International Journal of Emerging Technology in Computer Science & Electronics (IJETCSE) ISSN: 0976-1353 Volume 25 Issue 5 – APRIL 2018 (SPECIAL ISSUE).

PROVIDING SECURITY IN CLOUD FOR GENOMIC SEQUENCE

Mrs. L. Indu[#1], M. Harini[*2], J. Divya Lakshmi[*2], P. Lavanya[*3]

Assistant Professor; P.B.College of Engineering,Chennai.

[#1] *UG Scholars; P.B.College of Engineering; : harinimani23@gmail.com,Chennai.*

[#2]*UG Scholars; P.B.College of Engineering,Chennai.*

[#3]*UG Scholars; P.B.College of Engineering,Chennai.*

ABSTRACT- This paper watches out for the issue of sharing individual specific genomic progressions without slighting the assurance of their data subjects to help broad scale biomedical research wanders. The proposed procedure develops the framework. However, extends the results in different ways. One change is that our arrangement is deterministic, with zero probability of a wrong answer (instead of a low probability). We in like manner give another working point in the space-time tradeoff, by offering an arrangement that is twice as brisk as theirs however uses twofold the storage space. This point is impelled by how limit is more affordable than figuring in current circulated processing evaluating plans. Likewise, our encoding of the data makes it plausible for us to manage a wealthier plan of inquiries than revise organizing between the request and each gathering of the database.

INTRODUCTION

DNA or Deoxyribonucleic Acid is the medium of longterm storage and transmission of genetic information for all modern living organisms. Human DNA data (DNA sequences within the 23 chromosome pairs) are private and sensitive personal information. However, such data is critical for conducting biomedical research and studies, for example, diagnosis of pre-disposition to develop a specific disease, drug allergy, or prediction of success rate in response to a specific treatment. Providing a publicly available DNA database for fostering research in this field is mainly confronted by privacy concerns. Today, the abundant computation and storage capacity of cloud services enables practical hosting and sharing of DNA databases and efficient processing of genomic sequences, such as performing sequence comparison, exact and approximate sequence search and various tests. What is missing is an efficient security layer that preserves the privacy of individuals' records and assigns the burden of query processing to the cloud. Whereas anonymization techniques such as de-identification [2], data augmentation [3], or database partitioning [4] solve this problem partially, they are not sufficient because in many cases, re-identification of persons is possible [5]. It follows that the DNA data must be protected, not just unlinked from the corresponding persons. In this paper, we consider the framework proposed in [1] where the DNA records coming from several hospitals are encrypted and stored at a data storage site, and biomedical researchers are able to submit aggregate counting queries to this site. Counting queries are particularly interesting for statistical analysis. This paper provides a new method that addresses a larger set of problems and provides a faster query response time than the technique introduced in [1]. Our approach is based on the fact that, given current pricing plans at many cloud services providers, storage is cheaper than computing. Therefore, we favor storage over computing resources to optimize cost. Moreover, from a user experience point of view, response time is the most tangible indicator of performance; hence it is natural to aim at reducing it. Our method enhances the state of the art at both the conceptual level and the implementation level. More concretely:

- At the conceptual level, we provide a deterministic scheme, with zero probability of a wrong answer.
- We also provide a new operating point in the space-time tradeoff, by giving a scheme that is twice as fast as theirs but uses twice the storage space.
- Moreover, our encoding of the data makes it possible for us to handle a richer set of queries than exact matching between the query and each sequence of the database, including:

i. Counting the number of matches between the query symbols and a sequence;

ii. Logical OR matches where a query symbol is allowed to match a subset of the alphabet thereby making it possible to handle (as a special case) a “not equal to” requirement for a query symbol (e.g., “not a G”);

iii. Support for the extended alphabet of nucleotide base codes that encompasses ambiguities in DNA sequences (contrary to the previous item this happens on the DNA sequence side instead of the query side);

iv. Queries that specify the number of occurrences of each kind of symbol in the specified sequence positions (e.g., two ‘A’ and four ‘C’ and one ‘G’ and three ‘T’, occurring in any order in the query-specified sequence positions);

v. A threshold query whose answer is ‘yes’ if the number of matches exceeds a query-specified threshold (e.g., “7 or more matches out of the 15 query-specified positions”).

vi. For all query types we can hide the answers from the decrypting server, so that only the client learns the answer.

vii. In all cases the client deterministically learns only the query's answer, except for query type

II. RELATED WORK

There is no universal method to create a protocol for secure multi-party computation and handling aggregate queries on encrypted data is not an exception. Several homomorphic systems only support a subset of mathematical operations, like addition (Paillier [19], Benaloh [23]), multiplication (ElGamal [24], RSA [25]), or exclusive-or (Goldwasser and Micali [26]). From a security perspective, only the additive Paillier and the multiplicative ElGamal are classified to be IND-CPA (stands for indistinguishability under chosen plaintext attack) [27]. Partially homomorphic cryptosystems are more desirable from a performance point of view than somewhat homomorphic cryptosystems, which support a limited operation depth.

A. *Forensic databases*

In a forensic database, a suspect record has to be tested against an entire database. A record of the database can be decrypted only if it matches the suspect record. This protects the other records from being unveiled [7]. Similarly, negative databases prevent the enumeration of its members by reversely saving the non-members, in a compressed form [8].

B. Profile matching

In [9] the authors address a multitude of tests such as identity, ancestry and paternity tests based on Short Tandem Repeat (STR) profiles. The STR profile is composed of a number of loci and for each locus the number of repetitions for a given repeat structure. The authors translate each test into an algebraic expression and provide a homomorphic encryption scheme allowing two semi-honest parties to compare their stored profiles in a semantically secure manner. The proposed approach allows exact answers or small error tolerance as practically required by the tests.

C. Sequence comparison

The edit distance is the optimal cost of insertion, deletion and substitution of characters to go from a sequence λ to a sequence μ. The edit script is the chart of the steps leading to the optimal edit distance. Atallah et al. [10] offers a solution for securely outsource a dynamic programming solution for finding the edit distance and the edit script for two given sequences (particularly genomic sequences with small alphabet size). The outsourcing protocol is based on two noncolluding (honest-but-curious) agents that securely collaborate to performing table lookup and minimum finding. The secure minimum finding protocol determines the minimum of an additively split vector based on Yao’s garbled circuits and a blind-and-permute protocol for hiding the index of this minimum. In [11] their scheme has been improved for performance and requires space only linear in the input size. The work in [12] addresses a similar dynamic programming solution for finding the longest common subsequence. By using the “four Russians” technique in a new way, the authors propose a communication-efficient SMC protocol that improves over the generic solution based on Yao’s garbled circuits.Their scheme, based on computation with obscured data, preserves a reasonable level of accuracy but does not provably protect the privacy of the inputs.

D. Sequence testing by finite automata

Sometimes the queries on DNA need to take into account various errors such as irrelevant mutations, incomplete specifications and sequencing errors. Therefore, the pattern of the query should be expressed using regular expressions. Many works address practical and privacy-preserving outsourcing of this regular expression type of queries, implemented as oblivious evaluation of finite automata [15]– [17].

E. Aggregate queries

For biomedical researchers, important queries have often the form "How many records contain a diagnosis of Alzheimer disease and gene variant X?" Secure outsourcing of the database and allowing such type of queries without requiring the server to decrypt the data has been addressed in [1]. The paper presents very practical results. For example, a count query over 40 records in a database of 5000 records takes 30 minutes. Our paper extends these results by proposing a variant storage and computation scheme.

III. ARCHITECTURE AND MODELLING

Computer scientists often represent DNA by a large sequence of characters from the alphabet $\Sigma = \{A,C,G,T\}$, representing the four nucleotide types. This alphabet can be augmented with additional characters representing ambiguity in the sequence. This extended alphabet is denoted by $\Sigma' = \{A,C,G,T,N,M,R,W,S,Y,K,V,H,D,B\}$ as defined by IUPAC [18], see Table 1. Given a database of d sequences $s1,s2, \ldots,sd$ each having m characters; the query is represented as a list of tuples (ji,vi) of characters vi and positions ji; for $i = 1..k$. The result of the query is the number of sequences where $s[ji] == vi$ for all the tuples (ji,vi). The pseudo (Python-like) code of a query in clear is shown in Listing 1.

NUCLEOTIDE BASE CODES (IUPAC)

Symbol	Nucleotide Base
A	Adenine
C	Cytosine
G	Guanine
T	Thymine
N	A or C or G or T
M	A or C
R	A or G
W	A or T
S	C or G
Y	C or T
K	G or T
V	Not T
H	Not G
D	Not C
B	Not A

In our model, hospitals who have DNA sequences do not have the computing and processing capabilities to process researchers' requests, so they all store their DNA sequences at a server (which is also more convenient to do queries across all hospitals). The clients, who are typically researchers, query the server to obtain statistics on the occurrence of a given subsequence in the pool of DNA sequences stored on the server.

LISTING 1
PSEUDO-CODE OF AN AGGREGATE QUERY

```
#Example of a query:
q=[(A,0), (A,2), (C,3), (G,6)]
#Example of sequence matching the query:
#AAACAGG
#D is the set of all sequences
count=0
for s in D:
  match=True
  for (v,j) in q:
    if s[j]!=v:
      match=False
      Break
  if(match):
    count+=1
```

To be more precise the security model is as follows:

- Hospitals want to protect the confidentiality of the DNA sequences that they own and no external party has the right to access these DNA sequences for privacy reasons.
- Additively homomorphic encryption is suitable for the purpose of performing count statistics on encrypted data. Paillier's homomorphic encryption [19] possesses the following properties: (i) It's a public key scheme, which means encryption can be performed by anyone who knows the public key, whereas decryption can only be done by the matching private key, known only to a trusted party. (ii) It is probabilistic. In other words, it is impossible for an adversary to tell whether two ciphertexts are encryptions of the same plaintext or not. (iii) It possesses the homomorphic properties for addition, in particular:
- $Epk((m1+m2) \ mod \ N)=Epk(m1)*Epk(m2) \ mod \ N2$
- $Epk((a*m1) \ mod \ N)=Epk(m1)a \ mod \ N2$

- We consider a framework similar to [1] composed of several hospitals, several clients representing biomedical researchers and two non-colluding servers (can be two different cloud providers, or one cloud provider and one trusted host). In Fig. 1 we call these two databases *database1* and *database2* to emphasize that the framework can be deployed in a cloud environment:
- Database1 represents the data store where all the encrypted DNA records are stored and is responsible of processing the queries.
- *Database2* is a trusted party that generates and holds the private and public keys of the homomorphic encryption scheme. In step 1 the public key is sent to the other parties. Database2 is later used as a decyption oracle and it also shares security associations with the *clients* in order to send them the results securely.
- The hospitals obtain the public key in order to encrypt their DNA records and upload them to *Database1* (step 2).
- A client representing a biomedical researcher submits a query to *Database1* (step 3). The database processes the query over the encrypted records and sends the results to *Database2* in order to be decrypted (step 4). Finally the client receives from *Database2* the decrypted count of matches (step 5) through a secure channel.

IV. STORAGE AND COMPUTATION SCHEMES

A.Summary of the scheme in [1]

The proposed protocol is based on a binary storage scheme. Each letter has a binary representation over two bits and each bit is encrypted using Paillier encryption. For example the letter ‘A’ is coded in binary as two bits 00. Similarly the query is translated to binary encoding. For example finding the letter ‘A’ at position 6 is equivalent to finding the bit 0 at position 12 in the encoded sequence and the bit 0 at position 13 in the encoded sequence. Therefore, the required storage capacity for a sequence is $2*m*2b$ where m is the length of the sequence and *2b* is the size for storing an encrypted value (b is the bit length of the key modulus). The query is computed as an algebraic expression that evaluates to an encryption of 0 for each record matching the query. Two random numbers are used in order to limit false positives. Without loss of generality, consider an encoded sequence and a query of the form $(ji,1)$, for $i=1..t$ and $(ji,0)$, for $i=t+1..2k$ where k is the length (i.e., number of letters) of the query; the server computes an expression of the form:

$$R(q,s)=((\Pi E(s[ji])\ ti=1)*E(-t))r1*(\Pi E(s[ji])\ 2ki=t+1)r0$$

Where r1 and r0 are random numbers. If s matches the query, the result of this expression is an encryption of zero with a high probability. The server sends a permutation of the results of expressions for all the sequences $s1,\ldots,sd$. The key holder decrypts and counts the zeros to obtain the result of the query. Note that the number of modular exponentiation required is equal to 2, in addition to $2k$ modular multiplications.

B. Our Schemes

We present two different schemes; the first one requires more storage capacity but provides better query response time than the second one.

1) Quaternary storage, quaternary query :

We encode a sequence $s=[Li]$, $i=1..m$, using four vectors:

- $sA=[1\ if\ Li=='A',\ 0\ otherwise,\ i=1..m]$
- $sC=[1\ if\ Li=='C',\ 0\ otherwise,\ i=1..m]$
- $sG=[1\ if\ Li=='G',\ 0\ otherwise,\ i=1..m]$
- $sT=[1\ if\ Li=='T',\ 0\ otherwise,\ i=1..m]$

For example, sequence “CCGATAT” is encoded as:

- $sA=[0,0,0,1,0,1,0]$
- $sC=[1,1,0,0,0,0,0]$
- $sG=[0,0,1,0,0,0,0]$
- $sT=[0,0,1,0,1,0,1]$

A query $q=(ji,vi)$, $i=1..k$ is decomposed into four queries, and represented by four vectors as follows:

- Initialize qA to a vector of m zeroes; then assign $qA[ji]=1\ if\ Li== 'A'$ in the query,$i=1..k$
- qC is a vector of m zeroes; $qC[ji]=1\ if\ Li== 'C',i=1..k$
- qG is a vector of m zeroes; $qG[ji]=1\ if\ Li== 'G',i=1..k$
- qT is a vector of m zeroes; $qT[ji]=1\ if\ Li== 'T',i=1..k$

The query simply needs to be as long as the position of the last element in the subsequence of the query (vk). The positions after that will all be automatically assumed as containing 0. The query is then computed over an encrypted sequence using the following equation:

$$R(q,s)=E(qAsA+qCsC+qGsG+qTsT)= E(\Sigma sA,jii,qA[ji]=1+\Sigma sC,jii,qC[ji]=1+\Sigma sG,jii,qG[ji]=1+\Sigma sT,jii,qT[ji]=1) =\Pi E(sLi,ji)qLi,jii=1..k$$

2) Ternary storage, quaternary query :

Since the presence of a letter in a given position of a sequence can be directly inferred by the absence of the three other letters, we can reduce the encoding to only three vectors:

- $sA=[1\ if\ Li=='A', 0\ otherwise, i=1..m]$
- $sC=[1\ if\ Li=='C', 0\ otherwise, i=1..m]$
- $sG=[1\ if\ Li=='G', 0\ otherwise, i=1..m]$

If we retake the same example, sequence “CCGATAT” is encoded as:

- $sA=[0,0,0,1,0,1,0]$
- $sC=[1,1,0,0,0,0,0]$
- $sG=[0,0,1,0,0,0,0]$

3) Match/No-Match answer:

Nevertheless the scheme can output a binary result of the query:
$R'(q,s)=(R(q,s)*E(-k))r$ where r is a random non-zero number. Note that one modular exponentiation is needed in this case. R' decrypts to 0 if a match is found and to a random number if not. Modular multiplication by a random perfectly hides the answer (except for 0) and has been widely used in blind signature protocols [20].

4) Space-time comparison

We consider one sequence as a comparison unit and show in Table 2 the space and time costs. We consider only queries and sequences over Σ to be able to compare our scheme against [1], as [1] does not support queries over Σ'. In terms of storage we assume all the sequences have the same length *m*.

5) Security evaluation

From a security perspective the framework that we use is similar to [1]. Both schemes are based on well-known security building blocks like Paillier’s encryption, public key encryption and symmetric key encryption.

CONCLUSION:

In this paper, we have revisited the challenge of sharing person-specific genomic sequences without violating the privacy of their data subjects in order to support large-scale biomedical research projects. We have used the framework proposed by Kantarcioglu et al. [1] based on additive homomorphic encryption, and two servers: one holding the keys and one storing the encrypted records. The proposed method offers two new operating points in the space-time tradeoff and handles new types of queries that are not supported in earlier work. Furthermore, the method provides support for extended alphabet of nucleotides which is a practical and critical requirement for biomedical researchers.

Big data analytics over genetic data is a good future work direction. There are rapid recent advancements that address performance limitations of homomorphic encryption techniques.

REFERENCE:

[1] M. Kantarcioglu, W. Jiang, Y. Liu, and B. Malin, “A cryptographic approach to securely share and query genomic sequences,” Inf. Technol. Biomed. IEEE Trans., vol. 12, no. 5, pp. 606–617, 2008.

[2] B. Malin and L. Sweeney, “How (not) to protect genomic data privacy in a distributed network: using trail re-identification to evaluate and design anonymity protection systems,” J. Biomed. Inform., vol. 37, no. 3, pp. 179–192, 2004.

[3] Z. Lin, A. B. Owen, and R. B. Altman, “Genomic research and human subject privacy,” Science (80-.)., vol. 305, no. 5681, p. 183, 2004.

[4] A. E. Nergiz, C. Clifton, and Q. M. Malluhi, “Updating outsourced anatomized private databases,” in Proceedings of the 16th International Conference

[5] L. Sweeney, A. Abu, and J. Winn, “Identifying Participants in the Personal Genome Project by Name,” Available SSRN 2257732, 2013.