Efficient and Precise Secure Generalized Edit Distance and Beyond

Ruiyu Zhu, Yan Huang, *Indiana University, Bloomington*

Abstract—Secure string-comparison by some non-linear metrics such as edit-distance and its variations is an important building block of many applications including patient genome matching and text-based intrusion detection. Despite the significance of these string metrics, computing them in a provably secure manner is very expensive. In this paper, we improve the performance of secure computation of these string metrics without sacrificing security, generality, composability, and accuracy. We explore a new design methodology that allows us to reduce the asymptotic cost by a factor of $O(\log n)$ (where n denotes the input string length). In our experiments, we observe up to an order-of-magnitude savings in time and bandwidth compared to the best prior results. We have also extended our semi-honest protocols to work in the malicious model.

Index Terms—secure string matching, secure genome comparison, secure edit-distance, secure Needleman-Wunsch, secure LCS.

1 Introduction

String comparison is a useful primitive that finds applications in many real-world scenarios. Among the metrics for comparing strings, many non-linear metrics such as edit distance are most interesting thanks to their versatility in adapting its cost model to field applications. For example, when genomes are denoted by strings, it is customary to use non-linear metrics such as weighted edit distance and Needleman-Wunsch distance [1] to help diagnosing genetic diseases [2], [3], [4]. In many other scenarios where the strings may represent file segments, sequences of system calls, or snippets of network traffic, these non-linear metrics are important enabling techniques of computer immunology [5] and intrusion detection [6], [7].

Often, the input strings in these applications carry highly sensitive information, thus are intended to stay encrypted throughout the computation. However, securely computing these non-linear metrics is a highly challenging research task. Researchers have studied intensively secure protocols to match strings based on edit distance, an epitome metric of its kind. When designing these protocols, several properties are vitally important.

First, one would prefer the protocols to be **generic**. This implies a number of desirable features: 1) The resulting protocol is ready to be used as a subroutine in another secure protocol using standard composition methods; 2) It is easy to modify the protocol to also work with other variants of string-metrics; 3) It allows to upgrade the security guarantees, e.g., from semi-honest to covert or malicious threat models, using well-known cryptographic techniques.

Second, it is desirable for the protocols to produce **accurate** results. Imprecise results can cause false decisions that will undermine the value of some security-critical systems. Secure protocols that can always provide accurate results irrespective of the secret input data can be used in many very different scenarios.

Third, the protocols are expected to be rigorously **proven secure and free of leakage**, which is necessary for safe use of such protocols in real-world applications.

Finally, we surely wish to have protocols as **efficient** as possible, such that they can be adopted in more performance-critical settings.

Unfortunately, existing protocols cannot yet provide a satisfactory solution to meet all the design expectations above. The heuristics-based protocols [8], [9] are efficient, but missed the first three design requirements entirely. On the other hand, while protocols using state-of-the-art generic garbled circuits [10] or ABY [11] are generic, accurate and proven-secure, they are very expensive in terms of cost.

1.1 Methodology and Threat Model

Motivated by the limitation of existing protocols, we ask:

> Can we design secure string comparison protocols that are as *secure*, *accurate* and *generic* as required by the standard definition of secure computation, while being *significantly more efficient* than the best existing generic solutions?

In this work, we answer this question with a new methodology. We adapt the garbling scheme itself to the public properties of target computations. In the context of computing string-comparison metrics, for example, we made two key observations and exploited them in our protocol design: (*a)* There are useful *public* patterns in such computations that correlate the secret values on intermediate wires. E.g., in edit distance, the two input numbers to the min circuit will differ by at most 1. (*b)* Many parts of string-comparison computations can be realized more efficiently using *arithmetic* (instead of binary) circuits. By exploiting these insights, we are able to securely compute a number of representative string-metrics significantly more efficiently than the best previous secure protocols.

Threat Models. In this work, we consider both semi-honest and malicious adversaries. We will discuss the semi-honest protocols first and then show how to upgrade them to thwart full-malicious attacks in Section 5.

1.2 Contributions

We propose a new design methodology for building efficient privacy-preserving computations. We customize the garbling to exploit public properties of the target computations. We apply this methodology in developing secure protocols

TABLE 1: Performance Highlights (semi-honest model)

	Edit Distance			Weighted ED			Needleman-Wunsch			LCS		
	Time		B/W	Time		B/W	Time		B/W	Time		B/W
	LAN	WAN		LAN	WAN		LAN	WAN		LAN	WAN	
Best Prior	286	1776	39.4	360	2257	50.1	1030	6747	155	202	1224	27.1
This Work	**23.7**	**178**	**4.09**	**83.3**	**625**	**14.3**	**142**	**1073**	**25.6**	**18.9**	**135**	**3.07**

Tested with 127-bit computational security. Times are in seconds and B/W in GB. Computation inputs are two 4000-nucleotide genomes. The weight tables used in Weighted ED and Needleman-Wunsch are given in Figure 1. "Best Prior" results are measured on efficient implementations based on the ideas of Huang et al. [12] and emp-toolkit [13], an updated framework integrating Free-XOR, AESNI, and Half-Gates. Detailed experiment setup is given in Section 6.

for several representative string-comparison metrics. Like the protocols of [10], [14], [15], our protocols work in the random oracle model. Unlike prior works, our approach leverages low-cost bounded-input comparison, minimum, and table-lookup, while keeping arithmetic addition free. The overall complexity of our secure string-comparison protocols is $O(n^2)$ (with n being the length of each input string), in contrast to $O(n^2 \log n)$ of prior protocols using best previous garbling schemes [10], [14]. We formally proved the security of our scheme (Section 3.2) and presented ways to extend our garbling schemes to handle arbitrary functions through tethering it to binary garbled circuits (Section 4).

We have strengthened the semi-honest protocols into efficient actively-secure string-metrics computation protocols (Section 5), which are by far the best of its kind. Equipped with state-of-the-art cut-and-choose strategies, the cut-and-choose parameters of our protocols can be selected based on the actual cost ratio between checking and evaluating a GC for improved performance [30]. Security of our protocols can be guaranteed as long as one correct GC is evaluated.

We have experimentally evaluated our approach on a range of string-comparison metrics including edit distance, weighted edit distance, Needleman-Wunsch distance, LCS, HCS. In the semi-honest model, our protocols are able to run up to 16 times faster and use an significantly less bandwidth than best existing GC-based protocols (see Table 1). Unlike the heuristics-based protocols [8], [9], our approach is generic, accurate, and proven-secure, and does *not* use any public reference. As a first step in this direction of research, our findings would shed some light on designing other application-specific MPC protocols in the future.

1.3 Related Work

1.3.1 Heuristics-based private string matching

Researchers have proposed some interesting heuristics to *approximate best matches* of *low-entropy* strings by their edit-distances. Two seminal works of this kind are by Wang et al. [9] and Asharov et al. [8]. Wang et al. estimated edit-distance of human genomes through solving set-difference-size problems that were efficiently sketched using a public reference genome. Asharov et al. divided genome strings into short segments then approximated edit-distance-based match.[1] Although these protocols offered high efficiency, they also suffered from leakage, accuracy, and generality issues: (a) They assume a weaker threat model that *does* leak more than what is allowed by the standard definition of secure computation, while it is hard to argue that the leaked information is not what an attacker wanted. (b) They produce input-data-specific errors with their results, making them inapplicable in scenarios where errors are less tolerable. (c) It is hard to use these protocols as generic building blocks in other secure protocols or on inputs other than low-entropy genomes. It is neither clear how to modify them to compute other variant string-metrics such as Needleman-Wunsch, LCS, or to work against stronger adversaries. Moreover, both protocols rely a "good" public reference string, which may not be available in many use cases. We note that the quality of the reference string can severely affect the accuracy and cost of these protocols (see experiments in Section 6.1), while methods of picking "good" references were yet to be studied.

Finally, those protocols worked only in the honest-but-curious threat model and can be hard to convert to ones secure in the presence of active attacks.

1. The protocol of [8] can't really calculate edit-distance, but aimed at computing the closest matches under the edit-distance metric (a task that doesn't necessarily require computing edit-distances).

1.3.2 Garbled-circuit-based approach

Generic protocols using optimized garbled circuits (GC) were also used to compute edit-distance [12], [16]. This type of protocols always produce accurate results, offering strong security guarantees satisfying the standard definition of security for MPC, and are generally applicable to other string-comparison metrics. In addition, these GC-based protocols can be used as black-box components in larger secure computations. There are standard practical transformations to upgrade these protocols to work in the presence of active adversaries.

On the flip side, the costs of such protocols are prohibitive, partly because of the large constant factor blowup from translating the computation into binary circuits. We have implemented GC-based protocols to securely compute edit distance, weighted edit distance, Needleman-Wunsch, LCS and HCS. In the baseline implementation, we used all applicable state-of-the-art optimizations including fixed-key hardware AES [17], [18], Half-Gate garbling [10], and free-XOR technique [19]. The performance of these protocols is reported as "Best Prior" row in Table 1 and the performance charts of Figure 3, Figure 4 in Section 6.1. These baseline performance numbers are already significantly better than any generic protocols found in the literature, since we used

all possible state-of-the-art optimizations. Still, their performance wouldn't be satisfactory in many practical settings.

1.3.3 Comparison with Ball-Malkin-Rosulek [14]

Ball, Malkin and Rosulek proposed a garbling scheme where plaintext signals are encoded in their CRT-representations (Chinese Remainder Theorem). The CRT-representation encodes a plaintext value as elements in field $\mathrm{GF}(p_1 \times \cdots \times p_n)$ where $p_1, \ldots, p_n$ are a number of distinct small primes. Their scheme can be considered as an extension of Half-Gate's wire-label encoding, which is over the field $\mathrm{GF}(2^\kappa)$, with general projection gates. They show with calculation that this garbling scheme can be of theoretical interest in saving bandwidth for certain computations that consist of many high fan-in threshold and modular addition gates. However, they did not consider any practical end-to-end secure computation protocols and practical time efficiency. In contrast, we focus on a class of important string comparison metrics computations, discovering some key properties of these computations, and advocate customizing the protocol to leverage the public properties for efficiency improvements. Our work considers both semi-honest and malicious adversaries. We show with experiments that our protocols are up to an order-of-magnitude better in both time and bandwidth than best existing generic protocols.

In fact, an important technical distinction between the two works is that their garbling schemes rely on the *point-and-permute* technique to evaluate the garbled gates, while we use *zero-tag*s to allow the evaluator to identify failed trial-decryptions. The point-and-permute technique is not compatible with the bounded-value projection technique which has significantly boosted the performance of our protocols. This is because if any garbled rows skip the transmission, the point-and-permute mechanism allows the evaluator to learn something about the secret permutation from observing how the (publicly-known) omitted entries are moved before and after the permutation.

1.3.4 Comparison with ABY [11] and ABY3 [20]

The ABY framework enables generic secure computations using one or a mixture of GMW-based **A**rithmetic/**B**inary circuits and **Y**ao's binary GC. However, it won't outperform our protocols since it does not support bounded-value projection. Nor does it offer convenient upgrade to malicious model security as our protocols do. In fact, ABY cannot even outperform our GC-based baseline (see Section 6.3).

In comparison to ABY's secret-share conversion techniques, we stress that our GC-based arithmetic encoding is very different from ABY's GMW-based arithmetic encoding. Henceforth, the conversion methods we present in Section 4.1 differs from those of ABY in essential ways.

Mohassel and Rindal extended ABY to ABY3 for machine learning computations. However, ABY3 only works for a completely different threat model (3PC with honest majority) which is out of the scope of this paper.

1.3.5 Comparison with DKS+ [21]

The work by Dessouky et al. suggests that some custom-built circuits could benefit from efficient OT extension specific for short messages by a constant (and application-dependent) factor (2–4x for AES and PSI) of savings. However, they didn't consider the string metrics that we study here and their idea doesn't support bounded-value projection.

2 Background

Notations. We let κ be the computational parameter; let "$a := b$" denote assigning the value of b to a; and let "$x \leftarrow S$" denote assigning to x a uniformly element of the set S.

2.1 Secure Garbling

First proposed by Yao [22], garbled circuits were later formalized by Bellare et al. [18] as a cryptographic primitive of independent interest. Following the notations of Bellare et al., 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 $\mathcal{S}_{\mathsf{prv}}$ such that for all x,

$$\left\{(F, X, d) : \begin{array}{l}(F, e, d) \leftarrow \mathsf{Gb}(1^k, f),\\ X \leftarrow \mathsf{En}(e, x).\end{array}\right\} \approx \left\{\mathcal{S}_{\mathsf{prv}}(1^k, f, f(x)\right\}.$$

  where "$\approx$" symbolizes computational indistinguishability.

- **Obliviousness:** There exists an efficient $\mathcal{S}_{\mathsf{oblvs}}$ such that $\forall x$,

$$\left\{(F, X) : \begin{array}{l}(F, e, d) \leftarrow \mathsf{Gb}(1^k, f),\\ X \leftarrow \mathsf{En}(e, x).\end{array}\right\} \approx \left\{\mathcal{S}_{\mathsf{oblvs}}(1^k, f)\right\}.$$

- **ϵ-Authenticity:** For all efficient $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$,

$$\Pr\left(\begin{array}{c}Y \neq \mathsf{Ev}(F, X)\\ \textbf{and}\\ \mathsf{De}(d, Y) \neq \perp\end{array} : \begin{array}{l}(f, x) \leftarrow \mathcal{A}_1(1^k),\\ (F, e, d) \leftarrow \mathsf{Gb}(1^k, f),\\ X \leftarrow \mathsf{En}(e, x),\\ Y \leftarrow \mathcal{A}_2(1^k, F, X).\end{array}\right) \leq \epsilon.$$

Optimizations have been proposed and improved garbling in many aspects such as bandwidth [10], [14], [23], evaluator's computation [23], memory consumption [12], and using dedicated hardware [10], [17], [24]. 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], [17], [24].

2.2 Edit Distance and Other Metric Variants

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 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, $c_{ins} \coloneqq 1$, $c_{del} \coloneqq 1$, and $c_{sub} \coloneqq (\mathtt{s}[i] = \mathtt{t}[j])\ ?\ 0 : 1$, 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.

Algorithm 1 EditDistance(s, t)

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

Weighted Edit Distance. More generally, the c_{ins}, c_{del}, and c_{sub} above can be adjusted to fit the goals of specific applications. For example, in diagnosing certain genetic diseases [2], [4], 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 more rarely than substitution in some application domain so one would adjust the costs so that the changes are better captured by the editing model. For example, during DNA replication, *indel*s are much rarer than substitutes, so we would expect a good alignment to contain proportionally less *indel*s to reflect the natural clone of DNAs.

Needleman-Wunsch. As the statistical models of various operations were refined with respect to the symbols involved in the mutations, researchers [25], [26], [27], [28] have found many good reasons to also adjust the costs $c_{ins}, c_{del}, c_{sub}$ according to the specific characters to be 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.

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

Longest Common Subsequence (LCS). 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 two 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 represents the length of LCS($\mathtt{s}[1..i]$, $\mathtt{t}[1..j]$).

Algorithm 2 Longest common subsequence(s, t)

1: **for** $i \coloneqq 0$ to length(s) **do**
2: $\quad \mathtt{D}_{i,0} \coloneqq 0$;
3: **for** $j \coloneqq 0$ to length(t) **do**
4: $\quad \mathtt{D}_{0,j} \coloneqq 0$;
5: **for** $i \coloneqq 1$ to length(s) **do**
6: $\quad$ **for** $j \coloneqq 1$ to length(t) **do**
7: $\quad\quad \mathtt{D}_{i,j} \coloneqq \mathtt{max}(\ \mathtt{D}_{i-1,j},\ \mathtt{D}_{i,j-1},\ \mathtt{D}_{i-1,j-1} + w_{i,j})$;

With basic LCS, the matching reward, $\mathtt{w}_{i,j}$, is set to

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

Heaviest Common Subsequence (HCS). As a generalization of LCS, researchers [29] have introduced the concept of *heaviest common subsequence*, just like Needleman-Wunsch generalizes edit distance. The idea is to let different characters reward differently when they match. Therefore, $\mathtt{w}_{i,j}$ can be viewed as a matrix (to be indexed by $\mathtt{s}[i]$ and $\mathtt{t}[j]$) where only the diagonal entries will be positive while the rest of the matrix are filled by 0s.

3 The Semi-Honest Model

Next, we give our semi-honest protocols for efficiently computing string-comparison metrics.

3.1 Insights and Intuitions

First, we illustrate two important observations behind the design of our new garbling scheme.

Dominant Costs. A dominant cost of solving the general edit distance problem lies in the oblivious computation of *addition*, *equality* (or *table-lookup* in general), *minimum*. This is evident from the dynamic programming Algorithm 1. Therefore, it should be our foremost priority to make these oblivious computations efficient in our new garbling scheme.

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., $\min(a, b, c) = \min\big(\min(a, b),\ c\big)$. A key observation is that edit distances can be calculated such that all two-minimum gates are computed on such inputs (a, b) that $a - b$ is bounded by some constants independent of the absolute values of a and b. This observation opens up an opportunity to speed up private edit distance computation. We exploit this opportunity by designing special two-minimum gadgets which only need to work for inputs of bounded difference, but runs significantly more efficient

than generic minimum gates (that need to process all possible inputs).

Take basic edit distance as an example. We can show that every call to $\min(a, b)$ can be arranged so that $a - b \in \{-1, 0, 1, 2\}$. We can prove this fact as follows. 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

$$|D_{i-1,j} - D_{i-1,j-1}| \le 1, \quad (3)$$
$$|D_{i,j-1} - D_{i-1,j-1}| \le 1. \quad (4)$$

Thus,

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

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

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

Since

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

thus,

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

Finally, we have

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

Therefore, both constraints (1) and (2) must hold.

Generally, observations like the one above can also be shown for many other string-comparison metrics. Next, we state our general proposition of this insight, which we formally prove in Appendix B. We note that, unlike the example above, our proof for the general case does *not* rely on the triangle inequality property of the metrics.

Proposition 1. *Let* $\mathtt{s}, \mathtt{t}, D_{i,j}, c_{ins}, c_{del}, c_{sub}$ *be defined as in Section 2.2, 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 public constants C_1, C_2, C_3, C_4 *which are independent of* $D_{i,j}$*, such that for all valid indices* i, j*.*

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

3.2 The Garbling Scheme

Basic Idea. Since these computations only deal with integers, we generalize the idea of garbling binary signals to work directly on arithmetic signals. Recall that when garbling binary circuits, the garbler picks, for every wire in the circuit, a secret string $w_0 \leftarrow \{0,1\}^{128}$ to encode 0 and sets $w_1 \coloneqq w_0 \oplus \Delta$ to encode 1 (where Δ is a circuit-global secret uniformly sampled from $\{0,1\}^{127}$). To generalize this idea, we replace "$\oplus$", the adder on the binary field, with "$+_p$", the adder on the prime field $\mathbb{Z}_p$ (where p is public and sufficiently large, e.g., $p > 2^{87}$). In our scheme, the garbler will first pick a uniform global secret Δ from $\mathbb{Z}_p$. Then, for every wire in the arithmetic circuit, the garbler picks a uniform k_0 (called *wire-key*) from $\mathbb{Z}_p$ to denote 0; and encode every integer $a \in \mathbb{Z}_p$ as $k_a = k_0 +_p a \times_p \Delta$ where "$+_p$" and "$\times_p$" denote mod-p addition and multiplication, respectively.

To garble a gate, the garbler would use encoding of a gate's every possible input signal as a key to encrypt the encoding of its corresponding output signal; to evaluate the gate, the evaluator will decrypt *every* garbled row of the gate. To allow the evaluator to tell which row decrypts successfully, we add a constant tag of sufficient length to every wire-key k_a to form a *wire-label*. Thus, it is the output wire-labels (rather than wire-keys) that are actually encrypted.

If the zero-tags are short (e.g., 40-bits), one might worry that a wire-label could happen to successfully decrypt more than one garbled row in the same gate due to collision, which violates the correctness property of garbling. However, to semi-honest attackers, who cannot leverage side-computation to affect protocol execution, the length of the zero-tags is actually a *statistical* security parameter. To malicious attackers, the issue can be addressed, either by increasing the length of zero-tags (Section 4.2), or by fixing the random-tape of the Gb function to a collaboratively coin-tossed bit-string (so the garbler cannot precompute and cherry-pick a particular random-tape to produce a problematic garbled gate).

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 (A) indicates the id of the wire to which this wire-label is associated and the subscript (a) indicates the plaintext signal that the wire-label encodes. When the wire name is irrelevant to a discussion, the superscript can be omitted. In our terminology, generating (or sampling) a fresh wire-label, say w_a^A, for a plaintext value a means first picking $k_0^A \leftarrow \mathbb{Z}_p$ (unless k_0^A is already known)

then setting $k_a^A := k_0^A +_p a \times_p \Delta$ 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.

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

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

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

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

$$w_a^A = 0^{40} \| (k_0^A +_p a \times_p \Delta)$$

with a public constant c can simply be realized as:

1) the garbler sets $w_0^C = c \times_p w_0^A$; 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 public-constant multiplication) is also free—no expensive cryptographic computation nor network traffic is used—but only a mod-p addition (or mod-p multiplication, respectively) on each side of the protocol.

Equality. When computing c_{sub}, an equality test is needed to decide whether two input characters are identical. Let $a, b \in \{0, 1, \ldots, \zeta\}$ be two integers, and w_a, w_b are the wire-labels corresponding to a and b, respectively. To securely compute if a equals b, first $d = a - b$ is securely computed, hence the garbler knows k_0^D and Δ while the evaluator knows $w_d^D = 0^{40} \| (k_0^D +_p d \times_p \Delta)$. Then, since $d \in \{-\zeta, \ldots, \zeta\}$,

1) the garbler samples a fresh pair of wire-labels w_0^Z and w_1^Z to encode signal 0 and 1 on the output-wire Z; and sends the following $2\zeta + 1$ garbled rows

$$\mathbf{Enc}_{w_0^D}(w_1^Z, id); \text{ and}$$
$$\mathbf{Enc}_{w_i^D}(w_0^Z, id), \forall i \neq 0, i \in \{-\zeta, \ldots, \zeta\}$$

in a randomly permuted order. Note that id_Z is the identifier of this projection gate.

2) the evaluator tries to decrypt the above $2\zeta+1$ ciphertexts using w_d^D as the key. Thus, only the ciphertext encrypted with key w_d^D will be successfully decrypted to reveal the valid wire-label w_z^Z encoding $(a = b)?1 : 0$.

Namely, the evaluator will learn w_1^Z if and only if $a = b$; and otherwise, will learn w_0^Z.

The cost of the 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, when $a - b$ can be bounded by a constant (for application-specific reasons), our approach reduce can reduce the cost by a factor of $\min(\log a, \log b)$.

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 realized using the same idea of secure projection like in the equality gadget above. Let X, Z be the input and output wires, respectively, and assume $x \in \{-\zeta, \ldots, \zeta\}$, the garbler simply sends the following $2\zeta + 1$ ciphertexts in a randomly permuted order:

$$\mathbf{Enc}_{w_i^X}(w_0^Z, id), \forall i \in \{-\zeta, \ldots, -1\}; \text{ and}$$
$$\mathbf{Enc}_{w_i^X}(w_i^Z, id), \forall i \in \{0, \ldots, \zeta\}$$

where for $0 \leq i \leq \zeta$, w_i^Z is the wire-label representing plaintext value i on the wire Z. When a, b are large but $|a - b|$ is bounded by some constant (which is indeed the case for the string metrics considered in this paper), we can save a factor of $\min(\log a, \log b)$ than traditional garbling.

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 $w_{v_i}^V$ that actually encodes v_i. In our scheme, this can be straightforwardly realized as follows:

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

$$\mathbf{Enc}_{w_i^I}(w_{v_i}^V, id), \ \forall i \in \{0, \ldots, n-1\}$$

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 $w_{v_i}^V$ 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 $k = i * m + j$

2. For every specific computation, 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 4.

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 as the treatment to securely look up 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 [12]. However, it is unclear how this can be efficiently realized 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.

Handle 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, a set of addition and public-constant multiplication gadgets will be used to translate the bits representation of input values into their arithmetic representations.

Implementation. Today's high-performance garbling schemes rely heavily on ideal block ciphers instantiated with *fixed-key* AES. Our scheme can also leverage fast fixed-key AES garbling accelerated by AESNI. For all the building blocks, 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. Similar to Half-Gates [10], 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 GF(2^{128})) and π is a random permutation realized using fixed-key AES. We can implement $\mathbf{Dec}_{w_{in}}(i, c)$ as

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

where K is as defined before.

3.3 Formal Analysis

Complexity. With our approach, the dominating cost can be attributed to the projection gates (used in computing the minimum and equality). For edit-distance, to compute each entry of the n^2-entry dynamic programming matrix, only two projection gates are needed: one 8-row projection for equality and another 8-row projection for minimum. So overall, the cost is $16n^2$ garbled rows. In comparison, using Half-Gate's [10] garbling to compute edit-distance, computing each entry of the n^2 DP matrix requires a fixed-width equality gate (2 garbled rows), two variable-width minimum gates ($2 \log n \times 2$ rows), and one variable-width addition gates ($2 \log n$ rows), totaling at $(6 \log n + 2)n^2$ garbled rows. With Ball-Malkin-Rosulek, even if additions are ignored, the cost per matrix entry will still be $c_1 \log n$ rows for equality plus $c_2 \log n$ rows for minimum (where c_1, c_2 are fairly large constants depending on the choice of the CRT-representation), totaling at $(c_1 + c_2)n^2 \log n$ rows. Therefore, our approach brings $\log n$-factor savings over best existing generic garbling schemes.

Correctness. We formalize our garbling scheme in Figure 2. It is easy to verify that *correctness* of this garbling scheme fails only if more than one row in the same gate decrypt to valid (but different) wire-labels. To semi-honest attackers, the length of the zero-tags provides statistical security. Thus correctness fails only when multiple *honestly-garbled* rows in the same gate *happen* to decrypt to different wire-labels each with a valid zero tag (such that an evaluator would be confused), which is bounded by 2^{-40} in each gate. One might also worry that large circuits may be more likely to fail since a circuit with a $|C|$ non-free gates might fail with probability $\sum_{i=1}^{|C|} 2^{-40} \cdot n_i$ (this is actually a loose upper-bound) where n_i is the number of rows in the i^{th} gate. However, this is not the case because for every internal gate, the evaluator can always try all seemingly-valid wire-labels in a subsequent gate to eliminate such spurious wire-labels. So a spurious wire-label can only propagate with $2^{-40}n_1 \cdot 2^{-40}n_2$ probability at most, where n_1, n_2 are the number of garbled rows in the two connected gates, respectively. Since the number of rows in every garbled gate is bounded by a small constant in practice, we have $2^{-40}n_1 \cdot 2^{-40}n_2 \ll 2^{-40}$. Therefore, the overall failure probability only depends on the final layer of gates, that is, at most $\sum_{i=1}^{|C_{out}|} 2^{-40} \cdot n_i$ where $|C_{out}|$ is the number of non-free gates in the final layer and n_i is the number of rows in the i^{th} final-layer-gate. Since $|C_{out}|$ is typically a small constant in MPC applications (e.g., $4 \leq |C_{out}| \leq 20$ to denote the output distance/score in private string-comparison applications), our garbling schemes are correct except for a negligible probability (in concrete sense). In Section 4.2, we give a technique to increase both the computational and statistical security to 127-bit, which will address the concern even in presence of malicious attackers.

Security. Note that equality, minimum and table-lookup gadgets are all essentially realized by a primitive operation called *projection*. Secure projection obliviously maps an input signal a_i to a predefined output signal b_i based on a publicly table $\{(a_1, b_1), \ldots, (a_n, b_n)\}$. Thus, to prove the garbling scheme to be secure, it suffices to just consider addition and projection.

Theorem 1. *If π is an ideal block cipher that is used to realize* **Enc** *and* **Dec** *as described above. the scheme in Figure 2 satisfies the privacy and obliviousness definitions given in Section 2.1, and*

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

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

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

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

Fig. 2: The garbling scheme

an application-dependent notion of authenticity.

Proof of Theorem 1 is given in Appendix A.

4 Extensions

In this section, we discuss three extensions of our approach: one for garbling arbitrary computations, the second for increasing the security parameters, and the third for achieving application-independent authenticity.

4.1 Garbling Arbitrary Computations

Our garbling scheme as is described so far can't handle generic computations because we haven't discussed how to *multiply* two secret values efficiently. To efficiently handle arbitrary computations, our basic idea is to tether the above scheme with a traditional binary circuit garbling such as Half-Gate.

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 $w_{i,a_i} = w_{i,0} \oplus a_i\Delta$. We describe two methods to accomplish this goal that exhibit complementary tradeoffs between performance and generality.

4.1.1 *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 [10] to run any followup computation over a by starting from their respective shares m and $a \oplus m$.

To convert an arithmetic wire, it costs $\zeta + 1$ 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 an alternative conversion method suitable for large ζs.

4.1.2 Via generic secure modular-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 binary encodings $w'_{a_0}, \ldots, w'_{a_n}$ to arithmetic encodings $w_{a_0}, \ldots, w_{a_n}$, then w_a can be derived from $w_{a_0}, \ldots, w_{a_n}$ through local constant multiplication and local addition.

4.2 Increase Security Parameters

The scheme as we described thus far only guarantees 87 bits computational and 40 bits statistical security for semi-honest adversaries. Next, we show how to modify our scheme to provide 127 bits computational and 128 bits statistical security for semi-honest adversaries (or 128 bit computational security for malicious adversaries).

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 add to each garbled row $\mathbf{Enc}_{w_{in}}(i, w_{out})$ a 128-bit tag. That is,

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

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 GF(2^{128}) and π is an ideal block cipher realized by 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

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

for all but a negligible probability.

4.3 Application-Independent Authenticity

The authenticity of the garbling scheme described before (as well as the CRT-based garbling scheme of Ball-Malkin-Rosulek) is application-dependent since its authenticity-error n/p can grow with n (see proof of Theorem 1), the size of the application-dependent plaintext-domain. To provide the standard, application-independent notion of authenticity, we can modify our garbling scheme so that every wire's plaintext value a is encoded as a pair $\left(k_0 +_p a \times_p \Delta, \hat{k}_0 +_p a \times_p \hat{\Delta}\right)$ where $\Delta, \hat{\Delta}$ are the garbler's two independently-sampled, circuit-global secrets and $k_0, \hat{k}_0$ are the garbler's two independently-sampled wire-specific secrets. In more detail,

- **Encode.** To encode a plaintext value $a \in \mathbb{Z}_p$, the garbler picks uniform $k_0, \hat{k}_0, \Delta, \hat{\Delta} \in \mathbb{Z}_p$ and computes

  $$\mathsf{L}_a := \left(k_0 +_p a \times_p \Delta, \hat{k}_0 +_p a \times_p \hat{\Delta}\right)$$

  as the encoding (i.e., wire-label) of a.

  If the garbler (who knows $k_0, \hat{k}_0, \Delta, \hat{\Delta}$) receives an encoding $\mathsf{L} = \left(L, \hat{L}\right)$, to check the validity of the encoding, he/she will verify

  $$(L - k_0) \times_p \Delta^{-1} = \left(\hat{L} - \hat{k}_0\right) \times_p \hat{\Delta}^{-1}$$

and decode L to $(L - k_0) \times_p \Delta^{-1}$ only if the above equality holds.

- **Addition.** Since the encoding is additively homomorphic, given two encodings L_a and L_b, the encoding of their sum can be locally computed as $\mathsf{L}_{a+b} := \mathsf{L}_a +_p \mathsf{L}_b$.
- **Projection.** To garble a projection $(v_1 \mapsto u_1, \ldots, v_t \mapsto u_t)$, a t-row garbled gate is computed as follows:

$$\mathbf{Enc}_{\mathsf{L}_{v_1}}(\mathsf{L}_{u_1}), \quad \mathbf{Enc}_{\mathsf{L}_{v_2}}(\mathsf{L}_{u_2}), \quad \ldots, \quad \mathbf{Enc}_{\mathsf{L}_{v_t}}(\mathsf{L}_{u_t}).$$

Theorem 2. *The improved garbling scheme of Section 4.3 satisfies the privacy, obliviousness, and authenticity properties outlined in Section 2.1.*

Proof of Theorem 2 can be found in Appendix C.2.

5 The Malicious Model

In this section, we give a general approach to compile our semi-honest protocols into ones secure against malicious adversaries. We consider the standard definition of active-security of secure two-party computation with respect to the standard ideal model execution: the trusted party, upon receiving input string x and y from party P_1 and P_2, respectively, computes the agreed string metric between x and y and sends the result to P_2.

Protocol Design Intuition. We use the *cut-and-choose* technique, where the circuit generator sends n garbled circuits to the evaluator, k of which will be checked and the rest will be evaluated to derive the final outcome. For improved performance, we used the probabilistic cut-and-choose strategy of [30] to fix n but pick k from a public distribution, based on the observed cost ratio between checking and evaluation per GC. Also, the garbler sends only hashes of GCs in the "garble" step to save bandwidth, but re-generates the evaluation GCs in the "evaluate" step. Our protocol succeeds as long as at least one of the evaluation circuit is correctly generated. Due to page limit, we describe our malicious model protocols in Appendix C.1 and formally prove its security in Appendix C.3, but state Theorem 3 below for completeness.

Theorem 3. *The protocol of Appendix C.1 securely computes f in presence of malicious adversaries.*

6 Evaluation

In this section, we evaluate a set of secure string comparison protocols which motivated our work.

Experiment Setup. We used two `n1-standard-1` instances (1vCPU, 3.75GB memory, priced at 3 cents/hour) on Google Cloud Platform. The LAN setting has 2Gbps with $1ms$ latency. The WAN has 200Mbps with $40ms$ latency. The computational security parameter κ is 127 and the statistical security parameter $s = 40$. Unless specified otherwise, the performance numbers are averaged over 10 runs.

We implemented our scheme in C/C++, using Intel AESNI intrinsic instructions to realize the fixed block cipher π. We use emp-tool [13]'s implementation of Half-Gate [10] garbling and efficient OT extension [31], [32] to construct the baseline protocols to compare with. For fair comparison, all baseline protocols use their best possible custom circuits optimized to take advantage of free-XOR [19] benefits.

6.1 Application Performance

We applied the proposed garbling scheme to implementing five string metrics: edit distance, weighted edit distance, Needleman-Wunsch, longest common subsequence (LCS), and heaviest common subsequence.

Table 1 highlighted the performance improvements of our semi-honest protocols in comparison with best previous results. Generally, the gains of our approach are slightly bigger on Edit Distance and LCS, since the choices of weights can affect the sizes of the projection gates (as indicated by Proposition 1). We observe that running times on LAN and WAN all conform very well with the linear cost model:

$$\mathrm{Time}_{\mathsf{overall}} = \mathrm{Time}_{\mathsf{computation}} + \mathrm{Size}_{\mathsf{traffic}}/\mathrm{Speed}_{\mathsf{network}}.$$

Thus, in the scalability experiments below we will focus on the LAN setting.

Figure 3 and Figure 4 delineate the time and bandwidth 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 asymptotic complexity of the underlying dynamic programming algorithms. We set c_{ins} and c_{del} to 5 and c_{sub} to 1 for Weighted-ED. Our Needleman-Wunsch used the weight tables of Figure 1.

6.2 Comparison with [9] and [8]

These heuristics-based protocols are still more efficient than our protocols. However, those protocols are only able to approximate certain computation over very restricted sets of low-entropy strings and are not provably secure with respect to the standard definition of security for MPC protocols. It is also crucial to select a "good" reference string since as the accuracy of these heuristic protocols can be very sensitive on the choice of the reference strings. However, no secure methods to choose "good" reference strings were known.

The quality of an approximation method can be measured by Root-Mean-Square Relative-Error (RMSRE)

$$\sqrt{\left(\sum_{i=1}^{n}[(v_i - u)/u]^2\right)/n}$$

where $\{v_1, \ldots, v_n\}$ are n approximations of a ground-truth value u using the method. Typically, approximation methods with RMSRE greater than or equal to 50% are not usable in most real-world string-comparison applications. We run an experiment over the same dataset used by [8], [9], where we picked uniformly 2000 pairs of 3500-nucleotide genome strings, computed the edit-distances between them using the protocols of [8], [9] with a randomly generated reference string. We observed a RMSRE of 75% and 59% using [9]'s and [8]'s approach, respectively. Both numbers clearly indicate serious accuracy issues of applying their methods in practice.

In contrast, our approach doesn't require any public reference string to work, can always produce accurate results, and can work for many variant string metrics over arbitrary strings. However, without knowing how to pick good reference strings, it is not possible to draw meaningful performance comparisons, even merely for the standard edit-distance case.

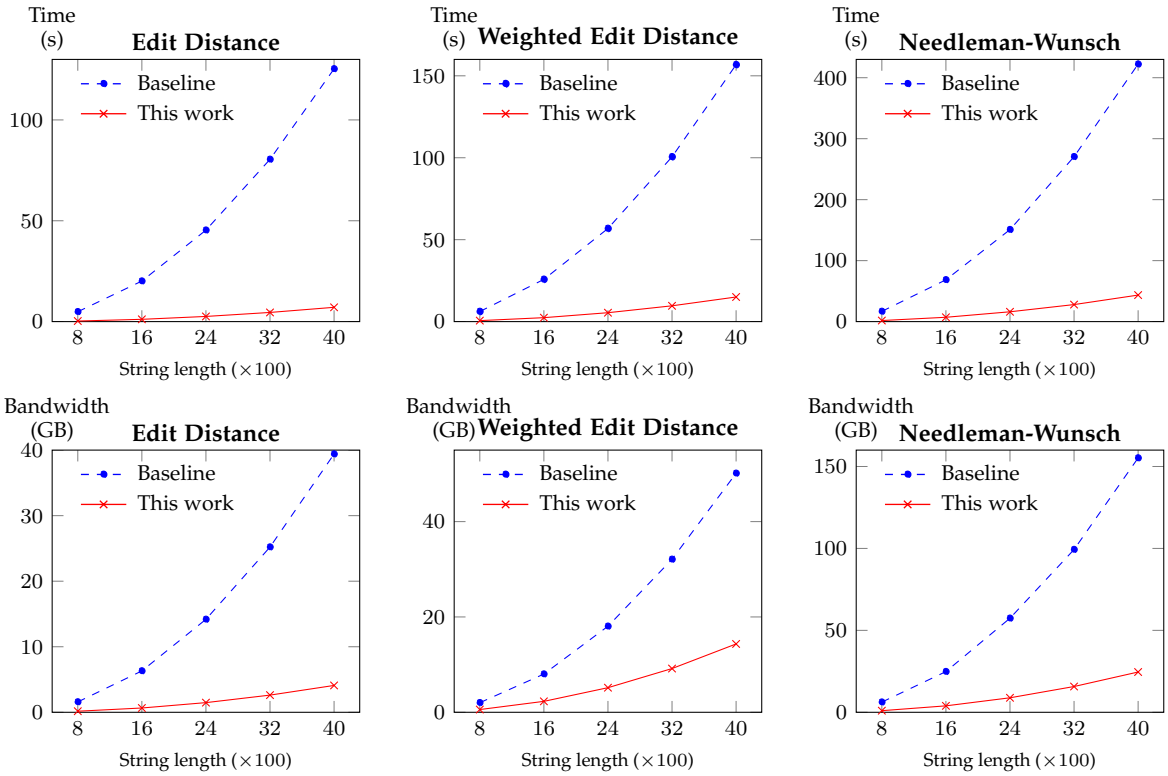


Fig. 3: Edit Distance, Weighted Edit Distance, and Needleman-Wunsch. ($\kappa = 127$)

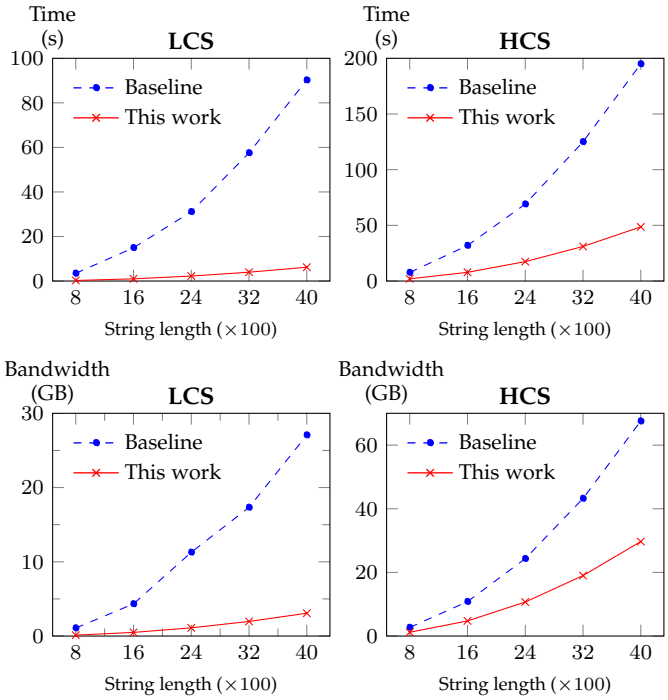


Fig. 4: LCS and HCS. ($\kappa = 127$)

6.3 Comparison with protocols using ABY

We also find our GC-based baseline better than ABY-based protocols. Analytically speaking, this is because, for the string metrics considered in this paper,

(1) a pure **Y** approach is essentially the same as our baseline;

(2) a pure **B** approach (i.e. GMW on binary circuits) has comparable cost to **Y**, but allows to move the expensive cryptography into an input-independent offline phase (at the cost of linear online rounds). Thus, for overall efficiency, it neither makes sense to combine **B** and **Y**;

(3) even if **A** allows free addition, it can't do secure comparison efficiently (other than first translate arithmetic encodings into binary encodings, then compare using either **B** or **Y**). In the best known circuits for computing these string metrics, every addition gate is immediately followed by a comparison gate. Because secure wire-label conversion is not cheaper than secure addition using **Y** or **B**, using **A** alone or mixing it with **B** or **Y** won't produce better protocols than our baseline.

Micro-benchmarks. We also measured the costs of addition, projection, and wire-label conversion. Due to page limit, we report our micro-benchmark experiments in Appendix D.

7 Conclusion

Customizing garbling schemes to specific computations can bring dramatical efficiency benefits. We have taken a first step to explore this methodology in constructing secure protocols for several representative string-comparison metrics. Our protocols are up to an order-of-magnitude more efficient than best existing results, but also generic, accurate, and provably secure under the standard, preferred definition of security. Our protocols can also be converted in standard ways to offer active security. Our findings would shed some light on designing other application-specific MPC protocols in the future.

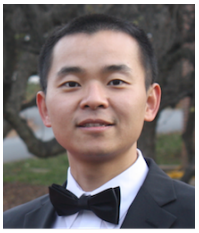
ACKNOWLEDGEMENT

Research reported in this paper was partly supported by NIH under award number U01EB023685.

REFERENCES

[1] S. Needleman and C. 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.
[2] Cancer Genome Atlas Network, "Comprehensive molecular portraits of human breast tumours," *Nature*, vol. 490, no. 7418, pp. 61–70, 2012.
[3] 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.
[4] W. E. Evans and M. V. Relling, "Moving towards individualized medicine with pharmacogenomics," *Nature*, vol. 429, no. 6990, pp. 464–468, 2004.
[5] S. Forrest, S. A. Hofmeyr, and A. Somayaji, "Computer immunology," *Communications of the ACM*, vol. 40, no. 10, pp. 88–96, 1997.
[6] C. Warrender, S. Forrest, and B. Pearlmutter, "Detecting intrusions using system calls: Alternative data models," in *IEEE Symposium on Security and Privacy*, 1999.
[7] D. Gao, M. K. Reiter, and D. Song, "Behavioral distance for intrusion detection," in *International Workshop on Recent Advances in Intrusion Detection*, 2006.
[8] G. Asharov, S. Halevi, Y. Lindell, T. Rabin, "Privacy-preserving search of similar patients in genomic data." in *Privacy Enhancing Technologies Symposium*, 2018.
[9] X. 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 Conference on Computer and Communications Security (CCS)*, 2015.
[10] S. Zahur, M. Rosulek, and D. Evans, "Two halves make a whole: reducing data transfer in garbled circuits using half gates," in *International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT)*, 2015.
[11] D. Demmler, T. Schneider, M. Zohner, "ABY: A framework for efficient mixed-protocol two-party computation," in *Network and Distributed System Security Symposium (NDSS)*, 2015.
[12] Y. Huang, D. Evans, J. Katz, and L. Malka, "Faster secure two-party computation using garbled circuits," in *USENIX Security Symposium*, 2011.
[13] A. Malozemoff and X. Wang, "EMP-Toolkit," https://github.com/emp-toolkit, 2016.
[14] M. Ball, T. Malkin, and M. Rosulek, "Garbling gadgets for boolean and arithmetic circuits," in *ACM Conference on Computer and Communications Security (CCS)*, 2016.
[15] X. Wang, S. Ranellucci, and J. Katz, "Global-scale secure multiparty computation," in *ACM Conference on Computer and Communications Security (CCS)*, 2017.
[16] S. Jha, L. Kruger, and V. Shmatikov, "Towards practical privacy for genomic computation," in *IEEE Symposium on Security and Privacy*, 2008.
[17] M. Bellare, V. T. Hoang, S. Keelveedhi, and P. Rogaway, "Efficient garbling from a fixed-key blockcipher," in *IEEE Symposium on Security and Privacy*, 2013.
[18] M. Bellare, V. T. Hoang, and P. Rogaway, "Foundations of garbled circuits," in *ACM Conference on Computer and Communications Security (CCS)*, 2012.
[19] V. Kolesnikov and T. Schneider, "Improved garbled circuit: Free XOR gates and applications," in *International Colloquium on Automata, Languages and Programming (ICALP)*, 2008.
[20] P. Mohassel and P. Rindal, "ABY3: a mixed protocol framework for machine learning," in *ACM Conference on Computer and Communications Security (CCS)*. 2018.
[21] G. Dessouky, F. Koushanfar, A.-R. Sadeghi, T. Schneider, S. Zeitouni, and M. Zohner, "Pushing the communication barrier in secure computation using lookup tables." in *Network and Distributed System Security Symposium (NDSS)*, 2017.
[22] A. C.-C. Yao, "How to generate and exchange secrets (extended abstract)," in *IEEE Symposium on Foundations of Computer Science (FOCS)*, 1986.
[23] B. Pinkas, T. Schneider, N. P. Smart, and S. C. Williams, "Secure two-party computation is practical," in *International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT)*, 2009.
[24] S. Gueron, Y. Lindell, A. Nof, and B. Pinkas, "Fast garbling of circuits under standard assumptions," in *ACM Conference on Computer and Communications Security (CCS)*, 2015.
[25] 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.
[26] 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.
[27] 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.
[28] F. Tajima and M. Nei, "Estimation of evolutionary distance between nucleotide sequences." *Molecular biology and evolution*, vol. 1, no. 3, pp. 269–285, 1984.
[29] A. Amir, Z. Gotthilf, and B. R. Shalom, "Weighted LCS," *Journal of Discrete Algorithms*, vol. 8, no. 3, pp. 273–281, 2010.
[30] R. Zhu, Y. Huang, J. Katz, and A. Shelat, "The cut-and-choose game and its application to cryptographic protocols," in *USENIX Security*, 2016.
[31] V. Kolesnikov and R. Kumaresan, "Improved OT extension for transferring short secrets," in *International Cryptology Conference (CRYPTO)*, 2013.
[32] Y. Ishai, J. Kilian, K. Nissim, and E. Petrank, "Extending oblivious transfers efficiently," in *International Cryptology Conference (CRYPTO)*, 2003.
[33] R. Zhu and Y. Huang, "Jimu: Faster lego-based secure computation using additive homomorphic hashes," in *International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT)*, 2017.
[34] R. Canetti, "Security and composition of multiparty cryptographic protocols," *Journal of Cryptology*, vol. 13, no. 1, pp. 143–202, 2000.
[35] M. Keller, E. Orsini, and P. Scholl, "Actively secure OT extension with optimal overhead," in *International Cryptology Conference (CRYPTO)*, 2015.
[36] X. Wang, S. Ranellucci, and J. Katz, "Authenticated garbling and efficient maliciously secure two-party computation," in *ACM Conference on Computer and Communications Security (CCS)*, 2017.

Ruiyu Zhu received his bachelor degree in mathematics from Peking University. Since 2015, he has been working toward the doctorate degree in Computer Science at Indiana University, Bloomington. His research mainly focuses on improving the practicality of secure multiparty computation techniques and customizing MPC protocols for real-world applications.



Yan Huang is broadly interested in security and cryptography. He has worked on developing cryptographic protocols that provide strong security guarantees for generic computations. His research aims at building elegant theoretical ideas into practically usable systems that address real-world security problems. His work marks a confluence of computer science theory, program analysis, artificial intelligence, and software engineering.

APPENDIX A
PROOF OF THEOREM 1

Theorem 1. *If π is an ideal block cipher that is used to realize* **Enc** *and* **Dec** *as described above. the scheme in Figure 2 satisfies the privacy and obliviousness definitions given in Section 2.1, and an application-dependent notion of authenticity.*

Proof. ***Privacy:*** Figure 5 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.

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

Fig. 5: The Simulator for Proving Privacy

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 $\mathsf{Sim}^{x_0}_{\mathsf{prv}}(1^k, f, f(x))$. Should $\mathsf{Sim}^{x_0}_{\mathsf{prv}}$ 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'. Note that $\mathsf{Sim}^{x}_{\mathsf{prv}}$ outputs exactly the distribution (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 $\hat{e}$ is a tuple of uniform strings and $\mathsf{Sim}^{x}_{\mathsf{prv}}$ and $\mathsf{Sim}^{x_0}_{\mathsf{prv}}$ used exactly the same procedure to produce all garbled gates.

(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); from the $w_t^{O_i}$ learned by the distinguisher, the distinguisher can only get $f(x)_i$ 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 $\mathsf{Sim}^{x}_{\mathsf{prv}}$ simply by dropping the input $f(x)$ and the third component d in the output. The proof of privacy can be carried over to prove obliviousness.

Application-dependent ***Authenticity:*** We note that due to the construction of **Enc**, if the adversary $\mathcal{A}$ can provide any Y' such that $Y' \neq \mathsf{Ev}(F, X))$ 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 $w_k = w_0 +_p k \times_p \Delta$, 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, let n be the size of the domain of the plaintext k, then 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. Since n can vary with application, we call this notion of authenticity application-dependent. □

Remark. In many practical applications such as string-comparison, it is easy to bound the value of n to small (application-specific) constants (e.g., < 300 in all applications considered in this paper) so that n/p is negligible.

APPENDIX B
PROOF OF PROPOSITION 1

Proposition 1. *Let* $\mathtt{s}, \mathtt{t}, D_{i,j}, c_{ins}, c_{del}, c_{sub}$ *be defined as in Section 2.2, where* c_{ins}, c_{del} *are generalized to one-dimensional tables and* c_{sub} *is generalized to a two-dimensional table. Let*

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

Then, there exist public constants C_1, C_2, C_3, C_4 which are independent of $D_{i,j}$, 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 $|D_{i,j-1} - D_{i-1,j-1}| \le c_{ins}[\mathtt{s}[i]]$, therefore

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

so,

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

hence,

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

So we can set

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

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

Symmetrically, we can derive that

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

(8) − (5) yields

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

(7) − (6) yields

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

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

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

and from (8) and (10) that

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

Finally, by defining

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

we proved $C_3 \le v_{i,j} \le C_4$. □

APPENDIX C
ACTIVELY-SECURE PROTOCOLS

We use three ideal functionalities $\mathcal{F}_{\mathsf{IHash}}$ (interactive hash), $\mathcal{F}_{\mathsf{COT}}$ (correlated OT), and $\mathcal{F}_{\mathsf{coin\text{-}toss}}$ (coin tossing) defined in Appendix C.4. First, the garbler is required to use a coin-tossed randomness to run Gb. This prevents an adversarial garbler from compromising the correctness any garbled table through selecting problematic randomness. To ensure P_1's input wire-labels to the evaluation circuits denote the same plaintext value, we used $\mathcal{F}_{\mathsf{IHash}}$, an XOR-homomorphic interactive hash implementation that was also used by JIMU [33] for similar purposes. Each initial input and final output wire in f's circuit is associated with a random permutation bit λ^I and the evaluator knows $\langle \lambda^I \rangle$ (i-hash of bit λ^I) and $\langle m^I_{\lambda^I} \rangle$ (i-hash of the wire-label denoting bit λ^I. Thanks to the XOR-homomorphism of $\mathcal{F}_{\mathsf{IHash}}$, it is easy for the evaluator to securely translate a wire-label m^I_b (a label on the master circuit denoting b) into $m^{I,i}_b$ (a label on the i-th GC denoting b) for any $b \in \{0,1\}$, given their i-hashes and their XOR-differences (see Step 5.).

We assume y has more than 40 bits. To ensure the evaluator use consistent y in all evaluation GCs, the parties run the correlated OT functionality $\mathcal{F}_{\mathsf{COT}}$ once for the evaluator to learn the wire-labels $\left\{m^I_{y^I}\right\}_{I \in \mathsf{Inp}(P_2)}$, which represent y on the master circuit. For evaluation, these master wire-labels are translated to wire-labels on each evaluation GC using XOR-differences whose validity is guaranteed by $\mathcal{F}_{\mathsf{IHash}}$.

Finally, $\mathcal{F}_{\mathsf{IHash}}$ also allows the evaluator to what output wire-labels are valid and identify inconsistent but valid output wire-labels. Note that inconsistent valid wire-labels reveals δ of the master circuit, and further reveals the garbler's input x when the garbler cheats (see Step 6.).

C.1 Full Protocol Description

Let s be the statistical security parameter. Assume P_1 (the circuit generator holding input string x) and P_2 (the circuit evaluator holding input string y that has more than s bits) want to compute a string-comparison metric f between x, y.

1. **Setup.** On cut-and-choose parameter n and computational security parameter κ, P_1 and P_2 call $\mathcal{F}_{\mathsf{coin\text{-}toss}}$ for P_1 to learn $\{seed_i \in \{0,1\}^\kappa\}_{i \in [n]}$. For $i \in [n]$, P_1 sets

$$(\delta_i, \Delta_i) := \left(\mathbf{PRG}(seed_i, \text{“delta”}), \mathbf{PRG}(seed_i, \text{“Delta”})\right)$$

and sends $\{\langle \delta_i \rangle\}_{i \in [n]}$ to P_2 through $\mathcal{F}_{\mathsf{IHash}}$.

The master circuit. P_1 samples $\delta \in \{0,1\}^\kappa$. For every input-wire I of P_1's input to f, P_1 samples uniform random bit $\lambda^I := \mathbf{PRG}(\delta, I \| \text{“lambda”})$; for every input-wire I of P_2's input to f and every output-wire I of f, P_1 sets $\lambda^I := 0$. For every input-wire or output-wire I of f, P_1 samples $m^I_0 \in \{0,1\}^\kappa$, sets $m^I_1 := m^I_0 \oplus \delta$, and sends $\langle \lambda^I \rangle, \langle m^I_{\lambda^I} \rangle, \langle \delta \rangle$ to P_2 through $\mathcal{F}_{\mathsf{IHash}}$.

OT of seeds. P_1 picks a uniform $\Delta \in \{0,1\}^\kappa$. P_1 and P_2 call $\mathcal{F}_{\mathsf{coin\text{-}toss}}$ for P_2 to learn an n-bit string $\mathcal{J}$ sampled from certain public distribution (see [30] for the details on how this public distribution of $\mathcal{J}$ is calculated). Then P_1 with input $(\Delta, \{seed_i\}_{i \in [n]})$ and P_2 with input $\mathcal{J}$ call $\mathcal{F}_{\mathsf{COT}}$ for P_2 to learn $\{seed_i \,|\, \mathcal{J}_i = 1\}$ and $\{seed_i \oplus \Delta \,|\, \mathcal{J}_i = 0\}$.

2. **Inputs.** For every input-wire I in the i-th GC, P_1 sets

$$m^{I,i}_0 := \mathbf{PRG}(\delta_i, (I,i) \| \text{“label”}),$$
$$m^{I,i}_1 := m^{I,i}_0 \oplus \delta_i.$$

Then,

a) P_1**'s Input.** For every input-wire I of P_1's input in the i-th garbled circuit, P_1 sets

$$\lambda^{I,i} := \mathbf{PRG}(\delta_i, (I,i) \| \text{“lambda”}).$$

b) P_2**'s Input.** For every input-wire I of P_2's input in the i-th garbled circuit, P_1 sets

$$\lambda^{I,i} := 0.$$

(C-OT) P_1 with $\left(\delta, \{m_0^I\}_{I \in \mathsf{Inp}(P_2)}\right)$ and P_2 with $\{y^I\}_{I \in \mathsf{Inp}(P_2)}$, invoke $\mathcal{F}_{\mathsf{COT}}$ so P_2 learns $\{m_{y^I}^I\}_{I \in \mathsf{Inp}(P_2)}$. P_2 verifies that $m_{y^I}^I$ matches with $\langle m_0^I \rangle \oplus y^I \langle \delta \rangle$ for all I, and aborts otherwise.

Now, for every input-wire I in the i-th GC, P_1 sends $\langle \lambda^{I,i} \rangle, \langle m_{\lambda^{I,i}}^{I,i} \rangle$ to P_2 through $\mathcal{F}_{\mathsf{IHash}}$.

3. **Garble.** P_1 generates n garbled circuits $\{GC_i\}_{i \in [n]}$ for f as follows:

a) For every input-wire I of the i-th garbled circuit, P_1 generates arithmetic wire-labels

$$w_0^{I,i} := \mathbf{PRG}(\delta_i, I, i)$$
$$w_1^{I,i} := w_0^{I,i} +_p \Delta_i,$$

then sends an ordered pair

$$\left[\mathbf{Enc}_{m_{\lambda^{I,i}}^{I,i}}\left(w_{\lambda^{I,i}}^{I,i}\right), \mathbf{Enc}_{m_{1 \oplus \lambda^{I,i}}^{I,i}}\left(w_{1 \oplus \lambda^{I,i}}^{I,i}\right)\right].$$

which will allow securely translating binary field encodings into their $\mathbb{Z}_p$ encodings.

b) For addition, subtraction, constant multiplication and bounded-value projection gates, P_1 runs the Gb algorithm of the garbling scheme of Figure 2.

c) For every output-wire I of the i-th garbled circuit, P_1 sends a secure projection table allows to translate arithmetic encoding $w_v^{I,i}$ (v takes a bounded number of values) to its binary encodings $m_{b_0}^{I,i,0}, \ldots, m_{b_k}^{I,i,k}$ where $v = b_0 b_1 \cdots b_k$ and $m_0^{I,i,j} := \mathbf{PRG}(\delta_i, I, i, j)$, $m_1^{I,i,j} := m_0^{I,i,j} \oplus \delta_i$ for all $j \in [k]$. P_1 sends $\left\{\langle m_0^{I,i,j} \rangle\right\}_{I \in \mathsf{Output}(f), i \in [n], j \in [k]}$ to P_2 via $\mathcal{F}_{\mathsf{IHash}}$.

P_1 sends $H(GC_i)$ to P_2 (H is a collision-resistant hash).

4. **Check.** For each check-circuit GC_i, namely those $i \in [n], \mathcal{J}_i = 1$, P_2 use $seed_i$ to verify that P_1 have played honestly in all previous steps; and aborts otherwise. In particular, P_2 checks the following constraints:

a) GC_i generated from $seed_i$ matches its hash $H(GC_i)$.

b) δ_i generated from $seed_i$ matches $\langle \delta_i \rangle$ via $\mathcal{F}_{\mathsf{IHash}}$.

c) $m_0^{I,i}$ generated from δ_i matches $\langle m_0^{I,i} \rangle$ via $\mathcal{F}_{\mathsf{IHash}}$.

d) $\lambda^{I,i}$ generated from δ_i matches $\langle \lambda^{I,i} \rangle$ via $\mathcal{F}_{\mathsf{IHash}}$.

e) For all $I \in \mathsf{Output(f)}, j \in [k]$, wire-label $m_0^{I,i,j}$ of GC_i matches $\langle m_0^{I,i,j} \rangle$ via $\mathcal{F}_{\mathsf{IHash}}$.

5. **Evaluate.** For each evaluation-circuit GC_i, namely those $i \in [n], \mathcal{J}_i = 0$, P_2 sends $seed_i \oplus \Delta$ to P_1 who verifies the consistency of the value. Then, P_1 and P_2 collaborate to evaluate these circuits. For every evaluation-circuit GC_i, P_1 sends $\delta \oplus \delta_i$ to P_2, who verifies it with $\langle \delta \rangle \oplus \langle \delta_i \rangle$.

a) P_1**'s Input.** For every input-wire I of P_1's input x^I, P_1 sends $m_{x^I}^I, \lambda^I \oplus \lambda^{I,i}, m_{\lambda^I}^I \oplus m_{\lambda^{I,i}}^{I,i} \oplus (\lambda^I \oplus \lambda^{I,i})\delta_i$ to P_2, who verifies their validity against their i-hashes through $\mathcal{F}_{\mathsf{IHash}}$. P_2 computes $m_{x^I}^{I,i} := m_{x^I}^I \oplus (m_{\lambda^I}^I \oplus m_{\lambda^{I,i}}^{I,i} \oplus (\lambda^I \oplus \lambda^{I,i})\delta_i) \oplus (\lambda^I \oplus x^I)(\delta \oplus \delta_i)$.

b) P_2**'s Input.** For every input-wire I of P_2's input y^I, P_1 sends $m_0^I \oplus m_0^{I,i}$ to P_2, who verifies their validity against their i-hashes through $\mathcal{F}_{\mathsf{IHash}}$. P_2 computes $m_{y^I}^{I,i} := m_{y^I}^I \oplus (m_0^I \oplus m_0^{I,i}) \oplus y^I(\delta \oplus \delta_i)$.

c) **Eval.** With the wire-labels obtained above, P_2 evaluates the garbled circuit according to the garbling scheme's Ev method.

d) **Check.** P_2 verifies that all GC_i ($i \in [n]$) received match their hashes received in Step 3..

6. **Output.** For every output-wire I in the i-th evaluation circuit, P_1 sends $m_0^I \oplus m_0^{I,i}$ to P_2, who verifies its validity through $\mathcal{F}_{\mathsf{IHash}}$. P_2 validates every output wire-label obtained from circuit evaluation against their i-hashes, then translates them to plaintext values.

a) If P_2 all valid wire-labels obtained from evaluating the $n - k$ circuits refer to the same plaintext value, then P_2 outputs this value and halts.

b) If P_2 obtains two valid output wire-labels on the same output wire I which decode to different plaintext values, then P_2 can obtain valid m_0^I and m_1^I simultaneously by:

$$\begin{aligned} m_0^I &:= m_0^{I,i_1} \oplus (m_0^I \oplus m_0^{I,i_1}) \\ m_1^I &:= m_1^{I,i_2} \oplus (m_0^I \oplus m_0^{I,i_2}) \oplus (\delta \oplus \delta_{i_2}) \end{aligned}$$

for some i_1, i_2. Then P_2 can learn $\delta := m_0^I \oplus m_1^I$, whose value can be validated through $\mathcal{F}_{\mathsf{IHash}}$. With δ, P_2 can learn $\{\lambda_I\}_{I \in \mathsf{Inp}(P_1)}$, and further recovers P_1's input x from $\{\lambda_I\}_{I \in \mathsf{Inp}(P_1)}$, $\{\langle m_{\lambda^I}^I \rangle\}_{I \in \mathsf{Inp}(P_1)}$ and those $\{m_{x^I}^I\}_{I \in \mathsf{Inp}(P_1)}$ it received in Step 5.. P_2 locally computes $f(x, y)$ and outputs it.

C.2 Proof of Theorem 2

Theorem 2. *The improved garbling scheme of Section 4.3 satisfies the privacy, obliviousness, and authenticity properties outlined in Section 2.1.*

Proof. Note that the garbling mechanism for addition and projection is the same as that of our basic garbling scheme described in Section 3. The proofs for privacy and obliviousness properties are essentially the same as that of the main theorem (Theorem 1). Thus, below we focus on the proof of authenticity. Assume for the purpose of contradiction that the adversary $\mathcal{A}$ can provide some Y' such that $Y' \neq \mathsf{Ev}(F, X))$ but $\mathsf{De}(d, Y') = k \neq \perp$ (where $(F, e, d) \leftarrow \mathsf{Gb}(1^k, f), X := \mathsf{En}(e, x)$). Then $\mathcal{A}$ must know $\mathsf{L}_a = \left(k_0 +_p a \times_p \Delta, \hat{k}_0 +_p a \times_p \hat{\Delta}\right)$ for some a. However, without knowing any of $k_0, \hat{k}_0, \Delta, \hat{\Delta}$, for any particular a, the probability that $\mathcal{A}$ succeeds in guessing a $\mathsf{L}_a = \left(L, \hat{L}\right)$ such that

$$(L - k_0) \times_p \Delta^{-1} = \left(\hat{L} - \hat{k}_0\right) \times_p \hat{\Delta}^{-1}$$

is $1/p$ at the best, (because both sides of the equation above are uniformly distributed over $\mathbb{Z}_p$). Therefore, with e.g. $p > 2^{87}$, the authenticity error will be less than 2^{-87}. □

C.3 Proof of Theorem 3

Theorem 3. *The protocol of Appendix C.1 securely computes f in presence of malicious adversaries.*

Proof. We prove the security in a hybrid-model where the parties have access to ideal functionalities for $\mathcal{F}_{\mathsf{lHash}}$, $\mathcal{F}_{\mathsf{COT}}$, and $\mathcal{F}_{\mathsf{coin\text{-}toss}}$. The standard composition theorem [34] implies security when the sub-routines are instantiated with secure implementations of these functionalities.

If P_1 is corrupted. We construct an efficient simulator $\mathcal{S}$ interacting with the ideal string-metrics functionality as P_1. $\mathcal{S}$ runs the corrupted real-model P_1 as a subroutine, interacting with it like real-model P_2 with input $y = 0$ using the protocol of Appendix C.1, except for the following changes:

1) In Step 1., through the simulated $\mathcal{F}_{\mathsf{lHash}}$, $\mathcal{S}$ learns $\{\lambda^I\}_{I\in\mathsf{Input}(P_1)}$.
2) In Step 5.a, $\mathcal{S}$ learns $\{m^I_{x^I}\}_{I\in\mathsf{Input}(P_1)}$. For all $I \in \mathsf{Input}(P_1)$, if $m^I_{x^I}$ matches $\langle m^I_{\lambda^I}\rangle$, then $\mathcal{S}$ sets $x^I := \lambda^I$; if $m^I_{x^I}$ matches $\langle m^I_{\lambda^I}\rangle \oplus \langle\delta\rangle$, then $\mathcal{S}$ sets $x^I := \overline{\lambda^I}$.
3) In Step 6., $\mathcal{S}$ submits x to the trusted functionality and outputs whatever P_1 outputs.

To show that the joint out distribution in this ideal-model involving $\mathcal{S}$ is indistinguishable from that of the real-model involving the corrupted P_1, we consider a series of experiments each with a slightly modified simulator.

1) $\mathbf{Hybrid}_1$ The simulator $\mathcal{S}_1$ interacts with the corrupted P_1 running the real-model protocol, where $\mathcal{S}_1$ uses P_2's actual input y as its input. Their interaction is identical to the real-model execution.
2) $\mathbf{Hybrid}_2$ Simulator $\mathcal{S}_2$ acts the same way as $\mathcal{S}_1$ in $\mathbf{Hybrid}_1$, except:
   a) In Step 1., through the simulated $\mathcal{F}_{\mathsf{lHash}}$, $\mathcal{S}_2$ learns $\{\lambda^I\}_{I\in\mathsf{Input}(P_1)}$.
   b) In Step 5.a, $\mathcal{S}_2$ learns $\{m^I_{x^I}\}_{I\in\mathsf{Input}(P_1)}$. For $I \in \mathsf{Input}(P_1)$, if $m^I_{x^I}$ matches $\langle m^I_{\lambda^I}\rangle$, then $\mathcal{S}_2$ sets $x^I := \lambda^I$; if $m^I_{x^I}$ matches $\langle m^I_{\lambda^I}\rangle\oplus\langle\delta\rangle$, then $\mathcal{S}_2$ sets $x^I := \overline{\lambda^I}$.
   c) In Step 6., $\mathcal{S}_2$ outputs $f(x, y)$.

   We claim $\mathbf{Hybrid}_2 \approx \mathbf{Hybrid}_1$ because
   - To the corrupted P_1, the only messages it got from the simulators are $\{seed_i \oplus \Delta \mid i \in [n], \mathcal{J}_i = 0\}$ in Step 4.. However, these messages in $\mathbf{Hybrid}_1$ and $\mathbf{Hybrid}_2$ are identically distributed, a fact guaranteed by $\mathcal{F}_{\mathsf{COT}}$.
   - In both experiments, the simulators correctly output $f(x, y)$ if at least one correct GC is evaluated.
3) $\mathbf{Hybrid}_3$ Simulator $\mathcal{S}_3$ acts the same way as $\mathcal{S}_2$ in $\mathbf{Hybrid}_2$, except:
   a) $\mathcal{S}_3$ runs the corrupted P_1 as a sub-routine and take an ideal P_1's role to interact with the ideal string-metrics functionality.
   b) In Step 6., $\mathcal{S}_3$ submits x to the ideal string-metrics functionality and outputs whatever the corrupted P_1 outputs.

   We claim $\mathbf{Hybrid}_3 \approx \mathbf{Hybrid}_2$ because
   - $\mathcal{S}_3$'s output is the same as the corrupted P_1's in $\mathbf{Hybrid}_2$.
   - The ideal P_2 in $\mathbf{Hybrid}_3$ and $\mathcal{S}_2$ in $\mathbf{Hybrid}_2$ both output $f(x, y)$.
4) $\mathbf{Hybrid}_4$ Simulator $\mathcal{S}_4$ acts the same way as $\mathcal{S}_3$, except s_4 uses $y = 0$ instead of P_2's actual input as its input when interacting with the corrupted P_1. $\mathcal{S}_4$ is identical to $\mathcal{S}$. This is the ideal-model execution.

We claim $\mathbf{Hybrid}_4 \approx \mathbf{Hybrid}_3$ because the real-model P_2's outgoing-message distributions (including whether and when P_2 aborts) do not depend on the value of its input y.

If P_2 is corrupted. We construct an efficient simulator $\mathcal{S}$ interacting with the ideal string-metrics functionality as an ideal-model P_2. $\mathcal{S}$ will run the corrupted P_2 as a sub-routine, interacting with it as real-model P_1 with input $x = 0$ using the protocol of Appendix C.1, except for the following changes:

1) In Step 1., $\mathcal{S}$ learns $\mathcal{J}$ through the simulated $\mathcal{F}_{\mathsf{COT}}$.
2) In Step 2., $\mathcal{S}$ learns y through the simulated $\mathcal{F}_{\mathsf{COT}}$. $\mathcal{S}$ sends y to the ideal functionality and gets back $z = f(x, y)$.
3) In Step 3., for all $i \in [n], \mathcal{J}_i = 1$, $\mathcal{S}$ generates GC_i honestly. For all $i \in [n], \mathcal{J}_i = 0$, $\mathcal{S}$ runs the simulator $\mathcal{S}_{\mathsf{prv}}(f, z)$ to produce GC_i (see the *privacy* definition of garbling for $\mathcal{S}_{\mathsf{prv}}$).
4) In Step 6., $\mathcal{S}$ outputs whatever the malicious P_2 outputs.

To show that the joint out distribution in this ideal-model involving $\mathcal{S}$ is indistinguishable from that of the real-model involving the corrupted P_2, we consider a series of experiments each with a slightly modified simulator.

1) $\mathbf{Hybrid}_1$ The simulator $\mathcal{S}_1$ interacts with the corrupted P_2 using the real-model protocol with P_1's actual input x. This is the real-model execution.
2) $\mathbf{Hybrid}_2$ The simulator $\mathcal{S}_2$ is the same as $\mathcal{S}_1$ in $\mathbf{Hybrid}_1$, except:
   a) In Step 1., $\mathcal{S}$ learns $\mathcal{J}$ through the simulated $\mathcal{F}_{\mathsf{COT}}$.
   b) In Step 2., $\mathcal{S}_2$ learns y through the simulated $\mathcal{F}_{\mathsf{COT}}$.
   c) In Step 3., for all $i \in [n], \mathcal{J}_i = 1$, $\mathcal{S}$ generates GC_i honestly. For all $i \in [n], \mathcal{J}_i = 0$, $\mathcal{S}$ runs the simulator $\mathcal{S}_{\mathsf{prv}}(f, z)$ to produce GC_i (see the *privacy* definition of garbling for $\mathcal{S}_{\mathsf{prv}}$).

   We claim $\mathbf{Hybrid}_2 \approx \mathbf{Hybrid}_1$ because our garbling scheme is proven to be *private*, hence the corrupted P_2 cannot tell if a GC is honestly garbled or simulated with a chosen output z.
3) $\mathbf{Hybrid}_3$ Simulator $\mathcal{S}_3$ is the same as $\mathcal{S}_2$ in $\mathbf{Hybrid}_2$, except:
   a) $\mathcal{S}_3$ runs the corrupted P_2 as a subroutine and interacts with the ideal string-metrics functionality as an ideal-model P_2.
   b) In Step 3., instead of computing $f(x, y)$, $\mathcal{S}_3$ submits y to the ideal functionality and receives $f(x, y)$.
   c) In Step 6., $\mathcal{S}_3$ outputs whatever the corrupted P_2 outputs.

   We claim $\mathbf{Hybrid}_3 \approx \mathbf{Hybrid}_2$ because
   - $\mathcal{S}_3$'s output is the same as the corrupted P_2's in $\mathbf{Hybrid}_2$.
   - The ideal-model P_1 in $\mathbf{Hybrid}_3$ and $\mathcal{S}_2$ in $\mathbf{Hybrid}_2$ both have no output.
4) $\mathbf{Hybrid}_4$ the simulator $\mathcal{S}_4$ is the same as $\mathcal{S}_3$ in $\mathbf{Hybrid}_3$, except that it uses $x = 0$ as its input to interact with the

corrupted P_1. $\mathcal{S}_4$ is identical to $\mathcal{S}$ and this is the ideal-model execution.

We claim $\mathbf{Hybrid}_3 \approx \mathbf{Hybrid}_2$ because the real-model P_1's outgoing-message distributions (including whether and when P_1 aborts) do not depend on the value of x. □

C.4 Definition of $\mathcal{F}_{\mathsf{lHash}}$, $\mathcal{F}_{\mathsf{COT}}$, and $\mathcal{F}_{\mathsf{coin\text{-}toss}}$

The $\mathcal{F}_{\mathsf{lHash}}$ Functionality. We adopt the definition of $\mathcal{F}_{\mathsf{lHash}}$ from that of JIMU [33]. Note that $\mathcal{F}_{\mathsf{lHash}}$ allows to verify the validity of the XOR-difference among several previously hashed messages. $\mathcal{F}_{\mathsf{lHash}}$ also enables the receive to verify any single message by calling Verify on a single message (i.e., $t = 1$).

- Hash. Upon receiving $(\texttt{Hash}, m_1, \ldots, m_\nu)$ ($\nu \geq 1$) from P_1 (where with respect to P_2, each m_i has κ-bit entropy): for every i, pick a fresh number id_i, record (id_i, m_i) and generate a delayed output $(\texttt{Receipt}, id_i)$ to P_2.
- Verify. Upon receiving $(\texttt{Verify}, id_1, \ldots, id_t, d)$ ($t \geq 1$) from P_2: if there are recorded values $(id_1, m_1), \ldots, (id_t, m_t)$ (otherwise do nothing), set $z = 1$ if $m_1 \oplus \cdots \oplus m_t = d$, and $z = 0$, otherwise; generate a delayed output $(\texttt{VerifyResult}, z)$ to P_2.

Fig. 6: The ideal functionality $\mathcal{F}_{\mathsf{lHash}}$.

The $\mathcal{F}_{\mathsf{COT}}$ Functionality. $\mathcal{F}_{\mathsf{COT}}$ is the correlated OT functionality as defined in Figure 7. It can be efficiently realized with small modification to the actively-secure OT-extension protocol of Keller et al. [35]. The idea of $\mathcal{F}_{\mathsf{COT}}$ was also used in authenticated garbling [15], [36] to construct authenticated multiplicative triples.

The $\mathcal{F}_{\mathsf{coin\text{-}toss}}$ Functionality. On receiving "init" from both parties, $\mathcal{F}_{\mathsf{coin\text{-}toss}}$ samples a uniform bit-string s and send it to the designated party (while allowing premature aborts).

Appendix D
Micro-benchmark Experiments

We measured the performance of several basic operations under our garbling scheme. All experiments in this subsection are conducted with respect to 87-bit computational security.

Secure Addition. Table 2 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.

Correlated-OT: Upon receiving $\left(\Delta, \{m_i\}_{i\in[l]}\right)$ where $\Delta, m_i \in \{0,1\}^\kappa \in \{0,1\}^\kappa$ from S and $\mathbf{x}$ where $\mathbf{x} \in \{0,1\}^l$ from R, send $\{m_i \oplus \mathbf{x}_i\Delta\}_{i\in[l]}$ to R.

Fig. 7: The Correlated OT functionality $\mathcal{F}_{\mathsf{COT}}$.

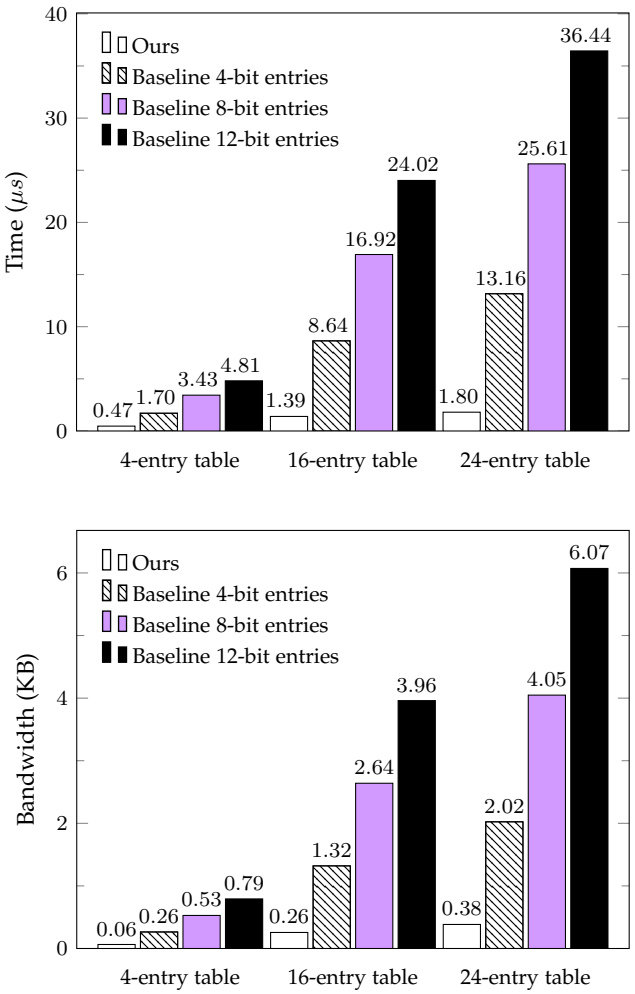


Fig. 8: Costs of secure table-lookup. (Timings are measured by averaging over 10^6 runs.)

Secure Table-Lookup. This is also the essential enabling primitive for secure comparison and bounded range minimum computations. Figure 8 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 binary circuit garbling scheme into arithmetic wire-labels in our scheme is highly efficient, at about 420ns (and ∼32 bytes bandwidth) per bit of Boolean wire-label, since it involves only two garbled rows per Boolean wire (Table 3).

Converting arithmetic wire-labels into Boolean ones

TABLE 2: 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 [10]	1420	2770	5520	11100	154	330	682	1386
This Work	**2.8**				**0**			

We note the timings coincide well with the cost of AESNI-based garbling ($\sim$ 45 ns/row) and that of modulo arithmetic with respect to an 88-bit prime ($\sim$2.8 $ns/+_p$). Timings are averaged over 10^6 runs for Half-Gates and 10^9 runs for ours.

TABLE 3: 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 runs while those in the third row are over 10^3 runs.

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}.