effectively because the same disease cases with the same phenotype

Table 4.1. The Variant Data and Descriptions

Sample Variant Data in VCF	Variant Data Representation	Description
./.	00	No disease-causing variant
0/0	00	No disease-causing variant
1/0	01	Disease-causing heterozygous variant
1/1	10	Disease-causing homozygous variant
0/2	01	Disease-causing heterozygous variant
2/2	10	Disease-causing homozygous variant

rarely occur in the population. For this reason, the number of patients' genome data that is available to researchers should be increased to enhance the performance of rare disease research. As a solution for this problem, rare disease analysis is proposed to be performed with privacy protection using the fully homomorphic encryption method. Thus, more genome data of rare disease patients can be accessed by researchers, and more effective treatments may be discovered.

In the privacy-preserving collaborative rare disease analysis scenario, the genome data of patients are stored in various hospitals. It is represented numerically and encrypted using various encryption schemes and libraries. The encrypted genome data from different sources is stored in files and shared with a cloud system in a secure way. When a researcher needs access to the genome data of patients for rare disease analysis, an encrypted query that contains an unknown inheritance pattern of variants is sent to the cloud system, and rare disease analysis is performed by filtering the disease-causing variants according to this encrypted query in the huge amount of encrypted genome data using the fully homomorphic encryption method in a privacy-preserving manner. As a result of the variant filtering process, encrypted filtered genome data is sent to the researcher. This process is shown in Figure 4.1. The encrypted genome data of different patients from various hospitals are sent to the cloud system, and on the cloud privacy-preserving rare disease analysis is realized considering the encrypted query sent by the researcher on the encrypted genome data. In this way, disease-causing variants can be detected without revealing the genome data of patients. After that, only the encrypted disease-causing variant data that are filtered as a result of this process are sent to the researcher.

As mentioned in Section 3.1., the variant filtering process is realized through specific steps. In this study, the proposed method is utilized to perform the step of variant filtering using inheritance model-specific filters. The privacy-preserving collaborative rare

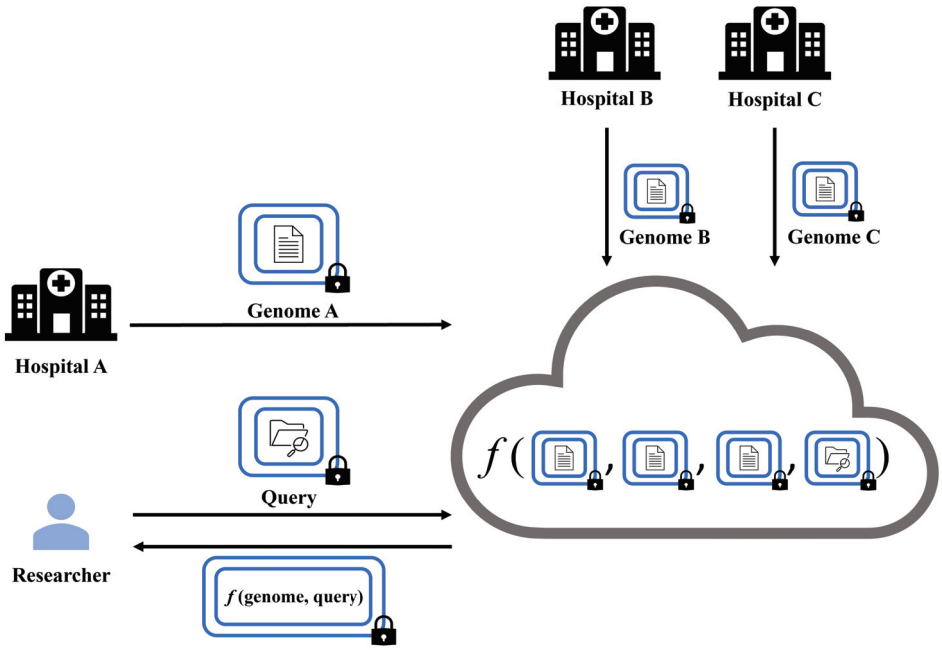

Figure 4.1. Privacy-Preserving Rare Disease Analysis Scenario

disease analysis is provided by employing the variant filtering process on the encrypted genotype data which is extremely confidential data present in the genome. This analysis is performed considering specific inheritance models that are recessive, dominant, and de novo inheritance models. Two different approaches that utilize the fully homomorphic encryption method are employed for secure variant filtering in this study: The boolean circuit method and the integer arithmetic method. These methods are implemented and applied to the encrypted genome data. These approaches are explained in the following subsections.

4.2.1. Boolean Circuit Method

Firstly, boolean operations such as AND, OR, and NOT are performed for privacy-preserving rare disease analysis considering the recessive inheritance model. The details of the boolean circuit method are introduced as follows. Firstly, the encryption details of the genome data are mentioned in Section 4.2.1.1. Furthermore, the implementation details of the boolean circuit method are explained in Section 4.2.1.2.

4.2.1.1. Encryption of Genome Data

To start with, the genotype data represented as binary bits are encrypted using the CGGI scheme [51]. Each bit in the genotype data is encrypted as a ciphertext respectively. Some of the encryption parameters are explained in the following.

- **LWE Dimension** is a parameter that determines the number of scalars in the LWE mask and the length of the LWE secret key.
- **General LWE (GLWE) Dimension** is a parameter that determines the number of polynomials of the GLWE mask and the size of the GLWE secret key.
- **Polynomial Size** refers to the number of coefficients of a polynomial.
- **Standard Deviation** refers to a distribution parameter that uses the standard deviation as the representation.

After the encryption step, encrypted elements are stored in a vector and saved to a file. Thus, the file that contains ciphertexts can be shared for rare disease analysis without any privacy concerns.

4.2.1.2. Implementation

For the rare disease analysis process, boolean gate operations are performed over the encrypted genotype data. Fundamentally, two different boolean circuits are executed. The first boolean circuit is used to filter disease-causing variants in the encrypted genotype data considering the encrypted query that is sent by the researcher. Each element that contains encrypted 1 or encrypted 0 as a value of the genotype data and the query is compared using the first circuit as follows: Firstly, the XNOR operation is applied to each bit of the variants in the genotype and the query. Then, the AND operation is performed on all XNOR results and a comparison result is obtained. The performed operations for the first boolean circuit are shown in Figure 4.2 on the variant data of 4 sample individuals. In the figure, sample 1 and sample 2 are parents that have heterozygous variants, and sample 3 and sample 4 are the affected children that have homozygous disease-causing variants. The variant filtering is performed considering the recessive inheritance model.

The operations given in Figure 4.2. are applied to each variant in the genome data respectively. Thus, an encrypted filtering result is obtained for each variant according to whether it complies with the query.

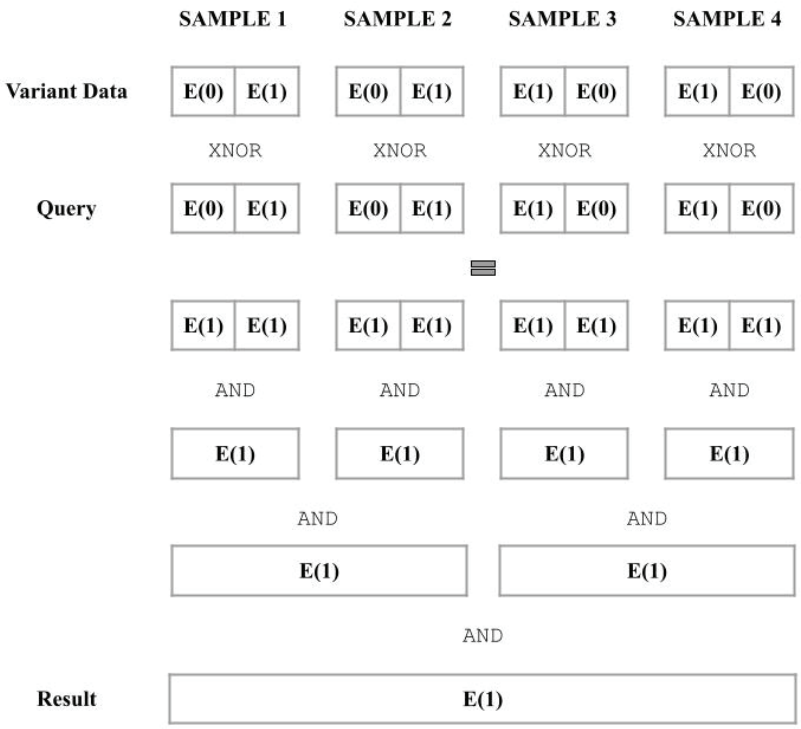

Figure 4.2. Boolean Operations for Variant Filtering Process

Next, another boolean circuit is used to calculate the number of disease-causing variants that comply with the query. The binary addition operation is applied to the filtering results of each variant using XOR and AND operations in this boolean circuit. The performed operations for the second boolean circuit are shown in Figure 4.3. The details for the binary sum operation of three different comparison results are represented in this figure. As a result of these operations, the encrypted sum and the encrypted carry results are obtained.

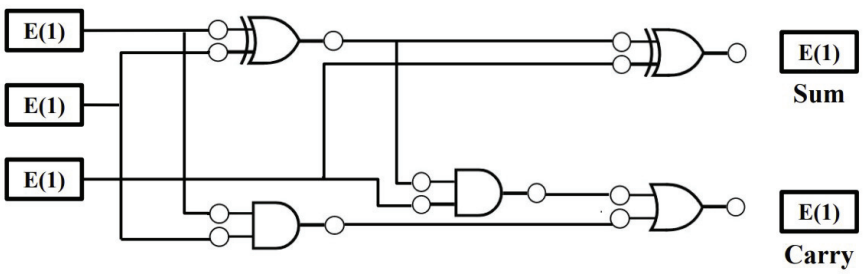

Figure 4.3. Boolean Circuit for Binary Addition

4.2.2. Integer Arithmetic Method

Addition and multiplication arithmetic operations are used to provide secure rare disease analysis by finding the disease-causing variants in the whole genome data. The privacy-preserving rare disease analysis is performed for recessive, dominant, and de novo inheritance models using the integer arithmetic method. The details of the integer arithmetic method are introduced as follows. Firstly, the encryption details of the genome data are mentioned in Section 4.2.2.1. Besides, the implementation details of the integer arithmetic method are explained in Section 4.2.2.2.

4.2.2.1. Encryption of Genome Data

Firstly, the genotype data represented as integer numbers are encrypted using the BFV scheme [48]. The genotype data of patients are encrypted by dividing the whole data into blocks. The elements of each block are stored in a vector and each vector is encoded as plaintext. And finally, obtained plaintext is encrypted as a ciphertext. Some of the parameters used for the encryption are explained in the following.

- **Plaintext Modulus** refers to a prime number that determines the maximum value for the elements of the plaintext.
- **Ciphertext Modulus** is the parameter that determines how many computations are allowed according to the value of the multiplicative depth.
- **Ring Dimension** is the parameter that determines the maximum number of elements in the polynomial ring.
- **Multiplicative Depth** is the parameter to determine the maximum number of operations that can be performed on the ciphertext without the noise level exceeding the threshold.

After the encryption process, each ciphertext vector is stored in a file and these files can be shared for the privacy-preserving rare disease analysis process.

4.2.2.2. Implementation

For the rare disease analysis process, arithmetic operations are performed on the encrypted genotype data in files in a Single Instruction Multiple Data (SIMD) manner. Since each encrypted vector consists of many integer values, multiple elements can be processed by applying a single addition or multiplication operation to a vector.

Firstly, an arithmetic expression is used for the comparison of the query and all variants in the genome data that consists of encrypted 1 or 0 values considering recessive and dominant inheritance models. Each block of variants and the query are compared using the following operations: The multiplication of the query and the block of sample variant data is calculated. Besides, the multiplication of the inverse of the query and the inverse of the block is performed. The inverse of an element is calculated as the subtraction of this element from 1. Then, these two multiplication operation results are summed. Additionally, the comparison results of the first index and the second index of the sample variant data are multiplied. These operations are applied to all samples in the block. After this operation, the comparison results of all samples are multiplied and a vector that contains comparison results is obtained for the variant data block. The algorithm of this process is given below.

Algorithm 1: Variant Filtering (Recessive and Dominant)

Data: V is a vector of variant samples, V_j is a variant sample, V'_j is the inverse of a variant sample, Q is a vector of query elements, Q_j is an element of query, Q'_j is the inverse of a query element, s is the number of samples, b is the number of blocks, R_j is a filtering result for a sample, R_s is a vector of filtering results for all samples, R_i is a vector of filtering result for a block

Result: R is a vector of filtering results for all blocks

```
1  for i ← 1 to b do
2      for j ← 1 to 2 × s do
3          M ← V_j × Q_j;    // Multiplication of variant sample and the query element
4          V'_j ← 1 − V_j;    // Inverse of variant sample
5          Q'_j ← 1 − Q_j;    // Inverse of query element
6          M' ← V'_j × Q'_j;    // Multiplication of inverses
7          R_j ← M + M';    // Sum of multiplication results (filtering result)
8          append(R_s, R_j);
9      end
10     R_i ← EvalMultMany(R_s);    // Multiplication of filtering results (all samples)
11     append(R, R_i) ;
12 end
```

Furthermore, the operations for the variant filtering process of the variant data of a sample individual considering the query that contains a heterozygous variant are shown in Figure 4.4. In this figure, the operations are applied to a block of 5 variants for simple representation. The operations are performed for all samples and a comparison result is obtained for each sample. Next, all comparison results are multiplied. As a result of this process, disease-causing variants that comply with the query are specified. Moreover, the comparison results of all variant data blocks are summed, and an encrypted result that represents the number of disease-causing variants that match the query is obtained.

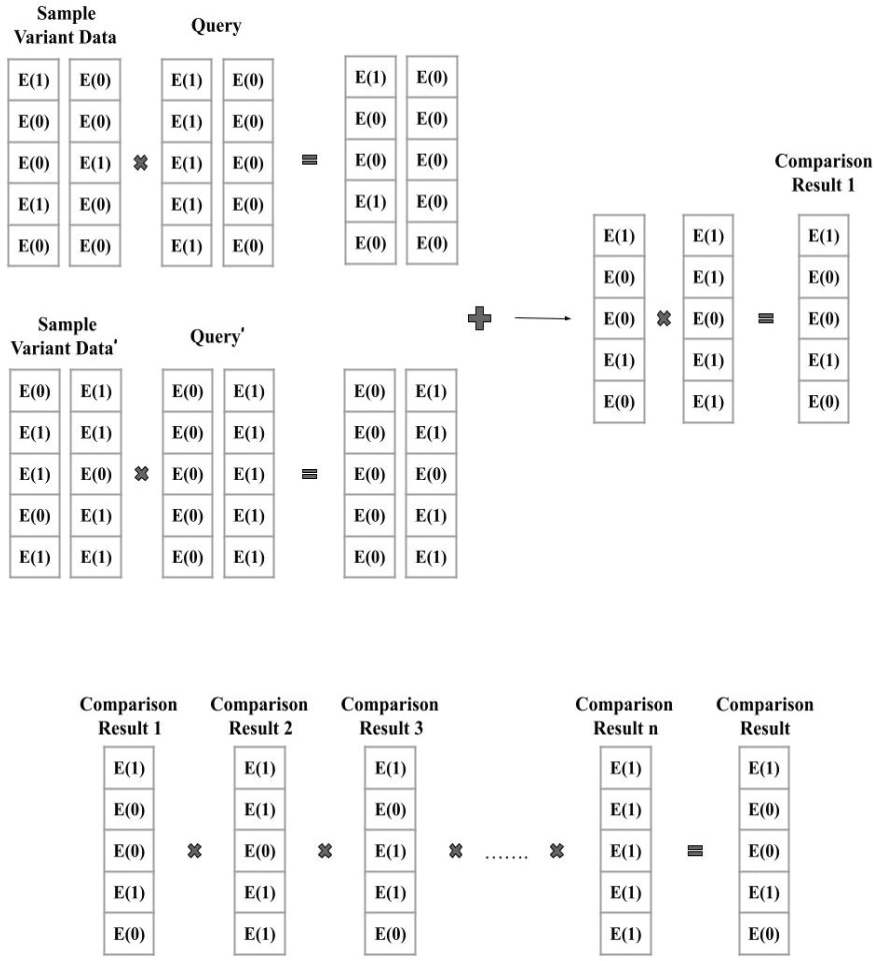

Figure 4.4. Comparison Process of a Block of Variant Data using Integer Arithmetic Method 1 for Recessive and Dominant Inheritance Models

Secondly, another arithmetic expression is utilized to decrease the number of multiplication operations for the comparison process of the query and the variants in the genome data considering recessive and dominant inheritance models. In the expression, the following operations are performed for each block of variant data: Firstly, the addition

operation is applied to the block of sample variant data. The value of the block is summed with itself and this addition result is multiplied by the value of the query. After that, the addition of variant data block and the query is calculated and subtracted from the multiplication result that is obtained in the previous step. Next, the value of encrypted 1 is added to the final result. Lastly, the final results of the first index and the second index of the sample variant data are multiplied, and the comparison result is obtained for the block of sample variant data. The algorithm of this process is given below.

Algorithm 2: Variant Filtering (Recessive and Dominant)

Data: V is a vector of variant samples, V_j is a variant sample, Q is a vector of query elements, Q_j is an element of query, s is the number of samples, b is the number of blocks, R_j is a filtering result for a sample, R_s is a vector of filtering results for all samples, R_i is a vector of filtering result for a block

Result: R is a vector of filtering results for all blocks

```
1  for i ← 1 to b do
2      for j ← 1 to 2 × s do
3          A ← V_j + V_j;   // Addition of the sample variant with itself
4          M ← A × Q_j;   // Multiplication of addition result and query element
5          S ← M − V_j − Q_j;   // Subtraction of variant and query element
6          R_j ← S + 1;   // Addition of subtraction result and 1 (filtering result)
7          append(R_s, R_j) ;
8      end
9      R_i ← EvalMultMany(R_s);   // Multiplication of filtering results (all samples)
10     append(R, R_i) ;
11 end
```

Furthermore, the performed operations for the variant filtering process of the variant data of a sample individual considering the query that contains a heterozygous variant are shown in Figure 4.5. As mentioned before, the operations are applied to a block of 5 variants for simple representation. These operations are performed for all samples and a comparison result is obtained for each sample. After that, all comparison results are multiplied, and disease-causing variants are filtered. After the comparison results are obtained for each variant data block, the comparison results of all blocks are summed, and an encrypted number of disease-causing variants that complies with the query is obtained.

Finally, another arithmetic expression is used for rare disease analysis considering the de novo inheritance model. The arithmetic operations are performed on each block of variant data as follows: First, sample variant data of individuals are specified by whether

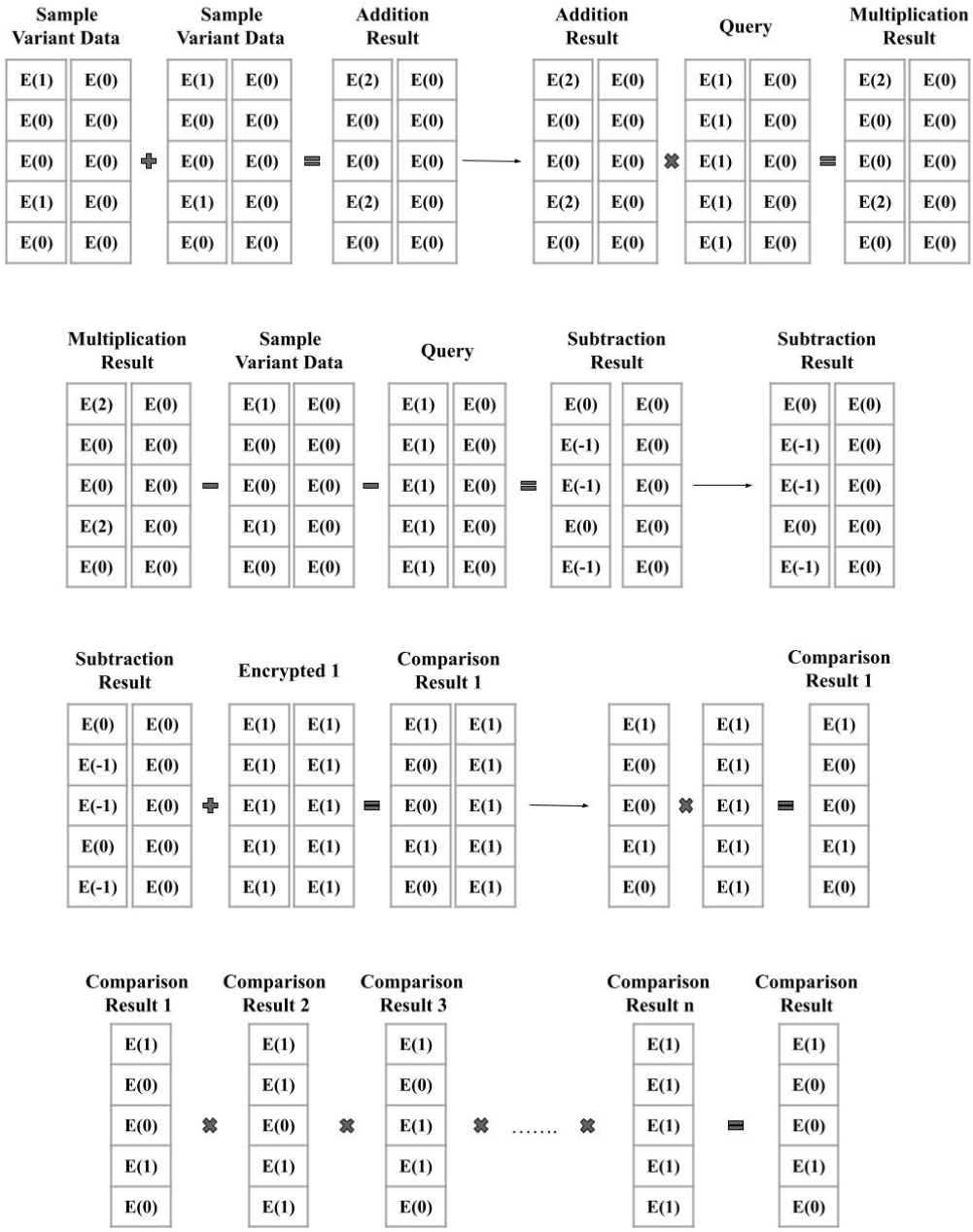

Figure 4.5. Comparison Process of a Block of Variant Data using Integer Arithmetic Method 2 for Recessive and Dominant Inheritance Models

or not they are affected by the rare disease. The inverse of the sample variant data of unaffected individuals is calculated by subtracting the data from the encrypted 1. After that, the first index and the second index of the sample variant data of affected and unaffected individuals are multiplied separately. Lastly, these two multiplication results are summed. Thus, the filtering results are obtained considering whether the variant data complies with the de novo inheritance model or not. The algorithm of this process is given in Algorithm 3.

Algorithm 3: Variant Filtering (De Novo)

Data: A is a vector of affected variant samples, U is a vector of unaffected variant samples, U_j is an unaffected variant sample, U'_j is the inverse of an unaffected variant sample, a is the number of affected samples, u is the number of unaffected samples, b is the number of blocks, R_u is a vector of unaffected sample inverses, R_i is a vector of filtering result for a block

Result: R is a vector of filtering results for all blocks

```
for i ← 1 to b do
    M_a ← EvalMultMany(A);    // Multiplication of affected samples
    for j ← 1 to u do
        U'_j ← 1 − U_j;    // Inverse of unaffected sample
        append(R_u, U'_j) ;
    end
    M_u ← EvalMultMany(R_u);    // Multiplication of inverses
    M ← M_a × M_u;    // Multiplication of the multiplication results
    R_i ← M_0 + M_1;    // Sum of the first and second indices (filtering result)
    append(R, R_i) ;
end
```

Furthermore, the performed operations for the variant filtering process of the variant data of sample affected and unaffected individuals are shown in Figure 4.6. In this figure, the disease-causing variants that are present in all affected individuals and are not present in unaffected individuals are detected using the applied operations. These operations are performed for all blocks of variant data. After the filtering results are obtained for each variant data block, the filtering results of all blocks are summed to obtain the number of disease-causing variants that match the case of de novo.

Finally, the details of the sum operation that is applied to the filtering results of the variant data blocks can be explained as follows. Firstly, the results of all variant data blocks are summed. After that, all elements of the vector that contains the sum result are summed and the encrypted number of filtered variants is obtained as a result of this

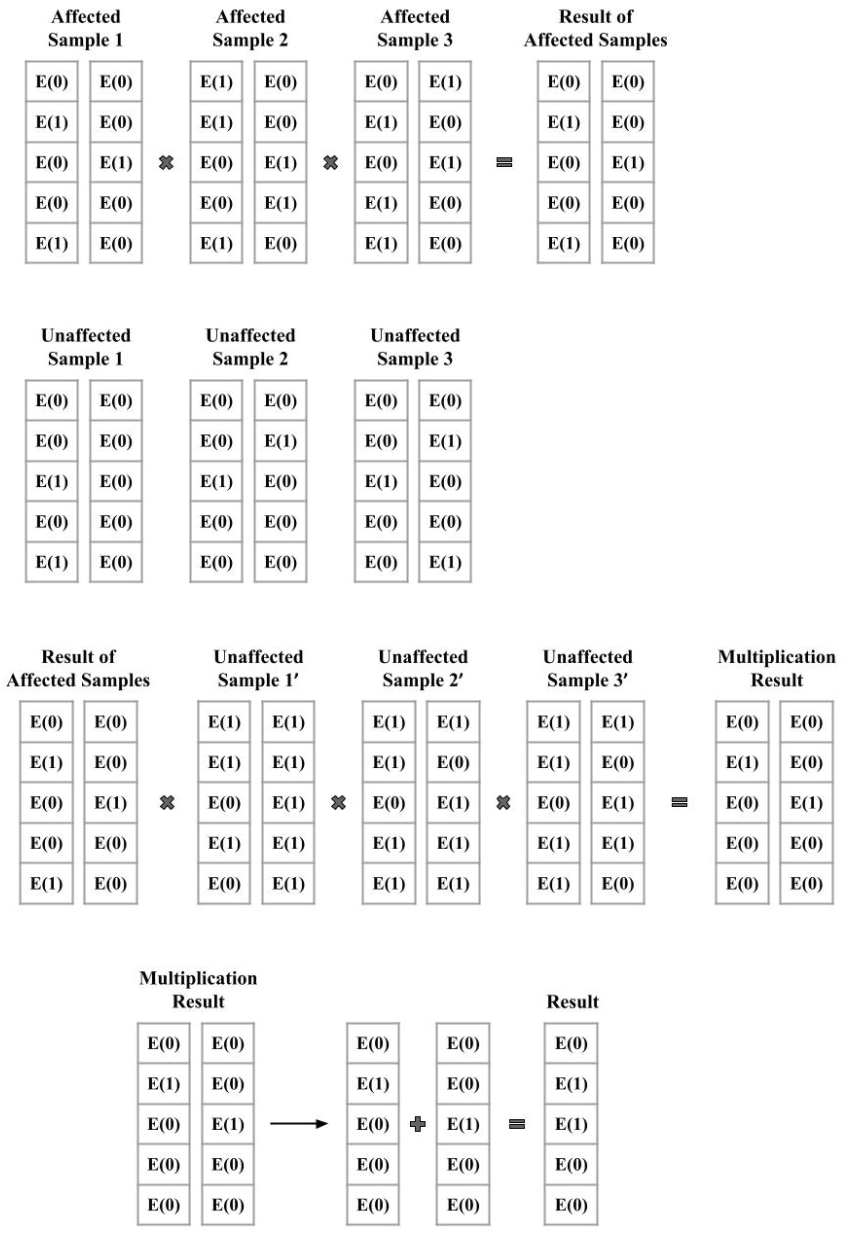

Figure 4.6. Filtering Process of a Block of Variant Data for De Novo Inheritance Model

process. A sample sum operation is shown in Figure 4.7. The results of the five blocks each containing five different variant data are summed, and the encrypted value of 5 is obtained as the number of matching variants that cause rare diseases.

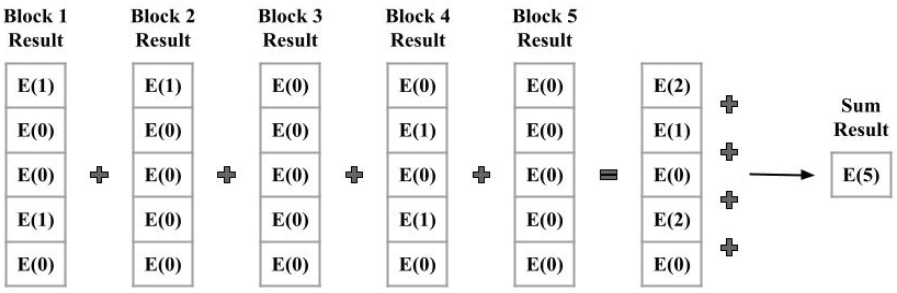

Figure 4.7. Sum Operation of the Block Results

CHAPTER 5

EXPERIMENTS AND RESULTS

The performed experiments for privacy-preserving rare disease analysis and the corresponding results are explained in this chapter. Firstly, the boolean circuit experiments and their results are discussed in the first section. Two different experiments are performed using the proposed boolean circuits considering the recessive inheritance model, and the effect of using different numbers of threads is discussed. In the second section, the integer arithmetic experiments and their results are explained. Three different integer arithmetic experiments are performed. The first and the second experiments are performed considering recessive and dominant inheritance models, and the third experiment is performed considering the de novo inheritance model. The effect of using different parameters for the encryption is discussed, and the results of using different numbers of variants and samples for the integer arithmetic experiments are presented. Finally, in the third section, the overall results of the performed experiments and the efficiency of the proposed solutions are discussed.

5.1. Boolean Circuit Experiments

In this section, the experiments performed using the boolean circuit method are discussed. The privacy-preserving rare disease analysis is performed considering the recessive inheritance model in these experiments.

5.1.1. Experimental Setup

The experiments are conducted in a Ubuntu Server with 6 Intel Xeon Gold 6254 CPUs each having 18 cores (108 cores in total). Furthermore, the sample genome data used for experiments are generated as the genome data of patients are difficult to access because of privacy concerns.

The boolean circuit method is implemented using a Rust library called Concrete [54]. Concrete-Boolean which is a unit of the Concrete library is utilized. Concrete-Boolean provides tools to execute boolean gates over encrypted bits. The Concrete library implements a variant of Fully Homomorphic Encryption over the Torus (TFHE). TFHE [55] is a library that is implemented in C/C++ and based on the CGGI scheme [51].

The genome data represented as binary bits are encrypted using the Concrete-Boolean library. Each bit is encrypted as a ciphertext using the default parameter values provided by the library with the 128-bit security level. The values of parameters used for the encryption of the genome data are shown in Table 5.1.

Table 5.1. The Encryption Parameters for Boolean Circuit Experiments

LWE Dimension	GLWE Dimension	Polynomial Size
777	3	512

5.1.2. Experiments

For the first experiment, rare disease analysis is realized for 1,000 variants containing sample data of 4 individuals using various numbers of threads to determine the most efficient number of threads. Firstly, the variant filtering process is performed for a specific part of the encrypted genome data concurrently in each thread and the comparison results are obtained for each part. Next, the binary addition operation is applied to each comparison result concurrently in each thread. After that, the pairs of obtained results are summed simultaneously. The same operation is applied recursively and an encrypted sum result is obtained after these operations. The computation time results are shown in Table 5.2.

As seen in Table 5.2., the most efficient computation time results are obtained using 64 threads for rare disease analysis of 1,000 variants. The computation time decreases with the increase in the number of threads. The curve of the change in the computation time with the use of an increasing number of threads is shown in Figure 5.1.

Furthermore, another experiment is performed using 16, 32, and 64 threads for the increasing number of variants. In this experiment, different numbers of variants containing sample data from 4 individuals are analyzed. The computation time results of the variant filtering process and sum operation for the number of variants of 2,000, 4,000, 8,000, and

Table 5.2. Computation Time Results of Boolean Circuit Experiment 1

Boolean Circuit Experiment 1			
Number of Variants	**Number of Samples**	**Number of Threads**	**Computation Time**
1,000	4	1	9 min 32 s
1,000	4	2	4 min 32 s
1,000	4	4	2 min 31 s
1,000	4	8	1 min 43 s
1,000	4	16	1 min 7 s
1,000	4	32	44 s
1,000	4	64	29 s

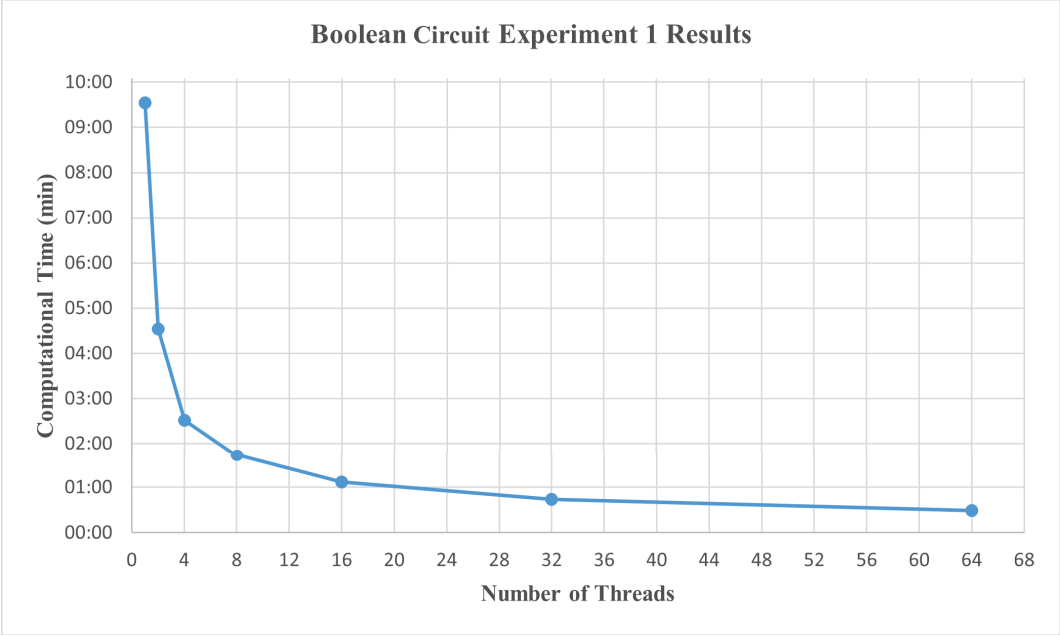

Figure 5.1. Computation Time Results of Boolean Experiment 1

Table 5.3. Computation Time Results of Boolean Circuit Experiment 2

Boolean Circuit Experiment 2			
Number of Variants	**Number of Samples**	**Number of Threads**	**Computation Time**
2,000	4	16	2 min 17 s
2,000	4	32	1 min 32 s
2,000	4	64	1 min 1 s
4,000	4	16	4 min 39 s
4,000	4	32	3 min 8 s
4,000	4	64	2 min 5 s
8,000	4	16	9 min 18 s
8,000	4	32	6 min 21 s
8,000	4	64	4 min 13 s
10,000	4	16	11 min 27 s
10,000	4	32	7 min 41 s
10,000	4	64	5 min 7 s

10,000 are provided in Table 5.3.

According to the given results in Table 5.3., the increase in the number of threads provides more efficient results for the increasing number of variants. The use of 64 threads provides the most efficient results. The comparison of the computation time results of 16, 32, and 64 threads for the increasing number of variants is shown in Figure 5.2.

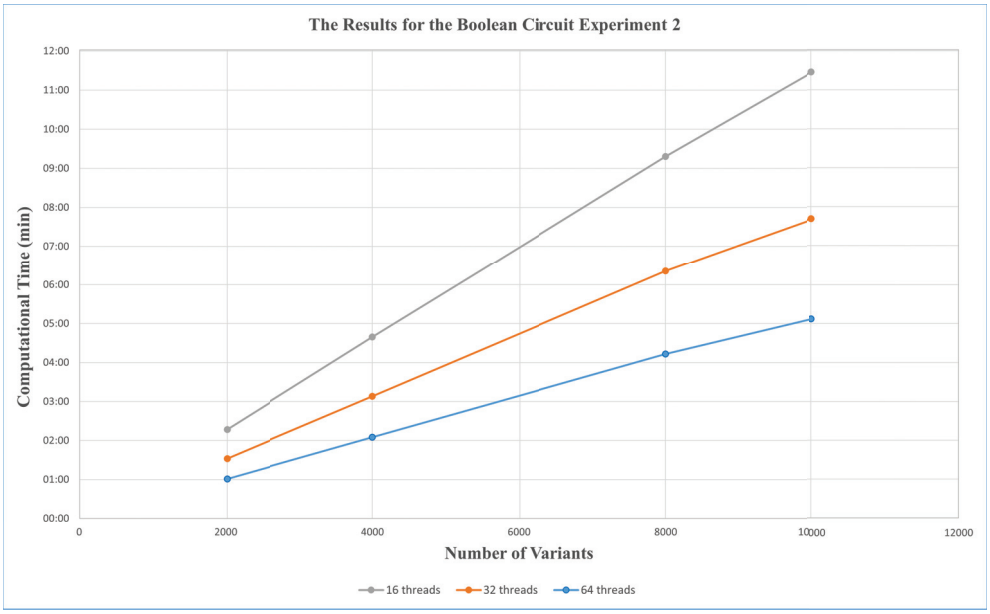

Figure 5.2. The Comparison of Computation Time Results of Boolean Experiment 2 for The Different Number of Threads

5.2. Integer Arithmetic Experiments

The performed experiments using the integer arithmetic method are explained in this section. Three different experiments that utilize arithmetic operations are performed for recessive, dominant, and de novo inheritance models.

5.2.1. Experimental Setup

The experiments are conducted in a computer that has a 2.60 GHz Intel Core i7-10750H CPU and 16 GB RAM. Besides, the sample genome data used for experiments are generated as the genome data of patients are difficult to access because of privacy concerns.

For the implementation of the integer arithmetic method, a C++ library called OpenFHE [56] is utilized. OpenFHE is an FHE library that enables performing computations over encrypted data and has implementations of some FHE schemes such as BFV, BGV, CKKS, and CGGI. The BFV-RNS [49] implementation of OpenFHE is used to perform addition and multiplication operations over the encrypted data for this method.

The genome data represented as integer numbers are encrypted using the BFV-RNS implementation of the OpenFHE library. The data is divided into blocks because of its large amount and the elements of each block are stored in a vector. Then, each vector is encoded as plaintext using the packed encoding type. The packed encoding provides packing integer elements into a vector. After that, obtained plaintext is encrypted as a ciphertext using various encryption parameters with the 128-bit security level. The values of parameters used for the encryption of the genome data for different block sizes are shown in Table 5.4.

Table 5.4. The Encryption Parameters for Integer Arithmetic Experiments

Block Size	Plaintext Modulus	Ring Dimension	Multiplicative Depth
25,000	7,340,033	65,536	6
50,000	7,340,033	131,072	6
100,000	7,340,033	262,144	6

The relation between the encryption parameters is ruled by the following formula in the library, where p is the plaintext modulus and N is the ring dimension:

$$1 \equiv p \bmod 2N$$

The ring dimension values are selected as a power of 2 which is greater than 2 times the block size value. The value of the plaintext modulus is 7,340,033 considering the above rule, which means the maximum possible value as the result of the computations is 7,340,033. This value is selected considering the millions of variants that are processed. Furthermore, the multiplicative depth is determined as 6, which means the maximum value of multiplication operations that can be performed on a ciphertext is $2^6 = 64$.

5.2.2. Experiments

As the first experiment, rare disease analysis is provided for various numbers of variants and individuals considering the recessive and dominant inheritance models, and the first arithmetic expression shown in Figure 4.4. is used as follows: The effect of ciphertext size is analyzed using 3 different block sizes. The variant data is encrypted by dividing it into blocks of sizes 25,000, 50,000, and 100,000 respectively. In addition, the corresponding values of 65,536, 131,072, and 262,144 are used as the ring dimension parameter for these block sizes. The value of 6 is used for the multiplicative depth parameter, and the value of the plaintext modulus parameter is determined as 7,340,033. The computation time results of the Integer Arithmetic Experiment 1 for the different block sizes are shown in Table 5.5.

Table 5.5. Computation Time Results of Integer Arithmetic Experiment 2 for Recessive and Dominant Inheritance Models

Experiment 1 for Recessive and Dominant Inheritance Models				
Number of Variants	**Block Size**	**Number of Samples**	**Variant Filtering**	**Sum**
100,000	25,000	4	18 s 14 ms	893 ms
100,000	50,000	4	19 s 419 ms	2 s 22 ms
100,000	100,000	4	20 s 807 ms	4 s 572 ms
100,000	25,000	8	36 s 115 ms	893 ms
100,000	50,000	8	38 s 182 ms	2 s 39 ms
100,000	100,000	8	40 s 640 ms	4 s 142 ms
100,000	25,000	16	1 min 16 s 635 ms	971 ms

(cont. on next page)

Table 5.5. Computation Time Results of Integer Arithmetic Experiment 2 for Recessive and Dominant Inheritance Models (cont.)

Number of Variants	Block Size	Number of Samples	Variant Filtering	Sum
100,000	50,000	16	1 min 18 s 724 ms	2 s 100 ms
100,000	100,000	16	1 min 22 s 414 ms	4 s 317 ms
100,000	25,000	32	2 min 40 s 990 ms	930 ms
100,000	50,000	32	2 min 49 s 975 ms	2 s 206 ms
100,000	100,000	32	2 min 57 s 661 ms	4 s 494 ms
200,000	25,000	4	38 s 68 ms	953 ms
200,000	50,000	4	38 s 762 ms	2 s 112 ms
200,000	100,000	4	41 s 489 ms	4 s 714 ms
200,000	25,000	8	1 min 17 s 290 ms	965 ms
200,000	50,000	8	1 min 27 s 853 ms	2 s 183 ms
200,000	100,000	8	1 min 31 s 102 ms	4 s 828 ms
200,000	25,000	16	2 min 39 s 236 ms	968 ms
200,000	50,000	16	2 min 55 s 656 ms	2 s 210 ms
200,000	100,000	16	3 min 121 ms	4 s 631 ms
200,000	25,000	32	5 min 21 s 168 ms	982 ms
200,000	50,000	32	5 min 42 s 613 ms	2 s 260 ms
200,000	100,000	32	6 min 16 s 408 ms	4 s 726 ms
400,000	25,000	4	1 min 14 s 926 ms	987 ms
400,000	50,000	4	1 min 23 s 550 ms	2 s 237 ms
400,000	100,000	4	1 min 24 s 373 ms	4 s 770 ms
400,000	25,000	8	2 min 27 s 335 ms	878 ms
400,000	50,000	8	2 min 46 s 110 ms	2 s 194 ms
400,000	100,000	8	2 min 58 s 887 ms	4 s 565 ms
400,000	25,000	16	5 min 17 s 700 ms	980 ms
400,000	50,000	16	5 min 35 s 778 ms	2 s 218 ms
400,000	100,000	16	5 min 56 s 260 ms	4 s 789 ms
400,000	25,000	32	10 min 19 s 112 ms	941 ms
400,000	50,000	32	11 min 17 s 431 ms	1 s 980 ms
400,000	100,000	32	11 min 24 s 988 ms	4 s 493 ms
800,000	25,000	4	2 min 36 s 501 ms	981 ms
800,000	50,000	4	2 min 38 s 778 ms	2 s 78 ms
800,000	100,000	4	3 min 5 s 105 ms	5 s 211 ms

(cont. on next page)

Table 5.5. Computation Time Results of Integer Arithmetic Experiment 2 for Recessive and Dominant Inheritance Models (cont.)

Number of Variants	Block Size	Number of Samples	Variant Filtering	Sum
800,000	25,000	8	5 min 8 s 784 ms	988 ms
800,000	50,000	8	5 min 20 s 858 ms	2 s 354 ms
800,000	100,000	8	5 min 49 s 423 ms	4 s 459 ms
800,000	25,000	16	10 min 26 s 472 ms	990 ms
800,000	50,000	16	11 min 5 s 997 ms	2 s 254 ms
800,000	100,000	16	11 min 38 s 179 ms	4 s 593 ms
800,000	25,000	32	21 min 707 ms	1 s 2 ms
800,000	50,000	32	22 min 14 s 713 ms	2 s 232 ms
800,000	100,000	32	24 min 9 s 652 ms	4 s 698 ms
1,600,000	25,000	4	5 min 11 s 40 ms	1 s 86 ms
1,600,000	50,000	4	5 min 21 s 361 ms	2 s 194 ms
1,600,000	100,000	4	5 min 44 s 118 ms	4 s 684 ms
1,600,000	25,000	8	10 min 23 s 622 ms	1 s 61 ms
1,600,000	50,000	8	11 min 0 s 473 ms	2 s 290 ms
1,600,000	100,000	8	11 min 44 s 667 ms	4 s 719 ms
1,600,000	25,000	16	21 min 2 s 797 ms	1 s 59 ms
1,600,000	50,000	16	22 min 8 s 517 ms	2 s 299 ms
1,600,000	100,000	16	23 min 59 s 579 ms	4 s 603 ms
1,600,000	25,000	32	43 min 35 s 843 ms	1 s 82 ms
1,600,000	50,000	32	44 min 31 s 31 ms	2 s 297 ms
1,600,000	100,000	32	47 min 55 s 293 ms	4 s 724 ms
3,200,000	25,000	4	10 min 20 s 609 ms	1 s 387 ms
3,200,000	50,000	4	11 min 15 s 354 ms	2 s 593 ms
3,200,000	100,000	4	12 min 19 s 722 ms	4 s 989 ms
3,200,000	25,000	8	21 min 10 s 205 ms	1 s 282 ms
3,200,000	50,000	8	22 min 24 s 502 ms	2 s 439 ms
3,200,000	100,000	8	23 min 19 s 868 ms	4 s 841 ms
3,200,000	25,000	16	42 min 46 s 819 ms	1 s 218 ms
3,200,000	50,000	16	44 min 54 s 287 ms	2 s 420 ms
3,200,000	100,000	16	47 min 29 s 696 ms	4 s 887 ms
3,200,000	25,000	32	1 h 25 min 15 s 858 ms	1 s 146 ms
3,200,000	50,000	32	1 h 30 min 2 s 671 ms	2 s 366 ms

(cont. on next page)

Table 5.5. Computation Time Results of Integer Arithmetic Experiment 2 for Recessive and Dominant Inheritance Models (cont.)

Number of Variants	Block Size	Number of Samples	Variant Filtering	Sum
3,200,000	100,000	32	1 h 35 min 28 s 157 ms	4 s 801 ms

According to the given results in Table 5.5., it can be stated that the most efficient results are obtained with the block size value of 25,000. The ciphertext size decreases with a smaller value of block size, which allows operations to be performed more efficiently. The comparison of the computation time results of the variant filtering process of 1,600,000 variants for the block size values 25,000, 50,000, and 100,000 with the increasing number of samples is shown in Figure 5.3.

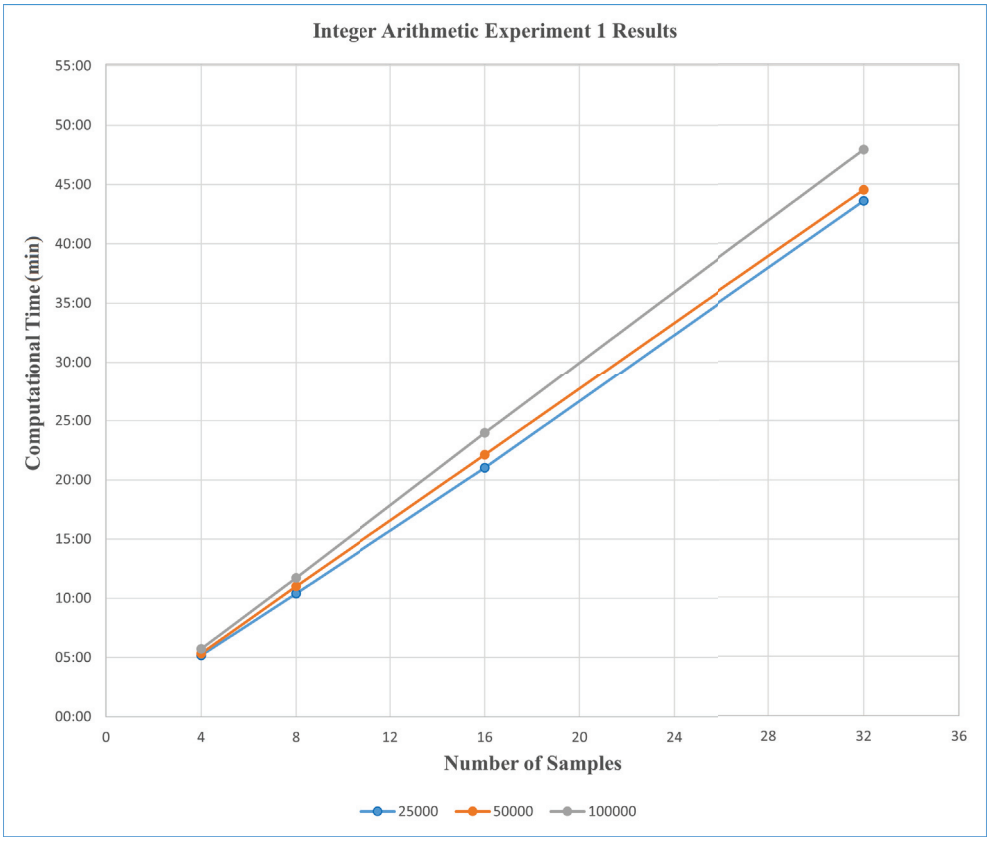

Figure 5.3. The Comparison of Computation Time Results of Integer Arithmetic Experiment 1 for The Different Numbers of Block Size

As seen in Table 5.5., the computation time increases linearly with the increasing number of variants and samples. The number of samples and variants is multiplied by 2 constantly and as a result of this, the computation time is approximately doubled.

The relative comparison of computation time results for various numbers of samples and variants using the block size value 25,000 is shown in Table 5.6. In this table, the value of x is 18 seconds, which is the computation time result for 100,000 variants and 4 samples.

Table 5.6. Relative Comparison of Computation Time Results for Different Numbers of Samples and Variants

		Number of Samples			
		4	**8**	**16**	**32**
Number of Variants	**100,000**	1x	2x	4.22x	8.89x
	200,000	2.11x	4.28x	8.83x	17.83x
	400,000	4.11x	8.17x	17.61x	34.39x
	800,000	8.67x	17.11x	34.78x	70x
	1,600,000	17.28x	34.61x	70.11x	145.28x
	3,200,000	34.44x	70.55x	142.55x	284.17x

Furthermore, another experiment is performed to realize the privacy-preserving rare disease analysis considering recessive and dominant inheritance models. The second arithmetic expression shown in Figure 4.5. is utilized to enhance the computation time results by decreasing the number of multiplication operations. For this experiment, only the block size value of 25,000 is used considering the results of the first experiment, and the value of 65,536 is used as the corresponding ring dimension. Moreover, the value of multiplicative depth is kept as 6, and the value of plaintext modulus as 7,340,033. The computation time results of this experiment are shown in Table 5.7.

According to the given results in Table 5.7., the computation time results are improved using the second arithmetic expression for the variant filtering process. The difference in the computation time results of the first and the second integer arithmetic experiments for the value of 1,600,000 variants with the increasing number of samples is shown in Figure 5.4.

Finally, as the third experiment, privacy-preserving rare disease analysis is performed considering the de novo inheritance model. The last arithmetic expression shown in Figure 4.6. is used to filter the variant data that complies with the de novo inheritance model. The block size value is kept as 25,000 and the ring dimension value is kept as 65,536 same as the second integer arithmetic experiment. Also, the multiplicative depth and plaintext modulus values are kept as 6 and 7,340,033 respectively. The computation time results are shown in Table 5.8. for this experiment.

According to the given computation time results in Table 5.8., it can be stated the

Table 5.7. Computation Time Results of Integer Arithmetic Experiment 2 for Recessive and Dominant Inheritance Models

Integer Arithmetic Experiment 2				
Number of Variants	**Block Size**	**Number of Samples**	**Variant Filtering**	**Sum**
100,000	25,000	4	12 s 973 ms	887 ms
100,000	25,000	8	25 s 910 ms	796 ms
100,000	25,000	16	55 s 194 ms	1 s 81 ms
100,000	25,000	32	1 min 50 s 511 ms	964 ms
200,000	25,000	4	23 s 882 ms	799 ms
200,000	25,000	8	53 s 617 ms	937 ms
200,000	25,000	16	1 min 48 s 601 ms	1 s 28 ms
200,000	25,000	32	3 min 45 s 797 ms	974 ms
400,000	25,000	4	47 s 785 ms	814 ms
400,000	25,000	8	1 min 44 s 649 ms	934 ms
400,000	25,000	16	3 min 43 s 588 ms	983 ms
400,000	25,000	32	7 min 42 s 154 ms	989 ms
800,000	25,000	4	1 min 37 s 185 ms	891 ms
800,000	25,000	8	3 min 43 s 642 ms	1 s 23 ms
800,000	25,000	16	7 min 37 s 201 ms	1 s 16 ms
800,000	25,000	32	15 min 28 s 260 ms	1 s 27 ms
1,600,000	25,000	4	3 min 37 s 768 ms	1 s 69 ms
1,600,000	25,000	8	7 min 15 s 438 ms	1 s 71 ms
1,600,000	25,000	16	14 min 53 s 549 ms	1 s 57 ms
1,600,000	25,000	32	31 min 2 s 383 ms	968 ms
3,200,000	25,000	4	7 min 28 s 117 ms	1 s 342 ms
3,200,000	25,000	8	15 min 7 s 791 ms	1 s 221 ms
3,200,000	25,000	16	31 min 5 s 171 ms	1 s 212 ms
3,200,000	25,000	32	1 h 2 min 21 s 682 ms	1 s 144 ms
6,400,000	25,000	4	15 min 9 s 741 ms	1 s 675 ms
6,400,000	25,000	8	30 min 39 s 423 ms	1 s 628 ms
6,400,000	25,000	16	1 h 2 min 48 s 605 ms	1 s 523 ms
6,400,000	25,000	32	2 h 4 min 6 s 285 ms	1 s 505 ms

Table 5.8. Computation Time Results of Integer Arithmetic Experiment for De Novo Inheritance Model

Integer Arithmetic Experiment 3				
Number of Variants	**Block Size**	**Number of Samples**	**Variant Filtering**	**Sum**
100,000	25,000	4	6 s 826 ms	891 ms
100,000	25,000	8	14 s 158 ms	802 ms
100,000	25,000	16	30 s 699 ms	891 ms
100,000	25,000	32	1 min 3 s 197 ms	909 ms
200,000	25,000	4	13 s 320 ms	891 ms
200,000	25,000	8	29 s 695 ms	890 ms
200,000	25,000	16	1 min 2 s 323 ms	910 ms
200,000	25,000	32	2 min 14 s 155 ms	962 ms
400,000	25,000	4	27 s 405 ms	898 ms
400,000	25,000	8	59 s 187 ms	916 ms
400,000	25,000	16	2 min 1 s 478 ms	944 ms
400,000	25,000	32	4 min 21 s 68 ms	963 ms
800,000	25,000	4	54 s 952 ms	880 ms
800,000	25,000	8	2 min 1 s 404 ms	975 ms
800,000	25,000	16	4 min 18 s 619 ms	1 s
800,000	25,000	32	8 min 44 s 816 ms	1 s 4 ms
1,600,000	25,000	4	1 min 47 s 51 ms	1 s 54 ms
1,600,000	25,000	8	4 min 7 s 819 ms	1 s 63 ms
1,600,000	25,000	16	8 min 39 s 772 ms	1 s 54 ms
1,600,000	25,000	32	17 min 58 s 160 ms	1 s 55 ms
3,200,000	25,000	4	3 min 51 s 247 ms	1 s 319 ms
3,200,000	25,000	8	8 min 27 s 290 ms	1 s 223 ms
3,200,000	25,000	16	16 min 56 s 955 ms	1 s 200 ms
3,200,000	25,000	32	35 min 57 s 265 ms	1 s 181 ms
6,400,000	25,000	4	7 min 53 s 108 ms	1 s 599 ms
6,400,000	25,000	8	16 min 53 s 502 ms	1 s 586 ms
6,400,000	25,000	16	34 min 49 s 47 ms	1 s 507 ms
6,400,000	25,000	32	1 h 11 min 51 s 778 ms	1 s 565 ms

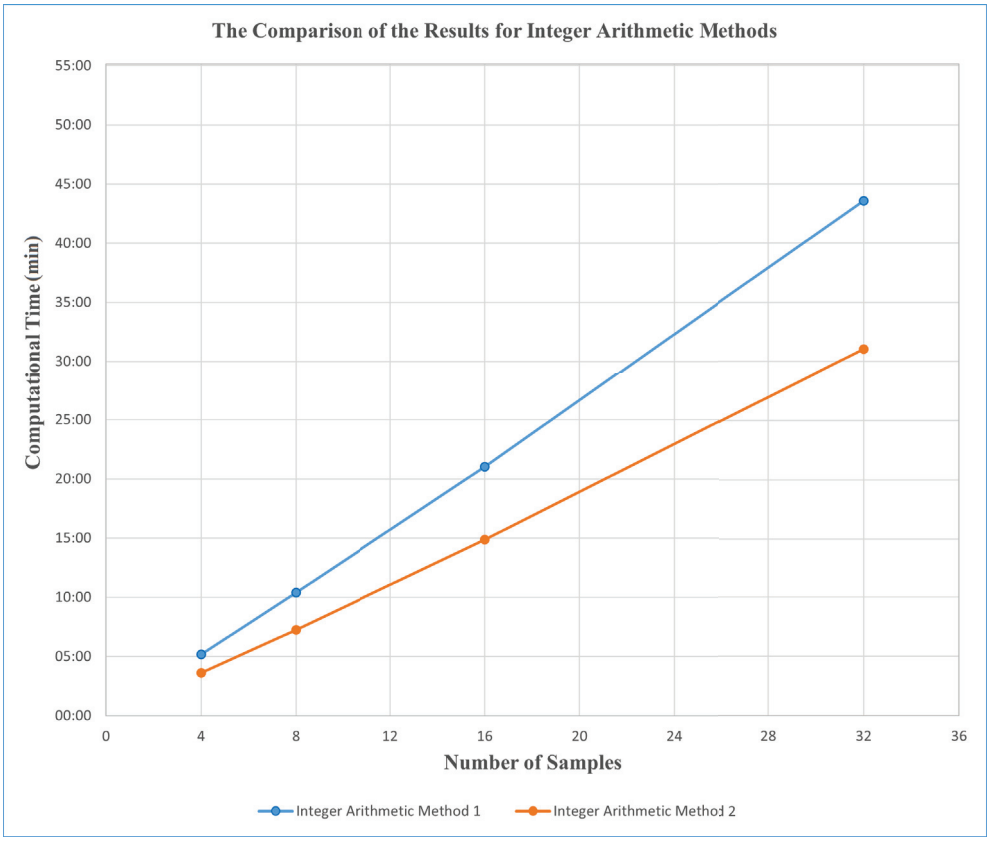

Figure 5.4. The Comparison of Computation Time Results for Integer Arithmetic Methods

duration of the variant filtering process for the de novo inheritance model is less than the duration for recessive and dominant inheritance models. The reason is that there is no query to compare with the variant data for the de novo inheritance model, and this results in a decrease in the number of operations. Thus, the disease-causing variants that comply with the de novo inheritance model are filtered in less computation time. The computation time results of the third experiment are shown in Figure 5.5. for the increasing number of variants with various numbers of samples.

5.3. Discussion

The results of the boolean circuit and the integer arithmetic experiments are discussed in this section respectively. Furthermore, the methods are compared according to the results of experiments.

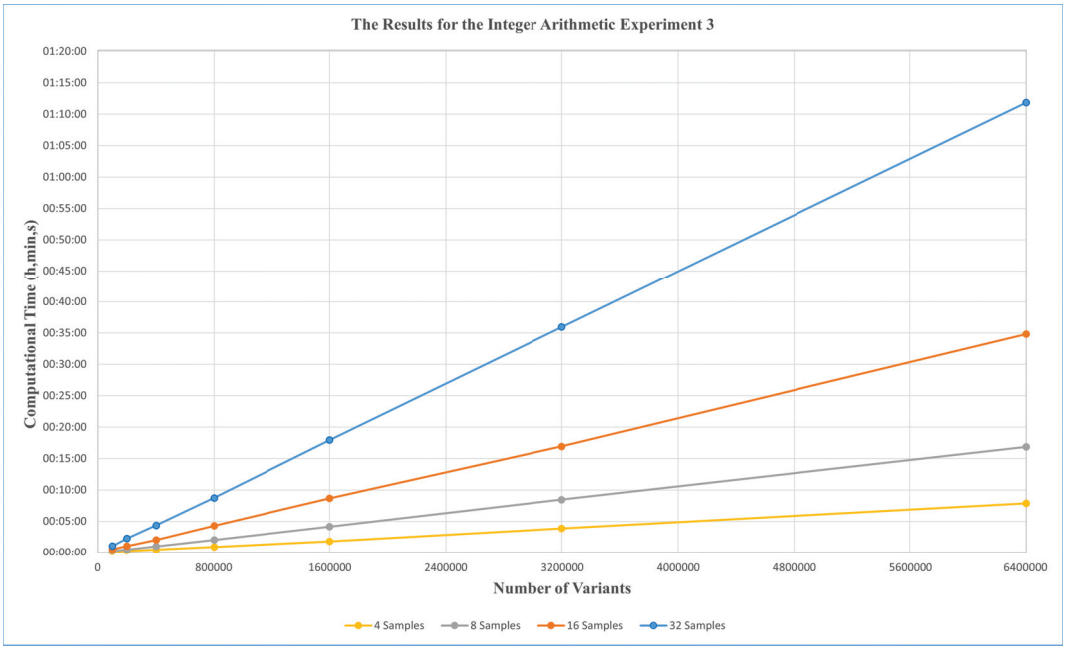

Figure 5.5. The Computation Time Results for Integer Arithmetic Experiment 3

5.3.1. Results for Boolean Circuit Experiments

Two different experiments are performed considering the recessive inheritance model to observe the efficiency of the boolean circuit method. As an outcome of these experiments, the following results are obtained.

As seen in Figure 5.1., using more threads provides better time efficiency for the variant filtering process. Furthermore, the computation time for the variant filtering process may also be improved by using more threads for the larger number of variants as seen in the results in Figure 5.2.

The use of the boolean circuit method does not provide highly efficient time results, especially considering the fact that genome data contains millions of variants. The implementation of the boolean circuit method for millions of variants should require huge resources. However, the computation time efficiency may be improved using multithreading with advanced hardware components. For instance, the Graphics Processing Unit (GPU) may be utilized to provide better computation time performance.

5.3.2. Results for Integer Arithmetic Experiments

Three different experiments are performed considering the recessive, dominant, and de novo inheritance models to observe the performance of the proposed integer arithmetic methods. As a consequence of these experiments, the following results are obtained.

As seen in Figure 5.3., the ciphertext size is a significant parameter for the time efficiency of the variant filtering process on the encrypted genome data. Better results are obtained as the ciphertext size decreases. Besides, the number of multiplications performed is also found to be an important factor for the computation time results. According to the given results in Figure 5.4., the reduction in the number of multiplication operations significantly improves the computation time efficiency. Furthermore, the variant filtering process is completed in less time for the de novo inheritance model compared to the recessive and dominant inheritance models due to the required number of arithmetic operations to be performed. The number of operations for the de novo inheritance model is less than the recessive and dominant inheritance models because there is no query for comparison while specifying the disease-causing variants. In addition, the duration of the sum operation is very short compared to the variant filtering process in all integer arithmetic experiments.

In conclusion, using the integer arithmetic method for rare disease analysis provides acceptable computation time results while filtering millions of variants. However, the computation time should be accelerated using methods like multithreading to realize rare disease analysis in real-life applications. Besides, the computation time results may be improved using advanced hardware components such as GPU.

5.3.3. Comparison of Boolean Circuit and Integer Arithmetic Methods

According to the results of the boolean circuit and the integer arithmetic experiments, it can be stated that the integer arithmetic method provides more efficient computation time results compared to the boolean circuit method. The boolean circuit method does not provide efficient enough computation time results considering the genome data contains millions of variants. On the contrary, the integer arithmetic method enables a larger number of variants to be processed for privacy-preserving rare disease analysis in less time. We can claim that the integer arithmetic method provides an applicable privacy-preserving solution for real-life applications of the rare disease analysis problem.

CHAPTER 6

CONCLUSION

Rare diseases have a profound impact on the lives of many individuals worldwide, presenting numerous challenges. While researchers conduct extensive studies on rare diseases, accessing patients' genome data poses privacy concerns. Therefore, it is crucial to develop a solution that ensures patient privacy while enabling researchers to analyze genome data collaboratively.

In this thesis, we propose a privacy-preserving solution for rare disease analysis. By utilizing the fully homomorphic encryption method, which allows operations on encrypted data, we protect the sensitive genome data of rare disease patients. Our aim is to facilitate researchers' access to this data while safeguarding the privacy of patients. Our privacy-preserving rare disease analysis employs two methods to detect disease-causing variants: Firstly, we utilize boolean circuits to identify the disease-causing variants within the encrypted genome data of patients. Secondly, we employ integer arithmetic operations to conduct rare disease analysis on the encrypted genome data. We conduct various experiments considering different inheritance models to assess the efficiency of these methods. We observe that increasing the number of threads in the boolean circuit experiments improves efficiency. Moreover, in the integer arithmetic experiments, the size of processed ciphertexts significantly influences the variant filtering process. Decreasing the ciphertext size results in more efficient computation time. The number of multiplication operations also plays a crucial role in computation time. Minimizing the number of these operations ensures efficient privacy-preserving rare disease analysis. Based on the experiment results, the integer arithmetic method shows promise for performing privacy-preserving rare disease analysis. Further improvements can be made by exploring alternative techniques such as multithreading.

By prioritizing patient privacy and fostering collaboration among medical institutions, our proposed solution aims to advance rare disease analysis in a manner that benefits both patients and researchers.

6.1. Future Work

As future work, there are several opportunities to enhance the performance of the integer arithmetic method by exploring alternative techniques. One potential direction is to investigate the impact of employing different numbers of threads for the variant filtering process. Introducing multithreading capabilities holds the potential for improving the performance of the integer arithmetic method significantly.

Furthermore, the privacy-preserving rare disease analysis could benefit from leveraging computers equipped with more advanced hardware components. By harnessing the power of such systems, computation time results can be further improved. This advancement would enable the efficient processing of patients' genome data in practical settings, eliminating privacy concerns. Consequently, researchers would be able to conduct thorough analyses of rare diseases by identifying disease-causing variants with precision.

These future endeavors aim to optimize the performance of the integer arithmetic method and create a framework that supports efficient and privacy-preserving rare disease analysis. By leveraging advanced techniques and hardware, we can facilitate effective research in this domain while ensuring patient privacy.

Bibliography

(1) Wang, Y.; Kung, L.; Byrd, T. A. *Technological Forecasting and Social Change* **2018**, *126*, 3–13.

(2) Chen, M.; Hao, Y.; Hwang, K.; Wang, L.; Wang, L. *IEEE Access* **2017**, *5*, 8869–8879.

(3) Begenau, J.; Farboodi, M.; Veldkamp, L. *Journal of Monetary Economics* **2018**, *97*, 71–87.

(4) Omar, M. A.; Inaba, K. *Journal of economic structures* **2020**, *9*, 37.

(5) Bakhshinategh, B.; Zaiane, O. R.; ElAtia, S.; Ipperciel, D. *Education and Information Technologies* **2018**, *23*, 537–553.

(6) Khan, A.; Ghosh, S. K. *Education and information technologies* **2021**, *26*, 205–240.

(7) Hathaliya, J. J.; Tanwar, S. *Computer Communications* **2020**, *153*, 311–335.

(8) van Zoonen, L. *Government Information Quarterly* **2016**, *33*, Open and Smart Governments: Strategies, Tools, and Experiences, 472–480.

(9) Conti, M.; Sandeep Kumar, E.; Lal, C.; Ruj, S. *IEEE Communications Surveys Tutorials* **2018**, *20*, 3416–3452.

(10) Kunwar, V.; Chandel, K.; Sabitha, A. S.; Bansal, A. In *2016 6th International Conference - Cloud System and Big Data Engineering (Confluence)*, 2016, pp 300–305.

(11) Kwekha-Rashid, A. S.; Abduljabbar, H. N.; Alhayani, B. *Applied Nanoscience* **2023**, *13*, 2013–2025.

(12) Zhao, Q.-F.; Tan, L.; Wang, H.-F.; Jiang, T.; Tan, M.-S.; Tan, L.; Xu, W.; Li, J.-Q.; Wang, J.; Lai, T.-J.; Yu, J.-T. *Journal of Affective Disorders* **2016**, *190*, 264–271.

(13) Newaz, A. I.; Sikder, A. K.; Rahman, M. A.; Uluagac, A. S. *ACM Trans. Comput. Healthcare* **2021**, *2*, DOI: 10.1145/3453176.

(14) Carss, K. J.; Arno, G.; Erwood, M.; Stephens, J.; Sanchis-Juan, A.; Hull, S.; Megy, K.; Grozeva, D.; Dewhurst, E.; Malka, S., et al. *The American Journal of Human Genetics* **2017**, *100*, 75–90.

(15) Graham, E.; Lee, J.; Price, M.; Tarailo-Graovac, M.; Matthews, A.; Engelke, U.; Tang, J.; Kluijtmans, L. A.; Wevers, R. A.; Wasserman, W. W., et al. *Journal of Inherited Metabolic Disease* **2018**, *41*, 435–445.

(16) Institute, N. C. Genome Sequencing, https://www.cancer.gov/publications/dictionaries/cancer-terms/def/genomic-sequencing, Last accessed on 2023-07-26, 2023.

(17) Frizzo-Barker, J.; Chow-White, P. A.; Charters, A.; Ha, D. *Computer Supported Cooperative Work (CSCW)* **2016**, *25*, 115–136.

(18) Shringarpure, S. S.; Bustamante, C. D. *The American Journal of Human Genetics* **2015**, *97*, 631–646.

(19) Research, E. U.; innovation European Commision EU research on rare diseases, https://research-and-innovation.ec.europa.eu/research-area/health/rare-diseases_en, Last accessed on 2023-06-04, 2023.

(20) For Rare Disorders, N. O. List of rare diseases: A-Z database: NORD, https://rarediseases.org/rare-diseases/, Last accessed on 2023-06-03, 2023.

(21) Lung, N. H.; Institute, B. Cystic Fibrosis - Symptoms, https://www.nhlbi.nih.gov/health/cystic-fibrosis/symptoms, Last accessed on 2023-06-09, 2023.

(22) Union, E. General Data Protection Regulation (GDPR), https://gdpr.eu/tag/gdpr/, Last accessed on 2023-06-04, 2023.

(23) Lauter, K.; López-Alt, A.; Naehrig, M. In *Progress in Cryptology - LATINCRYPT 2014*, ed. by Aranha, D. F.; Menezes, A., Springer International Publishing: Cham, 2015, pp 3–27.

(24) Wang, S.; Zhang, Y.; Dai, W.; Lauter, K.; Kim, M.; Tang, Y.; Xiong, H.; Jiang, X. *Bioinformatics* **2015**, *32*, 211–218.

(25) Zhang, Y.; Dai, W.; Jiang, X.; Xiong, H.; Wang, S. In *BMC medical informatics and decision making*, 2015; Vol. 15, pp 1–11.

(26) Zhang, Y.; Blanton, M.; Almashaqbeh, G. In *BMC medical informatics and decision making*, 2015; Vol. 15, pp 1–12.

(27) Chen, F. et al. *Bioinformatics* **2016**, *33*, 871–878.

(28) Spielman, R. S.; McGinnis, R. E.; Ewens, W. J. *American journal of human genetics* **1993**, *52*, 506.

(29) Wang, M.; Ji, Z.; Wang, S.; Kim, J.; Yang, H.; Jiang, X.; Ohno-Machado, L. *Bioinformatics* **2017**, *33*, 3716–3725.

(30) Jagadeesh, K. A.; Wu, D. J.; Birgmeier, J. A.; Boneh, D.; Bejerano, G. *Science* **2017**, *357*, 692–695.

(31) Yao, A. C.-C. In *27th Annual Symposium on Foundations of Computer Science (sfcs 1986)*, 1986, pp 162–167.

(32) Akgün, M.; Ünal, A. B.; Ergüner, B.; Pfeifer, N.; Kohlbacher, O. *Bioinformatics* **2020**, *36*, 5205–5213.

(33) Danecek, P.; Auton, A.; Abecasis, G.; Albers, C. A.; Banks, E.; DePristo, M. A.; Handsaker, R. E.; Lunter, G.; Marth, G. T.; Sherry, S. T.; McVean, G.; Durbin, R.; Group, 1. G. P. A. *Bioinformatics* **2011**, *27*, 2156–2158.

(34) VCFtools, N. The variant call format specification VCFv4. 3 and BCFv2. 2, 2021.

(35) Rivest, R. L.; Shamir, A.; Adleman, L. *Commun. ACM* **1978**, *21*, 120–126.

(36) Acar, A.; Aksu, H.; Uluagac, A. S.; Conti, M. *ACM Comput. Surv.* **2018**, *51*, DOI: 10.1145/3214303.

(37) Paillier, P. In *Advances in Cryptology — EUROCRYPT '99*, ed. by Stern, J., Springer Berlin Heidelberg: Berlin, Heidelberg, 1999, pp 223–238.

(38) Yao, A. C. In *23rd Annual Symposium on Foundations of Computer Science (sfcs 1982)*, 1982, pp 160–164.

(39) Boneh, D.; Goh, E.-J.; Nissim, K. In *TCC*, 2005; Vol. 3378, pp 325–341.

(40) Gentry, C., *A fully homomorphic encryption scheme*; Stanford university: 2009.

(41) Marcolla, C.; Sucasas, V.; Manzano, M.; Bassoli, R.; Fitzek, F. H. P.; Aaraj, N. *Proceedings of the IEEE* **2022**, *110*, 1572–1609.

(42) Munjal, K.; Bhatia, R. *Complex & Intelligent Systems* **2022**, 1–28.

(43) Van Dijk, M.; Gentry, C.; Halevi, S.; Vaikuntanathan, V. In *Advances in Cryptology – EUROCRYPT 2010*, ed. by Gilbert, H., Springer Berlin Heidelberg: Berlin, Heidelberg, 2010, pp 24–43.

(44) Coron, J.-S.; Naccache, D.; Tibouchi, M. In *Advances in Cryptology – EUROCRYPT 2012*, ed. by Pointcheval, D.; Johansson, T., Springer Berlin Heidelberg: Berlin, Heidelberg, 2012, pp 446–464.

(45) Brakerski, Z.; Vaikuntanathan, V. *SIAM Journal on Computing* **2014**, *43*, 831–871.

(46) Brakerski, Z.; Vaikuntanathan, V. In *Advances in Cryptology – CRYPTO 2011*, ed. by Rogaway, P., Springer Berlin Heidelberg: Berlin, Heidelberg, 2011, pp 505–524.

(47) Brakerski, Z.; Gentry, C.; Vaikuntanathan, V. *ACM Trans. Comput. Theory* **2014**, *6*, DOI: 10.1145/2633600.

(48) Fan, J.; Vercauteren, F. Somewhat Practical Fully Homomorphic Encryption, Cryptology ePrint Archive, Paper 2012/144, https://eprint.iacr.org/2012/144, 2012.

(49) Bajard, J.-C.; Eynard, J.; Hasan, M. A.; Zucca, V. In *Selected Areas in Cryptography – SAC 2016*, ed. by Avanzi, R.; Heys, H., Springer International Publishing: Cham, 2017, pp 423–442.

(50) Cheon, J. H.; Kim, A.; Kim, M.; Song, Y. In *Advances in Cryptology – ASIACRYPT 2017*, ed. by Takagi, T.; Peyrin, T., Springer International Publishing: Cham, 2017, pp 409–437.

(51) Chillotti, I.; Gama, N.; Georgieva, M.; Izabachène, M. *Journal of Cryptology* **2020**, *33*, 34–91.

(52) Hoffstein, J.; Pipher, J.; Silverman, J. H. In *Algorithmic Number Theory*, ed. by Buhler, J. P., Springer Berlin Heidelberg: Berlin, Heidelberg, 1998, pp 267–288.

(53) López-Alt, A.; Tromer, E.; Vaikuntanathan, V. In *Proceedings of the Forty-Fourth Annual ACM Symposium on Theory of Computing*, Association for Computing Machinery: New York, New York, USA, 2012, pp 1219–1234.

(54) Chillotti, I.; Joye, M.; Ligier, D.; Orfila, J.-B.; Tap, S. In *WAHC 2020 - 8th Workshop on Encrypted Computing & Applied Homomorphic Cryptography*, [Virtual], France, 2020.

(55) Chillotti, I.; Gama, N.; Georgieva, M.; Izabachène, M. TFHE: Fast Fully Homomorphic Encryption Library, https://tfhe.github.io/tfhe/, August 2016.

(56) Badawi, A. A. et al. OpenFHE: Open-Source Fully Homomorphic Encryption Library, Cryptology ePrint Archive, Paper 2022/915, `https://eprint.iacr.org/2022/915`, 2022.