Privacy Preserving Fisher's Exact Test on Genomic Data

Anna Poon
Computer Science
California State University, Bakersfield
Bakersfield, CA, United States
apoon1@csub.edu

Steve Jankly
Computer Science
California State Polytechnic University
Pomona, CA, United States
spjankly@cpp.edu

Tingting Chen
Computer Science
California State Polytechnic University
Pomona, CA, United States
tingtingchen@cpp.edu

***Abstract*—Privacy of genomic data has become increasingly significant as genome sequencing is more readily available for research. It is imperative to protect genomic data as we use it for the progression of medicine. In this paper, we propose a new privacy preserving Fisher's Exact Test algorithm for genomic data based on Boneh-Goh-Nissim (BGN) cryptosystem. This is a novel approach that has yet to be done to the best of our knowledge. Due to BGN's homomorphic properties, researchers can keep data private while calculating the correct test results without ever decrypting the data itself. We investigate the usage of the BGN cryptosystem on statistical computations, Fisher's Exact Test in particular, analyzing its security, efficiency, and correctness in the real world of genomic data research. We implement our BGN-based privacy preserving Fisher's Exact Test algorithm and test it extensively using real genomic data from international genome database. The result shows that our algorithm is efficient and practical.**

I. Introduction

From the Human Genome Project [17] to the current day, the Genome Wide Association Study (GWAS) has found revolutionizing discoveries about the human body. The study of phenotypes and genotypes has grown rapidly due to the dramatic decrease in the cost of sequencing genes. This increase in gene sequencing has pushed for research in many fields such as personalized medicine, genetic engineering, epidemiology, etc. Many initiatives, such as Genome Data Sharing and Personal Genome Project, have been spearheaded in this revolution [13]. Using genomic data we are able to see the linkage between genes and their predictions to hereditary traits. Certain single-nucleotide polymorphisms (SNPs), which are micro-array sequences, can be common variants of an attribute. When these SNPs are identified from our genome, we can determine our susceptibility to certain features. These discoveries have empowered researchers to study personalized medicine and preventive health care. Meaningful genetic analysis is critically valuable for the progression of medical advancements.

Conversely, the exposure of one's genomic data can result in high privacy-risks. An individual's genome can imply sensitive and personal information such as the existence or the probability of developing a certain disease. Exposure of this information to an unauthorized party is to the disadvantage of the person concerned and sometimes may be life-threatening [1] [17].

This work was supported in part by NSF grant CNS-1758017.

On the other hand, protecting genomic data privacy has its unique challenges as this data has distinct features that make this a more sensitive and dire issue. First of all, genetic data is permanent and identifiable to a specific individual, thus any risk allows a lifetime of vulnerability for the person exposed. Therefore, cryptographic approaches in general are facing the challenge from brute force attackers with longer time to possibly crack the ciphertext. Secondly, our genome is linked to relatives, causing domino effect of risk to them as well. That is, one's blood relatives' genome sequence information can also be inferred even though their data is not directly exposed. The association and linkage among genome data from different people thus need to be considered when addressing the privacy issue using a non-cryptographic approach [1]. Lastly, traditional anonymization methods that remove personally identifiable information are not as effective either, since genome itself is the ultimate identifier. In short, as the genomic data becomes more easily processed, it becomes more important and yet challenging to keep this data private for individuals and institutions [1].

In this work, we propose a new privacy preserving algorithm to perform statistical analysis (Fisher Exact Test in particular) on individuals' genome data in a cloud. Due to the huge volume of genome data, the cloud is a practical platform to store and analyze such data. However, although the cloud platform promises easy access for genomic data sharing, it cannot always be assumed secure and trustworthy. As a matter of fact, the Cloud Security Alliance has highlighted several security vulnerabilities including malicious insiders [3]. As to defend against possible adversarial penetration into the cloud, our algorithm only stores ciphertext of genome data in the cloud to ensure data protection. In this paper, we assume that the cloud service is semi-honest.

Fisher's Exact Test is a classic statistical test in enrichment analysis of genes, and often used as a cross-validation scheme to ensure the statistically stability in genetic analysis. It has been widely applied in various biomedical research work, such as detecting differentially expressed genes [5]. Providing a privacy preserving Fisher's Exact Test will enable more individuals to participate in this biomedical research, and protect their personal privacy at the same time.

Our goal in this paper is to design a cryptographic algorithm to enable individual users to encrypt their genomic data, and send the ciphertext to the cloud. On the cloud, Fisher's Exact Test can be correctly performed on the ciphertexts. Utilizing

978-1-5386-5035-6/18/$31.00 ©2018 IEEE

this method, nothing about individual genome data except the Fisher's Exact Test results can be learned. Popular solutions are to leverage the homomorphism of some cryptographic schemes such as Paillier or El Gamal [20], so that some properties of the plaintext genome can be preserved in the ciphertext space. However, established cryptographic algorithms like Paillier El Gamal usually require large key sizes (e.g., 1024 to 2048 bits) to ensure a certain level of security, and thus the time to encrypt and decrypt is significant, especially on the user side. The huge volume of genomic data makes this problem even harder to solve. Therefore, we propose to use Boneh-Goh-Nissim (BGN) cryptosystem, which is somewhat-fully homomorphic, both additive and multiplicative. More importantly, BGN is a pairing-based cryptosystem using pairings from elliptic curves. Therefore, compared with Paillier and El Gamal, BGN has the advantage of efficiency. In addition, when computing genomic data which mostly consists of nucleotides: A, C, G, T, the small size of message space makes BGN a good fit.

Our contribution of this paper is summarized as follows:

- To the best of our knowledge, this is the first work to use BGN to provide a privacy preserving algorithm on genomic data.
- We design a new algorithm to enable a private Fisher's Exact Test on the ciphertext of users' genomic data in cloud. The algorithm is correct and secure in the semi-honest security model.
- We implement our BGN-based privacy preserving Fisher's Exact Test algorithm on top of the Pairing-Based Cryptography (PBC) library and test it extensively using real genomic data from the international genome database. The results show that our algorithm is effective and efficient.

The rest of this paper is organized as follows. Section II discusses the related work. Section III covers the preliminaries of this research, which will serve as a reminder on some technical terms. Section IV introduces the system model and security model of our privacy preserving Fisher's Exact test on genomic data. Section V presents the detailed algorithm built for this system, and the analysis of security, efficiency and correctness of the algorithm. Lastly, in our Section VI, we will discuss the system implementation and experiment results on efficiency.

II. Related Work

The issue of privacy in genomic data has been a problem since the Human Genome Project commenced in 2003 [1]. Many researchers have attempted to tackle secure multi-party computation on genomic data with different approaches and solutions. As Akgun et al. notes in [1], the security in genomic data is not up to par with the increasing genomic testing activities. He warns that despite such health and medical advances, there is still a lack in privacy measures. Additionally, due to the vast size of genomic data as a whole, researchers are concerned with throughput. Computation must be practical for real-world use without neglecting privacy [17]. According to Naveed et. al, we need to push forward stronger privacy-preserving mechanisms to ensure security for future genome uses.

Homomorphic encryption on the cloud has been extensively studied. For example, Tebaa et al, [20] noted that using homomorphic encryption like Paillier, any data could be encrypted before sending to the cloud and allowing for the encrypted data to be computed without the need to decrypt. Wu [23] asserts that homomorphic encryption in the cloud as the "Encryption's Holy Grail" since Gentry et. al [6] produced a bootstrapping method that can transform a some-what homomorphic cryptosystem to a fully homomorphic cryptosystem.

Due to genomic privacy concerns, researchers explored homomorphic encryption on genomic data for secure multiparty computation on the cloud. In 2016, Lauter et al [11] proposed ways of providing meaningful computations on statistical algorithms: Chi-Squared Test, Linkage Disequilibrium, Estimation Maximization, and Cochran-Armitage Test for Trend with Lopez-Alt and Naehrigs homomorphic cryptosystem. These statistical computations are calculated using counts of genotypes as the frequency of the genotypes. With many ciphertext of the genomic data, cloud service provider computes the result and returns it as a ciphertext to the researcher to decrypt. This sets the groundwork to all homomorphic encryption computations on genomic data. Also in 2016, Jankly [8] investigated GPU hardware acceleration for genomic data processing, which greatly improves processing time for homomorphic cryptosystems due to the massively parallel computing architecture. The research investigated GPU acceleration of the Paillier cryptosystem on genomic data sets, and some of the work has been utilized in this paper.

Since Lauter et al's work, more research has been done in homomorphically encrypted genomic data in the cloud. Hasan et al. [10] proposed a new way for secure sharing and computation on genomic data using a secure count query with the Paillier cryptosystem. From the genomic sequences, this method allowed them to select a count of a particular SNP sequence. Another method developed for statistical computation on genomic data was proposed by Qiu et al [21] called the Frequent Itemset Mining method which uses the Paillier cryptosystem. In this implementation, a miner mines the frequency of the itemsets that are encrypted. Recently in 2018, Sadat et al [18] developed SAFETY, which is a new method that uses homomorphic encryption with Intels SGX for much faster computation speed than previous proposed systems. To the best of our knowledge, we are the first to implement a BGN cryptosystem for genomic data on the Fishers Exact Test.

III. Preliminaries

This section highlights the technical preliminaries needed to know for further discussion of this paper.

A. Homomorphic Encryption

Homomorphic Encryption allows computations on encrypted data without ever needing to decrypt the data. This

makes homomorphic encryption a viable solution for securing data and keeping some properties of the plaintext data. Homomorphic cryptosystems have the ability to operate addition and/or multiplication on encrypted data. A cryptosystem is deemed multiplicatively homomorphic: if given two ciphertexts $Encrypt(Pk, a)$ and $Encrypt(Pk, b)$, the product of the two ciphertexts equals $Encrypt(Pk, a * b)$. Similarly, it is additively homomorphic if there is a way to calculate $Encrypt(Pk, a + b)$ based on the ciphertexts of a and b, without decrypting any message [2]. A cryptosystem is called fully homomorphic if it satisfies both at the same time.

B. Boneh-Goh-Nissim Scheme

Many homomorphic cryptosystems can only have either additive or multiplicative homomorphism. Boneh-Goh-Nissim cryptosystem, however, is a Somewhat Homomorphic Encryption (SWHE) scheme that renders additive homomorphism for an arbitrary amount of additions and multiplicative homomorphism with one multiplication [2]. This is possible since BGN is a pairing-based cryptosystem, where pairings are taken from elliptic curve.

In BGN scheme, G_1, G_2 fields are both input groups and G_T is the output group, all of prime order p. Let $P \in G_1$, $Q \in G_2$ be generators of G_1 and G_2, respectively. Our implementation uses F_q field: $y^2 = x^3 + x$ which provides type A pairings for q = 3 mod 4. The fields: G_1, G_2, G_T are all in prime order and are cyclic groups where a pairing in this elliptic curve is $e : G_1 * G_2 \rightarrow G_T$. For the pairing, bilinearity holds:

$$\forall a, b \in \mathbb{Z}_p^* : e(P^a, Q^b) = e(P, Q)^{ab}.$$

Given the public key $PK = \{N, G, G_1, e, g, h\}$, to encrypt a message m, pick a random r from $\{1, 2, ..., N\}$, and compute $C = g^m h^r$. To decrypt a ciphertext C using the secret key $SK = q_1$, it suffices to compute the logarithm of C^{q_1}, since $C^{q_1} = (g^m h^r)^{q_1} = (g^{q_1})^m$.

BGN is additively homomorphic. Given the two ciphertexts C_1, C_2, for cleartexts m_1 and m_2 respectively, the public key h element and r, a randomly generated number less than N, the equation below shows the way to calculate the $a + b$'s ciphertext using C_1 and C_2.

$$C_1 C_2 h^r = (g_1^m * h^{r_1}) * (g^{m_2} * h^{r_2}) * h^r = g^{m_1+m_2} h^{r_1+r_2+r} \quad (1)$$

The multiplicative homomorphism of BGN is based on bilinear map. Let $g_1 = e(g, g)$ and $h_1 = e(g, h)$. g_1 is of order N and h_1 is of order q_1. We will have an unknown $\alpha \in \mathbb{Z}$ such that: $h = g^{\alpha q_2}$. Using these formulas we can calculate for the two ciphertexts:

$$C = e(C_1, C_2) h_1^r = e(g^{m_1} h^{r_1}, g^{m_2} h^{r_2}) * h_1^r$$

It can be derived that

$$C = e(g, g)^{m_1 m_2} * h_1^{r+m_1 r_2+m_2 r_1+\alpha q_2 r_1 r_2}$$

Thus, we can see that the BGN is both additively homomorphic and has the ability to operate one multiplication on ciphertext using its bilinear map [2] [25].

The number of operations on BGN can additionally be enhanced with bootstrapping, to add a $Rencrypt(Encrypt(Pk, y), sk)$ function on top of the noise-filled ciphertext, which then will lessen the noise from the operations done on the ciphertext, along with adding one more multiplication operation [6] [23]. Though computationally expensive, adding the bootstrapping functionality transforms the homomorphic scheme into a Fully Homomorphic Encryption System (FHE).

The BGN Cryptosystem is only able to work on a limited size of message space, due to the discrete logarithm needed with decryption. This constraint poses no issue for genomic data since we will not need to return a large message when computing genomic data which will be consisting of the nucleotides: A, C, G, T.

C. Statistical Tool: Fisher's Exact Test

Studying genomic data can show the gene variants and its correlation to specific traits. In this paper, we focus on providing the privacy preserving Fisher's Exact Test on genomic data. This test is used with two nominal variables that are statistically significant through P-value. In short, Fisher's Exact Test will test if a percentage of a variable provides different values of the other variable [16].

According to Fisher's Exact Test, given a 2 x 2 contingency table as shown in Table I, the probability of obtaining any such set of values is following the hypergeometric distribution.

TABLE I
2 X 2 CONTINGENCY TABLE SAMPLE

	Category 1	Category 2	Row Total
Group 1	a	b	a+b
Group 2	c	d	c+d
Column Total	a+c	b+d	a+b+c+d=N

$$p = \frac{((a+b)!(c+d)!(a+c)!(b+d)!)}{a!b!c!d!N!} \quad (2)$$

As one can see, Fishers Exact provides the exact value of probability. Fisher's Exact has generally been avoided due to its large computational task. For smaller populations assuming independence, this equation must be used to avoid incorrect calculations.

The difference between using the Chi-Squared Test and Fisher's Exact, is that the Chi-Squared Test is an approximation that cannot be used in smaller sample sizes [14]. GWAS researchers need to use Fishers Exact due to the importance of finding the exact probability value. Fishers Exact is used in genetics for populations less than 20, or when any cell is in the 2 x 2 contingency table is less than 5.

Overall, Fisher's Exact Test is used often for nominal sample sizes to provide the exact probability value, instead of an approximation. As suggested in Wang et. al [22], it is important to use Fisher's Exact Test as a cross-validation scheme to ensure the statistically stability in genetic analysis. We will show that this significant stability can be provided in a privacy preserving way on top of BGN.

IV. System Overview

In this paper, we propose a system architecture that calls for all genomic data to be encrypted before storing it inside the cloud platform. Researchers and scientists can then query the cloud for statistical computations and decrypt the result of these computations with the secret key. Our system architecture enables genomic data providers to send cryptographically secure genomic data into the cloud service platform and allows for the researchers to query statistical computation to receive insightful results. Our system consists of: researchers, cloud service platform, genomic data providers, and certificate authority.

A. System Model

In this subsection we will discuss the parties in the system model and our assumptions. The parties involved in our system architecture include the following:

- *Genomic Data Providers:* this party involves individuals or entities who have data to be stored in a cloud service provider. They have an interest in keeping the genomic data secure and protected from outsiders in order to follow certain privacy guidelines. These entities can be hospitals, research institutions or individual genomic data owners.
- *Authorized Researchers:* this party involves researchers who want to query the cloud for mulit-party computation. Their goal is to use the encrypted data to query and learn from it. They send their query to the cloud for results.
- *Cloud Service Platform (CSP):* the CSP is used to store and compute over the ciphertext. It contains the collective storage of the protected data. It also performs queries to compute over the ciphertext and return the appropriate statistical result.
- *Certificate Authority (CA):* the Certificate Authority is an entity that generates, distributes and manages the keys of the cryptosystem. It assigns the associated secret and public keys for encryption and decryption of the genomic data. The public key pairs of legitimate users of the system for authentication purposes are also maintained by the CA.

The system overview is described as shown in Figure 1. Initially, the keys are assigned by Certificate Authority. They provide genomic data providers a public key to encrypt and the researchers the corresponding secret key to decrypt the Fisher's Exact result. Overall, the Certificate Authority will arrange the system's key management throughout the process.

Genomic data providers encrypt their data with the public key and send their data to the public cloud service platform. CSP stores all encrypted data from data providers and performs the query on encrypted data as requested by the Authorized Researchers. Authorized Researchers are allowed to query the CSP for the computations of statistical functions like Fisher's Exact Test. After the computation result as a ciphertext is returned to the Authorized Researcher, he can decrypt the result using secret key. This similar structure have been adopted by other works [10] [11] [18] [20] [21].

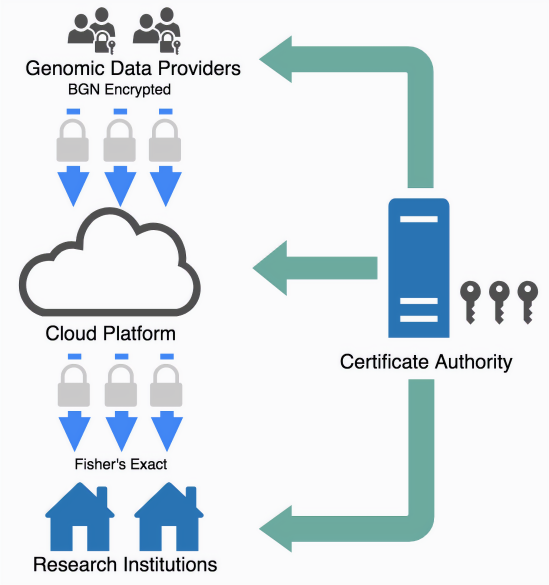

Fig. 1. System Architecture

The main body of the privacy preserving Fisher's Exact Test algorithm that we will present in Section V is executed on CSP. Other statistical functions can also be implemented following the similar system architecture. Multiple genomic data providers and researchers to store in and query the cloud for meaningful computation in our system architecture are allowed. This system can be used for collective research among many facilities without sacrificing privacy and secrecy.

B. Threat Model

In our system, we assume that all parties, i.e., the cloud service platform, certificate authority, authorized researchers and genomic data providers are all honest but curious, who will honestly follow the protocol but try to discover others' private data as much as possible. The parties do not fully trust each other. Genomic data providers do not disclose private data to CSP or the researchers directly. The cloud is not fully trusted, in that CSP can not learn the private data from the procedure of executing the privacy preserving Fisher's Exact Test and from the encrypted intermediate results. Researchers in this system can only receive the ciphertext of the query's result. They are only able to decrypt the result with the associated secret key. The encrypted genomic data stored in cloud is not accessible to them.

V. Privacy Preserving Fisher-Exact Test on Genomic Data

In this section, we will present the privacy preserving Fisher's Exact Test algorithm in the cloud system. We will also provide analysis of its security, correctness, and efficiency.

A. Algorithm Design

The Fisher's Exact Test we are introducing in this section is based on a 2 x 2 contingency table, for clarity of presentation.

Please note that our privacy preserving algorithm can be easily extended to be applied to Fisher's Exact Tests with a mxn contingency table in general.

In the algorithm design, the first challenge is to convert the genomic data, i.e., nucleotides A, C, G, T, to the integer space so that the BGN cryptosystem defined on cyclic groups can be applied. We solve this challenge by encypting the genomic data on the data providers side, which is tailored to the Fisher's Exact Test. Specifically, we encode the relevant genomic information into a quadruple for each genomic data provider i: $(x_i^1, x_i^2, y_i^1, y_i^2)$. To determine the values in the quadruple, particular sections of the genome are scanned to consider whether this genome belongs to group 1 or group 2, category 1 or category 2. If it belongs to group 1, category 2, for example, then $(x_i^1 = 0, x_i^2 = 1, y_i^1 = 0, y_i^2 = 0)$. In words, x corresponds to group 1, and y corresponds to group 2. Each superscript is corresponding to the category.

After the genomic data is encoded, we can run our privacy preserving Fisher's Exact Test algorithm as shown in Algorithm 1. In particular, each genomic data provider first encrypts the quadruple using BGN public key. The 4 ciphertexts will look random to other parties, even though the plaintext is either 0 or 1, due to the probabilistic property of BGN. After data providers send ciphertexts to the CSP. CSP calculate the product of all first components' ciphertexts in the quadruple collected, and adds another round of re-randomization by multiplying h^{r^1}. Similar calculations are conducted on the other three components in the quadruple, rendering four messages in ciphertext space. After receiving these four messages from CSP, the researcher using BGN secret key decrypts them, and calculates the final Fisher's Exact Test result as shown in line 7 and 8.

Algorithm 1 Privacy Preserving Fisher-Exact Test Scheme on Genomic Data Using a 2 x 2 Contingency Table

INPUT: Each genomic data provider i holds his encoded genomic data $(x_i^1, x_i^2, y_i^1, y_i^2)$, the public key of BGN: $PK = \{N, G, G_1, e, g, h\}$. The authorized researcher holds the secret key of BGN: SK

OUTPUT: Authorized researcher obtains Fisher's Exact Test Result p on a set genomic data from providers $\{1, 2, ..., n\}$.

1: **repeat**
2: Genomic data provider i encrypts genomic data using BGN, and obtain 4 ciphertexts $E(x_i^1, PK)$, $E(x_i^2, PK)$, $E(y_i^1, PK)$, and $E(y_i^2, PK)$.
3: Provider i sends the 4 ciphertext to the cloud service platform in order.
4: **until** all providers in $\{1, 2, 3, ..., n\}$ has finished encryption and ciphertext transmission.
5: Cloud service platform calculates the following, where r^1, r^2, r^3, r^4 are positive random numbers less than N.
$\Pi_{i=1}^n E(x_i^1, PK) * h^{r^1}$, $\Pi_{i=1}^n E(x_i^2, PK) * h^{r^2}$,
$\Pi_{i=1}^n E(y_i^1, PK) * h^{r^3}$, $\Pi_{i=1}^n E(y_i^2, PK) * h^{r^4}$.
6: The cloud service platform returns the 4 products in ciphertext to the researcher.
7: The researcher calculate the following:
$a' = D(\Pi_{i=1}^n E(x_i^1, PK) * h^{r^1}, SK)$,
$b' = D(\Pi_{i=1}^n E(x_i^2, PK) * h^{r^2}, SK)$,
$c' = D(\Pi_{i=1}^n E(y_i^1, PK) * h^{r^3}, SK)$,
$d' = D(\Pi_{i=1}^n E(y_i^2, PK) * h^{r^4}, SK)$.
8: The researcher calculates p according to Equation (2).

B. Security Analysis

We will provide a brief security analysis of our privacy preserving Fisher's Exact Test against semi-honest CSP and authorized researchers. In Algorithm 1, the messages that CSP receives are the encrypted quadruples from genomic data privacy. CSP cannot learn the plaintext of any component, i.e., whether any provider belongs to any group or category, due to the security of BGN which is based on the difficulty of subgroup decision problem. The calculations on CSP do not provide more information about plaintexts than what the messages received render. For the researcher who receives 4 messages in the ciphertext space, all she can learn after decrypting the message is the statistics, i.e., the total number of samples belong to each group and category. No individual genomic data provider's group or category information can be revealed. Overall, all parties cannot learn anything other than what are sent to them and the output of the algorithm, under the semi-honest model.

C. Correctness Analysis

Based on the encoding method described in Section V-A, we know that $\sum_{i=1}^n x_i^1$ equals the number of samples in total which fall in group 1 and category 1, i.e., the a in the contingency table as shown Table I. If we can show that a' in line 7 equals $\sum_{i=1}^n x_i^1$, then it means $a = a'$, the same for b, c, d as well, and thus the correctness of Algorithm 1 is verified. In fact, due the additively homomorphic property of BGN (as shown in Equation (1)), we have

$$\begin{aligned} a' &= D(\Pi_{i=1}^n E(x_i^1, PK) * h^{r^1}, SK) \\ &= D(E(\sum_{i=1}^n x_i^1, PK), SK) \\ &= \sum_{i=1}^n x_i^1 = a \end{aligned}$$

D. Efficiency Analysis

We analyze the efficiency of Algorithm 1 in terms of computation cost and communication cost. We denote the time cost to conduct one multiplication in group G as M, and the time cost for one exponentiation in group G as EXP.

For one encryption operation in BGN, it has two exponentiation and 1 multiplication. Therefore, each genomic data provider needs $4 * (2\text{M} + \text{EXP})$ computation time to execute Algorithm 1 (line 2). It takes CSP $4((n + 1)\text{M} + \text{EXP})$ computation time to perform line 5 in the algorithm. The

authorized researcher mainly needs four BGN decryption time to complete Algorithm 1.

The message transmissions in Algorithm 1 include 4 ciphertexts from each genomic data provider to CSP, and 4 ciphertexts from CSP to the researcher. Each ciphertext message is equal to the length of N. As we can see our algorithm is very efficient both in computation cost and communication cost.

VI. Implementation and Evaluation

In this section, we will discuss the implementation issues of the system, and evaluate our algorithm through extensive tests.

A. Implementation Details

We first implemented the BGN cryptosystem in C/C++ programming language. Using this cryptosystem, we will be testing the Fisher's Exact Test in terms of the time efficiency and usability of the system for genomic data.

We used the GMP [7] and PBC library [15] in the implementation The GMP library, which stands for GNU Multi-Precision Library, allows for big number integers to be used specific for cryptosystems such as BGN. Then we use the Pairing Based Cryptography (PBC) library for the BGN cryptosystem since BGN is a pairing based cryptosystem. We also use Libsodium library [2] [12] [19], with the Mersenne Twister Algorithm to produce random numbers.

This implementation uses polynomials to represent the message and the encrypted message. Polynomials are created using mpf_t, GMP big floating numbers. All unencrypted polynomials are members of the struct Plaintext with int64_t as the coefficients. All encrypted polynomials are members of the struct Ciphertext with element_t, which is a PBC data type storing groups, rings, and fields, as the coefficients. To decode the polynomial back into an integer, we can use the String(plaintext) function. We implemented Horner's method [19] to evaluate polynomials in the String function. Horner's method runs in O(n) time where n is the number of polynomials.

Table II shows the implementation of BGN key generation, encryption and decryption procedures. The key generation function NewKeyGen produces a Publickey and a Secertkey. The NewKeyGen function takes in 8 parameters: key bits, message space, base, floating point base, floating point precision, Publickey, and Secretkey. Key bits is the number of bits that the program will use to generate p and q. We choose 3 as the polynomial base. A polynomial in base 3 can represent big integers with small numbers. Floating point base is the same as base and floating point precision is the floating point we'll be representing up to. Parameter "deterministic" makes the homomorphic operations deterministic or non-deterministic.

Inside the decrypt function we compute the discrete logarithm by iterating over all possible values until ct is equal to $ct^{sk.Key}$. This process can be slow since this is a brute force approach. We would like to note that there are faster methods to compute the discrete logarithm problem such as probabilistic algorithms like Pollard's Rho or deterministic algorithms like Baby-Step Giant-Step [29].

TABLE II
BGN Key Generation, Enc. and Dec. Implementation

Key Generation: NewKeyGen(keybits, base, floatbase, floatPrec, deterministic, pk, sk)	
Input: (keybits, base, floatbase, floatPrec) $\in \mathbb{Z}$ deterministic $\in$ [0, 1]	
Compute	n = p×q where p, q $\in \mathbb{P}$
Choose	cyclic group G of order n then
Set	pairing$\leftarrow$ G x G $\rightarrow$ G_1
Choose	g, u $\in$ G
Set	h $\leftarrow$ u^q
Output: pk $\in$ PublicKey, sk $\in$ SecretKey	
pk $\leftarrow$ (pairing, g, u, h, n, base, floatbase, floatPrec, deterministic)	
sk $\leftarrow$ (p, base)	

Encryption: Encrypt(pt,pk)	
Input: pt $\in$ Plaintext pk $\in$ PublicKey	
Choose	r $\in \mathbb{Z}_{\mathbb{N}}$
Compute	C $\leftarrow$ ($g^{pt.Coefficients[i]}$) $\times$ (h^r)
Output: C $\in$ Ciphertext	
C $\leftarrow$ (encryptedCoefficients, degree, scalebase, deterministic)	

Decryption: Decrypt(C, pk, sk)	
Input: C $\in$ Ciphertext pk $\in$ PublicKey sk $\in$ SecretKey	
Compute	Csecret $\leftarrow$ (encryptedCoefficents[i:size])^p where p $\in$ SecretKey
Compute	Gsecret $\leftarrow$ g^p where g $\in$ Publickey
Find	Gsecret^i $\equiv$ Csecret where i $\in \mathbb{Z}$
Output: i $\in \mathbb{Z}$	

B. Experiment Setup

All experiments were conducted on one machine with Intel (R) Xeon (R) CPU E5-2630 v3 @ 2.40GHz processors (32 cores) with 168GB of RAM, running Ubuntu 16.04 LTS. We will validate the effectiveness and efficiency of our algorithm using genomic datasets from InternationalGenome.com [28]. There are varieties in the genomic data files from varieties like: sex, population of the individual, and different data collections. For this research, we used low coverage WGS files in the Genome Reference Consortium Human Build 3 (GRCh38) data collection [28]. We use a utility software for manipulating genetic sequence alignments, i.e., SAMtools [30] to convert genomic data files to the .sam format. We also have developed a parser to process the .sam files into the codes that our algorithm can recognize.

To test the efficiency of our algorithm, we plan three sets of experiments with different variables as follows.

- Different Key Bits
- Different Prime Numbers of Message Space
- Different PolyBases

C. Evaluation Results

This subsection will present the results from the experiments we have performed. In particular we evaluate the efficiency of three components in our algorithm: key generation, encryption, and decryption. We will examine the suitable parameters of BGN for Fisher's Exact Test on genomic data and provide recommendations based on the experiment results. More experiments, for example, on the communication overhead, are planned for future work.

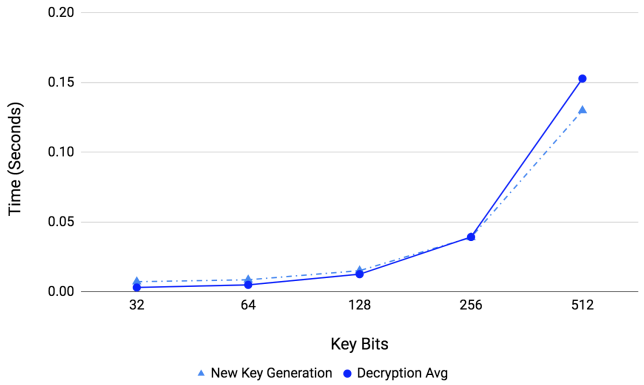

Fig. 2. Time of Key Generation and Decryption Based on Different Key Lengths

TABLE III
EFFICIENCY OF ENCRYPTION AND DECRYPTION WITH VARIOUS KEY SIZES

$KeyBits$	$EncPerSec$	$DecPerSec$
32	753	331
64	524	204
128	281	80
256	97	26
512	24	7

*1) **Varying Key Size**:* First, we test the times for New Key Generation, Encryption, and Decryption functions for different key sizes from 32 to 2048 bits, with number of Sequences 20, Polybase 3 and Message Space 1021.

In Figure 2, we see the results from our experiment on key generation time and decryption time. Table III shows the time efficiency of encryption and decryption in terms of the number of times of executions per second.

As one can see, there is exponential relationship between the size of the key and the computation time showing a rough estimate of $O(2^n)$ for Key Generation, Encryption, and Decryption Functions. However, with 512 bits key, BGN scheme is still fast. This is due to the Elliptic Curve Cryptography used to generate the BGN key, which makes generation much faster than other methods such as RSA [31]. Additionally, in Elliptic Curve Cryptography (ECC) having smaller key bits will still ensure the same level of higher security. For example, having an ECC key size of 283 bits is equivalent to an RSA security key of 3072 bits. Consequently, just having a 283 bits makes the crytosystem very secure [31]. Seeing that this Key Generation is much faster for BGN than for Paillier we can say that this cryptosystem is additionally useful for genomic data providers who need to encrypt large amounts of data quickly. This means in order to have a practical computation time for this program, we should keep key bit size as 256 or lower.

*2) **Varying Polybases**:* Second, we look at how different polybases can change computation time.

According to Dowlin et. al, the odd bases are preferred due to space efficiency because with an odd base, we are able

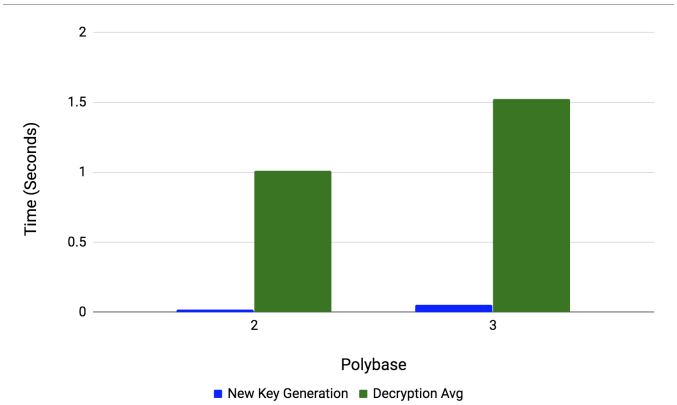

Fig. 3. Time of Key Generation and Decryption based on PolyBases

TABLE IV
ENCRYPTION TIME WITH DIFFERENT POLYBASES

$Polybase$	$Encryption(Seconds)$
2	16.737
3	22.556

to have signed coefficients to allow for encodings of such polynomials as: $p(X) = X^3 - X + 1$ for encoding number 25 [32]. Along with that higher polybases provide shorter polynomials with larger coefficients. Therefore, we wanted to investigate whether or not the changing polybases will make a big difference in time efficiency in general for our Fisher's Algorithm with BGN encryption and decryption.

Our results are plotted in Figure 3 and Table IV to depict differing time computations for the polybases. Looking at these results and keeping space efficiency in mind, we can see that odd polybases generally have a slightly higher computation time, but will pay off if space is an issue. Thus, odd bases more space efficient and will come in handy for larger messages.

*3) **Varying Message Space**:* Lastly, in our experiments testing BGN Parameters, we look at whether or not message space in BGN makes a difference in our implemented Fisher's Exact Algorithm. We run these experiments to further understand the true limitations with the BGN message space and how this can affect our genomic data result.

From Figure 4 and Table V, we can see that message space does not affect computation in either Key Generation

TABLE V
ENCRYPTION TIME WITH VARIOUS MESSAGE SPACE SIZES

$MessageSpace$	$Encryption(Seconds)$
1021	22.101
2003	22.269
4001	22.201
8009	22.471
10 007	22.596
100 003	22.861
10 000 019	23.052

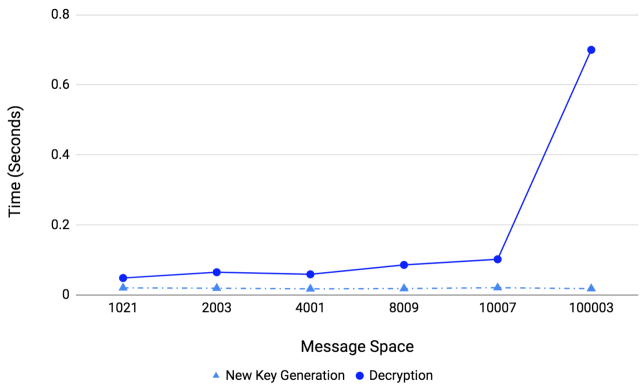

Fig. 4. Time of Key Generation and Decryption based on Different Sizes of Message Space

or Encryption, but will grow exponentially in decrypting messages larger than 10,000.

VII. Conclusion

This paper explores the usage and practicality of homomorphic encryption on Fisher's Exact Test for genomic data. We propose to use BGN Homomorphic Encryption to encrypt the data and sending the ciphertext to store in the cloud service platform. We provide security, correctness and efficiency analysis of our privacy preserving Fisher's Exact Test algorithm on genomic data. From our experiments, we see that setting certain parameters will assist us in optimizing computation time. We conclude that using the BGN cryptosystem with the Fisher's Exact Test is secure, practical, and efficient.

References

[1] M. Akgun, A. Bayrak, B. Ozer and M. Sarolu. Privacy preserving processing of genomic data: A survey. *Journal of Biomedical Informatics*, 56, pp.103-111, 2015.

[2] D. Boneh, E.-J. Goh, and K. Nissim, Evaluating 2-DNF Formulas on Ciphertexts, Theory of Cryptography Lecture Notes in Computer Science, pp. 325341, 2005.

[3] Cloud Security Alliance, "The Treacherous 12: Cloud Computing Top Threats in 2016", 2016.

[4] D. Freeman, Homomorphic Encryption and the BGN Cryptosystem, *http://theory.stanford.edu/∼dfreeman/cs259c-f11/lectures/bgn.pdf*, 18-Nov-2011.

[5] Z. Fang, J. Martin, Z. Wang, Statistical methods for identifying differentially expressed genes in RNA-Seq experiments. *Cell & Bioscience, 2:26, 2012*.

[6] C. Gentry, S. Halevi, and N. P. Smart, "Better bootstrapping in fully homomorphic encryption," *IACR Cryptology ePrint Archive 2011/680, vol. 2011, 2011*.

[7] The GNU MP Bignum Library, The GNU MP Bignum Library. [Online]. Available: https://gmplib.org/. [Accessed: 17-Oct-2018].

[8] S. Jankly, "GPU-accelerated privacy-preserving genomic data processing", Master's Thesis, California State Polytechnic University, Pomona, 2016.

[9] Goldwasser S, Micali S (1984) Probabilistic encryption. *Journal of Computer and System Sciences 28: 270–299*.

[10] M. Hasan, M. Mahdi, and N. Mohammed, (2018). Secure Count Query on Encrypted Genomic Data: A Survey. *IEEE Internet Computing, 22(2), pp.71-82*.

[11] K. Lauter, A. Lpez-Alt, M. Naehrig, "Private computation on encrypted genomic data" in *Progress in CryptologyLATINCRYPT, New York, NY, USA:Springer-Verlag, pp. 3-27, 2014.*

[12] Libsodium documentation, *Introduction - Libsodium documentation.* [Online]. Available: https://download.libsodium.org/doc/. [Accessed: 17-Oct-2018].

[13] Y. Liu, Z. Wan, W. Xia, M. Kantarcioglu, Y. Vorobeychik, W.E. Clayton, A. Kho, D. Carrel, B. Malin (2018) Detecting the Presence of an Individual in Phenotypic Summary Data. *Proceedings of the American Medical Informatics Association Annual Fall Symposium (AMIA)*.

[14] J. Ludbrook, Analysis of 2 x 2 tables of frequencies: matching test to experimental design., *International journal of epidemiology.*, Dec-2008. [Online]. Available: https://www.ncbi.nlm.nih.gov/pubmed/18710887.

[15] B. Lynn, PBC, PBC Library Manual 0.5.14. [Online]. Available: https://crypto.stanford.edu/pbc/manual/.

[16] J. McDonald, Handbook of Biological Statistics, Fisher's exact test of independence - Handbook of Biological Statistics. [Online]. Available: http://www.biostathandbook.com/fishers.html.

[17] Naveed, M. et al. Privacy in the genomic era. *ACM Comput. Surv. 48, 1, Article 6 (July 2015);* DOI: http://dx.doi.org/10.1145.2767007.

[18] M. Sadat, A. Nazmus, N. Mohammed, F. Chen, S. Wang, and X. Jiang, , SAFETY: Secure gwAs in Federated Environment Through a hYbrid solution with Intel SGX and Homomorphic Encryption, *arXiv:1703.02577* 07-Mar-2017.

[19] S. Servan-Schreiber, sachaservan/bgn, *GitHub*, 06-Mar-2018. [Online]. Available: https://github.com/sachaservan/bgn.

[20] M. Tebaa, S. E. Hajji and A. E. Ghazi, "Homomorphic encryption method applied to Cloud Computing," *2012 National Days of Network Security and Systems*, Marrakech, 2012, pp. 86-89.

[21] S. Qiu, B. Wang, M. Li, J. Liu and Y. Shi. Toward Practical Privacy-Preserving Frequent Itemset Mining on Encrypted Cloud Data. *IEEE Transactions on Cloud Computing*, pp.1-1, 2017.

[22] L. Wang, P. Jia, R. Wolfinger, X. Chen and Z. Zhao. Gene set analysis of genome-wide association studies: Methodological issues and perspectives. *Genomics*, 98(1), pp.1-8, 2011.

[23] D. Wu, Fully Homomorphic Encryption: Cryptography's holy grail, *XRDS: Crossroads, The ACM Magazine for Students*. [Online]. Available: https://dl.acm.org/citation.cfm?id=2730906.

[24] R. Xie, C. Xu, F. Li and C. He, "Ciphertext retrieval against insider attacks for cloud storage," *2016 2nd IEEE International Conference on Computer and Communications (ICCC)*, Chengdu, pp. 202-206, 2016.

[25] X. Yi, R. Paulet, and E. Bertino, Homomorphic Encryption, *Homomorphic Encryption and Applications*, Cham: Springer, pp. 2746, 2014.

[26] J. Yuan and S. Yu, "Privacy Preserving Back-Propagation Neural Network Learning Made Practical with Cloud Computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 1, pp. 212-221, Jan. 2014.

[27] Z. Zhu, Z. Zheng, F. Zhang, Y. Wu, M. Trzaskowski, R. Maier, M. R. Robinson, J. J. McGrath, P. M. Visscher, N. R. Wray, and J. Yang, Causal associations between risk factors and common diseases inferred from GWAS summary data., *Nature communications.*, 15-Jan-2018. [Online]. Available: https://www.ncbi.nlm.nih.gov/pubmed/29335400.

[28] IGSR: The International Genome Sample Resource, 1000 Genomes — A Deep Catalog of Human Genetic Variation. [Online]. Available: http://www.internationalgenome.org/.

[29] S. Galbraith, P. Wang, and F. Zhang, Computing elliptic curve discrete logarithms with improved baby-step giant-step algorithm, *Advances in Mathematics of Communications*, vol. 11, no. 3, pp. 453469, 2017.

[30] SAMtools, SAMtools. [Online]. Available: http://samtools.sourceforge.net/.

[31] R. Sinha, H. K. Srivastava, S. Gupta, "Performance Based Comparison Study of RSA and Elliptic Curve Cryptography", *International Journal of Scientific and Engineering Research*, Volume 4, Issue 5, May, 2013.

[32] N. Dowlin, R. Gilad-Bachrach, K. Laine, K. Lauter, M. Naehrig, and J. Wernsing. Manual for using homomorphic encryption for bioinformatics. *Proceedings of the IEEE*, 105(3):552567, 2017.