Privacy-preserving Multiple Sequence Alignment Scheme for Long Gene Sequence

Yatong Jiang
School of Cyber Science and Technology, Beihang University
Beijing, China
jiangyatong@buaa.edu.cn

Tao Shang*
School of Cyber Science and Technology, Beihang University
Bejing, China
shangtao@buaa.edu.cn

Jianwei Liu
School of Cyber Science and Technology, Beihang University
Bejing, China
liujianwei@buaa.edu.cn

Abstract

Gene Multiple Sequence Alignment is crucial for genomic data analysis, forming the basis for studying its biological significance. The digitization of genomic data allows collaborative analysis on cloud platforms, improving the efficiency and precision of genomic research. However, gene sequences contain sensitive information, posing a risk of privacy leakage with unauthorized access. Balancing privacy, accuracy, and efficiency in multiple sequence alignment for long gene sequences remains a challenge. In this paper, we propose a distributed privacy-preserving multiple sequence alignment scheme for long sequences based on secure multi-party computation. Our scheme includes a method for segmenting long sequences to achieve partially distributed computing and a privacy-preserving method for calculating edit distance among subsequences using secret sharing. The scheme consists of a distributed computing phase and an aggregate computing phase, optimizing efficiency by dropping repeated subsequences alignment. Our proposed scheme achieves accurate and efficient privacy-preserving alignment for long gene sequences.

Keywords

Multiple Sequence Alignment, Secure Multi-Party Computation, Cloud Platform, Privacy-preserving, Distributed computing

1 Introduction

With the advancements in gene sequencing technology and the adoption of digital storage for gene sequences, the utilization of genomic data has become widespread across various domains, including scientific research, disease treatment, legal forensics, and drug research and development [27]. Consequently, there has been a notable escalation in the volume of genomic data, driven by the enthusiastic pursuit of genomic information by researchers. To improve the effectiveness of gene sequence analysis, collaborative analysis of gene sequences is commonly performed among different institutions. In particular, multiple sequence alignment (MSA) of gene sequences from different data sets holds a significant role in comparative genomic research, which is a fundamental tool in the domain of computational biology [50]. MSA enables researchers to identify commonalities and differences between gene sequences, allowing for the prediction of molecular structure and function, as well as the construction of phylogenetic trees. By performing MSA between diverse gene data sets, researchers gain valuable insights into the evolutionary relationships and functional associations among different organisms or individuals [6]. Gene sequences encompass a substantial amount of sensitive personal information, particularly the extended gene sequence data, which can uniquely identify individuals through the information on mutation sites within the long gene sequence. This disclosure of personal information may include details concerning heredity, diseases, and kinship, thereby posing a risk to the privacy of individuals. Furthermore, the interrelated nature of gene sequences means that the disclosure of personal information may also result in the exposure of sensitive information pertaining to relatives or ethnic groups, leading to severe consequences. Importantly, gene sequences are inherently stable, meaning that if sensitive information within the gene sequence data is leaked, the associated risks persist throughout a personal entire lifespan.

Simultaneously, the progress of technology and digitalization has engendered more accessible methods for individuals to acquire gene sequencing information. The internet platform provides a convenient avenue for obtaining gene sequences, which, in turn, contain a substantial amount of privacy information pertaining to the data providers. Researchers have proposed various techniques to get sensitive information from gene sequences, including individual identification [16, 25, 42, 52], linkage attack [12, 24, 30, 32], genotype inference [2, 8, 11, 31], and Bayesian inference [38, 45, 49]. These attacks on gene sequences can extract private information about data providers from publicly available genomic data and subsequently trace them back to their actual identities. In 2013, Gymrek et al. [11] showed that by analyzing short tandem repeats on the Y chromosome and querying the pedigree database, their last names can be recovered from the personal gene. In 2017, Lippert et al. [26] proposed a method of phenotypic prediction. This prediction can match phenotypic data with individual-level genotype data obtained from Whole Gene Sequencing (WGS). In 2018, Deznabi et al. [8] used the observable Markov model and the haplotype recombination model to perform inference attacks on genomic data shared on public platforms. In 2019, Zhang et al. [49] proposed a scheme that utilizes GWAS statistics to infer private information about individuals, employing a Bayesian network construction method and addressing inference problems related to trait and genotype inference. In 2020, Her et al. [14] proposed an inference attack algorithm based on belief propagation in factor graphs to predict target genotypes in GWAS.

As we know, genomic data is not only related to personal privacy information but also related to the interdependent privacy [4] of

Proceedings on Privacy Enhancing Technologies 2025(1), 236–249

https://doi.org/10.56553/popets-2025-0014

relatives, countries, and even ethnic groups. Numerous researchers have employed cryptographic techniques, such as homomorphic encryption [9, 13, 18, 41], garbled circuit [20, 43], blockchain [22, 23, 47], and secure multiparty computation [1, 36, 39, 44], to address the privacy concerns associated with genomic data privacy. Privacy-preserving gene sequence alignment based on SMPC emerges as a crucial research direction among various privacy protection approaches for genomic data. In 2015, Wang et al. [44] proposed a highly efficient and precise edit distance protocol based secure SPQ(SPQ: Similar Patient Query) primitive, that enables privacy-preserving search in large-scale distributed genomic databases. In 2018, Asharov et al. [1] proposed an efficient and secure computation approach for a SPQ problem, combining an approximation method with optimized two-party protocols. Shen et al. [40] proposed an efficient and privacy-preserving set intersection(PSI) protocol for human genes, ensuring safe paternity and ancestry testing. In 2021, Schneider et al. [39] proposed EPISODE, an efficient privacy-preserving protocol by homomorphic encryption(HE) that enables the identification of genetically similar individuals in an outsourced genomic database. In 2022, Nakagawa et al. [34] proposed a technique that utilizes FM-index for secure gene substring matches in databases and improving efficiency compared to non-indexed approaches.

As shown in Tab. 1, researchers have done a lot of outstanding work on gene sequence privacy-preserving alignment and considered diverse scenarios such as similar patient queries, large-scale database queries, and outsourced database queries. However, those schemes typically involve approximate calculations for pairwise alignment or substring alignment and the security aspect of MSA has been overlooked. In particular, the single gene sequences involved in MSA are frequently longer than 1,0000 base pairs, containing more individual information than in normal pairwise alignment [33]. The increased length and multiple numbers of gene sequences make ensuring data privacy while performing MSA more difficult. Given the significant research implications of MSA and the practical importance of collaborative analysis, this paper proposes a privacy-preserving multiple sequence alignment scheme designed specifically for long gene sequences to address this challenge.

The main contributions of this paper are summarized as follows.

- We design the first privacy-preserving multiple sequence alignment scheme for long gene sequence. Our scheme realized efficient and accurate MSA for over 10000 bps length gene sequences and achieve data privacy query privacy and output privacy.
- We provide a segmentation method for long gene sequence. This segmentation method can achieve efficient accurate MSA by combination of distributed computing and aggregate computing accurate to reduce redundant calculations.
- We design a privacy-preserving method for MSA. This method achieve privacy-preserving MSA for long gene sequence based on the segmentation method and improves the efficiency of alignment by optimizing the selection of different secret sharing methods of ABY.

This paper is structured as follows. In Section II, we introduce the related concepts, including gene sequence alignment and secure multi-party Computation. Section III focuses on the overview of the privacy-preserving long unequal-length gene multiple sequence alignment , Section IV descibes segmenting method for distributed computing and Section V descibes privacy-preserving MSA method for aggregate computing based on ABY. Section VI focuses on scheme analysis and Section VII includes the result of our experiment and discussions. Section VIII is our conclusion.

2 Preliminaries

We introduce some preliminaries for a better understanding of our scheme, including the gene sequence alignment and ABY framework.

2.1 Gene Sequence Alignment

DNA sequence consists of four bases: A (Adenine, A), T (Thymine, T), C (Cytosine, C), and G (Guanine, G), and is the carrier of biological genetic characteristics. Gene sequence alignment involves comparing the DNA base composition of two sequences to calculate their differences, aiming to identify highly similar gene subsequences, known as homologous sequences, among different gene sequences.

2.1.1 Gene Pairwise Sequence Alignment. Pairwise sequence alignment refers to the alignment of two sequences to find their similarity relationship. When pairwise sequence alignment is required, edit distance and scoring rules are used to determine the similarity of two gene sequences. When there are two different DNA chains, through some operations, the two DNA chains can become the same. The cost in this process is called edit distance, also called Levenshtein distance [15]. These operations on bases are divided into three types:

Ins: Insert one or more bases in the sequence.
Del: Delete one or more bases from the sequence.
Rep: Replace one base in the sequence with another base.

It is usually stipulated that the cost of each operation is 1, then the scoring rules are determined, so the edit distance between two gene sequences is the result of gene sequence alignment. Assuming that there are two gene sequences $\alpha = (\alpha[1], \alpha[2], ..., \alpha[m])$ and $\beta = (\beta[1], \beta[2], ...\beta[n])$, we will calculate the edit distance between the two sequences, i.e., the minimum cost required to change the gene sequence α to β, which is set as D_{ij}. According to the three operations defined before, the following operations can be performed.

Ins or **Del** $\alpha[i]$: $D_{ij} = D_{(i-1)j} + 1$.
Ins or **Del** $\beta[j]$: $D_{ij} = D_{i(j-1)} + 1$.
Rep $\alpha[i]$ or $\beta[j]$: $D_{ij} = D_{(i-1)(j-1)} + t, t \in \{0, 1\}$.
If $\alpha[i] = \beta[j]$, then $t = 1$, otherwise $t = 0$.
So the formula for calculating edit distance is

$$D_{ij} = min[D_{(i-1)j} + 1, D_{i(j-1)} + 1, D_{(i-1)(j-1)} + t] \quad (1)$$

2.1.2 Gene Multiple Sequence Alignment. Multiple sequence alignment is the generalization of pairwise sequence alignment. MSA aligns two or more gene sequences, comparing the similarities and differences of their bases column by column, and finds the operations method that makes each gene sequence as consistent as possible, so as to find their common structural features. MSA is mainly used for molecular evolution relationships, predicting the secondary structure and tertiary structure of proteins, estimating

Table 1: Related Researchs on Privacy-preserving Gene Sequence Alignment

Reference	Function	Methods	MSA	Years
[44]	Precise gene edit distance alignment for secure SPQ	GC	×	2015
[1]	Efficient approximate edit distance alignment method for secure SPQ	Secure two-party protocol	×	2018
[39]	Secure gene sequence alignment for outsourced genomic database	HE	×	2021
[34]	Secure gene substring matches in large databases	Secret Sharing	×	2022
Our	Privacy-preserving MSA for long gene sequences	ABY	✓	–

the total number of protein folding types, gene sequence analysis, etc.

Compared with pairwise sequence alignment, MSA can not only discover the connection between the two species but also build the connections between multiple species, which helps a lot in researching the origin and evolution of creatures. Besides, MSA can improve the accuracy of the alignment by eliminating the accidental effect of individual variation. MSA algorithm can be divided into progressive approach, iterative approach, centre star approach, heuristics, machine learning, and divide-and-conquer [51]. An MSA approach conducts three steps:

Step 1 Finding a sequence to be the basic sequence.
Step 2 Conducting the pairwise sequence alignment between other sequences and the basic sequence and constructing the distance matrix.
Step 3 According to the distance matrix from Step 2, construct the guide tree or find the center sequence to conduct multiple alignments.

2.2 ABY Framework

ABY is a SMPC mixed-protocol framework, it supports three different types of sharing (arithmetic sharing, Boolean sharing, and Yao sharing) and provides efficient conversions between them[7], as shown in Fig. 1.

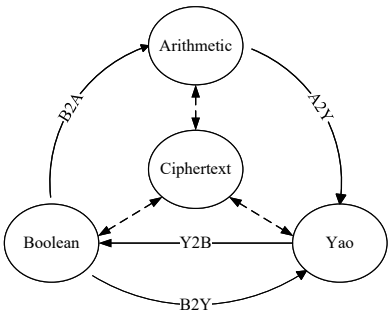

Figure 1: ABY Framework

2.2.1 Arithmetic Sharing. Arithmetic sharing protocol based on Beaver's multiplication triples [3]. For a value x with the length of l bits , it can be decomposed into the sum of two values by additive secret sharing on Z_{2^l}, which represents integers modulo 2^l.

Seg By using additive secret sharing, the secret value $\langle x\rangle^A$ is split into two values in the space Z_{2^l}. $\langle x\rangle_0^A + \langle x\rangle_1^A \equiv x mod 2^l$, which $\langle x\rangle_0^A, \langle x\rangle_1^A \in Z_{2^l}$.

Shr $Shr_i^A(x)$: Participant P_i chooses a random number $r \in_R Z_{2^l}$, set $\langle x\rangle_i^A = x - r$ and sent r to participant P_{1-i}, who sets $\langle x\rangle_{1-i}^A = r$.

Rec $Rec_i^A(x)$: P_{1-i} sends its shares $\langle x\rangle_{1-i}^A = r$ to P_i for reconstructing $x = \langle x\rangle_1^A + \langle x\rangle_2^A$.

Add $\langle z\rangle^A = \langle x\rangle^A + \langle y\rangle^A$: The addition can be performed directly locally.P_i calculates $\langle z\rangle_i^A = \langle x\rangle_i^A + \langle y\rangle_i^A \ mod 2^l$

Mul $\langle z\rangle^A = \langle x\rangle^A \cdot \langle y\rangle^A$: Multiplication requires the additive shaing of Beaver's multiplication triples $c = ab$ and needs online calulation. Firstly, P_i calculates $\langle e\rangle_i^A = \langle x\rangle_i^A - \langle a\rangle_i^A$ and $\langle f\rangle_i^A = \langle y\rangle_i^A - \langle b\rangle_i^A$. Secondly, $\langle e\rangle_i^A$ & $\langle f\rangle_i^A$ exchange online between P_i. Performing $Rec^A(e)$, $Rec^A(f)$ to get e , f. Finally, P_i calculates $\langle z\rangle_i^A = i \cdot e \cdot f + f \cdot \langle a\rangle_i^A + e \cdot \langle b\rangle_i^A + \langle c\rangle_i^A$.

2.2.2 Boolean Sharing. Boolean sharing uses bitwise *XOR* operations to share variables, and evaluates functions expressed as Boolean circuits based on Goldreich-MicaliWigderson (GMW) protocol [10].

Seg A bit $\langle x\rangle^B$ is shared based Boolean secret sharing between two participants, such as $\langle x\rangle_0^B + \langle x\rangle_1^B = x$, which $\langle x\rangle_0^B, \langle x\rangle_1^B \in Z_2$.

Shr $Shr_i^B(x)$: Participant P_i chooses a random number $r \in_R \{0,1\}$, set $\langle x\rangle_i^B = x \oplus r$ and sent r to participant P_{1-i}, who sets $\langle x\rangle_{1-i}^B = r$.

Rec $Rec_i^B(x)$: P_{1-i} sends its shares $\langle x\rangle_{1-i}^B = r$ to P_i for reconstructing $x = \langle x\rangle_0^B \oplus \langle x\rangle_1^B$.

Xor $\langle z\rangle^B = \langle x\rangle^B \oplus \langle y\rangle^B$: P_i locally calculates $\langle z\rangle_i^B = \langle x\rangle_i^B \oplus \langle y\rangle_i^B$

Add $\langle z\rangle^B = \langle x\rangle^B \wedge \langle y\rangle^B$: Addition requires a pre-computed Boolean multiplication triples $\langle c\rangle^B = \langle a\rangle^B \wedge \langle b\rangle^B$ and needs online calulation. Firstly, P_i calculates $\langle e\rangle_i^B = \langle x\rangle_i^B \oplus \langle a\rangle_i^B$ and $\langle f\rangle_i^B = \langle y\rangle_i^B \oplus \langle b\rangle_i^B$. Secondly, $\langle e\rangle_i^B$&$\langle f\rangle_i^B$ exchange online between P_i. Performing $Rec^B(e)$, $Rec^B(f)$ to get e , f. Finally, P_i calculates $\langle z\rangle_i^B = i \cdot e \cdot f \oplus f \cdot \langle a\rangle_i^B \oplus e \cdot \langle b\rangle_i^B \oplus \langle c\rangle_i^B$.

Mul Boolean Sharing can calculate the *Mul* gate, which is $Mul^B(x,y,b)$. If $b = 0$, $Mul^B(x,y,b) = x$, else $Mul^B(x,y,b) = y$.

2.2.3 Yao Sharing. Yao sharing is based on Yao's garbled circuit protocol for secure two-party computation [46], the garbler party encrypts a Boolean function to a garbled circuit, which is evaluated by the evaluator party. The garbler P_0 randomly chooses a global k-bit string R with $R[0] = 1$. For each wire ω, the wire keys consist of $k_w^0 \in_R \{0,1\}^k$ and $k_w^1 = k_w^0 \oplus R$.

Seg A bit $\langle x\rangle^Y$ is shared based Yao sharing between two participants as $\langle x\rangle_0^Y = k_0$, and $\langle x\rangle_1^Y = k_x = k_0 \oplus xR$.

Shr $Shr_0^Y(x)$: Participant P_0 samples $\langle x\rangle_0^Y = k_0 \in_R \{0,1\}^k$, and send $\langle x\rangle_1^Y = k_x = k_0 \oplus xR$ to participant P_1. $Shr_1^Y(x)$: both participants perform $C - OT_k^1$ where P_0 acts as a sender, inputs the correlation function $f_R(x) = (x \oplus R)$ and obtains $(k^0, k^1 = k^0 \oplus R)$ with $k^0 \in_R \{0,1\}^k$ and P_1 acts as receiver with choice bit x and obliviously obtains $\langle x\rangle_1^Y = k_x$.

Rec $Rec_i^Y(x)$: P_{1-i} sends its permutation bit $\pi = \langle x\rangle_{1-i}^Y[0]$ to P_i for reconstructing $x = \pi \oplus \langle x\rangle_i^Y[0]$.

Xor $\langle z\rangle^Y = \langle x\rangle^Y \oplus \langle y\rangle^Y$: By using the free-XOR [21] technique for evaluating, P_i locally calculates $\langle z\rangle_i^Y = \langle x\rangle_i^Y \oplus \langle y\rangle_i^Y$

And $\langle z\rangle^Y = \langle x\rangle^Y \wedge \langle y\rangle^Y$: P_0 creates a garbled table using $Gb_{\langle z\rangle_0^Y}(\langle x\rangle_0^Y, \langle y\rangle_0^Y)$, where Gb is a garbling function. P_0 sends the garbled table to P_1, who decrypts it using the keys $\langle x\rangle_1^Y$ and $\langle y\rangle_1^Y$ Secondly, $\langle e\rangle_i^B \& \langle f\rangle_i^B$ exchange online between P_i. Performing $Rec^B(e)$, $Rec^B(f)$ to get e , f. Finally, P_i calculates $\langle z\rangle_i^B = i \cdot e \cdot f \oplus f \cdot \langle a\rangle_i^B \oplus e \cdot \langle b\rangle_i^B \oplus \langle c\rangle_i^B$.

3 Privacy-preserving MSA Scheme

Multiple gene data sets want to share data on a cloud platform server for multiple sequence alignment joint analysis. To protect the data privacy of the gene data sets, we propose a scheme on this scene to keep data ciphertext during online computing. In this section, we will describe the framework of this scheme, clarify the meaning of the terminologies and notations, and clarify the operations flow of our scheme.

3.1 Scheme Framework Description

Multiple gene data sets want to share data on a cloud platform server for multiple sequence alignment joint analysis. The data must be ciphertext during aggregate computing to achieve privacy-preserving MSA. For privacy-preserving calculation, the data is stored in a cloud server and calculated by two independent nodes A and B through secret sharing, assuming no collusion. Nodes A and B are honest but curious participants, which means they will follow the protocol's flow to complete the task, do not launch active attacks, but remain curious about each other's input data. Our scheme can achieve accurate multiple sequence alignment for long gene sequences on cloud servers. The scheme will not disclose the private information of both gene data sets in the process. Based on ABY secret sharing methods, the scheme realizes the distributed privacy-preserving MSA between the gene data sets. The system model is shown in Fig. 2. Different gene data sets are allocated for distributed computing as child nodes, and A B on the cloud server are tasked with aggregate computing as master nodes.

3.2 Terminology and Notation Description

To better describe the scheme, we first clarify the terminologies and notations in Tab. 2.

3.3 Scheme Flow Description

Our privacy-preserving MSA scheme is divided into two phases: the distributed computing phase, which performs our segmenting method for long gene sequences in plaintext at child nodes, and the aggregate computing phase, which performs our privacy-preserving MSA method on ciphertexts between two master nodes.

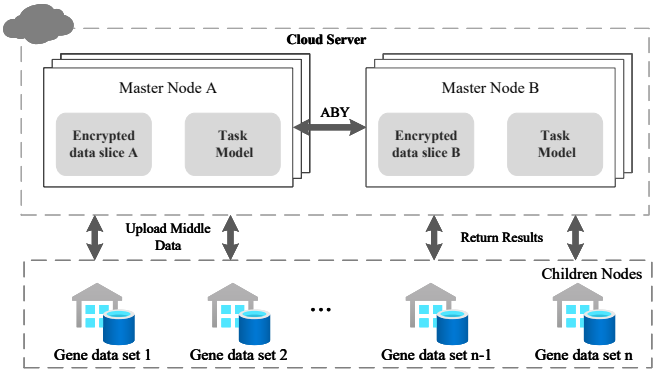

Figure 2: Distributed Privacy-preserving MSA System Model

The distributed computing phase includes subsequence segmenting, local common subsequence establishing, and seed sequence chaining to segment the long gene sequences. The aggregate computing phase includes common subsequences searching and extension sequence scoring to achieve privacy-preserving multiple sequences alignment. After the distributed calculating and the aggregate computing, the master nodes will send the result in ciphertext to both child nodes, and then all the gene data sets will get the results. The operations flow of our scheme is shown in Fig. 3.

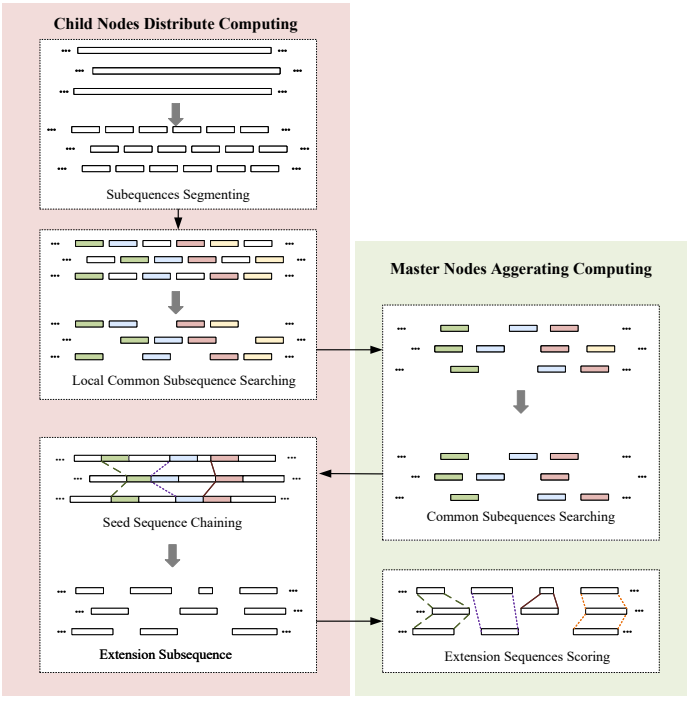

Figure 3: Operation Flowchart of Scheme

According to our scheme operation flow in Fig. 3, the operation and communication steps of our scheme between child nodes and master nodes is shown in Fig. 4. Assuming that a child node C

Table 2: Terminology Statement

Terminology	Notation	Explanation
gene sequence	S	S refers to any gene sequence in a data set.
window size	k	k refers to the window size used to segment gene sequence into some continuous overlapping subsequences.
subsequence	/	The continuous segments of one gene sequence with length k.
local common subsequence	/	The common subsequences of all local gene sequences at one node with length k.
common subsequence	/	The common subsequences of all gene sequences in all child nodes with length k.
seed sequence	/	The subsequences obtained by chaining the contiguous common subsequences and removing the overlap. Their length is greater than or equal to k.
extension subsequence	es	The subsequences of a long sequence that needs to be calculated after cutting the seed sequence.
hash value	H_{sub}	H_{sub} denotes the hash of any subsequence, $H_{sub} = H(subsequence)$.
hash list	S_H	S_H denotes the hash list of long sequence S including each H_{sub} of all subsequences of S .
/	$AncPos$	$AncPos$ records the hash value of common subsequences and its postion in each long sequence. $AncPos = (pos_0, ...pos_N, hash\_values)$

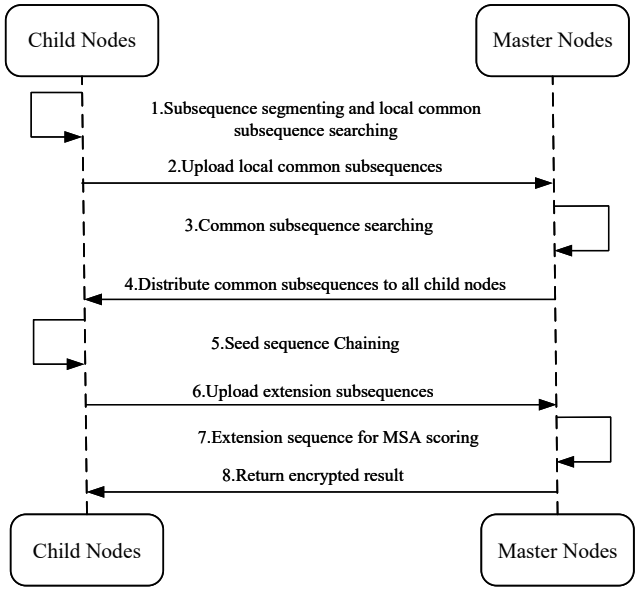

Figure 4: Computional Process Between Nodes

wants to perform MSA with other child nodes by the master nodes A and B, those nodes need to perform the following steps:

Step 1 C finds the shortest sequence in the local data set, performs subsequence segmenting in Section 4.1, and does the local common searching in Section 4.2 to find local common subsequences. Then C performs secret sharing on local common subsequences, and uploads share separately to master nodes A and B.

Step 2 A and B perform the common subsequence searching(CSS) in Section 5.2 to the shares received from C to get the share of the common subsequences and distribute it to the child nodes.

Step 3 After receiving the share of the common subsequence from A and B, C recovers the plaintext of the common subsequence by the shares and performs seed sequence chaining in Section 4.3 to get the seed sequences of this MSA. C removes the seed sequence from all the long sequences and obtains the extended subsequences that need to be calculated. Then C performs secret sharing on extended subsequences and uploads share separately to A and B.

Step 4 A and B perform the extension sequence scoring(ESS) in Section 5.3 to obtain the editing distance of these sequences in the form of secret sharing.

Step 5 A and B send the shares of the result to C to construct the MSA result.

As mentioned in Section 2.1, a basic MSA approach has three steps. Our scheme mainly performs Step 2 in a basic MSA approach distributed without information leakage. Steps 1 and 2 can be performed by C locally with the distance matrix to get the MSA result. In the following sections, we will describe how to implement the privacy-preserving MSA scheme in detail, including the segmenting method for distributed computing and the privacy-preserving MSA method for aggregate computing.

4 Segmenting Method For Distributed Computing

In order to facilitate the distributed computing of MSA, we design a segmenting method for long gene sequences, aiming to minimize the number of bases involved in aggregate computations and enhance the practicality of privacy-preserving MSA. Based on the low entropy characteristics of gene sequences, our segmentation method primarily involves dividing the long gene sequence into equidistant subsequences, finding local common subsequences at each children node, and seed sequence chaining based on the common subsequences returned by the master nodes. The child nodes

will obtain a set of common subsequences across all data sets. After the seed sequence chaining, the children nodes will get the seed sequences for all long sequences. Subsequently, the seed sequences are concatenated to serve as the basis for segmenting and aligning the long gene sequences during MSA. Our method is executed through a three-step process as follows.

4.1 Subsequence Segmenting

Gene data sets want to do MSA between each other without information leakage, which requires preprocessing gene sequences to make the data easier to privacy-preserving alignment and reduce the time-consuming. The gene sequence segmenting method aims to segment long gene sequences into short equal subsequences for establishing local common subsequences at the distributed computing phase and searching common subsequences at the aggregate computing phase. Our method can be divided into two parts: dividing the long gene sequence into short subsequences and encoding the subsequences.

A gene data set $D = \{S^0, S^1, ..., S^n\}$ has several long gene sequences. Each long sequence S is divided into overlapping multiple subsequences of length k, denoting as $\{s_0, s_1, ..., s_{m-1}, m = length(S) - k + 1\}$, and the subsequences are encoded to get a hash of S. The encoding method is described in the following.

The four types of bases A, T, C, G that make up the gene sequence are encoded as $h(A) = 0, h(T) = 1, h(C) = 2, h(G) = 3$. So the hash value of a subsequence is calculated as $H_{sub} = H(h_i) = H(h_0||h_1||...||h_{k-1})$, where $H(\bullet)$ denotes hash operation and $h_i \in \{h(A) = 0, h(T) = 1, h(C) = 2, h(G) = 3\}$. Then each long sequence S can be encoded as a hash list that can not be reconstructed as $S_H = [H_0, H_1, ..., H_{m-1}]$.

To reduce the computational complexity, we use the sliding convolution window with size k and step length 1 to segment long gene sequences. And we also design an algorithm to calculate the hash table of the S, which is described in Alg. 1.

Algorithm 1 Sliding convolution algorithm for calculating hash table

Require: S , k
Ensure: S_H
$sub\_seq = S[0 : k]$;
$S_H[] = []$;
for $(i = 0; i < (len(S) - k); i++)$;
$sub\_seq = S[i : i + k]$;
$h = hash(sub\_seq)$;
$S_H.append[h]$;
Output the hash table S_H.

In Alg. 1, we adopt sliding window to calculate the hash value. After gene sequence segmenting, each long gene sequence can be converted into a hash list S_H. The elements in this hash list are the hash of basic subsequences, and the local common subsequences can be determined by aligning S_H.

4.2 Local Common Subsequence Searching

Sequences that control the same expression in a gene sequence often contain amounts of repeated subsequences. In long gene sequence alignment, it is a feasible method to reduce the length of the sequence to be compared by searching common subsequences. To match the common subsequences of multiple long sequences, we do local match among S_H to find local common subsequences at the distributed computing phase and do aggregate searching among all the local common subsequences at the aggregate computing phase. We define some variables to describe the method to complement common subsequence searching as follows:

- S^N $S^N = \{s_0, s_1, ..., s_{l_N}\}$, one of the long sequences of a set, where N denotes the index of the sequences and it is unique in one set.
- S_H^N $S_H^N = [H_0^N, H_1^N, ..., H_m^N]$, the hash list of long sequence S^N.
- S_H^{LC} $S_H^{LC} = [H_0^{LC}, H_1^{LC}, ..., H_u^{LC}]$, the hash table of local common subsequences, which has u local common subsequences at one child node.
- H_i^{LC} The ith hash values in S_H^{LC} of local common subsequences, where $i \in [0, u]$.
- $AncPos_i^N$ $AncPos_i^N = (a_i^N, H_i^{LC})$, a tuple as a binary set containing the hash value H_i^{LC} and its positiona_i^N in the hash list S_H^N of every local long sequence S^N.

Firstly, the local common subsequences searching should be conducted. We choose the shortest sequence as the reference sequence S^R and encode it into a hash table $S_H^R = [H_0^R, H_1^R, ..., H_m^R]$. By comparing the H_i^R with the hash list of other local sequences, the local common subsequences that exist in all sequences are found. We record the position of local common subsequences in $AncPos$, so as to maintain the order of subsequence and prepare for the chaining of seed sequences at Section 4.3. The position of all local common subsequences can be record in $AncPos = [AncPos^0, AncPos^1, ..., AncPos^{v-1}]$, where v is the number of local common subsequences and $AncPos_i = [a_i^0, a_i^2, ..., a_i^N ..., a_i^{t-1}, H_i^{LC}]$ where t is the number of local long sequences .

After searching, the step 1 in Fig. 4 is finished and the child nodes can get the hash table of local common subsequences S_H^{LC} and their position $AncPos$. In step 2, all child nodes send S_H^{LC} to the master nodes to get the common seed sequences in step 3 at the aggregate computing phase by using the function designed in Section 5.2.

4.3 Seed Sequence Chaining

After the step 4 in Fig. 4, child nodes obtained common subsequences of all long sequences, and common subsequences need to be arranged in order and chaining to get seed sequences for alignment. Firstly, we need to delete high frequency subsequences to avoid multiple positions that will affect the subsequent chaining. Then we connect the reduced seed sequences by chaining method in FMindex [28] to connect a series of seed sequences.

Assumed that genomic data D to be aligned has t long sequencese, noted as $D = \{S_1, S_2, ..., S_t\}$. The common subsequences can be noted as $AncPos^C = [AncPos_0^C, AncPos_1^C, ..., AncPos_{Z-1}^C]$, where $AncPos_i^C = [a_i^0, a_i^2, ..., a_i^U ..., a_i^t, a_i^{t-1}, H_i^C]$, a_i^U are the position of the common subsequence i on long sequence U, H_i^C is the hash value

of the seed sequence, z represents the number of the common subsequences. On this basis, the scoring function $f(i)$ for searching the nearest seed sequence and sorting the seed sequences is defined as follows:

$$f(i) = \begin{cases} 0 & i = 0 \\ max(maxf(j) + \alpha(i,j) - \beta(i,j), 0) & i > 1 \end{cases}$$

The function $\alpha(j,i)$ evaluates the distance between common subsequences i and j by their positon in the hash table, and the calculation method is as follows:

$$\alpha(j,i) = \sqrt{\frac{1}{t}\sum_{u=1}^{t}(a_i^u - a_j^u)^2}$$

The function $\beta(i,j)$ is an affine gapenalty that can be changed according to different demands. We evaluate the order of the subsequences by $f(i)$, and we considered the subsequences with higher scores to be closer to each other. For downstream processing in SMPC, we will not link the common subsequences containing overlap, but only sort the subsequences by the index of the tuple *AncPos* according to the chaining value. We define the previous sequence position of target subsequence i as $P(i)$, so $P(i)$ is defined as follows:

$$P(i) = \begin{cases} 0 & f(i) = 0 \\ argmax(maxf(j) + \alpha(i,j) - \beta(i,j), 0) & f(i) > 0 \end{cases}$$

After that, we can get an ordered subsequences recorded in *AncPos*. The overlapping subsequences are connected in order to form unequal length seed sequences as the segmentation anchors of the long sequences at the aggregate computing phase. The chaining of the seed sequences on different long sequences are shown in Seed Sequences Chaining part of Fig. 3 .

It can be seen from Fig. 3 that subsequences 1 and 2 may have overlapping parts on some sequences. Then seed sequence chaining the common subsequences by merging the overlapping part and then record the beginning position. The record of the ordered and merged seed sequences $AncPos^S = [AncPos_0^S, AncPos_1^S, ..., AncPos_{w-1}^S]$ is obtained, where w represents the number of the seed sequences. After that, child nodes can remove the seed sequences of all long gene sequences and finally get extension subsequences need to be calculated at aggregate computing. Then child nodes apply the arithmetic secret sharing method to share those extension subsequences set $ES = \{ES^1, ES^2, ..., ES^t\} = \langle ES\rangle_A + \langle ES\rangle_B$, where $ES^1 = \{es_0^1, es_1^1, ..., es_t^1\}$. Through the distributed calculating, child nodes upload the ciphertexts of $\langle ES\rangle_A$ to master node A and $\langle ES\rangle_B$ to master node B.

5 Privacy-preserving MSA Method For Aggregate Computing

By comparing the position of the seed sequences, master nodes can determine the extension subsequences that need to be aligned. Master node A has the share $\langle ES\rangle_A$ and master node B has the share $\langle ES\rangle_B$. Alignment of the starting position and edit distance calculation operations are performed on extension subsequences. Therefore, we design the privacy-preserving method, which includes the common subsequence search for step 3 and the extension sequence scoring (ESS) for step 7 in Fig. 4 to conduct the privacy-preserving calculation of MSA, and by combining basic functions. Finally, our scheme conduct the privacy-preserving calculation of MSA through these functions.

5.1 Basic Functions

In order to achieve common subsequence search and the extension sequence scoring in privacy, equality (EQ) function and the minimum of three elements (MIN3) function need to be done as basic functions.

5.1.1 Equality Evaluating (EQ). EQ function is using to compare the value in different hash list to find out whether it is a common seubsequence.

Given x_1 and x_2 to be compared are uploaded by Node A and Node B as a secret sharing data format, that is $\langle x_1\rangle = \langle x_1\rangle_A + \langle x_1\rangle_B$ and $\langle x_2\rangle = \langle x_2\rangle_A + \langle x_2\rangle_B$. The EQ function conduct the function:

$$f_{eq}(x_1, x_2) = \begin{cases} 0 & x_1 = x_2 \\ 1 & x_1 \neq x_2 \end{cases}$$

We design the equality garbled circuit to implement function $f_{eq}(x_1, x_2)$ with only one add gate. Before inputing the data into EQ circuit, servers need to conduct some pre-calculated. After the secret sharing, Node A obtains $\langle x_1\rangle_A = \langle x_1\rangle - \Delta x_A$ and $\langle x_2\rangle_A = \langle x_2\rangle - \Delta x_B$. Node B obtains $\langle x_1\rangle_B = \Delta x_A$ and $\langle x_2\rangle_B = \Delta x_B$. Node A inputs $\langle x_1\rangle_A - \langle x_2\rangle_A$ and Node B inputs $\langle x_1\rangle_B - \langle x_2\rangle_B$. The EQ circuit adds the two inputs, and outputs the sum θ.

$$\begin{aligned}\theta &= \langle x_1\rangle_A - \langle x_2\rangle_A + \langle x_1\rangle_B - \langle x_2\rangle_B \\ &= (\langle x_1\rangle_A + \langle x_1\rangle_B) - (\langle x_2\rangle_A + \langle x_2\rangle_B) \\ &= \langle x_1\rangle - \langle x_2\rangle\end{aligned}$$

Obviously, if θ is equal to 0, then x_1 is equal to x_2. Through this circuit, we design the EQ function which main step is described in Alg. 2.

Algorithm 2 Equality Evaluating Algorithm

Require: Node A inputs $\langle x_1\rangle_A$, $\langle x_2\rangle_A$, Δ_0 and $(1-\Delta_1)$
Node B inputs $\langle x_1\rangle_B$, $\langle x_2\rangle_B$, $-\Delta_0$ and Δ_1
Ensure: Node A outputs $\langle f_{eq}(x_1,x_2)\rangle_A$
Node B outputs $\langle f_{eq}(x_1,x_2)\rangle_B$
$in1 = \langle x_1\rangle_A - \langle x_2\rangle_A$
$in2 = \langle x_1\rangle_B - \langle x_2\rangle_B$
$ADD(in1, in2) = \theta$;
if $\theta = 0$ **then**
Node A: $\langle f_{eq}(x_1,x_2)\rangle_A = \Delta_0$;
Node B: $\langle f_{eq}(x_1,x_2)\rangle_B = -\Delta_0$;
else Node A: $\langle f_{eq}(x_1,x_2)\rangle_A = (1-\Delta_1)$;
Node B: $\langle f_{eq}(x_1,x_2)\rangle_B = \Delta_1$;

We will implement the EQ circuit using a garbled circuit[46] and Oblivious Transfer[37] protocol since there is only one gate circuit so the expend of by using Yao's sharing is acceptable. To perform CSS function, we need to make some changes to the output of Algorithm 2 to provide calibration for aligning and slicing of the common seed sequences by recording the position of them. When EQ is used in the CSS function, Node A only inputs $(\langle x_1\rangle_A - \langle x_2\rangle_A)$, Δ_0 and Node B only inputs $(\langle x_1\rangle_B - \langle x_2\rangle_B)$. While Node A and Node

B do not need to output $f_{eq}(x_1, x_2)$ in the form of secret sharing, in fact Node A and Node B will directly obtain the result of the EQ function, and record the corresponding elements in *AncPos*.

5.1.2 Minimum of Three Calculating (MIN3). MIN3 function is using to find minimum edit distance during the extension sequence scoring.

Given x_1, x_2 and x_3 to be compared are uploaded by Node A and Node B as a secret sharing data format, that is $\langle x_1 \rangle = \langle x_1 \rangle_A + \langle x_1 \rangle_B$, $\langle x_2 \rangle = \langle x_2 \rangle_A + \langle x_2 \rangle_B$ and $\langle x_3 \rangle = \langle x_3 \rangle_A + \langle x_3 \rangle_B$. The MIN3 conduct the function:

$$f_{min3}(x_1, x_2, x_3) = \begin{cases} x_1 & x_1 < x_2, x_1 < x_3 \\ x_2 & x_2 \leq x_1, x_2 < x_3 \\ x_3 & x_3 \leq x_1, x_3 \leq x_2 \end{cases}$$

In order to conduct this function in MIN3 function. Firstly, we design a garbled circuit shown in Fig. 5 to campare the two inputs. The MIN circuit conduct the function:

$$y(x_1, x_2) = y_{12} = \begin{cases} 1 & x_1 < x_2 \\ 0 & others \end{cases}$$

This function $f_{min}(x_1, x_2)$ conducted by circuit compares the two

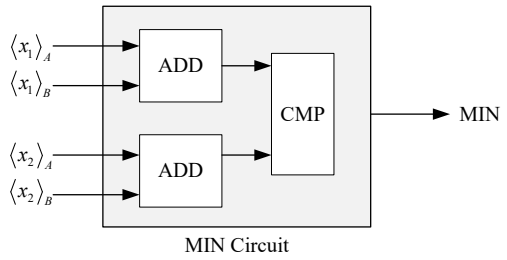

Figure 5: Minimum Circuit

inputs, and outputs 0 or 1 after processing. Secondly, we can construct the $f_{min3}(x_1, x_2, x_3)$ based on it.

$$f_{min3}(x_1, x_2, x_3) = y_{12} \cdot y_{13} \cdot x_1 + \overline{y_{12}} \cdot y_{23} \cdot x_2 + \overline{y_{13}} \cdot \overline{y_{23}} \cdot x_3$$

Through this equation, we design the MIN3 function which can calculate the minimum of three elements is described in Alg. 3.

Algorithm 3 Minimum of Three Elements Algorithm

Require: Node A inputs $\langle x_1 \rangle_A$, $\langle x_2 \rangle_A$, and $\langle x_3 \rangle_A$
Node B inputs $\langle x_1 \rangle_B$, $\langle x_2 \rangle_B$, and $\langle x_3 \rangle_B$
Ensure: Node A outputs $\langle f_{min3}(x_1, x_2, x_3) \rangle_A$
Node B outputs $\langle f_{min3}(x_1, x_2, x_3) \rangle_B$
$y_{12} = MIN(x_1, x_2)$;
$y_{13} = MIN(x_1, x_3)$;
$y_{23} = MIN(x_2, x_3)$;
$f_{min3}(x_1, x_2, x_3) = y_{12} \cdot y_{13} \cdot x_1 + \overline{y_{12}} \cdot y_{23} \cdot x_2 + \overline{y_{13}} \cdot \overline{y_{23}} \cdot x_3$ Node A: $\langle x_{min} \rangle_A$;
Node B: $\langle x_{min} \rangle_B$;

After the calculation in Alg. 3, this function can obtain the minimum of x_1, x_2, x_3 without leaking information to master nodes A and B.

5.2 Common Subsequence Searching (CSS)

The common subsequence searching (CSS) is to find the common subsequences of all the gene data sets. Master node A input the hash of local common subsequencees $AncPos^{LC}$ into CSS, and obtains the common subsequences $AncPos^{C}$. The algorithm is described in Alg. 4.

Algorithm 4 Common Seed Search Algorithm

Require: Node A inputs $AncPos^{LC}$
Ensure: Node A outputs $AncPos^{C}$
$AncPos^{C}$ = **reduce**(lamda x,y: x&y,AncPos[h])
Node A: $AncPos^{C}$ = $[AncPos_0^C, AncPos_1^C, ..., AncPos_z^C]$;

After the CSS, we can obtain the common subsequenceS $AncPos^{C}$ recorded the starting position of each subsequence in the form of secret sharing, and master nodes send them to child nodes to perform step 5 in Fig. 4. During the aggregate computing of the CSS, no information about the gene sequences is leaked.

5.3 Extension Sequence Scoring (ESS)

ESS is using to calculate the edit distance between the correspond extension subsequences of different sequences. After child nodes perform step 5 by the way in Section 4.3 and perform step 6, master nodes can obtain the share of extension subsequences $\langle ES \rangle_A$ and $\langle es \rangle_B$ as the input of ESS. Generally, edit distance is used to score the direct distance of two extension subsequences. The ESS conducts the function:

$$f_{ses}(es_1, es_2) = min \begin{cases} d_{i-1,j} + c_{ins} \\ d_{i,j-1} + c_{del} \\ d_{i,j} + c_{sub} \end{cases}$$

$$0 \leq i < len(es_1), 0 \leq j < len(es_2)$$

The master nodes input two extension sequences es_1, es_2 in the form of secret sharing, and obtain the edit diatance in the form of secret sharing too. The algorithm is is described in Alg. 5.

Algorithm 5 Extension Sequence Scoring Algorithm

Require: Node A inputs $\langle es_1\rangle_A, \langle es_2\rangle_A$
Node B inputs $\langle es_1\rangle_B, \langle es_2\rangle_B$
Ensure: Node A outputs $\langle d\rangle_A$
Node B outputs $\langle d\rangle_B$
for $(i = 0; i < len(es_1); i++)$ **do**
Node A: $\langle d_{i,0}\rangle_A = i$; Node B: $\langle d_{i,0}\rangle_B = 0$;
for $(j = 0; j < len(es_2); j++)$ **do**
Node A: $\langle d_{0,j}\rangle_A = 0$; Node B: $\langle d_{0,j}\rangle_B = j$;
for $(i = 0; i < len(es_1); i++)$ **do**
for $(j = 0; j < len(es_2); j++)$ **do**
$c_{sub} = EQ(es_1(i-1, j-1), es_2(i-1, j-1))$;
$d_{sub} = d_{i-1,j-1} + c_{sub}$
$d_{ins} = d_{i-1,j} + c_{ins}$
$d_{del} = d_{i,j-1} + c_{del}$
$d = MIN3(d_{sub}, d_{ins}, d_{del})$
Node A: $\langle d\rangle_A = \langle d[len(es_1)][len(es_2)]\rangle_A$
Node A: $\langle d\rangle_B = \langle d[len(es_1)][len(es_2)]\rangle_B$

Through ESS, we obtain the edit distance between the sequences without leaking the base order of them. Neither Node A nor Node B can get plaintext gene sequences or the results. After calculating the edit distance of unmatchable extension subsequences on the long sequences, and master nodes can obtain the secret sharings of edit distance.

The edit distance of the seed sequences is considered to be 0, so the result of pairwise alignment is the sum of the edit distance of the unmatchable extension subsequences. For example, the edit distance between long sequences S^1 and S^2 can be calculated as $ED(S^1, S^2) = ED(ES^1, ES^2) = \sum_{i=0}^{z} d_i(ES_1[i], ES_2[i])$. The algorithm is is described in Alg. 6.

Algorithm 6 Privacy-preserving MSA Algorithm

Require: Node A inputs $\langle ES_1\rangle_A, \langle ES_2\rangle_A$
Node B inputs $\langle ES_1\rangle_B, \langle ES_2\rangle_A$
Ensure: Node A outputs $\langle ED\rangle_A$
Node B outputs $\langle ED\rangle_B$
for $(i = 0; i < z); i++)$ **do**
$es_1 = ES_1[i]$, $es_2 = ES_2[i]$
$d_i = ESS(es_1, es_2)$
$ED(S^1, S^2) = ED(ES^1, ES^2) = \sum_{i=0}^{z} d_i$
Node A: $\langle ED\rangle_A$
Node A: $\langle ED\rangle_B$

The algorithm is described in Alg. 6. We obtain the edit distance of long gene sequences without exposing the order of bases. To achieve MSA, it is necessary to calculate the extension score for different sequences based on seed sequences. Then, we can calculate the MSA result between different gene data sets by doing this several times or construct a multiple demensiong matrixes by Alg. 6. By distributed computing on child nodes, our scheme reduces the number of repeated alignments on master nodes, and improves the efficiency of privacy-preserving MSA.

Table 3: Communication Overhead During Secret Sharing

	Cipher Computation	Communication	Messsages
Shr^A	0	l	1
Rec^A	0	l	1
$Y2A$	$6l$	$lk + (l^2 + l)/2$	2
Shr^Y	$6l$	$2lk$	2
$A2Y$	$12l$	$6lk$	2

6 Scheme Analysis

To prove the security and practicality of the scheme, we analyze the scheme from three aspects: security, privacy, and efficiency.

6.1 Security Analysis

Our security analysis is predicated on the semi-honest parties' assumption. Specifically, the parties involved in the implementation of the scheme are assumed to be honest yet curious. They are capable of carrying out the required calculations and communication to the protocol's specifications. However, their curiosity extends to the content of the communication, potentially leading them to store intermediate variables to glean additional information. It is important to note that these parties do not engage in active attacks or attempt to compromise the integrity of the protocol. Moreover, the servers participating in the computation are assumed to be non-colluding, indicating that they do not disclose their respective information to one another.

Our scheme is based on foundational cryptographic principles and aims to achieve secure multiple sequence alignment through distributed computing and aggregate computing. The data transmitted between child nodes and master nodes is ciphertext, so the data transmission is secure. The aggregate computing conducted between A and B is also a ciphertext operation. In adherence to the principle of compositional security, the overall security of our scheme is contingent upon the individual security of each function achieved by secret sharing. Therefore, the data is secure during the computing and transmission, our scheme can be deemed secure as a whole.

6.2 Communication Overhead Analysis

Our scheme uses the ABY framework to implement privacy preserving MSA, inculding child nodes conduct gene sequences segmentation in the distributed computing phase and master nodes conduct MSA by secret sharing in the aggregate computing phase. The child nodes do the setup phase of the scheme, which is not included in the calculation of the ciphertext communication overhead. Assuming that the length of the message to be transmitted is l bits and the symmetric security parameter is k, the communication overhead in this scheme is listed in Tab. 3.

As is listed in Tab. 3, the communication of A2Y is the most expensive part, and the part of sharing and reconstructing message in arithmetic sharing is almost free. We need to minimize the number of times we need to use A2Y transformations to improve efficiency.

In our scheme, the ESS function requires one A2Y transition and one Y2A transiton, and the rest part of our scheme needs Shr^A and Rec^A which can be done by arithmetic sharing and reconstructing. Besides, we put the construction of Shr^A at the child nodes, so there is no extra communication overhead, and further improve the query efficiency of the aggregated computing.

7 Experimental Evaluation

Our scheme is implemented in Ubuntu Linux OS 20.04. For the execution of our scheme, we employ fast garbled circuits[17] for Yao's share to achieve the compare calculation functions and using Paillier homomorphic encrytion[35] to construct the Beaver triple for Arithmetic sharing. The Oblivc[48] is utilized for GC and file processing, while the construction of offline seeds is performed using Python 3.8.

We simulate distributed computing by conducting the preprocess of genomic files locally and output the middle data by packing them in a file. The aggregate computing is simulated through the localhost connection of the Linux system, including the middle files transferring between the child node and master nodes and the OT communication between Nodes A and B. Consider that the communication between master nodes and child nodes are similar, so our experiment architecture only including two master nodes and two child nodes. It is tested with the real-life dataset 2019nCoVR_20200301 in the form of fasta files. There are 131 long sequences in this dataset, the minimum sequence length is 29409 bps, the longest sequence length is 29927 bps, the average length is 29860 bps, and the dataset size is 4. 7MB. According to the experimental data of Liu et al., the length of the subsequence of the long sequences that is more efficient and accurate is 39-41 bps. In the experiment, we chose a sliding window length of 40 bpS sand step size of 1 bp to cut the long sequence.

7.1 Time Consuming Analysis

Our experimental scheme is divided into two phases: a distributed computing phase and an aggregate computing phase. In the distributed computing phase, we processed all the gene long sequence files in the dataset, obtained the segmented 40 bps subsequence fragments, established the seed sequences of the gene data set through local hash value comparison, and recorded the starting position and sequence length of each seed sequence. In the aggregate computing phase, the hash values of seed sequences from all child nodes are compared, the common seed sequences is reconstructed, and the subsequence that needs to be edited by distance alignment is segmented. Then, by constructing a confounding circuit, the unmatching subsequences are calculated one by one, and the final result is obtained after summing the ciphertext results. For different data sizes, the comparison between sequence lengths need to be calculated by edit distance after using our scheme, and the initial length is listed in Tab. 4. From Tab. 4, With the increase in the number of long gene sequences involved in alignment, the length of gene sequences requiring edit distance alignment also increased, but the maximum was not more than 6.16%. It is understandable that as the number of sequences in a multiple sequence alignment increases, the common part of the long sequences decreases, so the length of the sequence to be aligned increases. But after using the segmentation method of our scheme, it can be seen that the base length of edit distance alignment for long gene sequences after sequence segments is reduced to under 6.16%, which reduces the depth and number of gates needed to construct secret sharing circuits, to improve the efficiency of privacy-preserving multiple sequence alignment. The time of each phase of privacy-preserving MSA using the segmented method of our scheme is shown in Fig. 6.

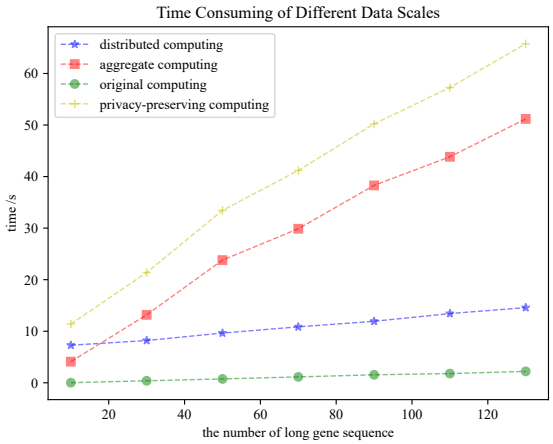

Figure 6: Time Consuming of Different Data Scales

With the increase in the number of long gene sequences, the time cost required by our scheme presents a linear relationship. However, because we constructed the common seed sequences in the offline phase, the offline construction phase only be performed once. Compared with the multiple sequence alignment method for every single comparison, the increase in the time cost of our scheme with the increase in the number of sequences is lower. As shown in Fig. 6, with the expansion of the data scale, the online phase is the most time-consuming part, and the time cost is almost equal to the overall cost. In the case of the maximum data size of $120 * 29856$ bps, the time required for a single alignment in the online phase is 51.19s. We compare our scheme with the MSA scheme MAFFT [19] with edit distance calculation by pairwise alignment on this dataset. The MAFFT takes 22.38s, while the time of our scheme is 65.77s. It takes about three times as long as MAFFT because it adds the overhead of the privacy-preserving method on the aggregate computing phase. After adding the privacy-preserving method, our time cost increases compared with the plaintext calculation of edit distance, but it is still within the acceptable range.

7.2 Accuracy Rating Analysis

The calculation of the edit distance can be conceptualized as the count of operations needed to transform two distinct sequences into identical sequences. So the MSA is basically how to make multiple gene sequences equal to each other. For disparate gene sequences, alignment is typically achieved by employing dynamic programming to fill gaps. In our approach, we have devised an

Table 4: The Length of a Gene Sequence Required Edit Distance Calculations

Sequence Number	Full Sequence Calculated Method			Our Method			Percentage %
	max	min	average	max	min	average	
10	29899	29890	29892	53	44	46	0.15
50	29903	29782	29871	739	618	707	0.15
90	29903	29409	29856	1404	910	1357	4.55
130	29927	29409	29860	1907	1389	1839	6.16

Percentage: percentage = (the subsequences length to be calculated / the total sequence length) ∗100%.

Table 5: Comparison of Different Filling Methods of Unequal-Length Gene Sequences

Method \ Scale	10		50		90		130	
	average l	accuracy rate	average l	accuracy rate	average l	accuracy rate	average l	accuracy rate
Our Method	29899	100.00	29903	100.00	29903	100.00	29927	100.00
Begining Method	29899	75.11	29903	52.24	29903	48.31	29927	51.02
Ending Method	29899	96.67	29903	81.04	29903	80.29	29927	79.04
Random Method	29899	75.11	29903	42.74	29903	39.13	29927	41.47
Original Method	29899	100.00	29903	100.00	29903	100.00	29927	100.00

Original Method denotes the method of calculating edit distance for MSA with sequences' full length.
Length l denotes the length of gene sequence.
Beginning/Ending/Random Method denotes different filling methods for filling empty spaces at the different positions of the sequence.

optimized gap-filling method specifically designed for lengthy sequences, aiming to enhance the efficiency of privacy-preserving computation without compromising accuracy. We used a few different filling methods, including filling empty spaces at the beginning, end, and random positions of the unequal sequence. Compared to our scheme and the original dynamic programming method of edit distance calculation, and the accuracy contrast result is listed in Tab. 5.

As can be seen from Tab. 5, for multiple sequence alignment with different data scales, the accuracy rate of multiple sequence alignment after filling in different ways is relatively random and not high, but the accuracy rate of segmentation and then edit distance calculation using the long sequence segmentation method proposed in our scheme is high. For the sequence data set of gene group length at the same locus, the edit distance calculated by our scheme is the same as that calculated by the original edit distance, with an accuracy of up to 100, which can achieve privacy-preserving and accurate long MSA.

Besides, we experimented with different datasets to compare our scheme with the general MSA method MAFFT [19], which supports performing the MSA for long gene sequences. The parameters with MAFFT are as follows: the scoring matrix is 1PAM/k=2, the gap opening penalty is 1, and the offset value is 0. We use editing distance to score MSA, and the result of MSA is measured as the average score, which is the sum of pairs divided by the number of sequences. In average score calculating, 0 is added to the score for the two bases from the same column that are matched, 1 is added to the score for mismatched, and 1 is added to the score with a gap. The score of MAFFT output files denotes as S_{MAFFT}, and the score of our scheme denotes as S_{our}. The result is listed in Tab. 6. Except for the HIV dataset, our results are nearly equal to MAFFT, with an acceptable margin of error due to the selection of MSA central sequences. As for the multi-sequence alignment of HIV, the deviation rate is 6.5%; because the similarity of HIV data sets is lower and the number of data is small, the choice of basic sequence in MSA will cause more considerable differences. Considering that the privacy-preserving calculation demands lower computation complexity, we construct the MSA using the progress method mentioned in Section 2.1.2, which begins with pairwise alignment. In pairwise comparison, our scheme is still accurate for HIV data, but the choice of the basic sequence and the similarity of the dataset influences our output. We will discuss the problems of low sequence similarity for experiments in the section on limitation and discussion.

7.3 Privacy and Utility Analysis

In the distributed computing phase, the data processed by local child nodes causes no privacy leakage. The aggregate computing phase includes the communication between child nodes and master nodes and the interaction between two master nodes,and the data held by the two master nodes are subjected to secret sharing. The privacy of this phase relies on the privacy of the secret sharing algorithm. We use the ABY framework to achieve privacy-preserving results, so we use this as a baseline to assess our scheme.

During aggregate computing, our scheme employs anchors, which store the positions of different seed sequences to facilitate the segmentation of long sequences. While the position information of the common seed sequences may be exposed, the master node can use this position to guess some information about the base. Assuming that the average length of sequences is l, the window step is s, and

Table 6: Score Comparison With MAFFT

Score / Dataset	Sequence Number	S_{MAFFT}		S_{our}		Deviation Rate %
		sum score	average score	sum score	average score	
2019nCoVR_20200310	130	5113	39.33	5095	39.19	0.36
SARS-CoV-2	50	2016	40.32	2016	40.32	0
mt genome (20x)[29]	300	10637	35.45	10630	35.43	0.06
HIV data main 1000[5]	27	50521	1871.15	54038	2001.41	6.5

Table 7: Comparison of Different Dataset MSA

Dataset	Average l	Calculated l	Percentage%	Sequence Number	Utility Rate%	Window Size k
2019nCoVR_20200301	29860	1005	3.37	130	100.00	20
	29866	1371	1.24	50	100.00	30
	29860	1292	4.33	130	100.00	30
	29860	1839	6.16	130	100.00	40
SARS-CoV-2	29857	324	1.09	50	100.00	20
	29857	398	1.33	50	100.00	30
	29858	549	1.84	80	100.00	30
	29857	581	1.94	50	100.00	40
	29858	1764	2.56	80	100.00	40
mt genome (20x)[29]	16569	4244	25.61	180	89.54	20
	16569	8113	48.96	180	99.46	40
	16569	9767	58.95	300	100.00	40
HIV data main 1000[5]	10287	10085	98.03	27	85.19	13
	10287	10144	98.61	27	100.00	15

Length l denotes the average length of all gene sequences in one dataset.
Calculated l denotes the average length that needs to be computed of all gene sequences in one dataset.
Percentage: percentage = (the subsequences length to be calculated / the total sequence length) $*100\%$.
Utility Rate: denotes the utility after using our scheme based on the global MSA on long gene sequences, it equals to the accuracy rate here.

the number of common seed sequences is m. The privacy loss using our scheme is

$$PLoss = 2^{2*m}/(2^{2*s} * 2^{2*l}) = \frac{1}{2^{2*(l-m+s)}}$$

Our scheme aims for long gene sequences, so $l >> m$. For example, on the dataset SARS-CoSARS-CoV-2, the $l = 29899$, $m = 125$ and $s = 1$, so $Loss = \frac{1}{2^{2*28775}}$, which can be considered remains undisclosed.

As for utility analysis, secret sharing is a lossless privacy preserving method, so the utility depends on the segment method's parameter. In our scheme, the utility loss is from the window step s and the chaining method. The loss caused by the chaining method comes from the approximate feature of MSA itself. The loss caused by window step s comes from a small read mismatch; the larger s is, the easier it is to ignore the mismatch of a short read. When $s = 1$, single-base mismatch detection can be realized. The s influences the loss of privacy and utility. At the same time, s is negatively correlated with privacy loss and positively correlated with utility loss. So, for long gene sequences, we can make a tradeoff to choose $s = 1$ because of $l >> m$. Otherwise, we need to adjust s to reduce privacy loss. Our scheme aims for long gene sequences, so we can choose $s = 1$ to remain the utility of the dataset for global MSA. We experimented our scheme with different gene data sets of various lengths. Those gene data files are in the form of fasta, which concludes the error small read caused by gene sequencing. The result is shown in Tab. 7.

As shown in Tab. 7, our scheme can remain high utility for different datasets, but it depends on the sequence number, window length, and the similarity of all the sequences in a dataset. For 2019nCoVR_20200301 and SARS-CoV-2, the average sequence length of the two datasets is near 30000 bps, and they are highly similar, with the suitable window size 40, the accuracy rate can reach 100% with the computation length reduced under 10% of the total length. For mt genome(20x), the accuracy rate can also reach a high level by adjusting the window length and increasing the number of sequences. However, the computation length only reduces to nearly 50%. As for HIV data main 1000, it does not have a high similarity; although our scheme can achieve a high accuracy rate, it can barely reduce the computation length. In conclusion, our scheme can optimize privacy-preserving MSA of relatively long similar sequences with high accuracy, but it cannot achieve good efficiency for data sets with low similarity. We will discuss this part in the limitation analysis.

7.4 Limitation and Discussion

From the efficiency, accuracy, and privacy and utility analysis perspective, our scheme can conduct efficient privacy-preserving MSA, but it still has some limitations. We analyze limitation of the scheme and discuss the future direction that can be optimized.

Firstly, our scheme operates under the assumption of semi-honest parties, where participants cannot collude with each other. This is a crucial aspect of ensuring privacy. Therefore, our scheme must be implemented within this secure model, requiring at least two isolated master nodes. Additionally, while we simulate the OT communication at local hosts, it's essential to consider the potential communication overhead in a complex cloud server environment.

Secondly, our scheme achieves the privacy-preserving MSA only on the global alignment algorithm. If future work wants to achieve local alignment using this method, further work needs to be extended to get the scores for local matching, especially on the segment method.

Besides, regarding the conduction on different datasets, we found that our scheme performed well on the highly similar long gene sequences MSA because the choice of basic sequence influences more when those sequences have more differences. However, they need more optimization on gene sequences with low similarity. Moreover, we also find that window size strongly influences the accuracy of our results. It cannot perform well in reducing calculation length if it is too short, making it poorly optimized on larger data scales. If it is too short, it can decrease the accuracy of the scheme. Further discussion is needed on the relativity of the MSA datasets' sequence length, entropy, and segment window size. Further discussion is needed on the relativity of the MSA datasets' sequence length, entropy, and segment window size.

Our scheme needs to find a balance of accuracy and efficiency for conducting on low similar gene data sets. For better time consumption and accuracy performance on large data scales, we still need to further optimize the secret sharing method to find a suitable combination of ABY and the best segment method choice.

8 Conclusion

In this paper, we propose a privacy-preserving scheme to achieve the multiple sequence alignment. The scheme encompasses a segmentation method for handling long gene sequences by distributed computing and utilizes ABY secret sharing to achieve privacy-preserving MSA. Our scheme can efficiently perform multiple sequence alignment for long gene sequences with high accuracy. Future research directions may involve enhancing the utility of the alignment scheme through parallel alignment or optimizition on segment method.

Acknowledgments

This project is supported by the Key R&D project of Hebei Province (22340701D) and the National Natural Science Foundation of China (No.61971021).

References

[1] Gilad Asharov, Shai Halevi, Yehuda Lindell, and Tal Rabin. 2018. Privacy-Preserving Search of Similar Patients in Genomic Data. *Proc. Priv. Enhancing Technol.* 2018, 4 (2018), 104–124. https://doi.org/10.1515/popets-2018-0034

[2] Michael Backes, Pascal Berrang, Matthias Bieg, Roland Eils, Carl Herrmann, Mathias Humbert, and Irina Lehmann. 2017. Identifying Personal DNA Methylation Profiles by Genotype Inference. In *2017 IEEE Symposium on Security and Privacy, SP 2017, San Jose, CA, USA, May 22-26, 2017*. IEEE Computer Society, 957–976. https://doi.org/10.1109/SP.2017.21

[3] Donald Beaver. 1991. Efficient Multiparty Protocols Using Circuit Randomization. In *Advances in Cryptology - CRYPTO '91, 11th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1991, Proceedings (Lecture Notes in Computer Science, Vol. 576)*, Joan Feigenbaum (Ed.). Springer, 420–432. https://doi.org/10.1007/3-540-46766-1_34

[4] Gergely Biczók and Pern Hui Chia. 2013. Interdependent Privacy: Let Me Share Your Data. In *Financial Cryptography and Data Security*, Ahmad-Reza Sadeghi (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 338–353.

[5] Apetrei C, Hahn B, Rambaut A, and et al. 2021. HIV Sequence Compendium 2021. In *Published by Theoretical Biology and Biophysics Group*. Los Alamos National Laboratory, NM, LA–UR–23–22840.

[6] Maria Chatzou, Cedrik Magis, Jia-Ming Chang, Carsten Kemena, Giovanni Bussotti, Ionas Erb, and Cédric Notredame. 2016. Multiple sequence alignment modeling: methods and applications. *Briefings Bioinform.* 17, 6 (2016), 1009–1023. https://doi.org/10.1093/bib/bbv099

[7] Daniel Demmler, Thomas Schneider, and Michael Zohner. 2015. ABY - A Framework for Efficient Mixed-Protocol Secure Two-Party Computation. In *22nd Annual Network and Distributed System Security Symposium, NDSS 2015, San Diego, California, USA, February 8-11, 2015*. The Internet Society. https://www.ndss-symposium.org/ndss2015/aby---framework-efficient-mixed-protocol-secure-two-party-computation

[8] Iman Deznabi, Mohammad Mobayen, Nazanin Jafari, Oznur Tastan, and Erman Ayday. 2018. An Inference Attack on Genomic Data Using Kinship, Complex Correlations, and Phenotype Information. *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 15, 4 (2018), 1333–1343.

[9] Reza Ghasemi, Md Momin Al Aziz, Noman Mohammed, Massoud Hadian Dehkordi, and Xiaoqian Jiang. 2017. Private and Efficient Query Processing on Outsourced Genomic Databases. *IEEE J. Biomed. Health Informatics* 21, 5 (2017), 1466–1472. https://doi.org/10.1109/JBHI.2016.2625299

[10] Oded Goldreich, Silvio Micali, and Avi Wigderson. 2019. How to play any mental game, or a completeness theorem for protocols with honest majority. In *Providing Sound Foundations for Cryptography: On the Work of Shafi Goldwasser and Silvio Micali*, Oded Goldreich (Ed.). ACM, 307–328. https://doi.org/10.1145/3335741.3335755

[11] M. Gymrek, A. L. Mcguire, D. Golan, E. Halperin, and Y. Erlich. 2013. Identifying Personal Genomes by Surname Inference. *Science* 339, 6117 (2013), 321–324.

[12] Arif Harmanci and Mark Gerstein. 2016. Quantification of Private Information Leakage from Phenotype-Genotype Data: Linking Attacks. *Nature methods* 13 (02 2016). https://doi.org/10.1038/nmeth.3746

[13] Mohammad Zahidul Hasan, Md Safiur Rahman Mahdi, Md. Nazmus Sadat, and Noman Mohammed. 2018. Secure count query on encrypted genomic data. *J. Biomed. Informatics* 81 (2018), 41–52. https://doi.org/10.1016/j.jbi.2018.03.003

[14] Zaobo He, Jiguo Yu, Ji Li, Qilong Han, Guangchun Luo, and Yingshu Li. 2020. Inference Attacks and Controls on Genotypes and Phenotypes for Individual Genomic Data. *IEEE ACM Trans. Comput. Biol. Bioinform.* 17, 3 (2020), 930–937. https://doi.org/10.1109/TCBB.2018.2810180

[15] Daniel S. Hirschberg. 1997. Serial computations of Levenshtein distances. In *Pattern Matching Algorithms*. Oxford University Press, 123–141.

[16] Nils Homer, Szabolcs Szelinger, Margot Redman, David Duggan, Waibhav Tembe, Jill Muehling, John Pearson, Dietrich Stephan, Stanley Nelson, and David Craig. 2008. Resolving Individuals Contributing Trace Amounts of DNA to Highly Complex Mixtures Using High-Density SNP Genotyping Microarrays. *PLoS genetics* 4 (09 2008), e1000167. https://doi.org/10.1371/journal.pgen.1000167

[17] Yan Huang, David Evans, Jonathan Katz, and Lior Malka. 2011. Faster Secure Two-Party Computation Using Garbled Circuits. In *20th USENIX Security Symposium, San Francisco, CA, USA, August 8-12, 2011, Proceedings*. USENIX Association. http://static.usenix.org/events/sec11/tech/full_papers/Huang.pdf

[18] Yatong Jiang, Tao Shang, and Jianwei Liu. 2023. Secure Counting Query Protocol for Genomic Data. *IEEE ACM Trans. Comput. Biol. Bioinform.* 20, 2 (2023), 1457–1468. https://doi.org/10.1109/TCBB.2022.3178446

[19] Kazutaka Katoh, Kazuharu Misawa, Kei-ichi Kuma, and Takashi Miyata. 2002. MAFFT: a novel method for rapid multiple sequence alignment based on fast Fourier transform. *Nucleic acids research* 30 14 (2002), 3059–66. https://api.semanticscholar.org/CorpusID:10960997

[20] Duhyeong Kim, Yongha Son, Dongwoo Kim, Andrey Kim, Seungwan Hong, and Jung Hee Cheon. 2019. Privacy-preserving Approximate GWAS computation based on Homomorphic Encryption. *IACR Cryptol. ePrint Arch.* (2019), 152. https://eprint.iacr.org/2019/152

[21] Vladimir Kolesnikov and Thomas Schneider. 2008. Improved Garbled Circuit: Free XOR Gates and Applications, Vol. 5126. Springer, 486–498.

[22] Tsung-Ting Kuo, Hyeon-Eui Kim, and Lucila Ohno-Machado. 2017. Blockchain distributed ledger technologies for biomedical and health care applications. *J. Am. Medical Informatics Assoc.* 24, 6 (2017), 1211–1220. https://doi.org/10.1093/

jamia/ocx068
[23] Tsung-Ting Kuo, Hugo Zavaleta Rojas, and Lucila Ohno-Machado. 2019. Comparison of blockchain platforms: a systematic review and healthcare examples. *J. Am. Medical Informatics Assoc.* 26, 5 (2019), 462–478. https://doi.org/10.1093/jamia/ocy185
[24] Sujun Li, Nuno Bandeira, XiaoFeng Wang, and Haixu Tang. 2016. On the Privacy Risks of Sharing Clinical Proteomics Data. In *Summit on Clinical Research Informatics, CRI 2016, San Francisco, CA, USA, March 21-24, 2016*. AMIA. http://knowledge.amia.org/amia-59309-cri2016-1.3011827/t003-1.3012832/f003-1.3012833/a024-1.3012920/a025-1.3012915
[25] Zhen Lin, ART B. Wen, and Russ B. Altman. 2004. Genomic Research and Human Subject Privacy. *Science* 305 (07 2004), 183.
[26] Christoph Lippert, Riccardo Sabatini, M Cyrus Maher, Eun Yong Kang, and J Craig Venter. 2017. Identification of individuals by trait prediction using whole-genome sequencing data. *Proc Natl Acad Sci USA* 114, 38 (2017), 10166.
[27] Hai Liu, Changgen Peng, Zhenqiang Wu, Youliang Tian, and Feng Tian. 2021. A survey of the Theories and Methods of Privacy Preserving of Genome Data. *Chinese Journal of Computers* 44, 7 (2021), 51. http://cjc.ict.ac.cn/online/bfpub/lh-2021127101719.pdf
[28] Huan Liu, Quan Zou, and Yun Xu. 2022. A novel fast multiple nucleotide sequence alignment method based on FM-index. *Briefings Bioinform.* 23, 1 (2022). https://doi.org/10.1093/bib/bbab519
[29] Tanaka M, Cabrera VM, González AM, and et al. 2004. Mitochondrial genome variation in eastern Asia and the peopling of Japan. In *Genome Res*, Vol. 14. PubMed, 1832–50. https://doi.org/10.1101/gr.2286304
[30] Bradley A. Malin and Latanya Sweeney. 2000. Determining the identifiability of DNA database entries. In *AMIA 2000, American Medical Informatics Association Annual Symposium, Los Angeles, CA, USA, November 4-8, 2000*. AMIA. https://knowledge.amia.org/amia-55142-a2000a-1.606968/t-001-1.609408/f-001-1.609409/a-108-1.609653/a-109-1.609650
[31] Bradley A. Malin and Latanya Sweeney. 2002. Inferring Genotype from Clinical Phenotype through a Knowledge Based Algorithm. In *Proceedings of the 7th Pacific Symposium on Biocomputing, PSB 2002, Lihue, Hawaii, USA, January 3-7, 2002*, Russ B. Altman, A. Keith Dunker, Lawrence Hunter, and Teri E. Klein (Eds.). 41–52. http://psb.stanford.edu/psb-online/proceedings/psb02/malin.pdf
[32] Bradley A. Malin and Latanya Sweeney. 2004. How (not) to protect genomic data privacy in a distributed network: using trail re-identification to evaluate and design anonymity protection systems. *J. Biomed. Informatics* 37, 3 (2004), 179–192. https://doi.org/10.1016/j.jbi.2004.04.005
[33] Elliott H. Margulies, Christina W. Chen, and Eric D. Green. 2006. Differences between pair-wise and multi-sequence alignment methods affect vertebrate genome comparisons. *Trends in Genetics* 22, 4 (2006), 187–193. https://doi.org/10.1016/j.tig.2006.02.005
[34] Yoshiki Nakagawa, Satsuya Ohata, and Kana Shimizu. 2022. Efficient privacy-preserving variable-length substring match for genome sequence. *Algorithms Mol. Biol.* 17, 1 (2022), 9. https://doi.org/10.1186/s13015-022-00211-1
[35] Pascal Paillier. 1999. Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In *Advances in Cryptology - EUROCRYPT '99, International Conference on the Theory and Application of Cryptographic Techniques, Prague, Czech Republic, May 2-6, 1999, Proceeding (Lecture Notes in Computer Science, Vol. 1592)*, Jacques Stern (Ed.). Springer, 223–238. https://doi.org/10.1007/3-540-48910-X_16
[36] Anna Poon, Steve Jankly, and Tingting Chen. 2018. Privacy Preserving Fisher's Exact Test on Genomic Data. In *IEEE International Conference on Big Data (IEEE BigData 2018), Seattle, WA, USA, December 10-13, 2018*, Naoki Abe, Huan Liu, Calton Pu, Xiaohua Hu, Nesreen K. Ahmed, Mu Qiao, Yang Song, Donald Kossmann, Bing Liu, Kisung Lee, Jiliang Tang, Jingrui He, and Jeffrey S. Saltz (Eds.). IEEE, 2546–2553. https://doi.org/10.1109/BigData.2018.8622575
[37] Michael Rabin, O. 2005. How To Exchange Secrets with Oblivious Transfer. *IACR Cryptology ePrint Archive* 2005 (01 2005), 187.
[38] Eric Schadt, Sangsoon Woo, and Ke Hao. 2012. Bayesian method to predict individual SNP genotype from gene expression data. *Nature genetics* 44 (04 2012), 603–8. https://doi.org/10.1038/ng.2248
[39] Thomas Schneider and Oleksandr Tkachenko. 2021. EPISODE: Efficient Privacy-PreservIng Similar Sequence Queries on Outsourced Genomic DatabasEs. *IACR Cryptol. ePrint Arch.* (2021), 29. https://eprint.iacr.org/2021/029
[40] Liyan Shen, Xiaojun Chen, Dakui Wang, Binxing Fang, and Ye Dong. 2018. Efficient and Private Set Intersection of Human Genomes. In *IEEE International Conference on Bioinformatics and Biomedicine, BIBM 2018, Madrid, Spain, December 3-6, 2018*, Huiru Jane Zheng, Zoraida Callejas, David Griol, Haiying Wang, Xiaohua Hu, Harald H. H. W. Schmidt, Jan Baumbach, Julie Dickerson, and Le Zhang (Eds.). IEEE Computer Society, 761–764. https://doi.org/10.1109/BIBM.2018.8621291
[41] Kana Shimizu, Koji Nuida, and Gunnar Rätsch. 2016. Efficient privacy-preserving string search and an application in genomics. *Bioinform.* 32, 11 (2016), 1652–1661. https://doi.org/10.1093/bioinformatics/btw050
[42] Nora von Thenen, Erman Ayday, and A. Ercüment Çiçek. 2019. Re-identification of individuals in genomic data-sharing beacons via allele inference. *Bioinform.* 35, 3 (2019), 365–371. https://doi.org/10.1093/bioinformatics/bty643
[43] Justin Wagner, Joseph N. Paulson, Xiao Wang, Bobby Bhattacharjee, and Héctor Corrada Bravo. 2016. Privacy-preserving microbiome analysis using secure computation. *Bioinform.* 32, 12 (2016), 1873–1879. https://doi.org/10.1093/bioinformatics/btw073
[44] Xiao Shaun Wang, Yan Huang, Yongan Zhao, Haixu Tang, XiaoFeng Wang, and Diyue Bu. 2015. Efficient Genome-Wide, Privacy-Preserving Similar Patient Query based on Private Edit Distance. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, October 12-16, 2015*, Indrajit Ray, Ninghui Li, and Christopher Kruegel (Eds.). ACM, 492–503. https://doi.org/10.1145/2810103.2813725
[45] Yue Wang, Jia Wen, Xintao Wu, and Xinghua Shi. 2016. Infringement of Individual Privacy via Mining Differentially Private GWAS Statistics. In *Big Data Computing and Communications - Second International Conference, BigCom 2016, Shenyang, China, July 29-31, 2016. Proceedings (Lecture Notes in Computer Science, Vol. 9784)*, Yu Wang, Ge Yu, Yanyong Zhang, Zhu Han, and Guoren Wang (Eds.). Springer, 355–366. https://doi.org/10.1007/978-3-319-42553-5_30
[46] Andrew Chi-Chih Yao. 1986. How to Generate and Exchange Secrets (Extended Abstract). In *27th Annual Symposium on Foundations of Computer Science, Toronto, Canada, 27-29 October 1986*. IEEE Computer Society, 162–167. https://doi.org/10.1109/SFCS.1986.25
[47] Hongru Yu, Haiyang Sun, Danyi Wu, and Tsung-Ting Kuo. 2019. Comparison of Smart Contract Blockchains for Healthcare Applications. In *AMIA 2019, American Medical Informatics Association Annual Symposium, Washington, DC, USA, November 16-20, 2019*. AMIA. https://knowledge.amia.org/69862-amia-1.4570936/t005-1.4574828/t005-1.4574829/3203516-1.4574836/3202134-1.4574833
[48] Samee Zahur and David Evans. 2015. Obliv-C: A Language for Extensible Data-Oblivious Computation. *IACR Cryptol. ePrint Arch.* (2015), 1153. http://eprint.iacr.org/2015/1153
[49] Lu Zhang, Qiuping Pan, Yue Wang, Xintao Wu, and Xinghua Shi. 2019. Bayesian Network Construction and Genotype-Phenotype Inference Using GWAS Statistics. *IEEE ACM Trans. Comput. Biol. Bioinform.* 16, 2 (2019), 475–489. https://doi.org/10.1109/TCBB.2017.2779498
[50] Pinglu Zhang, Huan Liu, Yanming Wei, Yixiao Zhai, Qinzhong Tian, and Quan Zou. 2024. FMAlign2: a novel fast multiple nucleotide sequence alignment method for ultralong datasets. *Bioinformatics* 40, 1 (01 2024), btae014. https://doi.org/10.1093/bioinformatics/btae014 arXiv:https://academic.oup.com/bioinformatics/article-pdf/40/1/btae014/56416850/btae014_supplementary_data.pdf
[51] Yongqing Zhang, Qiang Zhang, Jiliu Zhou, and Quan Zou. 2022. A survey on the algorithm and development of multiple sequence alignment. *Brief Bioinform* 23, 3 (2022), PMID: 35272347.
[52] Xiao-yong Zhou, Bo Peng, Yong Fuga Li, Yangyi Chen, Haixu Tang, and XiaoFeng Wang. 2011. To Release or Not to Release: Evaluating Information Leaks in Aggregate Human-Genome Data. In *Computer Security - ESORICS 2011 - 16th European Symposium on Research in Computer Security, Leuven, Belgium, September 12-14, 2011. Proceedings (Lecture Notes in Computer Science, Vol. 6879)*, Vijay Atluri and Claudia Díaz (Eds.). Springer, 607–627. https://doi.org/10.1007/978-3-642-23822-2_33