Computer Methods and Programs in Biomedicine 165 (2018) 129–137

Contents lists available at ScienceDirect

Computer Methods and Programs in Biomedicine

journal homepage: www.elsevier.com/locate/cmpb

Secure large-scale genome data storage and query

Luyao Chen[a,1], Md Momin Aziz[b,1,*], Noman Mohammed[b], Xiaoqian Jiang[c]

[a] *Heinz College, Carnegie Mellon University, United States*
[b] *Computer Science, University of Manitoba, Canada*
[c] *School of Biomedical Informatics, University of Texas Health Science Center at Houston, United States*

ARTICLE INFO

Article history:
Received 5 March 2018
Revised 26 July 2018
Accepted 8 August 2018

Keywords:
Secure genome data storage
Graph database
Secure computation on genome data
Homomorphic encryption
Genome data storage Neo4j

ABSTRACT

Background and Objective: Cloud computing plays a vital role in big data science with its scalable and cost-efficient architecture. Large-scale genome data storage and computations would benefit from using these latest cloud computing infrastructures, to save cost and speedup discoveries. However, due to the privacy and security concerns, data owners are often disinclined to put sensitive data in a public cloud environment without enforcing some protective measures. An ideal solution is to develop secure genome database that supports encrypted data deposition and query.
Methods: Nevertheless, it is a challenging task to make such a system fast and scalable enough to handle real-world demands providing data security as well. In this paper, we propose a novel, secure mechanism to support secure count queries on an open source graph database (Neo4j) and evaluated the performance on a real-world dataset of around 735,317 Single Nucleotide Polymorphisms (SNPs). In particular, we propose a new tree indexing method that offers constant time complexity (proportion to the tree depth), which was the bottleneck of existing approaches.
Results: The proposed method significantly improves the runtime of query execution compared to the existing techniques. It takes less than one minute to execute an arbitrary count query on a dataset of 212 GB, while the best-known algorithm takes around 7 min.
Conclusions: The outlined framework and experimental results show the applicability of utilizing graph database for securely storing large-scale genome data in untrusted environment. Furthermore, the crypto-system and security assumptions underlined are much suitable for such use cases which be generalized in future work.

1. Introduction

Over the past decade, different technical breakthroughs have made genome sequencing more affordable. The next generation sequencing techniques made this growth somewhat exponential as we are starting to observing datasets in volume of petabytes [1]. This increasing availability of genome data of different individuals gives us an opportunity to zoom into the micro level and analyze the complex correlation or causation. However, this is deeply challenging due to the size of the data, computational complexity, and inherent privacy issues.

As mentioned earlier, the immense size of genome data comes at a price of higher storage space. An economical solution will be leveraging the cost-efficient commercial cloud computing services (i.e., Amazon EC2, Microsoft Azure, or Google Cloud Platform, etc.) to host data and conduct required analysis on demand. For example, Amazon S3 and Azure Storage Services charge only $0.0208 to store 50 terabytes on a monthly base [2,3]. More importantly, these cloud services also reduce the operational costs of running large scale experiments on such large-scale data.

Surely the commercial cloud services can provide a cost-effective and efficient solution to the ongoing genome data storage and computation issues. However, the privacy of these records is another notable aspect as public (/unrestricted) access of genome data might lead to re-identification attacks [4], surname recovery [5], facial and voice traits reconstruction [6,7]. Thus, genome data are highly sensitive because they are irrevocable and have stigmatizing consequences to both the individuals and their family, particularly first-degree relatives [8]. There are some surveys that demonstrate and discuss these privacy and security issues [9,10].

Due to these concerns and reported vulnerability of the public cloud [11], data custodians are not comfortable in depositing sensitive genome data in a third-party environment (untrusted) without enforcing necessary protection [12]. An ideal approach is to develop a secure genome database, i.e., encrypting the data and

* Corresponding author.
E-mail address: azizmma@cs.umanitoba.ca (M.M. Aziz).
1 Work done during an internship at Department of Biomedical Informatics, University of California San Diego.

https://doi.org/10.1016/j.cmpb.2018.08.007
0169-2607/© 2018 Elsevier B.V. All rights reserved.

providing a security layer on top of the operations interface for safeguarding the data analysis process. Assuming the cloud service provider is semi-honest (honest but curious [13]), and we only want to protect the data from external malicious users, data custodians can run queries on the encrypted data without establishing a complete, trusted relationship.

However, this computation on encrypted data induces a cost on performance as these security primitives are not efficient as their plaintext counterparts. Scalability is another challenge as large memory consumption imposed by these security protocols might hinder the practicability of a realistic system. Thus, in this paper, we look into the balance between privacy and efficiency of the computation of genome data. We consider the count query operation which is the building block for various statistical analysis on genome data. A count query procedure to obtain the number of individuals satisfying a SQL-like query can be represented as:

```
SELECT count(∗) FROM Sequences
    WHERE SNP1 = ‘A′ AND SNP2 = ‘T′ AND...
    AND Disease = Yes
```

Single Nucleotide Polymorphism (SNP) refers to a variation of a single position on a DNA sequence (of a certain individual) such that more than 1% of the population does not carry the same value. Although not all SNPs correspond to disorders, some of them are known to be associated with some diseases. A count query between SNPs and a specific condition is the first step to explore the correlations and serves as the building block for genome-wide association studies (GWAS).

1.1. Contributions

In this paper, we propose a framework that provides better scalability and handles security issues of large-scale computations on genome data outsourced (transferred/stored) to a third party, public cloud server. Furthermore, we utilized a homomorphic cryptographic combined with Garbled Circuit scheme to ensure the security and tree structure to represent the arbitrary genome data for computational efficiency. The major contributions of the paper can be summarized below:

- We propose a method utilizing graph-based database to store and allow computations on real-world genome data *securely*.
- A novel indexing scheme is proposed on such database to make the secure query operations more efficient.
- We test the proposed approach along with the corresponding indexing scheme on a large-scale genome dataset containing 735,317 human SNPs ($\sim$ 200 GB data).
- Experimental results show that it takes less than a minute for a query compared to best-known attempts where it required around 7 min [14,15].

The rest of the paper are organized as follows. Necessary backgrounds are discussed in Section 2. We discuss the proposed methods in Section 3 and show the results in Section 4. In Section 6 we discuss some of the related work. Finally, we conclude and discuss some future works in Section 7.

2. Preliminaries

In this section, we introduce some of the concepts (related to cryptography and genome data) required in understanding the proposed method.

2.1. Data representation

In this paper, we consider the *Single Nucleotide Polymorphism* (SNP) of human DNA and its association with specific disease. For example, a mutation in BRCA1/2 genes has been reported to be associated with breast cancer. A variant in BRCA1 is *rs1799950* is one of *25 SNPs* to express an increased risk for breast cancer [16]. We considered such a SNP dataset with has a specific disease association. The data is represented in Table 1.

Table 1
Considered genomic data containing multiple patients and corresponding SNPs.

	Genomic sequence					Phenotype
Patient	SNP_1	SNP_2	SNP_3	...	SNP_5	Disease
1	A	T	G		C	Yes
2	T	C	C		G	No
3	A	T	C		C	No
4	A	C	C		C	Yes

2.2. Graph database

Graph database uses different interconnected graph compositions to represent the data. In contrast to relational (traditional) database, graph database considers data points as the nodes and the relation between them as edges. This approach has proved much useful [17,18] in different literature and use cases as most of the relational data can be represented as a hierarchical data where one record is closely related to another. Graph database consists of nodes and edges where the nodes are interconnected with edges. Furthermore, there might be directional edges defining the connectivity of the nodes, though for simplicity we will only consider the non-directional edges throughout the rest of the paper. Regardless of the directions, the edges usually represent the relation between the nodes. In Fig. 1 we depict the difference between a relational and graph database.

Formally, in a graph database (compared to relational tables), there are *relationships* which connect the *entities*. These entities can have specific properties. The relationships commonly described by verbs, for example, a patient ‘get’ certain conditions or a patient ‘has’ many SNPs. A relationship also has properties, for instance, the property ‘has’ describes the detail data of SNP.

2.3. Homomorphic encryption

Homomorphic encryption (HE) is an encryption scheme which allows computations under encryption. For example, consider two numeric values 2, 3 and the resulting homomorphic encryption are two random numbers $E(2)$ and $E(3)$. The result of $E(2) + E(3)$ will be the same as $E(5)$.

In this work, we utilized Paillier encryption [19] which has this additive property. However, there are other HE schemes with additive and/or multiplicative functionality. Regardless, we only need the additive property for our method and opted for this simple HE scheme.

2.4. Garbled Circuit

In 1986, Yao proposed Garbled Circuit (GC) which establishes a two-party protocol which allows the secure execution of an arbitrary Boolean function f(x,y) against semi-honest adversaries [20]. Here, x and y are two inputs from two individual parties, and they are kept secret from each other while the output of $f(x, y)$ is disclosed. For example, in the millionaire problem, Alice and Bob want to know who has more money. They engage in a GC protocol where x and y are their net worth, respectively. The output will be a Boolean value representing $f(x, y) = x > y$. If the value is one then $x > y$ (denotes that Alice is richer) and vice versa.