SECRET: Secure Edit-distance Computation over homomoRphic Encrypted daTa

Yuchen Zhang
Department of Electronic Engineering
Shanghai Jiao Tong University
Shanghai 200240, China
slyzzhangyc@sjtu.edu.cn

Wenrui Dai
Department of Electronic Engineering
Shanghai Jiao Tong University
Shanghai 200240, China
daiwenrui@sjtu.edu.cn

Shuang Wang
Department of Biomedical Informatics
University of California, San Diego
San Diego, CA, 92122
shw070@ucsd.edu

Miran Kim
Department of Mathematics
Seoul National University
Seoul, 92122, Republic of Korea
alfks500@snu.ac.kr

Kristin Lauter
Microsoft Research
San Diego, CA, 92122
klauter@microsoft.com

Jun Sakuma
Department of Computer science
University of Tsukuba
Tsukuba, Ibaraki 305-8577, Japan
jun@cs.tsukuba.ac.jp

Hongkai Xiong
Department of Electronic Engineering
Shanghai Jiao Tong University
Shanghai 200240, China
xionghongkai@sjtu.edu.cn

Xiaoqian Jiang
Department of Biomedical Informatics
University of California, San Diego
San Diego, CA, 92122
x1jiang@ucsd.edu

ABSTRACT

The biomedical community benefits from the increasing availability of genomic data, which enables institutions and researchers to develop personalized treatment and discover precision medicine. However, privacy and confidentiality of genomic data have been becoming a major concern for both patients and researchers in the data storage, transfer and analysis phases. In this paper, we proposed protocols for Secure Edit-distance Computation over homomoRphic Encrypted daTa (SECRET), which leverages Homomorphic Encryption (HME) to securely outsource both genomic data and edit-distance computation in an untrusted cloud environment without sacrificing data privacy. The proposed SECRET protocols develop both binary-vector HME comparison primitives and improved integer based counterparts for exact edit distance computation. Furthermore, we improved the efficiency of secure edit distance calculation by using an optimized pathfinding based approach. Parallel implementation is also investigated to improve the performance in concurrently calculating multiple sequence pairs. Experimental results demonstrate that the computational costs of the proposed protocol are significantly reduced in comparison to the existing state-of-the-art HME-based method.

Categories and Subject Descriptors

J.3 [**Computer Applications**]: LIFE AND MEDICAL SCIENCES, Biology and genetics

General Terms

Algorithms

Keywords

Homomorphic encryption (HME), edit distance, privacy protection, genomic data.

1. INTRODUCTION

Biomedical science is moving towards data-driven methodologies [1], which rely on collecting, integrating, and analyzing big data. The required storage and computational capacity of a biomedical study can easily overwhelm the availability in a single institution. Cloud computing, which relies on the sharing of resources to achieve coherence and economies of scale [2], provides an elastic way to conduct biomedical research in the era of big data. Researcher can rely on a cloud environment to speed up discovery or outsource storage and computationally demanding tasks to reduce cost.

While cloud computing has many benefits, it also comes with a larger risk of information leakage when compared to the traditionally (local) computing environment. There are various methods for secure cloud storage and computation but customized methods still need to be developed to meet the need of biomedical researchers. In this paper, we focus on edit distance calculation, which plays an important role in genomic data comparison. We use a semi-honest adversary model [3], in which the cloud faithfully follow the protocol (no tampering) but may try to infer additional information from the received data.

Data security and privacy are major concerns of biomedical researchers who would like to leverage cloud computing (e.g., Amazon EC2) to reduce the cost and improve computational scalability. Biomedical data are extremely vulnerable to information leakage and recent studies have indicated critical privacy risks of exposing human genome data to the public [4]–[6]. On the other hand, data protection mechanisms have advanced significantly in the last decade, which shed light to enable privacy-preserving data analysis in the cloud [7], [8]. The NIH is also working on a revised policy to permit dbGAP data analysis in the cloud [9]. Thanks to these exciting changes, it is the right time to develop practical secure and privacy-preserving biomedical data storage, sharing and analysis protocols that can be readily deployed to facilitate ordinary biomedical researchers in conducting data analysis tasks.

In this paper, we proposed a Secure Edit-distance Computation over homomoRphic Encrypted daTa (SECRET). The main contributions of this paper are two-fold.

- We developed the SECRET protocol to securely calculate exact edit distances between genome sequences. We designed binary-vector based HME comparison and addition primitives to achieve Wagner-Fischer edit distance algorithm over encrypted genome sequences.
- We improved the efficiency of SECRET protocol with the pathfinding based optimization and integer based HME comparison primitives to minimize the number of homomorphic operations. The improved SECRET protocol can reduce the circuit depth and computational costs with accurate edit distance outputs.

The rest of this paper is organized as follows. In section II, we review the related work. Section III presents the details of the proposed SECRET protocol, including the binary-vector based HME comparison and addition primitives and the corresponding HME based edit distance calculation algorithm. Section IV improves the SECRET protocol by adopting pathfinding based optimization, integer based HME comparison, and parallel computation. Experimental results and discussion are presented in Section V. Finally, we draw our conclusions in Section VI.

2. RELATED WORK

Secure multiparty computation (SMC) supports multiple parties to jointly compute a function over their inputs without revealing their private inputs. A number of predictive models, using Support Vector Machine, Logistic Regression, etc., have been developed based on this framework [10]–[12]. Blanton et al. [13] proposed an efficient sequence comparisons algorithm using garbled circuits. By using additive secrete sharing and additively homomorphic encryption, Rane et al. [14] developed a privacy preserving string comparison algorithm based edit distance. Sasakawa et al. [15] studied oblivious evaluation based DNA sequence comparison. However, many SMC-based approaches rely on significant multi-party communication (in the worst case, peer-to-peer), which is not easily supported in the healthcare IT. Another solution is to use secure co-processors (e.g., IBM 4758) on the servers to perform sensitive local computing and interaction with other nodes [16]. Despite the cost (more than $2,000 each), it is not feasible to push big data storage functionalities to a secure co-processor due to the complexity of specialized programming. Gentry [17] developed the first practical fully homomorphic encryption scheme, which makes it possible to support storage and computation on encrypted data in an untrusted cloud environment. Recently, Brakerski et al. [18], [19] improved the HME design by using the idea of "learning with errors" (LWE). In [20], Lauter et al. presented the implementation of a few statistical algorithms in genetic association studies using homomorphic encryption. Ayday et al. [21] demonstrated the use of additively homomorphic encryption to protect the data privacy for predicting disease susceptibility. Besides, Togan et al. [22] studied the integer comparison problem over fully homomorphic encrypted data. Recently, Graepel et al. [23] and Lauter et al. [24] also showed that certain machine learning algorithms can be implemented using HME. A similar work of HME based edit distance calculation is described in [25], which proposed a protocol to calculate the approximate edit distance by finding the minimum of values derived from shortest paths. Since their greedy algorithm only considered a few possible paths, the results provide the upper bound of the exact edit distance. In contrast, the proposed protocol in this paper includes all the possible paths and provides an accurate output. Unlike the protocol in [25] which is based on binary-vector comparison, the proposed protocol is improved by using integer based comparisons, which requires less circuit depth and number of homormorphic multiplications.

3. METHODS

3.1 Homomorphic Encryption

Homomorphic encryption [24] is a form of encryption, in which a specific algebraic operation performed on the plaintext is equivalent to another algebraic operation performed on the ciphertext, and when decrypted, matches the results of the same operation performed on the plaintext. Figure 1 illustrates the difference between homomorphic encryption and traditional encryption methods.

In specific, there are three types of homomorphic encryption techniques [26]: (1) partially homomorphic encryption that is specialized in a single type of operation (either addition or multiplication) [27], (2) fully homomorphic encryption that operations on both operations but less efficient [19], [28], and (3) leveled homomorphic encryption that operates on both *operations* for a limited number of iterations [18]. Partially homomorphic encryption techniques like unpadded RSA: $\mathrm{E}(x) = x^{\mathrm{e}} \bmod m$ (modulus m and exponent e as the public key) are very efficient: $\mathrm{E}(x_1)\mathrm{E}(x_2) = x_1^e\, x_2^e \bmod m = (x_1 x_2)^{\mathrm{e}} \bmod m = \mathrm{E}(x_1 x_2)$, but they cannot be combined with another operation like

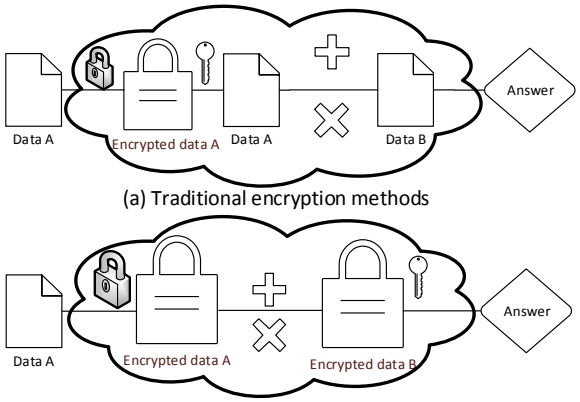

Figure 1. Illustration of homomorphic encryption vs. traditional encryption.

addition. Since more complex primitives involve multiple operations, partially homomorphic encryption has limitations. A possible mitigation is to conduct a single operation at a time with additional encryption and decryption steps using a trusted server but the data network is not fully protected. A fully homomorphic encryption system can operate on both operations without limitation, which is more powerful than the partially homomorphic schemes. However, a practical solution to implement the fully homomorphic encryption system has been elusive for more than a few decades. Despite some recent progresses, the efficiency of fully homomorphic encryption system is still too low to be considered a practical system. Leveled homomorphic encryption technique, which requires a pre-specification of the number of operations, is a compromise to the above extremes and offers reasonable efficiency. In the paper, we will first develop two basic primitives (i.e., HME comparison and addition based on binary vectors), with which we will introduce the proposed HME edit-distance computation protocol.

3.2 Edit-Distance Computation

Edit distance is a way to quantify the similarity between two sequences by counting the minimum number of operations (e.g., insertion, deletion and substitution) to transfer one sequence to the other.

Consider two genome sequences $\boldsymbol{S} = s^n, s^{n-1}, \dots, s^1$ and $\boldsymbol{T} = t^n, t^{n-1}, \dots, t^1$, where n is the number of bases in each genome sequence, respectively. The proposed algorithm can be generalized to the case of sequences with different lengths. However, for the sake of simplicity, we assume both sequences have the same length in the rest of this paper. For arbitrary $1 \leq u, v \leq n$, s^u and t^v take their values in $\{A, T, G, C\}$. To compute the edit distance between sequences $\boldsymbol{S}$ and $\boldsymbol{T}$, we define the costs of insertion, deletion and substitution as

$$\mathrm{INS}(v) = \mathrm{DEL}(u) = 1 \tag{1}$$

and

$$\mathrm{SUB}(u, v) = \begin{cases} 1, & if\ s^u \neq t^v \\ 0, & otherwise \end{cases} \tag{2}$$

Then, the problem of calculating the edit distance can be solved by dynamic program with the complexity of $O(n^2)$.

First, let us start with $D(0,0) = 0$ and define $D(u,0) = u$ and $D(0,v) = v$ for $1 \leq u, v \leq n$. Then, the edit distance $D(u,v)$ between subsequences $s^u, s^{u-1}, \dots, s^1$ and $t^v, t^{v-1}, \dots, t^1$ can be derived recursively as follows.

$$D(u,v) = \min \begin{cases} D(u-1, v) + \mathrm{DEL}(u), \\ D(u, v-1) + \mathrm{INS}(v), \\ D(u-1, v-1) + \mathrm{SUB}(u, v) \end{cases} \tag{3}$$

As a result, $D(n,n)$ is the edit distance between the two sequences $\boldsymbol{S}$ and $\boldsymbol{T}$ with length n. We can find that, in each step, the evaluation of edit distance only involves the minimization operation among three integer numbers, one comparison between two bases (i.e., s^u and t^v), and three addition operations between two integers. Therefore, the secure edit distance calculation is feasible, if we can develop the corresponding HME based comparison and addition primitives.

3.3 Binary Comparison and Addition

In this subsection, we will first introduce the binary HME comparison and addition primitives proposed for accurate edit distance calculation in SECRET, where an improved integer based comparison protocol will be presented later in subsection 4.2. To simplify the notation, we denote $\widehat{\boldsymbol{a}}_B = \hat{a}_B \hat{a}_{B-1} \dots \hat{a}_1$ in bold as the HME counterpart in B-bit binary vector representation of an integer a and $\hat{f}(\widehat{\boldsymbol{a}}_B)$ a secure function evaluated over the homomorphic encrypted binary vector $\widehat{\boldsymbol{a}}_B$. Moreover, we suppose the addition between encrypted bits are over the modular two (i.e., $\hat{1} + \hat{1} = \hat{0}$).

As discussed in the previous subsection, we need to develop HME comparison primitives to allow $\widehat{\mathrm{SUB}}_{bin}(\hat{\boldsymbol{s}}_B^u, \hat{\boldsymbol{t}}_B^v) = \hat{1} + \widehat{\mathrm{EQU}}_{bin}(\hat{\boldsymbol{s}}_B^u, \hat{\boldsymbol{t}}_B^v)$ and $\widehat{\mathrm{MIN}}_{bin}(\hat{\boldsymbol{x}}_B, \hat{\boldsymbol{y}}_B)$, where $\hat{\boldsymbol{s}}_B^u$ and $\hat{\boldsymbol{t}}_B^v$ are the binary-vector representation of the DNA bases, such as $A = 00$, $G = 01$, $T = 10$ and $C = 11$. Here, $\widehat{\mathrm{EQU}}_{bin}(\hat{\boldsymbol{s}}_B^u, \hat{\boldsymbol{t}}_B^v)$ takes two binary encoded DNA base vectors as inputs, then it outputs 1 if $\boldsymbol{s}_B^u == \boldsymbol{t}_B^v$ or 0 for other conditions. Moreover, $\widehat{\mathrm{MIN}}_{bin}(\hat{\boldsymbol{x}}_B, \hat{\boldsymbol{y}}_B)$ returns the encrypted minimum value $\hat{\boldsymbol{z}}_{\boldsymbol{b}}$ (in binary-vector format $\hat{\boldsymbol{z}}_{\boldsymbol{b}} = \hat{z}_B \hat{z}_{B-1}, \dots, \hat{z}_1$) between the two encrypted binary vector inputs $\hat{\boldsymbol{x}}_B$ and $\hat{\boldsymbol{y}}_B$.

Table 1. Truth table for the condition of $\boldsymbol{x_i} > \boldsymbol{y_i}$ and $\boldsymbol{x_i} == \boldsymbol{y_i}$ as well as their equivalent transform (ET) using only additional and multiplication.

$\hat{x}_i$	$\hat{y}_i$	$\boldsymbol{x_i} > \boldsymbol{y_i}$	ET: $\hat{x}_i + \hat{x}_i\hat{y}_i$	$\boldsymbol{x_i} == \boldsymbol{y_i}$	ET: $\hat{x}_i + \hat{y}_i + \hat{1}$
$\hat{0}$	$\hat{0}$	$\hat{0}$	$\hat{0} + \hat{0} * \hat{0} = \hat{0}$	$\hat{1}$	$\hat{0} + \hat{0} + \hat{1} = \hat{1}$
$\hat{0}$	$\hat{1}$	$\hat{0}$	$\hat{0} + \hat{0} * \hat{1} = \hat{0}$	$\hat{0}$	$\hat{0} + \hat{1} + \hat{1} = \hat{0}$
$\hat{1}$	$\hat{0}$	$\hat{1}$	$\hat{1} + \hat{1} * \hat{0} = \hat{1}$	$\hat{0}$	$\hat{1} + \hat{0} + \hat{1} = \hat{0}$
$\hat{1}$	$\hat{1}$	$\hat{0}$	$\hat{1} + \hat{1} * \hat{1} = \hat{0}$	$\hat{1}$	$\hat{1} + \hat{1} + \hat{1} = \hat{1}$

For example, as shown in Table 1, a 'BT' (bigger than) condition or 'EQU' (equal) condition of two encrypted binary variables $\hat{x}_i$ and $\hat{y}_i$ can be expressed by its equivalent transform as $\widehat{\mathrm{BT}}_{bin}(\hat{x}_i, \hat{y}_i) = \hat{x}_i\hat{y}_i + \hat{x}_i$ and $\widehat{\mathrm{EQU}}_{bin}(\hat{x}_i, \hat{y}_i) = \hat{x}_i + \hat{y}_i + \hat{1}$, respectively. Then, the bit-level operations could be generalized for handling operations with integer. For example, let's denote by $\hat{\boldsymbol{x}}_{\boldsymbol{b}} = \hat{x}_2\hat{x}_1$ and $\hat{\boldsymbol{y}}_{\boldsymbol{b}} = \hat{y}_2\hat{y}_1$ two encrypted 2-bits integers. To evaluate the condition $\boldsymbol{x_B} > \boldsymbol{y_B}$, we can decompose the operations as

$$\begin{aligned} \boldsymbol{x_B} > \boldsymbol{y_B} &\Leftrightarrow x_2x_1 > y_2y_1 \\ &\Leftrightarrow (x_2 > y_2) OR \big((x_2 == y_2)\ AND\ (x_1 > y_1)\big) \end{aligned} \tag{4}$$

Based on the equivalent transforms in Table 1, we can obtain the equivalent transform of $x_2x_1 > y_2y_1$ as $(x_2 + x_2y_2) + (x_2 + y_2 + 1)(x_1 + x_1y_1)$. Similarly, one can build equivalent transform for different conditions (e.g., $\boldsymbol{x_B} < \boldsymbol{y_B}$, etc.) and different number of bits B. Togan et al. [22] proposed a divide-and-conquer algorithm to efficiently compute multiple-bits integer. In our implementation, we follow the same paradigms in implementing $\widehat{\mathrm{EQU}}_{bin}(\hat{\boldsymbol{s}}_B^i, \hat{\boldsymbol{t}}_B^i)$ and $\widehat{\mathrm{BT}}_{bin}(\hat{\boldsymbol{x}}_B, \hat{\boldsymbol{y}}_B)$.

Given the HME primitive $\widehat{\mathrm{BT}}_{bin}(\hat{\boldsymbol{x}}_{\boldsymbol{B}}, \hat{\boldsymbol{y}}_{\boldsymbol{B}})$, $\hat{\boldsymbol{z}}_{\boldsymbol{B}} = \widehat{MIN}_{bin}(\hat{\boldsymbol{x}}_{\boldsymbol{B}}, \hat{\boldsymbol{y}}_{\boldsymbol{B}})$ can be evaluated as follows

HME min primitive: $\hat{\boldsymbol{z}}_{\boldsymbol{B}} = \widehat{MIN}_{bin}(\hat{\boldsymbol{x}}_{\boldsymbol{B}}, \hat{\boldsymbol{y}}_{\boldsymbol{B}})$

1: Given two input HME binary vectors $\hat{\boldsymbol{x}}_{\boldsymbol{B}}, \hat{\boldsymbol{y}}_{\boldsymbol{B}}$,
2: For $i = 1, 2, \dots, B$
3: Update $\hat{z}_i = \widehat{\mathrm{BT}}_{bin}(\hat{\boldsymbol{x}}_{\boldsymbol{B}}, \hat{\boldsymbol{y}}_{\boldsymbol{B}})\ (\hat{x}_i + \hat{y}_i) + \hat{x}_i$
4: End for
5: Return output $\hat{\boldsymbol{z}}_{\boldsymbol{B}} = \hat{z}_B \hat{z}_{B-1}, \dots, \hat{z}_1$

As the aforementioned HME comparison is based on binary vector, it requires that the succeeding addition operations between any numbers must also be performed in the binary vector representation. To implement binary-vector based HME addition, let's define $\hat{\boldsymbol{z}}_B = \hat{z}_B \hat{z}_{B-1}, \dots, \hat{z}_1$ is the corresponding encrypted output of $\widehat{ADD}_{bin}(\hat{\boldsymbol{x}}_{\boldsymbol{B}}, \hat{\boldsymbol{y}}_{\boldsymbol{B}})$ represented by a B-bit binary vector. $\hat{\boldsymbol{c}}_B = \hat{c}_B \hat{c}_{B-1}, \dots, \hat{c}_1$ stores the carrying values in the HME addition with $\hat{c}_1 = 0$. The proposed HME addition primitive can be implemented as follows:

HME addition primitive: $\hat{\boldsymbol{z}}_{\boldsymbol{B}} = \widehat{ADD}_{bin}(\hat{\boldsymbol{x}}_{\boldsymbol{B}}, \hat{\boldsymbol{y}}_{\boldsymbol{B}})$

1: Given two input HME binary vectors $\hat{\boldsymbol{x}}_{\boldsymbol{B}}, \hat{\boldsymbol{y}}_{\boldsymbol{B}}$,
2: For $i = 1, 2, \dots, B$
3: Update $\hat{z}_i = \hat{x}_i + \hat{y}_i + \hat{c}_{i-1}$
4: $\hat{c}_i = (\hat{x}_i + \hat{y}_i)\hat{c}_{i-1} + \hat{x}_i\hat{y}_i$
5: End for
6: Return output $\hat{\boldsymbol{z}}_{\boldsymbol{b}} = \hat{z}_B \hat{z}_{B-1}, \dots, \hat{z}_1$

Table 2 presents an example of the proposed HME addition primitive for two 3-bit encrypted binary vectors.

Table 2. Example of HME addition, where $\widehat{x}_3 = \widehat{x}_3\widehat{x}_2\widehat{x}_1$ (i.e., $\widehat{1}_3 = \widehat{0}\widehat{0}\widehat{1}$) and $\widehat{y}_3 = \widehat{y}_3\widehat{y}_2\widehat{y}_1$ (i.e., $\widehat{3}_3 = \widehat{0}\widehat{1}\widehat{1}$) are the encrypted 3-bit binary vector representations of integers 1 and 3 . $\widehat{z}_3 = \widehat{z}_3\widehat{z}_2\widehat{z}_1 = \widehat{ADD}_{bin}\widehat{z}_3\widehat{z}_2\widehat{z}_1 = \widehat{ADD}_{bin}\widehat{z}_3\widehat{z}_2\widehat{z}_1 = \widehat{ADD}_{bin}(\widehat{x}_3, \widehat{y}_3)$ (i.e., $\widehat{1}\widehat{0}\widehat{0}$) is the corresponding encrypted binary output. $\hat{c}_3 = \hat{c}_3\hat{c}_2\hat{c}_1$ denotes the encrypted intermediate carrying values in the HME addition operations.

Bit index	$\widehat{x}_B$	$\widehat{y}_B$	Carrying $\hat{c}_i = (\hat{x}_i + \hat{y}_i)\hat{c}_{i-1} + \hat{x}_i\hat{y}_i$	Result $\hat{z}_i = \hat{x}_i + \hat{y}_i + \hat{c}_{i-1}$
1	1	1	0	0
2	0	1	1	0
3	0	0	1	1

3.4 Binary-vector based HME Edit-Distance Calculation

In this subsection, we present the binary-vector based secure edit-distance calculation algorithm. According to Wagner-Fisher edit distance algorithm in equation (3), SECRET recursively calculates the edit distance based on homomorphic comparison and addition.

The proposed protocol starts with $\widehat{D}(0,0) = \widehat{\mathbf{0}}_B$ and define the costs of accumulated deletion $\widehat{D}(u,0) = \widehat{\boldsymbol{u}}_B$ for $1 \le u \le n$ and the costs of accumulated insertion $\widehat{D}(0,v) = \widehat{\boldsymbol{v}}_B$ for $1 \le v \le n$ in B-bit binary vector representation. Subsequently, each $\widehat{D}(i,j)$ can be recursively calculated based on Equation (3). Furthermore, the binary-vector based HME edit distance calculation $\widehat{D}(i,j)$ can be simplified into the following two steps given two sequences S and T:

$$\widehat{D}_{tmp}(u,v) = \widehat{\mathrm{ADD}}\left(\widehat{\mathrm{MIN}}\left(\widehat{D}(u-1,v), \widehat{D}(u,v-1)\right), \widehat{\mathbf{1}}_B\right) \tag{5}$$

and

$$\widehat{D}(u,v) = \widehat{\mathrm{MIN}}\left(\widehat{D}_{tmp}(u,v), \widehat{\mathrm{ADD}}\left(\widehat{D}(u-1,v-1), \widehat{\mathrm{SUB}}(\hat{\boldsymbol{s}}_B^u, \hat{\boldsymbol{t}}_B^v)\right)\right) \tag{6}$$

Algorithm 1 describes the SECRET protocol based on binary-vector comparison and addition.

Algorithm 1: SECRET protocol with binary-vector comparison and addition

0: **Inputs:** encrypted start point $\widehat{D}(0,0) = \widehat{\mathbf{0}}_B$, encrypted costs of accumulated deletion $\widehat{D}(u,0) = \widehat{\boldsymbol{u}}_B$ and encrypted costs of accumulated insertion $\widehat{D}(0,v) = \widehat{\boldsymbol{v}}_B$ for $1 \le u, v \le n$

1: **For** each row $u = 1,2,\dots,n$

2: **For** each categorical group $v = 1,2,\dots,n$

3: $\widehat{D}_{tmp}(u,v) = \widehat{\mathrm{ADD}}\left(\widehat{\mathrm{MIN}}\left(\widehat{D}(u-1,v), \widehat{D}(u,v-1)\right), \widehat{\mathbf{1}}_B\right)$

7: $\widehat{D}(u,v) = \widehat{\mathrm{MIN}}\left(\widehat{D}_{tmp}(u,v), \widehat{\mathrm{ADD}}\left(\widehat{D}(u-1,v-1), \widehat{\mathrm{SUB}}(\hat{\boldsymbol{s}}_B^u, \hat{\boldsymbol{t}}_B^v)\right)\right)$

8: **end for**

9: **end for**

10: **Outputs**: $\widehat{D}(m,n)$

4. OPTIMIZATION OF ENCRYPTED EDIT DISTANCE COMPUTATION

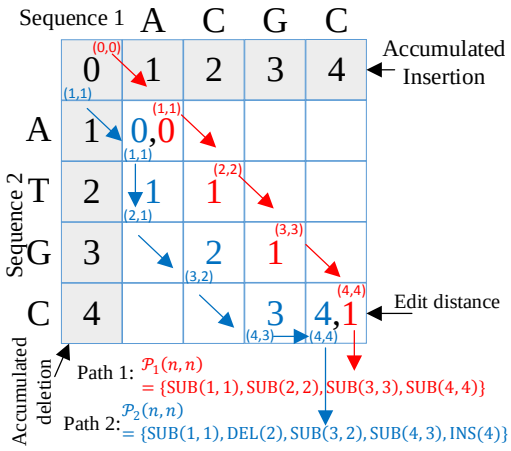

Figure 2. An example of pathfinding based edit distance computation, where we included 2 different paths in this example. The path in red color yields the minimum costs.

The method proposed in the previous section is computation intensive, as it need to evaluate all $\widehat{D}(u,v)$ for $0 \le u, v < n$ to obtain the encrypted edit distance $\widehat{D}(m,n)$. To effectively reduce the circuit depth, we propose an optimized method based on pathfinding and integer based HME comparison in this subsection.

4.1 Pathfinding based Optimization

The calculation of edit distance is equivalent to find a path from $(0,0)$ to (n,n) that achieves minimum accumulated cost for insertions, deletions and substitutions. Denote $\mathcal{P}_l(n,n) = \{\mathrm{OPE}_l^1(u_l^1, v_l^1), \mathrm{OPE}_l^2(u_l^2, v_l^2), \dots, \mathrm{OPE}_l^{m_l}(u_l^{m_l}, v_l^{m_l})\}$ as a set of operations in the l –th candidate path, where an operation can be $\mathrm{OPE}_l^i(u_l^i, v_l^i) \in \{\mathrm{SUB}(u_l^i, v_l^i), \mathrm{INS}(v_l^i), \mathrm{DEL}(u_l^i)\}$. Figure 2 shows an example of pathfinding based edit distance computation, where we included 2 different paths with $m_1 = 5$ and $m_2 = 6$ in this example. The path in red color yields the minimum costs. The transitions of indices and costs due to the substitution operation $\mathrm{SUB}(u_l^i, v_l^i)$ can defined by

$$\begin{aligned} u_l^i &= u_l^{i-1} + 1, \qquad v_l^i = v_l^{i-1} + 1 \\ D_l(u_l^i, v_l^i) &= D_l(u_l^{i-1}, v_l^{i-1}) + \mathrm{SUB}(u_l^i, v_l^i) \end{aligned} \tag{7}$$

Similarly, we define the insertion operation $\mathrm{INS}(v_l^i)$ as

$$\begin{aligned} u_l^i &= u_l^{i-1}, \qquad v_l^i = v_l^{i-1} + 1 \\ D_l(u_l^i, v_l^i) &= D_l(u_l^{i-1}, v_l^{i-1}) + 1 \end{aligned} \tag{8}$$

and deletion operation $\mathrm{DEL}(u_l^i)$ as

$$\begin{aligned} u_l^i &= u_l^{i-1} + 1, \qquad v_l^i = v_l^{i-1} \\ D_l(u_l^i, v_l^i) &= D_l(u_l^{i-1}, v_l^{i-1}) + 1 \end{aligned} \tag{9}$$

Then, a path $\mathcal{P}_l(n,n)$ can be further represented by a set of segments, where each segment ends at the position with $u_l^i = v_l^i$.

$$\mathcal{P}_l(m,n) = \mathcal{P}_l^0(\gamma^0,\sigma^0,k_l^0)\mathcal{P}_l^1(\gamma^1,\sigma^1,k_l^1)\cdots\mathcal{P}_l^r(\gamma^r,\sigma^r,k_l^r) \quad (10)$$

where $\mathcal{P}_l^j(\gamma_l^j,\sigma_l^j,k_l^j)$ is the j-th segment between $\left(u_l^{\gamma^j},v_l^{\gamma^j}\right)$ and $\left(u_l^{\gamma^{j+1}-1},v_l^{\gamma^{j+1}-1}\right)$ that contains k_l^j insertion and deletion operations, respectively. Here, r is the nuber of segments and γ_l^j is the first index of the coordinates $\left(u_l^{\gamma^j},v_l^{\gamma^j}\right)$ in the j-th segment. The parameter σ_l^j indicates the location of the segment $\mathcal{P}_l^j(\gamma_l^j,\sigma_l^j,k_l^j)$ in the matrix as shown in Figure 2, i.e., on diagonal for $\sigma_l^j=0$, lower triangular for $\sigma_l^j=1$, and upper triangular for $\sigma_l^j=-1$. For example, the blue path $\mathcal{P}_2(n,n)$ can be divided into two segments: $\mathcal{P}_2^1(\gamma_2^1=0,\sigma_1^2=0,k_2^1=0)=\{\mathrm{SUB}(1,1)\}$ on the diagonal and $\mathcal{P}_2^2(\gamma_2^2=1,\sigma_2^2=1,k_2^2=1)=\{\mathrm{DEL}(2),\mathrm{SUB}(3,2),\mathrm{SUB}(4,3),\mathrm{INS}(4)\}$ in the lower triangular with one insertion and deletion operation. Thus, we can derive the cost $D_l(n,n)$ for path $\mathcal{P}_l(n,n)$.

$$\begin{aligned} D_l(n,n) &= \sum_{j=1}^{r} D_l^j\left[\mathcal{P}_l^j(\gamma_l^j,\sigma_l^j,k_l^j)\right] \\ &= 2k_l + \mathrm{SUB}\left(u_l^{i_1},v_l^{i_1}\right)+\cdots+\mathrm{SUB}\left(u_l^{i_{n-k_l}},v_l^{i_{n-k_l}}\right) \end{aligned} \quad (11)$$

where $D_l^j\left[\mathcal{P}_l^j(\gamma_l^j,\sigma_l^j,k_l^j)\right]$ is the derived cost for segment $\mathcal{P}_l^j(\gamma_l^j,\sigma_l^j,k_l^j)$ and $k_l=\sum_{j=1}^r k_l^j$. In path $\mathcal{P}_l(n,n)$, we assume $n-k_l$ substitution operations $\mathrm{SUB}(\cdot,\cdot)$ are conducted at positions $\left(u_l^{i_1},v_l^{i_1}\right),\cdots,\left(u_l^{i_{n-k_l}},v_l^{i_{n-k_l}}\right)$. Therefore, the edit distance is the minimum of costs for all L_n candidate paths.

$$D(n,n) = \min_{1\le l\le L_n} D_l(n,n) \quad (12)$$

Comparing Equation (11) and (12), we can find each candidate path $\mathcal{P}_l$ is constrained by $\left|u_l^i-v_l^i\right|\le\lceil n/2\rceil-1, \forall 1\le i\le n, \forall l$ and $k_l\le\lceil n/2\rceil-1$, as $D(m,n)\le n$. Moreover, for arbitrary $\mathcal{P}_l^i(\gamma^i,\sigma^i,k_l^i)$, its cost should not exceed $u_l^{\gamma^{j+1}-1}-u_l^{\gamma^j}$, so we can obtain that

$$2k_l^i < u_l^{\gamma^{j+1}-1}-u_l^{\gamma^j} \quad (13)$$

Equation (13) is rooted from the fact that the edit distance of a block with the diagonal from $\left(u_l^{\gamma^j},v_l^{\gamma^j}\right)$ to $\left(u_l^{\gamma^{j+1}-1},v_l^{\gamma^{j+1}-1}\right)$ is not greater than $u_l^{\gamma^{j+1}-1}-u_l^{\gamma^j}$. Thus, Proposition 1 shows that we can obtain all the candidate paths for encrypted edit distance calculation based on a pruning process.

Proposition 1 For arbitrary block with the diagonal from $\left(u_l^{\gamma^j},v_l^{\gamma^j}\right)$ to $\left(u_l^{\gamma^{j+1}-1},v_l^{\gamma^{j+1}-1}\right)$, all the segments $\mathcal{P}_l^j(\gamma_l^j,\sigma_l^j,k_l^j)$ with $u_l^{\gamma^{j+1}-1}-u_l^{\gamma^j}\le 2k_l^i$ shall be excluded from $\mathcal{P}_l$.

Proof. The edit distance for the block is bounded by

$$\sum_{i=1}^{u_l^{\gamma^{j+1}-1}-u_l^{\gamma^j}} \mathrm{SUB}\left(u_l^{\gamma^j}+i,v_l^{\gamma^j}+i\right)\le u_l^{\gamma^{j+1}-1}-u_l^{\gamma^j}$$

This fact means that the segment $\mathcal{P}_l^i(\gamma,\sigma^i,k_l^i)$ with $u_l^{\gamma^{j+1}-1}-u_l^{\gamma^j}\le 2k_l^i$ violates the constraint (13). Thus, it cannot consist a candidate path passing from $\left(u_l^{\gamma^j},v_l^{\gamma^j}\right)$ to $\left(u_l^{\gamma^{j+1}-1},v_l^{\gamma^{j+1}-1}\right)$.

Proposition 1 excludes the paths that are impossible to achieve the minimum cost, which is easier to implement than directly traversing all the possible candidate paths. In practice, we can check all the blocks with diagonal lengths ranging from 3 (for $k_l=1$) to n. It is worth mentioning that the set of candidate paths can be generated on the fly, as it is only associated with the lengths of sequences S and T. This fact implies that the path-based optimization is able to reduce the circuit depth for encrypted edit distance calculation.

Denote $\Omega_n\left(\left\{\left(k_l^j,\sigma_l^j\right),k_l^j>0\right\}\right)$ as the number of paths containing segments $\{\mathcal{P}_l^j(\gamma_l^j,\sigma_l^j,k_l^j),k_l^j>0\}$ for $1\le j\le r$. According to Proposition 1, the number of candidate paths L_n can be obtained by summing up $\Omega_n\left(\left\{\left(k_l^j,\sigma_l^j\right)\right\},k_l^j>0\right)$ over all possible configurations for $\left\{\left(k_l^j,\sigma_l^j\right)\right\}$ with $\sum_j k_l^j=k_l$ and $k_l^j>0$. For each configuration of $\left\{\left(k_l^j,\sigma_l^j\right),k_l^j>0\right\}$, $\Omega_n\left(\left\{\left(k_l^j,\sigma_l^j\right),k_l^j>0\right\}\right)$ can be represented by the combination of $\Omega_{n'}\left(\left\{\left(k'^j_l,\sigma_l^j\right),k'^j_l>0\right\}\right)$ with $n'<n$ and $\sum_{j:k'^j_l>0}k'^j_l=\sum_{j:k_l^j>0}k_l^j-1$.

Table 3. The number of the candidate paths categorized by various cases for $n=8$

k_l	$\left\{\left(k_l^j,\sigma_l^j\right)\right\}$	# of candidate paths
0	$\{(0,0)\}$	1
1	$\{(1,1)\}$	$\Omega_7(\{(1,1)\})+(n-3+1)=21$
	$\{(1,-1)\}$	$\Omega_7(\{(1,-1)\})+(n-3+1)=21$
2	$\{(2,1)\}$	$\sum_{j=5}^{8}(n-j+1)\Omega_{j-1}(\{(1,1)\})=65$
	$\{(2,-1)\}$	$\sum_{j=5}^{8}(n-j+1)\Omega_{j-1}(\{(1,-1)\})=65$
	$\{(1,1),(1,-1)\}$	$\sum_{j=3}^{5}\Omega_j(\{(1,1)\})\Omega_{n-j}(\{(1,-1)\})=15$
	$\{(1,-1),(1,1)\}$	$\sum_{j=3}^{5}\Omega_j(\{(1,-1)\})\Omega_{n-j}(\{(1,1)\})=15$
	$\{(1,1),(1,1)\}$	$\Omega_3(\{(1,1)\})\Omega_{n-5}(\{(1,1)\})=1$
	$\{(1,-1),(1,-1)\}$	$\Omega_3(\{(1,-1)\})\Omega_{n-5}(\{(1,-1)\})=1$
3	$\{(3,1)\}$	$\sum_{j=7}^{8}(n-j+1)\Omega_{j-1}(\{(2,1)\})=55$
	$\{(3,-1)\}$	$\sum_{j=7}^{8}(n-j+1)\Omega_{j-1}(\{(2,-1)\})=55$
	$\{(2,1),(1,-1)\}$	$\binom{2}{1}\Omega_5(\{(2,1)\})\Omega_3(\{(1,-1)\})=6$
	$\{(2,-1),(1,1)\}$	$\binom{2}{1}\Omega_5(\{(2,-1)\})\Omega_3(\{(1,1)\})=6$
Total		327

We provide an example for calculating the number of paths when $n=8$. Table 3 shows the number of paths categorized by various $\left\{\left(k_l^j,\sigma_l^j\right),k_l^j>0\right\}$. There are 327 paths in total when considering all the configurations of $\left\{\left(k_l^j,\sigma_l^j\right),k_l^j>0\right\}$ for $\sum_j k_l^j=k_l$ and $k_l=0,1,2,3$. For each case, the number of paths with $\sigma_l^j=1$ for the first pair (k_l^j,σ_l^j) equals the one with $\sigma_l^j=-1$. It shows that numbers of paths with higher n and k_l can be recursively estimated based on those with lower n and k_l. Table 4 shows the results obtained for various configurations $\left\{\left(k_l^j,\sigma_l^j\right),k_l^j>0\right\}$ under $n=4,5,6,7,8$ and $k_l=0,1,2,3$. The numbers of candidate paths with various $\left\{\left(k_l^j,\sigma_l^j\right),k_l^j>0\right\}$ are calculated similarly to those in Table 3. It should be noted that the cases of $k_l=0$ and 1 are included in Table 3 and 4 as the starting points for recursion.

Table 4. The number of candidate paths for $n=4,\cdots,8$

n	4	5	6	7	8

k_l	$\{(k_l^j, \sigma_l^j)\}$	# of candidate paths				
0	{(0,0)}	1	1	1	1	1
1	{(1,1)}	3	6	10	15	21
	{(1,−1)}	3	6	10	15	21
2	{(2,1)}	0	3	12	31	65
	{(2,−1)}	0	3	12	31	65
	{(1,1),(1,−1)}	0	0	1	5	15
	{(1,−1),(1,1)}	0	0	0	0	1
	{(1,1),(1,1)}	0	0	0	0	1
	{(1,−1),(1,−1)}	0	0	1	5	15
3	{(3,1)}	0	0	0	12	55
	{(3,−1)}	0	0	0	12	55
	{(2,1),(1,−1)}	0	0	0	0	6
	{(2,−1),(1,1)}	0	0	0	0	6
Total		7	19	47	127	327

4.2 Integer Comparison

Given L_n candidate paths, the edit distance $D(n,n)$ is the minimum of values $D_l(n,n), 1 \le l \le L_n$ derived from these paths. To determine the value of $D(n,n)$, we need to find the minimal d in $[0,\mathrm{n}]$ that guarantees there exist at least one path satisfying $D_l(n,n) = d$. When the sequences S and T are encrypted, it is equivalent to verify whether the product of differences between $\widehat{D}_l$ and $\hat{d}$ for $\forall l \in \{1,2,\cdots,L_n\}$ is zero. Let us denote $\hat{v}_l^{(d)} = \widehat{D}_l(n,n) - \hat{d}$ the difference for l-th path and $\hat{v}^{(d)} = \prod_{1\le l\le L_n} \hat{v}_l^{(d)}$ the product of differences for all the L_n paths. Consequently, we design the function $\hat{\mathcal{E}}(\hat{v}^{(d)})$ to verify whether $\hat{v}^{(d)}$ is zero over the encrypted data.

$$\hat{\mathcal{E}}(\hat{v}^{(d)}) = \begin{cases} \hat{1}, & \text{if } \hat{v}^{(d)} = \hat{0} \\ \hat{0}, & \text{if } \hat{v}^{(d)} \neq \hat{0} \end{cases} \tag{14}$$

Here, the encrypted output $\hat{1}$ can be used to indicate that there exists at least one path $\mathcal{P}_l\big((0,0)\,(n,n)\big)$ satisfying $D_l(n,n) = d$. Equation (14) can be realized through the proposed secure integer comparison protocol discussed below.

From the Wilson's Theorem, it always holds for a prime number p greater than 2 that

$$(p-1)! \equiv -1 \pmod{p} \tag{15}$$

Thus, we can realize the secure comparison function $\hat{\mathcal{E}}(\hat{v}^{(d)})$ to test its equality with arbitrary encrypted integer inputs for $v^{(d)} \in [0, p-1]$.

$$\hat{\mathcal{E}}(\hat{v}^{(d)}) \equiv -\prod_{\ell=1}^{p-1}\big(\hat{\ell} - \hat{v}^{(d)}\big) \pmod{q} \tag{16}$$

where the prime number p is the plaintext modulus and q is the ciphertext modulus in the form of a product of primes under double-CRT representation in the BGV scheme [17]. It should be noted that $v_l^{(d)}$ will always be mapped into the interval $[0, p-1]$ based on the modulus p, in which a negative $v_l^{(d)}$ will be encoded by its complement integer $p - v_l^{(d)}$.

Equation (16) can be verified in the plaintext domain. Given $v^{(d)} = 0$, $\mathcal{E}(v^{(d)}) = -(p-1)! \equiv 1 \pmod{p}$. While the integer $v^d \in [1, p-1]$, one of the terms $(\ell - v^{(d)})$ will yield a zero for $\ell = 1,2,\ldots,p-1$, which means the product of these terms will also be zero. Therefore, Equation (14) can be realized by Equation (16) in the ciphertext domain.

Since the evaluation of Equation (16) requires $p-1$ times homomorphic multiplications (HMs), it could be computation intensive for homomorphic encrypted data, especially when p is a large number. Considering that p is typically greater than n, we can choose p as the minimal prime number greater than n to reduce the number of HMs.

4.3 Encrypted Edit Distance Calculation

Algorithm 2 (A2) describes the SECRET protocol based on integer comparison and path-based optimization. Given the L_s candidate paths, Algorithm 2 obtain the edit distance by finding the minimum cost with the integer-based comparison. A2 line 2 computes the cost $\widehat{D}_l(n,n)$ for path $\mathcal{P}_l$ from $D(0,0) = 0$. Remarkably, we cannot compute $\widehat{\mathrm{SUB}}(u,v)$ directly using binary comparison rule, as $\hat{x} + \hat{y} + \hat{1}$ may be larger than 1 for $p > 2$. To efficiently implement substitution operation $\widehat{\mathrm{SUB}}(u,v)$ under plaintext base $p > 2$, we still encode the DNA bases s^u and t^v into two bits $s^u = s_1^u s_2^u$ and $t^v = t_1^v t_2^v$, respectively. Thus, $\widehat{\mathrm{SUB}}(u,v)$ can be achieved by

$$\begin{aligned}\widehat{\mathrm{SUB}}(u,v) = 1 - \big(1 - (\hat{s}_1^u - \hat{t}_1^v) \times (\hat{s}_1^u - \hat{t}_1^v)\big) \\ \times \big(1 - (\hat{s}_2^u - \hat{t}_2^v) \times (\hat{s}_2^u - \hat{t}_2^v)\big)\end{aligned} \tag{17}$$

The derived costs in A2 line 2 are compared with all the possible edit distance $d = 0,1,\cdots,n$ and the products $\hat{v}^{(d)}$ of these differences are obtained in A2 line 5. Subsequently, A2 line 6 verifies whether $\hat{v}^{(d)}$ is zero with the integer-based comparison function $\hat{\mathcal{E}}\big(\hat{v}^{(d)}\big)$ proposed in Equation (16). To simplify, we denote $\hat{f}^{(d)} = \hat{\mathcal{E}}\big(\hat{v}^{(d)}\big)$ as the encrypted flag of verification. Finally, the SECRET protocol outputs the set of flags $\hat{\boldsymbol{f}} = \{\hat{f}^{(d)}, d = 0,1,\cdots,n\}$.

Algorithm 2: SECRET protocol based on integer comparison and path-based optimization

0: **Inputs:** encrypted start point $\widehat{D}(0,0) = \hat{0}$, sequence length n, and candidate paths $\mathcal{P}_l(n,n)$, $l = 1,2,\cdots,L_n$

1: **For** each path $l = 1,2,\cdots,L_n$

2: Compute the cost $\widehat{D}_l(n,n)$ for path $\mathcal{P}_l$

$\widehat{D}_l(n,n) \leftarrow 2\hat{k}_l + \widehat{\mathrm{SUB}}\big(u_l^{i_1}, v_l^{i_1}\big) + \cdots + \widehat{\mathrm{SUB}}\left(u_l^{i_{n-k_l}}, v_l^{i_{n-k_l}}\right)$

3: **end for**

4: **For** possible edit distance $d = 0,1,\cdots,n$

5: Calculate the product of encrypted differences

$\hat{v}^{(d)} \leftarrow \prod_{l=1}^{L_n}(\widehat{D}_l^{(d)}(n,n) - \hat{d})$

6: Evaluate $\hat{f}^{(d)} \leftarrow \hat{\mathcal{E}}(\hat{v}^{(d)})$ according to Equation (16)

7: **end for**

8: **Outputs**: $\hat{\boldsymbol{f}} = \{\hat{f}^{(d)}, d = 0,1,\cdots,n\}$

As mentioned in Section 4.2, $\hat{f}^{(d)}$ would be $\hat{1}$, if there exists at least one path $\mathcal{P}_l$ with cost $D_l(n,n) = d$. When $\hat{f}^{(0)}, \hat{f}^{(1)}, \cdots, \hat{f}^{(n)}$ are decrypted, the minimum of $D_l(n,n)$ is exactly the smallest d

that makes $f^{(d)} = 1$. Thus, $\hat{f}$ can be utilized to determine the edit distance without leaking the information of genome sequences.

4.4 Parallel Computation with Multiple Slots

Since HME schemes with ciphertext space $\mathbb{Z}_q^{L_s}$ support single instruction multiple data (SIMD) with L_s slots, parallel computation can be performed to reduce the number of homomorphic multiplications (HMs) and homomorphic additions (HAs). Suppose $\hat{\boldsymbol{a}} = (\hat{a}_1, \hat{a}_2, \dots, \hat{a}_{L_s})$ and $\hat{\boldsymbol{b}} = (\hat{b}_1, \hat{b}_2, \dots, \hat{b}_{L_s})$ are two encrypted ciphertexts with L_s slots. SIMD is applicable to simultaneous computation of the addition $\hat{\boldsymbol{a}} + \hat{\boldsymbol{b}} = (\hat{a}_1 + \hat{b}_1, \hat{a}_2 + \hat{b}_2, \dots, \hat{a}_{L_s} + \hat{b}_{L_s})$ and multiplication $\hat{\boldsymbol{a}} \cdot \hat{\boldsymbol{b}} = (\hat{a}_1 \cdot \hat{b}_1, \hat{a}_2 \cdot \hat{b}_2, \dots, \hat{a}_{L_s} \cdot \hat{b}_{L_s})$. Considering a group of L_s pairs of genome sequences $\{(S_1, T_1), (S_2, T_2), \cdots, (S_{L_s}, T_{L_s})\}$ with $S_i = s_i^n, s_i^{n-1}, \dots, s_i^1$ and $T_i = t_i^m, t_i^{m-1}, \dots, t_i^1$. Let us denote $s^j = (s_1^j, s_2^j, \dots, s_{L_s}^j)$ the vector of j-th bases in $(S_1, S_2, \dots, S_{L_s})$. In order to get $\hat{s}^j$, we encrypt the vector $(s_1^j, s_2^j, \dots, s_{L_s}^j)$ into one ciphertext. In two ciphertexts, only two slots in the same position can operate with each other. We can obtain the edit distance for each sequence pair in parallel. The use case of the scenario is that data owner can securely contribute the DNA sequences with the same length for study using the same public key and the cloud can figure out the edit distances for all the sequences pairs at the same time.

5. RESULTS

In this section, we evaluate the proposed SECRET protocol using both the binary-vector comparison and addition primitives (SECRET-BIN) and the integer comparison and path-based optimization (SECRET-INT). Both the two versions of SECRET protocols were implemented in the HElib [29], which is one of the most efficient open-source HME libraries based on the LWE theory [18], [19]. The evaluations were made on an Ubuntu 14.04 server with Intel Xeon CPU E5-2687W @ 3.10GHz and 256 GB memory. In order to check the accuracy and efficiency of SECRET, pairs of sequences (S, T) with various lengths (n, n) were randomly generated for edit distance calculation.

5.1 Performance of SECRET-BIN

We implemented and tested the binary-vector based version of the proposed SECRET protocol with a security level $k = 80$. The plaintext base p is set to 2 for the binary case and $L_s = 3512$ slots are used for parallel computation. We used 4-bit binary vector to represent the costs $D(u, v)$ for $0 < u, v \leq 8$. For genome sequences, DNA bases are encoded as $A = 00$, $G = 01$, $T = 10$ and $C = 11$, respectively.

Table 5. Average execution time (second) for the proposed SECRET-BIN protocol based on binary-vector comparison and addition primitives. 3512 sequence pairs (S, T) with lengths $n = 4$ are randomly generated for test.

Each unit					Total
	1	2	3	4	
1	0.324	0.348	0.376	0.368	5.98
2	0.345	0.404	0.402	0.368	
3	0.378	0.416	0.382	0.361	
4	0.385	0.406	0.363	0.358	

HElib is able to evaluate all the slots in the ciphertext in parallel, where each ciphertext provides 3,512 slots to simultaneously compute the edit distance between 3,512 pairs of genome sequence. Table 5 provides the average execution time for calculating the exact edit distance of each pair of sequence with length 4. For each unit, It shows the average time cost for updating the value at the corresponding position. The averaged time cost for edit distance between each pair of sequences is 5.98 seconds, which is obtained by aggregating the averaged time from each unit.

5.2 Performance of SECRET-INT

Table 6 elaborates the experimental setups for the SECRET-INT protocol based on integer-based comparison and pathfinding based optimization. Here, the plaintext modulus p is set to the minimal prime number greater than n. Consequently, the number of levels L in modulus chain and the number of slots L_s for parallel computation also vary with n.

Table 6. Experimental setups for pairs of sequences (S, T) with various lengths n in the proposed SECRET-INT protocol, where p is plaintext base; r is lifting parameter for plaintext base; k specifies the security level; L is number of levels in modulus chain; c is the number of columns in key switching matrix; w is hamming distance; and L_s is number of slots for parallel computation. Moreover, the storage costs of key generation are also provided as reference.

(n, m)	p	r	k	L	c	w	L_s	Public key size	Private key size
(3,3)	5	1	80	9	2	64	390	82.5 MB	83.4MB
(4,4)				10			240	55.9 MB	56.9MB
(5,5)	7			11			180	92.4 MB	93.9MB
(6,6)				13			130	142.4 MB	144.3 MB
(7,7)	11			15			300	317.4 MB	319.8 MB
(8,8)				17			1436	509.7 MB	512.8 MB
(9,9)				19			792	452.4 MB	456.0 MB
(10,10)				21			838	802.1 MB	806.7 MB

For improved SECRET-INT protocol based on integer comparison and path-based optimization, candidate paths for given (n, n) can be generated on the fly or in advance. These paths were obtained by pruning candidate paths according to Proposition 1. In Table 7, the second row is the number of the paths for various sequence lengths. The depth depends on the number of paths and is shown in the last row.

Table 7. The number of candidate paths and the circuit depth for the sequence pairs (S, T) with different lengths (n, n)

(n, n)	(3,3)	(4,4)	(5,5)	(6,6)	(7,7)
# of paths	3	7	19	47	127
Depth of Hom. Enc.	7	8	10	11	13
(n, m)	(8,8)	(9,9)	(10,10)	(11,11)	(12,12)
# of paths	327	887	1719	6353	16937
Depth of Hom. Enc.	15	16	17	19	21

Similarly to SECRET-BIN, we can use HElib to compute the edit distances of many pairs of sequences in parallel with multiple slots. Table 8 provides the average execution time for the proposed SECRET-INT protocol using different parameters. The

first column is the length of the sequence pair. The second column is the key generation time cost. The third column is the encryption time cost. The last column is the average time by using multiple slots. For example, the averaged total time for the sequence pair with length (8,8) is 0.2998s with 1436 slots. In our experiments, the key generation used around twenty seconds and the encryption of initial variables took several seconds.

Table 8. Average execution time (second) for the proposed SECRET protocol using different parameters

(n, m)	Key generation	Encryption	Execution time	
			Total	Average
(3,3)	8.29956	1.0717	3.8163	0.00979
(4,4)	8.77634	1.29391	8.57664	0.03574
(5,5)	11.7857	1.80728	25.0378	0.1391
(6,6)	14.4993	2.29973	54.1867	0.4168
(7,7)	18.2703	3.38198	158.273	0.52758
(8,8)	20.34	5.00516	430.512	0.2998
(9,9)	31.2945	9.02025	2311.19	2.9182
(10,10)	37.8116	10.7296	6187.17	7.38326

We also compare our results with the method proposed in [25], which approximates the upper bound of edit distance calculation by evaluating a few possible paths in a greedy way. In contrast, the proposed SECRET-INT provides accurate edit distance calculation by taking all the candidate paths into account. Moreover, SECRET-INT can still improves the performance by using integer based comparison and addition, when compared with the method based on binary operations. Table 9 compares both circuit depth and time cost between Cheon's method in [25] and the proposed method.

Table 9. Performance comparison between [25] **and the proposed algorithm in terms of circuit depth and computational cost**

(n, m)	Circuit Depth		Average time cost (sec)	
	Proposed	Cheon et al. [25]	Proposed	Cheon et al. [25]
(3,3)	**7**	8	**0.00979**	0.0544
(4,4)	**8**	9	**0.03574**	0.1071
(5,5)	**10**	N/A	**0.1391**	N/A
(6,6)	**11**	16	**0.4168**	2.6555
(7,7)	**13**	N/A	**0.52758**	N/A
(8,8)	**15**	19	**0.2998**	25.4366
(9,9)	**16**	N/A	**2.9182**	N/A
(10,10)	**17**	N/A	**7.38326**	N/A

5.3 Performance Analysis

In HME, the time cost mostly depends on the number of HMs and the circuit depth. The circuit depth of the SECRET protocol is composed of three parts: the first part is intorduced when we calculate $\widehat{\mathrm{SUB}}(u, v)$. Using two binaries to encode DNA bases, we can use Equation (17) to implement $\widehat{\mathrm{SUB}}(u, v)$ with a depth of two under $p > 2$, rather than a varying depth related with p. We calculate all under $p > 2$, The second part is produced by multiplication of the paths $\prod_{1 \le l \le L_n} \hat{v}_l^{(d)}$. It depends on L_n. The second part of the depth is $\log_2 L_n$. The third part is produced when we use Equation (16) to make sure if $\hat{v}^{(d)}$ equals to $\hat{0}$. It depends on the plaintext modulus p. The third part of the depth is $\log_2 p$. The whole depth is the sum of all three parts: $2 + \log_2 L_n + \log_2 p$.

In Comparison to Cheon's method [25], the proposed protocol uses integer based comparison and addition to replace binary-vector comparison and addition. In binary-vector addition, the number of homormorphic multiplications depends on the length of the binary vector and integer addition increases the circuit depth. Our protocol calculates $\hat{v}^{(d)} = \prod_{1 \le l \le L_n} \hat{v}_l^{(d)}$ rather than makes comparison for all the candidate paths $\widehat{D}_l(n, n)$, which also avoided the deep circuit comparison operation. Our protocol is effective to solve the problem because we include the prior knowledge: $\widehat{D}_l(n, n) \in [0, \mathrm{n}]$, by which we only need to compare a set of paths to find the minimum distance.

6. CONCLUSION

In this paper, we proposed the SECRET protocol for Secure Edit-distance Computation over homomoRphic Encrypted daTa. In specific, we first developed two cryptographic primitives: HME based comparison and HME based addition for binary vectors. We also proposed a more effective protocol combining path-finding with integer comparison and addition. The proposed SECRET protocol demonstrated the feasibility of securely outsourcing genomic data and edit distance computation in an untrusted cloud.

7. ACKNOWLEDGMENTS

This work was funded in part by the NHGRI (K99HG008175), NLM (R00LM011392, R21LM012060), NHLBI (U54HL108460), NSFC (61425011, 61271218, and U1201255), and "Shu Guang" project (13SG13).

8. REFERENCES

[1] D. Howe, M. Costanzo, P. Fey, T. Gojobori, L. Hannick, W. Hide, D. P. Hill, R. Kania, M. Schaeffer, S. St Pierre, S. Twigger, O. White, and S. Y. Rhee, "Big data: The future of biocuration.," *Nature*, vol. 455, no. 7209, pp. 47–50, Sep. 2008.

[2] *The NIST Definition of Cloud Computing*. National Institute of Standards and Technology.

[3] C. Hazay and Y. Lindell, *Efficient Secure Two-Party Protocols*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010.

[4] M. Naveed, E. Ayday, E. W. Clayton, J. Fellay, C. A. Gunter, J.-P. Hubaux, B. A. Malin, and X. Wang, "Privacy and Security in the Genomic Era," May 2014.

[5] S. Sankararaman, G. Obozinski, M. I. Jordan, and E. Halperin, "Genomic privacy and limits of individual detection in a pool.," *Nat. Genet.*, vol. 41, no. 9, pp. 965–7, Sep. 2009.

[6] N. Homer, S. Szelinger, M. Redman, D. Duggan, W. Tembe, J. Muehling, J. V Pearson, D. A. Stephan, S. F. Nelson, and D. W. Craig, "Resolving individuals contributing trace amounts of DNA to highly complex mixtures using high-density SNP genotyping microarrays," *PLoS Genet.*, vol. 4, no. 8, p. e1000167, 2008.

[7] M. Zhou, R. Zhang, W. Xie, W. Qian, and A. Zhou, "Security and Privacy in Cloud Computing: A Survey," in *2010 Sixth International Conference on Semantics, Knowledge and Grids*, 2010, pp. 105–112.

[8] W. Wang, Y. Hu, and L. Chen, "Accelerating fully homomorphic encryption using GPU," in *IEEE Conference on High Performance Extreme Computing (HPEC)*, 2012, pp. 1–5.

[9] U. G. Thomas, "NIH Working on Revised Policy to Permit dbGAP Data Analysis in the Cloud," 2014. [Online]. Available: http://www.genomeweb.com/informatics/nih-working-revised-policy-permit-dbgap-data-analysis-cloud. [Accessed: 02-Sep-2014].

[10] Y. Wu, X. Jiang, J. Kim, and L. Ohno-Machado, "Grid Binary LOgistic REgression (GLORE): building shared models without sharing data.," *J. Am. Med. Inform. Assoc.*, vol. 2012, no. 5, pp. 758–64, Apr. 2012.

[11] W. Jiang, P. Li, S. Wang, Y. Wu, M. Xue, L. Ohno-Machado, and X. Jiang, "WebGLORE: a web service for Grid LOgistic REgression.," *Bioinformatics*, vol. 29, no. 24, pp. 3238–40, Dec. 2013.

[12] S. Wang, X. Jiang, Y. Wu, L. Cui, S. Cheng, and L. Ohno-Machado, "EXpectation Propagation LOgistic REgRession (EXPLORER): Distributed Privacy-Preserving Online Model Learning," *J. Biomed. Inform.*, vol. 46, no. 3, pp. 1–50, 2013.

[13] M. Blanton, M. J. Atallah, K. B. Frikken, and Q. Malluhi, "Secure and efficient outsourcing of sequence comparisons," in *Computer Security--ESORICS 2012*, Springer, 2012, pp. 505–522.

[14] S. Rane and W. Sun, "Privacy preserving string comparisons based on Levenshtein distance," in *Information Forensics and Security (WIFS), 2010 IEEE International Workshop on*, 2010, pp. 1–6.

[15] H. Sasakawa, H. Harada, D. duVerle, H. Arimura, K. Tsuda, and J. Sakuma, "Oblivious Evaluation of Non-deterministic Finite Automata with Application to Privacy-Preserving Virus Genome Detection," in *Proceedings of the 13th Workshop on Privacy in the Electronic Society*, 2014, pp. 21–30.

[16] K. Hamlen, M. Kantarcioglu, L. Khan, and B. Thuraisingham, "Security Issues for Cloud Computing," *Int. J. Inf. Secur. Priv.*, vol. 4, no. 2, pp. 36–48, 2010.

[17] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proceedings of the 41st annual ACM symposium on Symposium on theory of computing - STOC '09*, 2009, pp. 169–178.

[18] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, "(Leveled) fully homomorphic encryption without bootstrapping," in *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference on - ITCS '12*, 2012, vol. 111, no. 111, pp. 309–325.

[19] Z. Brakerski and V. Vaikuntanathan, "Efficient fully homomorphic encryption from (standard) LWE," *SIAM J. Comput.*, vol. 43, no. 2, pp. 831–871, 2011.

[20] K. Lauter, A. López-Alt, and M. Naehrig, "Private computation on encrypted genomic data," in *14th Privacy Enhancing Technologies Symposium, Workshop on Genome Privacy (GenoPri'14)*, 2014.

[21] E. Ayday, J. L. Raisaro, P. J. Mclaren, J. Fellay, and J. Hubaux, "Privacy-Preserving Computation of Disease Risk by Using Genomic , Clinical , and Environmental Data," in *Proceedings of USENIX Security Workshop on Health Information Technologies (HealthTech" 13)*, 2013.

[22] M. Togan and C. Plesca, "Comparison-based computations over fully homomorphic encrypted data," in *Communications (COMM), 2014 10th International Conference on*, 2014, pp. 1–6.

[23] T. Graepel, K. Lauter, and M. Naehrig, "ML confidential: Machine learning on encrypted data," in *Information Security and Cryptology--ICISC 2012*, Springer, 2013, pp. 1–21.

[24] M. Naehrig, K. Lauter, and V. Vaikuntanathan, "Can homomorphic encryption be practical?," in *Proceedings of the 3rd ACM workshop on Cloud computing security workshop - CCSW '11*, 2011, p. 113.

[25] J. H. Cheon, M. Kim, and K. Lauter, "Homomorphic Computation of Edit Distance," in *WAHC'15 - 3rd Workshop on Encrypted Computing and Applied Homomorphic Cryptography*.

[26] C. Fontaine and F. Galand, "A Survey of Homomorphic Encryption for Nonspecialists," *EURASIP J. Inf. Secur.*, pp. 1–15, 2007.

[27] K. Gjøsteen, "A New Security Proof for Damgard's ElGamal," in *Topics in Cryptology – CT-RSA*, 2006, pp. 150–158.

[28] C. Gentry, "A fully homomorphic encryption scheme," Stanford University, 2009.

[29] S. Halevi and V. Shoup, "https://github.com/shaih/HElib." [Online]. Available: https://github.com/shaih/HElib.