Efficient and Privacy-preserving Similar Patients Query Scheme over Outsourced Genomic Data

Dan Zhu, Hui Zhu, *Senior Member, IEEE,* Xiangyu Wang, Rongxing Lu, *Fellow, IEEE,* and Dengguo Feng

Abstract—Over the past decade, genomic data has grown exponentially and is widely used in promising medical and health-related applications, which opens up new opportunities for the field of medicine. Similar patients query (SPQ), which can help physicians formulate an optimal therapy, is one of such popular applications. Despite its popularity, since human genomes are usually highly sensitive, a series of policies have been launched by the government to strictly control its acquisitions and utilization. Thus, how to prevent privacy disclosure becomes of great importance to the flourish of SPQ services. In this paper, aiming at the above challenge, we first design a novel genetic BK-tree (GBK-tree) for a genomic database. Then, combined with a random sorting mechanism and some existing encryption techniques, we propose an efficient and privacy-preserving similar patients query scheme over encrypted cloud data, named CASPER. With CASPER, a medical institution can securely outsource its private genomic database to a cloud server, and physicians can request SPQ services from the cloud server while keeping her/his query secret. Detailed security analysis shows that CASPER can preserve privacy in the presence of different threats. Furthermore, extensive performance evaluations demonstrate the high accuracy and efficiency of our proposed scheme.

Index Terms—Genomic data, similar patients query, privacy-preserving, genetic BK-tree, approximate edit distance.

1 Introduction

With the booming of high-throughput sequencing technology, the cost for sequencing per human genome has been reduced by a factor of 100 in the past ten years [1], and a huge amount of available genomic data has been generated in the medical domain. To better make use of human genomic information, various popular services based on genomic data have attracted increasing attention [2], [3], [4], [5], [6], [7], [8]. Among them, similar patients query (SPQ) services, which aim to find the patients whose genomic data similar to a new patient, is becoming more and more popular [9], [10], [11], [12], [13]. Meanwhile, edit distance [14], as one of the most important and frequently-used metrics for human genomic research, is usually considered as an indispensable footstone in SPQ. Once SPQ is applied, a physician can initially propose an optimal therapy for a new patient according to her/his similar patients' data and treatments. As shown in Fig. 1, when a physician requests SPQ service, she/he first needs to collect a DNA sample from the new patient and submit the extracted genomic data to her/his medical institution. Then, the medical institution performs similarity calculations with all genomic data stored in the database. Finally, the physician can receive the identifiers of the similar patients.

However, due to the massiveness and high dimensionality of genomic data (the human genome is always composed of about 2910 million base pairs [15]), SPQ's computation tasks are usually delivered to one or more third parties. Nevertheless, the genomic data of everyone is unique, the identity and close relatives of a target person can be inferred by analyzing her/his genomes [16], [17]. Therefore, the flourish of similar patients query service is still confronted with some serious obstacles including the privacy and security problem of genomic data [17], [18], [19], [20]. For one thing, the genomic database maintained by a medical institution is usually composed of large-scale patients' identifiers and genomes. Once directly outsourcing them to a third party with ulterior motives, sensitive patients' individual information may be exposed, which will undoubtedly disturb patients. For example, patients suffering from certain diseases may be discriminated against by different industries in the process of employment. For another, the query sent by a physician always contains a new patient's genomic data. In addition to the original genomes, the patient's potential

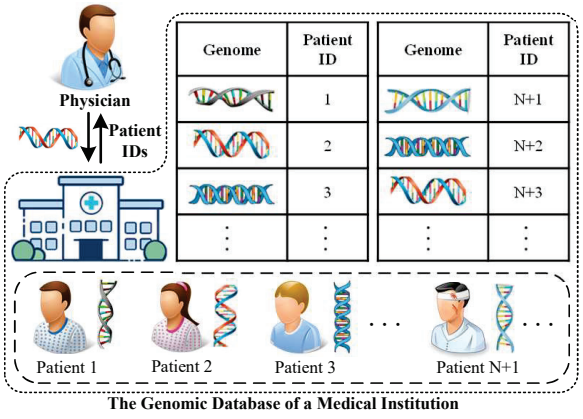


Fig. 1. Conceptual architecture of similar patients query service

- *D. Zhu, H. Zhu, X. Wang are with the National Key Laboratory of Integrated Services Networks, Xidian University, Xi'an, 710071, China (e-mail: zhudan@stu.xidian.edu.cn, zhuhui@xidian.edu.cn, xywang_xidian@163.com, corresponding author: Hui Zhu).*
- *R. Lu is with the Faculty of Computer Science, University of New Brunswick, Fredericton, NB E3B 5A3, Canada (e-mail: rlu1@unb.ca).*
- *D. Feng is with the State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing, 100190, China (e-mail: fengdg@263.net).*

TABLE 1
Comparison with prior work

Features	EPISODE [8]	Mahdi et al. [11]	Zhu et al. [13]	EFSS [7]	GENSETS [9]	CASPER
Similarity metric	AED-1	Hamming distance	Cosine distance	AED-2	AED-2	AED-2
Search method	kNN	Compressed prefix tree	Hierarchical index tree	kNN	kNN	GBK-tree
Sub-linear query latency	✗	✓	✓	✗	✗	✓
Efficiency	Low	Medium	Low	Medium	Low	High
Security	IND-SCPA	IND-SCPA	IND-SCPA	IND-SCPA	IND-SCPA	IND-SCPA

Notations. AED-1and AED-2 correspond to two different computation methods of the approximate edit distance. IND-SCPA represents the indistinguishability secure under chosen-plaintext attacks.

genetic disease will also be obtained by the third party. Under the temptation of interests, the third party is likely to disclose users' information to an insurance company, and then the insurance company can refuse the health coverage of relevant users to avoid economic losses. In a word, the untrusted third parties will increase the privacy risk of SPQ while reducing the computing burden on the terminals of the medical institution and physicians.

To address the above-mentioned challenges, plenty of privacy-preserving schemes have been proposed and achieved some remarkable results [21]. Specifically, the straightforward solution is to make use of homomorphic encryption technology, such as BGV [22] and YASHE [23], to encrypt all genomic data as performed in [24], [25]. However, due to involving lots of time-consuming operations, existing homomorphic encryption techniques always bring massive computation cost. Aiming at improving the efficiency of genomic data similarity computation, some solutions which can speed up the (approximate) edit distance secure computation have been applied over genomic data [8], [9], [10], [26], [27], [28]. Besides, from the perspective of reducing the search time, some scholars paid attention to introducing data structures based on different distance metrics [11], [13], [29], [30], such as hamming distance and cosine distance. However, due to the complexity of (approximate) edit distance, it has not been taken into consideration.

In this paper, based on an improved approximate edit distance computation method, we first design a novel data structure called genetic BK-tree (GBK-tree) for a genomic database. Then combined with a random sorting mechanism, HMAC and symmetric-key encryption (SKE), we present an efficient and privacy-preserving similar patients query scheme over encrypted cloud data, named CASPER. With CASPER, the medical institution can securely outsource its genomic database to a cloud server in the form of an encrypted GBK-tree, and the physicians can request SPQ service without revealing their patients' genomic data. Meanwhile, the cloud sever can store the encrypted GBK-tree and return high-precise results to physicians. Specifically, the main contributions of this paper are fourfold.

- *Novel genetic BK-tree.* We propose a novel GBK-tree as the basis of CASPER. Specifically, we first design an Improved genomic Edit Distance Approximate computation algorithm (IEDA). Then, based on IEDA, we construct a GBK-tree over converted genomes, which can provide more precise and efficient similar patients query services.
- *Privacy-preserving SPQ services.* CASPER keeps all patients' original genomes secret during the search process. When CASPER is applied, both the genomic data stored in a medical institution and generated by new patients are converted and encrypted by random sorting, HMAC and SKE, which cannot be obtained by a cloud server or a malicious third party.
- *High-precise query results.* By additionally considering two special situations to improve an existing approximate edit distance algorithm, we improve the accuracy of similarity matching result. And the experimental results on a real-world dataset show that there are almost no query errors in the query results, which means CASPER is high-accurate.
- *Efficiency.* The thorough evaluations using real-world and synthetic datasets show that the proposed IEDA does a good performance in secure approximate edit distance computation, and the construction of GBK-tree strictly reduces the computation overhead. The detailed comparative analysis demonstrates CASPER is more efficient than some state-of-the-art similar schemes.

In TABLE 1, we give a comparison of our proposed scheme and the prior work. As far as we know, CASPER is the first work which applies a data structure to genomic data by using approximate edit distance as a metric. Meanwhile, under the premise of achieving the same security level, CASPER shows the best performance.

The remainder of this paper is organized as follows. In Section 2, we present the system and threat models, and identify design goals. Then, some basic knowledge used in our work are introduced in Section 3. After that, we propose a novel genetic BK-tree in Section 4, and construct the framework of CASPER in Section 5. Next, the security and performance of the proposed scheme are analyzed in Section 6 and Section 7, respectively. Finally, we give the related work in Section 8 and draw a conclusion in Section 9.

2 Models and Design Goals

In this section, we formalize the system model, threat model, and identify our design goals.

2.1 System model

The system model mainly focuses on how to construct an encrypted GBK-tree over a genomic database owned by a medical institution, and provide secure similar patients query services for registered physicians via a cloud server. The medical institution and each physician are equipped with a PC/mobile device, which can be used for local

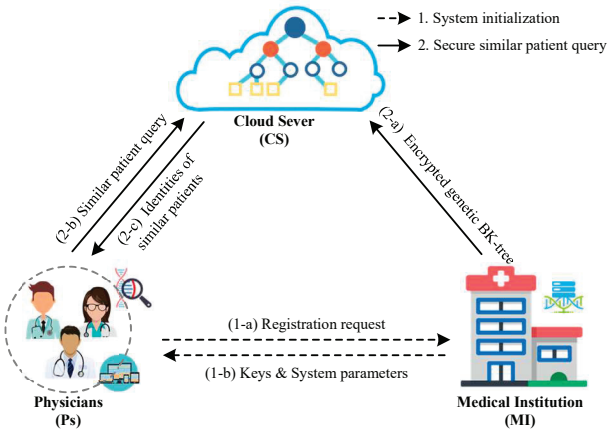


Fig. 2. System model under consideration

computation and communication. Particularly, as shown in Fig. 2, the system consists of three main entities: physicians (Ps), a medical institution (MI), and a cloud server (CS).

- Ps: Ps is a collection of physicians, in which each physician has registered in MI (as step 1-a, 1-b). Only a registered P can generate compliance encrypted queries according to patients' private genomic data, and further enjoy the secure SPQ services offered by CS (as step 2-b, 2-c).
- MI: MI owns a genomic database which stores plenty patients' genomic data. To provide Ps with efficient SPQ services and keep the original genomic database confidential, MI will generate an encrypted GBK-tree and upload it to CS for searching similar patients (as step 2-a).
- CS: CS has abundant storage space and powerful computing ability, which is also regarded as a link between Ps and MI. For one thing, it can store the encrypted GBK-tree uploaded by MI (as step 2-a). For another, it can perform calculations over stored cloud data to provide search services for Ps (as step 2-c).

2.2 Threat model

In the threat model, we consider that CS is *honest-but-curious* [31], MI and Ps are *trusted*. Specifically, CS will execute operations honestly to provide SPQ services, but it may try to analyze encrypted GBK-tree and query requests to obtain more sensitive information. MI will faithfully provide CS with an encrypted GBK-tree, and Ps will follow the protocol strictly to launch a query request. Besides, we assume there is no collusion between MI/Ps and CS. Note that, there may be some other attacks such as poisoning attacks and denial of service during the service. However, since we focus on the privacy and efficiency of SPQ services, those active or passive attacks are beyond the scope of this paper and will be discussed in our future work. Considering the above security issues, we aim to achieve the following security requirements in this work.

- *R1*: The genomic database owned by MI, which contains patients' identifiers and genomes, is the private property of MI. Therefore, CS should not be able to infer any information about the database by analyzing the stored encrypted GBK-tree.
- *R2*: The queries submitted by Ps involve new patients' sensitive information, which should not be available to CS. Thus, according to the received encrypted queries, CS should not obtain any private information about new patients.
- *R3*: The query results returned to Ps consists of all similar patients' identifiers, which can assist physicians in locating the corresponding patients. Therefore, even if CS owns the final results, it should has no ability to read the identifiers.

2.3 Design goals

In this paper, we are devoted to designing an efficient and privacy-preserving similar patients query scheme under the above-mentioned system and threat models. Specifically, the following privacy and performance goals should be met simultaneously.

- *Privacy Preservation.* Since a large amount of personal privacy information is hidden in human genomes, privacy preservation is the basic requirement of our proposed scheme. That means, whether the genomic data stored in a medical institution or newly extracted by a physician, it only can be accessed by themselves. Besides, the query results consists of similar patients' identifiers should not be read by CS.
- *High accuracy.* The proposed scheme is designed for SPQ services, based on the query result, Ps can initially determine an optimal therapy for a new patient. Thus, the accuracy of the query results directly affects the availability of the proposed scheme. In order to help Ps formulate a feasible treatment, the algorithm used by CS to match similar genomes should be as accurate as possible.
- *Efficient performance.* Although the storage and computing capabilities of CS are thought as powerful, considering that multiple physicians may request at the same time and a large-scale GBK-tree would be uploaded to CS, we are still committed to making the process of searching similar patients as efficient as possible to provide Ps with quality real-time services.

3 PRELIMINARIES

In this section, we briefly review the edit distance algorithm [14], BK-tree data structure [32], a genomic edit distance approximation algorithm [9] and symmetric-key encryption [33]. These will serve as the basis of our scheme.

3.1 Edit distance algorithm

Edit distance [14] is an algorithm to find minimum number of operations (insertion, deletion, and substitution) required to convert one string s_1 into the other string s_2. Denote the lengths of s_1 and s_2 as l_1 and l_2, respectively. Then, we can convert the process of calculating edit distance to the process of filling a $(l_1 + 1) \times (l_2 + 1)$ matrix $\mathcal{M}$ with each

element $(m_{i,j})_{0\leq i\leq l_1,0\leq j\leq l_2}$. And $m_{i,j}$ can be achieved by the following equation:

$$m_{i,j} = \begin{cases} 0, & \text{if } i = 0 \text{ and } j = 0; \\ m_{i,j-1} + 1, & \text{if } i = 0 \text{ and } j > 0; \\ m_{i-1,j} + 1, & \text{if } i > 0 \text{ and } j = 0; \\ min \begin{cases} m_{i,j-1} + 1, \\ m_{i-1,j} + 1, \\ m_{i,j} + f, \end{cases} & \text{if } i > 0 \text{ and } j > 0, \end{cases}$$

where f takes the value of 0 if $s_1[i-1] = s_2[j-1]$ and 1 otherwise. Finally, the value of m_{l_1,l_2} which represents the edit distance between s_1 and s_2 can be achieved. The details of edit distance computation can be found in **Algorithm 1**, where $ED(\cdot,\cdot)$ denotes the edit distance between two strings.

Algorithm 1: Edit distance computation

Input: Two strings s_1, s_2.
Output: Edit distance between s_1 and s_2, $ED(s_1, s_2)$.

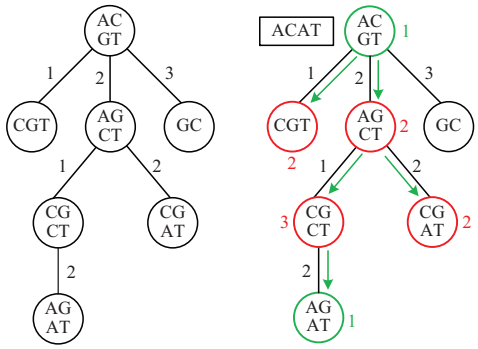
```
compute l1 = len(s1), l2 = len(s2);
initialize a (l1 + 1) × (l2 + 1) matrix M with each
  element M[i][j] = 0, 0 ≤ i ≤ l1, 0 ≤ j ≤ l2;
for 0 ≤ i ≤ l1 do
    for 0 ≤ j ≤ l2 do
        if i == 0 then
            M[i][j] = j;
        else if j == 0 then
            M[i][j] = i;
        else if s1[i − 1] == s2[j − 1] then
            M[i][j] = M[i − 1][j − 1];
        else
            M[i][j] = 1 + min{M[i − 1][j], M[i][j −
              1], M[i − 1][j − 1]};
return M[l1][l2].
```

3.2 BK-tree data structure

BK-tree [32] is proposed for searching a set of strings $\mathcal{S}$ to find some close strings to a given query string with edit distance. Similar to other trees, BK-tree consists of nodes and edges, the nodes represent all elements in $\mathcal{S}$, and the edges with some integer weights indicate the edit distance from one node to another. For example, if the edge from node s_1 to node s_2 has a weight w ($s_1, s_2 \in \mathcal{S}$), the integer w represents the edit distance between s_1 and s_2. Moreover, the construction of BK-tree should satisfy two conditions: 1) an arbitrary element $s \in \mathcal{S}$ can be selected as the root node; 2) for any sub-tree with a parent node s_p, all nodes in the sub-tree have the same edit distance from s_p.

After constructing a BK-tree, the close strings for a given query string q should be found out by searching the tree. Particularly, a tolerance limit $\mathbb{T}$ which represents the maximum edit distance from s_q to the strings in $\mathcal{S}$ should be set at first. Then starting from the root node, for each node which can be seen as a parent node, its specific child nodes need to be searched lie within the tolerance limit. For a parent node s_p, $ed = ED(s_p, s_q)$, instead of iterating over all s_p's children, only children with edge-weight in range $[ed-\mathbb{T}, ed+\mathbb{T}]$ need to be iterated over. And just the children whose edit distances from s_q not larger than $\mathbb{T}$ are the close strings to s_q.

To better illustrate how to construct and search a BK-tree, and considering that each unit on human genome is a nucleotide (A, C, T or G), an example is given as follows. We set $\mathcal{S} = \{CGT, AGCT, CGCT, GC, ACGT, CGAT, AGAT\}$, $q = ACAT$ and the tolerance limit $\mathbb{T} = 1$, then Fig. 3(a) and 3(b) show the final data structure and query result $\{AGAT\}$, respectively.



(a) Construction. (b) Search.

Fig. 3. An example of BK-tree construction and search process.

3.3 Genomic edit distance approximation algorithm

The genomic edit distance approximation algorithm [9] is designed for approximating the edit distance between two genomes, which relies on the size of the symmetric difference between the two genomes' single-character edits sets. The whole approximation algorithm consists of the following three steps.

Step 1: Choose a public reference genome *Ref*, then compress two genomes G_A and G_B into two sets of edits vcf_A and vcf_B through recording the minimum edits sequence to derive G_A and G_B from *Ref*, respectively. As a toy example, assuming the public reference genome $Ref = ACGTAAC$, the genome sequence $ATGGTA$ can be converted into a set of multi-character edits $vcf = \{(1, ins, T), (2, sub, G), (6, del, 2)\}$, where ins, sub and del denote operations insert, substitute and delete in respective.

Step 2: Decompose each edit in the multi-character edits sets into a single-character edit to form $vcf_A{}'$ and $vcf_B{}'$. Specifically, denote a multi-character edit as (loc, op, val), where loc is the edit location on *Ref*, op is the type of operation (ins, sub and del), and val is characters or values. For different types of edit, the decomposition follows different rules.

- Delete: Deleting a θ-length string at location loc, denoted as (loc, del, θ), is converted into $\{(loc, del, \perp), \ldots, (loc + \theta - 1, del, \perp)\}$.
- Insert: Inserting a θ-length string $(s_1 \ldots s_\theta)$ at location loc, denoted as $(loc, ins, s_1 \ldots s_\theta)$, is converted into $\{(loc, ins, 1, s_1), \ldots, (loc, ins, \theta, s_\theta)\}$.
- Substitute: Substituting the character at location loc with a character s, denoted as (loc, sub, s), is originally a single-character edit, no conversion required.

TABLE 2
Definition of main variables in GBK-tree

Variables	Definition
GD	A genomic database consists of n genomes.
Ref/G_0	The public reference genome randomly chosen from GD.
vcf_0	A set of three-dimensional vectors converted by genome G_0, and $vcf_0 = \varnothing$.
$vcf_{k,0<k<n}$	A set of three-dimensional vectors converted by genome G_k.
$ED(\cdot,\cdot)$	The edit distance between two genomes.
$\lvert\cdot\rvert$	The length of a set or a string.
$Aed(\cdot,\cdot)$	The approximate edit distance between two genomes.
$\mathcal{T}$	A GBK-tree constructed based on the converted edits sets $\{vcf_0,\ldots,vcf_{n-1}\}$.
$\mathbb{T}$	A tolerance limit.

Step 3: Compute the size of symmetric set difference between the two single-character edits sets $vcf_A{}'$ and $vcf_B{}'$, $Diff(vcf_A{}', vcf_B{}') = (vcf_A{}' - vcf_B{}') \cup (vcf_B{}' - vcf_A{}')$, and output it as the approximation edit distance between G_A and G_B.

3.4 Symmetric-key encryption

Symmetric key algorithms [33] are algorithms for cryptography that use the same cryptographic keys for both the process of encryption and decryption. Mathematically, symmetric-key encryption may be described as follows.

Definition 1. *A symmetric-key encryption scheme consists of a map $\Pi : K \times M \to C$, such that for each $k \in K$, the map $\Pi_k : M \to C, m \to c = \Pi(k, m)$ is invertible. The elements $m \in M$, $c \in C$ and $k \in K$ are the plaintexts, ciphertexts and keys, respectively. Π_k is the encryption function with respect to the key k, $\Delta_k := \Pi_k^{-1}$ is the decryption function. It is assumed that efficient algorithms to compute Π_k and Δ_k exist.*

4 Proposed Genetic BK-tree

In this section, we first design an improved genomic edit distance approximation algorithm. Then, we propose a novel genetic BK-tree (GBK-tree) for a genomic database, which serves as the building block of CASPER. Meanwhile, for easier expression, we give the description of the main notations used in the process in TABLE 2.

4.1 Improved genomic edit distance approximation algorithm

Although **Algorithm 1** has given the way to compute the edit distance between two strings, it cannot show the specific edits of converting s_1 into s_2. However, to calculate the approximate edit distance between two genomes, the specific edits are needed. Thus, in this subsection, based on the matrix $\mathcal{M}$ generated from **Algorithm 1**, we first give the details of converting original genomes to single-character edits sets, then present the process of computing more-precise approximate edit distance over converted genomes.

Conversion. Given a public reference genome *Ref*, two genomes G_A and G_B, based on the Wagner-Fischer algorithm [34], we introduce a method to record the minimum edits sequence from *Ref* to G_A and G_B, the process of G_A's conversion is chosen as an example.

- Firstly, calculate $ED(Ref, G_A)$ through filling a matrix $\mathcal{M}$ with elements $(m_{x,y})_{0\le x\le l_1, 0\le y\le l_2}$ according to **Algorithm 1**. Obviously, $ED(Ref, G_A) = m_{l_1,l_2}$.
- Secondly, start from $m_{x,y}$, $x = l_1, y = l_2$, if *Ref*'s x-th character $Ref[x-1]$ is equal to G_A's y-th character $G_A[y-1]$, no edit is required at location x of *Ref*, skip to $m_{x-1,y-1}$. Otherwise, we first check whether $m_{x,y} = m_{x-1,y-1} + 1$, if it does, $(x, sub, G_A[y-1])$ is added to vcf_A, skip to $m_{x-1,y-1}$; if it does not and $m_{x,y} = m_{x,y-1} + 1$, $(x, ins, G_A[y-1])$ is added to vcf_A, skip to $m_{x,y-1}$; if it does not and $m_{x,y} = m_{x-1,y}+1$, $(x, del, \perp)$ is added to vcf_A, skip to $m_{x-1,y}$. Update x, y and repeat the above operation to expand vcf_A until $x = y = 0$.
- Thirdly, sort all added single-character edits in the final vcf_A in ascending location number. However, at the same location, there may exist multiple operations ins, which will hinder the design of CASPER. To solve this problem, we convert the set $\{(loc, ins, s_1), (loc, ins, s_2), \ldots, (loc, ins, s_\theta)\}$ into $\{(loc, ins, s_1), (loc, ins + r, s_2), \ldots, (loc, ins + r + (\theta - 2), s_\theta)\}$, where r is a randomly chosen number. Note that (loc, ins, s_b) must be added in vcf_A earlier than (loc, ins, s_a) for $\forall a < b \in [1, \theta]$.

After finishing the genomes' conversion, both vcf_A and vcf_B are sets consisting of single-character edits:

$$\{(loc^{(\eta)}_{\{A,B\}}, op^{(\eta)}_{\{A,B\}}, val^{(\eta)}_{\{A,B\}}) | \eta \in [1, l_{\{A,B\}}]\},$$

where $loc^{(\eta)}_{\{A,B\}} \in [0, l_1]$, $op^{(\eta)}_{\{A,B\}} \in \{sub, del, ins, ins + r, ins + r + 1, \ldots\}$, $val^{(\eta)}_{\{A,B\}} \in \{A, G, C, T, \perp\}$ and $l_{\{A,B\}} = |vcf_{\{A,B\}}| = ED(Ref, G_{\{A,B\}})$.

To make this process clearer, a toy example is given in Fig 4. Assume that $Ref = ACGT$, $G_A = CAAGGT$, due to $m_{2,4} = m_{1,3} + 1$, $m_{0,2} = m_{0,1} + 1$, $m_{0,1} = m_{0,0} + 1$, the result $vcf_A = \{(0, ins, C), (0, ins + r, A), (2, sub, G)\}$ can be achieved.

		'C'	'A'	'A'	'G'	'G'	'T'
	0	1	2	3	4	5	6
'A'	1	1	1	2	3	4	5
'C'	2	1	2	2	3	4	5
'G'	3	2	2	3	2	3	4
'T'	4	3	3	3	3	2	3

$vcf_A = \{(0, ins, C), (0, ins + r, A), (2, sub, G)\}$

Fig. 4. An example of converting G_A to vcf_A

Computation. To obtain the approximate edit distance from G_A to G_B, the solution given in [9] is to directly compute the value of $Diff(vcf_A, vcf_B)$, which inevitably leads to the loss of accuracy. In order to make up for the lack, an improved genomic edit distance approximate computation algorithm (IEDA) is proposed as shown in **Algorithm 2**.

- Firstly, compute a set $Loc = \{loc_1, loc_2, \ldots, loc_\gamma\} = \{loc_A^{(1)}, \ldots, loc_A^{(l_A)}\} \bigcap \{(loc_B^{(1)}, \ldots, loc_B^{(l_B)}\}$, in which $loc_1 < loc_2 < \cdots < loc_\gamma$. Then, all single-character edits with $loc_A^{(\alpha)} \in Loc$, $\alpha \in [1, l_A]$ are extracted to form a new set $vcf_A{}'$, and all edits with $loc_B^{(\beta)} \in Loc$, $\beta \in [1, l_B]$ are extracted to form a new set $vcf_B{}'$, too. Besides, whether an edit is in $vcf_A{}'$ or $vcf_B{}'$, if its operation value (i.e., $op_{\{A,B\}}^{(\alpha)}$) is not equal to del or sub or ins (e.g. $ins + r$), there should be an edit with the same values of location and operation in the other set; otherwise the edit will be deleted from its set. And the edits in the two sets are sorted in increasing order of location value, if the location values are same, they will be sorted in increasing order of operation value. After updating $vcf_A{}'$ and $vcf_B{}'$, the initial value of the approximate edit distance between G_A and G_B can be set to $Aed(G_A, G_B) = |vcf_A| + |vcf_B| - |vcf_A{}'| - |vcf_B{}'|$ ($|vcf_A{}'| = |vcf_B{}'|$).
- Secondly, compare the two edits with the same position in $vcf_A{}'$ and $vcf_B{}'$ in sequence by using the following method. Let $(loc_A^{(\alpha)}, op_A^{(\alpha)}, val_A^{(\alpha)})$ and $(loc_B^{(\beta)}, op_B^{(\beta)}, val_B^{(\beta)})$ be a pair, where $loc_A^{(\alpha)} = loc_B^{(\beta)}$. If $op_A^{(\alpha)} \neq op_B^{(\beta)}$ and $val_A^{(\alpha)} \neq val_B^{(\beta)}$, $Aed(G_A, G_B)$ is updated by $Aed(G_A, G_B) + 2$; if $op_A^{(\alpha)} \neq op_B^{(\beta)}$ but $val_A^{(\alpha)} = val_B^{(\beta)}$, updated by $Aed(G_A, G_B) + 1$. When the two single-character edits have the same operation value, i.e., $op_A^{(\alpha)} = op_B^{(\beta)}$, the value of $Aed(G_A, G_B)$ remains unchanged if $val_A^{(\alpha)} = val_B^{(\beta)}$. In addition to the above cases, if $op_A^{(\alpha)} = op_B^{(\beta)}$ but $val_A^{(\alpha)} \neq val_B^{(\beta)}$, $Aed(G_A, G_B)$ is updated by $Aed(G_A, G_B) + 1$. After completing all the comparisons, the final achieved $Aed(G_A, G_B)$ denotes the genomic approximate edit distance from G_A to G_B.

Algorithm 2: Improved genomic edit distance approximate computation (IEDA)

Input: Genomes G_A, G_B, public reference genome *Ref*.
Output: $Aed(G_A, G_B)$, the genomic approximate edit distance from G_A to G_B.

```
compress G_A and G_B into two sets of single-character edits vcf_A and vcf_B, respectively;
extract the edits in vcf_A and vcf_B that need to be compared to form two sets vcf_A' and vcf_B';
compute len = |vcf_A'| = |vcf_B'|;
initialize Aed = |vcf_A| + |vcf_B| − 2 · len;
denote vcf_A' as array A[len][3], vcf_B' as array B[len][3];
for 0 ≤ i ≤ len − 1 do
    if A[i][1]! = B[i][1] && A[i][2]! = B[i][2] then
        Aed+ = 2;
    else if A[i][1]! = B[i][1] && A[i][2] == B[i][2] then
        Aed+ = 1;
    else if A[i][1] == B[i][1] && A[i][2]! = B[i][2] then
        Aed+ = 1;
    else Aed+ = 0 ;
return Aed.
```

Correctness of improved genomic edit distance approximation algorithm. Given two genomes G_A and G_B, and converting them into vcf_A and vcf_B. Based on the experimental results given in [9], we can know that the size of symmetric set difference between vcf_A and vcf_B is very close to the edit distance from G_A to G_B, which can be expressed as $|Diff(vcf_A, vcf_B)| \approx ED(G_A, G_B)$. According to the definition of symmetric set difference, $Aed(G_A, G_B) = |Diff(vcf_A, vcf_B)| = |vcf_A| + |vcf_B| - 2 \cdot |vcf_A \cap vcf_B|$. This shows that all three-dimensional vectors which are not in vcf_A and vcf_B simultaneously, are all considered to be the edits that affect the process of converting G_A to G_B, which lowers the accuracy of approximate edit distance. Under the condition of $loc_A^{(\alpha)} = loc_B^{(\beta)}$, we further improve the accuracy by considering the following situations.

Situation 1. $op_A^{(\alpha)} \neq op_B^{(\beta)} \in \{sub, ins\}$, and $val_A^{(\alpha)} = val_B^{(\beta)} \in \{A, C, G, T\}$. When converting G_A to G_B (or G_B to G_A), only one edit with operation ins or del is needed. For example, assume $Ref_0 = AAGGT$, $G_A = ACGGT$ and $G_B = AACGGT$, thus $ED(G_A, G_B) = 1$, $vcf_A = \{(2, sub, C)\}$, $vcf_B = \{(2, ins, C)\}$. If we compute $Aed(G_A, G_B)$ based on the size of symmetric set difference, $Aed(G_A, G_B) = |Diff(vcf_A, vcf_B)| = 2$. But if we compute $Aed(G_A, G_B)$ based on **Algorithm 2**, $Aed(G_A, G_B) = 1$, which is equal to $ED(G_A, G_B)$. Specifically, the edit needed to convert G_A to G_B is $(1, ins, C)$, and the edit needed to convert G_B to G_A is $(2, del, \perp)$.

Situation 2. $op_A^{(\alpha)} = op_B^{(\beta)} \in \{sub, ins\}$, and $val_A^{(\alpha)} \neq val_B^{(\beta)} \in \{A, C, G, T\}$. When converting G_A to G_B (or G_B to G_A), only one edit with operation sub is needed. For example, assume $Ref_0 = AAGGT$, $G_A = ACGGT$ and $G_B = AGGGT$, thus $ED(G_A, G_B) = 1$, $vcf_A = \{(2, sub, C)\}$, $vcf_B = \{(2, sub, G)\}$. If we compute $Aed(G_A, G_B)$ based on the size of symmetric set difference, $Aed(G_A, G_B) = |Diff(vcf_A, vcf_B)| = 2$. But if we compute $Aed(G_A, G_B)$ based on **Algorithm 2**, $Aed(G_A, G_B) = 1$, which is equal to $ED(G_A, G_B)$. Specifically, the edit needed to convert G_A to G_B is $(2, sub, G)$, and the edit needed to convert G_B to G_A is $(2, sub, C)$. The same is true when $op_A^{(\alpha)} = op_B^{(\beta)} = ins$.

Therefore, the improved genomic edit distance approximation algorithm is correct.

4.2 A proposed novel GBK-tree

In this subsection, we focus on how to construct and search a GBK-tree for a genomic database and a given query genome. Different from BK-tree, GBK-tree utilizes approximate edit distance as a metric instead of edit distance, and its nodes store sets of single-character edits instead of strings. Specifically, given a genomic database GD with n genomes and a query genome G_p, the entire process can be divided into construction and search.

• Stage 1. The construction of GBK-tree

Firstly, a genome from GD is randomly chosen as the public reference genome *Ref* of GD. In addition to *Ref*, the remaining genomes in GD will form a new collection $GD^* = \{G_1, G_2, \ldots, G_{n-1}\}$. Based on *Ref*, all genomes included in GD^* can be converted into sets of single-character edits $\{vcf_1, vcf_2, \ldots, vcf_{n-1}\}$ by following **Conversion** in Section 4.1.

Based on the converted data $\{vcf_0, vcf_1, \ldots, vcf_{n-1}\}$, a GBK-tree $\mathcal{T}$ can be constructed, where $vcf_0 = \varnothing$ is the single-character edits set of *Ref*. Specifically, denote vcf_0 as the

root node, $\{G_0 = Ref, G_1, \ldots, G_{n-1}\}$ as the corresponding genomes of $\{vcf_0, vcf_1, \ldots, vcf_{n-1}\}$. Start from vcf_1 (i.e., k=1), $\{vcf_1, \ldots, vcf_k, \ldots, vcf_{n-1}\}$ will be added into $\mathcal{T}$ in sequence in a top-down manner, and the details are given as follow.

- *Step 1*. Initialize $k' = 0$;
- *Step 2*. According to **Computation** in Section 4.1, compute the approximate edit distance between $G_{k'}$ and G_k, i.e., $Aed(G_{k'}, G_k)$;
- *Step 3*. If there exists no child node of $vcf_{k'}$ with edge-weight equal to $Aed(G_{k'}, G_k)$, add vcf_k into $\mathcal{T}$ as a child node of $vcf_{k'}$ with edge-weight equal to $Aed(G_{k'}, G_k)$, set $k = k + 1$, return to *Step 1*; otherwise, find the child node $vcf_{\overline{k}}$ whose edge-weight is $Aed(G_{k'}, G_k)$, set $k' = \overline{k}$, return to *Step 2*.

Until adding all $\{vcf_k | k \in [1, n-1]\}$ into the GBK-tree $\mathcal{T}$, the process of construction is finished. In short, if a node conflict occurs while inserting vcf_k into the tree, the insertion process will be propagated down the children of this node until an appropriate parent of vcf_k is found, and each insertion in the tree should start from the root node Ref_0. As shown in Fig. 5, if $\mathcal{T}$ is constructed over GD, the edge-weight of each node in $\mathcal{T}$ is equal to the approximate edit distance between the two nodes connected by the edge. And the approximate edit distance from any node in a sub-tree to the parent node of the sub-tree's root node is same, for example, $Aed(G_5, G_{66}) = Aed(G_5, G_7) = 1$.

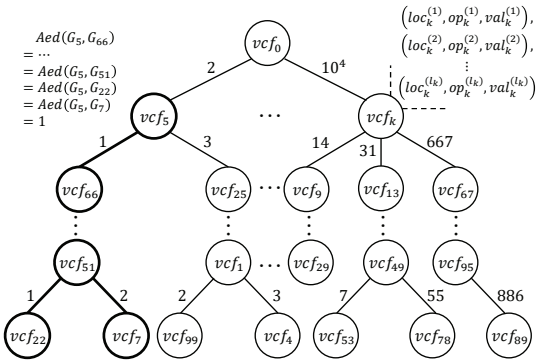


Fig. 5. A example of GBK-tree

• Stage 2. The search of GBK-tree

After completing the construction of $\mathcal{T}$, similar genomes in GD should be found for a given query genome G_p, and the maximum edit distance from these similar genomes to G_p should be not larger than a tolerance limit $\mathbb{T}$.

According to the public reference genome *Ref*, G_p can be converted into a set of single-character edits vcf_p at first. Then, we are able to form the set SD which consists of all G_p's similar genomes by executing the following operations.

- *Step 1*. Initialize SD$= \varnothing$, and the parent nodes set PN$=\{vcf_0\}$;
- *Step 2*. For each node $vcf_{k'} \in$ PN, compute the approximate edit distance between $G_{k'}$ and G_p, that is, $Aed(G_{k'}, G_p)$. If $Aed(Ref, G_p) \leq \mathbb{T}$, add $G_{k'}$ to SD; otherwise $G_{k'}$ does not belong to the similar genomes of G_p;
- *Step 3*. If PN consists of the most bottom leaf nodes, the process of search is over; otherwise, for each node $vcf_{k'} \in$ PN, calculate the range $[Aed(G_{k'}, G_p) - \mathbb{T}, Aed(G_{k'}, G_p) + \mathbb{T}]$ and pick up all its child nodes whose edge-weights belong to the range. All of the selected child nodes form a new parents nodes set PN to replace the old one, return to *Step 2*.

Finally, all G_p's similar genomes will be added to SD.

TABLE 3
Definition of main variables in CASPER

Variables	Definition
ξ	The number of segments that *Ref* is divided into.
sg_ω	The ω-th segment of *Ref*.
π_ω	The length of sg_ω.
$\mathcal{C}_{(\xi+1)\times 9}$	An index matrix that can query the assignment within segments.
k_1, k_2, k_3, k_4	Secret keys used in HMAC and SKE.
r	Random number used in edit ins.
$E(\mathcal{T})$	The encrypted form of $\mathcal{T}$.
$id_{k',0\leq k'<n}$	The identifies of the patient whose genome is $G_{k'}$.
$id^E_{k'}$	The encrypted form of identifier $id_{k'}$.
$[\![vcf_k]\!]$	The processed vcf_k through random sorting.
$E(vcf_k)$	The encrypted vcf_k via HMAC.
$F(sk, ch)$	Encrypt ch with HMAC via a secret key sk.
$l_{\{k,p\}}$	The number of elements in vcf_k or vcf_p.
ID	A set of encrypted identifies.
NS	A collection of $E(\mathcal{T})$'s nodes.

5 AN EFFICIENT AND PRIVACY-PRESERVING SIMILAR PATIENTS QUERY SCHEME

In this section, we are devoted to achieving an efficient and privacy-preserving similar patients query scheme on an outsourced GBK-tree, named CASPER, which mainly consists of four phases: *1) system initialization; 2) encrypted genetic BK-tree outsourcing; 3) search request generation* and *4) privacy-preserving similar patients query*. The overview of CASPER is described in Fig. 6. At first, MI generates system parameters and secret keys, and provides physicians with registration service (①②). Then, MI constructs a GBK-tree for its genomic database and encrypts the GBK-tree to outsource it to CS (③④). Next, the successfully registered Ps can generate encrypted queries and send them to CS for services request (⑤⑥). After receiving the encrypted queries, CS will search the encrypted GBK-tree and return Ps with similar patients' encrypted identifiers (⑦⑧). Finally, Ps read all identifiers by decrypting ciphertexts. To describe CASPER clearer, some definitions of new variables used in the following subsections are given in TABLE 3.

5.1 System initialization

In this phase, MI generates the system parameters and secret keys, and provides Ps with registration services.

At first, MI randomly chooses a genome from its genomic database GD as public reference genome $Ref = (s_1, s_2, \ldots, s_m) \in \{A, G, C, T\}^m$. Then, MI randomly divides *Ref* into ξ segments $\{sg_1, sg_2, \ldots, sg_\xi\}$, where $|sg_\omega| = \pi_\omega, \omega \in [1, \xi]$, each π_ω is a random number, $\sum_{j=1}^{\xi} \pi_j = m$. And the segments can be recorded as $sg_1 = (s_1, \ldots, s_{\pi_1})$,

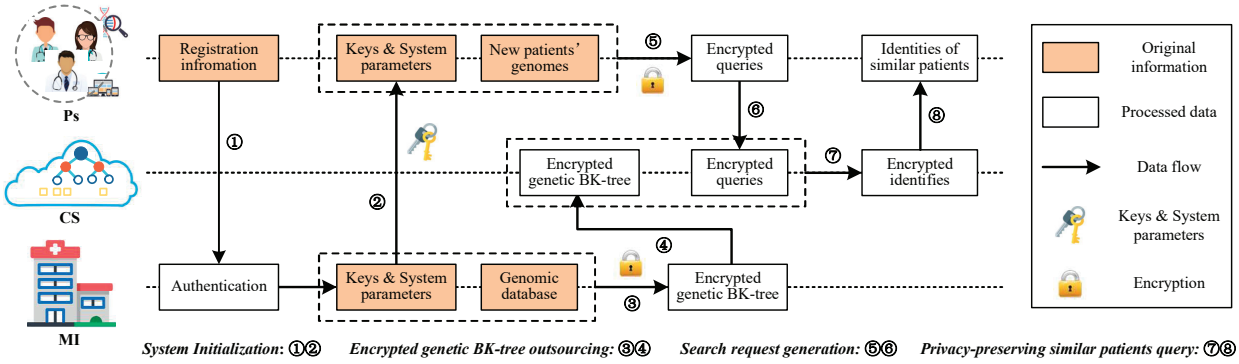


Fig. 6. Overview of CASPER

'del'	'ins'	'sub'
0	2	1
$c_{8,1}$	$c_{8,2}$	$c_{8,3}$

'A'	'C'	'G'	'T'	'⊥'
3	0	2	1	4
$c_{8,4}$	$c_{8,5}$	$c_{8,6}$	$c_{8,7}$	$c_{8,8}$

Fig. 7. Randomly sorted results in the 8-th segment

$sg_{\omega(\omega\geq 2)} = (s_{\sum_{j=1}^{\omega-1}\pi_j+1}, \ldots, s_{\sum_{j=1}^{\omega}\pi_j})$. After that, for each segment sg_ω, we randomly order each operation $op \in \{del, ins, sub\}$ and each character $val \in \{A, C, G, T, \perp\}$ which appear in the genomes conversion process, and an index matrix $\mathcal{C}_{(\xi+1)\times 9}$ can be formed as

$$\begin{bmatrix} \text{-1} & del & ins & sub & A & C & G & T & \perp \\ \dagger_1 & c_{1,1} & c_{1,2} & c_{1,3} & c_{1,4} & c_{1,5} & c_{1,6} & c_{1,7} & c_{1,8} \\ \dagger_2 & c_{2,1} & c_{2,2} & c_{2,3} & c_{2,4} & c_{2,5} & c_{2,6} & c_{2,7} & c_{2,8} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \dagger_\xi & c_{\xi,1} & c_{\xi,2} & c_{\xi,3} & c_{\xi,4} & c_{\xi,5} & c_{\xi,6} & c_{\xi,7} & c_{\xi,8} \end{bmatrix},$$

where $\dagger_\omega = \sum_{j=1}^{\omega}\pi_j$, $c_{\omega,\{1,2,3\}} \in \{0,1,2\}$, $c_{\omega,\{4,5,6,7,8\}} \in \{0,1,2,3,4\}$. And $c_{\omega,\{1,2,3\}}$ refer to the corresponding index values of operations after randomly sorting in the ω-th segment, $c_{\omega,\{4,5,6,7,8\}} \in \{0,1,2,3,4\}$ refer to the corresponding index values of characters, note that $c_{\omega,1} \neq c_{\omega,2} \neq c_{\omega,3}$, and $c_{\omega,4} \neq c_{\omega,5} \neq c_{\omega,6} \neq c_{\omega,7} \neq c_{\omega,8}$. A toy example is given in Fig. 7, it shows a possibility of the ninth row of $\mathcal{C}_{(\xi+1)\times 9}$.

Next, MI chooses an HMAC scheme, an SKE algorithm (e.g., AES) and four random secret keys k_1, k_2, k_3, k_4 which will be used in HMAC and SKE. Besides, a random number $r \geq 3$ is also selected to be used in genomes conversion process for distinguishing characters inserted in the same location of public reference genome.

Finally, MI provides physicians with registration services. After successfully verifying a physician's registration information, MI will share the chosen *Ref*, HMAC and SKE with the physician, and send the collection KP= $\langle k_1, k_2, k_3, k_4, r, \mathcal{C}_{(\xi+1)\times 9}\rangle$ to the physician.

5.2 Encrypted genetic BK-tree outsourcing

In this phase, MI first constructs a GBK-tree $\mathcal{T}$ over GD. Then with KP, HMAC and SKE, MI outsources an encrypted GBK-tree $E(\mathcal{T})$ to CS.

MI owns a genomic database GD consists of n genomes, and each genome in GD is labeled with a unique identifier of a patient. Except for the public reference genome (Ref, id_0), the remaining genomes in GD can be denoted as $\text{GD}^* = \{(G_1, id_1), \ldots, (G_{n-1}, id_{n-1})\}$.

To construct an encrypted GBK-tree $E(\mathcal{T})$, MI first converts all genomes into single-character edits sets based on *Ref*. Specifically, *Ref* is converted into $vcf_0 = \varnothing$, and the other genomes in GD* are converted into $\{vcf_1, \ldots, vcf_{n-1}\}$. After that, based on $\{vcf_0, \ldots, vcf_{n-1}\}$, a GBK-tree $\mathcal{T}$ can be constructed through the operations in **Stage 1** of Section 4.2. Next, for each

$$vcf_k = \{(loc_k^{(1)}, op_k^{(1)}, val_k^{(1)}), \ldots, (loc_k^{(l_k)}, op_k^{(l_k)}, val_k^{(l_k)})\}$$

stored in $\mathcal{T}$, where $k \in [1, n-1]$, $l_k = ED(Ref, G_k)$, the following steps will be performed to encrypt $\mathcal{T}$.

- *Step 1:* For each edit $(loc_k^{(\eta)}, op_k^{(\eta)}, val_k^{(\eta)}), \eta \in [1, l_k]$ in vcf_k, we first check which segment of *Ref* the location $loc_k^{(\eta)}$ belongs to. If $\mathcal{C}_{(\omega-1),0} + 1 \leq loc_k^{(\eta)} \leq \mathcal{C}_{\omega,0}$, $\omega \in [1, \xi]$, $op_k^{(\eta)}$ and $val_k^{(\eta)}$ will be replaced by the corresponding values in the ω-th row of $\mathcal{C}_{(\xi+1)\times 9}$. Moreover, if r appears in $op_k^{(\eta)}$, it will be replaced by the value in KP. After completing all replacements in vcf_k, the processed vcf_k can be denoted as

$$[\![vcf_k]\!] = \{([\![loc_k^{(\eta)}]\!], [\![op_k^{(\eta)}]\!], [\![val_k^{(\eta)}]\!]) | \eta \in [1, l_k]\}.$$

  Example. Assume that $vcf_k = \{(0, ins, C), (0, ins + r, A), (2, sub, G), (7, del, \perp)\}$, *Ref* is divided into ξ ($\xi \geq 3$) segments. And the random number r is assigned a value of 5, the 0-th, 1-th, 2-th rows of $\mathcal{C}_{(\xi+1)\times 9}$ are set to $(-1, del, ins, sub, A, C, G, T, \perp)$, $(6, 1, 0, 2, 4, 2, 3, 0, 1)$, $(17, 2, 1, 0, 1, 0, 3, 2, 4)$ in respective. Thus, the processed vcf_k can be obtained as $[\![vcf_k]\!] = \{(0, 0, 2), (0, 5, 4), (2, 2, 3), (7, 2, 4))\}$.
- *Step 2:* For each $([\![loc_k^{(\eta)}]\!], [\![op_k^{(\eta)}]\!], [\![val_k^{(\eta)}]\!])$ in $[\![vcf_k]\!]$, MI takes $(k_1, [\![loc_k^{(\eta)}]\!])$, $(k_2, [\![op_k^{(\eta)}]\!])$ and $(k_3, [\![val_k^{(\eta)}]\!])$ as input respectively to calculate $F(sk, ch)$, where $F(sk, ch) = E_{sk}(ch) = \text{HMAC}(sk, ch)$, sk refers to the secret key $\{k_1, k_2, k_3\}$, ch refers to the elements in

the processed edits sets. Then, the encrypted vcf_k (i.e., $E(vcf_k)$) can be represented as

$$\{(E_{k_1}(loc_k^{(\eta)}), E_{k_2}(op_k^{(\eta)}), E_{k_3}(val_k^{(\eta)}))|\eta \in [1, l_k]\}.$$

Finally, MI encrypts all identifiers by SKE via k_4 and outsources the encrypted GBK-tree $E(\mathcal{T})$, whose nodes are $(E(vcf_{k'}), id_{k'}^E), k' = \{0, 1, \ldots, n-1\}$, to CS, where $id_{k'}^E$ is an encrypted identifier.

5.3 Search request generation

In this phase, each registered P encrypts their new patients' genomes to generate query requests, and sends the search requests to CS for requesting SPQ services.

After successfully registering in MI, P will receive the chosen *Ref*, HMAC, SKE and the collection KP= $\langle k_1, k_2, k_3, k_4, r, \mathcal{C}_{(\xi+1)\times 9}\rangle$ from MI.

Assume G_p is a new patient's genome. To generate a search request for G_p, P first converts G_p into vcf_p based on *Ref*, where $vcf_p = \{(loc_p^{(\varsigma)}, op_p^{(\varsigma)}, val_p^{(\varsigma)})|\varsigma \in [1, l_p]\}$, $l_p = ED(Ref, G_p)$. Then, in the same way as $[\![vcf_k]\!]$ described in Section 5.2, for each edit $(loc_p^{(\varsigma)}, op_p^{(\varsigma)}, val_p^{(\varsigma)})$ in vcf_p, P finds out the ω-th row of $\mathcal{C}_{(\xi+1)\times 9}$ which satisfies $loc_p^{(\varsigma)} \in [\mathcal{C}_{(\omega-1),0}+1, \mathcal{C}_{\omega,0}]$. After that, $op_p^{(\varsigma)}$ and $val_p^{(\varsigma)}$ are replaced by r and the corresponding values in the ω-th row. Next, for each processed edit $([\![loc_p^{(\varsigma)}]\!], [\![op_p^{(\varsigma)}]\!], [\![val_p^{(\varsigma)}]\!])$, secret keys k_1, k_2, k_3 (sk) and $[\![loc_p^{(\varsigma)}]\!]$, $[\![op_p^{(\varsigma)}]\!]$, $[\![val_p^{(\varsigma)}]\!]$ (ch) are input in $F(sk, ch)$ in turn to encrypt vcf_p. The encrypted vcf_p can be denoted as

$$E(vcf_p) = \{(E_{k_1}(loc_p^{(\varsigma)}), E_{k_2}(op_p^{(\varsigma)}), E_{k_3}(val_p^{(\varsigma)}))|\varsigma \in [1, l_p]\}.$$

Finally, $\langle E(vcf_p), \mathbb{T}\rangle$ is sent to CS for requesting service, $\mathbb{T}$ is a tolerance limit selected by P.

5.4 Privacy-preserving similar patients query

In this phase, CS searches the encrypted GBK-tree $E(\mathcal{T})$ and returns the encrypted identifiers of similar patients to Ps according to their search requests.

After receiving $\langle E(vcf_p), \mathbb{T}\rangle$ from P, CS first initializes an empty identify set ID and a node set NS= $\{(E(vcf_0), id_0^E)\}$. Then, from root node to the most bottom leaf nodes, CS executes the following steps to finish the retrieval of $E(\mathcal{T})$.

- *Step 1:* Check whether the node(s) in NS store(s) similar genome(s) of G_p. For each node $(E(vcf_{k'}), id_{k'}^E)$ in NS, CS computes the approximate edit distance from G_p to $G_{k'}$, i.e., $Aed(G_p, G_{k'})$. If $Aed(G_p, G_{k'}) \leq \mathbb{T}$, add $id_{k'}^E$ into ID; otherwise no action is required.
- *Step 2:* Determine the retrieval range and update NS. For each node $(E(vcf_{k'}), id_{k'}^E)$ in NS, its child nodes in the next layer whose edge-weights belong to $[Aed(G_p, G_{k'}) - \mathbb{T}, Aed(G_p, G_{k'}) + \mathbb{T}]$ will form a new NS and replace the original.
- *Step 3:* Repeat *Step 1* and *2* until NS is not updated.

The final ID will contain the encrypted identifiers of the patients whose genomes' approximate edit distance from G_p is not larger than $\mathbb{T}$. However, since the query and GBK-tree sent to CS have been encrypted, the operations $\{del, sub, ins, ins+r, \ldots\}$ are indistinguishable during the above steps. To solve this problem, an improved secure genomic edit distance approximate computation algorithm (SEDA) based on IEDA is presented in **Algorithm 3**, which can calculate the approximate edit distance between two encrypted single-character edits sets $E(vcf_p)$ and $E(vcf_{k'}), k' \in [0, n-1]$. Compared with IEDA, SEDA is more suitable for the proposed privacy-preserving scheme.

Algorithm 3: Improved secure genomic edit distance approximate computation (SEDA)

Input: Two encrypted sets $E(vcf_p)$ and $E(vcf_{k'})$.
Output: Genomic approximate edit distance between G_p and $G_{k'}$, $Aed(G_p, G_{k'})$.

compute $len_p = |E(vcf_p)|, len_{k'} = |E(vcf_{k'})|$;
denote $E(vcf_p)$ as array $U[len_p][3]$, $E(vcf_{k'})$ as array $K[len_{k'}][3]$;
initialize two arrays $U'[\,][3], K'[\,][3]$;
initialize $len = 0$, $i = 0$, $j = 0$;
while $i \leq len_u - 1$ && $j \leq len_k - 1$ **do**
  if $U[i][0] == K[j][0]$ **then**
    for $0 \leq k \leq 2$ **do**
      $U'[len][k] = U[i][k]$;
      $K'[len][k] = K[j][k]$;
    $len+=1; i+=1; j+=1$;
  else if $U[i][0] < K[j][0]$ **then** $i+=1$;
  else $j+=1$;
return two arrays $U'[len][3]$ and $K'[len][3]$;
initialize $Aed = len_u + len_k - 2 \cdot len$;
for $0 \leq i \leq len - 1$ **do**
  if $U'[len][1]! = K'[len][1]$ && $U'[len][2]! = K'[len][2]$ **then**
    $Aed+=2$;
  else if $U'[len][1]! = K'[len][1]$ && $U'[len][2] == K'[len][2]$ **then**
    $Aed+=1$;
  else if $U'[len][1] == K'[len][1]$ && $U'[len][2]! = K'[len][2]$ **then**
    $Aed+=1$;
  else $Aed+=0$;
return Aed.

At last, CS returns the final ID to P, P can decrypt and obtain the identifiers of all similar patients via k_4.

5.5 Encrypted genetic BK-tree updating

In our system model, the genomic database of MI is bound to be expanded by the data generated by daily medical activities, thus we present a method to update the encrypted GBK-tree. Assume a new genome (G_e, id_e) is added in GD, $(E(vcf_e), id_e^E)$ will be added to $E(\mathcal{T})$ as follows.

- *Step 1* (@ MI): Based on *Ref*, convert G_e into vcf_e, and calculate $Aed(Ref, G_e) = |vcf_e|$. If $Aed(Ref, G_e)$ is not equal to any edge-weight of vcf_0's child nodes, it will be added as a new child node of vcf_0 with an edge-weight of $|vcf_e|$. Otherwise, the node vcf_{k^e} (one child node of vcf_0) with an edge-weight of $|vcf_e|$ should be located and $Aed(G_e, G_{k^e})$ will be calculated through algorithm IEDA, whether vcf_{k^e} is the parent node of vcf_e is determined by judging whether $Aed(G_e, G_{k^e})$ is not equal to any edge-weight of vcf_{k^e}'s child node. The above operation will be repeated until the appropriate

parent node vcf_{pn} appears, and the insertion position of vcf_e will be recorded as $(vcf_{pn}, Aed(G_e, G_{pn}), vcf_e)$.

- *Step 2* (@ MI): By following the encryption method proposed in Section 5.2, $E(vcf_e)$ and id_e^E can be generated. After that, the insertion position will be updated to $(id_{pn}^E, Aed(G_e, G_{pn}), E(vcf_e), id_e^E)$ and sent to CS for asking a new node insertion. If multiple nodes need to be added at the same time, send their insertion positions together.
- *Step 3* (@ CS): After receiving a node insertion request from MI, CS adds $(E(vcf_e), id_e^E)$ to $E(\mathcal{T})$ as a child node of the node whose identifier is id_{pn}^E, with an edge-weight of $Aed(G_e, G_{pn})$.

6 Security Analysis

In CASPER, we use an SKE algorithm (e.g. AES) to encrypt the identifiers of patients, which cannot be attacked successfully without secret key [35]. In addition, before being encrypted, the original genomes are converted into sets of triples (i.e., single-character edits) based on a public reference genome *Ref* at first, which means unless owning *Ref*, no one can restore the original genomes even if they have obtained all plaintext converted genomes. However, some genomic mutations in specific locations may reveal the privacy of patients. Thus, we mainly focus on the privacy of triples sets (i.e., vcf_k, $k \in [1, n-1]$ and vcf_q).

Specifically, we first define a leakage function $\mathcal{L}$ [36], [37], and then introduce the real and ideal environments in respective for the subsequent formal security definition and analysis.

6.1 Leakage function

CASPER is essentially equivalent to a symmetric-key searchable encryption scheme, as a necessary part of the security definition of searchable encryption, the leakage function $\mathcal{L}$ can capture all the information leaked during the evaluation on encrypted data in CASPER. Informally, considering the encrypted database $\{\varnothing, E(vcf_1), \ldots, E(vcf_{n-1})\}$ and the encrypted queries $E(vcf_p)$, $1 \le p \le \vartheta$, the $\mathcal{L}$ of CASPER can be summarized as three aspects (all of them are considered to be *default* information leakage):

- ***Size Pattern:*** The cloud sever learns the total number of genomes in the genomic database (n) and the query requests submitted by physicians (ϑ). Besides, the size of each converted genome (l_k and l_p) and the data structure of the encrypted genomic database ($E(\mathcal{T})$) also can be achieved.
- ***Access Pattern:*** The cloud server reveals the encrypted identifiers of similar patients that are returned for the SPQ query requests.
- ***Search Pattern:*** The cloud server can learn whether a same encrypted genome (i.e., encrypted GBK-tree node) is queried by two different queries.

6.2 Real and ideal environment

The security of CASPER will be proven in the real and ideal environments, which are defined with $\mathcal{L}$ as follows.

Real environment. The real environment involves a stateful probabilistic polynomial time (PPT) adversary $\mathcal{A}$ and a challenger $\mathcal{C}$, and the two participants interact as follows.

- **Initialization phase:** $\mathcal{A}$ generates a database $\mathrm{D}_{\mathcal{A}}$ consists of $\{\varnothing, vcf_1, \ldots, vcf_{n-1}\}$ in random, the GBK-tree $\mathcal{T}_{\mathcal{A}}$ constructed for $\mathrm{D}_{\mathcal{A}}$ owns the same structure as $E(\mathcal{T})$ and is sent to $\mathcal{C}$.
- **Setup phase:** Given a security parameter λ, $\mathcal{C}$ follows the operations in phase *system initialization* to create KP$= \langle k_1, k_2, k_3, r, \mathcal{C}_{(\xi+1)\times 9}\rangle$ and keep it private. Then, according the steps in phase *encrypted genetic BK-tree outsourcing*, $\mathcal{C}$ encrypts $\mathcal{T}_{\mathcal{A}}$ into $E(\mathcal{T}_{\mathcal{A}}) = \{E(vcf_{k'})\}_{k'=0}^{n-1}$, where $E(vcf_0) = \varnothing$, $E(vcf_k) = \{(E_{k_1}(loc_k^{(\eta)}), E_{k_2}(op_k^{(\eta)}), E_{k_3}(val_k^{(\eta)}))|\eta \in [1, l_k]\}$.
- **Query phase 1:** $\mathcal{A}$ adaptively chooses a number of queries $\{vcf_p\}_{p=1}^{p_1}$ and sends them to $\mathcal{C}$. In response, $\mathcal{C}$ returns $\{E(vcf_p)\}_{p=1}^{p_1}$ to $\mathcal{A}$ by executing the operations in phase *search request generation*.
- **Challenge phase:** $\mathcal{C}$ returns the encrypted GBK-tree $E(\mathcal{T}_{\mathcal{A}})$ to $\mathcal{A}$.
- **Query phase 2:** In the phase, $\mathcal{A}$ can also adaptively choose a number of queries $\{vcf_p\}_{p=p_1+1}^{p_2}$ and submit them to $\mathcal{C}$. Same as **Query phase 1**, $\{E(vcf_p)\}_{p=p_1+1}^{p_2}$ will be returned to $\mathcal{A}$.

Let $r_{\mathcal{A}}$ denote the internal random bits used by $\mathcal{A}$ in the real environment. And $\mathrm{View}_{\mathcal{A}}^{\mathrm{Real}}$ is used to denote the ensemble $(\mathrm{D}_{\mathcal{A}}, E(\mathcal{T}_{\mathcal{A}}), \{E(vcf_p)\}_{p=1}^{p_2}, r_{\mathcal{A}})$, which is essentially the view of $\mathcal{A}$ in the above-described real environment.

Ideal environment. The ideal environment involves a stateful PPT adversary $\mathcal{A}$ and a simulator $\mathcal{S}$, and the two participants interact as follows.

- **Initialization phase:** $\mathcal{A}$ generates a database $\mathrm{D}_{\mathcal{A}}$ consists of $\{\varnothing, vcf_1, \ldots, vcf_{n_1}\}$ in random, the GBK-tree $\mathcal{T}_{\mathcal{A}}$ constructed for $\mathrm{D}_{\mathcal{A}}$ owns the same structure as $E(\mathcal{T})$ and is sent to $\mathcal{S}$.
- **Setup phase:** $\mathcal{S}$ first create a matrix $\mathcal{C}'_{(\xi+1)\times 9}$ and a number r' in random, both of them have the basic properties mentioned in phase *system initialization*. After processing each vcf_k in $\mathcal{T}_{\mathcal{A}}$ based on $\mathcal{C}'_{(\xi+1)\times 9}$ and r', for the substitutions of locations, operations and characters, $\mathcal{S}$ randomly assigns fixed-length values respectively, the assignment satisfies the condition that the same plaintext corresponding to the same ciphertext. The final results form an encrypted GBK-tree $E'(\mathcal{T}_{\mathcal{A}})$.
- **Query phase 1:** $\mathcal{A}$ adaptively chooses a number of queries $\{vcf_p\}_{p=1}^{p_1}$ and sends them to $\mathcal{C}$. In response, $\mathcal{C}$ returns $\{E'(vcf_p)\}_{p=1}^{p_1}$ to $\mathcal{A}$ via the same process and assignment as vcf_k.
- **Challenge phase:** $\mathcal{C}$ returns the encrypted GBK-tree $E'(\mathcal{T}_{\mathcal{A}})$ to $\mathcal{A}$.
- **Query phase 2:** In the phase, $\mathcal{A}$ can also adaptively choose a number of queries $\{vcf_p\}_{p=p_1+1}^{p_2}$ and submit them to $\mathcal{C}$. Same as **Query phase 1**, $\{E'(vcf_p)\}_{p=p_1+1}^{p_2}$ will be returned to $\mathcal{A}$.

Again, let $r_{\mathcal{A}}$ denote the internal random bits used by $\mathcal{A}$ in the ideal environment. And $\mathrm{View}_{\mathcal{A},\mathcal{S}}^{\mathrm{Ideal}}$ is used to denote the ensemble $(\mathrm{D}_{\mathcal{A}}, E'(\mathcal{T}_{\mathcal{A}}), \{E'(vcf_p)\}_{p=1}^{p_2}, r_{\mathcal{A}})$, which is essentially the view of $\mathcal{A}$ in the above-described ideal environment.

6.3 Formal security definition and analysis

Based on the views of $\mathcal{A}$ in the real and ideal environments, we define and proof the security of CASPER.

Definition 2. *CASPER is said to be selectively secure under the chosen-plaintext attack model with leakage function $\mathcal{L}$ **iff** for any PPT adversary $\mathcal{A}$, who issues an encrypted GBK-tree and a polynomial number of encrypted queries, there exists an efficient simulator $\mathcal{S}$ such the advantage of $\mathcal{A}$ in distinguishing the views of real and ideal environments is negligible, i.e., the function $Adv_{\mathcal{A}}^{CASPER}(\lambda) = |Pr[View_{\mathcal{A}}^{Real} = 1] - Pr[View_{\mathcal{A},\mathcal{S}}^{Ideal} = 1]|$ is a negligible function in the security parameter λ.*

Theorem 1. *CASPER is selectively secure under chosen-plaintext attack model with $\mathcal{L}$.*

Proof. The security of CASPER can be proved if it can be proved that $\mathcal{A}$ has no ability to distinguish the views observed from the real and ideal environments. In the ideal environment, $\mathcal{S}$ generates $E'(\mathcal{T}_{\mathcal{A}})$ and $\{E'(vcf_p)\}_{p=1}^{p_2}$ by randomly choosing parameters and ciphertexts, thus the indistinguishability of $\text{View}_{\mathcal{A}}^{\text{Real}}$ and $\text{View}_{\mathcal{A},\mathcal{S}}^{\text{Ideal}}$ is equal to the indistinguishability of $\text{View}_{\mathcal{A}}^{\text{Real}}$ and random number. Meanwhile, the encryption method of vcf_k in $E(\mathcal{T}_{\mathcal{A}})$ and vcf_p are the same, both of them are encrypted through encrypting loc, op and val respectively. Therefore, we will proof the security from the following cases.

Case 1. Encrypted locations $E_{k_1}(loc_k^{(\eta)})$, $\eta \in [1, l_k], k \in [1, n)$ and $E_{k_1}(loc_p^{(\varsigma)})$, $\varsigma \in [1, l_p], p \in [1, p_2]$ are indistinguishable from random ciphertexts.

In CASPER, the length of *Ref* is m, thus the values of locations belong to $\{0, 1, \ldots, m\}$, we directly use HMAC to encrypt the locations with secret key k_1, HMAC is considered as a pseudo-random function, thus the encrypted locations must be indistinguishable from random number.

Case 2. Encrypted operations $E_{k_2}(op_k^{(\eta)})$, $\eta \in [1, l_k], k \in [1, n)$ and $E_{k_2}(op_p^{(\varsigma)})$, $\varsigma \in [1, l_p], p \in [1, p_2]$ are indistinguishable from random ciphertexts.

The operations belong to $\{sub, del, ins, ins + r, \ldots\}$, for different locations, $\{sub, del, ins\}$ are first replaced by different values from $\{0, 1, 2\}$ according to $\mathcal{C}_{(\xi+1)\times 9}$, then $\{ins+r, ins+r+1, \ldots\}$ are replaced by $\mathcal{C}_{(\xi+1)\times 9}$ and a randomly selected r. Finally, the processed values of operations are encrypted by HMAC with secret key k_2. Considering $\mathcal{C}_{(\xi+1)\times 9}$ and r is randomly generated, HMAC is pseudo-random, the encrypted operations must be indistinguishable from random number.

Case 3. Encrypted characters $E_{k_2}(val_k^{(\eta)})$, $\eta \in [1, l_k], k \in [1, n)$ and $E_{k_2}(val_p^{(\varsigma)})$, $\varsigma \in [1, l_p], p \in [1, p_2]$ are indistinguishable from random ciphertexts.

The encryption of characters $\in \{A, G, C, T, \perp\}$ is same as operations, we skip the details here due to space limitations. □

Most of the existing privacy-preserving SPQ schemes may adopt other encryption technologies to guarantee the security of original data, such as, hash function and symmetric-key matric encryption [7], secure multi-party computation (e.g., oblivious transfer) [8], [9], AES [11] as well as the customized bloom filter [13]. Same as our proposed scheme, directly applying these typical encryption technologies to privacy-preserving SPQ services are essentially equivalent to constructing searchable encryption schemes. Therefore, as shown in TABLE 1, relying on the security of these encryption primitives themselves, in the searchable encryption scenario, all the mentioned schemes can achieve the same security level, namely IND-SCPA.

7 Performance Evaluation

In this section, we mainly analyze and evaluate the performance of our CASPER in terms of computation cost and communication overhead. Besides, we also make a comparison with EFSS (i.e., EFSS_I, EFSS_II and EFSS_II(ES)) [7]. Same as CASPER, the distance metric used in EFSS also refers to the genomic approximate edit distance proposed in [9]. And among the state-of-the-art work similar to ours [7], [9], [10], [38], EFSS has been proven to maintain a significant advantage in their work. The comparison mainly focuses on three stages (i.e., *Encrypted genetic BK-tree outsourcing*, *Search request generation* and *Privacy-preserving similar patients query*), which constitute the core parts of SPQ services, and correspond to the phases *Index generation*, *Trapdoor generation* and *Query* in EFSS, respectively.

TABLE 4
Average accuracy of CASPER compared with plain domain

Chromosome number	CASPER			
	$\mathbb{T}=0$	$\mathbb{T}=1$	$\mathbb{T}=2$	$\mathbb{T}=3$
Chromosome 1 (540)	100%	98.70%	98.09%	97.02%
Chromosome 2 (519)	100%	98.04%	96.54%	95.45%
Chromosome 3 (397)	100%	98.99%	97.98%	97.73%
Chromosome 6 (368)	100%	98.46%	97.19%	96.01%
Chromosome 11 (389)	100%	97.45%	95.25%	94.13%
Chromosome 12 (351)	100%	98.88%	97.64%	96.60%
Chromosome 15 (315)	100%	96.93%	95.37%	97.48%
Chromosome 17 (383)	100%	97.39%	95.43%	93.49%
Chromosome number	Operation over plaintext			
	$\mathbb{T}=0$	$\mathbb{T}=1$	$\mathbb{T}=2$	$\mathbb{T}=3$
Chromosome 1 (540)	100%	98.70%	98.09%	97.02%
Chromosome 2 (519)	100%	98.04%	96.54%	95.45%
Chromosome 3 (397)	100%	98.99%	97.98%	97.73%
Chromosome 6 (368)	100%	98.46%	97.19%	96.01%
Chromosome 11 (389)	100%	97.45%	95.25%	94.13%
Chromosome 12 (351)	100%	98.88%	97.64%	96.60%
Chromosome 15 (315)	100%	96.93%	95.37%	97.48%
Chromosome 17 (383)	100%	97.39%	95.43%	93.49%

7.1 Evaluation environment

In order to measure the integrated performance of CASPER in the real environment, we implement CASPER with Java programming language on three computers (2.50 GHz four-core processor, 24GB RAM, windows 10 system), which are connected through LAN, to simulate MI, CS and P respectively. Then, SHA256 and AES128 are chosen as the algorithms of HMAC and SKE. Moreover, all performance evaluations in this section are based on a real-world dataset and a randomly generated synthetic dataset. The details of the two datasets are described as follows.

- Real-world dataset (CURATED). As a human genomic database curated by experts, CURATED consists of 28371 genomic data and contains 15068 kinds of gene-diseases caused by single nucleotide polymorphism.

Specifically, its genomic data is recorded as sets of edits [39], and the length of these sets ranges from 1 to 2258, but most of them are concentrated in $[10, 30]$.

- Synthetic dataset. In order to test the factors how to affect the performance of CASPER, we randomly generate a synthetic dataset, which consists of 10000 genomes. After genomes conversion, the length of single-character edits set ranges from 10 to 30.

7.2 Accuracy evaluation

To demonstrate the accuracy of CASPER, we select the genomes associated with different diseases from CURATED on eight distinct chromosomes. Specifically, after the duplicate genomic data on the same chromosome is deleted, there are a total of 3262 genomes, and the detailed information is shown in TABLE 4. For each chromosome, we first construct a GBK-tree based on all genomic data stored in it, and then use each data as a query request to find similar genomes. TABLE 4 indicates that the accuracy of finding genomes with the same gene-disease in plaintext and CASPER is the same, which means that the protective measures taken by CASPER do not compromise the accuracy.

TABLE. 4 also shows that the choice of tolerance limit $\mathbb{T}$ can influence the accuracy of CASPER. When $\mathbb{T}$ is set to 0, the physicians can find the same genome as their patients with the probability of 100%, but as the value of $\mathbb{T}$ increases, the accuracy of CASPER will decrease ($\geq 90\%$). However, considering the polymorphism of single nucleotide, to find the optimal therapeutic schedule, the value of $\mathbb{T}$ should be set large to dig out all possible treatments for reference.

7.3 Computation cost

In this section, we analyze and evaluate the computation cost of CASPER, then make a comparison with EFSS in theoretical analysis and experimental results, respectively.

• Theoretical analysis

To facilitate the description of the various types of operations involved in CASPER, we denote the replace operation, HMAC operation and comparison operation as $\mathcal{R}$, $\mathcal{H}_M$ and C_o, respectively. And $\mathcal{D}$ and $\mathcal{D}_s$ are used to represent approximate edit distance computation operation and secure computation operation, respectively. For EFSS, we use $\mathcal{H}$, C_a and C_m to represent the computation cost of a hash operation, an addition operation and a multiplication operation, respectively. Besides, $\mathcal{P}$ is used to represent an inner product operation of a matrix and a vector and $\mathcal{E}$ is a pseudo-random permutation. For the sake of simplicity, we also assume the number of genomic data is n, and the number of elements in $vcf_{k',0<k'<n}$ and vcf_p are all set to l.

Specifically, in stage *Encrypted genetic BK-tree outsourcing*, for each vcf_k stored in the $\mathcal{T}$'s node (except for the root node), MI first replaces all elements in vcf_k with random value, and then encrypts the random values by HMAC. Assume that the construction of $\mathcal{T}$ needs to compute approximate edit distance ρ times and compare ϱ times, this stage total requires $(\varrho \cdot C_o + \rho \cdot \mathcal{D}) + (n-1) \times 3l \times (\mathcal{R} + \mathcal{H}_M)$ operations. In stage *Search request generation*, P needs to process vcf_p as MI does vcf_k, this costs $3l \times (\mathcal{R} + \mathcal{H}_M)$ operations in total. Finally, CS obtains the identifies of similar patients through searching $E(\mathcal{T})$ in stage *Privacy-preserving similar patients query*, assume that ρ' approximate edit distance secure computation operations and comparison operations are needed to search the encrypted GBK-tree, a total of $\rho' \times (\mathcal{D}_s + C_o)$ are cost for finding the similar genomes.

For EFSS, in the phase of *Index generation*, EFSS_I first converts each edit in $vcf_{k,0\leq k\leq n-1}$ into a hash value, and then calculates the sum of all values in $\mathfrak{n}$ different segments. Due to the above-mentioned steps need to be repeated κ times, this phase total requires $n \times \kappa \times [l \cdot \mathcal{H} + (l-1) \cdot C_a]$ operations. Different from EFSS_I, EFSS_II also applies two $(2\mathfrak{n} + \mu + \nu + 2) \times (2\mathfrak{n} + \mu + \nu + 2)$ dimensional invertible matrices, which adds $(\mathfrak{n}-1) \cdot C_m + (2\mathfrak{n}+\mu+\nu+2) \cdot \mathcal{E} + 2\mathcal{P}$ operations to change positions and encrypt indexes. Moreover, EFSS_II(ES) divides n genomes into $\mathcal{K}$ classes and encrypts these $\mathcal{K}$ cluster centers, thus it needs extra $\mathcal{K} \times \kappa \times [l \cdot \mathcal{H} + (l-1) \cdot C_a + (\mathfrak{n}-1) \cdot C_m + (2\mathfrak{n}+\mu+\nu+2) \cdot \mathcal{E} + 2\mathcal{P}]$ operations. In the phase of *Trapdoor generation*, given a query request, similar to the encryption of vcf_k, EFSS_I requires $\kappa \times [l \cdot \mathcal{H} + (l-1) \cdot C_a]$ operations, both EFSS_II and EFSS_II(ES) require $\kappa \times [l \cdot \mathcal{H} + (l-1) \cdot C_a + (\mathfrak{n}-1) \cdot C_m + (2\mathfrak{n}+\mu+\nu+2) \cdot \mathcal{E} + 2\mathcal{P}]$ operations. Finally, for executing the similarity query in the phase of *Query*, EFSS_I and EFSS_II require $n \times (\kappa \cdot \mathcal{P} + \kappa \cdot C_a + C_m)$ operations. However, for EFSS_II(ES), it first finds the nearest cluster centers to narrow down the query range, thus, only $\frac{2n}{\mathcal{K}} \times (\kappa \cdot \mathcal{P} + \kappa \cdot C_a + C_m)$ operations in average are needed. As shown in TABLE 5, the computation cost of CASPER is smaller than that of EFSS in terms of theoretical analysis.

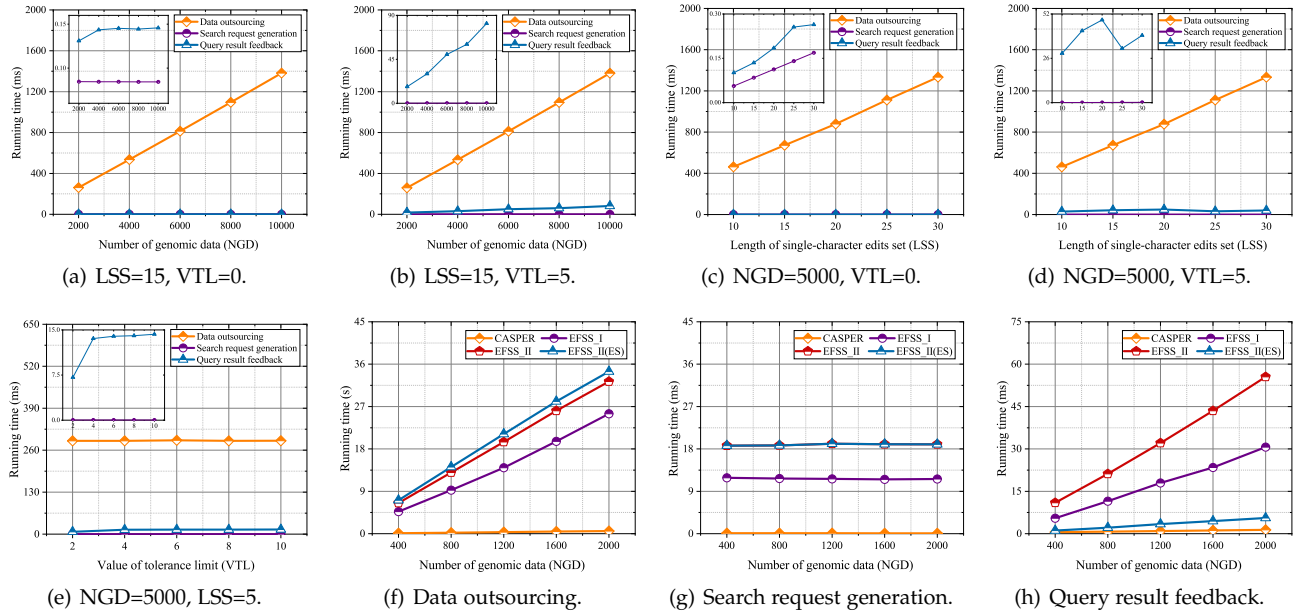
• Experimental results

To evaluate all factors which affect the computation cost of CASPER, we first use the synthetic dataset to conduct experiments, then make a comparison with EFSS to show the performance in practice. According to the above analysis, we can infer that the main factors influencing the running time of CASPER are the Number of Genomic Data (NGD), the Length of Single-character edits Set (LSS) and the Value of Tolerance Limit (VTL). In Figs. 8(a)-(e), we plot the average running time of all CASPER's stages varying NGD, LSS and VTL, respectively. Given VTL=0 and VTL=5, Figs. 8(a)-(b) are drawn with the NGD from 2000 to 10000 (LSS=15) and Figs. 8(c)-(d) are drawn with the LSS from 10 to 30 (NGD=5000). We can see that the running time of outsourcing data and feedbacking query results increases with the increase of NGD due to the growth of genomic data that needs to be calculated. And when LSS increases, the number of single-character edits also increases, each stage of CASPER inevitably spends more time processing genomic data. In Fig. 8(e), we plot the average running time of CASPER varying VTL from 2 to 10, where NGD=500 and LSS=5. Compared with NGD and LSS, VTL only affects the running time of search request generation, the larger LSS is, the higher the running time of search request generation is. In Figs. 8(f)-(h), we implement EFSS in the same evaluation environment introduced in Section 7.1, and set the parameters as $\kappa = 10$, $\mathfrak{n} = 100$, $\mu = 20$, $\nu = 10$ and $\mathcal{K} = 100$. Figs. 8(f)-(h) (VTL=5) show the average running time of CASPER and EFSS at different stages, respectively. We can see that the running time of CASPER and EFSS both increase with the increase of NGD, and in the stages data outsourcing, search request generation and query result

TABLE 5
Computation cost of EFSS [7] vs our CASPER

	Data outsourcing[1]	Search request generation[2]	Query result feedback[3]
EFSS_I	$n \times \kappa \times [l \cdot \mathcal{H} + (l-1) \cdot C_a]$	$\kappa \times [l \cdot \mathcal{H} + (l-1) \cdot C_a]$	$n \times (\kappa \cdot \mathcal{P} + \kappa \cdot C_a + C_m)$
EFSS_II	$n \times \kappa \times [l \cdot \mathcal{H} + (l-1) \cdot C_a + (\mathfrak{n}-1) \cdot C_m + (2\mathfrak{n}+\mu+\nu+2) \cdot \mathcal{E} + 2\mathcal{P}]$	$\kappa \times [l \cdot \mathcal{H} + (l-1) \cdot C_a + (\mathfrak{n}-1) \cdot C_m + (2\mathfrak{n}+\mu+\nu+2) \cdot \mathcal{E} + 2\mathcal{P}]$	$n \times (\kappa \cdot \mathcal{P} + \kappa \cdot C_a + C_m)$
EFSS_II(ES)	$(n+\mathcal{K}) \times \kappa \times [l \cdot \mathcal{H} + (l-1) \cdot C_a + (\mathfrak{n}-1) \cdot C_m + (2\mathfrak{n}+\mu+\nu+2) \cdot \mathcal{E} + 2\mathcal{P}]$	$\kappa \times [l \cdot \mathcal{H} + (l-1) \cdot C_a + (\mathfrak{n}-1) \cdot C_m + (2\mathfrak{n}+\mu+\nu+2) \cdot \mathcal{E} + 2\mathcal{P}]$	$\frac{2n}{\mathcal{K}} \times (\kappa \cdot \mathcal{P} + \kappa \cdot C_a + C_m)$
CASPER	$(\varrho \cdot C_o + \rho \cdot \mathcal{D}) + (n-1) \times 3l \times (\mathcal{R} + \mathcal{H}_M)$	$3l \times (\mathcal{R} + \mathcal{H}_M)$	$\rho' \times (\mathcal{D}_s + C_o)$

[1] Corresponding to the stage of *Encrypted genetic BK-tree outsourcing* and the phase of *Index generation*.
[2] Corresponding to the stage of *Search request generation* and the phase of *Trapdoor generation*.
[3] Corresponding to the stage of *Privacy-preserving similar patients query* and the phase of *Query*.



(a) LSS=15, VTL=0. (b) LSS=15, VTL=5. (c) NGD=5000, VTL=0. (d) NGD=5000, VTL=5.

(e) NGD=5000, LSS=5. (f) Data outsourcing. (g) Search request generation. (h) Query result feedback.

Fig. 8. Average computation cost of CASPER vs EFSS (test 500 times). (a)(b)(c)(d)(e) are the running time of CASPER varying NGD, LSS and VTL, respectively. (f)(g)(h) are comparisons of CASPER and EFSS at different stages varying NGD.

feedback, the maximum time of CASPER costed are 554ms, 0.1ms and 1.4ms, while that of the most efficient EFSS (i.e., EFSS_I) costed are 25467ms, 11.6ms and 30.7ms. Obviously, no matter at which stage, CASPER performs better.

7.4 Communication overhead

In this section, we analyze and evaluate the communication overhead of CASPER, and make a comparison with EFSS in theoretical analysis and experimental results, respectively.

• Theoretical analysis

In CASPER, during the stage of *Encrypted genetic BK-tree outsourcing*, MI sends their encrypted GBK-tree $E(\mathcal{T})$ to CS, $E(\mathcal{T})$ is composed of root node $(\varnothing, id_0)$ and the other $(n-1)$ nodes $(E(vcf_k), id_k^E)$. Assume that the average bit length of $\{E(vcf_k)\}_{k=1}^{n-1}$ and $\{id_0^E, id_1^E, \ldots, id_{n-1}^E\}$ are $l \times 3\mathfrak{h}_\mathcal{C}$ bits and $\mathfrak{a}$ bits respectively, the total bit length of $E(\mathcal{T})$ is $(n \times \mathfrak{a}) + (n-1) \times (l \times 3\mathfrak{h}_\mathcal{C})$ bits. During the stage of *Search request generation*, P needs to send the query request which made up with an encrypted genome and a tolerance limit $\mathbb{T}$ to CS. Under the assumptions mentioned above, the query request spends $l \times 3\mathfrak{h}_\mathcal{C} + \lceil \log_2 \mathbb{T} \rceil$ bits in total in length. Finally, after finishing the retrieval of $E(\mathcal{T})$, a set of encrypted identifies ID is returned to P in the stage of *Privacy-preserving similar patients query*. ID contains all encrypted identifies of similar patients whose genomes' approximate edit distance from G_p is not larger than $\mathbb{T}$. Assume that there are τ encrypted identifies in ID, a total of $\tau \times \mathfrak{a}$ bits in length are spent to feedback query results.

For EFSS, given a genomic database GD with n genomes GD$= \{G_1, G_2, \ldots, G_n\}$, and a query genome G_p. In the phase of *Index generation*, EFSS_I fist cuts each genome into $\mathfrak{n}$ segments, then sends encrypted indexes of all genome to CS. Assume that the bit length of each encrypted segment is $\mathfrak{h}_\mathcal{E}$ bits, the total bit length of the encrypted index for n genomes is $n \times \kappa \times (\mathfrak{n} \times \mathfrak{h}_\mathcal{E})$ bits, where κ is the number of cycle times. Since two $(2\mathfrak{n}+\mu+\nu+2) \times (2\mathfrak{n}+\mu+\nu+2)$ dimensional invertible matrices are introduced in EFSS_II, it will cost $2n \times \kappa \times (2\mathfrak{n}+\mu+\nu+2) \times \mathfrak{h}_\mathcal{E}$ bits in length in this phase. Compared with EFSS_II, EFSS_II(ES) also needs to generate encrypted indexes for K cluster center, thus $2\kappa \times (n+K) \times (2\mathfrak{n}+\mu+\nu+2) \times \mathfrak{h}_\mathcal{E}$ bits are spent in total by EFSS_II(ES). In the phase of *Trapdoor generation*,

an encrypted query result (i.e., trapdoor) should be sent to CS. Due to the method of encrypting G_p is the same as encrypting each genome in GD, EFSS_I spends $\kappa \times (\mathfrak{n} \times \mathfrak{h}_{\mathcal{E}})$ bits in length to generate trapdoor, the cost of both EFSS_II and EFSS_II(ES) are $2\kappa \times (2\mathfrak{n}+\mu+\nu+2) \times \mathfrak{h}_{\mathcal{E}}$ bits. Finally, CS is required to return the top $\mathcal{K}$ results in the phase of *Query*, thus the communication overhead of EFSS_I, EFSS_II and EFSS_II(ES) are all $\mathcal{K} \times \mathfrak{h}_{\mathcal{E}}$ bits. As shown in TABLE 6, the communication overhead of CASPER is smaller than that of EFSS in terms of theoretical analysis.

• **Experimental results**

We also conduct experiments on the synthetic dataset to demonstrate the communication overhead of CASPER and EFSS. Similar to computation cost, the main factors impacting the communication overhead of CASPER are NGD, LSS and VTL, too. Moreover, we performance the evaluation of EFSS under the same environment as CASPER, and the parameters of EFSS are set to $\kappa = 10$, $\mathfrak{n} = 100$, $\mu = 20$, $\nu = 10$ and $\mathcal{K} = 100$. Figs. 9(a)-(b) and Figs. 9(c)-(d) depict the communication overhead of CASPER in three stages varying NGD from 2000 to 10000 (LSS=15, VTL=0 and 5) and LSS from 10 to 30 (NGD=5000, VTL=0 and 5), respectively. To demonstrate the impact of VTL on communication overhead of CASPER, Fig. 9(e) is drawn with VTL from 2 to 10, NGD=5000 and LSS=5. From all the above-mentioned figures, we can observe that the communication overhead of outsourcing data increases with the increase of NGD and LSS because more and larger genomic data needs to be transmitted. The communication overhead of generating a search request is only affected by LSS due to the process of search request only related to the number of single-character edits. Besides, the communication overhead of feedbacking query results increases with the increase of NGD and VTL, because the query results consist of encrypted labels (128bits), the data needs to be transmitted to P has no connection with LSS. In Figs. 9(f)-(h), comparisons with EFSS in three stages are given varying NGD from 400 to 2000 (VTL=5). We can observe that our proposed scheme spends more communication overhead than EFSS in phase Query result feedback, but the overall communication overhead of CASPER is still much lower than EFSS. Specifically, the overall communication overhead costed by CASPER is 2.45MB, while that costed by the most efficient EFSS (i.e., EFSS_I) is 15.48MB.

8 Related work

In this section, we briefly review some related work on privacy-preserving (approximate) edit distance computation over genomic data. For this research problem, existing solutions can be divided into two parts, outsourced computing and federated computing.

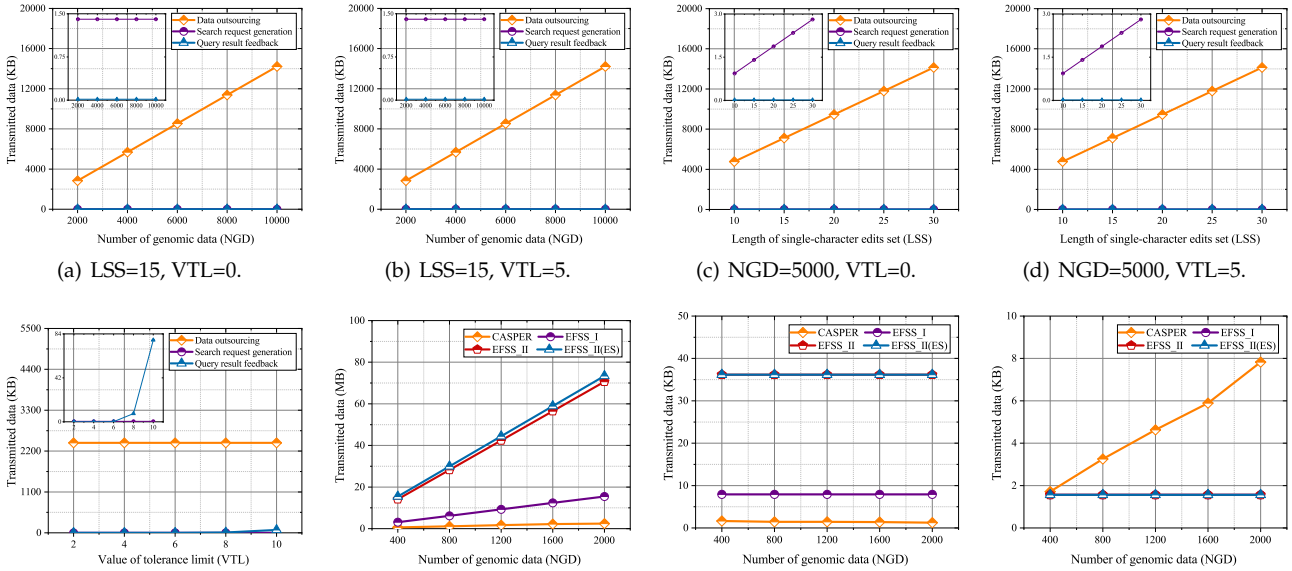
Outsourced computing based schemes. In the model of outsourced computing, encrypted databases should be outsourced to a cloud server and the main computation tasks are directly performed on the encrypted data by one or more cloud server(s). Specifically, Atallah et al. [40] proposed a basic protocol that can securely outsource the edit distance computation between DNA sequences to two servers. Later, considering the unsatisfactory performance, Atallah et al. [41] again presented a non-interactive privacy-preserving scheme to achieve more efficient query service than [40]. Besides, under the model of two non-colluding semi-honest third parties, Cheng et al. [26] and Schneider et al. [8] also proposed two secure similar sequence query scheme over outsourced genomic databases based on approximate edit distance. However, since the above schemes have multiple participated servers, all of them still face inevitable huge communication and computation overhead. To solve this problem, the single cloud model is increasingly favored by researchers. Wang et al. [42] designed a genomes partition protection framework which only keeps a very small portion of human genomes secret according to the different security levels. But the latest research report says, the concept of which genomes are sensitive or insensitive is still a matter of debate. For the safety of the whole genome, a novel genomic edit distance approximation algorithm is proposed in [9], and combined with a private set difference size protocols, a genome-wide, privacy-preserving similar patients query system was constructed. To improve the query efficiency, Mahdi et al. [30] presented a prefix tree-based indexing algorithm for supporting secure sequence similarity search on encrypted genomic data, but it replaced edit distance with the hamming distance which is not so suitable for genomes. Based on another new (approximate) edit distance algorithm, Asharov et al. [10] also put forward an approach for solving SPQ problem. Regrettably, both [10] and [9] proposed their scheme only from the perspective of efficiency. Taking data access control and personalized search requirements into account, an efficient and fine-grained similarity search scheme [7] was presented to enable retrieving resemble DNA sequences over encrypted cloud data. Once the model of outsourced computing is adopted, there must be communication overhead for sending encrypted databases to the cloud server(s).

Federated computing based schemes. In the model of federated computing, private data does not be outsourced to a semi-honest party, each party can only access its own data and needs to cooperate with other parties to complete a specific computation task. Jah et al. [43] contributed to the original work in the area of privacy-preserving edit distance computation on genomic data. They presented tree secure protocols on the basis of oblivious transfer, but due to the performance of oblivious transfer, their scheme is not suitable for large-scale computing tasks. Shimizu et al [44] combined the Burrows-Wheeler transform with additive homomorphic encryption to find target queries on a genomic database, but as we all know, homomorphic encryption technology is still time-consuming so far. Moreover, Al Aziz et al. [45] proposed two different approximation methods to securely compute edit distance based on several encrypted technologies, such as private set intersection, garbled circuits and so on. Beyond genomic data, Salem et al. [12] also considered the sensitivity of epigenomic and transcriptomic data, two schemes are proposed to process these data with the highest security and privacy guarantees. However, the federated computing model requires all participants to remain online until the computation is completed.

Considering the nature of genomic data, local computing will bring a heavy load to the terminals. In recent years, researchers still pay more attention to the model of outsourced computing [11], [38], [46], [47]. And a few of

TABLE 6
Communication overhead of EFSS [7] vs our CASPER (bits)

	Data outsourcing	Search request generation	Query result feedback
EFSS_I	$n \times \kappa \times (\mathfrak{n} \times \mathfrak{h}_{\mathcal{E}})$	$\kappa \times (\mathfrak{n} \times \mathfrak{h}_{\mathcal{E}})$	$\mathcal{K} \times \mathfrak{h}_{\mathcal{E}}$
EFSS_II	$2n \times \kappa \times (2\mathfrak{n} + \mu + \nu + 2) \times \mathfrak{h}_{\mathcal{E}}$	$2\kappa \times (2\mathfrak{n} + \mu + \nu + 2) \times \mathfrak{h}_{\mathcal{E}}$	$\mathcal{K} \times \mathfrak{h}_{\mathcal{E}}$
EFSS_II(ES)	$2\kappa \times (n + K) \times (2\mathfrak{n} + \mu + \nu + 2) \times \mathfrak{h}_{\mathcal{E}}$	$2\kappa \times (2\mathfrak{n} + \mu + \nu + 2) \times \mathfrak{h}_{\mathcal{E}}$	$\mathcal{K} \times \mathfrak{h}_{\mathcal{E}}$
CASPER	$(n \times \mathfrak{a}) + (n-1) \times (l \times 3\mathfrak{h}_{\mathcal{C}})$	$l \times 3\mathfrak{h}_{\mathcal{C}} + \lceil \log_2 \mathbb{T} \rceil$	$\tau \times \mathfrak{a}$



(a) LSS=15, VTL=0. (b) LSS=15, VTL=5. (c) NGD=5000, VTL=0. (d) NGD=5000, VTL=5.

(e) NGD=5000, LSS=5. (f) Data outsourcing. (g) Search request generation. (h) Query result feedback.

Fig. 9. Average communication overhead of CASPER vs EFSS (test 500 times). (a)(b)(c)(d)(e) are the size of data transmitted by CASPER varying NGD, LSS and VTL, respectively. (f)(g)(h) are comparisons of CASPER and EFSS at different stages varying NGD.

existing schemes [11], [13], [30] combines the query process with an efficient data structure based on hamming or cosine distance. But to the best of our knowledge, compared with hamming and cosine distance, edit distance is a more widely used metric to quantify the similarity of two genomes. And BK-tree is specially designed for edit distance, it can play a better role in SPQ when edit distance is chosen as the metric.

9 Conclusion

In this paper, we first designed a new method to construct and search a genetic BK-tree (GBK-tree) for a genomic database. Then, we proposed an efficient and privacy-preserving similar patients query scheme over GBK-trees, named CASPER. By taking our scheme, the medical institution can construct an encrypted GBK-tree to securely outsource its genomic database to a cloud server, and the cloud server could provide similar patients query services for registered physicians in a privacy-preserving way. Detailed security analysis showed its security strength and privacy-preserving ability, and extensive experiments were conducted to demonstrate its efficiency. In the future work, we will take the improvement of security into consideration.

Acknowledgments

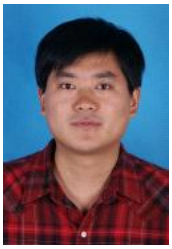
This work was supported by National Natural Science Foundation of China (61972304 and 61932015), National Natural Science Foundation of Shaanxi Province (2019ZDLGY12-02), Technical Research Program of Public Security Ministry (2019JSYJA01).

References

[1] W. KA, "Dna sequencing costs: Data from the nhgri genome sequencing program (gsp)," www.genome.gov/sequencingcostsdata, accessed October 30, 2019.
[2] Y. Huang, B. Gulko, and A. Siepel, "Fast, scalable prediction of deleterious noncoding variants from functional and population genomic data," *Nature genetics*, vol. 49, no. 4, pp. 618–624, 2017.
[3] M. Nassar, Q. Malluhi, M. Atallah, and A. Shikfa, "Securing aggregate queries for dna databases," *IEEE Transactions on Cloud Computing*, vol. 7, no. 3, pp. 827–837, 2017.
[4] B. Langmead and A. Nellore, "Cloud computing for genomic data analysis and collaboration," *Nature Reviews Genetics*, vol. 19, no. 4, pp. 208–219, 2018.
[5] H. Wang, B. J. Lengerich, B. Aragam, and E. P. Xing, "Precision lasso: accounting for correlations and linear dependencies in high-dimensional genomic data," *Bioinformatics*, vol. 35, no. 7, pp. 1181–1187, 2019.
[6] R. Chen, L. Yang, S. Goodison, and Y. Sun, "Deep-learning approach to identifying cancer subtypes using high-dimensional genomic data," *Bioinformatics*, vol. 36, no. 5, pp. 1476–1483, 2020.
[7] G. Xu, H. Li, H. Ren, X. Lin, and X. S. Shen, "Dna similarity search with access control over encrypted cloud data," *IEEE Transactions on Cloud Computing*, 2020.

[8] T. Schneider and O. Tkachenko, "Episode: Efficient privacy-preserving similar sequence queries on outsourced genomic databases," in *ACM Asia Conference on Computer and Communications Security (AisaCCS)*, 2019, pp. 315–327.
[9] 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 *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2015, pp. 492–503.
[10] G. Asharov, S. Halevi, Y. Lindell, and T. Rabin, "Privacy-preserving search of similar patients in genomic data," *Proceedings on Privacy Enhancing Technologies*, vol. 2018, no. 4, pp. 104–124, 2018.
[11] M. S. R. Mahdi, M. M. Al Aziz, D. Alhadidi, and N. Mohammed, "Secure similar patients query on encrypted genomic data," *IEEE journal of biomedical and health informatics*, vol. 23, no. 6, pp. 2611–2618, 2018.
[12] A. Salem, P. Berrang, M. Humbert, and M. Backes, "Privacy-preserving similar patient queries for combined biomedical data," *Proceedings on Privacy Enhancing Technologies*, vol. 2019, no. 1, pp. 47–67, 2019.
[13] X. Zhu, E. Ayday, R. Vitenberg, and N. R. Veeraragavan, "Privacy-preserving search for a similar genomic makeup in the cloud," *IEEE Transactions on Dependable and Secure Computing*, 2021.
[14] G. Navarro, "A guided tour to approximate string matching," *ACM computing surveys (CSUR)*, vol. 33, no. 1, pp. 31–88, 2001.
[15] J. C. Venter, M. D. Adams, E. W. Myers, P. W. Li, R. J. Mural, G. G. Sutton, H. O. Smith, M. Yandell, C. A. Evans, R. A. Holt *et al.*, "The sequence of the human genome," *science*, vol. 291, no. 5507, pp. 1304–1351, 2001.
[16] M. Humbert, E. Ayday, J.-P. Hubaux, and A. Telenti, "Addressing the concerns of the lacks family: quantification of kin genomic privacy," in *ACM SIGSAC conference on Computer & Communications Security (CCS)*, 2013, pp. 1141–1152.
[17] S. S. Shringarpure and C. D. Bustamante, "Privacy risks from genomic data-sharing beacons," *The American Journal of Human Genetics*, vol. 97, no. 5, pp. 631–646, 2015.
[18] M. Akgün, A. O. Bayrak, B. Ozer, and M. Ş. Sağıroğlu, "Privacy preserving processing of genomic data: A survey," *Journal of biomedical informatics*, vol. 56, pp. 103–111, 2015.
[19] H. Shen and J. Ma, "Privacy challenges of genomic big data," in *Healthcare and Big Data Management*. Springer, 2017, pp. 139–148.
[20] A. B. Carter, "Considerations for genomic data privacy and security when working in the cloud," *The Journal of Molecular Diagnostics*, vol. 21, no. 4, pp. 542–552, 2019.
[21] M. M. A. Aziz, M. N. Sadat, D. Alhadidi, S. Wang, X. Jiang, C. L. Brown, and N. Mohammed, "Privacy-preserving techniques of genomic data—a survey," *Briefings in bioinformatics*, vol. 20, no. 3, pp. 887–895, 2019.
[22] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, "(leveled) fully homomorphic encryption without bootstrapping," *ACM Transactions on Computation Theory (TOCT)*, vol. 6, no. 3, pp. 1–36, 2014.
[23] Z. Brakerski, "Fully homomorphic encryption without modulus switching from classical gapsvp," in *Annual International Cryptology Conference (CRYPTO)*. Springer, 2012, pp. 868–886.
[24] J. H. Cheon, M. Kim, and K. Lauter, "Homomorphic computation of edit distance," in *International Conference on Financial Cryptography and Data Security (FC)*. Springer, 2015, pp. 194–212.
[25] M. Kim and K. Lauter, "Private genome analysis through homomorphic encryption," in *BMC medical informatics and decision making*, vol. 15, no. S5. Springer, 2015, pp. 1–12.
[26] K. Cheng, Y. Hou, and L. Wang, "Secure similar sequence query on outsourced genomic data," in *ACM Asia Conference on Computer and Communications Security (AisaCCS)*, 2018, pp. 237–251.
[27] J. Chang and R. Lu, "Achieving privacy-preserving edit distance query in cloud and its application to genomic data," in *International Conference on Privacy, Security and Trust (PST)*. IEEE, 2019, pp. 1–9.
[28] Y. Zheng, R. Lu, J. Shao, Y. Zhang, and H. Zhu, "Efficient and privacy-preserving edit distance query over encrypted genomic data," in *International Conference on Wireless Communications and Signal Processing (WCSP)*. IEEE, 2019, pp. 1–6.
[29] A. Rheinländer, M. Knobloch, N. Hochmuth, and U. Leser, "Prefix tree indexing for similarity search and similarity joins on genomic data," in *International Conference on Scientific and Statistical Database Management (SSDBM)*. Springer, 2010, pp. 519–536.
[30] M. S. R. Mahdi, M. Z. Hasan, and N. Mohammed, "Secure sequence similarity search on encrypted genomic data," in *International Conference on Connected Health: Applications, Systems and Engineering Technologies (CHASE)*. IEEE, 2017, pp. 205–213.
[31] O. Goldreich, *Foundations of cryptography: volume 2, basic applications*. Cambridge university press, 2009.
[32] W. A. Burkhard and R. M. Keller, "Some approaches to best-match file searching," *Communications of the ACM*, vol. 16, no. 4, pp. 230–236, 1973.
[33] H. Delfs and H. Knebl, "Symmetric-key encryption," in *Introduction to Cryptography*. Springer, 2007, pp. 11–31.
[34] W. J. Masek and M. S. Paterson, "A faster algorithm computing string edit distances," *Journal of Computer and System sciences*, vol. 20, no. 1, pp. 18–31, 1980.
[35] P. Hamalainen, T. Alho, M. Hannikainen, and T. D. Hamalainen, "Design and implementation of low-area and low-power aes encryption hardware core," in *EUROMICRO conference on digital system design (DSD)*. IEEE, 2006, pp. 577–583.
[36] S. Kamara, C. Papamanthou, and T. Roeder, "Dynamic searchable symmetric encryption," in *Proceedings of the 2012 ACM conference on Computer and communications security*, 2012, pp. 965–976.
[37] D. Cash, S. Jarecki, C. Jutla, H. Krawczyk, M.-C. Roşu, and M. Steiner, "Highly-scalable searchable symmetric encryption with support for boolean queries," in *Annual cryptology conference*. Springer, 2013, pp. 353–373.
[38] B. Wang, W. Song, W. Lou, and Y. T. Hou, "Privacy-preserving pattern matching over encrypted genetic data in cloud computing," in *International Conference on Computer Communications (INFOCOM)*. IEEE, 2017, pp. 1–9.
[39] P. Janet, J. M. Ramírez-Anguita, J. Saüch-Pitarch, F. Ronzano, E. Centeno, F. Sanz, and L. I. Furlong, "The disgenet knowledge platform for disease genomics: 2020 update," https://www.disgenet.org/home/, accessed June, 2020.
[40] M. J. Atallah and J. Li, "Secure outsourcing of sequence comparisons," *International Journal of Information Security*, vol. 4, no. 4, pp. 277–287, 2005.
[41] M. Blanton, M. J. Atallah, K. B. Frikken, and Q. Malluhi, "Secure and efficient outsourcing of sequence comparisons," in *European Symposium on Research in Computer Security (ESORICS)*. Springer, 2012, pp. 505–522.
[42] R. Wang, X. Wang, Z. Li, H. Tang, M. K. Reiter, and Z. Dong, "Privacy-preserving genomic computation through program specialization," in *ACM Conference on Computer and Communications Security (CCS)*, 2009, pp. 338–347.
[43] S. Jha, L. Kruger, and V. Shmatikov, "Towards practical privacy for genomic computation," in *Symposium on Security and Privacy (S & P)*. IEEE, 2008, pp. 216–230.
[44] K. Shimizu, K. Nuida, and G. Rätsch, "Efficient privacy-preserving string search and an application in genomics," *Bioinformatics*, vol. 32, no. 11, pp. 1652–1661, 2016.
[45] M. M. Al Aziz, D. Alhadidi, and N. Mohammed, "Secure approximation of edit distance on genomic data," *BMC medical genomics*, vol. 10, no. 2, pp. 55–67, 2017.
[46] T. Schneider and O. Tkachenko, "Towards efficient privacy-preserving similar sequence queries on outsourced genomic databases," in *ACM Workshop on Privacy in the Electronic Society (WPES)*, 2018, pp. 71–75.
[47] J. Wei, Y. Lin, X. Yao, J. Zhang, and X. Liu, "Differential privacy-based genetic matching in personalized medicine," *IEEE Transactions on Emerging Topics in Computing*, pp. 1–16, 2020.

Dan Zhu currently is a Ph.D candidate in Xidian University. She received the B.S. degree with the School of Telecommunications Engineering from Xidian University, xi'an, China, in 2017. Her research interests include applied cryptography, data security and privacy.



Hui Zhu (M'13-SM'19) received the B.S. and Ph.D. degrees from Xidian University, Xi'an, China, in 2003 and 2009, respectively, and the M.S. degree from Wuhan University, Wuhan, China, in 2005. In 2013, he was with School of Electrical and Electronics Engineering, Nanyang Technological University as a Research Fellow. Since 2016, he has been the professor in the School of Cyber Engineering, Xidian University, China. His research interests include the areas of applied cryptography, data security and privacy.

Xiangyu Wang currently is a Ph.D candidate in Xidian University. He received the B.S. degree with the School of Cyber Engineering from Xidian University, xi'an, China, in 2017. His research interests include data security and secure computation outsourcing.

Rongxing Lu (S'09-M'11-SM'15-F'21) is an associate professor at the Faculty of Computer Science (FCS), University of New Brunswick (UNB), Canada. Before that, he worked as an assistant professor at the School of Electrical and Electronic Engineering, Nanyang Technological University (NTU), Singapore from April 2013 to August 2016. Rongxing Lu worked as a Postdoctoral Fellow at the University of Waterloo from May 2012 to April 2013. He was awarded the most prestigious "Governor General's Gold Medal", when he received his PhD degree from the Department of Electrical & Computer Engineering, University of Waterloo, Canada, in 2012; and won the 8th IEEE Communications Society (ComSoc) Asia Pacific (AP) Outstanding Young Researcher Award, in 2013. Also, Dr. Lu received his first PhD degree at Shanghai Jiao Tong University, China, in 2006. Dr. Lu is an IEEE Fellow. His research interests include applied cryptography, privacy enhancing technologies, and IoT-Big Data security and privacy. He has published extensively in his areas of expertise (with H-index 74 from Google Scholar as of April 2021), and was the recipient of 9 best (student) paper awards from some reputable journals and conferences. Currently, Dr. Lu serves as the Vice-Chair (Conferences) of IEEE ComSoc CIS-TC (Communications and Information Security Technical Committee). Dr. Lu is the Winner of 2016-17 Excellence in Teaching Award, FCS, UNB.

Dengguo Feng received the B.S. degree from Shaanxi Normal University, Xi'an, China, in 1988, the M.S. and Ph.D. degrees from Xidian University, Xi'an, China, in 1993 and 1995, respectively. He is currently a Professor with the Institute of Software, Chinese Academy of Sciences, Beijing, China. He is a recipient of China National Funds for Distinguished Young Scientists. He is the Vice-Chairmen of Chinese Association for Cryptologic Research and a Steering Committee Member of Information Technology in National High-Tech R&D Program of China.