Privacy-Preserving Pattern Matching over Encrypted Genetic Data in Cloud Computing

Bing Wang* Wei Song*† Wenjing Lou* Y. Thomas Hou*
*Virginia Polytechnic Institute and State University, Blacksburg, VA, USA
†Wuhan University, Wuhan, Hubei, China

***Abstract*—Personalized medicine performs diagnoses and treatments according to the DNA information of the patients. The new paradigm will change the health care model in the future. A doctor will perform the DNA sequence matching instead of the regular clinical laboratory tests to diagnose and medicate the diseases. Additionally, with the help of the affordable personal genomics services such as 23andMe, personalized medicine will be applied to a great population. Cloud computing will be the perfect computing model as the volume of the DNA data and the computation over it are often immense. However, due to the sensitivity, the DNA data should be encrypted before being outsourced into the cloud. In this paper, we start from a practical system model of the personalize medicine and present a solution for the secure DNA sequence matching problem in cloud computing. Comparing with the existing solutions, our scheme protects the DNA data privacy as well as the search pattern to provide a better privacy guarantee. We have proved that our scheme is secure under the well-defined cryptographic assumption, i.e., the sub-group decision assumption over a bilinear group. Unlike the existing interactive schemes, our scheme requires only one round of communication, which is critical in practical application scenarios. We also carry out a simulation study using the real-world DNA data to evaluate the performance of our scheme. The simulation results show that the computation overhead for real world problems is practical, and the communication cost is small. Furthermore, our scheme is not limited to the genome matching problem but it applies to general privacy preserving pattern matching problems which is widely used in real world.**

I. Introduction

Personalized medicine has been gaining popularity and is recognized as the health care model in the future. In this model, genetic testings are employed for selecting effective and optimal treatment based on the context of a patient's genetic information, such as DNA sequence, molecular, and cellular analysis. Compared with the traditional practice which takes a "one-size-fits-all" approach, personalized medicine is moving us to more precise, predictable and powerful health care which is customized for each patients. According to NIH, genetic testing, which could be used as diagnosis testing, prenatal testing, or predictive testing, is to search the existence of certain gene mutations over a patient's genetic data. For instance, certain mutations in the genes *BRCA1* and *BRCA2* are related to high risk of breast cancer. By testing the existence of those gene mutations in a patient's DNA, early intervention measures can be taken to prevent breast cancer from development. Due to the massive storage and computation requirement, many genetic services including personalized medicine are outsourced to or provided by third-party service providers, for example, Google Genomic[1].

While it is promising to have customized health care for each individual, there are many security and privacy risks which could thwart its wide adoption in cloud computing. The main concern is whether the patient's genetic information is exposed to unauthorized parties during testings, especially when the testings are performed by third-parties such as cloud providers. In reality, personal genomics companies enforce their privacy policies mainly relying on legislation such as Health Insurance Portability and Accountability Act (HIPAA). However, such an approach is ineffective against information leakage caused by system failures or hackers. On the other hand, de-identification, which removes or marks personal identifiers, is another widely adopted privacy preserving technique. Unfortunately, research works (e.g. [1], [2]) have found that attackers can re-identify participants using genetic data alongside with some public records.

A feasible and promising approach to protect the genetic data privacy is to encrypt the data before outsourcing it to the cloud. A challenging problem is to perform genetic test over the encrypted data. An authorized party, such as a doctor, should be able to perform genetic testing over the encrypted DNA data while no information about the patient should be leaked to cloud providers except the algorithm's output. Although the fully homomorphic encryption (FHE) [3] is a solution to the problem, its efficiency is still far from practical. Searchable encryption which builds secure indexes based on keywords is not applicable either because there are no explicitly given keywords in DNA sequences. Existing secure pattern matching schemes [4]–[8] are interactive protocols. The protocols assume the authorized party is remaining online during the entire testing process, which imposes usability and scalability issues. On the other hand, the communication overhead incurred by the interaction limits the practicability of those schemes. Another scheme [9] has limited privacy guarantee due to the leakage of the search pattern.

In this paper, we aim to study the privacy-preserving genetic testing in cloud computing and we focus ourselves on communication efficiency and strong privacy guarantee. To achieve genetic testing over the encrypted data while providing strong privacy guarantee, we adopt predicate encryption (PE) as the main cryptographic primitive. Using PE, the decryption

[1]https://cloud.google.com/genomics/

of the ciphertext depends on not only the secret key but also a pre-defined function. However, to integrate PE into a privacy-preserving sequence matching scheme is nontrivial. To design a secure and well-functioning scheme, there are three important design challenges, i.e., 1) data structures to support sequence matching over encrypted data, 2) security and privacy mechanism to prevent information leakages such as search pattern and result privacy, and 3) protocol design to achieve efficient communication. To this end, we make the following contributions.

- We propose a novel scheme for privacy-preserving genetic testing in cloud computing. To address the genetic sequence matching challenge, we modified the predicate encryption (PE) scheme in [10] to achieve approximate sequence (wildcard-based) matching for genetic sequences. In particular, the genetic sequence is encrypted with PE and an authorized party such as a doctor can submit a genetic testing request with a secure query sequence pattern to the cloud.
- Our scheme is provably secure under the well-defined subgroup decision assumption over a bilinear group. To provide strong privacy guarantee, search pattern and testing results are protected from the cloud server. In addition, the authorized party only learns the information which the patient allows. We utilize suffix tree structure to pre-process the genetic sequence so that the sequence matching computation can be done in a single round of communication.
- We thoroughly analyze of the complexity of the different approaches for the secure DNA sequence matching problem. We then compare our scheme with them to illustrate the strength of our scheme in practice. We further perform a simulation study using the public available DNA data. The results show that the computation overhead for the problem is reasonable, and the communication cost is small.

The remaining of the paper is organized as follows. In Section II, we review the literature of the related work. We formulate our problem in Section III. Our secure sequence matching scheme is presented in Section IV. We analyze the complexity of our scheme and compare it with the existing schemes in Section V. Finally, we conclude our paper in Section VI.

II. Related Work

Privacy-preserving computation over genetic data is always considered under two different system models. The first model assumes each party possesses its data and would like to compute certain functions over its input along with other parties' data. Secure multi-party computation is the main technique used under the first model. The second model is the secure outsourcing of computation model. In this model, the genetic data is stored in a semi-trusted party, usually in a cloud. The computation task is mainly carried out by the cloud server over encrypted data. We review the related works under each system model respectively.

A. Secure multi-party computation model

Atallah and Li [4] proposed a privacy-preserving protocol to compute the edit distance between two sequences based on dynamic programming. The protocol requires two non-colluding servers, each of them possessing one input sequence, to engage an interactive process. A secure look-up protocol is used to exchange the computation results of the servers in each iteration. Because the number of the iterations is the product of the lengths of the two input sequences, the computation, and the communication overhead are considerable. Jha *et al.* [6] improve the computation efficiency of [4]. Yet, it shares the same communication complexity since it is an iterative protocol as well. It is worth mentioning that these schemes can be applied to other problems that dynamic programming could solve. In [7], Wang *et al.* proposed a distributed framework for privacy-preserving genetic computing, which applies program specialization to partitioning genetic computation to different sensitivity levels. The expensive computation involving secure multi-party computation is only performed for the higher sensitivity level data. However, with our current knowledge, it is unclear whether the "insensitive" DNA data will be important in the future. Therefore, leaking these DNA data is not a satisfying practice. Troncoso-Pastoriza *et al.* [5] proposed a protocol to calculate edit distance with an encrypted input sequence through finite state machine (FSM). In their scheme, the server possesses the FSM for the target DNA sequence while the input of the FSM is the client's query sequence. The server and the client participate in an interactive protocol using oblivious transfer protocols to calculate the state transition of the FSM. Although the payload of each communication is smaller compared to the dynamic programming based approaches, the communication overhead is still considerable because the number of the iterations is a linear function of the lengths of the input sequence. It is worth mentioning that the computation used to generate the FSM can also be huge when the size of the sequence grows. Therefore, the poor scalability limits the usability of those schemes. In [8], Blanton and Aliasgari proposed a scheme which outsources the computation and the communication of [5] to multiple servers to improve efficiency in practice. But it is still based on an interactive algorithm which requires multiple rounds of communication. Wang *et al.* [11] proposed an privacy-preserving protocol to estimate the edit distance between two genome sequence. They estimate the edit distance by transforming the edit distance computation problem to the set intersection size approximation problem. The computation of the set intersection size is done through multi-party computation. Their scheme is extremely efficient as the edit distance computation time for two whole genome sequences can be finished in seconds with a relatively small error.

B. Secure outsourcing of computation model

Under the secure outsourcing of computation model, the cloud server is always considered as an honest-but-curious adversary. Therefore, the security objective is to perform

certain kinds of computation in cloud without leaking private information.

Lu *et al.* [12] proposed a secure outsourcing scheme for genome-wide association study (GWAS). The GWAS aims to discover the association between gene mutations and certain diseases. The major computation of the GWAS is based on the statistic information of the genetic data. In [12], the computation is performed over the encrypted statistic in the cloud to protect the data privacy. Barman *et al.* constructed a privacy-preserving computation scheme to calculate the health risk based on the known association between gene mutations and the diseases. Given a patient's gene mutation information as a vector and the known association mapping with a weight between a gene mutation and a disease, the cloud computes the aggregated risk. Chen *et al.* [13] proposed a secure DNA alignment scheme utilizing a hybrid cloud. The scheme first locates an approximated location by comparing the ciphertext of the genetic data in a public cloud. Then the alignment is performed in a private cloud using the plaintext of the genetic data. Baldi *et al.* [14] implement the secure paternity tests utilizing the private set intersection (PSI) technique. Those applications work on the short tandem repeats which are the number of the repeats of a specific nucleotides pattern. A secure index is built based on the genetic signatures, i.e., the short tandem, and the cloud calculates the occurrence for a specific short tandem over the encrypted genetic data. Ayday *et al.* [15] proposed a private DNA sequence retrieving scheme based on the order-preserving encryption. The scheme builds an index using the sequence position information instead of the genetic sequence in the SAM file. Kantarcioglu *et al.* [16] proposed a scheme to securely query an SNP database using an additive homomorphic encryption scheme. Those schemes focus on a specific application that involves little or no genetic sequence data, and thus, have different challenges compared to our problem.

Very recently, Chase and Shen [9] proposed a symmetric searchable encryption scheme supporting subsequence matching. Similar to our approach, their construction is based on suffix tree. However, the search pattern is leaked to the cloud because deterministic encryption algorithm is used in their scheme. Also, approximate sequence matching is not supported.

Secure DNA sequence matching is also related to searchable encryption which allows the user to query the encrypted documents with the encrypted keyword(s). A considerable amount of the searchable encryption schemes (e.g. [17]–[21]) has been proposed. The core technique of searchable encryption is to build a secure index for the keywords extracted from a document set. However, there is no explicitly given keyword in genetic sequence data. A possible solution is to treat meaningful subsequences as keywords. However, the amount of the possible subsequence is huge as the DNA sequence consists of millions of base pairs. On the other hand, subsequences with various length are used in the DNA sequence matching problem. Therefore, the searchable encryption schemes cannot be directly applied to the secure DNA sequence matching problem.

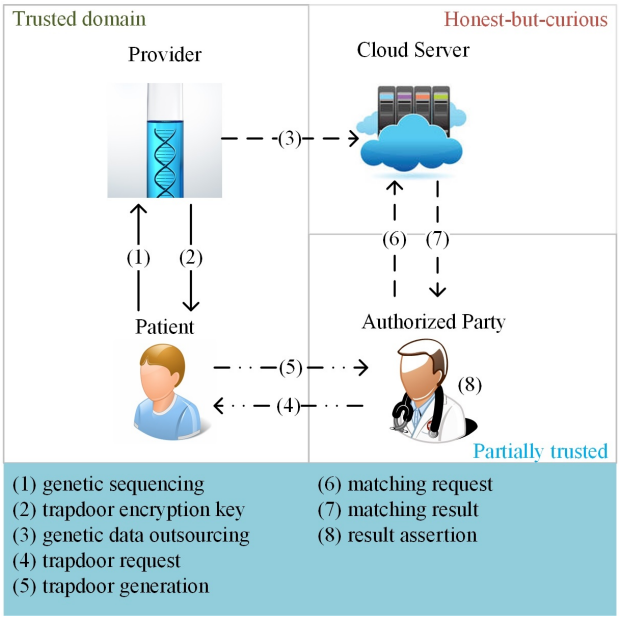

Fig. 1. System model of personalized medicine in cloud computing.

III. Problem Formulation

Human DNA contains important genetic instructions that define and function each. According to NIH, genetic testing is mainly used to search for abnormal gene mutations that lead to genetic disorder. In medical practice, searching the existence of the known gene mutations that relate to certain genetic diseases is one of the important methods to perform diagnosis testings.

A. System model

We consider a personalized medicine model where there are four entities in the system. The *provider* such as a laboratory or a personal genomics service company sequences the *patient*'s DNA from cell samples. The DNA sequence is then encrypted and outsourced into the *cloud*. When an *authorized party*, such as a doctor, would like to perform a genetic testing for the patient, he/she can submit a test request to the cloud and gets the result back. To ensure the patient privacy, the testing sequence, i.e., a DNA sequence pattern, should be encrypted as well. We denote the encrypted request as *trapdoor*. The proposed system model is shown in Fig. 1. Based on the system model, we define secure sequence matching as follows.

Definition 1: A secure sequence matching (SSM) is a collection of five polynomial time algorithms, i.e., *Setup*, *DataEnc*, *ReqEnc*, *Search*, and *Assert* such that

- $(k_S, k_T) \leftarrow Setup(1^\lambda)$: is a key generation algorithm run by the provider. It takes a security parameter λ and outputs a data encryption key k_S and a request encryption key k_T.
- $(\tilde{\mathcal{S}}, \Omega_S) \leftarrow DataEnc(k_S, \mathcal{S})$: is an algorithm run by the provider. It takes k_S and a sequence $\mathcal{S}$, and outputs the ciphertext $\tilde{\mathcal{S}}$ with an assertion token Ω_S.

- $(T, \Omega_T) \leftarrow ReqEnc(k_T, \Delta)$: is a probabilistic algorithm run by the patient. It takes k_T and a query sequence Δ; outputs an encrypted query T and the assertion token Ω_T.
- $R \leftarrow Search(\tilde{\mathcal{S}}, T)$: is a probabilistic algorithm run by the cloud. It takes $\tilde{\mathcal{S}}$ and T, and outputs the encrypted search result R.
- 1 or $\perp \leftarrow Assert(\Omega_S, \Omega_T, R)$: is a deterministic algorithm run by the authorized party. It takes both the assertion tokens and R. Output 1 if the query Δ is a subsequence of the sequence $\mathcal{S}$; otherwise, it outputs $\perp$.

B. Security model

We assume secure communication channels are used to defend against the outside attacker. Therefore, we focus on the possible inside privacy leakage of the system.

Among the four entities, the patient and the provider are fully trusted because they have the original DNA data. We assume the cloud is "honest-but-curious" because of the possible compromise caused by the system failure or hacking. At last, we allow the authorized party to learn only the test result. On the other hand, the authorized party must not be able to generate any valid query request unless the request is granted by the patient.

C. Design objective

To achieve privacy-preserving genetic testing, the core requirement is that the cloud cannot deduce any useful information about the patient's genetic data. The requirement must be enforced even the cloud has collected an abundance of encrypted queries and the corresponding matching results. We summarize the design objectives as follows.

- **Data confidentiality.** The cloud should not be able to recover any useful information from any encrypted data, which includes the encrypted genetic sequence, the encrypted request, and the encrypted matching result.
- **Trace indistinguishability.** Denote the trapdoor and its corresponding match result as a *trace*, the cloud should not be able to distinguish two traces. In other words, it is impossible for the cloud to link an encrypted request with a previously submitted one. It is also called search pattern privacy.
- **Efficiency and usability.** As genetic testing may not require a real-time result, the computation time at the cloud side can be tolerated in certain extend. However, the computation at the patient and the authorized party side must be constrained because their computation resource is usually limited. On the other hand, the interaction among the entities should be minimized to enjoy usability.

IV. Secure pattern matching

A. Scheme overview

Our main goal is to securely match one pattern to a genetic sequence. The key idea of our scheme is to compare a sequence pattern Δ, a gene mutation, with a target sequence $\mathcal{S}$, i.e., the patient's DNA, in a character-by-character manner. As the query pattern can appear at any position in the patient's DNA, we adopt the suffix tree structure to represent the DNA sequence. Given a sequence $\mathcal{S}$ of length n, a suffix $S_i, 1 \leq i \leq n$, is a subsequence of $\mathcal{S}$ from the position i to n. For example, S_2 of the sequence *ATGC* is *TGC*. The suffices are very useful in subsequence matching because every subsequence of $\mathcal{S}$ must be a prefix of $S_i, \exists i \in [1, n]$. The suffixes of $\mathcal{S}$ are often organized as a tree and every edge is labeled by a subsequence of $\mathcal{S}$. The concatenation of the string-labeled edges from the root to a leaf represents a suffix of $\mathcal{S}$. To encrypt the sequence, we utilize a predicate encryption scheme [22] which supports a secure inner product computation between two encrypted sequences. The encryption algorithm is based on a composite-order of three distinct primes bilinear group. The bilinear group is defined as follows.

Definition 2: $\mathbb{G}$ is a composite-order of three distinct primes bilinear group such that

1) $\mathbb{G}$ and $\mathbb{G}_T$ are two cyclic groups of finite order $N = pqr$, where p, q, r are distinct primes.
2) e is a non-degenerate bilinear map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$, i.e.,
   - $\forall g, h \in \mathbb{G}, a, b \in \mathbb{Z}_N, e(g^a, h^b) = e(g, h)^{ab}$,
   - if g is a generator of $\mathbb{G}$, $e(g, g)$ is a generator of $\mathbb{G}_T$ as well.
3) e can be calculated in polynomial time.

To achieve secure and efficient sequence matching using PE, there are three main technical challenges.

1) The same character has to be encrypted differently each time to ensure indistinguishability while the different ciphertexts of the same character have to be matched correctly to properly perform genetic testing.
2) As most of the gene mutations are single-nucleotide polymorphisms (SNP), the matching algorithm should be tolerant of character mismatches to correctly handle gene mutations.
3) The character-by-character matching results must be aggregated efficiently and properly to meet the requirement of the communication efficiency.

As the original PE in [22] does not support the character-by-character comparison functionality, we propose our error-tolerant character-by-character-comparison predicate encryption scheme.

B. Error-tolerant pattern matching scheme

The sequence comparison function can be expressed using the following equation.

$$F = \sum_{i=1}^{n}(x_i - y_i)^2 = \sum_{i=1}^{n} x_i^2 - 2\sum_{i=1}^{n} x_i y_i + \sum_{i=1}^{n} y_i^2,$$

where $x_i, y_i > 0$ are the characters of the sequences.

Following Def. 1, our scheme contains five algorithms, i.e., *Setup, DataEnc, ReqEnc, Search, Assert*.

- **Setup**$(1^\lambda, \mathcal{S})$ generates a public key $pk = (g_p, g_r)$, a master secret key $msk = (p, q, r, g_q)$, a pattern encryption key k_T and a sequence encryption k_S given the security parameter λ. The $k_T = \{h_{1,i}, h_{2,i}\}_{i=1}^{n}$, and the

$k_S = (g_q \cdot R_0, \{H_{1,i}, H_{2,i}\}_{i=1}^n)$, where n is the length of the sequence $\mathcal{S}$, $H_{1,i} = h_{1,i} \cdot R_{1,i}$, $H_{2,i} = h_{2,i} \cdot R_{2,i}$, and $R_0, R_{1,i}, R_{2,i}, \in \mathbb{G}_r$ for i from 1 to n.

- **DataEnc**$(k_S, \mathcal{S})$ encrypts the sequence $\mathcal{S}$ and generates an assertion token Ω_S. The encrypted sequence is generated as follow,
  1) Randomly choose $s, \alpha, \beta \in \mathbb{Z}_N$. and $R_{3,i}, R_{4,i}$, $R_{5,i}, R_{6,i} \in \mathbb{G}_r$ for j from 1 to n.
  2) Encrypt each
  $$x_i \in S$$
  as a 4-tuple
  $$\begin{aligned}(C_{1,i},C_{2,i},C_{3,i},C_{4,i}) = \\ (H_{1,i}^s \cdot k_T^{\alpha \cdot \mathcal{H}(x_i,i)} \cdot R_{3,i}, \\ H_{2,i}^s \cdot k_S^{\beta \cdot \mathcal{H}(x_i,i)} \cdot R_{4,i}, \\ H_{1,i}^s \cdot k_T^{-\alpha \cdot \mathcal{H}(x_i,i)^2} \cdot R_{5,i}, \\ H_{2,i}^s \cdot k_S^{-\beta \cdot \mathcal{H}(x_i,i)^2} \cdot R_{6,i}),\end{aligned}$$
  where $\mathcal{H}$ is a cryptographic hash function.
  3) Denote
  $$\tilde{S} = \{C_{1,i}, C_{2,i}, C_{3,i}, C_{4,i}\}_{i=1}^n$$
  as the ciphertext for S.
- **ReqEnc**(k_T, Δ) encrypts the query sequence and generates an trapdoor assertion token Ω_T. Denote the character at the wildcard positions as $\star$. The trapdoor is encrypted as follow,
  1) Randomly choose $R_7 \in \mathbb{G}_r$, $f_1, f_2 \in \mathbb{Z}_q$, and $R_8 \in \mathbb{G}_q$.
  2) let $\Delta = (y_1, y_2, \cdots, y_m), m = |\Delta|$, randomly choose $r_{1,i}, r_{2,i}, r_{3,i}, r_{4,i} \in \mathbb{Z}_N$ for i from 1 to m.
  3) Encrypt each $y_i \in \Delta$ as a 4-tuple
  $$\begin{aligned}(T_{1,i},T_{2,i},T_{3,i},T_{4,i}) = \\ (g_p^{r_{1,i}} \cdot g_q^{f_1 \cdot 2 \cdot \mathcal{H}(y_i,i)}, g_p^{r_{2,i}} \cdot g_q^{f_2 \cdot 2 \cdot \mathcal{H}(y_i,i)}, \\ g_p^{r_{3,i}} \cdot g_q^{f_1}, g_p^{r_{4,i}} \cdot g_q^{f_2})\end{aligned}$$
  if $y_i \neq \star$. Otherwise, encrypt $\star$ as
  $$\begin{aligned}(T_{1,i},T_{2,i},T_{3,i},T_{4,i}) = \\ (g_p^{r_{1,i}}, g_p^{r_{2,i}}, g_p^{r_{3,i}}, g_p^{r_{4,i}}).\end{aligned}$$
  Generate the trapdoor assertion token Ω_T as a 2-tuple
  $$\begin{aligned}(R_7 \cdot R_8 \cdot \prod_{i=1}^m h_{1,i}^{-r_{1,i}} h_{2,i}^{-r_{2,i}}, \\ e(g_q, g_q)^{-(\alpha f_1 + \beta f_2 \mod q) \sum_{i=1}^m \mathcal{H}(y_i,i)^2}).\end{aligned}$$
- **Search**$(\tilde{\mathcal{S}}, T)$ generates the search result γ_i for the ith suffix of $\tilde{\mathcal{S}}$ that starts at x_i as follow
  $$\prod_{j=1}^m e(C_{1,k}, T_{1,j}) \cdot e(C_{2,k}, T_{2,j}) \cdot e(C_{3,k}, T_{3,j}) \cdot e(C_{4,k}, T_{4,j})$$
  where $k = i + j - 1$.
- **Assert**$(\Omega_S, \Omega_T, \Gamma = \{\gamma_i\}_{i=1}^n)$ outputs 1 iff
  $$e(\Omega_S, \Omega_T[1]) \cdot \gamma_i \cdot \Omega_T[2] \stackrel{?}{=} 1$$
  stands for at least one $i \in [1, n]$.

Discussion. In our scheme, each character is encrypted along with a different random element each time. The introduced randomness hides the relationship of the underlying characters while has no side-effect when performing sequence matching. We add wildcard support during query request encryption algorithm to handle possible gene mutations. At last, the matching result for each character is combined to a single value to enable efficient communication. Therefore, our scheme fulfills the technical challenges.

C. Correctness and security analysis

We show in Fig. 2 that *Assert* outputs 1 if and only if the query sequence Δ is a prefix of at least one suffix of the sequence $\mathcal{S}$.

Notice that the $\star$, which is the character at a wildcard position, is encoded as 0 in the trapdoor. Therefore, the pairing operation $e(C_{k,i}, T_{k,j}), k \in \{1, 2, 3, 4\}$ will be 1 if jth position of the pattern is a $\star$. In another word, $\star$ matches with any character. Thus, it achieves wildcard matching function.

The security of scheme mainly relies on the security of the underlying predicate encryption algorithm. The security of the predicate encryption scheme in [22] is based on the following hardness assumption over a bilinear group.

Definition 3 (Subgroup decision problem): Given two cyclic groups $\mathbb{G}, \mathbb{G}_T$ of finite order $N = pqr$ and a bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$, we randomly choose

1) $P_0 \in \mathbb{G}_p, R_0 \in \mathbb{G}_r$,
2) $T_0 \in \mathbb{G}_{pq}, T_1 \in \mathbb{G}_p$,

where $\mathbb{G}_p, \mathbb{G}_r, \mathbb{G}_q, \mathbb{G}_{pq}$ are subgroups of $\mathbb{G}$ of orders p, r, q, pq, respectively.

Given $D = (N, \mathbb{G}, \mathbb{G}_T, e, P_0, R_0, T_b)$ where $b = 0$ or 1, we define the advantage of an algorithm $\mathcal{A}$ to solve the subgroup decision problem to be

$$Adv_{\mathcal{A}} = |Pr[\mathcal{A}(D, T_0) = 1] - Pr[\mathcal{A}(D, T_1) = 1]|.$$

The assumption is that for any PPT algorithm $\mathcal{A}$, $Adv_{\mathcal{A}}$ is negligible in the security parameter. The hardness of the subgroup decision problem relies on the hardness of factoring N. The proof of the above assumption is given in [22] and we refer our readers to that paper for more detail. As shown in [22], the encryption scheme is an *attribute-hiding* predicate encryption scheme against adaptive adversaries. Attribute-hiding means the adversary cannot learn the message even with the secret key as along as $F(I) \neq 1$, where I is the attributes. Therefore, the encryption has two layers of protection, i.e., the secret key and the attributes. In our scheme, we utilize the security key as our trapdoor but with slight modification. We retain part of the security key of the original encryption scheme as the trapdoor assertion token and retain part of the ciphertext as the sequence assertion token. Therefore, the secret in our scheme

Assert calculate the following equation.

$$
\begin{aligned}
& e(\Omega_S, \Omega_T[1]) \cdot \gamma_i \cdot \Omega_T[2] \\
=& e(g_p^s, R_7 \cdot R_8 \cdot \prod_{i=1}^{m} h_{1,i}^{-r_{1,i}} h_{2,i}^{-r_{2,i}}) \cdot \prod_{i=1}^{m} e(C_{1,i}, T_{1,i}) \cdot e(C_{2,i}, T_{2,i}) \cdot e(C_{3,i}, T_{3,i}) \cdot e(C_{4,i}, T_{4,i}) \\
=& e(g_p^s, \prod_{i=1}^{m} h_{1,i}^{-r_{1,i}} h_{2,i}^{-r_{2,i}}) \cdot \prod_{i=1}^{m} e(H_{1,i}^s k_T^{\alpha \cdot \mathcal{H}(x_i,i)} R_{3,i}, g_p^{r_{1,i}} g_q^{2 \cdot f_1 \cdot \mathcal{H}(y_i,i)}) \cdot e(H_{2,i}^s k_S^{\beta \cdot \mathcal{H}(x_i,i)} R_{4,i}, g_p^{r_{2,i}} g_q^{2 \cdot f_2 \cdot \mathcal{H}(y_i,i)}) \cdot \\
& \cdot \prod_{i=1}^{m} e(H_{1,i}^s k_T^{-\alpha \cdot \mathcal{H}(x_i,i)^2} R_{5,i}, g_p^{r_{1,i}} g_q^{f_1}) \cdot e(H_{2,i}^s k_S^{-\beta \cdot \mathcal{H}(x_i,i)^2} R_{6,i}, g_p^{r_{2,i}} g_q^{f_2}) \cdot \Omega_T[2] \\
=& e(g_p^s, \prod_{i=1}^{m} h_{1,i}^{-r_{1,i}} h_{2,i}^{-r_{2,i}}) \cdot \prod_{i=1}^{m} e(h_{1,i}^s g_p^{\alpha \cdot \mathcal{H}(x_i,i)}, g_p^{r_{1,i}} g_q^{2 \cdot f_1 \cdot \mathcal{H}(y_i,i)}) \cdot e(h_{2,i}^s g_p^{\beta \cdot \mathcal{H}(x_i,i)}, g_p^{r_{2,i}} g_q^{2 \cdot f_2 \cdot \mathcal{H}(y_i,i)}) \\
& \cdot \prod_{i=1}^{m} e(h_{1,i}^s g_p^{-\alpha \cdot \mathcal{H}(x_i,i)^2}, g_p^{r_{3,i}} g_q^{f_1}) \cdot e(h_{2,i}^s g_p^{-\beta \cdot \mathcal{H}(x_i,i)^2}, g_p^{r_{4,i}} g_q^{f_2}) \cdot \Omega_T[2] \\
=& e(g_q, g_q)^{(\alpha f_1 + \beta f_2) \sum_{i=1}^{m} 2\mathcal{H}(x_i,i)\mathcal{H}(y_i,i)} \cdot e(g_q, g_q)^{-(\alpha f_1 + \beta f_2) \sum_{i=1}^{m} \mathcal{H}(y_i,i)^2} \cdot e(g_q, g_q)^{-(\alpha f_1 + \beta f_2) \sum_{i=1}^{m} \mathcal{H}(x_i,i)^2} \\
=& e(g_q, g_q)^{\sum_{i=1}^{m} (\alpha f_1 + \beta f_2 \mod q)(2\mathcal{H}(x_i,i)\mathcal{H}(y_i,i) - \mathcal{H}(y_i,i)^2) - \mathcal{H}(x_i,i)^2)},
\end{aligned}
$$

Whether the above equation results 1 is determined by

$$
\sum_{i=1}^{m} (\alpha f_1 + \beta f_2 \mod q)(2\mathcal{H}(x_i,i)\mathcal{H}(y_i,i) - \mathcal{H}(y_i,i)^2 - \mathcal{H}(y_i,i)^2 - \mathcal{H}(x_i,i)^2) \tag{1}
$$

Clearly, if the query sequence Δ is a prefix of one of the suffix of the sequence $\mathcal{S}$, Eq. 1 equals 0, which means the output of *Assert* is 1.

Now we focus on the proof of sufficiency. Note that because $\mathcal{H}$ is a cryptographic hash function, the probability of $\mathcal{H}(x_i, i) = \mathcal{H}(y_j, j), i \neq j$ is negligible. If for i from 1 to m, $\mathcal{H}(x_i, i) = \mathcal{H}(y_i, i)$, i.e. $x_i = y_i$, then the above equation evaluates to 0. If there exits $i \in [1, m], h(x_i, i) \neq \mathcal{H}(y_i, i)$, i.e., the query sequence Δ is not a subsequence of $\mathcal{S}$, there are two cases. If $\sum_{i=1}^{m} (\mathcal{H}(x_i, i) - \mathcal{H}(y_i, i))^2 \neq 0 \mod q$, then the above equation evaluates to 0 with a negligible probability. The other case is that $\sum_{i=1}^{m} (\mathcal{H}(x_i, i) - \mathcal{H}(y_i, i))^2 = 0 \mod q$. However, this reveals a non-trivial factor of $N = pqr$. Because finding a non-trivial factor of N is hard according to the subgroup decision assumption, the probability of the second case is also negligible. Therefore, if *Assert* outputs 1, the query sequence Δ is a prefix of one of the suffix of the sequence $\mathcal{S}$. Combining the proof of necessity and sufficiency, we prove that our scheme correctly reveals the fact whether a query sequence Δ is a subsequence of $\mathcal{S}$.

Fig. 2. Correctness proof

is the *attributes* which correspond to the DNA sequence and the query pattern. Although our modification reveals part of the secret, i.e., the trapdoor, our additional assertion step hides the attribute evaluation from the adversary, which is the key security property of our scheme. The security rationale behind our scheme is that without the assertion tokens that we hide from the adversary, distinguishing the search result reduces to the subgroup decision problem. Therefore, our scheme is secure against the adaptive adversary based on the security of the predicate encryption scheme.

Trapdoor unforgeability. A trapdoor is generated for an authorized party when the genetic test request is granted by the patient. To forge a valid trapdoor, the authorized party must be able to forge both the trapdoor T and the assertion token Ω_T. As T is a vector that each element corresponds to a character, the authorized party can generate a valid T of his choice. However, Ω_T is generated through the trapdoor encryption key k_T which is kept by the patient. A single Ω_T is generated for a query sequence. Recall the trapdoor generation process, a collection of random numbers are introduced. Intuitively, without knowing k_T, it is hard to forge a valid verification token.

V. Performance analysis and evaluation

A. Computation complexity

Our scheme relies on the bilinear pairing operation, which is computationally expensive. We calculate the numbers of the multiplication as well as the exponentiation. The computation complexity of each algorithm is summarized as follows.

- In *DataEnc*, the sequence encryption takes $O(n)$ multiplications and $O(n)$ exponentiations, where n is the length of $\mathcal{S}$.
- In *ReqEnc*, $O(m)$ multiplications and $O(m)$ exponentiations are used to generate the trapdoor, and $O(m)$ multiplication to generate the assertion token, where m is the length of the query sequence.
- In *Search*, $O(mn)$ pairings are needed.
- In *Assert*, one pairing and $n + 1$ multiplications are required.

In summary, our scheme requires $O(mn)$ pairings, $O(n)$ exponentiations, and $O(n)$ multiplications. The dominance of the computation is the bilinear pairing. According to the benchmark of the public available libraries such as JPBC [23] or PBC [24], the average time for a 1024-bit discrete log security is 13 ms. Therefore, for a genetic sequence with five thousand base pairs, it takes about three hours to test a query sequence with one hundred base pairs. The computation time is within the reasonable range in practice.

In [6], Jha *et al.* proposed a secure dynamic programming scheme to compute the edit distance between two sequences. Their scheme has three phase. *Phase 0* involves no heavy computation. In *Phase 1*, one player evaluates $n \times m$ instances of Yao's secure circuit evaluation over an equality circuit. The other player needs to initiate $n \times m \times q$ 1-out-of-2 oblivious transfers, where $q = \log|\Sigma|$ and Σ is the alphabet, i.e., $\{A, T, G, C\}$ in our case. *Phase 2* needs $n \times m$ iterations. Each iteration involves one evaluation of an instance of a minimum-of-three circuit and $3\log(m+n)$ instances of 1-out-of-2 oblivious transfer. In summary, the scheme requires $2mn$ circuit evaluations, $3mn\log(n+m)+2mn$ oblivious transfers. As described in [6], their scheme takes about 35 seconds to compute for a (25×25) problem. Since the complexity of their scheme is linear in nm, a (5000×100) problems will take much more time.

In [5], Troncoso-Pastoriza *et al.* presented a finite automaton machine based scheme to perform error resilient subsequence testing. The key idea of the scheme is to split the inputs of the FSM to two servers and to compute the state of the FSM interactive protocol through secret sharing. The computation of each iteration is small since the process only involves a constant number of the Paillier homomorphic encryption operations [25]. However, the number of the states of the FSM grows as the size of the sequence, which increases the computation overhead as well as the storage overhead to store the transition matrix for the FSM. Additionally, it also takes extra computation to construct the FSM for a specific sequence.

Discussion. Among the three approaches, [5] incurs the least computation overhead but requires a pre-process to generate the FSM. Although our scheme uses expensive bilinear pairing operation, the computation complexity is similar to the scheme in [6]. On the other hand, we push most of the computation to the cloud which has powerful and plenty computation resources. Additionally, parallel computing techniques such as MapReduce [26] can be effortlessly applied to our search algorithm which is to compute bilinear pairing between two characters.

B. Communication consumption

Our scheme advances the existing works in the communication complexity. Our scheme only needs one round communication. The cloud returns n group elements in $\mathbb{G}_T$ to the authorized party. According to [27], a group of the size 128-bit over an elliptic curve is secure enough. Therefore, the total communication to transmit the result is small, i.e., in the magnitude of kilobytes.

The communication overhead of the scheme in [6] is high due to the interactions in *Phase 2* and the interactions of the oblivious transfer. In *Phase 0*, the communication cost is $(m+n)\log(m+n)$ bits. In *Phase 1*, there are $n \times m$ iterations. Each iteration consumes 4 bits. In Phase 2, there are $n \times m$ iteration. Each iteration contains $3\log(m+n)$ instances of 1-out-of-2 oblivious transfer. As shown in [6], for a (200×200) problem, the most efficient protocol consumes more than 360 megabytes bandwidth. Since the communication overhead is linear in the product of the size of the sequences, i.e., mn, the bandwidth cost will be increased dramatically.

In [5], the communication overhead comes mainly from the oblivious transfer. There are $O(mn)$ oblivious transfers and $O(n|Q|)$ ciphertext transfers, where $|Q|$ is the number of the state of the FSM and it is linear in n. Since the communication complexity is the same as in [6], it implies that the scheme in [5] will consume similar amount of the bandwidth as in [6].

Discussion. Comparing with the other schemes, our scheme incurs much less communication overhead. Because the bandwidth is more valuable in cloud computing compared with the computation and the storage, the communication efficiency is an important factor for the schemes to be practical. Additionally, the existing schemes are interactive protocols, which assume the both parties, i.e., the cloud and the doctor, to be online the entire process to participate the interaction. However, the assumption is not reasonable in practice and raises usability issues in the personalized medicine application scenario.

C. Simulation study

We carry out several experiments to evaluate the computation performance of our scheme. We utilize the JAVA Paring-Based Cryptography Library (jPBC) [23] to implement the proposed scheme in this paper. All the experiments are conducted on a laptop equipped with an Intel 2.6GHz processor and 8GB memory. The operating system is Ubuntu 14.04. We use the human genome data from the Ensembl project [28] in the form of SAM files. SAM file format is one of the most popular file formats to store genetic sequences. Each SAM file contains short DNA sequences of which the lengths are from hundreds to thousands of base pairs.

1) Sequence encryption cost: The sequence time for a suffix is linear in the size of the suffix. Since there are n suffices in a sequence of length n, the encryption cost is quadratic in the size of the sequence $\mathcal{S}$. It is worth mentioning that the algorithm is only performed once. However, it is still computation expensive considering that the provider may have limited computation power. To that end, we further optimize the process. Note that the only terms in the ciphertext involving the input sequence is $k_S^{\alpha h(x_i,i)}, k_T^{\beta h(x_i,i)}$. Therefore, the other parts of the ciphertext can be pre-computed to reduce the computation. Meanwhile, we would like to reduce the amount of the exponentiation by using a hash function $\mathcal{H} : \Sigma \times \mathcal{Z}^+ \rightarrow \mathcal{Z}_N$, where Σ is an alphabet for the DNA base pair, i.e.,

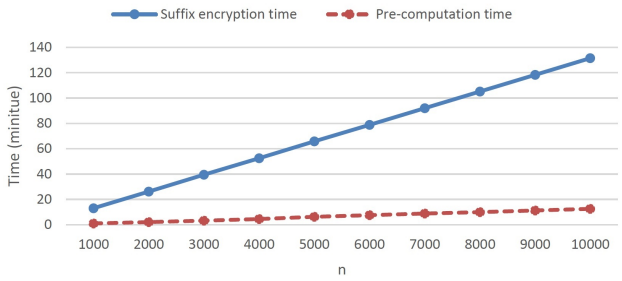

Fig. 3. Sequence encryption time for a suffix.

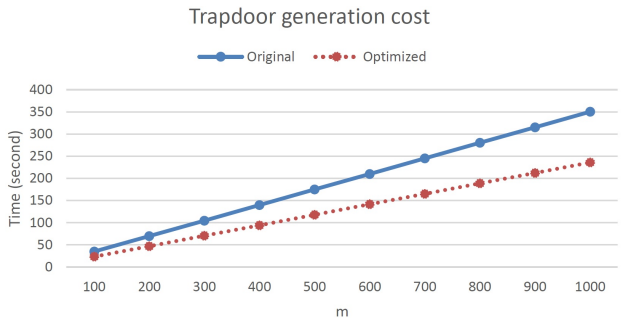

Fig. 4. Pattern encryption time.

$\{$A,T,G,C$\}$. We can encode each letter σ in Σ to a binary expression using a one-to-one mapping $\mathbb{B} : \Sigma \to \{0,1\}^{\log|\Sigma|}$. Then the h can be

$$\mathcal{H}(x_i, i) = \mathbb{B}(x_i)||F(i),$$

where $||$ is concatenation and $F : \{0,1\}^\star \to \{0,1\}^l$ is a cryptographic hash function such as SHA-1. The concatenation can be expressed as an addition as $B(x_i) \times 2^l + F(i)$. Because the size of the alphabet is limited, we can pre-compute $k_{Ind}^{\alpha \cdot B(\sigma) \cdot 2^l}$ and $k_{Ind}^{\alpha \cdot F(i)}$ for $\sigma \in \Sigma, i \in [1, n]$. Then during the index generation process, we only need to compute a multiplication instead of an exponentiation. We show the optimized encryption time in Fig. 3. The pre-computation time is shown as the red dash in Fig. 3. Note that this modification does not affect the correctness of our scheme because h utilizes a cryptographic hash function and is collision-free. We denote an alphabet of size n as Σ and define the following hash function $h : \Sigma \times \mathbb{Z} \to \mathbb{Z}_N$ as

$$\mathcal{H}(x, i) = \mathcal{B}(x)||F(i) \mod N, x \in \Sigma, i \in \mathcal{Z},$$

where $\mathcal{B} : \Sigma \to \{0,1\}^{\log n}$ is one to one mapping and $F : \{0,1\}^\star \to \{0,1\}^l$ is a collision-free cryptographic hash function. We now prove h is also a collision-free hash function.

Proof: 1: We prove our claim by contradiction. We assume the h is not a collision-free hash function. Then there must exist $x, y \in \Sigma$ and $i, j \in \mathbb{Z}$ such that $\mathcal{H}(x, i) = \mathcal{H}(y, j)$. Because the hash result from $\mathcal{H}$ is a concatenated string, we have the following equations $\mathcal{B}(x) = \mathcal{B}(y), F(i) = F(j)$. However, because $\mathcal{B}$ is a one-to-one mapping and F is a collision-free cryptographic hash function, the above equations hold iff $x = y, i = j$, which contradicts with our assumption. Therefore, h is a collision-free hash function as well.

2) Trapdoor generation cost: The trapdoor generation process is similar to the index generation process which involves only the multiplication and the exponentiation. The generation time is linear in the size of the query sequence because we encrypt each character only once. The patient can also pre-compute part of the terms in the trapdoor to optimize the computation. We show the trapdoor generation time for the both approaches in Fig. 4. Note that compared with the index generation which is run by the provider, the computation cost of the trapdoor is more important because the process might be run on the user's resource limited device. After optimization, our scheme generates a trapdoor for a 1000-long in about 4 minutes. Since our simulation is a proof-of-the-concept, we believe the overhead can be further reduced with the proper implementation and optimization.

3) Pattern matching cost: The search process is the most computation expensive algorithm due the amount of the pairing operation. As analyzed earlier, for a sequence $\mathcal{S}$ with length n and a query sequence with length m, the number of the pairing computation is linear in mn. We show the search cost in Table I when the size of $\mathbb{S}$ is fixed with $n = 500, 1000$, respectively. As shown in the table, the computation cost is linearly increasing on the size of the query sequence grows. Because the running time of the problem size (1000×300) takes more than 7 hours, we estimate that value based on our observations. Although the computation overhead seems high in our simulation, it is worth mentioning that the cloud should have better computation power than ours. For example, an Amazon EC2 M4.large instance which has 2 vCPU, 6.5 ECU and 8 GB memory costs 0.12$ for an hour. The computation power of the M4.large instance is roughly the same as our simulation platform. Therefore, the dollar amount cost to perform a matching is very cheap compared with most of the laboratory testings nowadays. Also, because the matching between the query and each suffix is independent, the computation task can be divided into multiple sub-tasks so that a parallel processing model such as MapReduce [26] is readily applied. On the other hand, in personalized medicine, the search process, i.e., the diagnosis test like a blood test, is not required to return the real-time result. Therefore, we consider our scheme is still reasonable.

VI. Conclusion

Efficient secure sequence matching over the encrypted genetic data is the key challenge to design practical personalized medicine system in cloud computing. In this chapter, we propose a novel scheme to address the unique challenge brought by secure sequence matching over the encrypted data in cloud computing. We achieve wildcard-based sequence pattern matching through our novel modification of the predicate encryption scheme in [22]. Our proposed scheme is provable secure under the well-defined subgroup decision assumption. We greatly reduce the communication overhead to $O(1)$ round

TABLE I
PATTERN MATCHING COST IN CLOUD

	Computation time (hours)			Cost in dollar amount		
n	m=100	m=200	m=300	m=100	m=200	m=300
500	1.724	3.510	5.311	0.25$	0.42$	0.67$
1000	3.484	6.948	10.440⋆	0.42$	0.84$	1.2$

The number with ⋆ is estimated value.
The dollar amount is based on the hourly rate of an Amazon EC2 M4.large instance.

compared with the existing schemes. Although the computation complexity of our scheme is similar to the existing schemes, we further optimize our encryption algorithm to reduce the computation overhead. Through implementation and simulation, we show that our solution is feasible and practical.

ACKNOWLEDGMENT

This work of Wang, Lou, and Hou was supported in part by the US NSF under grants CNS-1446478, CNS-1405747, and CNS-1217889. The work of W. Song was supported in part by NSFC under grants 61202034, 61232002, and 61572378.

REFERENCES

[1] X. Zhou, B. Peng, Y. Li, Y. Chen, H. Tang, and X. Wang, "To release or not to release: Evaluating information leaks in aggregate human-genome data," in *Computer Security ESORICS 2011*, ser. Lecture Notes in Computer Science, V. Atluri and C. Diaz, Eds., 2011, vol. 6879, pp. 607–627.

[2] M. Gymrek, A. L. McGuire, D. Golan, E. Halperin, and Y. Erlich, "Identifying personal genomes by surname inference," *Science*, vol. 339, no. 6117, pp. 321–324, 2013.

[3] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proceedings of the 41st annual ACM symposium on Symposium on theory of computing-STOC\'09*. ACM Press, 2009, pp. 169–169.

[4] M. J. Atallah and J. Li, "Secure outsourcing of sequence comparisons," *International Journal of Information Security*, pp. 277–287, 2005.

[5] J. R. Troncoso-Pastoriza, S. Katzenbeisser, and M. Celik, "Privacy preserving error resilient dna searching through oblivious automata," ser. CCS '07. New York, NY, USA: ACM, 2007, pp. 519–528.

[6] S. Jha, L. Kruger, and V. Shmatikov, "Towards practical privacy for genomic computation," in *Security and Privacy, 2008. SP 2008. IEEE Symposium on*, May 2008, pp. 216–230.

[7] R. Wang, X. Wang, Z. Li, H. Tang, M. K. Reiter, and Z. Dong, "Privacy-preserving genomic computation through program specialization," ser. CCS '09. New York, NY, USA: ACM, 2009, pp. 338–347.

[8] M. Blanton and M. Aliasgari, "Secure outsourcing of dna searching via finite automata," in *Data and Applications Security and Privacy XXIV*, ser. Lecture Notes in Computer Science, S. Foresti and S. Jajodia, Eds. Springer Berlin Heidelberg, 2010, vol. 6166, pp. 49–64.

[9] M. Chase and E. Shen, "Substring-searchable symmetric encryption," *Proceedings on Privacy Enhancing Technologies*, vol. 2015, no. 2, pp. 263–281, 2015.

[10] J. Katz, A. Sahai, and B. Waters, "Predicate encryption supporting disjunctions, polynomial equations, and inner products," in *Advances in Cryptology–EUROCRYPT 2008*. Springer, 2008, pp. 146–162.

[11] X. S. Wang, Y. Huang, Y. Zhao, H. Tang, X. Wang, and D. Bu, "Efficient genome-wide, privacy-preserving similar patient query based on private edit distance," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2015, pp. 492–503.

[12] W. Lu, Y. Yamada, and J. Sakuma, "Efficient secure outsourcing of genome-wide association studies," in *Security and Privacy Workshops (SPW), 2015 IEEE*. IEEE, 2015, pp. 3–6.

[13] Y. Chen, B. Peng, X. Wang, and H. Tang, "Large-scale privacy-preserving mapping of human genomic sequences on hybrid clouds." in *NDSS*, 2012.

[14] P. Baldi, R. Baronio, E. De Cristofaro, P. Gasti, and G. Tsudik, "Countering gattaca: Efficient and secure testing of fully-sequenced human genomes," in *Proceedings of the 18th ACM Conference on Computer and Communications Security*, ser. CCS '11. New York, NY, USA: ACM, 2011, pp. 691–702.

[15] E. Ayday, J. L. Raisaro, U. Hengartner, A. Molyneaux, and J.-P. Hubaux, "Privacy-preserving processing of raw genomic data," in *Data Privacy Management and Autonomous Spontaneous Security*. Springer, 2014, pp. 133–147.

[16] M. Kantarcioglu, W. Jiang, Y. Liu, and B. Malin, "A cryptographic approach to securely share and query genomic sequences," *Information Technology in Biomedicine, IEEE Transactions on*, vol. 12, pp. 606–617, Sept 2008.

[17] D. X. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *Security and Privacy, Proceedings. 2000 IEEE Symposium on*, 2000, pp. 44–55.

[18] D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano, "Public key encryption with keyword search," in *EUROCRYPT 2004*, ser. Lecture Notes in Computer Science, C. Cachin and J. Camenisch, Eds. Springer Berlin Heidelberg, 2004, vol. 3027, pp. 506–522.

[19] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky, "Searchable symmetric encryption: Improved definitions and efficient constructions," ser. CCS '06. New York, NY, USA: ACM, 2006, pp. 79–88.

[20] N. Cao, C. Wang, M. Li, K. Ren, and W. Lou, "Privacy-preserving multi-keyword ranked search over encrypted cloud data," in *INFOCOM, 2011 Proceedings IEEE*, April 2011, pp. 829–837.

[21] B. Wang, S. Yu, W. Lou, and Y. Hou, "Privacy-preserving multi-keyword fuzzy search over encrypted data in the cloud," in *INFOCOM, 2014 Proceedings IEEE*, April 2014, pp. 2112–2120.

[22] J. Katz, A. Sahai, and B. Waters, "Predicate encryption supporting disjunctions, polynomial equations, and inner products," in *Advances in Cryptology–EUROCRYPT 2008*. Springer, 2008, pp. 146–162.

[23] A. D. Caro and V. Iovino, "jpbc: Java pairing based cryptography," in *Proceedings of the 16th IEEE Symposium on Computers and Communications, ISCC 2011*. Kerkyra, Corfu, Greece, June 28 - July 1: IEEE, 2011, pp. 850–855.

[24] B. Lynn, "On the implementation of pairing-based cryptosystems," Ph.D. dissertation, Stanford University, 2007.

[25] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *EUROCRYPT 99*, ser. Lecture Notes in Computer Science, 1999, vol. 1592, pp. 223–238.

[26] J. Dean and S. Ghemawat, "Mapreduce: Simplified data processing on large clusters," *Commun. ACM*, vol. 51, no. 1, pp. 107–113, Jan. 2008.

[27] E. Barker, W. Burr, A. Jones, T. Polk, S. Rose, M. Smid, and Q. Dang, "Recommendation for key management part 3: Application-specific key management guidance," *NIST special publication*, vol. 800, p. 57, 2009.

[28] P. Flicek, M. R. Amode, D. Barrell, K. Beal, K. Billis, S. Brent, D. Carvalho-Silva, P. Clapham, G. Coates, S. Fitzgerald *et al.*, "Ensembl 2014," *Nucleic acids research*, p. gkt1196, 2013.