ISSN: 0067-2904

Preserving Genotype Privacy Using AES and Partially Homomorphic Encryption

Hiba M. Yousif*[1,2], Sarab M. Hameed[1]
[1]*Computer Science Department, College of Science, University of Baghdad, Baghdad, Iraq*
[2]*College of engineering, University of Information Technology and Communications*

Received: 9/2/2023 Accepted: 27/3/2023 Published: 30/3/2024

Abstract

Increasingly, the availability of personal genomic data in cloud servers hosted by hospitals and research centers has incentivized researchers to turn to research that deals with analyzing genomic data. This is due to its importance in detecting diseases caused by genetic mutations, detecting genes that carry genetic diseases, and attempting to treat them in future generations. Secure query execution on encrypted data is considered an active research area in which encryption is used to ensure the confidentiality of genomic data while restricting the ability to process such data without first decrypting it. To provide a secure framework and future insight into the potential contributions of homomorphic encryption to the field of genomic data, this paper proposes a framework for guaranteeing genomic data privacy using various partial homomorphic encryption techniques. By examining the characteristics of the three partial homomorphic encryptions based on different parameters. The framework has been online tested and compared based on different parameters. Three homomorphic encryption algorithms were adopted to ensure genomic data privacy by employing homomorphic operations in the query matching process. Experiments on real datasets, specifically MERS and SARSr-COV, showed that the proposed framework is efficient and improves query execution time by an average of 96% compared to existing work.

Keywords: ElGamal, Genomic data, Homomorphic encryption, Paillier, Privacy preserving, RSA

الحفاظ على خصوصية النمط الجيني باستخدام خوارزميه AES والتشفير المتماثل جزئيًا

هبة محمود يوسف* [1,2], سراب مجيد حميد[1]
[1]قسم علوم الحاسوب، كلية العلوم، جامعة بغداد، بغداد، العراق
[2]جامعة تكنولوجيا المعلومات و الاتصالات/ كلية الهندسة

الخلاصة

أدى التخزين المتزايد للبيانات الجينومية الشخصية في الخوادم السحابية التي تستضيفها المستشفيات ومراكز الأبحاث إلى تحول الباحثين إلى الأبحاث التي تتناول تحليل البيانات الجينية لأهميتها في اكتشاف الأمراض التي تسببها الطفرات الجينية والكشف عن الجينات التي تحمل الأمراض الوراثية ومحاولة علاجها في الأجيال القادمة. يعتبر تنفيذ الاستعلام الآمن على البيانات المشفرة مجال بحث نشط يستخدم فيه التشفير لضمان سرية البيانات الجينية مع تقييد القدرة على معالجة هذه البيانات دون فك تشفيرها أولاً. تقترح هذه الورقة إطارًا لضمان

*Email: heba.mahmoud1201a@sc.uobaghdad.edu.iq

خصوصية البيانات الجينومية باستخدام تقنيات تشفير جزئية مختلفة متجانسة الشكل لتوفير إطار عمل آمن ورؤية مستقبلية للمساهمات المحتملة للتشفير المتماثل في مجال البيانات الجينومية من خلال فحص خصائص التشفير الجزئي ثلاثي الشكل بناءً على معاملات مختلفة .تم اختبار إطار العمل عبر الإنترنت ومقارنته بناءً على معايير مختلفة. تم اعتماد ثلاث خوارزميات تشفير متجانسة الشكل في هذا العمل لضمان خصوصية البيانات الجينومية. توضح التجارب على مجموعات البيانات الحقيقية، وتحديداً MERS و SARSr−COV ، أن المقترح فعال ويحسن وقت تنفيذ الاستعلام بمعدل 96٪ مقارنة بالاعمال السابقة.

1. Introduction

Genomics is a discipline that studies the content, structure, and evolution of genomes. The primary goal is mass sequencing of nucleotide sequences. Tremendous technological evolution has been witnessed in the field and now it is no longer limited to the determination of sequences but also involves the analysis of both genes and proteins in terms of expression and function [1].

The genome is the carrier of biological genetic information and contains the important genetic information of human beings. The genome sequencing technology can analyze and calculate specific Deoxyribonucleic Acid (DNA) sequences in the genome, laying a foundation for further research and utilization. The Genome-Wide Association Study (GWAS) provides more possibilities for genomic data research and can help human beings know themselves better by exploring genes. However, GWAS is characterized by a huge data volume and complicated data processing. It is feasible to send genomic data to big data platforms for analysis and calculation. While big data provides support for bioinformatics research, it also faces unprecedented data security threats. The disclosure or improper use of genomic data will not only violate the personal privacy of data providers but also cause national and social problems. Therefore, the privacy protection of genomic data is an important link in GWAS [2].

Several bioinformatics processes are applied to digital genomic data. This raises the risk of exposing personal information. The main bioinformatics processes that can compromise privacy are sequence alignment, querying private genomic data, and searching a genomic database. Possible security issues when processing genomic data include insecure environments for sequence alignment, performing a query on private genomic data, searching a private genomic database, securely querying a public database, and performing secure queries on a private genomic database [3].

Moreover, since the utilization of cloud services has increased with the development of technology, the security and privacy of genomic data stored in the cloud should be ensured [4]. A breach of an individual genomic data may reveal their susceptibility to a specific disease, affecting their health insurance eligibility [5]. Therefore, privacy and security have become critical requirements [6].

This paper investigates how genomic data can be useful to researchers while maintaining data privacy so that subjects' identities are not revealed. The contributions of this paper are as follows:

- Propose a secure framework for ensuring the privacy of genomic data shared and stored on a cloud server.
- Introduce AES and three partial homomorphic encryption algorithms to protect genomic data from being discovered by unwanted parties including data privacy, query privacy, and output. . The query counting operation relies on the homomorphic operations of the Paillier, Rivest, Shamir, and Adleman (RSA), and ElGamal algorithms, which allows for the comparison of queried encrypted Single Nucleotides Polymorphism (SNP) values with stored encrypted SNP values without revealing their values.

This paper is organized as follows: Section 2 provides the related work for preserving genomic data privacy. Section 3 focuses on the fundamental concepts of genomic data and encryption methods. The proposed framework for secure counting queries is introduced in Section 4. Finally, Section 5 discusses the findings and future directions.

2. Related Works

The security and privacy of genomic data are fundamental issues, and many techniques are used to protect the privacy of this data. A common approach is to use cryptographic algorithms to protect genomic data and maintain its privacy.

Cetin et al. [5] developed a novel string-matching system that allows for privacy-preserving queries on homomorphically encrypted data. The protocol merged well-known techniques Permutation-based hashing and Permutation-based cuckoo hashing with private intersection protocols to minimize computational and communication costs.

Raisaro et al. [7] designed, implemented, and deployed a secure and efficient privacy-preserving solution for exploring genomic cohorts in a real operational scenario at the Lausanne University Hospital by employing differential privacy and Homomorphic Encryption (HE). While a combination of bootstrapping in Fully Homomorphic Encryption (FHE) with a scaling operation in fixed-point arithmetic was introduced by Chen et al. [8], through using of a minimax polynomial approximation to the sigmoid function and the 1-bit gradient descent approach to decrease plaintext growth in the training phase. They showed that training over encrypted data is possible, even if at a considerable computational expense. However, in critical applications, the approach can ensure the highest level of data privacy.
Hasan et al. [9] suggested methods to process biomedical data that contain genotype and phenotype by employing an index tree scheme, which significantly reduces the computational overhead cost, to securely execute the count query operation. They ensured the conventionality of the biomedical data by using encryption. A framework suggested by Chen et al. [10] to manage the security challenges of outsourced/transferred genomic data computations on a large scale by utilizing homomorphic encryption with the Garbled Circuit scheme, ensuring the privacy of genomic data. Additionally, Blatt et al. [11] suggested a set of statistical approaches that use HE to execute large-scale GWASs on encrypted genetic/phenotype data without having to decode the data. They implemented over a dozen crypto engineering modifications and rebuilt the GWAS tests to fully benefit from encrypted data packing and parallel computing coupled with very efficient statistical calculations. In contrast to claims that HE is unsuitable for large-scale GWASs, HE solutions are 30 times faster than the cutting-edge multiparty computing technique.

Mahdi et al. [12] suggested a protocol to handle the substring search query and set maximal matching problems. They made use of the suffix tree to create an index tree for the genomic data by using different algorithms AES and Garbled Circuit to ensure the privacy of data, query, and output. On the other hand, Yilmaz et al. [13] suggested a data-sharing system for genomic data that excludes some SNP states that are loosely linked with previously shared SNPs (and does not employ such states during data sharing). The suggested system determines a value to split among the non-eliminated states by establishing explicit privacy assurances. To make the shared data more useful, they demonstrated how to alter probability distributions for non-eliminated SNP states, and suggested an optimum and greedy approach for determining the processing order of SNPs in the proposed data-sharing algorithm to maximize the utility.

paper aims to develop a secure framework for storing and processing genomic data that ensures the confidentiality of search processes' data, queries, and results. The presented work

is similar to [9], however, the distinction is that it can work with encrypted datasets stored in the cloud and perform online encrypted operations on them while conducting searches. This is achieved by leveraging homomorphic subtraction and division operations to skip time-consuming extra phases. In addition to Paillier and AES algorithms used in [9], RSA and ElGamal algorithms were used to encrypt genomic data.

3. Preliminary Concepts

This section provides background information on genomic data, homomorphic encryption, and advanced encryption standard.

3.1 Genomic data

Genomics is a biology branch focusing on genomic structure, function, and activity. A genome is a human's full collection of DNA sequences. From a technological standpoint, one of the data sources for healthcare is genetic data; a person's genome shows their illness susceptibility and information about their family members [14]. A genome contains all information required for the function of single cells as well as very complex organisms. A genome is a collection of genes controlled in a range of cells whose division results in an organism. A genome is made up of noncoding sections, regulatory regions, and other components that work together to make life processes possible [15]. Approximately 99.9% of all people's DNA is identical, with the remaining 0.1 percent accounting for variations. The most prevalent source of variation in the human genome is Single Nucleotide Polymorphism (SNP). A single nucleotide alteration in the genome A, T, C, or G is known as an SNP, and there are around 50 million SNPs in the human genome [16].

3.2 Homomorphic encryption

Homomorphic Encryption (HE) is an encryption type that supports computation over encrypted data. A special type of HE that permit arbitrary computation on encrypted data was introduced in [7], which is called Fully Homomorphic Encryption (FHE). The output of these computations was also encrypted in [17]. It became a common method to secure data in the cloud [18].

HE is classified into several types based on its operational capabilities. FHE, Somewhat Homomorphic Encryption (SWHE), and Partially Homomorphic Encryption (PHE).
FHE is the most comprehensive HE system, allowing arbitrary functions to be evaluated on ciphertexts [19]. SWHE supports an unlimited number of operations, but each action generates noise, and after a certain number of operations, the underlying encrypted data is lost. As a result, SWHE systems can perform any operation for a limited amount of time. Every addition operation adds to the noise, and every multiplication action doubles the noise [20]. PHE is the oldest homomorphic encryption process, permitting only one operation on the ciphertexts [21].

This paper focuses on three PHE algorithms: Paillier, RSA, and ElGamal.

- **Paillier algorithm** is a probabilistic public-key cryptosystem. It is considered to be one of the most efficient encryption algorithms with an additive homomorphic property. It is widely used to preserve privacy and secure computation. Paillier algorithm consists of 3 stages: key generation, encryption, and decryption.

In a key generation, the algorithm produced a pair of keys: a public key (n, g), where n is the result of multiplying two large prime numbers p and $q \in \mathbb{Z}_{n^2}^*$. The secret key $(\lambda(n) = (p - 1) \times (q - 1))$.

In the encryption algorithm first, the alphanumeric symbols are converted into purely numeric ones, then a random $\mu \in \mathbb{Z}_n^*$ is chosen such that: $0 < \mu < n$. The ciphertext c is computed as in Equation 1 [22].

$$c = g^m \times \mu^n \ (mod\ n^2) \quad (1)$$

In the decryption algorithm, the message is computed according to Equation 2.

$$m = L(c^{\lambda(n)}(mod\ n^2)) \times (g^{\lambda(n)}(mod\ n^2)))^{-1}(mod\ n) \quad (2)$$

The subtraction operation is computed after obtaining the encrypted value of the first number (C_1) and the encrypted value of the second number (C_2), and applying Equation 3.

$$SUB_{result} = (C_1 \times C_2^{-1} n^2)\ mod\ n^2 \quad (3).\ [23]$$

- **Rivest, Shamir, and Adleman (RSA)** is the most widely used public-key cryptosystem since 1978. Multiplicative homomorphism is a property of the RSA scheme. The RSA homomorphic encryption is thus the product of two modulo n messages. In RSA semantic security, the integer factorization problem difficulty is employed. The security of data depends on the keys used, where strong keys lead to strong encryption and decryption [24] [25].

The key generation algorithm includes determining the public and private keys as follows:

- Choosing two prime numbers a, b, where $a \neq b$.
- Computing $n = a \times b$.
- Computing $\varphi(n) = (a-1) \times (b-1)$.
- Choosing an integer e, such that gcd $(\varphi(n), e)) = 1, 1 < e < \varphi(n))$.
- Computing $d = e^{-1}\ mod\ \varphi(n)$. The public and private keys are (n, e) and (n, d) respectively.
- After converting character symbols into integer form, the cipher text c can be obtained by applying Equation 4.

$$c = m^e\ mod\ n \quad (4)$$

In the decryption operation, the original message m can be retrieved using Equation 5 [26].

$$m = c^d\ mod\ n \quad (5)$$

Division operation can be calculated after computing the cipher value of the first number C_1, and the cipher value of the second number C_2, and applying Equation 6.

$$DIV_{result} = (C_1 \times C_2^{-1})^e\ mod\ n \quad (6)$$

- **ElGamal algorithm** [27] is an asymmetric key algorithm based on the Diffie-Hellman key exchange algorithm. It was created by Taher El Gamal in 1985, and is based on the difficulty of calculating the discrete logs of a large prime module. It includes key generation, encryption, and decryption.

First, in the key generation step: a large prime number p is chosen by randomly selecting g ∈ Z_q^* and x such that $(1 < x < p-1)$, storing it as a private key, then determining $h = g^x mod\ p$. The public key is (p, g, h). randomly chosen $r \in Z_q$, the ciphertext is computed using Equation 7.

$$a = g^r mod\ p,\ b = m.h^r mod\ p \quad (7)$$

The ciphertext of message m is the (a, b). To retrieve the message, Equation 8 is used [28].

$$u = a^x\ mod\ p$$
$$m = b.u^{-p}\ mod\ p \quad (8)$$

To perform division operation in El Gamal, two parts of each cipher (C_1, C_2) are computed $(a_1, b_1), (a_2, b_2)$. Then, Equation 9 is applied.

$$DIV_a = (a_1 \times a_2^{-1}) mod\ p, \qquad DIV_b = (b_1 \times b_2^{-1}) mod\ p \tag{9}$$

3.3 Advanced Encryption Standard

AES is a symmetric block cipher invented in 1998 by two Belgian cryptographers (Joan Daem and Vincent Rijmen). It was released in 2001 as FIPS Publication 1997 (Federal Information Processing Standards). AES encryption key lengths are 128, 192, and 256 bits. A typical data block size is 128 bits and uses the number of cipher rounds that can be configured for each block to be encrypted. In general, the 128-plaintext block size corresponds to one of three key lengths: 128, 192, or 256 bits in 10 rounds, 12 rounds, and 14 rounds, respectively [29].

4. The Proposed Secure Framework

The proposed security framework aims to provide secure storage of DNA sequences when outsourcing, and secure querying of tasks to perform bioinformatics processes such as counting query operations. Count query is determining how many records in the database matching a given query, and securely querying a public database. The proposed framework includes a mechanism to protect the confidentiality of genomic data so that the cloud does not learn about genomic data, and query confidentiality (the cloud does not know about queries executed by researchers).

The proposed framework, depicted in Figure 1, consists of four components: Information Sources (IS) that represent data sources, Cloud Servers (CS) where encrypted data is stored and queries are processed, Trusted Research Center (TRC), and the Researcher (R) who inquiries about information. The following assumptions support the proposed framework:

- CS is a semi-honest entity. It applies the protocol but monitors communications and tries to report additional information during the execution of the researcher query. However, it does not deviate from the computational protocol and operates maliciously.
- TRC regards itself as a trustworthy entity because it is responsible for sharing the keys required for IS with trusted researchers.
- There is no collusion among CS, R, and IS. Furthermore, neither CS, IS nor TRC act in fraudulent misrepresentation to generate incorrect output.

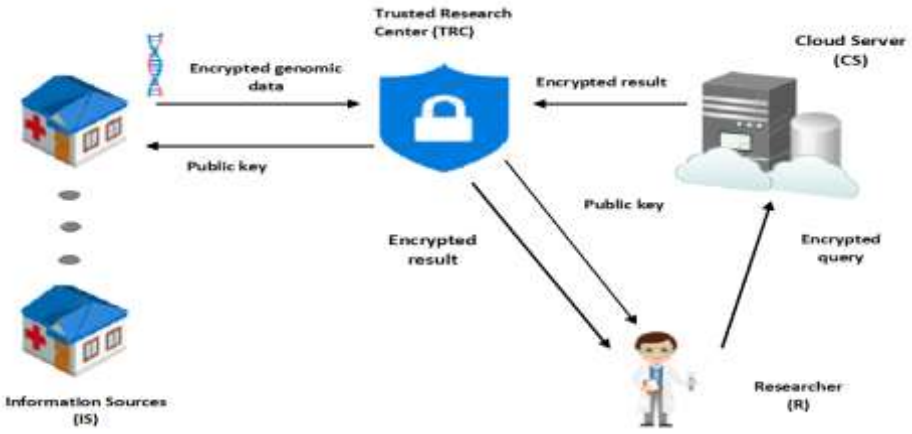


Figure 1: The proposed framework for protecting genomic data.

According to the framework, each component oversees a specific task that is distinct from the tasks assigned to the other components. The components work together to make the system safe and efficient. TRC generates key pairs (p_k, s_k) where $p_k$and $s_k$are the public and the

private keys respectively. The public key is sent from TRC to IS to encrypt the genomic data files. These files are sent from the IS to CS, which is responsible for storing them. On the other hand, a researcher who wants to query a specific genome sequence sends an encrypted query to the cloud. Then, the cloud returns the encrypted result to TRC. The result is decrypted by TRC, and then it is encrypted with the research public key and sent to the researcher.

4.1 Genomic data representation and encoding

The genomic data holds an organism's hereditary information. DNA is responsible for encoding the human genome. it consists of nucleotides that are represented as (A, C, G, T), and are bonded to each other in the form of A bonds to T and C bonds to G. SNP is the most popular form of DNA variation at a specific position in the genome, representing s a difference in a single nucleotide. Most SNPs don't have any effect on the health of humans. While some of these SNPs are responsible for developing particular diseases in humans.

The dataset used in this paper (https://www.kaggle.com/) is in the form of multiple fasta files (multi -fasta) that contains genomic data in the form of records of sequences with their corresponding descriptions. The description includes the unique identifier of the genome or gene along with its name.

Formally speaking, the dataset can be described as $\mathbb{S} = \{[D_1, S_1], \dots, [D_n S_n]\}$, where n is the number of human SNP sequences. $D_j \in S$ is the genomic data description that represents the SNP sequence ID. While $S_j \in S$ is a sequence of nucleotides (SNP). The genomic sequences are made up of four nucleotides (A, C, G, and T), and the SNP sequence, S_j is made up of two nucleotides as shown in Table 1.

Table 1: A sample of genomic data representation

	Description	Sequences						
		SNP1	SNP2	SNP3	SNP4	…..	SNP_{n-1}	SNP_n
1.	>NC_038294 |Betacoronavirus England 1| complete genome	AT	TT	AA	GA	….	AA	AA
2.	>MN541209 |Middle East respiratory syndrome-related coronavirus isolate SPC00440 S protein gene| partial cds	TG	TT	GA	TT		AA	GC
3.	>NC_019843 |Middle East respiratory syndrome coronavirus| complete genome	GA	TT	TA	AG		TC	GC
. . .	. . .							
n-2	>MG987420 |Middle East respiratory syndrome-related coronavirus isolate NL13892| complete genome	GA	TT	TA	AG		AA	CC
n-1	>MN541210 |Middle East respiratory syndrome-related coronavirus isolate SPC00441 S protein gene| partial cds	TG	TT	GA	TT		TC	CG
n	>MN541281 |Middle East respiratory syndrome-related coronavirus isolate SPC00579 S protein gene| partial cds	TG	TT	GA	TT		AA	CG

The genomic data needs to be encoded to be used for homomorphic computation. In this paper, each SNP is made up of two nucleotides. There are 16 possible combinations represented by AA, AT, AG, etc. The genomic sequence is encoded as an integer ranging from 1 to 16 as

shown in Table 1. For example, the sequence "ATCGAGTGCC" can be expressed as ["AT", "CG", "AG", "TG", "CC"] and is encoded to [1, 7, 3, 12, 5] as shown in Table 2.

Table 2 - Nucleotides encoding

AA	AC	AG	AT	CC	CA	CG	CT	TT	TA	TC	TG	GG	GC	GA	GT
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

4.2 Encryption of genomic data

Data owners send genomic data, $\mathbb{S}$, to TRC where encryption takes place to produce encrypted data, $\mathbb{S}'$. TRC encrypts the genomic data, which is the dataset $\mathbb{S}$, where each row represents the individual data (description part) and each column represents the attribute of the individual (sequence part). the encryption is done as follows: First, the description part contains a unique identifier and description for the sequence so it is important to encrypt this part of the record to avoid attempts to identify the sequence using the AES algorithm with a key size equal to 256 bits. Second, the sequence part which consists of a series of 'A', 'T', 'G', and 'C' must be encoded according to Table 1 and finally encrypted using one of the partially homomorphic encryption algorithms Paillier, RSA, or ElGamal. Algorithm 1 clarifies the encryption process of genomic data.

Algorithm 1. Genomic sequence encryption

Input:
$\mathbb{S}$: Genomic data
Output:
$\mathbb{S}'$: Encrypted genomic data

```
Begin
sequences ← [ ]
sequence_object ← read(𝕊)
foreach seq in sequence_object
    sequences ← append(seq)
For i ← 0 to | 𝕊 |
    Encrypt D_i with AES
    For j ← 0 to | S_i |
        Encode the nucleotide pairs according to table 1
        Encrypting S_ij by a partially homomorphic algorithm (Paillier, RSA, or
ElGamal) using Eq.1, Eq.4, or Eq.7.
    End for
End for
End foreach loop
End
```

4.3 Tree generation

The encrypted data, $\mathbb{S}'$, is uploaded to CS. When researchers attempt to query information, q, from the encrypted data. Researchers encrypt the query, q', and send it to the CS. CS generates an indexed tree, T, from the encrypted data, $\mathbb{S}'$ and performs a search operation. The adoption of indexed tree generation is because it is simpler to implement, uses less memory, and allows for faster node traversal. Each node in the tree, except the root, has four elements: value that is an encrypted form of the SNP, id (which is SNP position in the sequence), the count that represents the occurrences of SNP at this position, and child that represents

subsequent SNPs in the sequence. The generation of the tree from genomic data is shown in Algorithm 2.

Algorithm 2. Indexed tree generation	
Input: $\mathbb{S}'$: encrypted sequences of SNPs **Output**: Tree: Indexed Tree	
1:	Begin
2:	Tree ← root
3:	for i ← 1 to $\lvert \mathbb{S}' \rvert$
4:	SNP ← $\mathbb{S}'$ [i]
5:	sid ← i// index of the node
6:	child ← TreeNode(SNP,sid)
7:	flag ← 0 //to find repeated occurrences for the SNP
8:	n ← TreeNode
9:	for each node in the n. children
10:	Check the equality of values for node n and child
11:	count(n) + ← 1
12:	flag ← 1
13:	End foreach loop
14:	Check the flag value to add the child to the parent.
15:	End for
16	End

4.4 Secure count query

The purpose of the query count operation is to find the number of records that match the requested information. The researcher query is encrypted according to the algorithm used in encrypting the genomic data set (using Eq1, 4, or 7). The search operation depends on the homomorphic operation, which is either subtraction or division between encrypted SNPs in q' and encrypted SNPs in the encrypted indexed tree to ensure the equality of two SNPs in the predetermined positions, as shown in Algorithm 3. The use of a homomorphic operation allows the comparison of queried encrypted SNP values with stored encrypted SNP values without revealing their values.

Algorithm 3. Secure count query	
Input: Root: root node, list of encrypted SNPs, list of positions **Output**: encrypted result	
1:	Begin
2:	Encrypt the list of SNPs that will be queried with a partially homomorphic algorithm
3:	(Paillier, RSA, or ElGamal) using Eq.1, Eq.4, or Eq.7.
4:	Check the length of the list of encrypted NPSs and the positions
5:	QueryResult ← [ ] // list for results
6:	q ← [root] //insert tree in list
7:	Insert the first node of the tree into the list
8:	While length (list of encrypted SNPs) > 0
9:	

11: 12:	Perform a homomorphic operation using one of the equations (3, 6, or 9) between the stored nodes value in q and the first value of the queried node as appending the result in QueryResult list End while End

4.5 Counting query decryption

The encrypted query is delivered from the researcher (R) to the cloud (CS), where the search process is carried out for the required information. The results are returned in an encrypted form to the TRC, which in turn decrypts it using a decryption equation with the adopted algorithm using Equation 2, 5, or 8. Then TRC delivers the result to the intended researcher after encrypting it with the public key of the researcher.

5. Experimental Results

The proposed framework allows any computation to be transformed into a homomorphic operation that protects privacy from an attacker. For the proposed framework, three partial homomorphic encryption algorithms were investigated. CS and CI were run in two separate machines with Intel(R) Xeon(R) CPU @ 2.20GHz, 12 G RAM, and 107G Hard disk.

Furthermore, the proposed framework was tested on two real datasets from (https://www.kaggle.com/datasets/): MERS and SARSr-COV. The first dataset contains 1345 genomic sequences (records), with a minimum length of 110 nucleotides, a maximum length of 30119 nucleotides, and an average length of 14197 nucleotides. The second dataset contains 1578 records with a minimum of 87 nucleotides, a maximum of 29903 nucleotides, and an average of 18440 nucleotides. The length of the genomic sequences varies from 87 to 30119 nucleotides. In addition, the results report the impact of the key size on the performance of the proposed framework by setting the key size of the partial homomorphic encryption algorithms to 512 and 1024 bits and the key size of AES to 256.

5.1 Security analysis

To evaluate the security of the proposed framework, it is considered that the system lacks security if there is any revealing in the SNP sequence, query, and output.

In the proposed framework, IS sends genomic data to TRC, and TRC is responsible for encryption and uploads to CS, so no data leakage occurs during encryption and generation of encrypted genomic data. CS cannot infer any information since it received encrypted data. Additionally, TRC makes the public key available to researchers after they trust them to encrypt their queries, thus preventing leakage during research queries. Finally, since the query is encrypted and neither the TRC nor CS knows about it, it protects against leaks while receiving the results. Once the results are sent back to TRC, it is decrypted and passed on to the researchers since the query is not revealed, it will not know what the result means

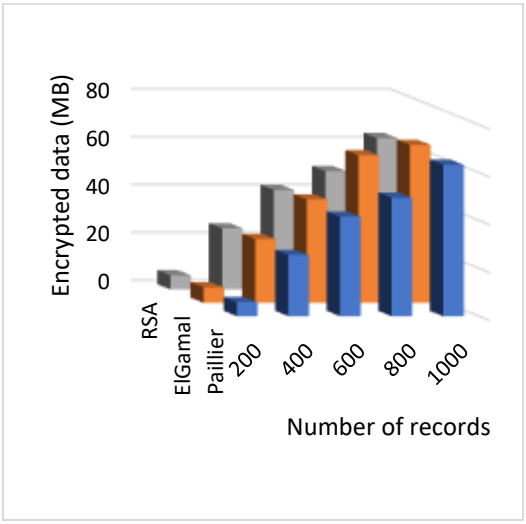
.

5.2 Performance analysis

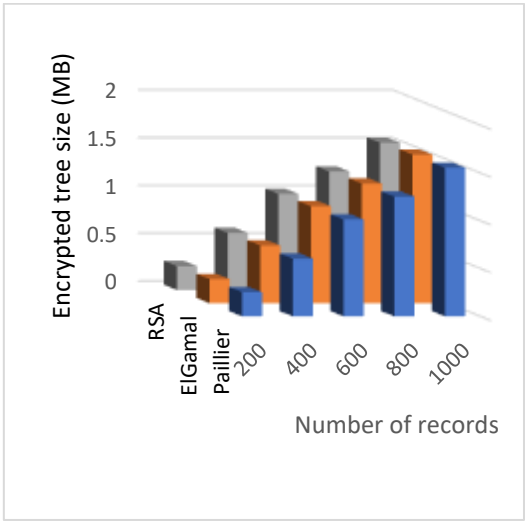
This section demonstrates the performance of the proposed framework in terms of storage, upload time, and search time. In addition, to assess the effect of data size on the performance of the proposed framework, each dataset was portioned into 200, 400, 600, 800, and 1000 records. Furthermore, an evaluation with different query sizes (10, 20, 30) was performed to demonstrate the effect of query size on the framework performance.

A. Storage analysis

The storage spaces required to represent the encrypted MERS dataset and its corresponding encrypted tree using three homomorphic ciphers Paillier, RSA, and ElGamal with two different key sizes (1024 and 512 bits) are shown in Figures 2 and 3 (A and B) respectively. The storage space for the encrypted data using Paillier, RSA, and ElGamal was 64,462 MB, 64,321 MB, and 68,320 MB respectively. While the storage of spaces required for the tree was 1740, 1542, and 1902 MB, respectively. As can be seen, representing the data as a tree saves approximately 95% of the storage space, which provides an effective storage solution for genomic datasets and influences query execution speed. Furthermore, as the number of records increased, the encrypted data and encrypted tree storage space increased linearly.

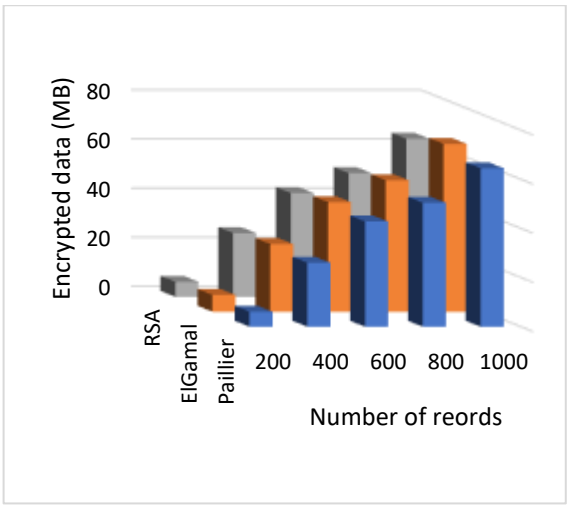


(A) encrypted data

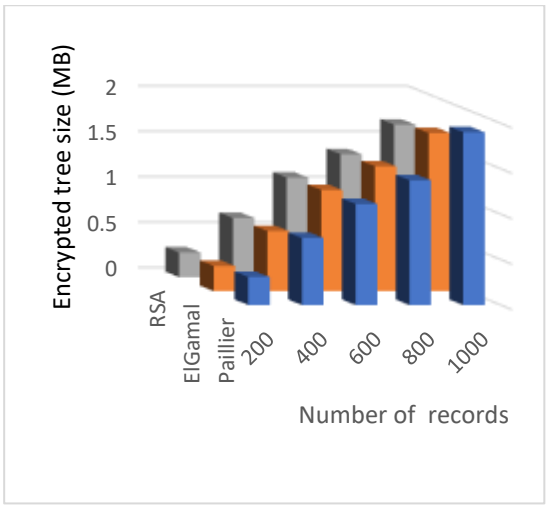


(B) Encrypted tree

Figure 2: Storage space of encrypted data and encrypted tree when key equals 512



(A) encrypted data



(B) Encrypted tree

Figure 3: Storage space of encrypted data and encrypted tree when key equals 1024

B. Upload and query execution times

The performance of the proposed method in terms of upload and query execution (search) times are reported in Tables 3 -6. As can be seen, upload time increases linearly as the number of records increases. When the data size and query size are small, the RSA algorithm

outperforms Paillier and ElGamal in terms of query time execution, but when the data size is medium or large, Paillier outperforms the RSA and ElGamal. The execution time of the query is proportional to the number of SNPs. Although the first and second datasets have the same number of records, the query execution time differs because it depends on the depth of the tree (the index tree in the second data set is larger) and there is no repeating sequence as in the first data set. The size of the tree and the number and position of SNPs being queried all affect the speed of query execution. Furthermore, the size of the encrypted tree is too small in comparison to the encrypted data. This reduces computational effort and storage requirements compared to raw data.

Table 3: Comparison of the three homomorphic encryption algorithms in terms of upload time, encrypted tree size, and search time with a key size of 1024 for MERS

Algorithm	# Records	Upload time (second)	Encrypted genomic data size (MB)	Encrypted tree size (MB)	Search time (second)		
					10 Query size	20 Query size	30 Query size
Paillier	200	1.3345	211	0.308	0.0383	0.0541	0.0844
	400	1.8987	300	0.738	0.0316	0.0913	0.1394
	600	9.5509	1510	1.112	0.0502	0.1041	0.1592
	800	11.3219	1790	1.373	0.0595	0.1259	0.1855
	1000	14.4845	2290	1.902	0.0586	0.1301	0.2057
ElGamal	200	2.6774	423.3	0.356	0.0405	0.0828	0.1278
	400	9.9936	1580	0.854	0.0536	0.1305	0.2226
	600	19.1018	3020	1.438	0.0748	0.1457	0.237
	800	22.7071	3590	1.774	0.1248	0.1748	0.2687
	1000	29.0322	4590	2.202	0.1931	0.2144	0.4962
RSA	200	1.2371	195.6	0.272	0.0341	0.0621	0.1027
	400	5.5351	875.1	0.652	0.0371	0.1058	0.1368
	600	8.7919	1390	1.096	0.0546	0.1125	0.1791
	800	10.4364	1650	1.353	0.0550	0.1239	0.2169
	1000	13.4092	2120	1.679	0.0649	0.1440	0.2481

Table 4: Comparison of the three homomorphic encryption algorithms in terms of upload time, encrypted tree size, and search time with a key size of 512 for MERS

Algorithm	# Records	Upload time (second)	Encrypted data size (MB)	Encrypted tree size (MB)	Search time (second)		
					10 Query size	20 Query size	30 Query size
Paillier	200	0.6768	107	0.252	0.0252	0.0420	0.0733
	400	3.0310	479.2	0.603	0.0423	0.0812	0.1227
	600	4.9412	781.2	1.014	0.0461	0.1043	0.1746
	800	5.8672	927.6	1.252	0.0538	0.1267	0.1835
	1000	7.3371	1160	1.555	0.0633	0.1304	0.2083
ElGamal	200	1.3593	214.9	0.308	0.0326	0.0682	0.1081
	400	6.0898	962.8	0.738	0.0582	0.1103	0.1674
	600	9.6774	1530	1.242	0.0772	0.1454	0.2063
	800	11.5117	1820	1.533	0.0864	0.1497	0.2548
	1000	14.7375	2330	1.904	0.1092	0.1997	0.3126
RSA	200	0.6230	98	0.25	0.0296	0.0550	0.0847
	400	2.7875	440.7	0.598	0.0415	0.0991	0.1387
	600	4.5433	718.3	1.006	0.0472	0.1018	0.1718
	800	5.3960	853.1	1.241	0.0640	0.1349	0.1841
	1000	6.7679	1070	1.542	0.0679	0.1104	0.1568

Table 5: Comparison of the three homomorphic encryption algorithms in terms of upload time, encrypted tree size, and search time with a key size of 1024 for SARSr-COV

Algorithm	# Records	Upload time (second)	Encrypted database size (MB)	Encrypted tree size (MB)	Search time (second)		
					10 Query size	20 Query size	30 Query size
Paillier	200	3.0449	481.4	0.664	2.4459	2.4588	2.5154
	400	4.3700	690.9	1.257	2.7594	2.8056	2.8166
	600	6.9576	1100	1.697	3.0999	3.1197	3.1513
	800	8.4756	1340	2.45	3.3302	3.3659	3.4210
	1000	9.1081	1440	2.782	3.8816	3.9043	3.9570
ElGamal	200	1.3377	489	0.664	3.5183	3.5361	3.6225
	400	4.3700	701.8	1.257	4.0440	4.0667	4.1061
	600	7.0208	1130	1.697	4.4761	4.4802	4.5877
	800	8.4756	1360	2.45	4.9013	4.9646	5.0322
	1000	9.1081	1460	2.782	5.7848	5.8527	5.9256
RSA	200	2.7413	433.4	0.654	3.3881	3.4152	3.4694
	9400	3.9500	624.5	1.239	3.9882	3.9909	4.0245
	600	6.3883	1010	1.671	4.5593	4.5644	4.5721
	800	7.7166	1220	2.412	4.8764	4.9495	4.9882
	1000	8.2226	1300	2.74	5.5872	5.5898	5.7742

Table 6: Comparison of the three homomorphic encryption algorithms in terms of upload time, encrypted tree size, and search time with a key size of 512 for SARS-COV2

Algorithm	# Records	Upload time (second)	Encrypted database size (MB)	Encrypted tree size (MB)	Search time (second)		
					10 Query size	20 Query size	30 Query size
Paillier	200	1.5395	243.4	0.607	2.4278	2.4461	2.4909
	400	2.2093	349.3	1.143	2.7553	2.7803	2.7917
	600	3.6312	574.1	1.535	3.0323	3.0996	3.1320
	800	4.3978	695.3	2.222	3.2574	3.3201	3.3882
	1000	4.7033	743.6	2.52	3.8981	3.9449	3.9653
ElGamal	200	1.5395	963	0.607	3.3932	3.4491	3.4857
	400	2.2093	1350	1.143	3.8863	3.9270	3.9537
	600	3.6312	2220	1.535	4.3951	4.4950	4.5362
	800	4.3978	2690	2.222	4.8025	4.8361	4.9637
	1000	4.7033	2870	2.52	5.6095	5.6955	5.7233
RSA	200	1.3807	218.3	0.602	3.3160	3.3312	3.3225
	400	1.9898	314.6	1.133	3.7308	3.7588	3.8418
	600	3.2802	518.6	1.521	4.2414	4.2781	4.3051
	800	3.9765	628.7	2.201	4.5629	4.6100	4.6333
	1000	4.2536	672.5	2.497	5.5554	5.5725	5.6961

C. Comparison results

A performance comparison between the proposed framework and [9] was performed to evaluate the process of secure counting queries. The proposed framework uses Paillier to

encrypt datasets stored in the cloud, homomorphic subtraction, and division operations in the search process.

From the results shown in Tables 7 and 8, it is clear that the proposed framework is superior. The work in [9] takes 0.8441 seconds to process a secure query consisting of 10 SNPs, while the proposed method takes 0.0252 seconds to process the same query. This is due to the use of homomorphic operations, which eliminates the need for the additional operation (garbled circuit) required in [9]. Therefore, the proposed framework improvement for handling secure search queries compared to [9] is an average of 96% in query execution time.

Table 7- Comparison between Hasan et al. [9] and the proposed method using Paillier with key size =512

Method	Query size	Search time (second)				
		200 records	400 records	600 records	800 records	1000 records
Hasan et al. [8]	10	0.8441	1.4086	2.3957	2.7720	3.0322
	20	0.9218	1.4366	2.5341	2.8975	3.1445
	30	1.0774	1.4724	2.5661	2.9828	3.5133
The Proposed method	10	0.0252	0.0423	0.0461	0.0538	0.0633
	20	0.0420	0.0812	0.1043	0.1267	0.1304
	30	0.0733	0.1227	0.1746	0.1835	0.2013

Table 8- Comparison between Hasan et al. [9] and the proposed method using Paillier with key size =1024

Method	Query size	Search time (second)				
		200 records	400 records	600 records	800 records	1000 records
Hasan et al. [8]	10	0.9305	1.7054	2.5369	2.6322	3.2347
	20	0.9596	1.7313	2.6561	2.7167	3.4242
	30	1.1329	1.7622	2.8134	3.0957	3.6124
The Proposed method	10	0.0383	0.0316	0.0502	0.0595	0.0586
	20	0.0541	0.0913	0.1041	0.1259	0.1301
	30	0.0844	0.1394	0.1592	0.1855	0.2057

6. Conclusion

In this paper, we presented a secure framework for storing and performing operations on encrypted genomic data that returns results in an encrypted form, thereby providing a secure work environment to protect genomic data privacy. The proposed method converts the encrypted data stored on the cloud into an indexed tree to improve search speed and save space when running the counting query. The proposed method used ensure that the processing of this data takes place with a high degree of confidentiality and is not disclosed at any stage of the data processing, and homomorphic algorithms are used to protect the data and take advantage of their properties. The presented work significantly improves query execution time compared to existing work.

In the future, the framework can be developed further to be suitable for working with genome sequences in a fully encrypted environment.

7. Disclosure and conflict of interest

The authors declare that they have no conflicts of interest.

References

[1] M. Achour and A. Belmadani, "Towards Protect a Specific Information in Genomic Data," International Journal of Intelligent Engineering and Systems, vol. 14, no. 4, pp.91-104, 2021.

[2] Y. Jiang, T. Shang, and J. Liu, "SM algorithms-based encryption scheme for large genomic data files," Digital Communications and Networks, vol. 7, pp. 543-550, 2021.

[3] M. Akgün, A. O. Bayrak, B. Ozer, and M. Ş. Sağıroğlu, "Privacy preserving processing of genomic data: A survey," *Journal of Biomedical Informatics*, vol. 56, pp. 103–111, Aug. 2015, doi: https://doi.org/10.1016/j.jbi.2015.05.022.

[4] G. O. Ogunleye and S. E. Akinsanya, "Elliptic Curve Cryptography Performance Evaluation for Securing Multi-Factor Systems in a Cloud Computing Environment," Iraqi Journal of Science, vol. 63, no. 7, pp. 3212-3224, 2022.

[5] G. S. Çetin, H. Chen, K. Laine, K. Lauter, P. Rindal, and Y. Xi, a, "Privatequeries on encrypted genomic data," BMC Medical Genomics, vol. 10, no. 45, pp. 1-14, 2017.

[6] M. S. Fadhil, A. K. Farhan and M. N. Fadhil, "A lightweight AES Algorithm Implementation for Secure IoT Environment," Iraqi Journal of Science, vol. 62, no. 8, pp. 2759-2770, 2021.

[7] J. L. Raisaro, G. Choi, S. Pradervand, R. Colsenet, N. Jacquemont, N. Rosat, V. Mooser and J. Hubaux, "Protecting Privacy and Security of Genomic Data in i2b2 With Homomorphic Encryption and Differential Privacy," IEEE/ACM transactions on computational biology and bioinformatics, vol. 15, no. 5, pp. 1413-1426, 2018.

[8] H. Chen, R. Gilad-Bachrach, K. Han, Z. Huang, A. Jalali, K. Laine, and K. Lauter, "Logistic regression over encrypted data from fully homomorphic encryption," BMC Medical Genomics, vol. 11, no. 81, pp.3-12, 2018.

[9] M. Z. Hasan, S. R. Mahdi N. Sadat and N. Mohammed, "Secure count query on encrypted genomic data," Journal of Biomedical Informatics, ELSEVIER , vol. 81, pp. 41-52, 2018.

[10] L.Chen, M. Aziz, N. Mohammed, and X. Jiang, "Secure large-scale genome data storage and query," Computer Methods and Programs in Biomedicine, ELSEVIER, vol. 165, pp. 129-137, 2018.

[11] M. Blatt, A. Gusev, Y. Polyakov, and S. Goldwasser, "Secure large-scale genome-wide association studies using homomorphic encryption," Proceedings of the National Academy of Sciences, vol. 117, no. 21, pp. 11608-11613, 2020.

[12] S. R. Mahdi, M. Al Aziz, N. Mohammed, and X. Jiang, "Privacy-preserving string search on encrypted genomic data using a generalized suffix tree," Informatics in Medicine Unlocked, vol. 23, no. 100525, 2021.

[13] E. Yilmaz, T. Ji, E. Ayday and P. Li, "Genomic Data Sharing under Dependent Local Differential Privacy," in proceedings of 12th ACM conference on data and application security and privacy, Baltimore, 2022.

[14] A. K. Manocha, M. Singh, S. Jian, and V. Jain, "Smart Computational Intelligence in Biomedical and Health Informatics," Boca Raton, Taylor & Francis Group, L.L.C., CRC Press, 2022.

[15] K. V. Chaitanya, "Genome and Genomics: From Archaea to Eukaryotes, "Singapore: Springer, 2019.

[16] K. Ayoz, E. Ayday, and A. E. Cicek, "Genome Reconstruction Attacks Against Genomic Data-Sharing Beacons," Proceedings on Privacy Enhancing Technologies, vol. 3, pp.28-48, 2021.

[17] E. M. Alsaedi and A. K. Farhan, "Retrieving Encrypted Images Using Convolution Neural Network and Fully Homomorphic Encryption," Baghdad science journal, vol. 20, no. 1, pp. 206-220, 2023.

[18] R. K. Challa and V. K. Gunta, "A Modified Symmetric Key Fully Homomorphic Encryption Scheme Based on Read-Muller Code," Baghdad Science Journal, vol. 18, no. 2, pp. 899-906, 2021.

[19] A. Chatterjee and K. M. Aung, "Fully Homomorphic Encryption in Real World Application," Singapore: Springer Nature Singapore, Pte Ltd., 2019.

[20] K. Munjal and R. Bhatia, "A systematic review of homomorphic encryption and its contributions in healthcare industry," Complex & Intelligent Systems, pp.1-28, 2022.

[21] M. Tehranipoor, "Emerging Topics in Hardware Security," Switzerland: Springer, 2021.

[22] Ç. K. Koç, F. Özdemir, and Z. Ö. Özger, "Partially Homomorphic Encryption," Turkey: Springer Nature, 2021.

[23] M. Liu, Y. Luo, C. Yang, D. Xu, and T. Wu, " Method and Application of Homomorphic Subtraction of the Paillier Cryptosystem in Secure Multi-party Computational Geometry. In:

Zhang, X., Liu, G., Qiu, M., Xiang, W., Huang, T. (eds) Cloud Computing, Smart Grid and Innovative Frontiers in Telecommunications. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, Springer, Cham. vol 322, pp.569-581, 2020.

[24] H. J. Kiratsata and M. Panchal, "A Comparative Analysis of Machine Learning Models developed from Homomorphic Encryption based RSA and Paillier algorithm," in proceedings of 5th International Conference on Intelligent Computing and Control Systems (ICICCS), pp.1458-1465, 2021.

[25] N. H. Hussein and M. A. Ali, "Medical Image Compression and Encryption Using Adaptive Arithmetic Coding, Quantization Technique and RSA in DWT Domain," Iraqi Journal of Science, vol. 63, no. 5, pp. 2279-2296, 2022.

[26] X. Yi, R. Paulet, and E. Bertino, "Homomorphic Encryption and Applications," London: Springer London, 2014.

[27] O. A. Imran, S. F. Yousifa, I. S. Hameeda, W. N. Abeda, "Implementation of El-Gamal algorithm for speech signals encryption and decryption," special issue of Procedia Computer Science, vol. 167, pp. 1028-1037, in proceedings of International Conference on Computational Intelligence and Data Science (ICCIDS 2019), 2020.

[28] N. Domadiya and U. P. Rao, "ElGamal Homomorphic Encryption-Based Privacy Preserving Association Rule Mining on Horizontally Partitioned Healthcare Data," Journal of The Institution of Engineers (India): Series B, vol. 103, no. 3, pp. 817-830, 2022.

[29] A. Hamza and B. Kumar, "A Review Paper on DES, AES, RSA Encryption Standards," in proceedings of 9th International Conference on System Modeling and Advancement in Research Trends (SMART), Moradabad, India, pp. 333-338, 2020.