SECURE COUNTING QUERY PROTOCOL FOR GENOMIC DATA

By
Vaishnavi Lalu Nathe
(TE-IT Roll No:43 Year:2023-24)

28 October 2023

Guided By
Prof Y.S.Algat

Department of Information Technology K.K.Wagh Institute of Engineering Education and Research Nashik - 422003

K.K.Wagh Institute of Engineering Education and Research,Nashik-422003

CERTIFICATE

This is to certify that**Vaishnavi Lalu Nathe** has Successfully Delivered Seminar on topic **SECURE COUNTING QUERY PROTOCOL FOR GENOMIC DATA** under the guidence of **Prof Y.S.Algat** towords the partial fulfillment of Bachelor's Degree in Information Technology of Savitribai Phule Pune University, during Academic Year 2023-2024

Date:
Place:Nashik

Prof Y.S.Algat
Guided By

Acknowledgement

I take this opportunity to express my deepest sense of gratitude ans sincere thanks to everyone who helped me to complete this work successfully.I express my sincere thanks to DR P.D.Bhamre Head of department information Technology ,K.K.Wagh Institute of engineering Education and Research,Nashik for providing me with all the necessary support. I would like to Express my Sincere gratitude to my seminar guide Prof.Y.S.Algat for their support,co-operation,Guidance and mentor-ship throughout the course.

Finally I Thank to my friends who contribute to successful fulfilment of this seminar work.

Abstract

The Secure Counting Query Protocol for Genomic Data is a research paper that proposes a secure way to share and analyze genomic data on cloud platforms. The protocol allows for statistical analysis of genomic data while ensuring the privacy and security of the data. The paper describes the methodology used to develop the protocol and presents experimental results that demonstrate its feasibility and efficiency. The protocol has potential applications in the field of genomics research and can help advance ourunderstanding of the relationship between gene sequence and phenotype.

1. Firstmake the genetic data smaller so it's easier to work with.

2. Then use a special technique called "Paillier homomorphic" to share the data on the cloud securely. It's like sending a locked box that can still be used for certain types of calculations.

3. Once the cloud does these calculations, the results are sent back, and the researchers can unlock and understand them.

The calculations took less than 1 second, which is really fast. So, this methodcould be a good way to share genetic data securely on cloud platforms for studying the relationships between genes and traits.

Contents

Chapter 1

Introduction

1.1 Contribution

designed a secure multi-party computation protocol for sharing genomic data on cloud platforms, which allows for performing sample sequence counting queries on encrypted genomic data while ensuring data privacy, query privacy, output privacy, and access privacy. also describe their method for preprocessing genomic data in VCF files and designing a counting function using additive map of homomorphic encryption and modular operation.

1.2 Related Work

As genome sequencing technology advances, the availability of public genomic data on the internet has led to privacy concerns. Active attacks on genomic data privacy fall into two categories: statistical analysis and biological information reasoning attacks.
Statistical attacks include methods like SNP analysis and genome-wide association analysis to identify

individuals in public data. Inference attacks use shared genomic data to trace identities and gene phenotypes. These attacks can compromise genetic privacy.

To address these issues, privacy protection methods aim to maintain data availability without distortion. Secure multi-party computation, based on homomorphic encryption or garbled circuits, is commonly used to protect genomic data in a cloud platform environment.

Several techniques have been proposed, such as homomorphic encryption for secure querying of SNP data, privacy protection schemes based on hash functions, and systems for secure storage and processing of genomic data using encryption and integer comparison.

Other methods include hiding data size and position, information-theoretic confidentiality guarantees, fully homomorphic algorithms, public key encryption, secure outsourcing frameworks, and privacy protection models for genomic databases. These methods enable secure genome research and queries while safeguarding privacy.

In summary, various techniques and protocols have been developed to protect genomic data privacy in the face of evolving threats, enabling secure and confidential research and data sharing.

Chapter 2

Preliminaries

2.1 VCF File

A VCF file records all mutations in the gene position of the sample, including SNP, INDEL (Insertion and Deletion) and SV (Structural Variation). The VCF file is divided into two parts. One is the comment part that starts with “” and the other is the main part that does not start with “”. The main part contains ten columns: CHROM, POS, ID, REF, ALT, QUAL, FILTER, INFO, FORMAT, SAMPLES. SAMPLES represents samples, each sample corresponds to one column. Multiple samples correspond to multiple columns andthe main part of multiple samples has more than 10 columns. The columns of POS, ID, REF, ALT, and SAMPLES is used for the queryable counting. The column of FORMAT is involved in encryption and counting.

conclusions are drawn based on the results of GT. “0/0” indicates that the site in the sample is homozygous, which is consistent with the base type of REF.

"0/1" indicates that the site in the sample is a heterozygous mutation. There are two genotypes of REF and ALT. Some bases are the same type as REF bases, and some bases are the same type as ALT bases. "1/1" indicates that the site in the sample is a homozygous mutation, and the overall mutation type is consistent with the ALT base type.

"1/2" indicates that the position in the sample is a heterozygous mutation, and there are two genotypes of ALT1 and ALT2, which are partially the same as the ALT1 base type and partially the same as the ALT2 base type.

2.2 Pallier Homomorphic Encryption

In pallier Homomorphic Encryption before sharing data on platform user Firstly generates keys and transmitted between holders as :

Randomly select two large prime numbers p, q, so that they are independent of each other.

$gcd(p-1)(q-1) = 1$

calculate(n=pq,lambda=lcm(p-1)(q-1)

an integer g, $g \in Z_{n^2}^*$. $\mu = (L(g^\lambda mod\, n^2))^{-1}$, L is defined as $L(x) = \frac{x-1}{n}$. Public key is (n, g), private key is (λ, μ). If p, q are of equal length, the above key generation steps can be simplified as follows: $g = n + 1$, $\lambda = \varphi(n)$, $\mu = \varphi(n) mod\, n$, $\varphi(n) = (p-1)(q-1)$.

Encryption: In order to encrypt the plaintext m that satisfies $0 \leq m < n$, we randomly choose an integer r that satisfies $0 < r < n$, $r \in Z_{n^2}^*$, $gcd(r, n) = 1$. The ciphertext can be calculated as

$$c = E(m, r) = g^m \cdot r^n mod\, n^2 \tag{1}$$

Decryption: The ciphertext to be decrypted is c th at satisfies $c \in Z_{n^2}^*$. The plaintext can be calculated as

$$m = D(c) = L(c^\lambda mod\, n^2) \cdot \mu mod\, n \tag{2}$$

Homomorphic calculation: When performing homomorphic addition calculation, the ciphertext product is equal to the corresponding plaintext sum.

$$\begin{aligned} &D(E(m_1, r_1)) \cdot E(m_2, r_2) mod\, n^2 \\ &\quad = D(g^{m_1} \cdot r_1^n \cdot g^{m_2} \cdot r_2^n) mod\, n^2 \\ &\quad = D(g^{m_1+m_2} \cdot (r_1 r_2)^n) mod\, n^2 = m_1 + m_2 \end{aligned} \tag{3}$$

Homomorphic number multiplication, the ciphertext k power equals the plaintext number multiplied by k:

$$\begin{aligned} D(E(m_1, r_1)^k mod\, n^2) &= D((g^{m_1} \cdot r_1^n)^k) mod\, n^2 \\ &= D(g^{km_1} \cdot r_1^{kn}) mod\, n^2 \\ &= k m_1 mod\, n \end{aligned} \tag{4}$$

Figure 2.1: Enter Caption

Chapter 3

Secure Counting Query Protocol

The protocol enables secure counting queries on genomic data stored in a VCF file via a cloud platform, involving Genomic Data Owners (DO), Query Users (QU), and the cloud platform. It ensures data privacy and allows multiple parties to interact securely for statistical analysis and querying on the cloud platform.

3.1 Protocol Framework

The protocol framework consists of three participants: the data owner (DO), the cloud platform (CP), and the query party (QU). The DO owns the genomic data and

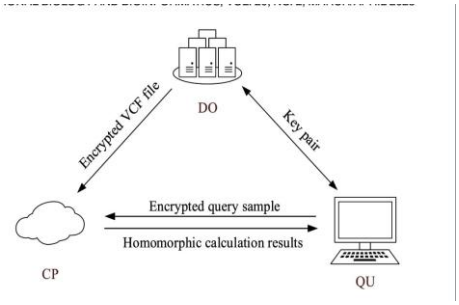

Figure 3.1: Model of secure Multi-party computation protocol

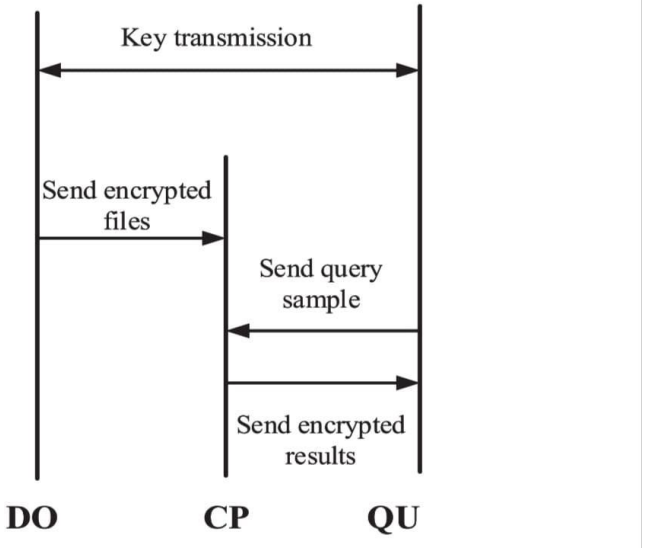

Figure 3.2: Sequence Diagram of Secure multi-party computational protocol

wants to share it securely on the CP, which is a cloud platform with data computing and sharing capabilitiesbut cannot protect the privacy of the uploaded data.The QU wants to use the data stored on CP to performcounting queries on gene sequence information without privacy leakage. The protocol is designed to ensure the privacy and security of the genomic data, and it is a tripartite protocol based on the different roles played bythe user and the cloud platform. The authors assumethat both CP and QU are semi-honest participation models, and they provide a model of secure multi-party computation protocol to ensure the privacy and security of the genomic data. Overall, the protocol framework provides a clear understanding of the roles and objectives of each participant in the protocol.

the Counting function, it is a function designed to

obtain the number of Q samples in the original genomic data. After performing the HomoCrypt operation, a matrix C is obtained, which is an m x n matrix. This matrix C can perform the counting operation, and if QU wants to retrieve the information of the same sequence as the query sequence Q, it only needs to initiate a query to DO according to the position of the sequence in the matrix. Overall, the Counting function is an essential part of the protocol, allowing for the secure and accurate counting of genomic data.

Chapter 4

VCF File Processing

Regarding VCF file processing, the authors explain that a VCF file records all mutations in the gene position of the sample, including SNP, INDEL, and SV. The VCF file is divided into two parts: the comment part and the main part. The main part contains ten columns, including CHROM, POS, ID, REF, ALT, QUAL, FILTER, INFO, FORMAT, and SAMPLES. The
proposed protocol does not encrypt the parameter information of CHROM, POS, ID, REF, ALT, and QUAL at the site, and they can be used as a searchable directory in plaintext format.
Dij 2 D; 0 i m; 0 j n
There are two encoding modes em that can be selected, and each encoding mode corresponds to its own encoding variable ev 2 f00; 11g and counting variable cv
2 f0; 1g.

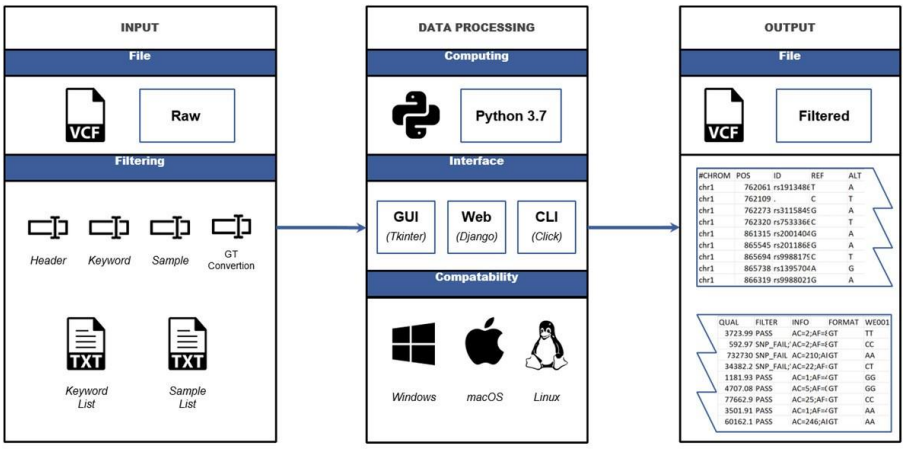

Figure 4.1: VCF Processing

Chapter 5

Homomorphic Encryption

homomorphic encryption, it is an encryption technique that allows computations to be performed on ciphertext, without the need to decrypt it first. This means that the encrypted data can be processed without revealing the original data, providing a high level of privacy and security. Homomorphic encryption algorithms can satisfy addition and multiplication homomorphism, which means that after performing these operations on the ciphertext, the value obtained by decrypting the result is equal to the value calculated by the same calculation for the plaintext. The Paillier algorithm is an example of a homomorphic encryption algorithm that has good computing performance for

homomorphic addition and number multiplication.

1)DO generates public and private key for pallier Encryption,then data homomorphically encrypted.

2)Key Transmission protocol between QU and DO .

3)Queries from QU are encoded and then homomorphically encrypted to obtain the encrypted query sequence

CP has the matrix $\mathcal{M}$ and the encrypted query sequence $\tilde{\mathcal{Q}}$, then homomorphic calculation can be performed on $\mathcal{M}$ and $\tilde{\mathcal{Q}}$ to obtain ciphertext the result matrix $\mathcal{R}$.

$$\mathcal{R} = \mathcal{M} + \tilde{\mathcal{Q}} = \begin{bmatrix} \mathcal{R}_{00} & \mathcal{R}_{01} & \cdots & \mathcal{R}_{0(n-1)} \\ \mathcal{R}_{10} & \mathcal{R}_{11} & \cdots & \mathcal{R}_{1(n-1)} \\ \vdots & \vdots & \ddots & \vdots \\ \mathcal{R}_{(m-1)0} & \mathcal{R}_{(m-1)1} & \cdots & \mathcal{R}_{(m-1)(n-1)} \end{bmatrix} = \begin{bmatrix} \mathcal{M}_{00} + \tilde{\mathcal{Q}}_0 & \cdots & \mathcal{M}_{0(n-1)} + \tilde{\mathcal{Q}}_0 \\ \mathcal{M}_{10} + \tilde{\mathcal{Q}}_1 & \cdots & \mathcal{M}_{1(n-1)} + \tilde{\mathcal{Q}}_1 \\ \vdots & \ddots & \vdots \\ \mathcal{M}_{(m-1)0} + \tilde{\mathcal{Q}}_{(m-1)} & \cdots & \mathcal{M}_{(m-1)(n-1)} + \tilde{\mathcal{Q}}_{(m-1)} \end{bmatrix} \quad (12)$$

$$\mathcal{C} = Dec(\mathcal{R})_{K_s} == \begin{bmatrix} \mathcal{C}_{00} & \mathcal{C}_{01} & \cdots & \mathcal{C}_{0(n-1)} \\ \mathcal{C}_{10} & \mathcal{C}_{11} & \cdots & \mathcal{C}_{1(n-1)} \\ \vdots & \vdots & \ddots & \vdots \\ \mathcal{C}_{(m-1)0} & \mathcal{C}_{(m-1)1} & \cdots & \mathcal{C}_{(m-1)(n-1)} \end{bmatrix} = \begin{bmatrix} Dec(\mathcal{R}_{00})_{K_s} & \cdots & Dec(\mathcal{R}_{0(n-1)})_{K_s} \\ Dec(\mathcal{R}_{10})_{K_s} & \cdots & Dec(\mathcal{R}_{1(n-1)})_{K_s} \\ \vdots & \ddots & \vdots \\ Dec(\mathcal{R}_{(m-1)0})_{K_s} & \cdots & Dec(\mathcal{R}_{(m-1)(n-1)})_{K_s} \end{bmatrix} \quad (13)$$

4)CP performs homomorphic calculations on data and generate the ciphertext result matrix .

5)CP sends Result matrix to QU, which decrypts it using private key to obtain the plaintext matrix.

6)Plaintext matrix can then be used for the counting operation.

Chapter 6

Counting Function

The algorithm's primary objective is to count the number of submatrices within a given matrix (C) that meet a specified binary condition(cv).These submatrices are consecutive rows and columns within the larger matrix. The algorithm systematically scans through the matrix to identify such submatrices.
It counts and returns the total number of such submatrices.This algorithm is useful for applications where you need to identify and count patterns in binary matrices or analyze data meeting specific criteria.

Algorithm 1. Counting Algorithm

```
Input: C, cv
Output: count
count = 0;
for j < m do
  num=0
  for i < n do
    equal1=C_ij mod 2
    equal2=(C_ij mod 10) mod 2
    if equal1! = cv or equal2! = cv then
      break;
    else
      num = num + 1;
    end
    if num == n then
      count=count+1;
    end
    i++
  end
  j++
end
```

Figure 6.1: Counting Algorithm

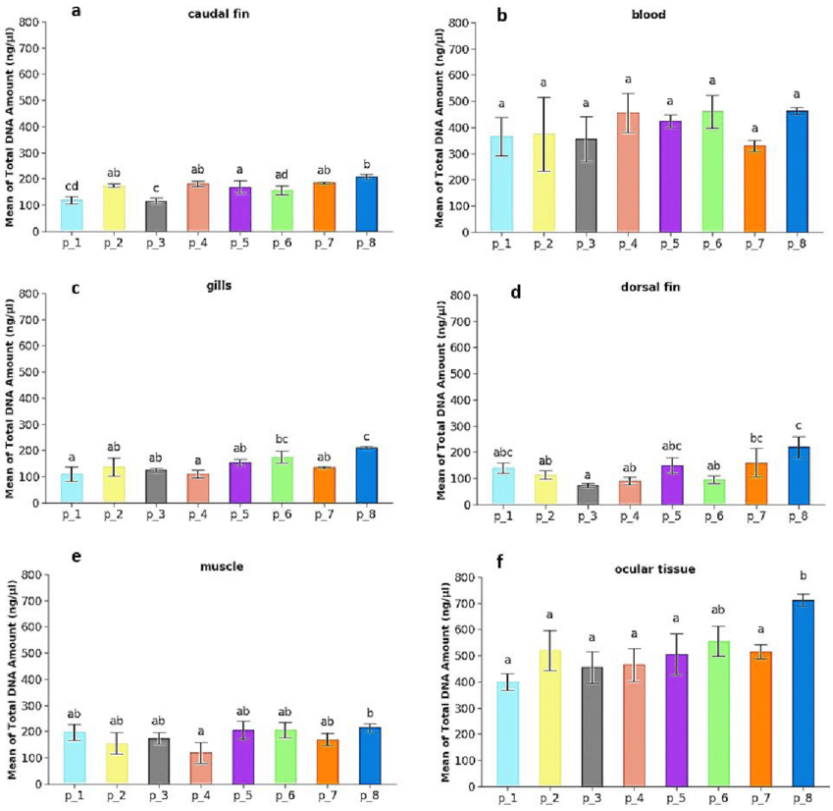

Figure 6.2: Enter Caption

Chapter 7

Scheme Analysis

7.1 security Analysis

Active Attack From Malicious Attacker:

security in active attack from malicious attacker system measures to ensure the privacy and integrity of the data. Firstly, a secure channel is used to transmit the Paillier key pair, ensuring that the attacker cannot obtain the Paillier key pair. Secondly, the protocol can resist intercepting and spoofing attacks from malicious attackers, as they cannot decrypt the content of the VCF file or the calculation result without the key. Finally, the protocol assumes two kinds of attacks: active attacks from malicious attackers and passive attacks from participants. To prevent active attacks, the protocol uses a trusted computing base, where CP, QU, and DO are trusted participants, and the malicious attacker is prevented from stealing genomic data information. Overall, the protocol provides a high level of security against active attacks from malicious attackers.

Passive Attack From Curious CP or QU Participants:

There are three participants CP,QU and DO all of Whome are curious data involved in a protocol, but they do not collaborate with each other for active attacks. Instead, they focus on passive attacks, where they gather information without actively disrupting the protocol.

The participants are restricted from exchanging information among themselves. CP can access encrypted genomic data in a VCF file and encrypted sample sequences but lacks the necessary Paillier encryption keys (Kp, Ks) to decrypt this data. This safeguards the genomic information of DO and QU from being accessed by CP. QU, on the other hand, can access the Paillier public key (Kp) and private key (Ks) but is unable to access the encoding method and variables used. Consequently, QU cannot access the encrypted genomic data in the VCF file and can only obtain the results of calculations performed by CP. This separation of encrypted genomic data and encryption keys ensures that the same participant does not possess both, enhancing the security of DO's data.

Furthermore, the query sequences sent to CP are homomorphically encrypted, and CP lacks the private key required for decryption. Only QU has the capability to

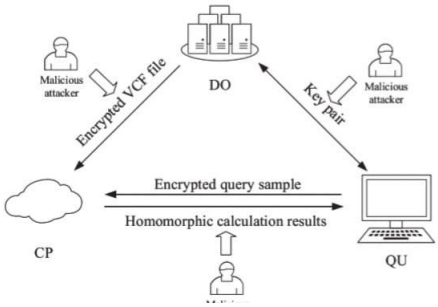

Figure 7.1: Active attacks by malicious attacker.

access the result matrix (C), preventing DO from deducing QU's query sequences through result analysis. As CP performs the traversal computation, neither CP nor DO can ascertain the access conditions of QU, preserving the access privacy of QU. In summary, this protocol design effectively safeguards the security and privacy of the data owned by DO, CP, and QU, while preventing unauthorized access and information exchange among the participants during passive attacks.

7.1 Efficiency Analysis

e time consumption in our protocol mainly comes from two parts, including homomorphic encryption and decryption and counting query. We use Paillier homomorphic encryption, so the computational complexity is the same as the Paillier algorithm. It can be seen from Equation 2 that the decryption process includes an exponential power mod n2, and the result mod n is multiplied by a fixed value Lcmod n2, and the computational complexity of decryption is exponential power mod n2. It can be seen from Algorithm 1 that

two loops are performed during the counting query, which are the rows and columns in the result matrix R, which meansthat the time complexity of the algorithm is related to the number of samples, and the correlation between the increase of time and the increase of the number of samples is lineargrowth.

Chapter 8

Experimental Results

8.1 Implimentation

a protocol implimented using Python 3.7 on Ubuntu 16.04. used publicly available VCF data to rigorously test the protocol, with a bandwidth of 100 M/s. To ensure secure communication, we employed the RSA asymmetric encryption algorithm for transmitting the Paillier encryption keys. Our implementation comprises four key components: VCF file processing, Paillier homomorphic encryption, RSA-based secure key transmission, and query counting.

VCF files contain diverse data, making direct encryption unsuitable for subsequent homomorphic calculations. We utilized the PyVCF library in Python to process the VCF data. By iterating through the data, we extracted information about mutation sites and created a genomic data matrix, aligning with proposed scheme.

RSA based Sequre Key transmission:

Our RSA implementation involves four essential steps.RSA key generation follows the RSA algorithm's key generation rules, generating public and private keys using random numbers and storing them in

separate files. We used TCP transmission to establish a secure connection between DO and QU and securely transmit the RSA public key. RSA encryption and decryption are based on the principles of the RSA public key encryption Algorithm:Pallier Algorithm.

Paillier homomorphic encryption also comprises four main parts. Key generation involves creating public and private keys, following Paillier's key generation rules. These keys are stored for local use and potential transmission. Homomorphic encryption and decryption functions are implemented according to Paillier's rules.

Importantly, Paillier encryption allows element-wise addition and multiplication calculations to be performed on ciphertext, preserving data security.

Query Counting:

We implemented the query counting process as outlined in Section 4.4. To evaluate the efficiency of our protocol, we measured the time required for each part of the query function.

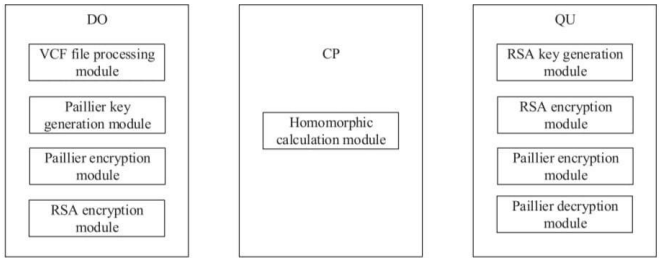

Figure 8.1: Implementation of our protocol.

TABLE 2
Time-Consuming of Different Specification Matrices

Functions\Data scale		10*10	50*50	50*100	50*200	50*400	50*600
Build $\mathcal{D}$	VCF reader	5.88s	5.98s	5.82s	5.87s	6.05s	6.14s
	Encryption	0.79s	19.63s	39.31s	78.83s	157.95s	237.68s
	Translation	0.01s	0.01s	0.01s	0.01s	0.01s	0.04s
	Total	6.68s	25.61s	45.14s	84.71s	165.80s	243.86s
Single query	Encryption	0.08s	0.39s	0.40s	0.39s	0.39s	0.39s
	Calculation	0.01s	0.09s	0.18s	0.36s	0.85s	1.11s
	Decryption	0.23s	5.68s	11.32s	22.74s	45.83s	68.01s
	Query	0.01s	0.01s	0.01s	0.01s	0.01s	0.01s
	Translation	0.01s	0.01s	0.01s	0.01s	0.02s	0.04s
	Total	0.25s	6.17s	11.91s	23.49s	47.09s	69.56s

Figure 8.2: Time-Consuming of Different Specification Matrices

8.2 Result Analysis

The protocol's performance is evaluated in terms of size, time, and query ability, and compared with other schemes in Table 3. The results show that the protocol has a competitive performance, with a small ciphertext size, low computation time, and high query ability. The protocol is also evaluated in terms of security, and it is shown to be resistant to active attacks from malicious attackers. Overall, the protocol provides a secure and efficient solution for genomic data analysis on cloud platforms, with potential applications in disease diagnosis, drug development, medical treatment, and medical research.

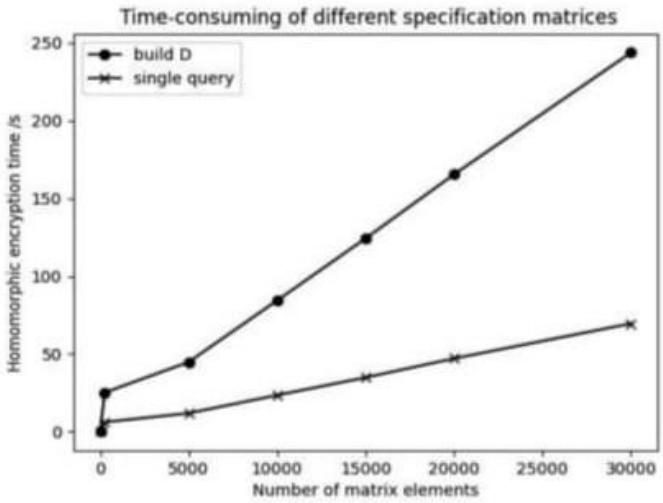

Fig. 5. Time-consuming of different specification matrices.

Chapter 9

Conclusion

The protocol is designed to be efficient, scalable, and secure, with a small ciphertext size, low computation time, and high query ability. The homomorphic encryption of genomic data only needs to be performed once, while the decryption and query parts need to be performed multiple times. The protocol is also resistantto active attacks from malicious attackers, ensuring the privacy and integrity of the data.

Overall, the protocol has potential applications in disease diagnosis, drug development, medical treatment, and medical research. It provides a secure and efficient solution for genomic data analysis on cloud platforms, enabling researchers to explore the relationship between gene sequence and phenotype.

Chapter 10

References

1)K. Ning and T. Chen, “Big data for biomedical research: Current status and prospective,” Chin. Sci. Bull., vol. 60, pp. 534–546, 2015.

2) . Naveed et al., “Privacy in the genomic era,”ACM Comput. Surveys, vol. 48, no. 1, pp. 6:1–6:44, 2015.

3) N. Homer et al., “Resolving individuals contributing trace amounts of DNA to highly complex mixtures using high-density SNP genotyping microarrays,” PLoS Genet. vol. 4, Aug. 2008, Art. no. e1000167

4) R. Wang, Y. F. Li, X. Wang, H. Tang, and X. Zhou, “Learning your identity and disease from research papers: Information leaks in genome wide association study,” in Proc. ACM Conf. Comput. Commun. Secur., 2009, pp. 1653662–1653726.

5) M. Kantarcioglu, W. Jiang, Y. Liu, B. Malin, “A Cryptographic approach to securely share and query genomic sequences,” IEEE Trans. Inf. Technol. Biomed., vol. 12, no. 5, pp. 606–617, Sep. 2008.