Received: December 31, 2020. Revised: March 22, 2021.

INASS

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

Towards Protect a Specific Information in Genomic Data

Moufida Achour[1]* **Abderrahim Belmadani[1]**

[1]*Computer Science Department, Faculty of Mathematics and Computer Science,*
Université des Sciences et de la Technologie d'Oran Mohamed-Boudiaf
USTO-MB BP 1505 El M'naouer, Oran, Algeria
* Corresponding author's Email: moufida.achour@univ-usto.dz

Abstract: Research on the privacy of human genomic data has significantly increased these last years to find out effective solutions to protect the most critical information resulting from an alignment operation such as the genomic position, cigar and Read content. In this paper, the authors propose an encoding technique based on homomorphic encryption to protect genomic positions of DNA sequences for a secure storage of BAM file. This technique uses two Discrete Gaussian Sampling algorithms for comparison purposes which is Gauss and Knuth-Yao algorithm. The framework was implemented on Java from scratch using container data structure with standard java HashMap for the representation of duplicates positions elements exiting in BAM file. The proposed method was assessed on the NA12878 dataset with initially 2 million positions to encrypt. Obtained results show that the FV-Knuth-Yao technique with HashMap architecture storing 165667 positions and a polynomial degree equal to 64 ensures a fast encryption time estimated to 6,5 minutes. The decryption operation needs 1 hour with zero loss of information. The acceptable security level reached 2^{64} operations to break the cryptosystem. To demonstrate the efficiency, security and reliability of our approach, the FV-Knuth-Yao was compared with the OPE Method as implemented in SECRAM tool which is based on a deterministic encryption scheme. Our work mainly focuses on the security of genomic data without compression of BAM file.

Keywords: : Bioinformatics, FV, Genomic data, Homomorphic encryption, Knuth-yao, Privacy, Security.

Nomenclature

Cigar	Compact Idiosyncratic Grapped Alignement Report
DNA	Deoxyribonucleic Acid
BAM	Binary Alignment Map
FV	Fan & Vercauteren
OPE	Order Preserving Encryption
SECRAM	Selective Retrieval on Encrypted and Compressed Reference Oriented Alignment Map

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 to no longer be limited to the determination of sequences, it now includes the analysis of the expression and function of both genes and proteins [1]. At the same time, Bioinformatics, a science that uses different scientific fields such as biology, medicine, computer science and physics, with the aim of solving issues in the field of Biology, is increasingly integrated to these analyses. The scientific community faced a large volume of genomic data since 2003 with the finalization of the human genome project. Today, a complete sequence of the human genome can be determined in an acceptable time, thanks to the latest generation of high-throughput sequencing technologies. Genome sequencing is used in the context of diagnosis of pathologies and preventive treatments. DNA which represents the unique genetic information can be obtained from a variety of sources like hair, skin, blood, and saliva. After DNA extraction, the sample is sequenced by using a high throughput sequencing

platform, which is called NGS (Next Generation Sequencing), the most used is Illumina. This manipulation generates a file in a digital format called FASTA or FASQ (stores both a DNA sequence and its corresponding quality scores) which contains the genomic data that can be given to an individual, researcher or clinician in several ways, for example on a removable disk or it can be downloaded from a cloud. To respond to a specific need for DNA or RNA sequences, a researcher or clinician uses Bioinformatics techniques. Fig. 1 shows the visual representation of the DNA/RNA Sequences Analysis Pipeline. Using a FASQ file, an alignment (mapping) is applied with respect to a reference genome (GRCh37 or GRCh38) pulled from Ensembl base. For DNA-SEQs, the Burrows-Wheeler Aligner (BWA) is recognized as the most appropriate alignment tool [2]. The result of this process is a Sequence Alignment Map (SAM) file or its binary format which is the BAM. This file contains biological sequences aligned to a reference sequence in a machine readable format. Methods like Samtools [3] and The Genome Analysis Toolkit (GATK) are recognized as being the most accurate for more advanced Bioinformatics analysis [4-5] such as variant calling (BAM to VCF), to look for a single-nucleotide polymorphism (SNP) "genetic mutations". The FreeBayes variant detector gives by far the best compromise in speed and sensitivity [6-7]. For the annotation of VCF, there is ANNOVAR and the Variant Effect Predictor (VEP) which is the most recommended [8]. These tools make it possible to have an enriched VCF on which we can apply a filtering and launch analysis requests. The standardization of smart card practice in the field of preventive health may be possible with the future developments of smart cards, Bioinformatics and preventive health care technologies. Each individual would have the variant of information on their own smart cards.

In this research work we focus on the BAM file which is formed of two parts. The head gives information on the genome or on the mapping and the body which is in tabular form is composed mainly of 12 fields (the name of the reading, binary information of the reading, the sequence of which the reading is mapped, position where the reading aligned to, the quality of the mapping, CIGAR which is used to understand how the query sequence aligns to the reference genome, RNEXT which represents the reference name of the next read, PNEXT represents a Position of the next read, Template Length "length of a fragment", Flag "this is where the alignment tool stores information on the read, it is represented by a combination of bitwise FLAGs and each bit has an explanation, Flag identifies mating in the case of Paired-end reads which are stored as two alignments records with the same QNAME" and finally an Auxiliary data "it is an optional information fields associated with tags") [9-10]. Table 1 shows a sample of parameters the BAM file displays by chromosome. Although, sequence alignment and private genomic data querying help researchers, biologists and clinicians to respond to issues, they introduce high risks of disclosure of private information. What impacts might be on the people concerned in the event of illegitimate access, unwanted modification, and disappearance? What measures of prevention and protection should be foreseen to reduce these risks to an acceptable level? What type of security and privacy requirements are mandatory at each stage of genomic data processing? By providing such a framework, we will be able to figure out the issues of privacy and security of genomic data.

The authors are working on a project of secure storage of genomic data for future queries needs, however they deal in a first phase with the encryption of the most sensitive fields present in a BAM file meaning the genomic positions.

To achieve this research work, the authors chose the FV cryptosystem because of the following characteristics:

a) **Efficiency**: Fv is a powerful homomorphic encryption based on the Ring Learning With Error (RLWE) problem which is hard solving [11] and is designed to withstand attacks by quantum computers.

b) **Resistance to attacks**: in presence of the necessary material resources, very high parameters values of the RLWE offer a high security of the cryptosystem.

c) **Unpredictability & Unbiasability:** because of the random polynomials coefficients of the public and private keys, FV cryptosystem becomes unpredictable and cannot be biased by a third party easily.

Initially, the FV cryptosystem with discrete Gaussian sampling algorithm was implemented and then compared to FV using a Knuth Yao Algorithm on the BAM file including all genomic positions. After analyzing the results obtained, we have detected duplicate elements which were encrypted several times, this caused a considerable computing time. To overcome this phenomenon, a HashMap container structure has been applied with a format (key, Value), such as:

Key: represents non duplicate genomic position.

Value: is an ArrayList regrouping the information corresponding to the different records having the same position.

This architecture provides many advantages to our scheme mainly an optimizing storage, faster data retrieval, an indexation of genomic data for future queried positions and a minimization of execution times of encryption and decryption operations (the latter is the main contribution of this paper).

Additionally to this, FV-Knuth-Yao was compared to OPE which offers the possibility to perform operations on encrypted data while preserving the numerical order of the plaintexts. OPE offers an unequal efficiency in terms of query processing and computation time for encryption and decryption operations but is not secure [11].

The remainder of this paper is organized as follows: section 2 reviews some related works in the field of the security of genomic data during the storage and query process. Section 3 presents a detailed description of the method which is used in the implementation of our approach. In section 4, the proposed approach is explained. Section 5 presents the experiments and results discussions. Section 6 describes a comparison study. Finally, authors give concluding remarks in Section 7.

2. Related works

Cryptography techniques based on homomorphic encryption have seen a considerable increase in the last years in order to remove the constraints related to the disclosure of genomic information during different phases of the genomic sequence processing; mainly during storage, search and query process.

In 2020, Marcelo Blatt et al. [12] proposed a secure framework based on a variant of many advanced homomorphic encryption schemes such as the Cheon-Kim-Kim-Song (CKKS) scheme, Double-Chinese Remainder Theorem (CRT), Residue Number System (RNS) and RLWE scheme. The proposed approach allows the GWAS (Genome-Wide Association Studies) analysis of 25000 participants (variants "SNP" analysis for detecting diseases) on encrypted form without any interaction of individuals. The privacy proposed model could perform a GWAS analyses of 100,000 individuals and 500,000 SNPs in 5.6 hours with one server node which is equivalent to 11 minutes with 31 server nodes. The GWAS developed protocol is publicly available, it was implemented with an open-source lattice cryptography library named PALISADE v1.4.0 on a server with 2 processor Intel(R) Xeon(R) CPU E5-2680 v4 @ 2.40GHz, each one with 14 cores and 500GB of memory.

Johannes Buchmann et al [13] proposed a secure storage solution of genetic data in a certified database for 100 years. The proposed framework, allows maintaining the integrity and authenticity of the full set of raw sequencer reads, alignment and genomic variant data and also a query of specific position in the genome. The scheme uses cryptographic algorithms such as hash functions: SHA-2 hash function family, signature based timestamps based on the XMSS signature and a hash function which is proposed by Halevi and Micali in [14]. The cryptography parameters used are based on the recommendations made by Lenstra and Verheul in [15] with Protect, Update, PartialProof, and Verify steps to protect a genomic data in different phases. The storage space needed for 100 years was estimated at 5.3 TB in 2019. They needed approximately 5.9 h for the initial integrity proof and up to 6.3 h to update it when cryptographic primitives must be replaced. The size of a partial integrity proof for a genome subset of size 105 after 100 years is approximately 130 MB with 5s for the checking. The authors estimated that the time could decrease in the future with a powerful material and resources.

Tan Ping Zhou et al [16] proposed a secure searching framework of genetic data which are stored in VCF format and are composed essentially of chrome (chr), position (pos), locus (loc), reference (ref), alternate (alt) and type. The proposed optimized model is based on RLWE and Ring-GSW homomorphic encryption algorithms and gives a solution to the KSC17 model [17] with a correction of three errors namely hash collision error (HCE), coefficient combination error (CCE) and losing of partial coefficient error (LPCE). It was implemented on a single core (i7-6700HQ, 2.60GHz) processor and the source code is publicly available. The model allows performing high efficiency query of gene data with $2^{-37.4}$ reduced probability of searching error on a semi-trusted business cloud comparing it to KSC17 solution.

Jean Louis Raisaro et al [18] designed a solution called i2b2 (Informatics for Integrating Biology and the Bedside) to allow a researcher to safely explore 3000 genetic variants on a cohort of 5000 individuals in a minimum of 5 seconds. This is the first operational clinical data warehouse solution in a hospital environment to maintain confidentiality.

Gizem S Çetin et al [19] presented a string matching protocol based on FV homomorphic encryption by using the Simple Encrypted Arithmetic Library (SEAL v2.1). The proposed framework allows launching search queries showing the presence of genetic mutations without revealing any information about the genomic data. The solution achieves an operating time of only 4s and a

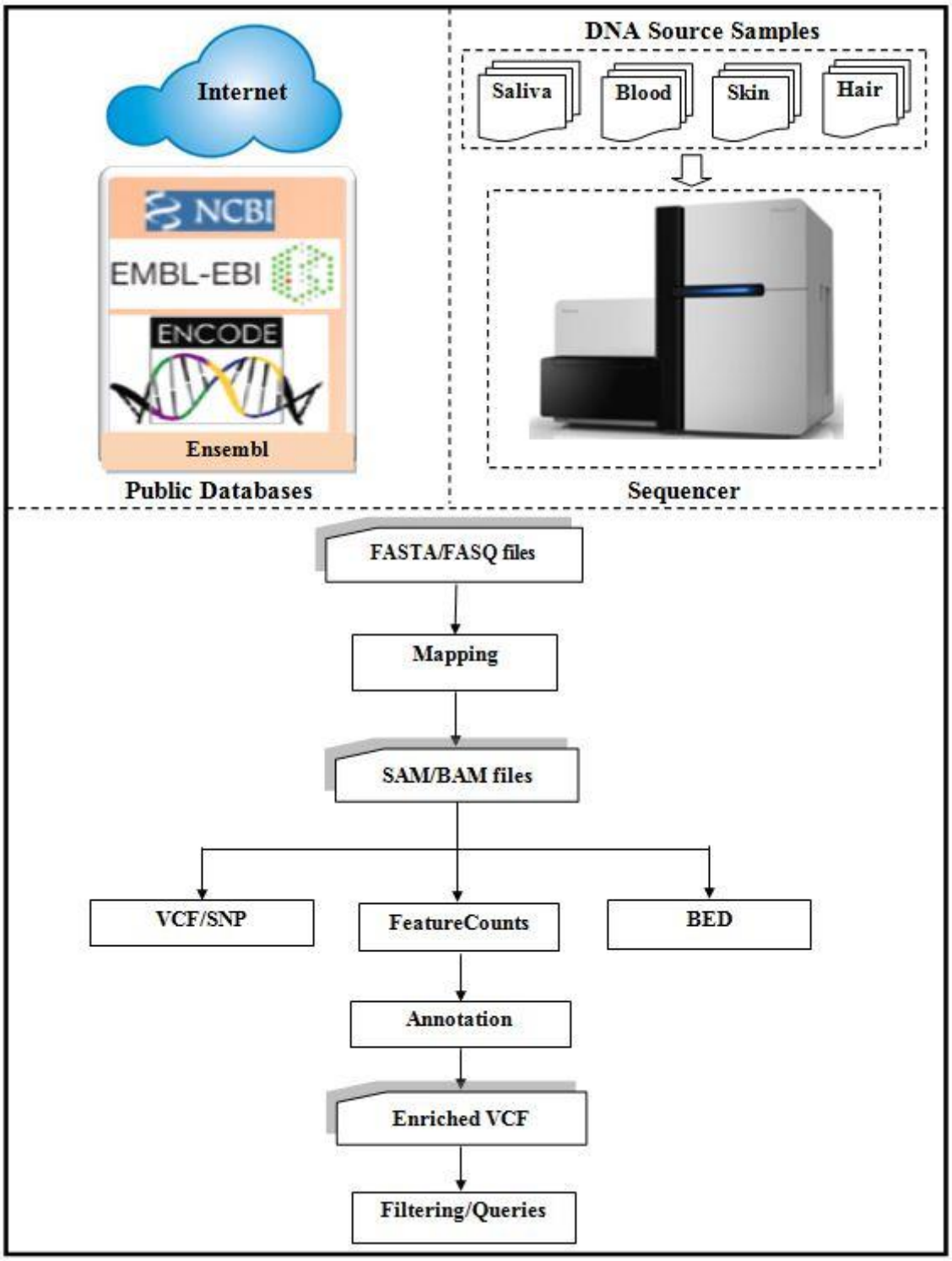

Figure. 1 DNA /RNA Sequences analysis pipeline

Table. 1 BAM file sample information for each chromosome

Position	Start	End	Chr	Qname	Read Length	Flag	Mapping	Rnext	Pnext	Tlen	QualityScore	Cigar	Tags	Read
4130029	4130029	4130053	1	M03878:7:000	121	161	44	9	33676658	0	ABBAAFBFBFFF	88S25M8S	MD:Z:25RG:	CTGTGGCTGAAGAAAAAGGAGC
10617886	10617886	10617912	1	M03878:7:000	121	145	44	2	123830577	0	...../111111////	8S27M86S	MD:Z:4C22R	CCCTCTCTCCTATCTTTTTTTTTT
14584451	14584451	14584476	1	M03878:7:000	121	129	44	7	120509803	0	BCCCDFFFFFFFC	88S26M7S	MD:Z:26RG:	ATCAGATAATAAAATGATAATA
14584471	14584471	14584498	1	M03878:7:000	121	129	44	3	27680576	0	CCDDDFFFFCDC	74S28M19S	MD:Z:25A2R	GTTAAGATCGGAAGAGCGTCGT
18197832	18197832	18197868	1	M03878:7:000	121	145	44	3	178927877	0	-/B--9///9--9---	69S37M15S	MD:Z:2T6T1	TTTTCTCTTCTGATCGCCCCCCCC
25248667	25248667	25248705	1	M03878:7:000	121	89	2	=	25248667	0	1:00.://///...A--	4S9M1I30M7	MD:Z:2C7C0	TGCATTGTTTCTTGTTTTTTTTTC

communication cost of 2 MB, by querying the presence of 5 mutations from a set of encrypted data of 100,000 mutations.

Chibuike Ugwuoke et al [20] presented two cryptography techniques based on homomorphic encryption schemes and secure multiparty computation for the storage and processing (calculation of LD "linkage disequilibrium" measures) of genomic data. The proposed approach made a consistent improvement of 98.4% for the storage and computational complexity by comparing it with the model designed in [21].

Ferdinando Montecuollo et al [22] have developped an open source tool in C++ named E2FM-index that can compute and encrypt a large amount of genomic sequences in FASTA format. These genomic sequences can be stored in a compressed and encrypted format at the same time and queries are also possible to be achieved. E2FM-index allows building auto-indexes which takes up to 1/20 of the storage required by the FASTA input file and 95% of storage is saved when indexing very similar sequences. Moreover, it can search the exact indexes built for patterns in times ranging from a few milliseconds to a few hundred milliseconds and this depends on the length of the pattern. Pattern-searching such as "count" and "locate" queries require decryption in memory of the data portion of the query.

Mohammad Zahidul Hasan [23] proposed a secure outsourcing solution for genomic data in a cloud server that allows three different computations (counting requests, hamming and editing distance). An encrypted way (advanced encryption standard in counter mode "AES in CTR mode") was used without revealing the information in order to preserve the confidentiality of data, queries, and results. The proposed method is reliable and efficient for performing safely the counting queries on encrypted genomic data.

Kana Shimizu and al [24] proposed a tool called PBWT-sec based on the Burrows-Wheeler transform method (PBWT more precisely) with an additive-homomorphic public-key encryption scheme (KeyGen; Enc; Dec) which allows to perform additive operations on encrypted values in order to realize SNP searches in large genomic databases without revealing the interrogated data. Compared to other tools, PBWT-sec is better in terms of time execution and data transfer. The estimated time on a single computer is 15.5s with homomorphic encryption for a database of 2184 genomes without losing the position of the query. Searches on a compute node are estimated at 18.6s and 133s, depending on the level of confidentiality.

Zhicong Huang et al [25] proposed a secure solution for storage genomic data called SECRAM. It allows converting a BAM file to SECRAM file and SECRAM to BAM file mainly through 6 procedures respectively (transposition, Inverse transposition, compression, decompression, encryption, and decryption). Position requests are made using OPE. The SECRAM tool is able to perform all operations in one step and in a record time of about 1 hour for a 100 GB BAM file with 24 threads and less than 0.25 seconds for each of the six procedures for 1 MB of data. In this scheme, only positions and Cigar are encrypted with OPE and Symmetric Encryption respectively. The position encryption is applied before compression in order to enable indexing. Unfortunately, as describe in [25] and also in the literature [11], OPE is not a safe encryption technique for sensitive information as genomic positions. The authors in [25] recommend to use other alternatives to the OPE to ensure a secure storage of the BAM file by encrypting positions and also to allow a search and query of the desired range information.

In our work, we focus overcoming these drawbacks. As far as we know, The FV method with the Knuth-Yao algorithm has never been used for the protection of genomic position. The present study provides from scratch implementation (without using existing libraries) a combination of a solid method of homomorphic encryption which is FV with the efficient discrete Gaussian sampling across multiple platform and this by following the process described in Fig. 2.

3. Methodology

3.1 Homomorphic encryption

Homomorphic Encryption is a cryptosystem that allows performing computations directly on encrypted data without decrypting it first. It is known to be partial or somewhat homomorphic encryption SHE when it allows making one or some operations on the encrypted data (for example: Paillier cryptosystem with homomorphic under Addition or ElGamal for multiplication) [26]. It is fully homomorphic encryption FHE when it allows making all possible operations on plaintext and ciphertext. This technique was first introduced in 2009 by Gentry in [27] through the introduction of a model based on the bootstrapping mechanism which serves to overcome the problem of noise that increases with each operation (addition, multiplication, etc...). If the noise exceeds a defined threshold, this will have an impact on the decryption. After 2009, many other patterns were developed,

thanks to Brakerski and Vaikuntanathan [28], followed by Brakerski, Gentry and Vaikuntanathan [29] and Fan & Vercauteren in 2012 [30]. According to the consortium European H2020 HEAT [31], Fan and Vercauteren (FV) is the most promising method for an practical homomorphic encryption.

3.2 Notations

The main mathematical object used in the implementation of our approach is the polynomial. In the Table 2, we present some mathematical notations that we used during the implementation of our approach [30-33].

3.3 RLWE

Ring Learning With Error or Learning With Error over Ring commonly referred as RLWE is a generalization of the LWE problem, it was introduced in [34] and constitutes the core of the new homomorphic encryption algorithms. It has been proved that fully homomorphic encryption is efficient with the use of RLWE [35-40]. The ring-LWE problem [21,41] consists of polynomial tuples $(a_i\,(x), b_i\,(x))$, where the coefficients of a_i are chosen uniformly from R_q and $b_i\,(x)$ is calculated as:

$$b_i\,(x) = a_i\,(x) \cdot s_i\,(x) + e_i\,(x) \in R_q \qquad (1)$$

with: $s \in R_q$ is secret polynomial.

3.4 FV scheme

FV named after its founder Junfeng Fan and Frederik Vercauteren is a public key cryptosystem encryption model appeared in 2012 which is based on the RLWE principle [30]. The FV model as a fully homomorphic scheme is based on different phases as Setting, Generate Key, Encryption, Decryption, Addition, Multiplication and Relinearisation. As far as our approach is concerned, we have been concentrating only on addition operation and the framework was implemented without bootstrapping step.

In the following, we will give a detailed description of the different phases:

3.4.1. Setting

This step consists of defining the set of parameters: λ, n, q, t, σ.

Table 2. List of mathematical notations

Parameter	Description
R	The polynomial ring $\mathbb{Z}[x] / (f(x))$
$f(x) = x^n + 1$	Cyclotomic polynomial of degree n
$\mathbb{Z}[x]$	The polynomial ring with coefficients in $\mathbb{Z}$
n	Degree of the cyclotomic polynomials, it is a power of 2
q	Modulus in the ciphertext space (coefficient modulus), it is a positive prime number
t	Modulus in the plaintext space (plaintext modulus).
$\mathbb{Z}_q$	A set of integers ranging from $\left(-q/2, q/2\right]$
R_q	A set of polynomials in R with coefficients in $\mathbb{Z}_q$
$[a]_q$	$a \bmod q$ in $\left]-q/2, q/2\right]$
$a = \sum_{i=0}^{n-1} a_i \cdot x_i$	The elements of the polynomial ring R
$\lfloor a \rceil$	The rounding of a to the nearest integer smaller than a
$\lfloor a \rceil$	The rounding of a to the nearest integer closet to a
λ	Security parameter (the size of the resulting public key).
$P(x) = e^{\left(\frac{-\pi\lvert x\rvert^2}{\sigma^2}\right)}$	Gaussian function discrete centered in zero.
χ	Discrete Gaussian Distribution
χ_{key}	Uniform and independent coefficients drawn from a narrow set $\{-1,0,1\}$.
χ_{err}	Coefficients drawn independently according to a Gaussian function discrete centred in zero
β	Bound of the distribution χ
σ	Standard deviation of χ
e_i	Error polynomial sampled from χ
Δ	Quotient on Euclidean division of q by t
m	Plaintext element ($m \in R_t$)
μ	Normalization factor

3.4.2. Key generation (keygen)

This phase of FV consists of finding:

The secret key sk, that is represented in the form of a polynomial which coefficients are drawn from χ_{key}.

$$sk = s \quad (2)$$

The public key pk represented as a pair of two polynomials $(p0, p1)$ (sample RLWE) [31] such as:

$$pk = (p0, p1) = ([-(a.s + e)]_q, a) \quad (3)$$

With: $a \leftarrow R_q, e \leftarrow \chi_{err}$

3.4.3. Encryption

Consists of encoding a message m for some positive integer t ($t \geq 2$) by calculating a ciphertext c formed of two polynomials $c0$ and $c1$ [42] represented as follows:

$$c0 = [\ \Delta \cdot m + p0 \cdot u + e_1\ ]_q \quad \in R_q \quad (4)$$

$$c1 = [\ p1 \cdot u + e_2\ ]_q \quad \in R_q \quad (5)$$

Knowing that:

$$\Delta = \lfloor q / t \rfloor \quad (6)$$

$$u \leftarrow \chi_{key} \ ; \quad e_1, e_2 \leftarrow (\chi_{err})^2$$

In our approach, we represent each plaintext in a binary form, so each position is encoded as a polynomial by using the binary encoding. Each plaintext must have his one ciphertext independently to others.

3.4.4. Decryption

Each ciphertext is decrypted by using a secret key. Starting from a ciphertext, decrypt it in message m consists on calculating [42]:

$$c0 + c1.s = \Delta \cdot m + p0 \cdot u + e_1 + (p1 \cdot u + e_2).s \quad (7)$$

$$c0 + c1.s = \Delta \cdot m + (-(a.s + e) + a.s).u + e_1 + s.e_2 \quad (8)$$

$$m = \lfloor (c0 + c1 \cdot s) \times t / q \rceil \ \in R_t \quad (9)$$

m is obtained after performing a scale and round. It should be noted that in the FV model the scalar product is of major importance. We used ROUND_HALF_EVEN java function to ensure the rounding to the nearest integer in the final step of computation.

3.4.5. Addition

The operation of adding two encrypted messages *m*1 and *m*2 is performed by computing: $ADD\ (c_0, c_1)$, knowing that:

$c_0 = \{c_{0,1}, c_{1,1}\}$ and $c_1 = \{c_{0,2}, c_{1,2}\}$ therefore [42]:

$$c_{add} = \{c_{0,1} + c_{0,2}, c_{1,1} + c_{1,2}\} \quad (10)$$

The result represents a ciphertext of *m1+m2.*

4. Proposed approach

In the FV model, there are several Discrete Gaussian Sampling algorithms that can be used to generate the error polynomials coefficients, the most commonly are: Cumulative Distribution Function (CDF), Rejection Sampler, Knuth-Yao Random Walk Method [37] and many other models cited in [41].

Since the Gaussian generation algorithm is one of the main operations of the FV model [40], the choice being focused on the Knuth-Yao Random Walk algorithm because of its performance and efficiency proven on different platform.

Knuth-Yao Random Walk Method was introduced on 1976 by Donald Ervin Knuth and Andrew Chi-Chih Yao in [43] who proposed an algorithm to sample values from the non-uniform distribution. The model consists of a sampling through using a set of input bits as close as possible to the value generated by non-uniform distribution entropy.

The Gaussian distribution $\mathcal{D}_{\mathbb{Z},\sigma}$ over $\mathbb{Z}$ with mean equal to zero and $\sigma > 0$ a standard deviation is represented as:

$$Pr(E = Z) = \frac{1}{\mu}\, e^{-z^2/2\sigma^2} \quad (11)$$

where

$$\mu = 1 + 2\sum_{z=1}^{\infty} e^{-z^2/2\sigma^2} \quad (12)$$

The value of μ depends to the standard deviation σ. It has been proved during the experiments and also in the literature [33] that the value of μ corresponds approximately to:

$$\mu \ \cong \ \sigma\sqrt{2\pi} \quad (13)$$

Starting from the Z (which depends to the tail bound value used), we calculate the probabilities P_i by the distribution $Pr(E = Z)$ and convert them

on binary expansion with a finite precision using IEEE 754 quadruple-precision binary floating-point format.

The binary representation of each of the calculated probabilities is written in the *Pmat* matrix consisting of a set of rows (the row number is based on the Tail bound value) and a set of columns (according to the value of precision bound). The storage size that is reserved is equivalent to the number of rows multiplied by the number of columns which causes a large consumption of the memory.

The last step is to construct the Discrete Distribution Generating Tree (DDG) corresponding to *Pmat*, using Algorithm 1. However, it is not mandatory to represent the algorithm using a tree; we can store the binary representations of P_i in a table [44].

The parameters used for FV with Gaussian distribution and Knuth-Yao distribution methods are:

General values chosen for both approaches:

($\lambda = 32\ bits, n = 32, q = 221, t = 2, \sigma = 3.2$).

this value of σ is commonly used in practice by many libraries, also because no attacks are known with [45].

Knuth-Yao parameters chosen to generate Pmat matrix probabilities are:

($Tail\ Bound = 39, Precision\ Bound = 107, \mu = 8.01$).

The value of the Discrete Gaussian Distribution bound is represented as:

$$\beta = 10 \cdot \sigma \quad (14)$$

5. Implementation & Experiments

The implementation of our approach as described above was carried out from scratch in Java Netbeans on a Lenovo Thinkpad T470 with 24 GB RAM using an Intel Core i5-6300U- 2.5GHZ CPU with hyper-threading disabling. The application allows loading a BAM file from a local media or connecting directly to a MySQL database corresponding to the BAM file in question. This permits a very fast loading of the information contained in the BAM file especially when the input file is large which is the case in general. The BAM file is sorted by chromosomes (from 1 to 24) with all the parameters.

To carry out this work, various data sources have been used in our implementation, for example Colorectal and Breast Cancer Data, downloadable from [46]. The BWA has been used as a tool for mapping and generating BAM files on a Linux server. However, for comparison with the SECRAM work as presented in [25] we opted for the NA12878 database (Table 3 shows a sample of dataset used) which is publicly available. The size of the file is about 400 Mo (equivalent approximatively to 2 million positions to encrypt for this file). To explore the BAM file, we used an HTSJDK API (unified Java library) to read all information. Technically, we created a SamReader for a BAM file and then iterated all BAM records with the SamReader. The position we focused on was retrieved from each BAM record with its Java API. Therefore, all the genomic positions of the BAM file are coded in binary with a binary encoding scheme and represented as polynomials of degrees defined during the setting phase of the FV scheme.

To speed up processing time for encryption and decryption operations, 24 threads were used (one thread per chromosome).

Our application allows also adding two ciphertexts for the need to retrieve information about positions when querying genomic data. Table 4 presents the results obtained for keyGen and addition of two ciphertexts and also the encryption and decryption operations applied on 2 million positions before optimizing the process with the HashMap representation.

The obtained results show that FV using Knuth-Yao Distribution is better than FV using a simple Gaussian Distribution in terms of computation time with zero loss of information at decryption operation. It should be noted that at each computation step (polynomial sum, polynomial product) a procedure of elimination of null coefficients was applied to speed up the calculations.

For the Knuth-Yao method, the coefficients of the polynomials e, $e1$, $e2$ are binary. The very small coefficients generated by the Gaussian probabilistic function used in FV-Gauss allow having precision but increase the computation time of key generation, encryption and decryption which are based mainly on polynomial sum and product.

The security level of an homomorphic scheme depends to the λ security parameter and strongly on the values of (n, q and σ). When n is large, the attacks become difficult but this will have a fundamental impact on the slowness of the scheme and will require adequate hardware resources to achieve the encryption and decryption operations. There is no determination procedure of the exact values to be taken for these parameters.

Many research works are in progress to find a good compromise between the safety and efficiency of an homomorphic scheme based essentially on the RLWE model. Ensuring an efficiency-safety-functionality compromise is an extremely delicate

Input: p_{mat} , random bit r
Output: Sample value μ

```
1:  begin
2:  d ← 0 ; /* Distance between the visited and the rightmost internal node */
3:  Hit← 0 ; /* This is 1 when the sampling process hits a terminal node */
4:  col ← 0 /* Column number of the probability matrix */
5:  while Hit=0 do
6:          r← RandomBit();
7:        d ← 2d + r
8:        for row from MAXROW by -1 to 0 do
9:              d ← d − p_mat[row][col]
10:             if d = −1 then
11:                 μ ← row;
12:                 Hit← 1
13:                 ExitForLoop();
14:           end for
15:   col ← col + 1
16: return μ;
17: end
```

Algorithm 1. Knuth-Yao Algorithm

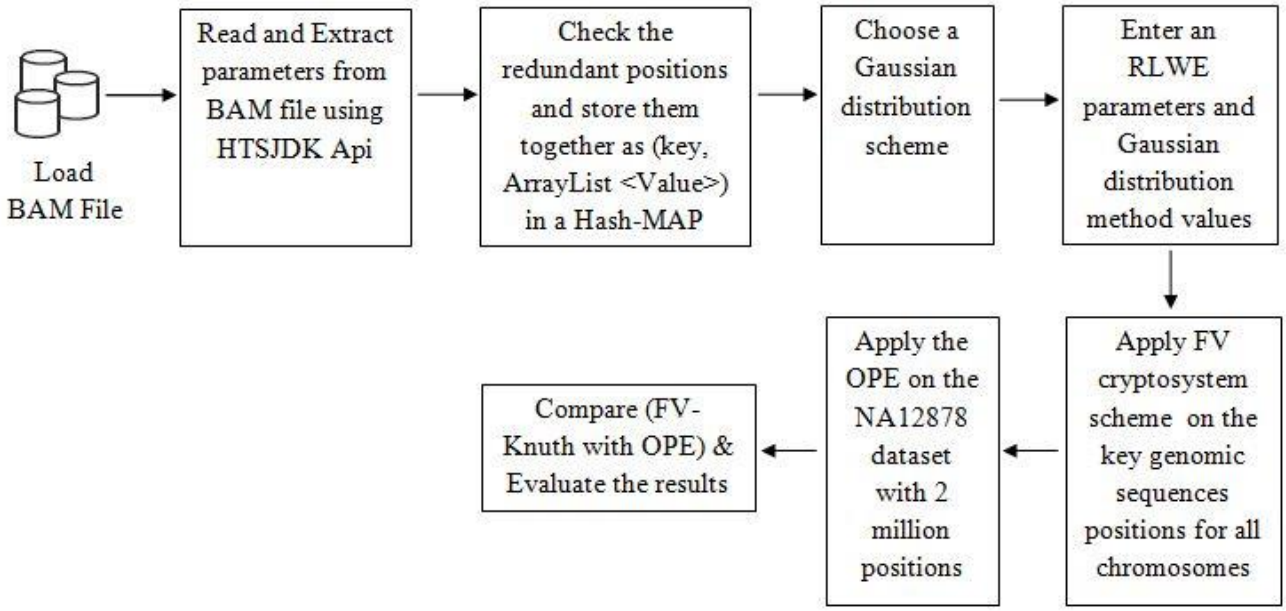

Figure. 2 Flowchart of the experimental process

research problem and could not be fully respected in our case given the material resources used. However, we were able to approve the efficiency and reliability of the FV model with the Simple Gaussian Distribution and Knuth-Yao scheme to generate the coefficients of the polynomials *e*, *e*1 and *e*2. Moreover, the use of the Knuth-Yao algorithm allows a faster execution time for all the steps of the FV model that have been implemented.

As shown in Tables 4, 5, 6 and 7, the FV model based on Knuth-Yao as a Discrete Gaussian sampling algorithm offers better results in terms of execution time. In order to increase the security of our scheme while optimizing storage and computation time with FV, we implemented an architecture based on a HashMap, this representation allowed us to encrypt only the keys that represent the genomic positions presented in our BAM input file.We were able to

Table 3. NA12878 sample dataset BAM file

Num	Read	Position	Start	End	Reference	Read Lenght	Cigar
1932907	TCCATTCCA...	58978480	58978480	58978512	Y	33	36H33M231H
1932908	GCAATAAAT...	58980553	58980553	58980782	Y	299	64S88M5I142M
1932909	CTACTCCCC...	58980635	58980635	58980767	Y	301	51S106M5I27...
1932910	CACTCCATT...	58982016	58982016	58982080	Y	65	58H65M39H
1932911	CACTCCATT...	58982016	58982016	58982080	Y	65	58H65M39H
1932912	GTCAAGCTA...	59008094	59008094	59008335	Y	242	242M
1932913	GTCAAGCTA...	59008094	59008094	59008335	Y	242	242M
1932914	TCTAAAATT...	59018476	59018476	59018757	Y	282	282M
1932915	TCTAAAATT...	59018476	59018476	59018757	Y	282	282M
1932916	AAAAAAAA...	59022279	59022279	59022327	Y	49	109H49M142H
1932917	AAAAAAAA...	59022281	59022281	59022327	Y	47	111H47M143H

Table 4. FV overall experimentations results using Gaussian and Knuth-Yao distributions without hashMap representation

	KeyGen	**Encryption**	**Decryption**	**Addition**
FV - Gauss	0.010s	945.75s	1261.69s	0.0017s
FV - Knuth.Yao	0.003s	287.02s	975.89s	0.0011s

Table 5. FV overall experimentations results using Gaussian and Knuth-Yao distributions with hashMap representation

	Encryption		**Decryption**	
	***n* =32**	***n* =64**	***n*=32**	***n* =64**
FV - Gauss	62.81s	512.06s	225.97s	3653.29s
FV - Knuth.Yao	35.16s	390.96s	216.34s	3610.19s

reduce the number of positions from 2 million to 165667. Tables 6 and 7 present the number of positions to be encrypted and decrypted with their respective durations estimated in second. The HashMap representation allowed us also to carry out the computation with a polynomial degree n=64 and λ security parameter equal to 64 bits (table 5 describes the overall experimental results obtained).

Notations used in the Tables 6, 7, and 8:

- Chr: Chromosome
- PEBHM: Positions to encrypt before HashMap.
- FVG: Fv with Gaussian sampling Algorithm
- FVKY: Fv with Knuth-Yao sampling Algorithm.
- PEAHM: Positions to encrypt after HashMap.
- PDBHM: Positions to decrypt before HashMap.
- PDAHM: Positions to decrypt after HashMap.Chr: Chromosome
- Pos: Position to encrypt
- ED: Encryption Duration time
- DD: Decryption Duration Time

6. Comparison study

In order to demonstrate the efficiency and robustness of our proposed method, we have compared the FV-Knuth-Yao scheme with the OPE method as implemented in an open source SECRAM library. The findings have showed that the OPE is faster than FV (see Tables 8 and 9) in terms of encryption and decryption execution time. For n=32, $\lambda = 32$, FV-Gauss is three times longer than OPE for encryption and 25 times longer for decryption operation. FV-Knuth-Yao is approximately two times longer than OPE for the encryption and 24 times longer for decryption operation. The speed of treatment with OPE is due to the non-complexity of the computational operations unlike FV where the whole model is based on a complex polynomial structure.

Since the complexity and the processing time are among the evaluation metrics of a cryptographic system, we can deduce that the FV model is stronger against intruder attacks than OPE particularly for sensitive information like genomic positions, which is the ultimate objective of our research work.

Table 6. Detailed FV encryption duration time (in seconds) for each chromosome

Chr	PEBHM	FVG	FVKY	PEAHM	FVG	FVKY
1	**32025**	945.752	287.025	**10534**	59.191	31.25
2	**157971**	2709.48	865.912	**17414**	62.81	34.67
3	**19024**	649.646	235.146	**6598**	54.44	28.2
4	**215415**	3119.34	998.016	**15341**	62.01	34.06
5	**174955**	2834.45	922.795	**13798**	61.175	33.41
6	**16914**	562.399	219.252	**5883**	53.378	26.96
7	**217558**	3120.19	998.569	**9782**	58.086	31.76
8	**54462**	1358.04	422.303	**6450**	54.66	27.69
9	**33474**	971.96	336.247	**6466**	54.049	28
10	**41724**	1145.94	352.829	**6466**	54.989	28.95
11	**78208**	1704.84	523.776	**6224**	52.952	27.63
12	**15186**	515.776	157.996	**5452**	23.706	26.02
13	**133305**	2418.13	743.211	**13126**	32.991	32.76
14	**48088**	1260.22	383.533	**4263**	20.225	24.16
15	**173211**	2833.46	865.272	**9097**	30.038	30.25
16	**10370**	371.457	159.166	**5016**	21.867	25.31
17	**218726**	3130.46	953.372	**6680**	26.258	28.57
18	**41773**	1135.11	351.429	**5412**	23.469	25.82
19	**218405**	3119.94	953.127	**4437**	20.498	23.74
20	**12706**	439.402	187.898	**3568**	17.615	21.66
21	**2781**	105.515	83.677	**1264**	9.534	16.06
22	**4809**	172.95	65.456	**2405**	13.907	18.5
X	**11666**	406.526	172.838	**5336**	23.569	26.08
Y	**161**	0.803	46.407	**107**	0.545	0.28

Table 7. Detailed FV decryption duration time (in seconds) for each chromosome

Chr	PDBHM	FVG	FVKY	PDAHM	FVG	FVKY
1	**32025**	1261.69	275.89	**10534**	191.64	186.9
2	**157971**	3666.79	829.28	**17414**	225.54	215.94
3	**19024**	819.664	241.51	**6598**	157.88	153.61
4	**215415**	4244.97	990.60	**15341**	216.83	209.22
5	**174955**	3889.35	987.46	**13798**	209.94	203.69
6	**16914**	741.147	214.20	**5883**	146.1	142.4
7	**217558**	4256.17	991.94	**9782**	187.51	184.88
8	**54462**	1834.53	461.93	**6450**	156.51	157.4
9	**33474**	1302.89	432.18	**6466**	156.29	152.61
10	**41724**	1546.60	445.70	**6466**	166.54	162.41
11	**78208**	2313.76	501.70	**6224**	153.11	151.5
12	**15186**	677.161	202.15	**5452**	139.94	139.24
13	**133305**	3290.97	601.93	**13126**	205.99	201.52
14	**48088**	1704.13	461.65	**4263**	114.67	109.90
15	**173211**	3881.50	883.14	**9097**	182.18	177.6
16	**10370**	483.70	125.30	**5016**	130.24	125.3
17	**218726**	4269.67	924.46	**6680**	159.52	154.8
18	**41773**	1545.11	378.47	**5412**	138.84	137.9
19	**218405**	4259.18	972.05	**4437**	117.97	113.1
20	**12706**	581.769	195.35	**3568**	103.65	97.25
21	**2781**	134.08	96.48	**1264**	34.614	34.09
22	**4809**	229.323	63.71	**2405**	65.789	62.42
X	**11666**	542.932	206.58	**5336**	138.69	133.01
Y	**161**	4.94	6.25	**107**	2.83	3.90

Table 8. Detailed OPE encryption duration time (in seconds) for each chromosome

Chr	Pos	ED	DD
1	**32025**	1.528	0.215
2	**157971**	1.557	0.679
3	**19024**	0.506	0.116
4	**215415**	1.703	1.066
5	**174955**	2.636	0.769
6	**16914**	0.424	0.096
7	**217558**	1.363	0.988
8	**54462**	0.629	0.271
9	**33474**	0.495	0.169
10	**41724**	0.570	0.197
11	**78208**	0.750	0.358
12	**15186**	0.405	0.081
13	**133305**	1.036	0.605
14	**48088**	0.504	0.202
15	**173211**	1.241	1.214
16	**10370**	0.364	0.051
17	**218726**	1.926	0.920
18	**41773**	0.415	0.173
19	**218405**	1.225	0.845
20	**12706**	0.250	0.057
21	**2781**	0.095	0.013
22	**4809**	0.175	0.023
X	**11666**	0.391	0.053
Y	**161**	0.0087	0.0008

Table 9. OPE overall execution time

	Encryption	Decryption
OPE	20.19s	9.16s

7. Conclusion

The originality of this work is based on the use of Knuth-Yao algorithm and a representation of BAM information in a HashMap container structure without redundancy positions. This technique allows speeding up the execution time of the encryption and decryption operations, accelerates the access and the reading of the BAM elements and will facilitate the indexing for the possible future operations of search and query of the desired information.

Compared to OPE method as implemented in SECRAM tool, the gap in terms of execution time for encryption and decryption operations with FV method remains large, especially by increasing the degree of polynomial. However the security cannot be ensured because OPE is vulnerable to running attacks.

We note that for n=32, the estimated time to encrypt 165667 positions via FV-Gauss is approximately two times longer than encrypting the same data with FV-Knuth-Yao. The decryption operation for both FV-Gauss & FV-Knuth-Yao is done in less than 4 minutes. For n=64, it takes 8.5

minutes for FV-Gauss and 6.5 minutes for FV-Knuth-Yao to encrypt the same quantity of data and approximatively one hour to perform the decryption operation by both algorithms.

For FV security, which is the ultimate purpose of our research work, whether by Gauss or Knuth-Yao sampling algorithms, attacks require 2^{32} operations (with $\lambda = 32$ bits) and 2^{64} operations ($\lambda = 64$ bits) to break our cryptosystem. The authors are currently working on the encryption for the non integer element as Cigar to extend the security of all sensitive data in a BAM file.

The authors recommend the deployment of the proposed model on an Hyperconverged Infrastructure Platform with considerable computing capacity to ensure product and sum polynomial operations with very large polynomial degrees (1024, 2048, or 4096). The use of secure Dockers containers would also enhance the security level of the proposed model.

Conflicts of Interest

The authors declare no conflict of interest.

Author Contributions

Conceptualization: Moufida Achour and Abderrahim Belmadani; methodology: Moufida Achour; software: Moufida Achour; validation: Moufida Achour and Abderrahim Belmadani; formal analysis: Moufida Achour and Abderrahim Belmadani; Investigation: Moufida Achour; resources: Moufida Achour; data curation: Moufida Achour; writing—original draft preparation: Moufida Achour; writing—review and editing: Moufida Achour and Abderrahim Belmadani; visualization : Moufida Achour; supervision : Abderrahim Belmadani; project administration : Abderrahim Belmadani; no funding acquisition.

Acknowledgments

The authors thank the following people for their valuable assistance:

Zhicong Huang for answering questions that Moufida Achour had on the SECRAM tool. The Professor Therese Commes for giving Moufida Achour the opportunity to benefit of one month internship within her "Bioinformatics and Biomarkers" research team at the Institute for Regenerative Medicine and Biotherapy (IRMB) INSERM (France), allowing her to acquire knowledges in biology and bioinformatics. Jerome Audoux and the whole SEQONE's team for their precious advice and availability.

References

[1] G. Gibson, S. V. Muse, Précis de Génomique, De Boeck, 2004.

[2] N. A. Fonseca, J. Rung, A. Brazma, and J. C. Marioni, "Tools for mapping high-throughput sequencing data", *Bioinformatics*, Vol. 28, No. 24, pp. 3169-3177, 2012.

[3] N. Philippe, "Developpement de methodes et d'algorithmes pour la caracterisation et l'annotation des transcriptomes avec les séquenceurs haut débit", *MONTPELLIER II University*, PHD Thesis, 2011.

[4] M. Kanzi, J. E. SAN, B. Chimukangara, E. Wilikinson, M. Fish, V. Ramsuran and T. Oliveira, "Next Generation Sequencing and Bioinformatics Analysis of Family Genetic Inheritance", *Font Genet*, Vol. 11, No. October, pp. 1-18, 2020.

[5] S. Zhao, O. Agafonov, A. Azab, T. Stokowy, and E. Hovig, "Accuracy and Efficiency of Germline Variant Calling Piplines for Human Genomic Data", *Sci Rep*, Vol. 10, No. 1, pp. 1-12, 2020.

[6] Talwalkar, J. Liptrap, J. Newcomb, C. Hartl, J. Terhorst, K. Curtis, M. Bresler, Y. S. Song, M. I. Jordan, and D. Patterson, "SMaSH: A Benchmarking Toolkit for Human Genome Variant Calling", *Bioinformatics*, Vol. 30, No. 19, pp. 2787-2795, 2014.

[7] D. Warden, A. W. Adamson, S. L. Neuhausen, and X. Wu, "Detailed Comparison of Two Popular Variant Calling Packages for Exome and Target Exon Studies", *PeerJ*, Vol. 2, p. e600, 2014.

[8] W. McLaren, L. Gill, S. E. Hunt, H. S. Riat, G. R. S. Ritchie, A. Thormann, P. Flicek and F. Cunningham, "The Ensembl Variant Effect Predictor", *Genome Biology*, Vol. 17, No. 1, pp. 1-14, 2016.

[9] 1000 Genome Projet Data and P.Subgroup.Https://github.com/samtools/hts-specs.

[10] The SAM/BAM Format Specification Working Group. [Online]. Https://samtools.github.io/hts-specs/SAMv1.PDF.

[11] V. Chialva and A. Dooms, "Conditionals in Homomorphic Encryption, and Machine Learning Applications", *arXiv*, pp. 1-16, 2019.

[12] M. Blatt, A. Gusev, Y. Polyakov, and S. Goldwasser, "Secure Large-Scale Genome-Wide Association Studies Using Homomorphic Encryption", In: *Proc. of the National Academy of Sciences*, U.S.A, pp. 1-38, 2020.

[13] J. Buchmann, M. Geihs, K. Hamacher, S. Katzenbeisser, and S. Stammler, "Long-Term Integrity Protection of Genomic Data", *EURASIP Journal on Information Security*, Vol . 2019, No. 1, 2019.

[14] S. Halevi and S. Micali, "Practical and Provably-Secure Commitment Schemes from Collision-Free Hashing", In: *Proc. of the 16h Annual International Cryptology Conference on Advances in Cryptology*, Santa Barbara, California, U.S.A, pp. 201-215, 1996.

[15] K. Lenstra and E. R. Verheul, "Selecting Cryptographic Key Sizes", *Journal of Cryptology*, Vol. 14, No. 14, pp. 446-465, 2000.

[16] T. P. Zhou, N. B. Li, X. Y. Yang, L. Q. Lv, Y. T. Ding, and X. A. Wang, "Secure Testing for Genetic Dieases on Encrypted Genomes with Homomorphic Encryption Schemes", *Hindawi Security and Communication Networks*, Vol. 2018, 2018.

[17] M. Kim, Y. Song, and J. H. Cheon, "Secure Searching of Biomarkers Through Hybrid Homomorphic Encryption Scheme", *BMC Medical Genomics*, Vol. 10, Spplement 2, pp. 102-118, 2017.

[18] J. L. Raisaro, G. Choi, S. Pradervand, R. Colsenet, N. Jacquemont, N. Rosat, V. Mooser, and J. P. Hubeaux, "Protecting Privacy and Security of Genomic Data in i2b2", *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, Vol. 15, No. 5, pp. 1-17, 2018.

[19] G. S. Çetin, H. Chen, K. Laine, K. Lauter, P. Rindal, and Y. Xia, "Private Queries on Encrypted Genomic Data", *BMC Medical Genomics*, Vol. 10, Spplement 2, pp. 1-14, 2017.

[20] Ugwuoke, Z. Erkin, and R. L. Lagendijk, "Privacy-Safe Linkage Analysis With Homomorphic Encryption", In: *Proc. of the 25th European Signal Processing Conference*, Kos Island, Greece, pp. 961-965, 2017.

[21] K. Lauter, A. Lopez-Alt, and M. Naehrig, "Private Computation on Encrypted Genomic Data", In: *Proc. of International Conference on Cryptology and Information Security*, Latin America, pp. 3-27, 2015.

[22] F. Montecuollo, G. Schmid, and R. Tagliaferri, "E2FM: An Encrypted and Compressed Full-Text Index for Collections of Genomic Sequences", *Bioinformatics*, Vol. 33, No. 18, pp. 2808-2817, 2017.

[23] M. Z. Hasan, "Private Computation on Genomic Data", *The University of Manitoba, Master Thesis*, 2017.

[24] K. Shimizu, K. Nuida, and G. Rätsch, "Efficient Privacy-Preserving String Search and an Application in Genomics", *Bioinformatics*, Vol. 32, No. 11, pp. 1652-1661, 2016.

[25] Z. Huang, E. Ayday, H. Lin, R. S. Aiyar, A. Molyneaux, Z. Xu, J. Fellay, L. M. Steinmetz, J. P. Hubeaux, "A Privacy- Preserving Solution for Compressed Storage and Selective Retrieval of Genomic Data", *Genome Research*, Vol. 26, No. 12, pp. 1687-1696, 2016.

[26] P. M. Fauzi, "On Fully Homomorphic Encryption", *Tartu University, Master Thesis*, 2012.

[27] Gentry, "Fully Homomorphic Encryption Using Ideal Laticies", In: *Proc. of 41st Annual ACM Symposium on Theory of Computing*, Bethesda MD, U.S.A, p. 169, 2009.

[28] Z. Brakerski and V. Vaikuntanathan, "Efficient Fully Homomorphic Encryption from (standard) LWE", In: *Proc. of 52nd Annual IEEE Symposium on Foundations of Computer Science*, NW Washington DC, U.S.A, pp. 97-106, 2011.

[29] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, "Fully Homomorphic Encryption Without Bootstrapping", In: *Proc. of the 3rd Innovations in Theoretical Computer Science Conference*, Cambridge Massachusetts, pp. 309–325, 2012.

[30] J. Fan and F. Vercauteren, "Somewhat Practical Fully Homomorphic Encryption", In: *Proc. of the 15th International Conference on Practice and Theory in Public Key Cryptography*, Darmstadt, Germany, pp. 1–16, 2012.

[31] J. C. Bajard, J. Eynard, M. A. Hasan, and V. Zucca, "A Full RNS Variant of FV Like Somewhat Homomorphic Encryption Schemes", In: *Proc. of the 24th International Conference on Selected Areas in Cryptography*, Ottawa, Canada , pp. 423–442, 2017.

[32] V. Zucca, Représentation de Nombres pour les Algorithmes Contexte, Sorbonne Université, Paris, 2017.

[33] Karmakar, S. S. Roy, O. Reparaz, F. Vercauteren, and I. Verbauwhede, "Constant-Time Discrete Gaussian Sampling", *IEEE Transactions on Computers*, Vol. 67, No. 11, pp. 1561–1571, 2018.

[34] V. Lyubashevsky, C. Peikert, and O. Regev, "On Ideal Lattices and Learning with Errors Over Rings", *ACM Journals*, Vol. 60, No. 6, 2013.

[35] J. W. Bos, K. Lauter, J. Loftus, and M. Naehrig, "Improved Security for a Rring-Based Fully Homomorphic Encryption Scheme", In: *Proc. of the 14th IMA International Conference on Cryptography and Coding*, Oxford, UK, pp. 45–64, 2013.

[36] Z. Brakerski, "Fully Homomorphic Encryption Without Modulus Switching from Classical GapSVP", In: *Proc. of the 32nd Annual Cryptology Conference*, Santa Barbara, California, U.S.A, pp. 868–886, 2012.

[37] J. S. Coron, T. Lepoint, and M. Tibouchi, "Scale-Invariant Fully Homomorphic Encryption Over The Integers", In: *Proc. of the 17th International Conference On Practice and Theory in Public-Key Cryptography*, Buenos Aires, Argentina, pp. 311-328, 2014.

[38] M. V. Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan, "Fully Homomorphic Encryption Over The Integers", In: *Proc. of the 29th Annual International Conference on The Theory and Applications of Cryptographic Techniques*, French Riviera, France, pp. 24-43, 2010.

[39] Gentry, S. Halevi, and N. P. Smart, "Homomorphic Evaluation of The AES Circuit", In: *Proc. of the 32nd Annual Cryptology Conference*, Santa Barbara, California, U.S.A, pp. 850-867, 2012.

[40] C. Gentry, A. Sahai, and B. Waters, "Homomorphic Encryption from Learning With Error: Conceptually-Simpler, Asymptotically-Faster, Attribute-Based", In: *Proc. of the 33nd Annual Cryptology Conference*, Santa Barbara, California, U.S.A, pp. 75-92, 2013.

[41] S. S. Roy, F. Vercauteren, J. Vliegen, and L. Verbauwhede, "Hardware Assisted Fully Homomorphic Function Evaluation and ENcrypted Search", *IEEE Transactions on Computers*, Vol. 66, No. 9, pp. 1562-1572, 2017.

[42] G. Bonnoron, "A journey Towards Practical Fully Homomorphic Encryption", *L'Ecole Nationale Supérieure Mines-Telecom Atlantique, Comue Université Bretagne Loire, PHD Thesis*, 2019.

[43] Knuth and A. C. Yao, "The Complexity of Non-Uniform Random Number Generation", In: *Proc. of A Symposium on New Directions and Recent Results in Algorithms and Complexity*, Pittsburgh, Pennsylvania, U.S.A pp. 357–428, 1976.

[44] N. C Dwarakanath, and S. D. Galbraith, "Sampling From Discrete Gaussians for Lattice-Based Cryptography on A Constrained Device", *Applicable Algebra in Engineering, Communications and Computing*, Vol. 25, No. 3, pp. 159–180, 2014.

[45] Homomorphic Encryption Standardisation [Online]. Http://homomorphicencryption.org/wp-content/uploads/2018/11/HomomorphicEncryptionStandardv1.1.pdf

[46] European Nucleotide Archive. [Online].Http://www.ebi.ac.uk/ena/data/view/SRR1057062&display=html