Efficient Privacy-Preserving Edit Distance and Beyond

Ruiyu Zhu
Indiana University
Email: zhu52@indiana.edu

Yan Huang
Indiana University
Email: yh33@indiana.edu

***Abstract*—Edit distance is an important non-linear metric that has many applications ranging from matching patient genomes to text-based intrusion detection. Privacy-preserving edit distance protocols have been a long-standing goal of many security researchers. In this paper, we propose efficient secure computation protocols for private edit distance as well as several generalized applications including weighted edit distance (with potentially content-dependent weights), longest common subsequence, and heaviest common subsequence. Our protocols run 20+ times faster and use an order-of-magnitude less bandwidth than their best previous counterparts. Alongside, we propose a garbling scheme that allows free arithmetic addition, free multiplication with constants, and low-cost comparison/minimum for inputs of restricted relative-differences. Moreover, the encodings (i.e. wire-labels) in our garbling scheme can be converted from and to encodings used by traditional binary circuit garbling schemes with light to moderate costs. Therefore, while being extremely efficient on certain kinds of computations, the new garbling scheme remains composable and capable of handling generic computational tasks, hence may be of independent interest.**

I. Introduction

Edit Distance quantifies the *dissimilarity* of two strings by the minimal number of editing operations (insert, delete, and substitute) to transform one string to the other. It finds many interesting applications ranging from diagnosis and treatment of genetic diseases [1–3] to computer immunology [4] and intrusion detection [5], [6]. The basic edit distance can be generalized to take into consideration that different types of editing operations may cost very differently (by the notion of *weighted edit distance*), and the weights of the edits can even depend on the characters (e.g., in *Needleman-Wunsch* [7]). To maximize utility, real world applications often favor variants of edit distance with weights empirically adjusted based on the likelihoods of various mutations [2], [6], [8].

Much effort has been devoted to design efficient and scalable private edit distance protocols [9–11]. However, despite a number of breakthroughs in the design and implementation techniques of generic secure computation in recent years [10], [12–15], state-of-the-art private edit distance protocols are still less than satisfactory in terms of supporting many real world applications.

To circumvent this deficiency, researchers have proposed novel heuristics [16, Challenge 2] to approximate edit distances using set-difference-size, which itself can be efficiently approximated by a randomized sketch algorithm [11]. As a result, they managed to achieve surprisingly good efficiency and scalability. However, several outstanding issues of that approach are yet to be addressed:

1) The heuristics to transform a genome edit-distance problem into a set-difference-size problem are highly tied to computing the basic edit distance, while not applicable to other (arguably more useful) metrics such as weighted edit distance and any similarity-score-based metrics (e.g., longest common subsequences).
2) The accuracy of their approximation is very sensitive to the fraction of mutations as well as the composition of different types of edits. Its accuracy deteriorates quickly when the input strings become shorter and mutation-rich, which could happen when studying genetic diseases where only relatively short DNA segments of rich mutations are involved.
3) Their protocol cannot compute optimal alignments and cannot be used as a component of a larger generic oblivious computation, i.e., when the input strings are encoded secrets that cannot be revealed, or a followup secure computation needs to be performed over the optimal alignments.

A. Contribution

In this paper, we explore a new secure garbling design exploiting the characteristics of a class of oblivious computations in the *semi-honest* threat model. Our main contributions include:

1) A secure *garbling* scheme for private edit-distance-like computations. Over a class of computations, our new garbling scheme features (almost) free addition, low cost comparison, minimum, and public table lookup (with secret indices) operations. The construction is conceptually simple to prove secure. Our scheme can also be extended to handle arbitrary functions through tethering with traditional garbling over binary gates. We have implemented our scheme through exploiting the high-performance fixed-key AES cipher accelerated by Intel AESNI instructions.
2) We applied our garbling scheme to realize a broad class of frequently-used private bioinformatic computations including edit distance, weighted edit distance, Needleman-Wunsch distance, longest common subsequences (LCS), heaviest common subsequence (HCS) [17]. Unlike the approximation approach, our protocols will always calculate *precise* results. Our experiments show that these new protocols run 20+ times faster and are an order-of-magnitude more bandwidth-efficient than their best existing counterparts. Table I highlights the performance of our protocols. Moreover, unlike secure approximation proto-

cols [11], our approach keeps all the secret intermediate states of the computation, which may be useful in a subsequent oblivious computation (e.g., to compute other information about the optimal alignments).

II. Background

Notations. We use "$a := b$" to denote assigning the value of b to a; use "$x \leftarrow S$" to denote uniformly sampling an element of the set S and assigning it to x.

A. Secure Garbling

First proposed by Yao [18], garbled circuits were later formalized by Bellare et al. [19] as a cryptographic primitive of its own interest. Following the notations of Bellare, Hoang, and Rogaway, a *garbling scheme* $\mathcal{G}$ is a 5-tuple $(\mathsf{Gb}, \mathsf{En}, \mathsf{Ev}, \mathsf{De}, f)$ of algorithms, where Gb is an efficient randomized *garbler* that, on input $(1^k, f)$, outputs (F, e, d); En is an *encoder* that, on input (e, x), outputs X; Ev is an *evaluator* that, on input (F, X), outputs Y; De is a *decoder* that, on input (d, Y), outputs y. The *correctness* of $\mathcal{G}$ requires that for every $(F, e, d) \leftarrow \mathsf{Gb}(1^k, f)$ and every x,

$$\mathsf{De}(d, \mathsf{Ev}(F, \mathsf{En}(e, x))) = f(x).$$

Bellare et al. have proposed three security notions for garbling: *privacy*, *obliviousness*, and *authenticity*, which we summarize as below.

- **Privacy:** there exists an efficient simulator $\mathcal{S}$ such that for every x,
$$\left\{ (F, X, d) : \begin{array}{l} (F, e, d) \leftarrow \mathsf{Gb}(1^k, f), \\ X \leftarrow \mathsf{En}(e, x). \end{array} \right\} \approx \left\{ \mathcal{S}(1^k, f, f(x)) \right\}.$$
where "$\approx$" symbolizes computational indistinguishability.
- **Obliviousness:** there exists an efficient simulator $\mathcal{S}$ such that for every x,
$$\left\{ (F, X) : \begin{array}{l} (F, e, d) \leftarrow \mathsf{Gb}(1^k, f), \\ X \leftarrow \mathsf{En}(e, x). \end{array} \right\} \approx \left\{ \mathcal{S}(1^k, f) \right\}.$$
- **ϵ-Authenticity:** for every efficient adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$,
$$\Pr \left(\begin{array}{c} Y \neq \mathsf{Ev}(F, X) \\ \textbf{and} \\ \mathsf{De}(d, Y) \neq \perp \end{array} : \begin{array}{l} (f, x) \leftarrow \mathcal{A}_1(1^k), \\ (F, e, d) \leftarrow \mathsf{Gb}(1^k, f), \\ X \leftarrow \mathsf{En}(e, x), \\ Y \leftarrow \mathcal{A}_2(1^k, F, X). \end{array} \right) \leq \epsilon.$$

Many optimizations have been proposed to improve circuit garbling in various aspects such as bandwidth [14], [20], [21], evaluator's computation [20], memory consumption [10], and using dedicated hardware [13], [14], [22]. State-of-the-art implementations of garbling schemes using AESNI can typically produce a garbled row of the garbled truth table in roughly every 25ns [13], [15], [22].

B. Edit Distance and its Variants and Generalizations

The edit distance (also known as *Levenshtein distance*) between any two strings `s` and `t` is the minimum number of edits needed to transform `s` into `t`, where an *edit* is typically one of three basic operations: insert, delete, and substitute.

Algorithm 1 EditDistance(s, t)

```
1: for $i := 0$ to length(s) do
2:     $\mathtt{D}_{i,0} := i$;
3: for $j := 0$ to length(t) do
4:     $\mathtt{D}_{0,j} := j$;
5: for $i := 1$ to length(s) do
6:     for $j := 1$ to length(t) do
7:         $\mathtt{D}_{i,j} := \min(\mathtt{D}_{i-1,j} + c_{ins},\ \mathtt{D}_{i,j-1} + c_{del},$
           $\mathtt{D}_{i-1,j-1} + c_{sub})$;
```

Algorithm 1 is a standard dynamic programming approach to compute the edit distance between two strings. The invariant is that $\mathtt{D}_{i,j}$ always represents the edit distance between $\mathtt{s}[1..i]$ and $\mathtt{t}[1..j]$. Lines 1–2 initialize the first row of the matrix `D` while lines 3–4 initialize the first column. Within the main nested loops (lines 5–7), $\mathtt{D}_{i,j}$ is set at line 7 to the smallest of $\mathtt{D}_{i-1,j} + c_{ins}$, $\mathtt{D}_{i,j-1} + c_{del}$, and $\mathtt{D}_{i-1,j-1} + c_{sub}$, where c_{ins}, c_{del}, and c_{sub} correspond to the cost of *insert*, *delete*, and *substitute* a single character (at any position). For basic edit distance, $c_{ins} := 1$, $c_{del} := 1$, and $c_{sub} := (\mathtt{s}[i] = \mathtt{t}[j])\,?\,\mathtt{0} : \mathtt{1}$, i.e., each single-character insert, delete, and substitute incurs one unit cost while matching characters costs zero. Once the minimal edit distance is computed, it is easy to *backtrack* (from $\mathtt{D}_{i,j}$) a sequence of edits that transform $\mathtt{s}[1..i]$ to $\mathtt{t}[1..j]$, e.g., for the purpose of deriving an optimal alignment.

Weighted Edit Distance. More generally, c_{ins}, c_{del}, and c_{sub} can be adjusted to fit the goals of specific applications. For example, in diagnosing certain genetic diseases [1], [3], it is customary to set c_{ins} and c_{del} to integers between 5–10 while setting the substitution cost to 1. The rationale behind the cost gaps is that insertions and deletions (called *indel*s) occur much rarer than substitution in DNA replication so we would expect the alignment contains proportionally less *indel*s to reflect this natural fact.

Needleman-Wunsch. As the statistical models of various operations were refined with respect to the symbols involved in the mutations, researchers [8], [23–25] have found many good reasons that justify varying the costs $c_{ins}, c_{del}, c_{sub}$ with the specific characters that are inserted, deleted, or substituted. In this case, c_{ins}, c_{del} and c_{sub} can be viewed as functions over the alphabet of all possible characters. For example, for genomes, they can be encoded as one- and two-dimensional tables (Fig. 1). Note that although the weight tables are publicly known, lookups over the arrays have to be obliviously computed because the indices used to lookup are secret.

Longest common subsequence. Unlike edit distance, the length of longest common subsequence measures the *similarity* of two strings. Given strings `s` and `t`, the length of the longest common subsequence between them can be computed using dynamic programming similar to that for edit distance (Algorithm 2). Comparing to Algorithm 1, the only changes are the initialization values in line 2 and 3; and the logic to derive $D_{i,j}$ (line 7). The invariant now is that $\mathtt{D}_{i,j}$ always

TABLE I: Performance highlights

	Edit Distance		Weighted Edit Distance		Needleman-Wunsch		LCS	
	Time (s)	B/W (MB)	Time (s)	B/W (MB)	Time (s)	B/W (MB)	Time (s)	B/W (MB)
Best Prior	14.2	2,260	19.2	3,150	48.51	10,420	10.5	1,710
This Work	**0.34**	**128**	**0.62**	**448**	**2.17**	**768**	**0.31**	**96**

All distances/scores are computed over two 1000-nucleotide genomes. The weight tables used in Weighted Edit Distance and Needleman-Wunsch are those described in Figure 1. “Best Prior” results are measured on efficient implementations combining the optimizations of Huang et al. [10] with emp-toolkit [15], the most efficient realization of Half-Gates [14] garbling to date.

A	G	C	T
5	5	6	6

(a) Example c_{ins} or c_{del}

	A	G	C	T
A	0	1	2	1
G	1	0	1	2
C	2	1	0	1
T	1	2	1	0

(b) Example c_{sub}

Fig. 1: Example weight tables of genomic Needleman-Wunsch

Algorithm 2 Longest common subsequence(s, t)

```
1: for $i := 0$ to length(s) do
2:     $D_{i,0} := 0$;
3: for $j := 0$ to length(t) do
4:     $D_{0,j} := 0$;
5: for $i := 1$ to length(s) do
6:     for $j := 1$ to length(t) do
7:         $D_{i,j} := \max(D_{i-1,j}, D_{i,j-1}, D_{i-1,j-1} + w_{i,j})$;
```

represents the length of LCS(s[1..i], t[1..j]).

With basic LCS, the matching reward, $w_{i,j}$, is designed as

$$w_{i,j} = \begin{cases} 1, & \text{if } s[i] = t[j] \\ 0, & \text{otherwise} \end{cases} .$$

Heaviest Common Subsequence. As a generalization of LCS, researchers [17] have introduced the concept of *heaviest common subsequence* (HCS), just like Needleman-Wunsch generalizes edit distance. The idea is to let different characters reward differently when they match. Therefore, $w_{i,j}$ can be viewed as a matrix (to be indexed by s[i] and t[j]) where only the diagonal entries will be positive while the rest of the matrix are filled by 0s.

III. Our Approach

In this section, we describe an efficient garbling scheme for private edit-distance-like problems.

A. Insights and Design Decisions

First, we illustrate two important observations we made to motivate the design of our new garbling scheme.

Dominating Cost Factors. The dominating cost of solving the general edit distance problem lies in the oblivious computation of *addition*, *equality*, *minimum*, and an optional oblivious *table-lookup*, whose operands are in fact general number field values. This is evident from the dynamic programming pseudocode of Algorithm 1. Therefore, it is our foremost priority to maximize the efficiency of these oblivious computations when designing new garbling schemes.

Bounded Difference Values. The edit distance computation makes a number of calls to the three-minimum function, which can be instantiated as two nested calls to the two-minimum function, i.e., $\min(a, b, c) = \min\big(\min(a, b), c\big)$. Our key observation is that edit distances can be calculated in a way that all invocations of two-minimum are made with a pair of inputs (a, b) where $a - b$ is bounded by a constant independent of the absolute values of a and b. This observation opens up an opportunity to speedup private edit distance computation by designing a two-minimum gadget that only works for two inputs that are not much apart from each other but is much more efficient than a generic minimum that handles all possible inputs. (Section III-B describes an actual implementation of such a secure two-minimum gadget.)

For example, we can show that, in computing the *basic* edit distance, every call to $\min(a, b)$ can be arranged so that $a - b \in \{-1, 0, 1, 2\}$. A simple proof of this fact can be derived as below. First, because

$$\begin{aligned} &\min(D_{i-1,j} + 1, D_{i,j-1} + 1, D_{i-1,j-1} + c_{sub}) \\ &= \min\big(\min(D_{i-1,j} + 1, D_{i-1,j-1} + c_{sub}), D_{i,j-1} + 1\big), \end{aligned}$$

let $m_{i,j} = \min(D_{i-1,j} + 1, D_{i-1,j-1} + c_{sub})$, our goal is then to show

$$\begin{aligned} (D_{i-1,j} + 1) - (D_{i-1,j-1} + c_{sub}) &\in \{-1, 0, 1, 2\}, \\ (D_{i,j-1} + 1) - m_{i,j} &\in \{-1, 0, 1, 2\}. \end{aligned}$$

Since all the quantities involved are integers, it suffices to show

$$-1 \le (D_{i-1,j} + 1) - (D_{i-1,j-1} + c_{sub}) \le 2, \text{ and} \quad (1)$$

$$-1 \le (D_{i,j-1} + 1) - m_{i,j} \le 2. \quad (2)$$

The triangle inequality of basic edit distance ensures,

$$|D_{i-1,j} - D_{i-1,j-1}| \leq 1, \tag{3}$$
$$|D_{i,j-1} - D_{i-1,j-1}| \leq 1. \tag{4}$$

Thus,

$$\begin{aligned} &|D_{i-1,j} - D_{i,j-1}| \\ =&|D_{i-1,j} - D_{i-1,j-1} - (D_{i,j-1} - D_{i-1,j-1})| \\ \leq&|D_{i-1,j} - D_{i-1,j-1}| + |D_{i,j-1} - D_{i-1,j-1}| \leq 2. \end{aligned}$$

Also because (3), (4), and $0 \leq c_{sub} \leq 1$, we know

$$\begin{aligned} &-1 \leq (D_{i-1,j} + 1) - (D_{i-1,j-1} + c_{sub}) \leq 2, \text{ and} \\ &-1 \leq (D_{i,j-1} + 1) - (D_{i-1,j-1} + c_{sub}) \leq 2. \end{aligned}$$

Since

$$\begin{aligned} (D_{i,j-1} + 1) - (D_{i-1,j} + 1) \leq |D_{i,j-1} - D_{i-1,j}| \leq 2, \\ (D_{i,j-1} + 1) - (D_{i-1,j-1} + c_{sub}) \leq 2, \end{aligned}$$

thus,

$$\begin{aligned} &(D_{i,j-1} + 1) - m_{i,j} \\ =&(D_{i,j-1} + 1) - \min\big((D_{i-1,j} + 1) - (D_{i-1,j-1} + c_{sub})\big) \leq 2. \end{aligned}$$

Finally, we have

$$\begin{aligned} &(D_{i,j-1} + 1) - m_{i,j} \\ =&(D_{i,j-1} + 1) - \min\big((D_{i-1,j} + 1) - (D_{i-1,j-1} + c_{sub})\big) \\ \geq&(D_{i,j-1} + 1) - (D_{i-1,j} + 1) \geq -1. \end{aligned}$$

Therefore, we have shown both constraints (1) and (2) hold.

In general, similar observations about the constrained use of two-minimum gadgets can be deduced over extended applications including edit distance with (possibly contents-specific) varying weights, and longest common sequence with (possibly contents-specific) varying weights. Here we state our general observation but formally prove it in appendix A. We stress that, unlike the illustrating example above, our proof for the general case does not need the triangle inequality.

Let $\mathtt{s}, \mathtt{t}, D_{i,j}, c_{ins}, c_{del}, c_{sub}$ be defined as in Section II-B, where c_{ins}, c_{del} are generalized to one-dimensional tables and c_{sub} is generalized to a two-dimensional table. Let

$$\begin{aligned} m_{i,j} &= \min\Big(D_{i,j-1} + c_{del}\big[\mathtt{t}[j]\big],\ D_{i-1,j-1} + c_{sub}\big[\mathtt{s}[i],\ \mathtt{t}[j]\big]\Big) \\ u_{i,j} &= \Big(D_{i,j-1} + c_{del}\big[\mathtt{t}[j]\big]\Big) - \Big(D_{i-1,j-1} + c_{sub}\big[\mathtt{s}[i],\ \mathtt{t}[j]\big]\Big) \\ v_{i,j} &= \Big(D_{i-1,j} + c_{ins}\big[\mathtt{s}[i]\big]\Big) - m_{i,j} \end{aligned}$$

Then, there exist $D_{i,j}$-independent public constants C_1, C_2, C_3, C_4 such that for all valid indices i, j.

$$C_1 \leq u_{i,j} \leq C_2, \qquad C_3 \leq v_{i,j} \leq C_4.$$

B. The Garbling Scheme

Basic Idea. Observing that the basic gadgets needed in private edit distance computation mostly work on integers, our intuition is to generalize the existing binary signal garbling scheme [14] to one that works directly on a large number field.

With Half-Gates [14], the garbler picks, for every wire in the circuit, a secret binary string $w_0 \leftarrow \{0,1\}^{128}$ to encode 0 and uses $w_1 = w_0 \oplus \Delta$ to encode 1, where Δ is a global secret value sampled from a large space (e.g., $\Delta \leftarrow \{0,1\}^{128}$). To generalize this idea, our scheme assumes a large *public* prime p (e.g., $p > 2^{87}$). The garbler picks the global secret Δ uniform-randomly from $\mathbb{Z}_p$. For every wire in an (arithmetic) circuit, it picks k_0 uniform-randomly from $\mathbb{Z}_p$ to denote $a = 0$; and encode every integer signal $a \in \mathbb{Z}_p$ as $k_a = k_0 +_p a \times_p \Delta$ where "$+_p$" and "$\times_p$" denote mod-p addition and multiplication, respectively. For authentication purpose, we prefix a 40-bit all-zero tag to every wire key k_a as defined above. Since we only consider semi-honest adversaries who don't deviate from protocol specification, the security provided by the 40-zero authentication tags is actually *statistical*.

Notation for Wire-labels. In the rest of the paper, we always use upper-case letters (e.g., A) to name wires. If w_a^A denotes a wire-label, the superscript (in this example, A) signifies the name of the wire to which this wire-label is associated and the subscript (in this example, a) signifies the plaintext signal that the wire-label encodes. When the wire name is irrelevant in the context of the discussion, the superscript can be omitted. In our terminology, generating (or sampling) a fresh wire-label, say w_a^A, to encode a plaintext value a means first picking $k_0^A \leftarrow \mathbb{Z}_p$ (unless k_0^A for wire A has already been known) and then set $k_a^A := k_0^A +_p a \times_p \Delta$ and $w_a^A := 0^{40} \| k_a^A$. We require $w_a^A \in \{0,1\}^{128}$, so if $k_a^A < p$, leading zeros are padded in front to ensure w_a^A has exactly 128 bits.

Next, we show how every gadget needed in the private edit distance computation can be efficiently instantiated with this generalized definition of wire-labels.

Addition. To securely add two plaintext signals $a, b \in \mathbb{Z}_p$ on two wires A and B, which are represented by wire-labels

$$\begin{aligned} w_a^A &= 0^{40} \| (k_0^A +_p a \times_p \Delta) \text{ and} \\ w_b^B &= 0^{40} \| (k_0^B +_p b \times_p \Delta), \text{ respectively,} \end{aligned}$$

it suffices for the garbler to set

$$w_0^C = w_0^A +_p w_0^B$$

while the evaluator locally computes

$$w_c^C := w_a^A +_p w_b^B.$$

Assuming there is no overflow[1], it is easy to verify that $w_c^C = (w_0^C +_p (a +_p b) \times_p \Delta)$, which is indeed the expected encoding of $a +_p b$ on wire C. Moreover, recall that if $a + b < p$, then

[1]For every specific computational task, this assumption can be guaranteed to hold by setting p to be a sufficiently large prime so that no intermediate values in the computation could overflow. For example, fixing p to the largest 88-bit prime suffices for edit-distance-based human genome comparisons. We also note that, without incurring significant overhead, it is possible to use a 128-bit prime p with the extension technique discussed in Section IV.

$a + b = a +_p b$. Therefore, this essentially realizes addition over $\mathbb{Z}$ whenever $a + b < p$.

As a natural extension of secure addition, multiplying a secret value a of a wire A, encoded by wire-label

$$w^A_a = 0^{40} \| (k^A_0 +_p a \times_p \Delta)$$

with a public constant c can simply be realized as:

1) the garbler sets $w^C_0 = c \times_p w^A_0$; and
2) the evaluator locally derives the wire-label $w^Z_z = c \times_p w^A_a$.

Again, note that if $c \times a < p$ then $c \times a = c \times_p a$. Hence, it realizes constant multiplication over $\mathbb{Z}$ if $c \times a < p$.

Obviously, addition (or known-constant multiplication) is almost free—no expensive cryptographic computation nor network traffic is required—but only runs a mod-p addition (or mod-p multiplication, respectively) on each side of the protocol.

Equality. When computing c_{sub}, equality test is needed to decide whether two input characters are identical. Let a, b be two integers whose values are publicly known to fall in $\{0, 1, \ldots, \zeta\}$. Let w_a and w_b be the wire-labels that encode signals a and b on wire A and B, respectively. To securely compute if a equals b, it suffices to first securely compute $d = a - b$ (which is almost free) so that the garbler knows k^D_0 and Δ while the evaluator learns $w^D_d = 0^{40} \| (k^D_0 +_p d \times_p \Delta)$, which encodes d; then noting $d \in \{-\zeta, \ldots, \zeta\}$,

1) the garbler samples a fresh pair of wire-labels w^Z_0 and w^Z_1 to encode signal 0 and 1 on the output-wire Z; and sends the following $2\zeta + 1$ ciphertexts

$$\mathbf{Enc}_{w^D_0}(w^Z_1, id); \text{ and}$$
$$\mathbf{Enc}_{w^D_i}(w^Z_0, id), \forall i \neq 0, i \in \{-\zeta, \ldots, \zeta\}$$

in a randomly permuted order. Note that id is the integer identifier of a gadget.
2) the evaluator tries to decrypt the above $2\zeta+1$ ciphertexts using w^D_d as the key. Note that only the ciphertext encrypted with key w^D_d will be successfully decrypted to reveal a valid wire-label w^Z_z that encodes $(a = b)?1 : 0$ to the evaluator. We stress that only a successful decryption will return a wire-label with a valid zero-tag.

Namely, the evaluator will learn w^Z_1 if and only if $a = b$; and otherwise, will learn w^Z_0.

The cost of our secure equality is linear in the range of $a - b$. Recall that the cost of traditional binary garbled circuit based integer comparison is linear in the number of bits to represent the input numbers. Therefore, our approach will be more efficient when the input numbers are large while (for application-specific reasons) the difference between them can take only a few possible values.

Minimum. First, we observe that given two integers a, b, $\min(a, b) = a - \langle a - b \rangle$, where "$\langle \cdot \rangle$" is a function defined as follows,

$$\langle x \rangle = \begin{cases} x, & \text{if } x \geq 0; \\ 0, & \text{otherwise.} \end{cases}$$

In essence, "$\langle \cdot \rangle$" is a generalized comparison, which can be accomplished in our garbling scheme with a similar idea that enables secure comparison described above. That is, let X, Z be the input- and output-wire, respectively, and assume $x \in \{-\zeta, \ldots, \zeta\}$, the garbler simply sends the following $2\zeta+1$ ciphertexts in a randomly permuted order:

$$\mathbf{Enc}_{w^X_i}(w^Z_0, id), \forall i \in \{-\zeta, \ldots, -1\}; \text{ and}$$
$$\mathbf{Enc}_{w^X_i}(w^Z_i, id), \forall i \in \{0, \ldots, \zeta\}$$

where for $0 \leq i \leq \zeta$, w^Z_i is a freshly generated wire-label representing plaintext value i on the output-wire Z.

Based on a very similar analysis for the cost of secure comparison, our secure minimum gadget will outperform the traditional binary garbled circuit approach when the input numbers are large but the difference between these numbers can take very limited number of values. We show earlier in Section III-A that, thanks to the "bounded increments" property, this is indeed the case in edit-distance type of applications.

Table-lookup. A one-dimensional table of n entries can be viewed as an association-list

$$\{(0, v_0), (1, v_1), \ldots, (n-1, v_{n-1})\},$$

where v_is are bounded integer values. A table-lookup gadget can be treated as an unary gate with input-wire I and output-wire V. Given a wire-label w^I_i that encodes plaintext index i, a secure table look-up will output a wire-label $w^V_{v_i}$ that actually encodes v_i—the value in the table corresponding to the index i. In our scheme, this can be straightforwardly realized as follows:

1) the garbler generates fresh wire-labels $w^V_{v_0}, \ldots, w^V_{v_{n-1}}$ to encode $v_0, \ldots, v_{n-1}$ on the output-wire V; and sends the following n ciphertexts in a randomly permuted order:

$$\mathbf{Enc}_{w^I_i}(w^V_{v_i}, id), \ \forall i \in \{0, \ldots, n-1\}$$

where w^I_i encodes i on the input index wire I.
2) the evaluator uses w^I_i as key to decrypt the above n ciphertexts. Due to the way the ciphertexts are constructed, precisely one of them will be successfully decrypted, revealing the wire-label $w^V_{v_i}$ that encodes v_i.

Moreover, looking up a multi-dimensional table with our scheme is readily reducible into a one-dimensional table lookup problem. Take the two-dimensional m-by-n-table lookup as an example. A two-dimensional table can always be mapped to a one-dimensional table by concatenating the rows, i.e., an index (i, j) (where $0 \leq i < m, 0 \leq j < n$) over the 2D-table can be translated into an index $k = i * m + j$ over a 1D-table of size mn. Since m is public, the affine mapping of wire-labels w^I_i and w^J_j (that encode the row and column indices) to the wire-label w^K_k (that encode the translated index) is almost free with our scheme. Once the translation is done, the secure 2D-table lookup reduces to sending and trial-decrypting mn ciphertexts—the same treatment to securely lookup a 1D-table of mn entries.

Recall that with traditional binary circuit garbling schemes, a generic multiplexer-based secure table lookup is significantly more expensive because: 1) each index and each content integer in the table need to be encoded by multiple wire-labels; 2) n multiplexers would be needed to scan the table while the cost of each multiplexer depends on the bit length of the table content values as well as the length of the index. Alternatively, if the table is small, a secure table lookup can be realized as a giant garbled truth table like Huang et al. suggested [10]. However, it is unclear how this can be accomplished efficiently with AESNI support because $\log n$ keys (one key per bit of the index) are involved in producing every garbled row. A more straightforward solution would use SHA hashing, which, however, is orders-of-magnitude slower than AESNI instructions. In contrast, secure table lookup with our garbling scheme is significantly cheaper.

Handling the Initial Inputs. We assume the initial circuit inputs to our (arithmetic) circuit are in *bits* and the processing of these binary input values resembles that in binary garbled circuit protocols, i.e., the circuit generator's private input bits are encoded by wire-labels that are directly sent to the evaluator while the circuit evaluator's private input bits are translated to their corresponding wire-labels through oblivious transfer. Though we stress that the format of the wire-labels that encode the initial input bits conforms to the mod-p field notion of wire-labels required by our garbling scheme. Therefore, an (almost-free) set of addition and public-constant multiplication gadgets will translate the bits representation of input values into their arithmetic representations.

Efficient Implementation. Today's high-performance garbling schemes rely heavily on the premise that they can be built from an ideal block cipher instantiated by *fixed-key* AES. Fortunately, our scheme can be implemented using fixed-key AES accelerated by AESNI instructions. In constructing all the gadgets above, our garbling scheme requires only one cryptographic primitive, $\mathbf{Enc}_{w_{in}}(id, w_{out})$, where w_{in} and w_{out} are 128-bit wire-labels with valid zero-tags and $i < 2^{128}$ is an integer serving as a gadget counter. Inspired by the idea of Zahur et al. [14], we implement $\mathbf{Enc}_{w_{in}}(id, w_{out})$ as

$$\mathbf{Enc}_{w_{in}}(i, w_{out}) = \pi(K) \oplus K \oplus w_{out}$$

where $K = 2w_{in} \oplus i$ (note that $2w_{in}$ refers to doubling w_{in} in the finite field GF(2^{128})) and π is a random permutation that can be instantiated as a fixed-key AES. Thus, $\mathbf{Dec}_{w_{in}}(i, c)$ can be naturally defined as

$$\mathbf{Dec}_{w_{in}}(i, c) = \begin{cases} m := \pi(K) \oplus K \oplus c, & m \text{ has an all-zero tag;} \\ \perp, & \text{otherwise.} \end{cases}$$

where K is as was defined above.

Security. We formalize our garbling scheme in Figure 2. Note that equality, minimum and table-lookup gadgets are all essentially realized in terms of a primitive operation we call *projection*. Secure projection can obliviously *project* an input signal a_i to its corresponding output signal b_i based on a publicly known map $\{(a_1, b_1), \ldots, (a_n, b_n)\}$. Thus, to prove the garbling scheme to be secure, it suffices to just consider additions and projections.

Next, we state and prove our main theorem.

Theorem 1: If π is an ideal block cipher that is used to realize $\mathbf{Enc}$ and $\mathbf{Dec}$ as described above. the scheme in Figure 2 satisfies the privacy, obliviousness, and authenticity definitions outlined in Section II-A.

Proof ***Privacy:*** Figure 3 describes a simulator $\mathsf{Sim}_{\mathsf{prv}}$ that can be used to show our garbling scheme is private. The construction of $\mathsf{Sim}_{\mathsf{prv}}$ is similar to Gb except for three changes that we highlighted in red: (1) $\mathsf{Sim}_{\mathsf{prv}}$ has a third input $f(x)$; (2) it uses $f(x)_i$ to replace t when producing the decoding information d^{O_i}; and (3) it calls En with an arbitrary legitimate input x_0 to produce X in the end.

For any x, consider (F, X, d) generated by

$$(F, e, d) \leftarrow \mathsf{Gb}(1^k, f)$$
$$X := \mathsf{En}(e, x)$$

and the tuple (F', X', d') produced by $\mathsf{Sim}_{\mathsf{prv}}(1^k, f, f(x))$. Should $\mathsf{Sim}_{\mathsf{prv}}$ know x, then it would not replace t with $f(x)_i$ in producing $d_t^{O_i}$ but simply call $\mathsf{En}(\hat{e}, x)$ in the end to generate X'. Let the output distribution generated by this $\mathsf{Sim}^x_{\mathsf{prv}}$ be (F'', X'', d''). It is easy to see that (F, X, d) and (F'', X'', d'') are identically distributed. Now, to see $(F'', X'', d'') \approx (F', X', d')$, we note that

(1) the distinguisher cannot tell the two distributions apart by examining any garbled gates because x_0 is a legitimate input and $\mathsf{Sim}_{\mathsf{prv}}$ followed exactly the same procedure to handle all garbled gates as Gb does.

(2) For every output-wire O_i, for every $w_t^{O_i}$ the distinguisher does not learn, $d_t^{O_i}$ is no different from a random string (because π is an ideal cipher); for the $w_t^{O_i}$ the distinguisher learns, it (always) only gets $f(x)_i$ from decrypting $d_t^{O_i}$, which is no different from what it would learn from examining (F, X, d).

Obliviousness: We simply observe that in $\mathsf{Sim}_{\mathsf{prv}}$, $f(x)$ is used only to compute d, which is dropped in the security definition of obliviousness. Thus, the simulator $\mathsf{Sim}_{\mathsf{obl}}$ can be derived from $\mathsf{Sim}^x_{\mathsf{prv}}$ by simply drop $f(x)$ in the input and the third component d in the output.

Authenticity: We note that due to the construction of $\mathbf{Enc}$, if the adversary $\mathcal{A}$ can provide any Y' such that $Y' \neq \mathsf{De}(d, \mathsf{Ev}(F, X)))$ and $\mathsf{De}(d, Y') = k \neq \perp$ (where $(F, e, d) \leftarrow \mathsf{Gb}(1^k, f), X := \mathsf{En}(e, x)$), then $\mathcal{A}$ must know the $w_k = w_0 +_p k \times_p \Delta$ which is the output wire-label corresponding to k. However, without knowing w_0 and Δ, for any particular k, $\mathcal{A}$ can only guess $w_k = w_0 +_p k \times_p \Delta$ correctly with probability at most $1/p$. Hence, if we know from the public circuit that an output-wire can have at most n possible different k values, the adversary can only succeed in guessing a valid output wire-label with probability at most n/p. Thus, our scheme guarantees n/p-authenticity.

■

```
$\mathsf{Gb}(1^k, f)$
  $\Delta \leftarrow \{0,1\}^k$
  for $I \in f.\mathsf{input\text{-}wires}$ do
    $w_0^I \leftarrow \mathbb{Z}_p$
  $\hat{e} := (w_0^1, \ldots, w_0^{|f.\mathsf{input\text{-}wires}|}, \Delta)$
  for $g \in f.\mathsf{gates}$ do
    if $g$ is Add-gate then
      $\{I_1, I_2\} := g.\mathsf{input\text{-}wires}$
      $O := g.\mathsf{output}$
      $w_0^O := w_0^{I_1} +_p w_0^{I_2}$
    else if $g$ is $\mathsf{Proj}_\phi$-gate then
      $I := g.\mathsf{input\text{-}wire}$
      $O := g.\mathsf{output\text{-}wire}$
      $\mathbb{Z}_\zeta := g.\mathsf{domain}$
      for $t \in \mathbb{Z}_\zeta$ do
        $w_t^I := w_0^I +_p t \times_p \Delta$
        $w_t^O := w_0^O +_p \phi(t) \times_p \Delta$
        $c_t^g \leftarrow \mathbf{Enc}_{w_t^I}(g, w_t^O)$
      $\boldsymbol{c}^g := \{c_0^g, \ldots, c_{\zeta-1}^g\}$
  $\hat{F} := (\boldsymbol{c}^1, \ldots, \boldsymbol{c}^{|f.\mathsf{Proj}|})$
  for $O_i \in f.\mathsf{output\text{-}wires}$ do
    $\mathbb{Z}_\zeta \leftarrow i.\mathsf{domain}$
    for $t \in \mathbb{Z}_\zeta$ do
      $w_t^{O_i} := w_0^{O_i} +_p t \times_p \Delta$
      $d_t^{O_i} \leftarrow \mathbf{Enc}_{w_t^{O_i}}(\mathsf{out} \| t)$
    $\boldsymbol{d}^i := \{d_0^O, \ldots, d_{\zeta-1}^O\}$
  $\hat{d} := (\boldsymbol{d}^1, \ldots, \boldsymbol{d}^{|f.\mathsf{output\text{-}wires}|})$
  return $(\hat{F}, \hat{e}, \hat{d})$

$\mathsf{En}(\hat{e}, \hat{x})$
  $(w_0^1, \ldots, w_0^{|f.\mathsf{input\text{-}wires}|}, \Delta) := \hat{e}$
  for $x_i \in \hat{x}$ do
    $X_i := w_0^i +_p x_i \times_p \Delta$
  return $\hat{X} := (X_1, \ldots, X_{|f.\mathsf{input\text{-}wires}|})$

$\mathsf{Ev}(\hat{F}, \hat{X})$
  $(X_1, \ldots, X_{|f.\mathsf{input\text{-}wires}|}) := \hat{X}$
  $(\boldsymbol{c}^1, \ldots, \boldsymbol{c}^{|f.\mathsf{Proj}|}) := \hat{F}$
  for $I_i \in f.\mathsf{input\text{-}wires}$ do
    $w^{I_i} := X_i$
  for $g \in f.\mathsf{gates}$ do
    if $g$ is Add-gate then
      $\{I_1, I_2\} := g.\mathsf{input\text{-}wires}$
      $O := g.\mathsf{output\text{-}wire}$
      $w^O := w^{I_1} +_p w^{I_2}$
    else if $g$ is $\mathsf{Proj}_\phi$-gate then
      $I := g.\mathsf{input\text{-}wire}$
      $O := g.\mathsf{output\text{-}wire}$
      $\mathbb{Z}_\zeta := g.\mathsf{domain}$
      $\{c_0, \ldots, c_{\zeta-1}\} := \boldsymbol{c}^g$
      for $t \in \mathbb{Z}_\zeta$ do
        if $\mathbf{Dec}_w(g, c_t) \neq \bot$ then
          $w^O := \mathbf{Dec}_w(g, c_t^g)$
  for $O_i \in f.\mathsf{output\text{-}wires}$ do
    $Y_i := w^{O_i}$
  $\hat{Y} := (Y_1, \ldots, Y_{|f.\mathsf{output\text{-}wires}|})$
  return $\hat{Y}$

$\mathsf{De}(\hat{d}, \hat{Y})$
  $(Y_1, \ldots, Y_{|f.\mathsf{output\text{-}wires}|}) := \hat{Y}$
  for $d_i \in \hat{d}$ do
    $\mathbb{Z}_\zeta := i.\mathsf{domain}$
    $\{d_0, \ldots, d_{\zeta-1}\} := \boldsymbol{d}^i$
    for $t \in \mathbb{Z}_\zeta$ do
      if $\mathbf{Dec}_{Y_i}(d_k) = \mathsf{out} \| k$ then
        $y_i := k$
  return $\hat{y} := (y_1, \ldots, y_{|f.\mathsf{output\text{-}wires}|})$
```

Fig. 2: The garbling scheme

IV. Extensions

In this section, we discuss two extensions of our approach: 1) to support garbling of arbitrary computations; and 2) to accommodate the need for higher computational security guarantee.

A. Garbling Arbitrary Computations

Our garbling scheme as is described so far seems not efficient for generic computations because we haven't discussed how to *multiply* two secret values efficiently. Below we discuss how to extend our garbling scheme to realize generic secure computation. The basic idea is to tether our garbling scheme with the traditional binary circuit garbling, which is known to be generic and can leverage Half-Gate technique [14].

Arithemtic Wire-labels to Binary Wire-labels. Suppose the circuit garbler knows $w_0 = 0^{40} \| k_0$ and Δ, whereas the evaluator knows $w_a = 0^{40} \| (k_0 +_p a \times_p \Delta)$. Let the binary form of the integer a be $a_1 a_2, \ldots, a_n$. After conversion, we hope the the garbler learns wire-labels $w_{1,0}, \ldots, w_{n,0}$ and δ while the evaluator learns $w_{1,a_1}, \ldots, w_{n,a_n}$ such that $w_{i,a_i} = w_{i,0} \oplus a_i \delta$.

```
$\mathsf{Sim}_{\mathsf{prv}}(1^k, f, f(x))$
  $\Delta \leftarrow \{0,1\}^k$
  for $I \in f$.input-wires do
    $w_0^I \leftarrow \mathbb{Z}_p$
  $\hat{e} := (w_0^1, \ldots, w_0^{|f.\text{input-wires}|}, \Delta)$
  for $g \in f$.gates do
    if $g$ is Add-gate then
      $\{I_1, I_2\} := g$.input-wires
      $O := g$.output
      $w_0^O := w_0^{I_1} +_p w_0^{I_2}$
    else if $g$ is $\text{Proj}_\phi$-gate then
      $I := g$.input-wire
      $O := g$.output-wire
      $\mathbb{Z}_\zeta := g$.domain
      for $t \in \mathbb{Z}_\zeta$ do
        $w_t^I := w_0^I +_p t \times_p \Delta$
        $w_t^O := w_0^O +_p \phi(t) \times_p \Delta$
        $c_t^g \leftarrow \mathbf{Enc}_{w_t^I}(g, w_t^O)$
      $\boldsymbol{c}^g := \{c_1^g, \ldots, c_{\zeta-1}^g\}$
  $F' := (\boldsymbol{c}^1, \ldots, \boldsymbol{c}^{|f.\text{Proj}|})$
  for $O_i \in f$.output-wires do
    $\mathbb{Z}_\zeta \leftarrow i$.domain
    for $t \in \mathbb{Z}_\zeta$ do
      $w_t^{O_i} := w_0^{O_i} +_p t \times_p \Delta$
      $d_t^{O_i} \leftarrow \mathbf{Enc}_{w_t^{O_i}}(\text{out}\| f(x)_i)$ {$f(x)_i$ denotes the value of $f(x)$ on the $i^{th}$ output-wire.}
    $\boldsymbol{d}^i := \{d_0^O, \ldots, d_{\zeta-1}^O\}$
  $d' := (\boldsymbol{d}^1, \ldots, \boldsymbol{d}^{|f.\text{output-wires}|})$
  $X' := \text{En}(\hat{e}, x_0)$ {$x_0 \in f$.domain denotes a legitimate plaintext input.}
  return $(F', X', d')$
```

Fig. 3: The Simulator for Proving Privacy

We describe two methods to accomplish this goal that exhibit complementary tradeoffs between performance and generality.

1) **Via secret shares.** If the range of a is publicly known to be restricted to $\{0, \ldots, \zeta\}$. The basic idea is to let the garbler send a random permutation of

$$\mathbf{Enc}_{w_i}(i \oplus m), \ \forall i \in \{0, \ldots, \zeta\}$$

where m is a $\lceil \log \zeta \rceil$-bit secret mask picked by P_1. Thus, the evaluator who has w_a is able to recover $a \oplus m$. Then, the two parties can use traditional garbled circuit protocols [14] to run any followup computation over a by starting from their respective shares m and $a \oplus m$.

Per converting an arithmetic wire, it will cost $\zeta + 1$ encryptions to send the encrypted masked-shares, 176 encryptions to translate the garbler's input bits and 88 oblivious transfers (for the evaluator's 88-bit input) in the second stage of the secure computation. This approach would be preferred when ζ is known to be relatively small. As ζ grows too big, it becomes infeasible to transmit $O(\zeta)$ encryptions, in which case we can opt to the following alternative conversion method suitable for large ζs.

2) **Via generic secure modulo-arithmetic.** The basic idea is to construct a binary garbled circuit to securely compute $(k_a - k_0)/\Delta$ where "$-$" and "$/$" are mod-p subtraction and division, respectively. By requiring the garbler to locally compute $(\Delta^{-1} \bmod p)$, we can reduce the above computation into a secure mod-p subtraction followed by a secure mod-p multiplication, both realized by a traditional binary circuit garbling scheme.

Because $k_0, k_a, p \in \{0,1\}^{88}$, the cost of this approach is that of a traditional garbled circuit secure computation protocol with 88×3 input bits (88×2 bits from the garbler and 88 bit from the evaluator), an 88-bit mod-p secure subtraction, and an 88-bit mod-p secure multiplication. Since it only depends on the computational security parameter rather than the range of the plaintext values, it fits better when the range of a can be very big (e.g., more than 2^{17}).

With either approach, we stress that the authenticity of the final output-wire labels holds if $a \ll p$, because without knowing w_0 and Δ, for any $a, b \in \mathbb{Z}_p$,

$$(w_0 +_p a \times_p \Delta, w_0 +_p b \times_p \Delta) \approx (X, Y)$$

where X, Y are uniform random samples from $0^{40} \| \mathbb{Z}_p$. So for example, when it is known that $a \leq 2^{32}$ from the application context, our approach can offer at least $87 - 32 = 55$ bits authenticity.

Binary Circuit Wire-labels to Arithmetic Wire-labels. Converting wire-labels from traditional binary circuit garbling to arithmetic wire-labels used in ours is more straightforward: the garbler only needs to send a randomly permuted pair of ciphertext

$$\left[\mathbf{Enc}_{w'_0}(w_0), \ \mathbf{Enc}_{w'_1}(w_1)\right]$$

per wire in the binary circuit, where w'_0, w'_1 are wire-labels conforming to the format required by the traditional garbling (e.g., $\forall b \in \{0,1\}, w'_b = w'_0 \oplus b\Delta, \Delta \in \{0,1\}^{128}$), and w_0, w_1 are freshly sampled labels based on our garbling scheme (e.g., $\forall b \in \{0,1\}, w_b = 0^{40} \| k_b, k_b = k_0 +_p b \times_p \Delta, \Delta \in \mathbb{Z}_p$). So the evaluator can decrypt the ciphertext corresponding to the binary circuit wire-labels it learns from the evaluation.

To derive an arithmetic wire-label w_a that encodes

$$a = a_0 + a_1 \times 2 + \cdots + a_n \times 2^{n-1}, \quad a_i \in \{0,1\},$$

from binary wire-labels $w'_{a_0}, \ldots, w'_{a_n}$, it suffices to first convert $w'_{a_0}, \ldots, w'_{a_n}$ to $w_{a_0}, \ldots, w_{a_n}$ that conform to wire-labels in our scheme, then w_a can be locally derived from

$w_{a_0}, \ldots, w_{a_n}$ using the (almost) free addition and constant multiplication.

B. Stronger Computational Security

The scheme as we described thus far can guarantee 87 bits computational security. Next, we show that it is possible to modify our scheme to provide at least 127 bits computational security.

The key idea is to set p to be a 128-bit prime (in doing so, we abandon the idea of using 40-bit all-zero tags to identify successful decryptions) and retrofit each garbled row $\mathbf{Enc}_{w_{in}}(i, w_{out})$ with an additional 128-bit authentication tag. That is, we redefine

$$\mathbf{Enc}_{w_{in}}(i, w_{out}) = (C_1, C_2)$$

where

$$\begin{aligned} C_1 &= \pi(K) \oplus K \oplus w_{out} \\ C_2 &= \pi(K \oplus 1) \oplus K \oplus w_{out} \\ K &= 2w_{in} \oplus i \end{aligned}$$

where $2w_{in}$ refers to doubling w_{in} in the finite field GF(2^{128}) and π can be instantiated as a fixed-key AES.

Symmetrically, we can define

$$\mathbf{Dec}_{w_{in}}(i, (C_1, C_2)) = \begin{cases} m_1, & m_1 = m_2 \\ \perp, & \text{otherwise} \end{cases}$$

where

$$\begin{aligned} m_1 &= \pi(K) \oplus K \oplus C_1 \\ m_2 &= \pi(K \oplus 1) \oplus K \oplus C_2, \end{aligned}$$

and K is as was defined above. Thus, the evaluator, who obtains w'_{out} by trial decrypting garbled rows in the i-th gate with wire-label w'_{in}, can verify whether

$$\pi(2w'_{in} \oplus i \oplus 1) \oplus 2w'_{in} \oplus i \oplus w'_{out} = C_2$$

to tell if the decryption was successful. The intuitive reason behind this is that if w'_{in} is not equal to w_{in} (the key used to generate (C_1, C_2), then $w'_{out} \neq w_{out}$ and

$$\pi(2w'_{in} \oplus i \oplus 1) \oplus 2w'_{in} \oplus i \oplus w'_{out} \neq \pi(2w_{in} \oplus i \oplus 1) 2w_{in} \oplus i \oplus w_{out},$$

except for a negligible probability.

V. Results

We have implemented our garbling scheme and applied it to several applications whose problem structures resemble that of edit distance. In this section, we evaluate the performance of our approach with both micro-benchmarks and end-to-end applications.

Experiment Setup. Our experiments are conducted on two Amazon EC2 free-tier `t2.micro` instances [26] running Ubuntu 14.04. The instances are connected over a 1 GB LAN with $0.031ms$ latency.

We implemented our scheme in C/C++, using Intel AESNI intrinsic instructions to realize the fixed block cipher π. We use the very recent C/C++ library emp-tool [15]'s realization of the Half-Gates [14] garbling scheme and efficient OT extension [27], [28] to construct the baseline protocols to compare with. For fair comparison, all baseline protocols are carefully designed with Boolean circuits optimized to take advantage of free-XOR [12] benefits.

A. Micro-benchmarks

Secure Addition. Table II shows the performance of secure addition in our approach. Recall that addition is (almost) free, our scheme is able to perform one addition every 2.8 nano-seconds, regardless of the bit-length of the numbers to add. This result is in line with the cost of computing a mod-p addition on this hardware. In contrast, costs of binary circuit based addition circuits (powered by Half-Gates) increase roughly linearly with the width of the adder. Ours are 500–40,000 times faster and consume no bandwidth.

Secure Equality and Minimum. We evaluate secure equality and minimum together, because the cryptographic mechanism implementing the two gadgets in our scheme are quite the same: both by obliviously mapping ζ encodings of a secret signal's ζ possible plaintext values to the encodings of their corresponding outputs (using ζ cipher calls). Their timings (as well as bandwidths) with respect to ζ (the maximum difference between the two inputs) overlap on the performance charts (Figure 4).

In comparison, the costs of existing binary garbled circuit based, general equality and minimum implementations grow with the number of bits needed to represent the inputs; while minimum is twice as expensive as comparison (internally, minimum is done using a secure comparison followed by a oblivious multiplexer). For example, although our gadgets are less general, we can run 6–70 times faster and 5–52 times more efficient in bandwidth when operating on 32-bit values (when $3 \leq \zeta \leq 25$).

Secure Table-Lookup. Figure 5 shows the efficiency of secure table lookup with our scheme and compares it to the best existing garbled-circuit-based implementation. Two relevant parameters are used to describe the table: the table size (i.e., the number of entries in the table) and the bit-length of each entry. With our scheme, the cost of secure table-lookup grows linearly with the number of entries in the table, but not the bit-length of the entries.

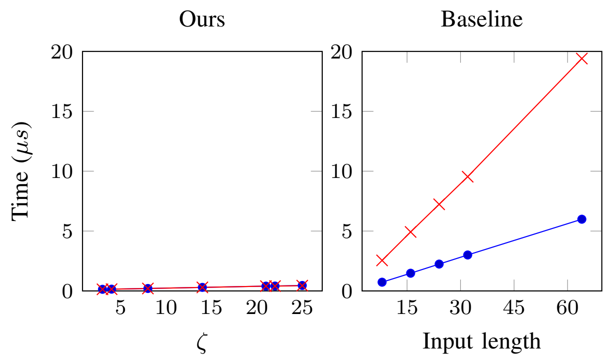
In contrast, a garbled circuit based table-lookup costs more when the values in the table grow bigger, because the secure multiplexers has to take wider inputs. In our experiments, we assumed the table contains either 4-, 8-, or 12-bit values, representing the value range of constant tables used in many practical applications. On these table parameters, our approach is 3.6–20 times faster and 6–23 times more bandwidth-efficient.

Wire-label Conversions. Converting Boolean wire-labels from the Half-Gates binary circuit garbling scheme into arithmetic wire-labels in our scheme is highly efficient, at about $420ns$ (and $\sim$32 bytes bandwidth) per bit of Boolean wire-

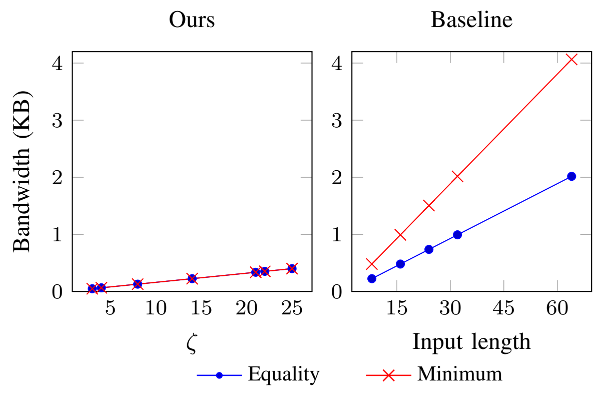
TABLE II: Costs of secure additions

	Time (ns)				Bandwidth (byte)			
	8-bit	16-bit	32-bit	64-bit	8-bit	16-bit	32-bit	64-bit
Half-Gates [14]	1420	2770	5520	11100	224	480	992	2016
This Work	**2.8**				**0**			

Above timings are in line with the cost of AESNI-based garbling ($\sim 45ns$ per garbled row) and the costs of modulo arithmetic with respect to an 88-bit prime ($2.8ns$ per modulo addition). Timings of Half-Gates are measured by averaging 10^6 iterations while that of this work is taken over 10^9 iterations.



(a) Time



(b) Bandwidth

Fig. 4: Costs of secure Equality and Minimum. (The costs of equality and minimum with our scheme grows with ζ (the size of the input signal's domain) while those using traditional binary gate garbling grows with the input length. Timings are measured over 10^6 iterations.)

label, since it involves only two garbled rows per Boolean wire (Table III).

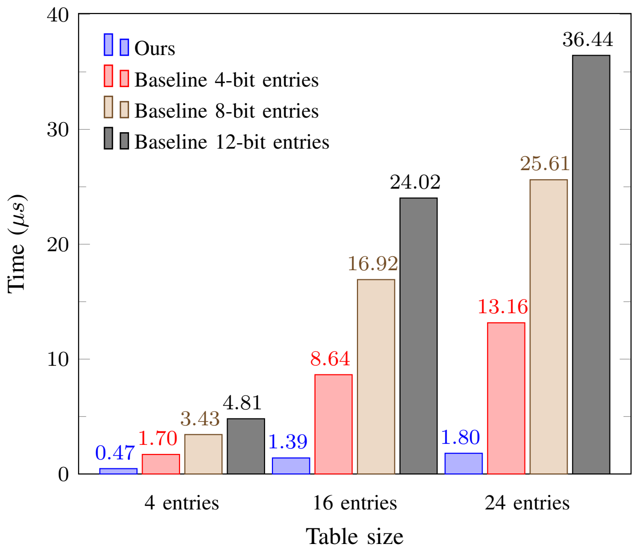
Converting arithmetic wire-labels into Boolean ones used in Half-Gates is comparatively more expensive. The generic method needs 9.6 millisecond and 2MB per arithmetic wire-label, mostly spent on oblivious mod-p multiplication under the Half-Gates garbling scheme. However, if the arithmetic wire-label is known to denote values of a smaller range (usually $< 2^{20}$ possibilities), the faster secret-sharing based label conversion method turns out very efficient. For example, if the range of the arithmetic signal is up to 2^8, the conversion an arithmetic wire-label takes only less than $11ns$ and 4.2KB bandwidth. We empirically find that the secret-sharing based conversion can outperform the generic method when the plaintext value is within 2^{16}.

B. Application Performance

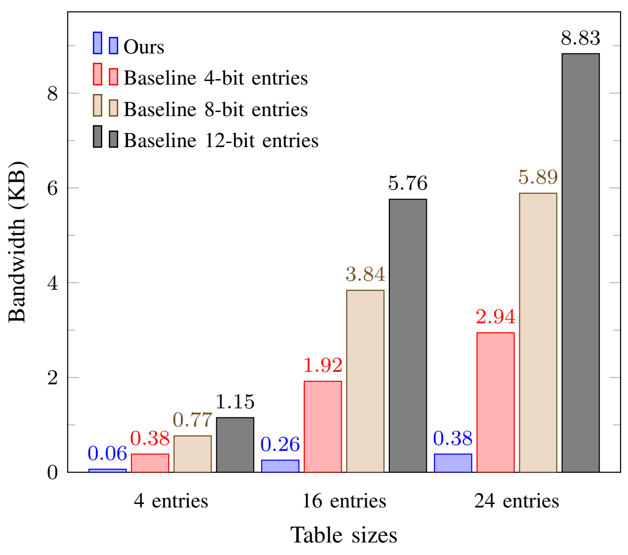
We applied the proposed garbling scheme to implementing five applications: edit distance, weighted edit distance, Needleman-Wunsch, longest common subsequence (LCS), and heaviest common subsequence. We delineated the time (Figure 6 and Figure 8) and bandwidth (Figure 7 and Figure 9) costs of these end-to-end applications over input strings of lengths 800–4000 characters. The curves all show a quadratic shape, which is consistent with the theoretical complexity of the underlying dynamic programming algorithms. We used $c_{ins} = c_{del} = 5$ and $c_{sub} = 1$ in the weighted edit distance experiments and the weight tables of Figure 1 in the Needleman-Wunsch experiments.

Efficiency Impact of Higher Security Parameters. We have also evaluated the performance degradation as a result of raising the computational security parameter from 87 bits to 126 bits. We observed 22% (for Needleman-Wunsch) to 50% (for basic Edit Distance) slowdowns and a uniform 100% increase in bandwidth for all applications. The variation of the time slowdown is due to the fact that the proportion of CPU time spent on sampling wire-labels (which also requires AESNI calls) varies from applications to applications, while increasing the computational security parameter from 87 to 126 does not affect the cost of generating fresh wire-labels.

Additional Optimizations. Notably, for the purpose of identifying similar patients, we can further take advantage of the fact that we only care about similar genomes whose distances from the query genome are below a public threshold (e.g., max distance set to 300 when comparing two 5000-nucleotide genomes). Thus, it suffices to scan a relatively narrow diagonal band of the matrix $D_{i,j}$. Given a distance threshold of 300, our protocol can *precisely* compute the edit distance between two 5000-character strings in 2.12 seconds (cf. 4.5+ hours in [10]), generating merely 0.284 GB (cf. 75+ GB in [10]) network traffic (not shown in the figures), which would suggest a performance milestone marking secure computation



(a) Time



(b) Bandwidth

Fig. 5: Costs of secure table lookup. (Timings are measured by averaging 10^6 iterations.)

TABLE III: Costs of label conversions.

	Time (μs)				Bandwidth (KB)			
	8-bit	16-bit	32-bit	64-bit	8-bit	16-bit	32-bit	64-bit
Boolean to Arithmetic	3.34	6.69	13.31	26.75	0.26	0.51	1.02	2.05
Arithmetic to Boolean (via secret-shares)	10.89	743.2	———		4.22	1048.83	———	
Arithmetic to Boolean (via generic secure modulo-arithmetic)	9628				2004.96			

Timings in the first two rows are averaged over 10^6 iterations while that in the third row is over 10^3 iterations.

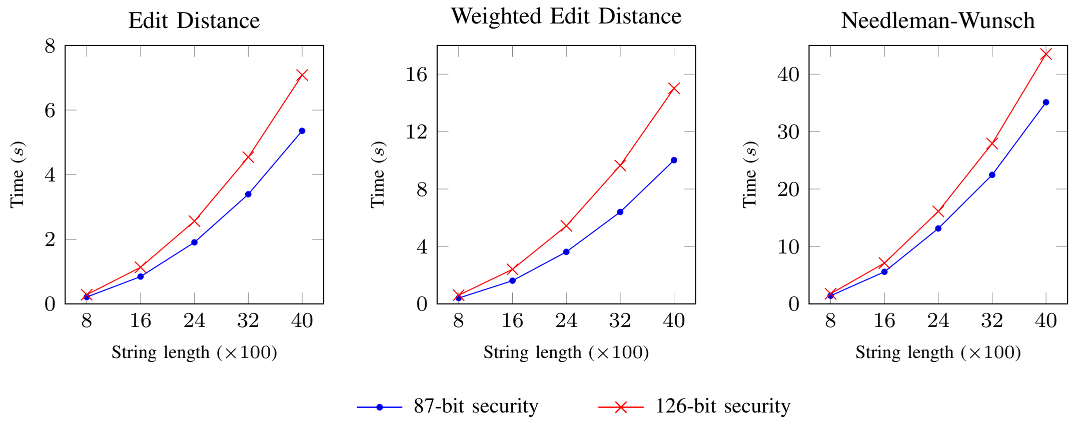


Fig. 6: Time cost of Edit Distance and its variants. (Timings are averaged over 10 iterations.)

technology becoming mature for privacy-preserving clinical applications.

Efficient Genome Database Search. As a demonstration of the efficiency and composability of our approach, we extended our private edit distance protocol into a two-stage patient genome database search protocol and tested our protocol on cancer patient genomes [11]. Our protocol is able to compute the edit distance between every pair of patient genomes

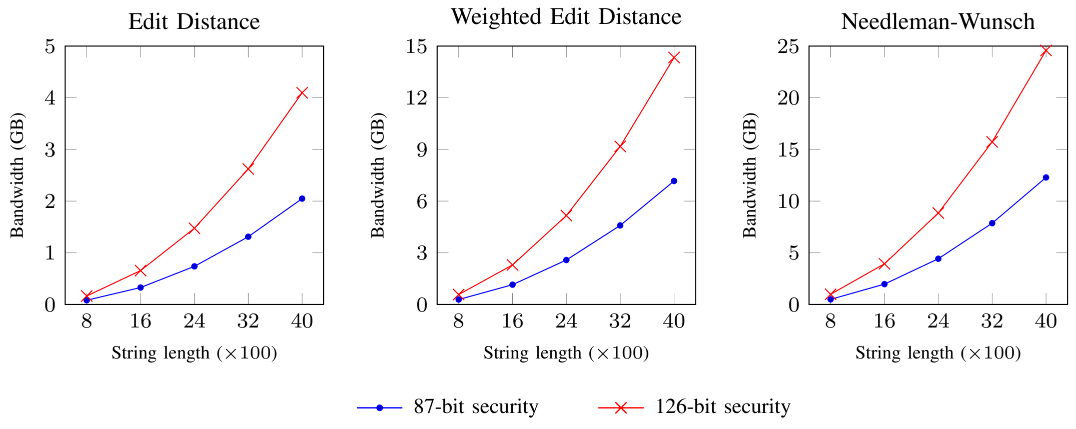


Fig. 7: Bandwidth cost of Edit Distance and its variants.

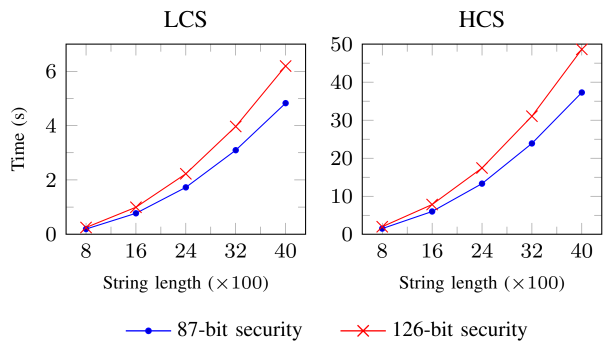


Fig. 8: Time cost of LCS and HCS. (Timings are averaged over 10 iterations.)

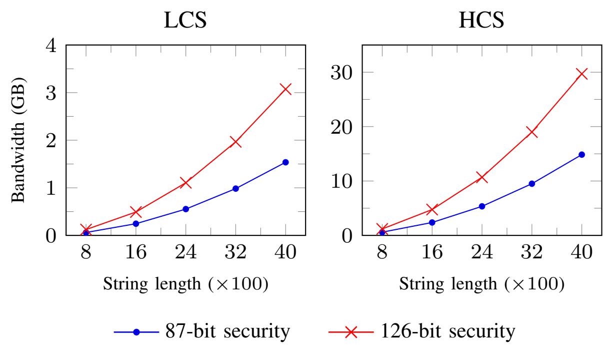


Fig. 9: Bandwidth cost of LCS and HCS.

(roughly 3500-nucleotides) in 3.7 seconds using 199 MB bandwidth. Thanks to flexible wire-label conversions, we can leverage Half-Gates garbling to obliviously sort the distance results to output the top-k most similar patients.

VI. Other Related Works

Secure Garbling. Ball, Malkin and Rosulek [21] have recently proposed a seminal secure garbling scheme that could circumvent the lower bound provided by Zahur et al [14]. Their work bears some similarities with our work, e.g., both works address the semi-honest threat model and support "free" additions and constant multiplications. Nevertheless, the two works differ significantly in their goals, techniques and concrete results that have been achieved.

First, their work aimed to improve the bandwidth efficiency for certain types of computations, i.e., high fan-in thresholding and mod-m ring arithmetic operations. Instead, our scheme is designed with both time and bandwidth efficiency in mind while targeted directly at realistic security challenges in secure genome and string comparisons.

Technical-wise, their work extended the traditional binary circuit garbling scheme by generalizing the wire-labels from a single $GF(2^{127})$ element to a 128-bit vector divided into entries of small prime field (i.e., $\mathbb{Z}_p$ where almost always $p \leq 103$) elements. To encode a plaintext value (e.g., a 32-bit integer), their scheme decomposes the plaintext value into its CRT-representation (Chinese Remainder Theorem) and requires a bundle of wire-labels each of which encodes one component of the CRT-representation. Hence, secure addition and constant multiplication in their scheme typically involves processing a bundle of wires and each wire-handling requires dozens of modulo additions/multiplications over small primes. In contrast, our approach is significantly simpler: we generalize the format of traditional binary field wire-labels directly to one over an extremely large prime field so that overflow will never be an issue when operating values in the realistic problem context. As a result, it requires only a single modulo operation to accomplish a secure addition or constant multiplication.

Moreover, their garbling scheme relies on the generalized point-and-permute technique for the evaluator to identify the "right" ciphertext to decrypt, whereas ours uses authentication tags to identify successful decryptions from failed trials. Although their approach saves the evaluator some CPU cycles, we stress that their point-and-permute trick is incompatible with the bounded-difference minimum gadgets that we introduce, since an adversary can infer the secret permutation by simply observing how the gap entries (i.e., the entries that do not correspond to a possible plaintext signal) move before and after the permutation, unless all gap entries are stuffed with dummy garbled rows (which will be unnecessarily more expensive).

Finally, while their garbling scheme is mostly of theoretical interest, without practical implementation, it is unclear whether it will actually bring performance benefit. In fact, garbling an entry with fixed-key AESNI instructions nowadays has become so cheap (roughly 10–50 ns) that even the so-called "free"-XOR gates (a few nano-seconds per gate) in traditional garbling schemes [14] are not really free. Computing dozens of mod-p additions/multiplications per "free" addition (or constant multiplication) is even much slower. The "free" additions and constant multiplications in their scheme is significantly more expensive than other non-free gadgets that requires just a few AESNI calls. In comparison, we experimentally verified our approach with a class of useful applications that our scheme is indeed more than an order-of-magnitude more efficient in both time and bandwidth than the state-of-the-art garbling schemes.

Private Edit Distance. Wang et al. [11] proposed a highly efficient private edit distance protocol that scales to human whole-genomes. They used some heuristics to convert a private edit distance problem into a set difference size problem, which is then solved by a sketch algorithm using secure computation. In comparison with our approach, the main differences are:

1) Our protocol is not constrained by the availability and any statistical properties of the reference strings, whereas their approach relies on some statistical traits of human genomes and require a "good" public reference genome to exploit the statistical properties of the inputs (such public reference string may not be available in general applications).
2) Our protocol always produces precise results on arbitrary input strings whereas their results on general inputs are far from accurate (though their protocol is significantly faster than ours).
3) We targeted a stronger model of security and functionality than their work. Their sketch algorithm requires a common public random tape to derive the results but it is unclear whether an attacker can learn extra information by relating the public randomness to the final results. Regarding functionality, it can be hard to use their protocol as part of a larger secure computation, e.g., when the input strings are results of an earlier oblivious computation, or the output needs to stay oblivious to be used in a later computation.

VII. Conclusion

Customizing a secure garbling scheme to leverage publicly exploitable traits of target computations can lead to secure computation protocols that are dramatically more efficient than traditional Boolean circuit based protocols. We have taken a first step to explore this methodology in the realm of constructing semi-honest protocols for obliviously computing several widely-used dynamic-programming-based string metrics including generalized edit-distance and weighted LCS. Our resulting protocols are an order-of-magnitude more efficient than their comparable state-of-the-art alternatives. We hope our findings shed some light on designing efficient application-specific secure computation protocols in the future.

Appendix

Let $\mathtt{s}, \mathtt{t}, D_{i,j}, c_{ins}, c_{del}, c_{sub}$ be defined as in Section II-B, where c_{ins}, c_{del} are generalized to one-dimensional tables and c_{sub} is generalized to a two-dimensional table. Let

$$m_{i,j} = \min\Big(D_{i,j-1} + c_{del}\big[\mathtt{t}[j]\big],\ D_{i-1,j-1} + c_{sub}\big[\mathtt{s}[i],\ \mathtt{t}[j]\big]\Big)$$
$$u_{i,j} = \Big(D_{i,j-1} + c_{del}\big[\mathtt{t}[j]\big]\Big) - \Big(D_{i-1,j-1} + c_{sub}\big[\mathtt{s}[i],\ \mathtt{t}[j]\big]\Big)$$
$$v_{i,j} = \Big(D_{i-1,j} + c_{ins}\big[\mathtt{s}[i]\big]\Big) - m_{i,j}$$

Then, there exist $D_{i,j}$-independent public constants C_1, C_2, C_3, C_4 such that for all valid indices i, j.

$$C_1 \le u_{i,j} \le C_2, \qquad C_3 \le v_{i,j} \le C_4.$$

Proof Because $|D_{i,j-1} - D_{i-1,j-1}| \le c_{ins}\big[\mathtt{s}[i]\big]$, therefore

$$D_{i-1,j-1} - c_{ins}\big[\mathtt{s}[i]\big] \le D_{i,j-1} \le D_{i-1,j-1} + c_{ins}\big[\mathtt{s}[i]\big]$$

so,

$$D_{i,j-1} + c_{del}\big[\mathtt{t}[j]\big] \ge D_{i-1,j-1} - c_{ins}\big[\mathtt{s}[i]\big] + c_{del}\big[\mathtt{t}[j]\big]$$
$$D_{i,j-1} + c_{del}\big[\mathtt{t}[j]\big] \le D_{i-1,j-1} + c_{ins}\big[\mathtt{s}[i]\big] + c_{del}\big[\mathtt{t}[j]\big]$$

hence,

$$\begin{aligned} u_{i,j} =& D_{i,j-1} + c_{del}\big[\mathtt{t}[j]\big] - \big(D_{i-1,j-1} + c_{sub}\big[\mathtt{s}[i],\ \mathtt{t}[j]\big]\big) \\ \ge& c_{del}\big[\mathtt{t}[j]\big] - c_{ins}\big[\mathtt{s}[i]\big] - c_{sub}\big[\mathtt{s}[i],\ \mathtt{t}[j]\big] \end{aligned} \tag{5}$$
$$\begin{aligned} u_{i,j} =& D_{i,j-1} + c_{del}\big[\mathtt{t}[j]\big] - \big(D_{i-1,j-1} + c_{sub}\big[\mathtt{s}[i],\ \mathtt{t}[j]\big]\big) \\ \le& c_{ins}\big[\mathtt{s}[i]\big] + c_{del}\big[\mathtt{t}[j]\big] - c_{sub}\big[\mathtt{s}[i],\ \mathtt{t}[j]\big] \end{aligned} \tag{6}$$

So we can set

$$C_1 := \min_{i,j}\big(c_{del}\big[\mathtt{t}[j]\big] - c_{ins}\big[\mathtt{s}[i]\big] - c_{sub}\big[\mathtt{s}[i],\ \mathtt{t}[j]\big]\big),$$
$$C_2 := \max_{i,j}\big(c_{ins}\big[\mathtt{s}[i]\big] + c_{del}\big[\mathtt{t}[j]\big] - c_{sub}\big[\mathtt{s}[i],\ \mathtt{t}[j]\big]\big),$$

and we have $C_1 \le u_{i,j} \le C2$.

Symmetrically, we can derive that

$$\begin{aligned} &\big(D_{i-1,j} + c_{ins}\big[\mathtt{s}[i]\big]\big) - \big(D_{i-1,j-1} + c_{sub}\big[\mathtt{s}[i],\ \mathtt{t}[j]\big]\big) \\ \ge& c_{ins}\big[\mathtt{s}[i]\big] - c_{sub}\big[\mathtt{s}[i],\ \mathtt{t}[j]\big] - c_{del}\big[\mathtt{t}[j]\big] \end{aligned} \tag{7}$$
$$\begin{aligned} &\big(D_{i-1,j} + c_{ins}\big[\mathtt{s}[i]\big]\big) - \big(D_{i-1,j-1} + c_{sub}\big[\mathtt{s}[i],\ \mathtt{t}[j]\big]\big) \\ \le& c_{ins}\big[\mathtt{s}[i]\big] + c_{del}\big[\mathtt{t}[j]\big] - c_{sub}\big[\mathtt{s}[i],\ \mathtt{t}[j]\big] \end{aligned} \tag{8}$$

(8) − (5) yields

$$\big(D_{i-1,j} + c_{ins}\big[\mathtt{s}[i]\big]\big) - \big(D_{i,j-1} + c_{del}\big[\mathtt{t}[j]\big]\big) \ge -2c_{del}\big[\mathtt{t}[j]\big] \tag{9}$$

(7) − (6) yields

$$\big(D_{i-1,j} + c_{ins}\big[\mathtt{s}[i]\big]\big) - \big(D_{i,j-1} + c_{del}\big[\mathtt{t}[j]\big]\big) \le 2c_{ins}\big[\mathtt{s}[i]\big] \tag{10}$$

Thus, we know from (7) and (9) that

$$\begin{aligned} v_{i,j} \ge \max\big(&c_{ins}\big[\mathtt{s}[i]\big] - c_{sub}\big[\mathtt{s}[i], \mathtt{t}[j]\big] - c_{del}\big[\mathtt{t}[j]\big], \\ &-2c_{del}\big[\mathtt{t}[j]\big]\big) \end{aligned}$$

and from (8) and (10) that

$$v_{i,j} \leq \max\left(c_{ins}\left[\mathtt{s}[i]\right] + c_{del}\left[\mathtt{t}[j]\right] - c_{sub}\left[\mathtt{s}[i], \mathtt{t}[j]\right],\right.$$
$$\left. 2c_{ins}\left[\mathtt{s}[i]\right]\right)$$

Finally, by defining

$$C_3 \coloneqq \min_{i,j}\Big(\max\left(c_{ins}\left[\mathtt{s}[i]\right] - c_{sub}\left[\mathtt{s}[i], \mathtt{t}[j]\right] - c_{del}\left[\mathtt{t}[j]\right], -2c_{del}\left[\mathtt{t}[j]\right]\right)\Big)$$
$$C_4 \coloneqq \max_{i,j}\Big(\max\left(c_{ins}\left[\mathtt{s}[i]\right] + c_{del}\left[\mathtt{t}[j]\right] - c_{sub}\left[\mathtt{s}[i], \mathtt{t}[j]\right], 2c_{ins}\left[\mathtt{s}[i]\right]\right)\Big)$$

we proved $C_3 \leq v_{i,j} \leq C_4$. ■

References

[1] C. G. A. Network *et al.*, "Comprehensive molecular portraits of human breast tumours," *Nature*, vol. 490, no. 7418, pp. 61–70, 2012.

[2] N. Waddell, M. Pajic, A.-M. Patch, D. K. Chang, K. S. Kassahn, P. Bailey, A. L. Johns, D. Miller, K. Nones, K. Quek *et al.*, "Whole genomes redefine the mutational landscape of pancreatic cancer," *Nature*, vol. 518, no. 7540, pp. 495–501, 2015.

[3] W. E. Evans and M. V. Relling, "Moving towards individualized medicine with pharmacogenomics," *Nature*, vol. 429, no. 6990, pp. 464–468, 2004.

[4] S. Forrest, S. A. Hofmeyr, and A. Somayaji, "Computer immunology," *Communications of the ACM*, vol. 40, no. 10, pp. 88–96, 1997.

[5] D. Gao, M. K. Reiter, and D. Song, "Behavioral distance for intrusion detection," in *International Workshop on Recent Advances in Intrusion Detection*, 2005, pp. 63–81.

[6] D. Gao, M. K. Reiter, and D. Song, "Behavioral distance measurement using hidden markov models," in *International Workshop on Recent Advances in Intrusion Detection*, 2006, pp. 19–40.

[7] S. B. Needleman and C. D. Wunsch, "A general method applicable to the search for similarities in the amino acid sequence of two proteins," *Journal of molecular biology*, vol. 48, no. 3, pp. 443–453, 1970.

[8] K. Tamura, D. Peterson, N. Peterson, G. Stecher, M. Nei, and S. Kumar, "Mega5: molecular evolutionary genetics analysis using maximum likelihood, evolutionary distance, and maximum parsimony methods," *Molecular biology and evolution*, vol. 28, no. 10, pp. 2731–2739, 2011.

[9] S. Jha, L. Kruger, and V. Shmatikov, "Towards practical privacy for genomic computation," in *IEEE Symposium on S&P*, 2008.

[10] Y. Huang, D. Evans, J. Katz, and L. Malka, "Faster Secure Two-Party Computation Using Garbled Circuits," in *USENIX Security Symposium*, 2011.

[11] X. S. Wang, Y. Huang, Y. Zhao, H. Tang, X. Wang, and D. Bu, "Efficient genome-wide, privacy-preserving similar patient query based on private edit distance," in *ACM CCS*, 2015.

[12] V. Kolesnikov and T. Schneider, "Improved garbled circuit: Free XOR gates and applications," in *ICALP*, 2008.

[13] M. Bellare, V. T. Hoang, S. Keelveedhi, and P. Rogaway, "Efficient garbling from a fixed-key blockcipher," in *IEEE Symposium on S&P*, 2013.

[14] S. Zahur, M. Rosulek, and D. Evans, "Two halves make a whole - reducing data transfer in garbled circuits using half gates," in *EUROCRYPT*, 2015.

[15] A. Malozemoff and X. Wang, "EMP-Toolkit," https://github.com/emp-toolkit, 2016.

[16] X. Jiang, B. Malin, L. Ohno-Machado, H. Tang, A. Telenti, X. Wang, S. Wang, and L. Xiong, "IDASH — Secure Genome Analysis Competition," http://www.humangenomeprivacy.org/2015/, 2015.

[17] A. Amir, Z. Gotthilf, and B. R. Shalom, "Weighted LCS," *Journal of Discrete Algorithms*, vol. 8, no. 3, pp. 273–281, 2010.

[18] A. C.-C. Yao, "How to generate and exchange secrets (extended abstract)," in *FOCS*, 1986.

[19] M. Bellare, V. T. Hoang, and P. Rogaway, "Foundations of garbled circuits," in *ACM CCS*, 2012.

[20] B. Pinkas, T. Schneider, N. P. Smart, and S. C. Williams, "Secure two-party computation is practical," in *ASIACRYPT*, 2009.

[21] M. Ball, T. Malkin, and M. Rosulek, "Garbling gadgets for boolean and arithmetic circuits," in *ACM CCS*, 2016.

[22] S. Gueron, Y. Lindell, A. Nof, and B. Pinkas, "Fast garbling of circuits under standard assumptions," in *ACM CCS*, 2015.

[23] S. Kumar, K. Tamura, and M. Nei, "Mega: molecular evolutionary genetics analysis software for microcomputers," *Computer applications in the biosciences: CABIOS*, vol. 10, no. 2, pp. 189–191, 1994.

[24] M. Kimura, "A simple method for estimating evolutionary rates of base substitutions through comparative studies of nucleotide sequences," *Journal of molecular evolution*, vol. 16, no. 2, pp. 111–120, 1980.

[25] F. Tajima and M. Nei, "Estimation of evolutionary distance between nucleotide sequences." *Molecular biology and evolution*, vol. 1, no. 3, pp. 269–285, 1984.

[26] "Amazon EC2 Instance Types," https://aws.amazon.com/ec2/instance-types, 2015.

[27] V. Kolesnikov and R. Kumaresan, "Improved OT extension for transferring short secrets," in *CRYPTO*, 2013.

[28] Y. Ishai, J. Kilian, K. Nissim, and E. Petrank, "Extending oblivious transfers efficiently," in *CRYPTO*, 2003.