Published in final edited form as:

IEEE J Biomed Health Inform. 2017 September ; 21(5): 1466–1472. doi:10.1109/JBHI.2016.2625299.

Private and Efficient Query Processing on Outsourced Genomic Databases

Reza Ghasemi, Md., Momin Al Aziz, Noman Mohammed, Massoud Hadian Dehkordi, and Xiaoqian Jiang

Abstract

Applications of genomic studies are spreading rapidly in many domains of science and technology such as healthcare, biomedical research, direct-to-consumer services, and legal and forensic. However, there are a number of obstacles that make it hard to access and process a big genomic database for these applications. First, sequencing genomic sequence is a time-consuming and expensive process. Second, it requires large-scale computation and storage systems to process genomic sequences. Third, genomic databases are often owned by different organizations and thus not available for public usage. Cloud computing paradigm can be leveraged to facilitate the creation and sharing of big genomic databases for these applications. Genomic data owners can outsource their databases in a centralized cloud server to ease the access of their databases. However, data owners are reluctant to adopt this model, as it requires outsourcing the data to an untrusted cloud service provider that may cause data breaches. In this paper, we propose a privacy-preserving model for outsourcing genomic data to a cloud. The proposed model enables query processing while providing privacy protection of genomic databases. Privacy of the individuals is guaranteed by permuting and adding fake genomic records in the database. These techniques allow cloud to evaluate *count* and *top-k* queries securely and efficiently. Experimental results demonstrate that a count and a top-*k* query over 40 SNPs in a database of 20,000 records takes around 100 and 150 seconds, respectively.

Index Terms

Genomic data private computation; privacy in cloud computing; homomorphic encryption; privacy in genomic data storage

I. Introduction

The recent advancement in genomic data generation and availability has impacted related scientific studies. These large genomic datasets help us to understand the relation between some diseases and our genes. Researchers can now analyze susceptibility to any particular disease or even make personalized medicine based on the patients genomic sequence [1]. Therefore, the continuation and advancement of medical research heavily depend on the availability of large genomic datasets.

There is an urgent need for a unified genomic data repository [2]. The current practice is to keep genomic records within organizations (hospitals, government agencies, etc.) and share them with specific researchers on request. It allows researchers to gain access to genome

sequences and conduct their studies. However, one critical flaw in this approach is each organization has to take responsibility of gathering, storing and sharing data independently. It instigates concerns among privacy and security specialists because these duties are time consuming and sometimes well beyond the abilities and expertise of small organizations. Also there are issues of sharing data beyond borders or even a single province of a country [3]. A possible solution that addresses these concerns is establishing a centralized data repository to store genomic data and securely share this information among researchers. Outsourcing genomic data in a unified repository will accelerate research and reduce required time and cost for research studies. Therefore, many projects have started to collect and centralize genome sequences and other sensitive attributes of individuals in a unified repository [4].

A recent survey documented 14 different techniques for breaching genomic data privacy, and classified these techniques under three categories: identity disclosure attacks, attribute disclosure attacks, and completion attacks [5]. These techniques question the efficacy of the protection mechanisms that are currently in practice for genomic data sharing. This reality demands new privacy preserving framework that can provide data privacy of genomic databases and prevent inference attacks against outsourced data.

The objectives of this paper is to present a privacy-preserving framework to store and execute queries on genomic data. In particular, the framework should ensure the privacy of individuals (e.g., genome sequence or sensitive attributes) and the query execution mechanism should be scalable to large databases.

A. Current Techniques

One of the earlier attempts to securely outsource genomic data in a cloud is a cryptographic model proposed by Kantarcioglu et al. [6]. Their proposed model encrypts every record by a homomorphic encryption scheme. Although encrypting all data prevents malicious observers from gaining any information and provides much security, their method has two shortcomings. First, the query execution time for their method is quite large. As mentioned in their paper, it takes around 30 mins to execute a count query over 40 SNPs in a database of 5000 records. Therefore, their method may not be scalable for big databases containing millions of records. Second, due to the use of homomorphic encryption scheme, which has a large ciphertext expansion factor, the encrypted database requires large storage space. Canim et al. later presented a cryptographic model to improve the efficiency of the previous scheme [7]. However, the proposed model requires a secure co-processor, which is not always possible to ensure in real-life scenarios. In practice, we outsource our data to cloud servers (e.g. Amazon EC2) where it is not easy to ensure the availability of a secure hardware. These are the only two solutions that are closely related to the problem of secure query execution over outsourced genomic data.

There are a number of other cryptographic solutions proposed for genomic data privacy over the past few years [8]–[11]. These methods, although related, do not address the problem discussed in this paper (see supplementary notes for more discussion).

B. Contributions

In this paper, we propose a private and efficient model that addresses the challenges of outsourcing genomic data for query execution. Our proposed model supports two types of queries: count query and top- k . In a count query, researchers indicate some positions of SNPs along with their values. The query result returns the number of individuals matching this query predicate. Top- k query determines k records in a database that are most similar to a given reference sequence. Other complex functions can be computed using these primitive queries.

The privacy of genomic databases in an outsourced model can be guaranteed by encrypting each record with a semantically secure encryption scheme (as done by the existing technique). Encrypted data provide protection against internal (i.e., cloud service provider) or external (i.e., adversaries exploiting system vulnerabilities) parties as the decryption key is only known to the data owner. However, the main shortcoming of this approach is efficiency. We propose a private and efficient model for data outsourcing *without encrypting the whole database*. In designing our solution, we attempt to trade off between the privacy, efficiency, and storage. Our proposed framework is very efficient for small number of phenotypes compared to other crypto-based solutions (see supporting documents for more discussion).

The contributions of this paper are summarized below:

- Privacy: It guarantees privacy of participants' genome sequences and their sensitive attribute (e.g., disease).
- Type of Queries: The proposed model can privately execute count and top- k queries..
- Efficiency: The proposed model has been implemented and tested with both real-life and synthetic datasets. Experimental results demonstrate that a count and a top- k query over 40 SNPs in a database of 20,000 records takes around 100 and 150 seconds, respectively.

The rest of the paper is organised as follows. Section II provides an overview of the proposed method. Section III reviews genomic dataset, and security requirement of the framework. In Section IV, we introduce the concept of valid permutation, which is used in our method to protect the privacy of individuals. We describe our proposed model in Section V. Section VI analyzes how the proposed model preserves privacy of individuals. The experimental results are presented in Section VII. We finally review previous works in supplementary notes and conclude in Section VIII.

II. Overview Of Presented scheme

The proposed model includes four major parties: proxy, cloud, researchers and data providers. The architecture of the proposed model is shown in Figure 1.

- Data Providers: Data providers (e.g., hospitals) possess raw genomic databases and provide their data to proxy. Proxy after preprocessing stores the data in the

cloud. Once stored in the cloud, no further interaction is needed between data providers and the cloud to execute any query.

- **Proxy:** Proxy is a trusted party which plays an important role in the proposed model. It preprocesses raw data by encrypting, permuting and adding fake genomic sequences. After preprocessing, it stores the modified version in the cloud. Proxy also translates queries received from researchers to facilitate query execution by the cloud. Finally, proxy receives encrypted results from the cloud and it returns query results to researchers after decryption.
- **Cloud Server:** Any commercial cloud service provider can play this role as they offer a cheap and highly scalable solution for any organization. Cloud receives translated queries from the proxy and it executes these queries on the stored data. The query execution process does not leak any information to cloud since the whole computation is done on modified data. In this paper, *DB* refers to the database inside the cloud.
- **Researchers:** Researchers might be an individual or an organization who want to execute queries over genomic databases. Researchers submit their queries to the proxy.

The proposed framework is practical for real-life application scenarios. We assume that the cloud is an untrusted entity as it is operated by third party organizations (e.g., Amazon, Microsoft). In addition, there are different privacy regulations (e.g., HIPAA, CLIA) which do not allow the usage of public cloud without ensuring privacy protection [12]. For example, NIH allows researchers to use public cloud; however, it is the responsibility of the researchers to ensure data security and privacy [13]. On the other hand, the proxy is unconditionally trusted in our framework. This assumption is not far-fetched because there are government organizations (e.g., National Institutes of Health (NIH) in US) that are trusted and currently mediate data sharing requests by researchers. These government organizations provide access controls and other services. Hence, these organization can play the role of proxy and connect researchers to the cloud.

III. Preliminary

In this section, we present the format of genomic dataset and privacy requirement of our model.

A. Data

Table I shows an example of raw genomic dataset (ignore the frequency rows for now). We can categorize the attributes of each record, Rec_i , into two categories: *a*) genomic sequence and *b*) phenotype. A genomic sequence is consisted of four letters *A*, *T*, *G* and *C*. Genome sequence is sensitive because it contains information regarding physical characteristics of an individual [14]. We store this sequence according to their positions. Phenotype attribute contains an individual's observable physical trait. In this paper, we assume that the phenotype attribute refers one disease affection status.

B. Privacy Requirements

A genomic database consists of genomic sequences and disease status of patients. Recent research results show that given some background information about an individual, an adversary can identify or learn sensitive information about the victim from the dataset [15], [16]. If an adversary is able to link a record to an individual then she will also learn the sensitive attribute (the value of the attribute Cancer), since the relation between genomic sequence and phenotypes are not eliminated. Therefore, a model provides privacy protection if it prevents revealing genome sequence and also thwarts inference attack against victim's sensitive value.

Definition 1—We consider a genomic dataset (DB) secure if,

- It does not reveal the original genome sequence.
- Given a victim's genome sequence g_{victim} , an adversary $\mathcal{A}$ after observing the DB cannot guess better than random ($1/2$) the value of the sensitive attribute.

In this paper, we assume that the cloud is *semi-honest* and *non-colluding* with other parties. A party is semi-honest when it follows the protocol. However, it might be curious to gain more information from the messages of the protocol execution. Non-colluding means a party does not collude with others to derive more information. Please see [17] for a formal definition regarding semi-honest adversary model.

C. Homomorphic Encryption

Homomorphic encryption allows one to compute any function on encrypted data without having the secret key. The scheme was defined soon after RSA in 1978 [18] but was in theory for 30 years. The scheme in a nutshell is: if $c_1 = \xi(m_1)$ and $c_2 = \xi(m_2)$ (where m_1 and m_2 are the plaintexts, c_1 and c_2 are the ciphertexts and $\xi = \text{Any randomized encryption}$), we can compute any function on c_1 and c_2 and get the same result as if we were computing with m_1 and m_2 .

In our proposed framework we use an additive homomorphic encryption scheme called *Paillier cryptosystem* [19] to do linear operations on ciphertexts. Paillier encryption has the following properties that are useful for the proposed method:

- *Probabilistic*: For any arbitrary message $m = 1$, there is a negligible probability that $\xi(1) = \xi(1)$.
- *Addition homomorphism*: For any public key p_k and arbitrary messages m_1, m_2 ,

$$\xi(m_1) + \xi(m_2) = (\xi(m_1) * \xi(m_2)) \bmod p_k^2$$

which denotes that we can do an efficient addition operation over ciphertexts.

IV. Valid Permutations

In this paper, we use a permutation technique over genome sequences to prevent the cloud from getting meaningful information about any genome sequences. Any permutation over a set with n elements can be represented as follows,

$$(i_1, i_2)(i_3, i_4) \cdots (i_{n-1}, i_n), i_j \in \{1, 2, \dots, n\} \quad (1)$$

We cannot use any permutation because of the available background knowledge regarding genome sequences. An adversary can use the background knowledge about a specific position and its allele frequency to undo a permutation. Further discussion on this is available in the supporting document.

For example, if the proxy applies (1, 4) permutation to Table I, background knowledge of allele frequency will reveal this permutation due to different frequencies. However, the permutation (1, 2) is a valid permutation because both the first and the second column have same nucleotide set (i.e., 'A' and 'T') and same nucleotide-frequency ('A' and 'T' appear with half probability).

Definition 2

A permutation over genomic sequences is called valid if it swaps columns with same nucleotide set having same frequency. The set of all of the valid permutation is denoted by $\mathcal{VP}$.

If valid permutations are applied, the permuted genome sequences will inherit statistical features of the original data. Therefore, no adversary can infer information about the permutation. Obviously, $\mathcal{VP}$ is strongly related to background knowledge about genomic sequences. For example, in Table I, the first and the second column can be swapped. Similarly, third and fourth columns can be swapped. Therefore, we have three valid permutations that can be applied $\mathcal{VP} = \{(1, 2), (3, 4), (1, 2)(3, 4)\}$ for Table I.

V. Proposed Model

In this section, we present our proposed model. The dataset is first preprocessed by the proxy and then stored at the cloud. The cloud uses the stored database to execute queries.

A. Data Preparation

Proxy receives genomic dataset (i.e., Table I) from the data provider. Data preparation phase has two steps: *Valid Permutation* and *Adding Fake Records*.

1. *Permutation*: Proxy permutes genome sequences according a valid permutation π ($\pi \in \mathcal{VP}$) as described in Section IV. Proxy then generates a public-private key pair for the homomorphic encryption $E(\cdot)$ and computes $(E(1), \pi(g), A_g)$ for each record where A_g is the sensitive attribute like disease association. Suppose the chosen valid permutation is (1, 2), then we get Table II from Table I.

2. *Adding Fake Records:* The second privacy requirement is to remove inference attack on sensitive attribute (e.g., disease) using the knowledge of the genomic sequence of a victim. We add fake genomic sequences to permuted sequences to achieve this objective.

Proxy adds fake records to original dataset to disassociate the relationship between genomic sequence and sensitive attribute. For any $g \in DB$, the proxy adds $(E(0), g, YES)$ or $(E(0), g, NO)$ to the dataset. The encryption of the tag 0, denoted by $E(0)$, indicates that this record is a fake record. All records are shuffled randomly before sending data to the cloud to hide fake records from original ones. The modified dataset after adding fake records as shown in Table III which will be shuffled and then sent to cloud for query execution.

B. Query Execution

The cloud is able to execute count and top- k queries on the stored database. Following, we elaborate the procedure for query execution.

1) Count Query

1. Researchers submit a count query as follows:

```
SELECT * FROM Sequences
WHERE POS_1='A' AND POS_2='T'
AND CANCER='YES'
```

2. Proxy translates this query to

```
SELECT * FROM Sequences
WHERE POS_1='T' AND POS_2='A'
AND CANCER='YES'
```

where $\pi(1) = 2$ and $\pi(2) = 1$.

3. After getting translated query, the cloud computes the output as follows:
 - a. $ANS = E(0)$.
 - b. For $(E(\theta_g), g, A_g) \in DB$ if g and A_g satisfies the translated query then $ANS = ANS + E(\theta_g)$, where $\theta_g \in \{0, 1\}$.
4. Cloud sends ANS to Proxy.
5. Proxy decrypts ANS and delivers the answer to the researcher.

The cloud gains no information about the output of the query as it is encrypted by a semantically secure encryption system.

Example. After receiving the translated query ($POS_1 = T$ and $POS_2 = A$ and $CANCER = YES$), the cloud initializes $ANS = E(0)$. In Table III, record number 2, 6 and 8 match the query predicate. Cloud adds their tags to $ANS = ANS + E(0) + E(0) + E(0) = E(0)$ and gives

ANS to the proxy. Proxy after decryption obtains the answer (which is zero) and finally sends it back to researcher.

2) Top-k Query

1. Researchers submit a top- k query as follows:

```
SELECT * FROM Sequences
WHERE POS_1='A' AND POS_2='T'
LIMIT k
```

where k is the number of records.

2. Proxy translates the query to

```
SELECT * FROM Sequences
WHERE POS_1='T' AND POS_2='A'
LIMIT n
```

where $\pi(1) = 2$, $\pi(2) = 1$ and n is security parameter and have no relation with k .

3. Cloud finds n nearest records and sends to the proxy.
4. Proxy decrypts tags of these records to verify if there are k original records. If there are less than k original records, proxy sends a message to the cloud to send the next n nearest records.
5. Proxy repeats this procedure until it receives k nearest original records.
6. After getting k records, proxy sends the records to the researcher.

The variable n is introduced to hide the variable k (required limit) from the cloud. In each iteration, the cloud sends n records containing both fake and real records. Therefore, the cloud does not know exactly what is the real value of k .

Example. Suppose in a top- k query ($POS_1 = T$ and $POS_2 = A$), the values for the parameters k and n are 5 and 3, respectively. Hence, the cloud finds 3 nearest records that are sent back to proxy. These records are # 1, 2 and 5 in Table III. After decryption of the tag, proxy obtains two (1, 5) original record. As its a top-5 query, it requests for the next three nearest records which are # 6, 7 and 8. After proxy decrypts these values, it gets only one original record. Therefore, the proxy asks for 3 more records which the cloud does not have. At this stage, proxy obtains k (or less) number of original records according to the query. Finally, it sends 3 records # 1, 5 and 7 to the researcher.

VI. Privacy Discussion

In this section, we discuss the privacy analysis of the proposed method. We provide a sketch of how this model thwarts privacy attacks and present the formal proofs in the Appendix.

A. Preventing Cloud to Retrieve Genomic Sequences

We mentioned two major privacy concerns. The first one is that the genomic sequence should not be revealed to the cloud. To achieve this requirement, proxy applies a valid permutation on genome sequences. The following theorem shows why applying a valid permutation prevents the cloud from gaining original genome sequences.

Theorem 1—Given a permuted genomic database DB and for any two possible valid permutation keys $\pi_1, \pi_2 \in \mathcal{VP}$, we get,

$$\Pr(\pi_{key}=\pi_1|DB)=\Pr(\pi_{key}=\pi_2|DB)$$

where π_{key} is the chosen key for this permuted database DB .

The proof of this Theorem in the supplementary notes. Therefore, permutation is irreversible and it is impossible for an observer to rearrange the columns and retrieve the original genome sequence.

B. Removing Disease Association

The second security requirement is to eliminate the association between a genomic sequence and the corresponding sensitive attribute (e.g., disease). We show that cloud cannot exploit relation between genomic sequences and sensitive attribute.

Theorem 2—Suppose a genomic sequence g is given. An adversary $\mathcal{A}$ after observing DB cannot guess better than half the sensitive value of g .

The formal proof is provided in the supplementary notes. Suppose, Alice's (victim) genomic sequence g_A is known to adversary (i.e., cloud). This does not help the adversary to gain knowledge regarding the sensitive attribute of Alice. This is because according to the Theorem 1, the adversary cannot gain information about the permutation. Moreover, the proxy disassociates the relationship between genomic sequence and sensitive attribute by adding fake sequences. Thus, an adversary finds that half of possible candidate records for Alice genomic sequence have cancer while the other half do not. Therefore, observing even the full database does not help the adversary to obtain Alice's disease *better than a random guess*.

C. Limitation

However, this framework is still vulnerable to Homer et al.'s attack [16] as we preserve the frequency of individual position. As we use commercial cloud platforms for data storage or computation, the risk of re-identification is not totally mitigated. In other words, any malicious cloud can re-identify specific person (if it has the original sequence) but still cannot guess better than random about the disease association. Since we do not encrypt the whole sequence for efficiency, this vulnerability remains for the malicious cloud (although we assume the cloud to be semi-honest in section II).

We assume for Theorem 1 that the polymorphic sites are independent. In real life, certain SNPs are correlated and might be utilized together to infer the valid permutation key. However, the risk from linkage disequilibrium between two different sites is hard to quantify in general and permutation on the original sequence impacts this as well. Thus, it is not clear how an adversary can use this information to undo a permuted database.

VII. Implementation

In this section, we experimentally evaluate the performance of the proposed model. To simulate the real-life scenario, we placed the proxy and the cloud in two different machines located in two different geographic regions. For the cloud, we use Amazon EC2 in Oregon U.S. with t2.micro configuration. The proxy was on a separate machine (Ram 8GB, Intel i5 3.3 Ghz) located in Winnipeg, Canada. The source code is available publicly at <https://github.com/mominbuet/PermutationGenomicData.git>.

We consider the following aspects in order to assess the efficiency of the our proposed method.

- *Query time*: Time needed to answer any count or ranked query.
- *Insertion time*: Time needed to process genomic database by the proxy.
- *Space complexity*: Storage space needed to store the database (both original and fake records) in the cloud.

We also compare our proposed method with a non-secure (i.e., no permutation and addition of fake records) scheme and the method proposed by Kantarcioglu et al. [6]. Their model is objective-wise similar to ours as it answers count query but uses homomorphic encryption to encrypt the entire database.

We use both real-life and synthetic data sets for evaluating our model. The real-life data is taken from iDash competition 2015 [20] where there are 311 different SNPs of 400 different participants divided into case and control groups. We use the case-control group separation to denote the value of the sensitive attribute *Cancer*. For synthetic data, we took the allele frequency of CHB, CHS, JPT and MXL population from *1000genomes* dataset (August 2010 Release) and generated 60,000 data rows according to that frequency.

We evaluate our method using the real-life data in Section VII-B. The comparison is done for both secure and non-secure versions. For non-secure version, we do computations on plaintext to get the run time for operations. We call this model as *regular model*. We then perform the same operations for the proposed model and report their run times. For synthetic dataset, we use 5K, 10K, 15K and 20K records to compare the model with the regular and the Kantarcioglu et al.'s model [6] (VII-C).

A. Insertion Time and Required Space

Two important factors in efficiency are insertion time and required space for the data. The time required for processing the raw data (provided by the data owners) and storing them in

the cloud is called insertion time. Table IV shows insertion time and required space for different models.

Unfortunately, the insertion time is not reported in Kantarcioglu et al's paper [6]. However, their method offers stronger security by encrypting the whole database and the expanding factor is bigger due to the encryption scheme (the size of each ciphertext is 1024 bits). Therefore, we can conclude that it has significantly higher insertion time than our scheme. Our model's insertion time is almost *four times* greater than that of regular model. Moreover, Table IV shows that the proposed model requires double the space of the regular model because of fake genomic sequences.

B. Query time for real-life data

The run times for count and top- k queries over the real-life data are shown in Figures 2 and 3.

According to the balancing technique, the size of database doubles due to the addition of fake records. Therefore, the required time for query execution also doubles for the additional records. In addition, the proxy needs to do small amount of work to get the final result. Figure 2 shows that our proposed model pays a small penalty in terms of execution time to achieve security.

Figure 3 shows the query time for top- k query. With bigger security parameter n , it takes lesser time to answer queries. This happens because the proxy asks for more records in each time resulting less number of iterations with the cloud.

C. Query time for synthetic data

In this section we present our experimental results for 5K, 10K, 15K and 20K records. Figure 5 shows the results for count query on different data sizes.

Experimental results show a big difference in running time of our model compared to the existing method (logarithmic scale). This is because the existing technique encrypts the whole database using homomorphic encryption while we only encrypt the tag (original/fake). Homomorphic encryption imposes heavy computational cost on the cloud effecting both running times and storage requirement.

Second type of query is top- k query. To the best of our knowledge, the problem of secure top- k query on outsourced databases has not yet addressed. Recent work addresses the problem of secure edit distance [21] which is related our problem; however, they have a different architecture and do not provide a benchmark for genomic data.

We also evaluated the scalability of the proposed method by varying the number of SNPs. The query time does not depend on the number of SNPs. Due to the space constrain, we report these results in the supplementary document.

VIII. Conclusion

In this paper, we propose a model for outsourcing genomic data to a public cloud. The proposed model ensures the security of outsourced data by permutation and addition of fake records. Cloud is able to evaluate two kinds of queries (count query and top- k) on the outsourced data. We demonstrate that storing genomic data using the proposed method does not reveal the identity of an individual and mitigates the risk of association between genome sequence and sensitive data against an adversary. We conducted extensive experiments and implemented the model using a public cloud. Experimental results on real-life and synthetic data demonstrate that the proposed model outperforms the existing technique in terms of query execution time and its performance is close to the insecure method.

Supplementary Material

Refer to Web version on PubMed Central for supplementary material.

Acknowledgments

We sincerely thank the reviewers for their insightful comments. The research is supported in part by the NSERC Discovery Grants (RGPIN-2015-04147) and University of Manitoba Startup Grant. Xiaoqian Jiang was partially funded in part by NLM (R00LM011392, R21LM012060).

References

1. Trinidad SB, Fullerton SM, Bares JM, Jarvik GP, Larson EB, Burke W. Genomic research and wide data sharing: views of prospective participants. *Genetics in Medicine*. 2010; 12(8):486–495. [PubMed: 20535021]
2. Ginsburg GS, Burke TW, Febbo P. Centralized biorepositories for genetic and genomic. *JAMA*. 2008; 299(11):1359–1361. [PubMed: 18349099]
3. Paltoo DN, Rodriguez LL, Feolo M, Gillanders E, Ramos EM, Rutter J, Sherry S, Wang VO, Bailey A, Baker R, et al. Data use under the nih gwas data sharing policy and future directions. *Nature genetics*. 2014; 46(9):934. [PubMed: 25162809]
4. Consortium GP, et al. A map of human genome variation from population-scale sequencing. *Nature*. 2010; 467(7319):1061–1073. [PubMed: 20981092]
5. Erlich Y, Narayanan A. Routes for breaching and protecting genetic privacy. *Nature Reviews Genetics*. 2014; 15(6):409–421.
6. Kantarcioglu M, Jiang W, Liu Y, Malin B. A cryptographic approach to securely share and query genomic sequences. *Information Technology in Biomedicine, IEEE Transactions on*. 2008; 12(5): 606–617.
7. Canim M, Kantarcioglu M, Malin B. Secure management of biomedical data with cryptographic hardware. *Information Technology in Biomedicine, IEEE Transactions on*. 2012; 16(1):166–175.
8. Al Aziz, MM., Hasan, MZ., Mohammed, N., Alhadidi, D. Secure and efficient multiparty computation on genomic data. *ACM*; 2016. p. 278-283.
9. Bos JW, Lauter K, Naehrig M. Private predictive analysis on encrypted medical data. *Journal of biomedical informatics*. 2014; 50:234–243. [PubMed: 24835616]
10. Shimizu K, Nuida K, Rätsch G. Efficient privacy-preserving string search and an application in genomics. *bioRxiv*. 2015:018267.
11. Wagner J, Paulson JN, Wang X, Bhattacharjee B, Corrada Bravo H. Privacy-preserving microbiome analysis using secure computation. *Bioinformatics*. 2016
12. Kahn SD, et al. On the future of genomic data. *Science(Washington)*. 2011; 331(6018):728–729. [PubMed: 21311016]

13. Contreras JL. Nih's genomic data sharing policy: timing and tradeoffs. *Trends in Genetics*. 2015; 31(2):55–57. [PubMed: 25620797]
14. Mount, DW., Mount, DW. *Bioinformatics: sequence and genome analysis*. Vol. 2. Cold spring harbor laboratory press; New York: 2001.
15. Gymrek M, McGuire AL, Golan D, Halperin E, Erlich Y. Identifying personal genomes by surname inference. *Science*. 2013; 339(6117):321–324. [PubMed: 23329047]
16. Homer N, Szleinger S, Redman M, Duggan D, Tembe W, Muehling J, Pearson JV, Stephan DA, Nelson SF, Craig DW. Resolving individuals contributing trace amounts of dna to highly complex mixtures using high-density snp genotyping microarrays. *PLoS Genet*. 2008; 4(8):e1000167. [PubMed: 18769715]
17. Goldreich, O. *Foundations of cryptography: volume 2, basic applications*. Cambridge university press; 2009.
18. Rivest RL, Adleman L, Dertouzos ML. On data banks and privacy homomorphisms. *Foundations of secure computation*. 1978; 4(11):169–180.
19. Paillier, P. *Advances in cryptology, EUROCRYPT99*. Springer; 1999. Public-key cryptosystems based on composite degree residuosity classes; p. 223-238.
20. Idash-privacy and security workshop on genomic data. 2015. <http://www.humangenomeprivacy.org/2015/competition-tasks.html>
21. Wang, XS., Huang, Y., Zhao, Y., Tang, H., Wang, X., Bu, D. Efficient genome-wide, privacy-preserving similar patient query based on private edit distance. *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*; ACM; 2015. p. 492-503.

Fig. 1. Main architecture of the proposed scheme is consisted of four entities: researchers, cloud, proxy and data providers

Fig. 2.

Running time (in seconds) of count query over real data with different query sizes

Fig. 3. Running time (in seconds) of top-k over real data with different security parameters where $k = 100$

Fig. 4.

Implementing model over a synthetic database with 20k records. In this implementation K is equal to 100 and fixed. The X-axis shows different security parameter for top- k query.

Fig. 5.

These tables show the results of implementation the model over four synthetic databases with 5k, 10k, 15k and 20k records. X-axis shows the query size and the Y-axis shows running time for addressing any query (logarithmic scale). These figures demonstrate that running time for count query is more or less twice the required time in regular model while the tables shows the difference in answering queries from Kantarcioglu et al's work [6]

TABLE I

Raw genomic data

Genomic Sequence										Phenotype
Rec #	POS ₁	POS ₂	POS ₃	POS ₄	POS ₅	Cancer				
1	A	T	T	G	C	No				
2	T	A	T	G	C	Yes				
3	A	T	G	T	C	No				
4	A	T	T	T	G	No				
5	T	A	G	T	C	No				
6	T	A	T	T	G	No				
Freq.	T=0.5	T=0.5	T=0.67	T=0.67	C=0.67	-				
	A=0.5	A=0.5	G=0.33	G=0.33	G=0.33					

TABLE II

Permuted genomic data by valid permutation (1,2) from Table I

Rec #	Tag	Permuted Genomic Sequence					Cancer
1	E(1)	T	A	T	G	C	No
2	E(1)	A	T	T	G	C	Yes
3	E(1)	T	A	G	T	C	No
4	E(1)	T	A	T	T	G	No
5	E(1)	A	T	G	T	C	No
6	E(1)	A	T	T	T	G	No

Dataset after adding fake records (indicated by white rows) which is shuffled and sent to the cloud for storage and computation

TABLE III

Rec #	Tag	Permuted Genomic Sequence					Cancer
1	E(1)	T	A	T	G	C	No
2	E(0)	T	A	T	G	C	Yes
3	E(1)	A	T	T	G	C	Yes
4	E(0)	A	T	T	G	C	No
5	E(1)	T	A	G	T	C	No
6	E(0)	T	A	G	T	C	Yes
7	E(1)	T	A	T	T	G	No
8	E(0)	T	A	T	T	G	Yes
9	E(1)	A	T	G	T	C	No
10	E(0)	A	T	G	T	C	Yes
11	E(1)	A	T	T	T	G	No
12	E(0)	A	T	T	T	G	Yes

TABLE IV

Insertion Time and Required Space for different data size

Model	Data Size	Insertion min.	Expanding Factor
Regular	400	2	1
	5K	70	
	10K	135	
	15K	213	
	20K	270	
Kantarcioglu [6]	Any	–	1024
Our Model	400	8.7	2
	5K	257	
	10K	511	
	15K	763	
	20K	1102	