# Homomorphic Computation of Edit Distance

Jung Hee Cheon[1], Miran Kim[1], and Kristin Lauter[2]

[1] Seoul National University (SNU), Republic of Korea
{jhcheon,alfks500}@snu.ac.kr
[2] Microsoft Research
klauter@microsoft.com

**Abstract.** These days genomic sequence analysis provides a key way of understanding the biology of an organism. However, since these sequences contain much private information, it can be very dangerous to reveal any part of them. It is desirable to protect this sensitive information when performing sequence analysis in public. As a first step in this direction, we present a method to perform the edit distance algorithm on encrypted data to obtain an encrypted result. In our approach, the genomic data owner provides only the encrypted sequence, and the public commercial cloud can perform the sequence analysis without decryption. The result can be decrypted only by the data owner or designated representative holding the decryption key.
In this paper, we describe how to calculate edit distance on encrypted data with a somewhat homomorphic encryption scheme and analyze its performance. More precisely, given two encrypted sequences of lengths $n$ and $m$, we show that a somewhat homomorphic scheme of depth $\mathcal{O}((n+m)\log\log(n+m))$ can evaluate the edit distance algorithm in $\mathcal{O}(nm\log(n+m))$ homomorphic computations. In the case of $n = m$, the depth can be brought down to $\mathcal{O}(n)$ using our optimization technique. Finally, we present the estimated performance of the edit distance algorithm and verify it by implementing it for short DNA sequences.

**Keywords:** Edit distance, Homomorphic encryption, Arithmetic circuit.

## 1 Introduction

In bioinformatics, the term "Sequence Analysis" refers to the process of arranging DNA, RNA, or peptide sequences to understand their structures and features. Relationships between sequences are usually discovered by aligning them appropriately and identifying the most closely matching subsequences. In this paper, we focus on the well-known edit distance algorithm [25], which measures the dissimilarity of two strings. Calculating the edit distance between public reference strings and patients' DNA sequences can be used to solve the problem of approximate string matching. In practice, there are deployed services to compare DNA sequences. For example, the European Bioinformatics Institute (EBI) website [6] provides "Bic-SW Database Searches" where one can apply a sequence analysis algorithm to any two DNA sequences (e.g., Smith-Waterman algorithm).

**Privacy Threats from Exposing Genomic Data.** There are many projects to collect DNA information from participants in order to discover genomic sequences associated with disease susceptibility. The Personal Genome Project displays genotypic and phenotypic information in a public database [21] and the HapMap Project has developed a public repository of genome sequences [12], which means that genomic data has become publicly accessible. However, even anonymized genomic data can leak significant information about the participants (see for example [7, 9, 23]). In fact, in 2012, an artist created portrait sculptures from analyses of genetic material collected in public places [24]. From some samples, he could infer physical characteristics of strangers such as the gender, eye color, nose size and so on. Secondly, even if DNA sequences are not associated with explicit identifiers such as name, sex, date of birth, or address, one can recover such personal data using re-identification methods: genotype-phenotype inference [19], location-visit patterns [20], family structure [10], and dictionary attacks. Thus, DNA sequences are sensitive and valuable enough that we should not reveal our own sequences even when performing sequence analysis.

**Privacy through Encryption.** In this work, we consider the potential for using homomorphic encryption to protect privacy in genomic computations. Compared with MPC protocols based on recent optimizations of garbled circuit techniques [14, 11], homomorphic encryption is often considered to be slower and less efficient. But homomorphic encryption has a number of other advantages, allowing for more flexible scenarios and functionality and requiring less interaction, thereby reducing communication complexity. Typically no interaction is required for applications of (single-key) homomorphic encryption. Also, homomorphic encryption schemes have become more practical recently, due to a number of improvements, including techniques which avoid the costly bootstrapping procedure for fixed computations, such as using leveled or somewhat homomorphic encryption (SWHE) schemes.

**Scenarios.** Homomorphic encryption allows the data owner to upload encrypted data to a cloud service. The cloud service can operate on the encrypted data without requiring the decryption key or any interaction with the data owner. The service returns the encrypted results to the data owner, who can decrypt using the secret key. A cloud provider could thus provide Direct-to-patient services in encrypted form, such as the service mentioned above provided by EBI.

As an extension to the scenario, additional functionality can be achieved using public key homomorphic encryption schemes by allowing third parties to upload data directly to the cloud service, encrypted using the public key of the data owner. This scenario could be of interest in situations relevant to genomic computation: for example the data owner is a hospital or clinic, and

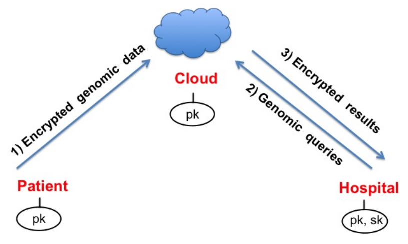

Fig. 1: Scenario of proposed system

the third parties are patients or other healthcare providers for those patients. The hospital would like to use the cloud service for analyzing lots of patients. Auxiliary data (from tests, genome sequencing, etc) can be uploaded to the service using the public key of the hospital. Computations on the encrypted data, such as comparing DNA sequences, output encrypted results which can be decrypted by the hospital or clinic. The secrecy of DNA sequences in the cloud can be protected under the semantic security of homomorphic encryption scheme.

**Our Contributions.** In this paper, we first describe the homomorphic evaluation of the edit distance algorithm which was suggested by Wagner and Fischer [25]. We show that the algorithm can be implemented on two encrypted sequences of lengths $n$ and $m$ with a somewhat homomorphic scheme of depth $\mathcal{O}((n+m)\log(\log(n+m)))$ in $\mathcal{O}(nm\log(n+m))$ homomorphic computations. Moreover, we introduce an optimization technique to reduce the depth required to implement the algorithm: Divide the edit distance matrix into sub-blocks of size-$(\tau+1)$ and solve the edit distance problem in each block. We can compute each of them diagonally, consuming $\mathcal{O}(\tau)$ levels in one diagonal-round. Namely, evaluating the circuits in each cell can be processed by a somewhat homomorphic encryption of a constant depth. In particular, in the case of $n=m$, it suffices to compute only a little part of the sub-blocks, so the depth can be brought down to $\mathcal{O}(n)$.

Finally, we estimate the running time of the proposed algorithm for a large $n$ and verify it by implementing it for short DNA sequences. For two encrypted DNA sequences of length 50, we expect that the algorithm would run in one day when estimated based on the recent CCK+ scheme [4]. We also demonstrate the experimental result that it takes about 27.5 seconds for $n=m=8$ using the GHS scheme [8].

**Related works.** Since Wagner and Fischer [25] introduced the problem of determining the edit distance between two strings and presented an algorithm for calculating the distance, there have been a number of approaches for private computation of the distance. In 2003, Atallah et al. [1] proposed a privacy-preserving protocol using an additive homomorphic encryption scheme and oblivious transfers, which had expensive computational and communication costs. Given two strings of lengths $n$ and $m$, the number of iterations is equal to $nm$ and the total online computational cost is $\mathcal{O}(nm\log(n+m))$. In 2008, Jha et al. [14] presented a more practical privacy-preserving protocol to compute the edit distance with Yao's "garbled circuits" method [18, 26], and it was improved by Huang et al. [11]. Their computation cost is tractable, but their protocol requires a lot of interactions (e.g., $\mathcal{O}(nm\log(n+m))$ oblivious transfers for Protocol 2 in [14]).

On the other hand, there is prior art on analyzing genomic data using homomorphic encryption. Some of the work is based on additively homomorphic encryption schemes: Kantarcioglu et al. [15], Kolesnikov et al. [16], and Ayday et al. [2]. In [15], they presented a novel cryptographic framework that allows organizations to support data mining without violating the privacy of the genomic sequences, and in particular they used the Paillier cryptosystem for experimental analysis. The garbled circuit protocols of [16] were given for secure computation of the minimum distance (Hamming distance and Euclidean distance). In [2], they proposed a "privacy-preserving disease susceptibility test" on encrypted genomic data using a modified Paillier cryptosystem. Meanwhile, Cristofaro et al. [5] presented an efficient and secure protocol called "Size- and position-hiding private substring matching" based on a multiplicative homomorphic ElGamal variant so as to check for the presence of DNA markers. Finally, Yasuda et al. [27] gave a practical solution for computation of multiple Hamming distance values using the LNV scheme [17], so that they could find the locations where a pattern occurs in a text. By contrast, the aim of this paper is to compute edit distance on encrypted sequences under somewhat homomorphic encryption schemes (which support additions and a limited number of multiplications of encrypted inputs). Besides DNA sequence analysis, edit distance has many other applications such as spelling correction or determining the longest common subsequences of two strings.

**Outline.** In Section 2, we review the main concept of homomorphic encryption and explain the edit distance algorithm. Section 3 presents the basic circuit building blocks for equality, comparison, and addition. Next, in Section 4, we describe our encrypted edit distance algorithm using these primitive circuits and give the analysis of our method. We also introduce optimizations to reduce the depth of implementing the algorithm. Finally, in Section 5, we estimate the performance of the proposed algorithm for large DNA sequences and present the real performance for our implementation of the algorithm for short sequences.

## 2 Preliminaries

In this section, we briefly review the concept of homomorphic encryption and describe the edit distance algorithm which is a measure to quantify the dissimilarity of two strings.

### 2.1 Homomorphic Encryption

We will encrypt bit-by-bit in this paper, so consider the concept of homomorphic encryption in this respect. For $x \in \{0,1\}$, we denote the encryption of $x$ by $\bar{x}$ or $\mathsf{Enc}(x)$. Let $\oplus$ and $\wedge$ be the XOR and AND gate, each of which corresponds to addition and multiplication over $\mathbb{Z}_2$, respectively. Also, we let $+$ and $\times$ denote homomorphic addition and multiplication over encrypted data. Then a homomorphic encryption $\mathsf{Enc}(\text{-})$ satisfies the following properties:

$$\mathsf{Enc}(x \oplus y) = \mathsf{Enc}(x) + \mathsf{Enc}(y), \quad \mathsf{Enc}(x \wedge y) = \mathsf{Enc}(x) \times \mathsf{Enc}(y).$$

In our paper, we focus on SWHE schemes for which additions are essentially free and a limited number of multiplications are supported. In particular, SWHE schemes [3, 8] use a practical noise-management technique-*modulus switching*,

**Algorithm** Edit distance

```
Input: $\alpha = \alpha_1 \dots \alpha_n$ and $\beta = \beta_1 \dots \beta_m$
1:  for $i \leftarrow$ 0 to $n$ do
2:      $D_{i,0} \leftarrow i$;
3:  end for
4:  for $j \leftarrow$ 0 to $m$ do
5:      $D_{0,j} \leftarrow j$;
6:  end for
7:  for $i \leftarrow$ 1 to $n$ do
8:      for $j \leftarrow$ 1 to $m$ do
9:          $t \leftarrow (\alpha_i = \beta_j)?\ 0 : 1$;
10:         $D_{i,j} \leftarrow \min\{D_{i-1,j-1} + t, D_{i,j-1} + 1, D_{i-1,j} + 1\}$;
11:     end for
12: end for
13. return $D_{n,m}$
```

which scales down the ciphertext after every multiplication to reduce the noise by its scaling factor. When we say the (multiplicative) *depth* $\mathbf{D}(\mathrm{C})$ of a circuit C under homomorphic encryption, it means the total number of reduced levels in the circuit that is being evaluated homomorphically.

## 2.2 Edit Distance

Assume that there are two strings $\alpha = \alpha_1 \dots \alpha_n$ and $\beta = \beta_1 \dots \beta_m$ over an alphabet $\Sigma$. One can make another string with the same length by inserting spaces "$-$", called *gaps*, and consider a matrix having two rows with these new strings. A gap in the first (resp. second) row is called Insertion (resp. Deletion). A column with the same (resp. distinct) characters is called Match (resp. Mismatch). Then the edit distance between two strings is the minimum number of these edit operations needed to transform one string into the other. More specifically, for two characters $\alpha_i$ and $\beta_j$, let us define $t_{i,j}$ as follows:

$$t_{i,j} = \begin{cases} 0 & \text{if } \alpha_i = \beta_j \text{ (Match)}, \\ 1 & \text{if } \alpha_i \neq \beta_j \text{ (Mismatch)}. \end{cases}$$

In Algorithm 1, we describe the *Wagner-Fischer edit distance algorithm* [25], and the edit distance is simply $D_{n,m}$.

## 3 Circuit Building Blocks

In this section, we present the basic circuit building blocks for computing the edit distance: equality circuit (for checking the equality of two numbers so as to determine match/mismatch of two characters), comparison circuit, and addition circuits. Since it may assume that we can evaluate homomorphic additions for free, it suffices to count the number of multiplication gates sequentially in order to compute the depth of a homomorphic encryption scheme. Thus, we focus on minimizing the number of sequential multiplication gates for circuits so that we can implement them efficiently.

For a circuit C, we denote the number of homomorphic additions and multiplications by $\mathbf{HA}(\mathrm{C})$ and $\mathbf{HM}(\mathrm{C})$. Note that addition with a constant is faster than a classical homomorphic addition, so those are not counted in the number of the homomorphic additions. In Table 1,2, and 4, the depth of homomorphic encryption is cumulative while the number of homomorphic computations is not cumulative.

We will express an unsigned $\mu$-bit integer in its binary representation $x_\mu \dots x_1$ and denote the $i$-th coordinate of $x$ by $x_i$ (or $x[i]$). Then the encryption of $x$ means $\{\bar{x}_1, \bar{x}_2, \dots, \bar{x}_\mu\}$.

## 3.1 Equality Circuit

A binary circuit for checking the equality of two $\mu$-bit values is defined to have value 1 if the inputs are the same and 0 otherwise. Then it can be written as an arithmetic circuit $\mathtt{EQU}(x,y) = \wedge_{i=1}^{\mu} (1 \oplus x_i \oplus y_i)$. Using a binary tree, we give the required depth and complexity in Table 3 where log is the binary logarithm.

## 3.2 Comparison Circuit

For two unsigned $\mu$-bit values $x$ and $y$, the comparison circuit is defined by

$$\mathtt{COM}(x,y) = \begin{cases} 0 & \text{if } x \geq y, \\ 1 & \text{otherwise,} \end{cases}$$

and this is written recursively as $\mathtt{COM}(x,y) := c_\mu$ where $c_i = ((x_i \oplus 1) \wedge y_i) \oplus ((x_i \oplus 1 \oplus y_i) \wedge c_{i-1})$ for $i \geq 2$ with an initial value $c_1 = (x_1 \oplus 1) \wedge y_1$. In Table 1, we provide a pseudocode description of this circuit together with an approximation of the levels that it consumes during these operations. Unlike the other steps, the fourth cannot be computed simultaneously for each $i$, so it consumes linear levels and we have $\mathbf{D}(\mathtt{COM}) = \mu$. On the other hand, the comparison circuit can be evaluated homomorphically with a logarithmic depth, which is formally captured in Lemma 1 below.

| **Comparison Circuit** | **Depth of Hom. Enc.** | **HA** | **HM** |
|---|---|---|---|
| **Input**: fresh ciphertexts $\bar{x}_i, \bar{y}_j$ | 0 | | |
| 1. compute $\bar{x}_i + 1$ for $i = 1, \dots, \mu$ | 0 | $-$ | $-$ |
| 2. $\bar{x}_{i1} \leftarrow (\bar{x}_i + 1) + \bar{y}_i$ for $i = 2, \dots, \mu$ | 0 | $\mu - 1$ | $-$ |
| 3. $\bar{x}_{i2} \leftarrow (\bar{x}_i + 1) \times \bar{y}_i$ for $i = 1, \dots, \mu$ (in particular, let $\bar{c}_1 \leftarrow \bar{x}_{12}$) | 1 | $-$ | $\mu$ |
| 4. $\bar{c}_i \leftarrow \bar{x}_{i1} + \bar{x}_{i2} \times \bar{c}_{i-1}$ for $i = 2, \dots, \mu$ | $\mu$ | $\mu - 1$ | $\mu - 1$ |
| Total | $\mu$ | $2\mu - 2$ | $2\mu - 1$ |

Table 1: Pseudocode of $\mathtt{COM}$ between two $\mu$-bit values and its complexity

**Lemma 1** *The Comparison circuit of Table 2 can be evaluated homomorphically on two $\mu$-bits with a somewhat homomorphic encryption of depth* $\log(\mu-1)+1$ *in* $\mathcal{O}(\mu\log\mu)$ *homomorphic computations.*

*Proof.* We consider the comparison circuit as the following expression:

$$\texttt{COM}(x,y) = d_1 \oplus d_2 \oplus \ldots \oplus d_\mu$$

where $d_i = (x_i \oplus 1) \wedge y_i \wedge (\wedge_{j=i+1}^{\mu}(x_j \oplus 1 \oplus y_j))$. From now, the following arguments are underlying ciphertexts for the above circuit. For simplicity, we denote $z_i := (\bar{x}_i + 1) + \bar{y}_i$ for $i = 2,\ldots,\mu$, and $\mathbf{HM}_i$ the number of homomorphic multiplications to evaluate $\prod_{j=i+1}^{\mu} z_j$ for $i = 1,\ldots,\mu-\ 2$.

We first construct a binary tree of product with $\{z_2,\ldots,z_\mu\}$. Then the total number of multiplications to proceed recursively with each of the two nodes is

$$1+2+4+\cdots+\frac{\mu-1}{2} \approx \mu-2,$$

and it needs $\log(\mu-1)$ levels. We observe that $\prod_{j=i+1}^{\mu} z_j$ has been computed if the number to be multiplied by is in the form of powers of 2 or $\mu-1$.

Now, we consider the case of $i \in \{1,2,\cdots,\mu-2\}$ with $\mu - i \neq 2^1, 2^2, \cdots, 2^{\lfloor \log(\mu-1)\rfloor}, \mu-1$. It is true that $\mu-i$ is uniquely written as $2^{k_{i_1}} + 2^{k_{i_2}} + \cdots + 2^{k_{i_l}}$ where $k_{i_j}$'s are increasing nonnegative numbers. Denote $\mu_{i_r} := \mu - \left(2^{k_{i_l}} + 2^{k_{i_{l-1}}} + \cdots + 2^{k_{i_{r+1}}} + 2^{k_{i_r}}\right)$ for $1 \le r \le l$ and $\mu_{i_{l+1}} = \mu$, then we have

$$\prod_{j=i+1}^{\mu} z_j = \prod_{r=1}^{l} (z_{\mu_{i_r}+1} z_{\mu_{i_r}+2} \cdots z_{\mu_{i_{r+1}}}).$$

Since all $z_{\mu_{i_r}+1} z_{\mu_{i_r}+2} \cdots z_{\mu_{i_{r+1}}}$'s have been computed as above, what we have to do is just to multiply them each other, which requires $\log l$ levels and $(l-1)$ homomorphic multiplications. From these observations, we see that

$$\sum_{2^{t-1}<u-i<2^t} \mathbf{HM}_i = \sum_{l=1}^{t-1} l \cdot \binom{t-1}{l} = (t-1)\cdot 2^{t-2}$$

for $t \in \{2,3,\ldots,\lfloor \log(\mu-\ 1)\rfloor\}$. So we have

$$\begin{aligned}
\sum_{i=1}^{\mu-2} \mathbf{HM}_i &= \sum_{u-i=2^1,2^2,\ldots,\mu-1} \mathbf{HM}_i + \sum_{t=2,3,\ldots,\lfloor \log(\mu-1)\rfloor} \left( \sum_{2^{t-1}<u-i<2^t} \mathbf{HM}_i \right) \\
&\approx (\mu-2) + \sum_{t=2,3,\ldots,\lfloor \log(\mu-1)\rfloor} (t-1)\cdot 2^{t-2} \\
&= \frac{(\mu-1)\log(\mu-1)}{2} - 2.
\end{aligned}$$

Table 2: Pseudocode of COM between two $\mu$-bit values and its complexity

| **Comparison Circuit** | **Depth of Hom. Enc.** | **HA** | **HM** |
|---|---|---|---|
| **Input**: fresh ciphertexts $\bar{x}_i, \bar{y}_j$ | $0$ | | |
| 1. compute $\bar{x}_i + 1$ for $1 \le i \le \mu$ | $0$ | $-$ | $-$ |
| 2. $\bar{d}_i \leftarrow (\bar{x}_i + 1) \times \bar{y}_i$ for $1 \le i \le \mu$ | $1$ | $-$ | $\mu$ |
| 3. $z_i \leftarrow (\bar{x}_i + 1) + \bar{y}_i$ for $2 \le i \le \mu$ | $0$ | $\mu - 1$ | $-$ |
| 4. $\prod_{j=i+1}^{\mu} z_j$ for $1 \le i \le \mu - 2$ | $\log(\mu - 1)$ | $-$ | $\frac{(\mu-1)\log(\mu-1)}{2} - 2$ |
| 5. $\bar{d}_i \leftarrow \bar{d}_i \times \prod_{j=i+1}^{\mu} z_j$ for $1 \le i \le \mu - 1$ | $\log(\mu - 1) + 1$ | $-$ | $\mu - 1$ |
| 6. $\overline{\texttt{COM}(x, y)} \leftarrow \bar{d}_1 + \cdots + \bar{d}_\mu$ | $-$ | $\mu - 1$ | $-$ |
| Total | $\log(\mu - 1) + 1$ | $2\mu - 2$ | $2\mu - 3 + \frac{(\mu-1)\log(\mu-1)}{2}$ |

Therefore, as described in Table 2, evaluating the COM circuit can be accomplished using

$$\mu + \left(\frac{(\mu - 1)\log(\mu - 1)}{2} - 2\right) + (\mu - 1) = 2\mu - 3 + \frac{(\mu - 1)\log(\mu - 1)}{2}$$

homomorphic multiplications with a SWHE scheme of depth $\log(\mu - 1) + 1$. □

In the following, we show that the comparison circuit leads to the the minimal circuits.

**Lemma 2** *Given two $\mu$-bit values $x = x_\mu \ldots x_1$ and $y = y_\mu \ldots y_1$, then $z = z_\mu \ldots z_1$ is the minimum value of $x$ and $y$ where*

$$z_i = (\texttt{COM}(x, y) \wedge x_i) \oplus (1 \oplus \texttt{COM}(x, y) \wedge y_i)\,.$$

*Proof.* Let us denote a multiplication over integers by "$\cdot$". Then it is true that

$$\begin{aligned} \min\{x, y\} &= \texttt{COM}(x, y) \cdot x + (1 \oplus \texttt{COM}(x, y)) \cdot y \\ &= \texttt{COM}(x, y) \cdot \Big(\sum_{i=1}^{\mu} x_i \cdot 2^{i-1}\Big) + (1 \oplus \texttt{COM}(x, y)) \cdot \Big(\sum_{i=1}^{\mu} y_i \cdot 2^{i-1}\Big) \\ &= \sum_{i=1}^{\mu} \Big( (\texttt{COM}(x, y) \cdot x_i) + ((1 \oplus \texttt{COM}(x, y)) \cdot y_i) \Big) \cdot 2^{i-1}, \end{aligned}$$

where the inputs $x$ and $y$ can be written as binary representations in the second line. Since "$\texttt{COM}(x, y) \cdot x_i$" and "$(1 \oplus \texttt{COM}(x, y)) \cdot y_i$" cannot simultaneously be "1", the lemma follows. □

From Lemma 2, we define minimum circuits $\texttt{MIN}^2 = (\texttt{MIN}^2_1, \ldots, \texttt{MIN}^2_\mu)$ by

$$\texttt{MIN}^2_i = (\texttt{COM}(x, y) \wedge x_i) \oplus ((1 \oplus \texttt{COM}(x, y)) \wedge y_i)\,.$$

Table 3: Complexity of primitive circuits between two $\mu$-bit values

| **Circuit** | **Depth of Hom. Enc.** | **HA** | **HM** |
|---|---|---|---|
| `EQU` | $\log\mu$ | $\mu$ | $\mu-1$ |
| `COM` | $\log(\mu-1)+1$ | $2\mu-2$ | $2\mu-3+\frac{(\mu-1)\log(\mu-1)}{2}$ |
| `ADD` | $\mu-1$ | $3\mu-3$ | $2\mu-3$ |

Then one can evaluate these circuits homomorphically with a SWHE scheme of depth $(\log(\mu-1)+2)$. We also obtain a natural generalization of computing the minimum value between many numbers: apply repeatedly the minimum circuits. Then this naive method has $\mathbf{D}(\texttt{MIN}^2) = (\log(\mu-1)+2)\cdot(\log k)$.

On the other hand, we consider another way to compute the minimum value which requires circuits of lesser depth: Given $\mu$-bit values $x^1,\dots,x^k$, we define $\texttt{MIN}^k = (\texttt{MIN}^k_1,\dots,\texttt{MIN}^k_\mu)$ by $\texttt{MIN}^k_i = \bigoplus_{j=1}^k \left(\mathfrak{c}_j \wedge x^j_i\right)$ where

$$\mathfrak{c}_j = \begin{cases} \texttt{COM}(x^1,x^2)\wedge\cdots\wedge\texttt{COM}(x^1,x^k) & \text{if } j=1,\\ (1\oplus\texttt{COM}(x^1,x^j))\wedge\cdots\wedge(1\oplus\texttt{COM}(x^{j-1},x^j))\wedge\texttt{COM}(x^j,x^{j+1})\wedge\cdots\wedge\texttt{COM}(x^j,x^k) & \text{if } 2\le j<k,\\ \left(1\oplus\texttt{COM}(x^1,x^k)\right)\wedge\cdots\wedge\left(1\oplus\texttt{COM}(x^{k-1},x^k)\right) & \text{if } j=k. \end{cases}$$

It is easy to show that this method has

$$\mathbf{D}(\texttt{MIN}^k) = \log(\mu-1)+\log(k-1)+2,$$
$$\mathbf{HM}(\texttt{MIN}^k) = \left(2\mu-3+\frac{(\mu-1)\log(\mu-1)}{2}\right)\frac{(k-1)(k-2)}{2}+k\,(k-2+\mu)\,.$$

### 3.3 Addition circuits

For two unsigned $\mu$-bit values $x$ and $y$, we assume that their sum over the integers is less than $2^\mu$, say $s_1+\cdots+s_\mu\cdot 2^{\mu-1}$. Then the standard method to add them is the *Ripple-carry* adder such that $\texttt{ADD}(x,y)$ is defined by $(s_1,\dots,s_\mu)$ satisfying

$$s_i = \begin{cases} x_1\oplus y_1 & \text{if } i=1,\\ x_i\oplus y_i\oplus e_{i-1} & \text{otherwise,}\end{cases}\quad e_i = \begin{cases} x_1\wedge y_1 & \text{if } i=1,\\ (x_i\wedge y_i)\oplus((x_i\oplus y_i)\wedge e_{i-1}) & \text{otherwise.}\end{cases}$$

From now, the $k$-th value $s_k$ of the sum is denoted by $\texttt{ADD}(x,y)[k]$. Table 3 reports the required depth and its complexity analysis.

## 4 Encrypted Edit Distance Algorithm

We now describe how to execute the homomorphic computation of the edit distance algorithm with regards to the primitive circuits and analyze the performance of our encrypted edit distance algorithm.

Let $|\Sigma|$ be the size of a alphabet and denote $\omega = \lceil \log |\Sigma| \rceil$. As mentioned before, let $\alpha$ and $\beta$ be two strings over $\omega$-bit alphabet. Then each character of the strings can be seen as an $\omega$-bit value. Suppose that each of them is given encrypted bit-by-bit through a homomorphic encryption.

### 4.1 Encrypted Edit Distance Algorithm

Since all the values $D_{i,j}$'s are less than $n+m-1$, we may assume that they are $\lceil \log(n+m-1) \rceil$-bits, say $\mu$. Suppose that we have computed $D_{i-1,j-1}, D_{i,j-1}, D_{i-1,j}$, and $\omega$-bit characters $\alpha_i$ and $\beta_j$. From the fact that $t_{i,j} = \mathtt{EQU}(\alpha_i, \beta_j) \oplus 1$, we know

$$\begin{aligned}(D_{i-1,j-1} + t_{i,j})[k] &= ((t_{i,j} \oplus 1) \wedge D_{i-1,j-1}[k]) \oplus (t_{i,j} \wedge \mathtt{ADD}(D_{i-1,j-1}, 1)[k]) \\ &= (\mathtt{EQU}(\alpha_i, \beta_j) \wedge D_{i-1,j-1}[k]) \oplus ((\mathtt{EQU}(\alpha_i, \beta_j) \oplus 1) \wedge \mathtt{ADD}(D_{i-1,j-1}, 1)[k])\end{aligned}$$

for $1 \le k \le \mu$ and

$$\mathtt{ADD}(D_{i-1,j-1}, 1)[k] = \begin{cases} D_{i-1,j-1}[1] \oplus 1 & \text{if } k = 1, \\ D_{i-1,j-1}[k] \oplus \left(\wedge_{l=1}^{k-1} D_{i-1,j-1}[l]\right) & \text{if } 2 \le k \le \mu. \end{cases}$$

In the same way as in Section 3.2, $\mathtt{ADD}(D_{i-1,j-1}, 1)$ can be implemented with a SWHE scheme of depth $\log(\mu - 1)$ in $\mu$ additions and $\left(\frac{(\mu-1)\log(\mu-1)}{2} - 2\right)$ multiplications since we only need to compute $\prod_{l=1}^{k-1} \mathsf{Enc}(D_{i-1,j-1}[l])$. From these observations, $D_{i,j} = \min\{D_{i-1,j-1} + t_{i,j}, D_{i,j-1} + 1, D_{i-1,j} + 1\}$ can be written as arithmetic circuits using the above circuits. Hence, given ciphertexts $\mathsf{Enc}(D_{i-1,j-1})$, $\mathsf{Enc}(D_{i,j-1})$, $\mathsf{Enc}(D_{i-1,j})$, $\mathsf{Enc}(\alpha_i)$, and $\mathsf{Enc}(\beta_j)$, one can apply these operations so as to compute the encryption of $D_{i,j}$. Continuing this way, we obtain the encrypted edit distance $\mathsf{Enc}(D_{n,m})$.

### 4.2 Performance Analysis of Encrypted Edit Distance Algorithm

In Table 4, we describe a pseudocode for obtaining the encrypted value $D_{i,j}$, and provide an approximation of the levels and computational complexity during homomorphic operations. By the building block algorithms of COM (in Lemma 1) and ADD (in Section 4.1), the one diagonal-round circuits have

$$\mathbf{D} = 2\log(\mu-1)+4,\ \mathbf{HA} = 15\mu+\omega-6,\ \mathbf{HM} = 3(\mu-1)\log(\mu-1)+11\mu+\omega-13.$$

It is possible to compute $D_{i,j}$'s simultaneously when $i+j$ is a fixed value from $1, 2, ..., (n+m-1)$, so we expect to consume $(2\log(\mu-1)+4)\cdot(n+m-1)$ levels for computing them diagonally, which requires $(15\mu+\omega-6)nm$ homomorphic additions and $(3(\mu-1)\log(\mu-1)+11\mu+\omega-13)nm$ multiplications in total. In other words, given two encrypted sequences of lengths $n$ and $m$, a SWHE scheme of depth $\mathcal{O}((n+m)\log(\log(n+m)))$ can evaluate the edit distance algorithm in $\mathcal{O}(nm\log(n+m))$ homomorphic computations.

Table 4: Pseudocode of computing the encrypted value $D_{i,j}$ and its complexity ($\mu = \log(n+m-1)$)

| Binary Circuit | Depth of Hom. Enc. | HA | HM |
|---|---|---|---|
| **Input**: $D_{i-1,j-1}, D_{i,j-1}, D_{i-1,j}, \alpha_i, \beta_j$ | | | |
| 1. $t \leftarrow \texttt{EQU}(\alpha_i, \beta_j)$ and compute $t \oplus 1$ | $\mathbf{D}(\texttt{EQU})$ | $\mathbf{HA}(\texttt{EQU})$ | $\mathbf{HM}(\texttt{EQU})$ |
| 2. compute $\texttt{ADD}(D_{i-1,j-1}, 1), \texttt{ADD}(D_{i,j-1}, 1), \texttt{ADD}(D_{i-1,j}, 1)$ | $\mathbf{D}(\texttt{ADD})$ | $3\mathbf{HA}(\texttt{ADD})$ | $3\mathbf{HM}(\texttt{ADD})$ |
| 3. for $k = 1, \dots, \mu$, | | | |
| $D_{i-1,j-1}[k] \leftarrow (t \wedge D_{i-1,j-1}[k]) \oplus ((t \oplus 1) \wedge \texttt{ADD}(D_{i-1,j-1}, 1)[k])$ | $1 + \mathbf{D}(\texttt{ADD})$ | $\mu$ | $2\mu$ |
| $D_{i,j-1}[k] \leftarrow \texttt{ADD}(D_{i,j-1}, 1)[k]$ | $\mathbf{D}(\texttt{ADD})$ | — | — |
| $D_{i-1,j}[k] \leftarrow \texttt{ADD}(D_{i-1,j}, 1)[k]$ | $\mathbf{D}(\texttt{ADD})$ | — | — |
| 4. $\mathfrak{c}_1 \leftarrow \texttt{COM}(D_{i-1,j-1}, D_{i,j-1})$ | $1 + \mathbf{D}(\texttt{ADD}) + \mathbf{D}(\texttt{COM})$ | $\mathbf{HA}(\texttt{COM})$ | $\mathbf{HM}(\texttt{COM})$ |
| $\mathfrak{c}_2 \leftarrow \texttt{COM}(D_{i-1,j-1}, D_{i-1,j})$ | $1 + \mathbf{D}(\texttt{ADD}) + \mathbf{D}(\texttt{COM})$ | $\mathbf{HA}(\texttt{COM})$ | $\mathbf{HM}(\texttt{COM})$ |
| $\mathfrak{c}_3 \leftarrow \texttt{COM}(D_{i,j-1}, D_{i-1,j})$ | $\mathbf{D}(\texttt{ADD}) + \mathbf{D}(\texttt{COM})$ | $\mathbf{HA}(\texttt{COM})$ | $\mathbf{HM}(\texttt{COM})$ |
| 5. $\mathfrak{c}_1 \leftarrow \mathfrak{c}_1 \wedge \mathfrak{c}_2, \mathfrak{c}_2 \leftarrow (1 \oplus \mathfrak{c}_1) \wedge \mathfrak{c}_3, \mathfrak{c}_3 \leftarrow (1 \oplus \mathfrak{c}_2) \wedge (1 \oplus \mathfrak{c}_3)$ | $2 + \mathbf{D}(\texttt{ADD}) + \mathbf{D}(\texttt{COM})$ | — | 3 |
| 6. for $k = 1, \dots, \mu$, $D_{i,j}[k] \leftarrow (\mathfrak{c}_1 \wedge D_{i-1,j-1}[k]) \oplus (\mathfrak{c}_2 \wedge D_{i,j-1}[k]) \oplus (\mathfrak{c}_3 \wedge D_{i-1,j}[k])$ | $3 + \mathbf{D}(\texttt{ADD}) + \mathbf{D}(\texttt{COM})$ | $2\mu$ | $3\mu$ |
| **Total** | $3 + \mathbf{D}(\texttt{ADD}) + \mathbf{D}(\texttt{COM})$ | $\mathbf{HA}(\texttt{EQU}) + 3\mathbf{HA}(\texttt{ADD})$ $+3\mathbf{HA}(\texttt{COM}) + 3\mu$ | $\mathbf{HM}(\texttt{EQU}) + 3\mathbf{HM}(\texttt{ADD})$ $+3\mathbf{HM}(\texttt{COM}) + 5\mu + 3$ |

**Remark 1** *Lemma 1 shows that we can compare two $\mu$-bits with a circuit of depth $\log \mu$ using a homomorphic bit-encryption scheme. If we consider a large integer ring $\mathbb{Z}_t$ as a message space instead of a binary field, an addition is performed with a degree-1 circuit. However, one can compute the equality circuit via the following method: $\texttt{EQU}(x, y) = 1 - (x - y)^{t-1}$ for a prime $t$. Then this circuit has $\mathbf{D}(\texttt{EQU}) \approx \log t \approx \log(n + m)$ using the square-and-multiply algorithm. Moreover, the comparison algorithm seems to require a circuit of at least depth $\log t$. This implies that a large message space increases the depth of one diagonal-round circuits to $\mathcal{O}(\log(n + m))$, so the edit distance algorithm can be evaluated with a SWHE scheme of depth $\mathcal{O}((n + m) \log(n + m))$.*

## 4.3 Optimization of Encrypted Edit Distance Algorithm

We present an optimization to reduce the depth during the homomorphic evaluations of the algorithm. Let us consider the $3 \times 3$ block $B$ in Figure 2.

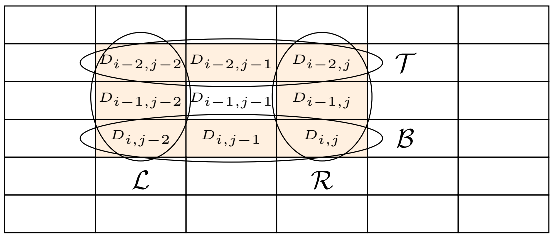

Fig. 2: Block of size 3

It is true that if we have computed the top and left values of this block, $D_{i-2,j-2}$, $D_{i-2,j-1}$, $D_{i-2,j}$, $D_{i-1,j-2}$, $D_{i,j-2}$, then all other values can be expressed in terms of them. For example, $D_{i,j}$ is the minimum value between the following 7 numbers:

$$D_{i-2,j-2}+t_{i-1,j-1}+t_{i,j},\ D_{i-2,j-1}+t_{i-1,j}+1,\ D_{i-2,j-1}+t_{i-1,j}+1,$$

$$D_{i-1,j-2}+t_{i,j-1}+1,\ D_{i-1,j-2}+t_{i,j-1}+1,\ D_{i-2,j}+2,\ D_{i,j-2}+2.$$

In general, we consider a block of size-$(\tau+1)$ which consists of the following sets:

$$\begin{aligned}\text{top}: \mathcal{T} &= \{D_{i-\tau,j-\tau}, D_{i-\tau,j-\tau+1}, \ldots, D_{i-\tau,j}\},\\ \text{left}: \mathcal{L} &= \{D_{i-\tau,j-\tau}, D_{i-\tau+1,j-\tau}, \ldots, D_{i,j-\tau}\},\\ \text{right}: \mathcal{R} &= \{D_{i-\tau,j}, D_{i-\tau+1,j}, \ldots, D_{i,j}\},\\ \text{bottom}: \mathcal{B} &= \{D_{i,j-\tau}, D_{i,j-\tau+1}, \ldots, D_{i,j}\}.\end{aligned}$$

Then all the values of $\mathcal{R}$ and $\mathcal{B}$ are expressed in terms of values of $\mathcal{T}$ and $\mathcal{L}$.

More precisely, consider the grid shown in Figure 3. One can only move one unit right or down on the grid: if moving right from $D_{i-k,j-l}$, then $t_{i-k+1,j-l+1}$ is added to the value and we obtain $D_{i-k,j-l}+t_{i-k+1,j-l+1}$. In the case of moving one unit down, "1" is added to it. We note that the number of shortest paths from $D_{i-\tau,j-k}$ to $D_{i-\tau+l,j}$ is $\frac{l!}{k!(l-k)!}=\binom{l}{k}$ for some $l \geq k$ since the paths include $k$ steps in the $x$ axis and $(l-k)$ steps in $y$ axis. It is seen as the the number of the functions of $D_{i-\tau+l,j}$ in terms of $D_{i-\tau,j-k}$. From these observations, $D_{i-\tau+l,j}$ is the minimum between $\sum_{k=0}^{l}\binom{l}{k}=2^l$ values. In particular, $D_{i,j}$ is the minimum between $2\cdot\sum_{k=0}^{\tau}\binom{\tau}{k}-\tau=2^{\tau+1}-\tau$ values because the set of all the paths of $D_{i,j}$ is symmetric with respect to the line from $D_{i-\tau,j-\tau}$ to $D_{i,j}$. We know that the minimum circuits consume the largest number of levels than others (equality circuit or addition circuits), and it needs $\mathcal{O}(\log k)$ levels to evaluate the minimum circuits $\texttt{MIN}^k$ that compute the minimum value between $k$ numbers, which requires $\mathcal{O}(k^2)$ homomorphic computations. Thus, one can compute a block of size-$(\tau+1)$ by evaluating the circuits with a SWHE of depth

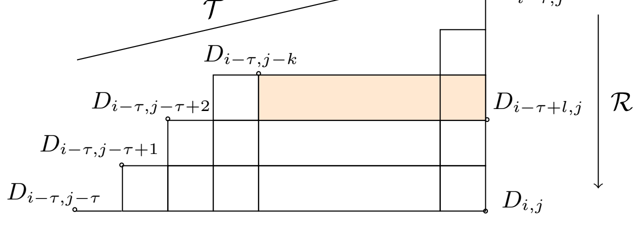

Fig. 3: Grid of size-$(\tau+1)$ block

$\mathcal{O}(\log(2^{\tau+1}-\tau)) \approx \mathcal{O}(\tau)$ in

$$\sum_{k=2,2^2,\ldots,2^{\tau-1}} \mathcal{O}(k^2) + \mathcal{O}((2^{\tau+1}-\tau)^2) \approx \mathcal{O}(2^{2\tau})$$

homomorphic operations. From the fact that all the blocks of size-$(\tau+1)$ can be computed diagonally while shares of the values of $\mathcal{T}$ and $\mathcal{L}$ have been computed, we can conclude that the edit distance algorithm can be implemented using $\mathcal{O}(2^{2\tau}\cdot\frac{nm}{\tau^2})$ homomorphic operations with a SWHE scheme of depth $\mathcal{O}(\tau\cdot(\frac{n+m}{\tau}-1)) \approx \mathcal{O}(n+m)$ for given two encrypted sequences of lengths $n$ and $m$. Hence, this optimization reduces the depth, but the entire computation increases as $\tau$ becomes larger. In particular, in the case of $n=m$, we can implement the algorithm with lesser depth circuits. The essence of the idea is formally captured in Lemma 3 below.

**Lemma 3** *Let $\sigma_j$ denote the elementary symmetric polynomial of degree $j$ in $x_1, x_2, \ldots, x_n$ and $\widetilde{\sigma}_j$ the binary circuit which is a conversion of $\sigma_j$ by the following rules: $+ \mapsto \oplus$ and $\cdot \mapsto \wedge$. Also, let $\mu := \lceil \log n \rceil$. Then the addition circuits $\mathtt{ADD}^n$ convert the sum of $n$ one-bit $x_i$'s into a $\mu$-bit integer, defined by $(S[1], S[2], \ldots, S[\mu])$ satisfying*

$$S[i] = \bigoplus_{1\le j\le n} \Big( \bigoplus_{\substack{1\le k\le j \\ k[i]=1}} \left[\binom{j}{k}\right]_2 \Big) \wedge\ \widetilde{\sigma}_j.$$

*Proof.* Denote $\mathfrak{S}_n$ the symmetric group on the $n$ letters and

$$X_k := \sum_{\zeta\in\mathfrak{S}_n} \left( x_{\zeta(1)}\cdots x_{\zeta(k)} \cdot (x_{\zeta(k+1)}+1)\cdots(x_{\zeta(n)}+1) \right).$$

Let us $c_j$ denote a coefficient of $\sigma_j$ in $X_k$ over integers. We show that $c_j \cdot \binom{n}{j} = \binom{n-k}{j-k}\cdot\binom{n}{k}$. More precisely, the number of monomials of degree $j$ in $X_k$ is $c_j\cdot\binom{n}{j}$ because $\binom{n}{j}$ can be seen as the number of the monomials of $\sigma_k$. Note that for a fixed $\zeta\in\mathfrak{S}_n$, the number of monomials of degree $j$ in $\left(x_{\zeta(1)}\cdots x_{\zeta(k)}\cdot(x_{\zeta(k+1)}+1)\cdots(x_{\zeta(n)}+1)\right)$ is $\binom{n-k}{j-k}$. Since the number of such polynomials is $\binom{n}{k}$, we have $c_j = \binom{j}{k}$ and $X_k = \sum c_j\sigma_j = \sum_{k\le j\le n}\binom{j}{k}\cdot\sigma_j$.

Now let us consider the binary circuit $\widetilde{X}_k$, that is,

$$\widetilde{X}_k = \bigoplus_{\zeta\in\mathfrak{S}_n} (x_{\zeta(1)}\wedge\cdots\wedge x_{\zeta(k)}\wedge(x_{\zeta(k+1)}\oplus 1)\wedge\cdots\wedge(x_{\zeta(n)}\oplus 1)),$$

so we have $\widetilde{X}_k = \oplus_{k\le j\le n}(\left[\binom{j}{k}\right]_2\wedge\widetilde{\sigma}_j)$. Hence, we can conclude that

$$S[i] = \bigoplus_{1\le k\le n} (\widetilde{X}_k\wedge k[i]) = \bigoplus_{1\le k\le n}\left( \bigoplus_{k\le j\le n}\left[\binom{j}{k}\right]_2 \wedge\widetilde{\sigma}_j \right)\wedge k[i]$$

$$= \bigoplus_{\substack{1\le k\le j\le n\\ k[i]=1}} \left[\binom{j}{k}\right]_2 \wedge\widetilde{\sigma}_j = \bigoplus_{1\le j\le n}\left( \bigoplus_{\substack{1\le k\le j\\ k[i]=1}}\left[\binom{j}{k}\right]_2\right)\wedge\widetilde{\sigma}_j.$$

The first equality follows since only $k$ values of $x_1, \ldots, x_n$ can be "1" (*i.e.*, $\sum_{i=1}^{n} x_i = k$) if and only if $\widetilde{X}_k = 1$. □

The lemma implies that if we have computed "$\oplus\left[\binom{j}{k}\right]_2$" satisfying $1 \leq k \leq j$ and $k[i] = 1$ (for $1 \leq i \leq \mu$ and $1 \leq j \leq n$), then $S_i$'s are expressed in terms of the symmetric polynomials with degree no more than $n$. The following proposition follows from Lemma 3.

**Proposition 4** *Encrypted Edit distance algorithm can be implemented on two sequences of length $n$ over an $\omega$-bit alphabet with a somewhat homomorphic scheme of depth*

$$\lceil \log \omega \rceil + \lceil \log n \rceil + \lceil \log \left( \lfloor \log(n + \lceil \frac{n}{2} \rceil - 1) \rfloor \right) \rceil + \lceil \log (n') \rceil + 2$$

*where* $n' = -\lceil \frac{n}{2} \rceil - 1 + 2 \sum_{i=0}^{\lceil \frac{n}{2} \rceil - 1} \binom{n}{i}$.

*Proof.* Let us consider a size-$(n+1)$ block. Since $D_{n,n} = D_n$ is less than $n$ and $D_{i,0}, D_{0,i}$ are greater than $2i$, $D_n$ can be expressed as a function of $D_{0,0}, D_{1,0}, \ldots, D_{\lceil \frac{n}{2} \rceil - 1, 0}$, $D_{0,1}, \ldots, D_{0, \lceil \frac{n}{2} \rceil - 1}$, and $t_{i,j}$'s satisfying $|i - j| \leq \lceil \frac{n}{2} \rceil$, as shown in Figure 4 (which means that it is enough to compute only a little part of the block).

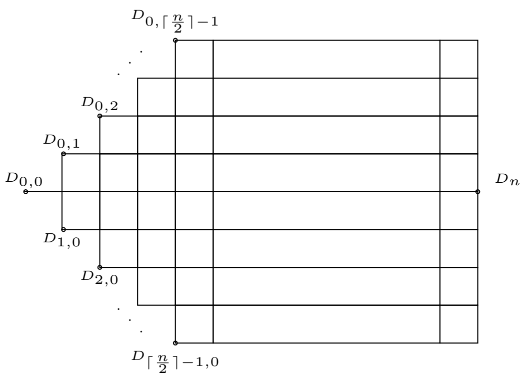
$D_{0,\lceil \frac{n}{2} \rceil - 1}$
$D_{0,2}$
$D_{0,1}$
$D_{0,0}$
$D_n$
$D_{1,0}$
$D_{2,0}$
$D_{\lceil \frac{n}{2} \rceil - 1, 0}$

Fig. 4: Grid of $(n+1)$-block

Firstly it needs $\lceil \log \omega \rceil$ levels to compute $t_{i,j}$'s with the equality circuits over $\omega$-bits. Next, from the fact that the number of the functions of $D_n$ with respect to $D_{i,0}$ is $\binom{n}{i}$, the edit distance $D_n$ is the minimum between $n' = -\lceil \frac{n}{2} \rceil + 2 \sum_{i=0}^{\lceil \frac{n}{2} \rceil - 1} \binom{n}{i}$ values which have the following form:

$$D_{i,0} + t_{i_1,j_1} + t_{i_2,j_2} + \cdots + t_{i_{n-k},j_{n-k}} + i = \ 2i + t_{i_1,j_1} + t_{i_2,j_2} + \cdots + t_{i_{n-k},j_{n-k}}$$

where $1 \leq i_1 \leq i_2 \leq \cdots \leq i_{n-k} \leq n$ and $1 \leq j_1 \leq j_2 \leq \cdots \leq j_{n-k} \leq n$. In particular, "$t_{1,1} + t_{2,2} + \cdots + t_{n,n}$" has binary circuits which consume the largest levels to be evaluated, and from Lemma 3 we expect that it needs $\lceil \log n \rceil$ levels. We note that all the values to be compared are less than $n + \lceil \frac{n}{2} \rceil - 1$ and they are considered to be of $\lfloor \log(n + \lceil \frac{n}{2} \rceil - 1) \rfloor + 1$-bit, so we have $\mathsf{D}(\texttt{COM}) = \lceil \log \left( \lfloor \log(n + \lceil \frac{n}{2} \rceil - 1) \rfloor \right) \rceil + 1$. Finally, the proposition follows that

$$\mathsf{D}(t_{i,j}) + \mathsf{D}(t_{1,1} + \cdots + t_{n,n}) + \mathsf{D}(\texttt{COM}) + \mathsf{D}(\texttt{MIN}^{n'})$$

$$= (\lceil \log \omega \rceil) + (\lceil \log n \rceil) + \left( \lceil \log(\lfloor \log(n + \lceil \frac{n}{2} \rceil - 1) \rfloor) \rceil + 1 \right) + (\lceil \log{(n')} \rceil + 1)\,.$$

$\square$

The result of Proposition 4 tells us that we can reduce the depth of computing edit distance to $\mathcal{O}(\log n') \approx \mathcal{O}(\log(2 \cdot 2^{\frac{n}{2}-1})) \approx \mathcal{O}(n)$. In particular, if $n = m = 8$, then the number of levels consumed by the edit distance algorithm is approximately 16.

## 5 Implementation and Discussions

In the following we give an estimated performance of the encrypted edit distance algorithm over DNA sequences and provide concrete timings for homomorphic evaluation of the algorithm with Shoup's NTL library [22] and Halevi-Shoup's HE library [13] over GMP. A complete description of this scheme is given in [8]. We may assume that $\omega = 2$ from the fact that $\Sigma = \{A, G, C, D\}$.

In our scenario, the third parties first partition their own DNA sequences into segments of length $n$ or $m$. Then each of the DNA sequences is expressed in a binary representation. After that, each bit is encrypted as a different ciphertext with a homomorphic encryption scheme. For parallel computation, we use an encryption scheme with plaintext space $\mathbb{Z}_2^\ell$ supporting SIMD operations with $\ell$ slots. Then one party sends the ciphertexts which hold the $\ell$ segments together to a cloud. Finally, the cloud service computes the edit distances of $\ell$ different sequence pairs simultaneously. The amortized time is computed as the total time of this algorithm evaluation divided by $\ell$.

### 5.1 Estimates

In addition to the modulus switching method, there is another noise-management technique-*bootstrapping* which evaluates the decryption circuit of homomorphic encryption scheme using the decryption key. This results in a different encryption of the ciphertext with reduced noise, so the number of homomorphic operations becomes unlimited, called *fully homomorphic encryption* (FHE).

If the length of DNA sequences is large, our encrypted edit distance algorithm requires large depth. So for sufficiently long sequences, we estimate the algorithm using an FHE scheme instead of an SWHE scheme. In particular, we present the estimated performance using the batch DGHV scheme [4]. Since bootstrapping is more costly than other operations and this scheme performs a bootstrapping right after each multiplication, the number of homomorphic multiplications directly affects the total evaluation performance. We note that the edit distance algorithm in Section 4.3 needs many more multiplications than the one in Section 4.1. For these reasons, the latter is more suitable for being evaluated via FHE.

We assume that the length of DNA sequence segments is less than 100 because a single DNA sequencer can generate millions of short DNA sequences with 100-120 nucleotides. We first count the total number of homomorphic multiplications in the edit distance algorithm up to size $(100, 100)$, which can be seen as the number of bootstrapping operations during the evaluations. Then it is multiplied by the timing for a single bootstrapping operation with their results (using the same parameters as in [4]). We present the estimates of the proposed algorithm in Table 5.

| $(\mathbf{n}, \mathbf{m})$ | **Toy** | **Small** | **Medium** | **Large** |
|---|---|---|---|---|
| Security | 42 | 52 | 62 | 72 |
| # of slots | 10 | 37 | 138 | 531 |
| pk size | 647kB | 13.3MB | 304MB | 5.6GB |
| $(1, 1)$ | 0.108s | 0.297s | 0.891s | 3.402s |
| $(2, 2)$ | 1.104s | 3.046s | 9.107s | 34.776s |
| $(3, 3)$ | 3.996s | 11.025s | 32.962s | 2min 5s |
| $(4, 4)$ | 7.104s | 19.600s | 58.599s | 3min 44s |
| $(6, 6)$ | 22.032s | 1min 1s | 3min 2s | 11min 34s |
| $(8, 8)$ | 39.168s | 1min 48s | 5min 23s | 20min 34s |
| $(10, 10)$ | 1min 18s | 3min 35s | 10min 43s | 40min 57s |
| $(20, 20)$ | 6min 19s | 17min 26s | 52min 8s | 3h 19min |
| $(30, 30)$ | 14min 13s | 39min 14s | 1h 57min | 7h 27min |
| $(50, 50)$ | 46min 30s | 2h 8min | 6h 23min | 1day 24min |
| $(100, 100)$ | 3h 34min | 9h 50min | 1day 5h | 4days 16h |

Table 5: Estimates of amortized timing for homomorphic edit distance computation using a FHE scheme [4]

## 5.2 Experimental result

Using the optimization techniques as described in Section 4.3, we can evaluate the edit distance algorithm homomorphically with low depth circuits for small DNA sequences. Taking 80-bits of security, we used many different parameters for several level parameters $L$ according to the length of the two DNA sequences, that is, we chose SWHE scheme so as to support the depth which are incurred by the computations for each case. In the set up stage, we determine the parameters of a SWHE scheme and generate a secret/public key pair and the modulus switching data.

We implemented the encrypted edit distance algorithm for two sequences of length $n$ and $m$. In our implementation, we use $\tau = n = m$ as mentioned before. The implementation results are described in Table 6. For example, it takes 27.5 seconds to obtain the encrypted edit distance from the two encrypted DNA sequences of length 8. This is about 45 times faster than the result of Section 5.1, which is expected to take 20 minutes for 72-bits of security.

| $(\mathbf{n}, \mathbf{m})$ | **Depth of Hom. Enc.** | Ring Modulus $\Phi_d$ | $\ell$ | Key Generation | Encryption | Total time | **Amortized time** |
|---|---|---|---|---|---|---|---|
| (1,1) | 1 | $d = 4369$ | 256 | 1.4761s | 0.1118s | 0.0693s | **0.0003s** |
| (2,2) | 2 | $d = 4369$ | 256 | 1.8358s | 0.2844s | 0.2532s | **0.0009s** |
| (3,3) | 8 | $d = 8191$ | 630 | 7.0162s | 1.7117s | 34.3091s | **0.0540s** |
| (4,4) | 9 | $d = 8191$ | 630 | 7.4489s | 2.4154s | 67.5116s | **0.1071s** |
| (6,6) | 16 | $d = 13981$ | 600 | 16.1076s | 9.9498s | 26min 33s | **2.6555s** |
| (8,8) | 19 | $d = 15709$ | 682 | 27.5454s | 16.4524s | 5h 13min | **27.5528s** |

Table 6: Timing of an implementation of homomorphic edit distance on an Intel Xeon i7 2.3GHz, 192GB (80 bit security)

## 6 Conclusion

In this paper, we proposed an algorithm to perform the edit distance algorithm on encrypted genomic sequences. More precisely, upon input two encrypted sequences of lengths $n$ and $m$ by a SWHE scheme, our algorithm outputs an encrypted value of their edit distance. We show that this can be done in $\mathcal{O}(nm \log(n+m))$ computations with a SWHE scheme which can homomorphically evaluate any circuit of depth $\mathcal{O}((n+m)\log(\log(n+m)))$. With our optimization technique, we can reduce the depth of computing edit distance to $\mathcal{O}(n+m)$ and the implementation shows that it takes 27.5 seconds for $n = m = 8$ using the Halevi-Shoup code [13].

Currently we could not implement our algorithm for larger parameters due to large memory requirements, but if one can manage large memory or improve the scheme to reduce the memory requirements, it is expected that the algorithm would run in one day for $n = m = 50$ when estimated based on the recent CCK+ scheme [4].

The proposed algorithm enables us to perform any sequence analysis over encrypted genomic sequences without worrying about privacy leakage. It would be very interesting to make our algorithm practical for larger parameters by improving the algorithm with the help of more efficient homomorphic encryption.

**Acknowledgements.** This work was supported by IT R&D program of MSIP/ KEIT [No. 10047212] and the MSIP (Ministry of Science, ICT&Future Planning), Korea, under the ITRC (Information Technology Research Center) support program (NIPA-2014-H0301-14-1010) supervised by the NIPA (National IT

Industry Promotion Agency). The authors would like to thank the anonymous reviewers of WAHC 2015 for their helpful comments.

## References

1. M. J. Atallah, F. Kerschbaum, and W. Du. Secure and private sequence comparisons. In *WPES*, pages 39-44, 2003.
2. E. Ayday, J.-P. Hubaux, J.L. Raisaro, and J. Rougemont. Protecting and evaluating genomic privacy in medical tests and personalized medicine. In *WPES*, pages 95-106, 2013.
3. Z. Brakerski, C. Gentry, and V. Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. In S. Goldwasser, editor, *ITCS*, pages 309-325, 2012.
4. J. H. Cheon, J.-S. Coron, J. Kim, M. S. Lee, T. Lepoint, and M. Tibouchi, and A. Yun. Batch fully homomorphic encryption over the integers. In T. Johansson and P. Q. Nguyen, editors, *EUROCRYPT*, LNCS 7881, pages 315-335, 2013.
5. E. D. Cristofaro, S. Faber, and G. Tsudik. Secure Genomic Testing with Size- and Position-Hiding Private Substring Matching. In *WPES*, pages 107-117, 2013.
6. The European Bioinformatics Institute. In http://www.ebi.ac.uk.
7. Y. Erlich and A. Narayanan. Routes for breaching and protecting genetic privacy. In *arXiv:1310.3197*, 2013.
8. C. Gentry, S. Halevi, and N. Smart. Homomorphic evaluation of the AES circuit. In R. Safavi-Naini and R. Canetti, editors, *Advances in Cryptology-Crypto*, LNCS 7417, pages 850-867, 2012.
9. M. Gymrek, A. L. McGuire, D. Golan, E. Halperin, and Y. Erlich. Identifying personal genomes by surname inference. In *Science 339*, pages 321-324, 2013.
10. M. Humbert, E. Ayday, J.-P. Hubaux, and A. Telenti. Addressing the concerns of the lacks family: Quantification of kin genomic privacy. Secure pattern matching using somewhat homomorphic encryption. In *CCSW*, ACM, pages 1141-1152, 2013.
11. Y. Huang, D. Evans, J. Katz, and L. Malka. Faster secure two-party computation using garbled circuits. In *Proceedings of the 20th USENIX Security Symposium*, pages 35-50, 2011.
12. HapMap. In http://www.hapmap.org/, 2007.
13. S. Halev and V. Shoup. Design and implementation of a homomorphic-encryption library. Technical report, IBM Technical Report, 2013.
14. S. Jha, L. Kruger, and V. Shmatikov. Towards practical privacy for genomic computation. In *IEEE Symposium on Security & Privacy*, pages 216-230, 2008.
15. M. Kantarcioglu, W. Jiang, Y. Liu, and B. Malin. A cryptographic approach to securely share and query genomic sequences. In *IEEE Transactions on Information Technology in Biomedicine*, pages 606-617, 2008.
16. V. Kolesnikov, A.-R. Sadeghi, and T. Schneider. Improved garbled circuit building blocks and applications to auctions and computing minima. In *CANS*, pages 1-20, 2009.
17. Kristin Lauter, Michael Naehrig, and Vinod Vaikuntanathan. Can homomorphic encryption be practical? In *CCSW*, ACM, pages 113-124, 2011.
18. Y. Lindell and B. Pinkas. A proof of Yao's protocol for secure two-party computation. In http://eprint.iacr.org/2004/175, 2004.
19. B. Malin and L. Sweeney. Inferring genotype from clinical phenotype through a knowledge based algorithm. In *Pacific Symposium on Biocomputing*, pages 41-52, 2002.

20. B. Malin and L. Sweeney. How (not) to protect genomic data privacy in a distributed network: using trail re-identification to evaluate and design anonymity protection systems. *Journal of Biomedical Informatics*, 37(3):571-588, 2004.
21. Personal Genome Project. In http://www.personalgenomes.org/community.html.
22. V. Shoup. NTL: A library for doing number theory. In http://www.shoup.net/ntl, 2009.
23. L. Sweeney, A. Abu, and J. Winn. Identifying Participants in the Personal Genome Project by Name, In *Harvard University. Data Privacy Lab. White Paper 1021-1*, 2013.
24. Stranger Visions. In http://deweyhagborg.com/strangervisions, 2012.
25. R. A. Wagner and M. J. Fischer. The string to string correction problem. *Journal of the ACM*, 21(1):168-173, 1974.
26. A. Yao. How to generate and exchange secrets. In R. Ostrovsky, editor, *FOCS*, pages 162-167, 1986.
27. M. Yasuda, T. Shimoyama, J. Kogure, K. Yokoyama, and T. Koshiba. Secure pattern matching using somewhat homomorphic encryption. In *CCSW*, ACM, pages 65-76, 2013.