Efficient Two-Online-Round Secure Comparison in the Plain Model

Koji Nuida[1,2] Satsuya Ohata[1,3]
[1] Kyushu University, Fukuoka, 819-0395, Japan
[2] AIST, Tokyo, 135-0064, Japan
[3] Byerlis Inc., Tokyo, 170-0013, Japan

August 20, 2026

Abstract

Secure comparison is a fundamental building block frequently employed in various applications of secure multiparty computation, such as secure machine learning. Such protocols based on secret sharing (SS) typically excel in throughput, but they historically suffer from higher (online) round complexity. To address this issue, Attrapadung et al. (AsiaCCS 2022) proposed a protocol with two online rounds in the client-aided model (i.e., assuming a trusted third party in the offline phase), and Lu et al. (USENIX Security 2025) proposed a protocol with three online rounds in the plain model. In this paper, we propose a plaintext comparison protocol with two online rounds in the plain model. While keeping the lowest online round complexity, our protocol also reduces the total bit complexity. For example, for the case of 64-bit integers, the total bit complexity of our protocol is 25.7% of Lu et al.'s protocol and 23.3% of Attrapadung et al.'s protocol adjusted to the plain model.

keywords: Secure Multiparty Computation, Secure Comparison

1 Introduction

Secure multiparty computation (MPC) is a cryptographic framework that allows multiple mutually distrustful parties to jointly compute a function without revealing their private inputs. While MPC spans a broad spectrum of research directions, we focus on a fundamental primitive: secure comparison protocols, traditionally known as the Millionaires' Problem [20]. Secure comparison protocols are frequently used in practical applications, and their high overhead is well-recognized. For example, a study on secure neural networks, CryptGPU [18], reported that operations involving comparison (specifically ReLU and Softmax) accounted for 42-73% of the total execution time (Table 5 in [18]). It has been an important topic to improve the efficiency of secure comparison protocols.

Among various approaches to MPC, protocols based on secret sharing (SS) are typically recognized for their superior efficiency in terms of both computation and communication volume (i.e., the total number of bits transferred); however, they suffer from higher round complexity than approaches based on other cryptographic primitives such as garbled circuits. In particular, it was a difficult challenge to realize SS-based constant-round comparison protocols. In the client-aided model [11] assuming a trusted third party in the offline phase, a constant-round protocol was realized earlier [12], and even a two-round protocol was proposed by Attrapadung et al. in 2022 [1]. On the other hand, in the plain model for pure two parties, in 2025, Lu et al. [10] proposed for the first time an SS-based plaintext comparison protocol with constant online round complexity; precisely, three online rounds (as demonstrated in [10], such comparison over plaintexts can be extended to secret-shared inputs by sacrificing a single bit of the plaintext space). Now we mention that Attrapadung et al.'s two-round protocol can be converted into the plain model by using protocols for the offline phase described in [1, Appendix D]. However, although Lu et al.'s protocol and Attrapadung et al.'s protocol have efficient online phases (e.g., with less than 140-byte communication for 64-bit integers),

Table 1: Comparison with the previous constant-round two-party plaintext comparison protocols for n-bit integers (where protocols in the client-aided model are excluded); λ denotes the security parameter; μ denotes the bit length of a group element satisfying a suitable computational assumption (e.g., CDH assumption); (†) under correlated keys generation scheme which performs $O(2^n)$ times Hash locally; and most of the values (and remarks) for previous protocols are quoted from [10, Table 1].

Approach	Protocol	Offline	Online	
		Communication	Communication	# of rounds
GC-based	Yao [19, 21]	$2n\lambda$	$2n\lambda$	2
HE-based	GSV [7]	–	$18\mu n + 8\mu$	9
FSS-based	Half-Tree [9]	$(n+2)\lambda$ (†)	$2n$	1
SS-based	Lu et al. [10]	$\approx 2n\lambda \log_2(2n) + n\log_2^2 n$	$(2n+3)(\log_2 n + 2)$	3
	Ours (§ 4)	$\approx 3n^2 + 2\mu + \lambda$ $+ 2(n+\mu+\lambda)\log_2 n$	$2n^2$	2

Table 2: Efficiency comparison of our protocol with the state-of-the-art SS-based constant-round protocols; the parameters (see Table 1 for their meanings) are set to $n = 64$, $\lambda = 128$, and $\mu = 512$.

Protocol	# of rounds	Communication		
		Online	Offline	Total
Lu et al. [10]	3	131 B	14.3 KiB	14.4 KiB
Attrapadung et al. [1]	**2**	**54.75 B**	15.8 KiB	15.9 KiB
Ours (§ 4)	**2**	1024 B	**2.7 KiB**	**3.7 KiB**

their offline bit complexities are fairly large (e.g., over 14 kibibytes for 64-bit integers). Considering practical protocols where a large number of comparison protocols are internally executed, it is important to improve the total bit complexity of such protocols while keeping the constant round complexity.

1.1 Our Contribution

In this paper, we propose a new SS-based plaintext comparison protocol in the plain model with two online rounds and improved total bit complexity. For n-bit integers, the total bit complexity of our proposed protocol is $O(n^2)$, while it is $O(\lambda n \log n)$ for Lu et al.'s protocol [10] (the total bit complexity of Attrapadung et al.'s protocol [1] for a general n was not described in their paper); see Table 1 for details. Hence, in a practical range of parameters $n \lessapprox \lambda$, the bit complexity of our protocol is lower than Lu et al.'s protocol. Moreover, for a concrete parameter $n = 64$, Table 2 shows a comparison of efficiency of our protocol with Lu et al.'s and Attrapadung et al.'s protocols. Here, as the offline bit complexity of Attrapadung et al.'s protocol was not described in their paper, we evaluated the offline bit complexity of their protocol; see Section 5 for details. Table 2 shows that, while keeping the lowest two-round complexity, the total bit complexity of our protocol is significantly improved, to 25.7% of Lu et al.'s protocol and 23.3% of Attrapadung et al.'s protocol.

1.2 Technical Overview

For SS-based comparison protocols in the plain model, traditional strategies, such as the divide-and-conquer approach or those utilizing parallel prefix adders to determine the sign of the difference between two integers, typically require $O(\log n)$ rounds. Lu et al. [10] introduced a new design paradigm based on their proposed oblivious selective zero check (OZC) protocol and achieved a constant three-round complexity. However, as

the OZC protocol is not flexible, it seems difficult to reduce the round complexity further by following Lu et al.'s approach.

In contrast, here we first give a generic construction of a three-round comparison protocol for n-bit inputs $x = (x_{n-1} \cdots x_1 x_0)_2$ and $y = (y_{n-1} \cdots y_1 y_0)_2$ using any one-round equality testing protocol for $O(\log n)$-bit inputs. Our starting point is the following natural idea: we have $x < y$ if and only if there is some (in fact, a unique) index i with the property that $(x_{n-1} \cdots x_{i+1})_2 = (y_{n-1} \cdots y_{i+1})_2$ and $x_i < y_i$. Now the former equality testing is reduced (in Round 1) to the equality testing of two $O(\log n)$-bit integers via the dimension reduction technique of Lu et al. [10, Section 3.2]. The latter equality testing is solved via an auxiliary one-round equality testing protocol in Round 2. Then in Round 3, we combine the result of this equality testing with the result of $x_i < y_i$, computed in a previous round in parallel, to obtain the final result. Specifically, our generic construction proceeds as follows, where $\mathbf{1}\{\mathsf{Cond}\} := 1$ if Cond is true and $\mathbf{1}\{\mathsf{Cond}\} := 0$ if Cond is false:

(Round 1) For each, say the i-th input bit, compute an arithmetic share of $\mathsf{neq}_{[n-1,i+1]} := \sum_{j=i+1}^{n-1} \mathbf{1}\{x_j \neq y_j\}$ and a Boolean share of $\mathsf{lt}_i := \mathbf{1}\{x_i < y_i\}$.

(Round 2) For each i, compute a Boolean share of

$$Z_{i+1} := \mathbf{1}\{(x_{n-1}, \ldots, x_{i+1}) = (y_{n-1}, \ldots, y_{i+1})\} = \mathbf{1}\{\mathsf{neq}_{[n-1,i+1]} = 0\}$$

by using any one-round equality testing protocol.

(Round 3) Compute a Boolean share of $\mathbf{1}\{x < y\} = \mathsf{lt}_{n-1} \oplus \bigoplus_{i=0}^{n-2} (Z_{i+1} \wedge \mathsf{lt}_i)$.

Although the round complexity is not yet reduced in comparison to Lu et al.'s protocol, our construction has an advantage that it has flexibility in the choice of the internal equality testing protocol.

Now if the AND operation in Round 3 could be "absorbed" into the equality testing in Round 2, then we would obtain a two-round protocol. However, the existing one-round equality testing protocol of Lu et al. [10, Section 3] does not enable the round absorption. Instead, we instantiate the internal one-round equality testing protocol as in Section 3, where a share of each Z_{i+1} is computed as XOR of ANDs, say $Z_{i+1} = \bigoplus_j X_{i+1,j} \wedge Y_{i+1,j}$ (see Section 3 for a precise definition of these terms). This enables the round absorption by computing

$$Z_{i+1} \wedge \mathsf{lt}_i = \bigoplus_j X_{i+1,j} \wedge Y_{i+1,j} \wedge \mathsf{lt}_i \ ,$$

but it requires offline-inefficient 3-fan-in ANDs. To avoid the use of multi-fan-in ANDs, we observe that the bits $X_{i+1,j}$ and $Y_{i+1,j}$ are known as plaintexts by party P_0 and party P_1, respectively. Moreover, we have $\mathsf{lt}_i = (\neg x_i) \wedge y_i$, and the bits $\neg x_i$ and y_i are also known as plaintexts by party P_0 and party P_1, respectively. Then, by arranging the computation as

$$X_{i+1,j} \wedge Y_{i+1,j} \wedge \mathsf{lt}_i = \big(X_{i+1,j} \wedge (\neg x_i)\big) \wedge \big(Y_{i+1,j} \wedge y_i\big) \ ,$$

each of the values in two large parentheses in the right-hand side can be locally computed by a single party, therefore the overall computation can be done by plain ANDs (instead of 3-fan-in ANDs) for "trivially" shared bits. Thus we obtain a two-round protocol without offline-inefficient multi-fan-in ANDs.

1.3 Related Work

Generally, garbled circuits (GC) [19, 21] provide the baseline for constant (two)-round comparison but incur high bandwidth costs $O(\lambda n)$ especially in the online phase. Constant-round comparison protocols based on (fully) homomorphic encryption (HE) are also proposed [7, 15, 5], but they either have higher round complexity or suffer from large computational and communication overhead. While various SS-based approaches have attempted to reduce rounds, they often rely on relaxed models (e.g., the five-round protocol in [12] based on the client-aided model [11]) or expensive pre-computation (e.g., the use of multi-fan-in gates [16, 17, 6] or function secret sharing (FSS) [9, 3]). Recently, Lu et al. [10] established a three-round SS-based comparison protocol, but its round complexity is still higher than GC-based two rounds. Our protocol directly addresses this gap.

1.4 Paper Organization

Section 2 describes the basic notation and the existing techniques used in this paper. Section 3 introduces the equality testing protocol that serves as the foundation for our comparison protocol. In Section 4, we present our comparison protocol along with its bit complexity analysis. In Section 5, we evaluate the offline bit complexity of the protocol in [1] in order to compare the efficiency with our proposed protocol. Finally, Section 6 concludes the paper and outlines future research directions.

2 Preliminaries

In this paper, let λ denote the security parameter. Let $||$ denote the concatenation of strings. Let $\lceil r \rceil$ for $r \in \mathbb{R}$ denote the minimum integer k satisfying $k \geq r$. For any condition Cond, let $\mathbf{1}\{\mathsf{Cond}\} := 1$ if Cond is true and $\mathbf{1}\{\mathsf{Cond}\} := 0$ if Cond is false. For integers $n \geq 0$ and $k > 0$, let $n \bmod k$ denote the remainder of n modulo k taken from the interval $0 \leq (n \bmod k) \leq k-1$. We write $\mathbb{Z}_N := \mathbb{Z}/N\mathbb{Z}$ in short. Let $(a_{\ell-1} \cdots a_1 a_0)_2$ denote the binary expression of an integer $\sum_{j=0}^{\ell-1} a_j 2^j$.

2.1 Model for MPC

We deal with secure two-party computation (2PC) protocols between parties P_0 and P_1 in the *preprocessing model*, consisting of an input-independent *offline phase* and an input-dependent *online phase*. The protocols in this paper are pure 2PC protocols, unlike the client-aided model [11] where the offline phase is supposed to be executed by a trusted third party. Unlike FSS-based approaches [9] using exponential offline computation, we require both phases to have polynomial computational/communication complexity.

We assume that the two parties are *semi-honest*, and follow the standard security notion in [8]. In this setting, owing to Composition Theorem [8, Theorem 7.3.3], for any protocol Π involving some subprotocol Π', if Π is secure assuming oracle access to the ideal functionality $\mathcal{F}'$ of Π' ($\mathcal{F}'$*-hybrid model*), then Π is secure provided Π' is securely implemented.

2.2 Oblivious Transfer

A 1-out-of-k *oblivious transfer* (OT) between a sender and a receiver is a 2PC protocol where the sender has input $x_0, \ldots, x_{k-1}$, the receiver has input $r \in \{0, \ldots, k-1\}$, and after the protocol, the sender obtains no information on the receiver's input and the receiver obtains x_r and no other information on the sender's input. In this paper, n parallel executions of 1-out-of-k OTs for ℓ-bit inputs for the sender are denoted by $\binom{k}{1}$-OT^n_ℓ, or simply $\binom{k}{1}$-OT_ℓ when $n = 1$. Let $\mathcal{F}_{\mathsf{OT}}$ denote the ideal functionality for OT with suitable parameters.

Here we suppose that an OT is executed by a standard technique as in [13] based on a vector oblivious linear evaluation (VOLE) correlation, which is generated according to [4] where the internal small OTs are executed with the method of [14]. Now the number of communication bits for executing a $\binom{k}{1}$-OT_ℓ (given a VOLE correlation) is $2^{\lceil \log_2 k \rceil}\ell + \lceil \log_2 k \rceil$ bits, while it spends $\lceil \log_2 k \rceil$ dimensions for a VOLE correlation. On the other hand, to generate a VOLE correlation of dimension d, the number of communication bits is approximately $2\mu + \lambda + (\mu + \lambda)\log_2 d$ bits where $\mu = \mu_\lambda$ denotes the bit length of an element of a group for which the CDH assumption holds. See Appendix A for details.

2.3 Secret Sharing

We use the following two kinds of 2-out-of-2 secret sharing (SS) schemes and a share conversion protocol.

2.3.1 $[]^B$-Sharing

A $[]^B$*-share* for $b \in \{0,1\}$ indicates an additive Boolean share $[b]^B = ([b]^B_0, [b]^B_1)$ with $b = [b]^B_0 \oplus [b]^B_1$; we assume that $[b]^B_0$ is held by party P_0 only and $[b]^B_1$ is held by party P_1 only. Now we introduce the following

terminology:

Definition 1. We call a share for $b \in \{0,1\}$ of the form $([b]_0^B, [b]_1^B) = (b, 0)$ (resp. $([b]_0^B, [b]_1^B) = (0, b)$) the *$P_0$-side* (resp. *$P_1$-side*) *trivial* $[]^B$*-share* for b.

2.3.2 $[]^A$-Sharing

A $[]^A$-*share* for $x \in \mathbb{Z}_N$ (where N is a suitably chosen integer) indicates an additive arithmetic share $[x]^A = ([x]_0^A, [x]_1^A)$ with $x = [x]_0^A + [x]_1^A \bmod N$; we assume that $[x]_0^A$ is held by party P_0 only and $[x]_1^A$ is held by party P_1 only.

2.3.3 Share Conversion

A $[]^B$-share $[b]^B$ for $b \in \{0,1\}$ can be converted into a $[]^A$-share $[b]^A$ for b (viewed as an integer) by the following protocol (cf. [17]), which we call $\mathsf{bit2A}$. This protocol requires auxiliary $[]^B$-share $[\beta]^B$ and $[]^A$-share $[\beta]^A$ for some common value $\beta \in \{0,1\}$.

1. For $i = 0, 1$, P_i sends $\alpha_i := [b]_i^B \oplus [\beta]_i^B$ to P_{1-i}.
2. P_0 locally sets $[b]_0^A \leftarrow [\beta]_0^A - 2(\alpha_0 \oplus \alpha_1) \cdot [\beta]_0^A$, and P_1 locally sets $[b]_1^A \leftarrow [\beta]_1^A + (\alpha_0 \oplus \alpha_1) - 2(\alpha_0 \oplus \alpha_1) \cdot [\beta]_1^A$.

$\mathsf{bit2A}$ requires 1 online round and 2-bit online communication. The correctness follows from a straightforward computation:

$$\begin{aligned}[b]_0^A + [b]_1^A &\equiv ([\beta]_0^A + [\beta]_1^A) + (\alpha_0 \oplus \alpha_1) - 2(\alpha_0 \oplus \alpha_1) \cdot ([\beta]_0^A + [\beta]_1^A) \\ &\equiv \beta + (\alpha_0 \oplus \alpha_1) - 2(\alpha_0 \oplus \alpha_1) \cdot \beta \\ &= \beta \oplus \alpha_0 \oplus \alpha_1 = \beta \oplus b \oplus \beta = b \pmod N \ .\end{aligned}$$

The security is satisfied as the bit $[b]_i^B$ is masked by a random bit $[\beta]_i^B$.

We note that the auxiliary inputs $[\beta]^B$ and $[\beta]^A$, where the plaintext space for $[\beta]^A$ is $\mathbb{Z}_N$, can be generated in the offline phase via $\binom{2}{1}$-$\mathsf{OT}_{\lceil \log_2 N \rceil}$ in the following manner, which is secure in the $\mathcal{F}_{\mathsf{OT}}$-hybrid model:

1. P_0 chooses random elements $[\beta]_0^B \in \{0,1\}$ and $[\beta]_0^A \in \mathbb{Z}_N$. P_1 chooses a random bit $[\beta]_1^B$.
2. P_0 as the sender and P_1 as the receiver execute $\binom{2}{1}$-$\mathsf{OT}_{\lceil \log_2 N \rceil}$, where P_0's input is $(([\beta]_0^B - [\beta]_0^A) \bmod N, (1 - [\beta]_0^B - [\beta]_0^A) \bmod N)$ and P_1's input is $[\beta]_1^B$. Then P_1 writes the output of the OT as $[\beta]_1^A$.

2.4 AND Operations for $[]^B$-Shares

Given $[]^B$-shares $[X]^B$ and $[Y]^B$ as inputs, a plain AND operation to compute a $[]^B$-share $[X \wedge Y]^B$ can be realized by using the following plain Beaver Triple (BT) [2]: $[a]^B$, $[b]^B$, and $[c]^B$, where $c = a \wedge b$. A plain BT $([a]^B, [b]^B, [c]^B)$ can be generated via $\binom{4}{1}$-OT_1 in the following manner, which is secure in the $\mathcal{F}_{\mathsf{OT}}$-hybrid model:

1. P_0, as the sender, chooses random bits $[a]_0^B$, $[b]_0^B$, and $[c]_0^B$, and sets the $([a]_1^B \ || \ [b]_1^B)$-th input for $\binom{4}{1}$-OT_1 to be $([a]_0^B \oplus [a]_1^B) \wedge ([b]_0^B \oplus [b]_1^B) \oplus [c]_0^B$ for each $[a]_1^B, [b]_1^B \in \{0,1\}$.
2. P_1, as the receiver, chooses random bits $[a]_1^B$ and $[b]_1^B$, and sets the input for $\binom{4}{1}$-OT_1 to be $[a]_1^B \ || \ [b]_1^B$. Then P_1 writes the output of the OT as $[c]_1^B$.

On the other hand, when $[X]^B$ is a P_0-side trivial $[]^B$-share and $[Y]^B$ is a P_1-side trivial $[]^B$-share (see Definition 1 for the terminology), the AND operation can be realized more efficiently by using "arithmetic blinding" [16] with a modified BT of the form $[a]^B = (a, 0)$, $[b]^B = (0, b)$, and $[c]^B$, where $c = a \wedge b$. We call this AND operation a *half-AND*. Similarly to the case of a plain BT, a modified BT $([a]^B, [b]^B, [c]^B)$ for

Table 3: Functionalities and costs of AND operations.

Operation	Input	Output	Offline cost	Online communication	
				Rounds	Bits
plain AND	$[X]^B$, $[Y]^B$	$[X \wedge Y]^B$	$\binom{4}{1}$-OT_1	1	4
half-AND	$[X]^B = (X, 0)$, $[Y]^B = (0, Y)$	$[X \wedge Y]^B$	$\binom{2}{1}$-OT_1	1	2

half-AND, where $[a]^B = (a, 0)$, $[b]^B = (0, b)$, and $c = a \wedge b$, can be generated via $\binom{2}{1}$-OT_1 where the sender's b-th input is a value corresponding to $[c]_1^B$.

The functionalities and the costs of those (plain or modified) AND operations are given in Table 3. We may write a plain AND operation and a half-AND operation simply by using the binary operation symbol $\wedge$.

3 Auxiliary One-Round Equality Testing Protocol

In this section, we give an equality testing protocol for plaintexts with one online communication round, which will be used in our comparison protocol. Although the protocol here for n_0-bit inputs requires $\Theta(2^{n_0})$-bit communication, it will not affect the polynomial complexity of the main protocol with n-bit inputs because this equality testing protocol is used with parameter $n_0 = O(\log n)$ owing to the fact that the bit length of the sum of all bits in an n-bit input is $O(\log n)$.

Let P_0's input be $x \in \mathbb{Z}_N$ and P_1's input be $y \in \mathbb{Z}_N$ for some parameter N. Our protocol is based on the following idea: let $E_0 = (E_{0,0}, E_{0,1}, \ldots, E_{0,N-1})$ be a one-hot vector encoding of x, i.e., $E_{0,i} = 1$ if $i = x$ and $E_{0,i} = 0$ if $i \neq x$. Similarly, let $E_1 = (E_{1,0}, E_{1,1}, \ldots, E_{1,N-1})$ be a one-hot vector encoding of y. Then the inner product of E_0 and E_1 gives the bit $\mathbf{1}\{x = y\}$. Precisely, the protocol proceeds as follows:

1. For $i = 0, 1, \ldots, N-1$, P_0 sets $[E_{0,i}]_0^B \leftarrow \mathbf{1}\{x = i\}$ and $[E_{1,i}]_0^B \leftarrow 0$; and P_1 sets $[E_{0,i}]_1^B \leftarrow 0$ and $[E_{1,i}]_1^B \leftarrow \mathbf{1}\{y = i\}$.
2. The parties compute $[\mathsf{eq}]^B \leftarrow \bigoplus_{i=0}^{N-1} [E_{0,i}]^B \wedge [E_{1,i}]^B$ by half-ANDs (see Section 2.4), and output $[\mathsf{eq}]^B$.

Correctness. We note that for each $i = 0, 1, \ldots, N-1$, we have $E_{0,i} \wedge E_{1,i} = 1$ if and only if $x = i = y$. Therefore, if $x \neq y$ then we have $E_{0,i} \wedge E_{1,i} = 0$ for every i, while if $x = y$ then we have $E_{0,i} \wedge E_{1,i} = 1$ for precisely one i, i.e., $i = x\ (= y)$. Hence we have $\mathsf{eq} = \mathbf{1}\{x = y\}$, as desired.

Security. The security of the protocol follows from the security of half-ANDs and Composition Theorem.

Round/Bit Complexity. The protocol requires 1 online round (for half-ANDs in Step 2) in total. The number of online communication bits is $2N = O(2^{n_0})$ bits for computing N half-ANDs in Step 2, and the number of offline communication bits is $3N = O(2^{n_0})$ bits (except for the setup of a VOLE correlation) for executing $\binom{2}{1}$-OT_1^N in generating N BTs for half-ANDs.

4 Our Proposed Comparison Protocol

4.1 A Three-Round Generic Construction

First, we give a three-round comparison protocol using any one-round equality testing protocol as a subprotocol. Let P_0's input be $x = (x_{n-1} \cdots x_1 x_0)_2$ and P_1's input be $y = (y_{n-1} \cdots y_1 y_0)_2$. Our protocol here is based on the following property: we have $x < y$ if and only if there is some (in fact, a unique) index i for which $x_j = y_j$ for every $i + 1 \leq j \leq n - 1$ and $x_i < y_i$. Now the former property, i.e., $x_j = y_j$ for every

$i+1 \le j \le n-1$, is checked by compressing at most $n-1$ bits $\mathbf{1}\{x_j \neq y_j\}$ into a single $O(\log n)$-bit integer and using a one-round equality testing protocol. Specifically, the protocol is described as follows:

1. (a) For $i = n-1, \ldots, 2, 1$, P_0 sets $[\mathsf{neq}_i]_0^B \leftarrow x_i$; P_1 sets $[\mathsf{neq}_i]_1^B \leftarrow y_i$; and the parties compute $[\mathsf{neq}_i]^A \leftarrow \mathsf{bit2A}([\mathsf{neq}_i]^B)$ (Round 1) where the plaintext space for $[\mathsf{neq}_i]^A$ is set to $\mathbb{Z}_n$.
   (b) For $i = n-1, \ldots, 1, 0$, P_0 sets $[\neg x_i]_0^B \leftarrow x_i \oplus 1$ and $[y_i]_0^B \leftarrow 0$; P_1 sets $[\neg x_i]_1^B \leftarrow 0$ and $[y_i]_1^B \leftarrow y_i$; and the parties compute $[\mathsf{lt}_i]^B \leftarrow [\neg x_i]^B \wedge [y_i]^B$ by half-ANDs (see Section 2.4) (Round 1).
2. (a) For $i = n-1, \ldots, 2, 1$, the parties locally compute $[\mathsf{neq}_{[n-1,i]}]^A \leftarrow \sum_{j=i}^{n-1} [\mathsf{neq}_j]^A$.
   (b) Then for $i = n-1, \ldots, 2, 1$, the parties compute a $[]^B$-share $[Z_i]^B$ of the bit $Z_i := \mathbf{1}\{[\mathsf{neq}_{[n-1,i]}]_0^A = (n - [\mathsf{neq}_{[n-1,i]}]_1^A) \bmod n\}$ by using some one-round equality testing protocol (Round 2).
3. The parties compute $[\mathsf{lt}]^B \leftarrow [\mathsf{lt}_{n-1}]^B \oplus \bigoplus_{i=0}^{n-2} ([Z_{i+1}]^B \wedge [\mathsf{lt}_i]^B)$ (Round 3) and output $[\mathsf{lt}]^B$.

Correctness. First note that $\mathsf{neq}_i = \mathbf{1}\{x_i \neq y_i\}$ and $\mathsf{lt}_i = \mathbf{1}\{x_i < y_i\}$ for every i. Then for each $i = n-1, \ldots, 2, 1$, we have $Z_i = 1$ (or equivalently, $\mathsf{neq}_{[n-1,i]} = 0$ in $\mathbb{Z}_n$) if and only if $\mathsf{neq}_j = 0$ (i.e., $x_j = y_j$) for every $j = i, i+1, \ldots, n-1$, namely $(x_{n-1}, \ldots, x_{i+1}, x_i) = (y_{n-1}, \ldots, y_{i+1}, y_i)$. Now the final step simply implements the comparison under the lexicographic ordering. Hence we have $\mathsf{lt} = \mathbf{1}\{x < y\}$.

Security. Owing to Composition Theorem, the security of the protocol follows from the security of $\mathsf{bit2A}$, plain ANDs, half-ANDs, and the internal one-round equality testing protocol.

4.2 Our Proposed Two-Round Protocol

Our proposed protocol is basically obtained by instantiating the generic construction in Section 4.1 with the concrete one-round equality testing protocol in Section 3. In this instantiation, for each $i = n-2, \ldots, 1, 0$, the $[]^B$-share $[Z_{i+1}]^B$ is computed as follows:

1. For each $j = 0, 1, \ldots, n-1$:
   - P_0 sets $[\mathsf{eq}_{i+1,0,j}]_0^B \leftarrow \mathbf{1}\{[\mathsf{neq}_{[n-1,i+1]}]_0^A = j\}$, and P_1 sets $[\mathsf{eq}_{i+1,0,j}]_1^B \leftarrow 0$.
   - P_0 sets $[\mathsf{eq}_{i+1,1,j}]_0^B \leftarrow 0$, and P_1 sets
   $$[\mathsf{eq}_{i+1,1,j}]_1^B \leftarrow \mathbf{1}\{n - [\mathsf{neq}_{[n-1,i+1]}]_1^A \bmod n = j\} \ .$$
2. The parties compute $[Z_{i+1}]^B \leftarrow \bigoplus_{j=0}^{n-1} ([\mathsf{eq}_{i+1,0,j}]^B \wedge [\mathsf{eq}_{i+1,1,j}]^B)$.

Then the AND operation $[Z_{i+1}]^B \wedge [\mathsf{lt}_i]^B$ in Round 3 of the generic construction can be absorbed into the AND operations to compute $[Z_{i+1}]^B$ as

$$[Z_{i+1}]^B \wedge [\mathsf{lt}_i]^B = \bigoplus_{j=0}^{n-1} ([\mathsf{eq}_{i+1,0,j}]^B \wedge [\mathsf{eq}_{i+1,1,j}]^B \wedge [\mathsf{lt}_i]^B) \ ,$$

which reduces one round and hence achieves two online rounds. However, this naive construction requires offline-inefficient 3-fan-in ANDs.

To avoid the use of multi-fan-in ANDs, we observe (by using the terminology in Definition 1) that $[\mathsf{eq}_{i+1,0,j}]^B$ is a P_0-side trivial $[]^B$-share, $[\mathsf{eq}_{i+1,1,j}]^B$ is a P_1-side trivial $[]^B$-share, and $[\mathsf{lt}_i]^B$ is the AND of a P_0-side trivial $[]^B$-share $[\neg x_i]^B$ and a P_1-side trivial $[]^B$-share $[y_i]^B$. Moreover, for two P_0-side trivial $[]^B$-shares $[X]^B = (X, 0)$ and $[Y]^B = (Y, 0)$, the $[]^B$-share $(X \wedge Y, 0)$ can be used as a result of $[X]^B \wedge [Y]^B$, which can be obtained locally. The same holds for the AND of two P_1-side trivial $[]^B$-shares. Hence, to

Protocol 1 Our Two-Round Plaintext Comparison Protocol

Input: P_0 has input $x = (x_{n-1} \cdots x_1 x_0)_2$, and P_1 has input $y = (y_{n-1} \cdots y_1 y_0)_2$
Output: $[]^B$-share $[\mathbf{1}\{x < y\}]^B$ of the bit $\mathbf{1}\{x < y\}$

1: **for** $i = n-1, \ldots, 2, 1$ **do**
2: P_0 sets $[\mathsf{neq}_i]_0^B \leftarrow x_i$, and P_1 sets $[\mathsf{neq}_i]_1^B \leftarrow y_i$
3: Compute $[\mathsf{neq}_i]^A \leftarrow \mathsf{bit2A}([\mathsf{neq}_i]^B)$ where the plaintext space for $[\mathsf{neq}_i]^A$ is set to $\mathbb{Z}_n$
4: **end for**
5: P_0 sets $[\neg x_{n-1}]_0^B \leftarrow x_{n-1} \oplus 1$, and P_1 sets $[\neg x_{n-1}]_1^B \leftarrow 0$
6: P_0 sets $[y_{n-1}]_0^B \leftarrow 0$, and P_1 sets $[y_{n-1}]_1^B \leftarrow y_{n-1}$
7: By using half-AND (see Section 2.4), compute $[\mathsf{lt}_{n-1}]^B \leftarrow [\neg x_{n-1}]^B \wedge [y_{n-1}]^B$
8: **for** $i = n-1, \ldots, 2, 1$ **do**
9: Locally compute $[\mathsf{neq}_{[n-1,i]}]^A \leftarrow \sum_{j=i}^{n-1} [\mathsf{neq}_j]^A$
10: **end for**
11: **for** $i = n-2, \ldots, 1, 0$ **do**
12: **for** $j = 0, 1, \ldots, n-1$ **do**
13: P_0 sets

$$[\varepsilon_{i,0,j}]_0^B \leftarrow \mathbf{1}\{[\mathsf{neq}_{[n-1,i+1]}]_0^A = j\} \wedge (x_i \oplus 1) \ ,$$

and P_1 sets $[\varepsilon_{i,0,j}]_1^B \leftarrow 0$
14: P_0 sets $[\varepsilon_{i,1,j}]_0^B \leftarrow 0$, and P_1 sets

$$[\varepsilon_{i,1,j}]_1^B \leftarrow \mathbf{1}\{(n - [\mathsf{neq}_{[n-1,i+1]}]_1^A) \bmod n = j\} \wedge y_i$$

15: **end for**
16: **end for**
17: By using half-AND, compute

$$[\mathsf{lt}]^B \leftarrow [\mathsf{lt}_{n-1}]^B \oplus \bigoplus_{i=0}^{n-2} \bigoplus_{j=0}^{n-1} \left([\varepsilon_{i,0,j}]^B \wedge [\varepsilon_{i,1,j}]^B\right)$$

18: **output** $[\mathsf{lt}]^B$

compute $[Z_{i+1}]^B \wedge [\mathsf{lt}_i]^B$, we can first compute $[\mathsf{eq}_{i+1,0,j}]^B \wedge [\neg x_i]^B$ and $[\mathsf{eq}_{i+1,1,j}]^B \wedge [y_i]^B$ locally and then compute their AND by using a (not multi-fan-in) half-AND:

$$\begin{aligned}
&[\mathsf{eq}_{i+1,0,j}]^B \wedge [\mathsf{eq}_{i+1,1,j}]^B \wedge [\mathsf{lt}_i]^B \\
&= [\mathsf{eq}_{i+1,0,j}]^B \wedge [\mathsf{eq}_{i+1,1,j}]^B \wedge \left([\neg x_i]^B \wedge [y_i]^B\right) \\
&= \left([\mathsf{eq}_{i+1,0,j}]^B \wedge [\neg x_i]^B\right) \wedge \left([\mathsf{eq}_{i+1,1,j}]^B \wedge [y_i]^B\right)
\end{aligned}$$

and the two terms in the parentheses in the right-hand side can be locally computed and are again trivial $[]^B$-shares.

Based on the argument, we obtain our two-round protocol as in Protocol 1. The correctness of Protocol 1 follows from the argument above.

Security. The security of Protocol 1 follows from Composition Theorem and the security of $\mathsf{bit2A}$ and half-ANDs.

4.3 Analysis of Bit Complexity

4.3.1 Online Bit Complexity

In Protocol 1, online communication occurs only in Lines 3, 7, and 17. For these lines, we have the following:

- For Line 3, $\mathsf{bit2A}$ is executed $n-1$ times in parallel, and each $\mathsf{bit2A}$ requires 2-bit communication. Hence the total number of communication bits for Line 3 is $2(n-1)$ bits.
- For Line 7, a single half-AND is executed, which requires 2-bit communication.
- For Line 17, half-AND is executed $(n-1)n$ times in parallel, and each half-AND requires 2-bit communication. Hence the total number of communication bits for Line 17 is $2(n-1)n$ bits.

Summarizing, the total number of online communication bits is $2(n-1)+2+2(n-1)n = 2n^2$ bits.

4.3.2 Offline Bit Complexity

In Protocol 1, offline communication is required only for Lines 3, 7, and 17. For these lines, we have the following:

- For Line 3, $\mathsf{bit2A}$ is executed $n-1$ times, and each $\mathsf{bit2A}$ requires $\binom{2}{1}\text{-}\mathsf{OT}_{n_0}$ (where $n_0 := \lceil \log_2 n \rceil$) in the offline phase to generate its auxiliary input. Hence the total offline cost for Line 3 is $\binom{2}{1}\text{-}\mathsf{OT}_{n_0}^{n-1}$.
- For Line 7, a single half-AND is executed, which requires $\binom{2}{1}\text{-}\mathsf{OT}_1$ in the offline phase to generate the associated modified BT.
- For Line 17, half-AND is executed $(n-1)n$ times, and each half-AND requires $\binom{2}{1}\text{-}\mathsf{OT}_1$ in the offline phase to generate the associated modified BT. Hence the total offline cost for Line 17 is $\binom{2}{1}\text{-}\mathsf{OT}_1^{(n-1)n}$.

Summarizing, the total offline communication cost for Protocol 1 is $\binom{2}{1}\text{-}\mathsf{OT}_{n_0}^{n-1} + \binom{2}{1}\text{-}\mathsf{OT}_1^{(n-1)n+1}$, which is (according to the description in Section 2.2) equivalent to communicating the following number of bits:

$$\begin{aligned}
&(n-1)\cdot(2^{\lceil \log_2 2\rceil}\cdot n_0 + \lceil \log_2 2\rceil) + ((n-1)n+1)\cdot(2^{\lceil \log_2 2\rceil}\cdot 1 + \lceil \log_2 2\rceil)\\
&= (n-1)(2n_0+1) + (n^2-n+1)\cdot 3\\
&= 2nn_0 + 3n^2 - 2n - 2n_0 + 2 \lessapprox 3n^2 + 2n\log_2 n \text{ (bits)}
\end{aligned}$$

and setting up a VOLE correlation of dimension

$$\begin{aligned}
d &:= (n-1)\cdot\lceil \log_2 2\rceil + ((n-1)n+1)\cdot\lceil \log_2 2\rceil\\
&= (n-1)\cdot 1 + (n^2-n+1)\cdot 1 = n^2 \ .
\end{aligned}$$

Hence, according to the description in Section 2.2, the total number of offline communication bits for Protocol 1 is evaluated as

$$\begin{aligned}
&\approx 3n^2 + 2n\log_2 n + 2\mu + \lambda + (\mu+\lambda)\log_2 d\\
&= 3n^2 + 2n\log_2 n + 2\mu + \lambda + (\mu+\lambda)\cdot 2\log_2 n\\
&= 3n^2 + 2(n+\mu+\lambda)\log_2 n + 2\mu + \lambda \text{ (bits)}.
\end{aligned}$$

5 Offline Bit Complexity of the Protocol in [1]

In order to compare the efficiency with our proposed protocol, in this section, we evaluate the offline bit complexity of the comparison protocol in [1]. Their comparison protocol (Protocol 7 of [1]) for 64-bit integers consists of three executions of $\mathsf{Overflow}$ protocols for 64-bit integers (and local operations). Each $\mathsf{Overflow}$

protocol (Protocol 6 of [1]) consists of $\mathsf{Overflow1}_{2^{\Delta_i}\to N}$ protocols for $i = 0, 1, \ldots, m-1$ and m' executions of $\mathsf{Unitv}_{N\to 2}$ protocol (and local operations) where m, N, Δ_i, and m' are parameters set as $m := 22$, $N := 32$, $\Delta_i := 2$ for $0 \le i \le 1$ and $\Delta_i := 3$ for $2 \le i \le 21$, and $m' := 12$ (see the footnotes for Table 1 of [1]). Each $\mathsf{Overflow1}_{2^t\to N}$ protocol (Protocol 5 of [1]) consists of $\mathsf{Unitv}_{2^{t+1}-1\to N}$ protocol (and local operations). Moreover, the offline phase of $\mathsf{Unitv}_{\nu\to\nu'}$ protocol (Protocol 1 of [1]) consists of $\mathsf{Unitv\text{-}prep}_{\nu\to\nu'}$ protocol. Finally, as shown in [1, Appendix D] (Protocol 20), $\mathsf{Unitv\text{-}prep}_{\nu\to\nu'}$ protocol consists of $\binom{\nu}{1}\text{-}\mathsf{OT}^{\nu}_{\lceil\log_2\nu'\rceil}$ (and local operations). Summarizing, the offline cost for their comparison protocol for 64-bit integers can be evaluated as follows:

$$\begin{aligned}
\mathsf{Comparison} &= 3\cdot\mathsf{Overflow}\\
&= 3\cdot(2\cdot\mathsf{Overflow1}_{2^2\to 32} + 20\cdot\mathsf{Overflow1}_{2^3\to 32} + 12\cdot\mathsf{Unitv}_{32\to 2})\\
&= 6\cdot\mathsf{Overflow1}_{2^2\to 32} + 60\cdot\mathsf{Overflow1}_{2^3\to 32} + 36\cdot\mathsf{Unitv}_{32\to 2}\\
&= 6\cdot\mathsf{Unitv}_{7\to 32} + 60\cdot\mathsf{Unitv}_{15\to 32} + 36\cdot\mathsf{Unitv}_{32\to 2}\\
&= 6\cdot\mathsf{Unitv\text{-}prep}_{7\to 32} + 60\cdot\mathsf{Unitv\text{-}prep}_{15\to 32} + 36\cdot\mathsf{Unitv\text{-}prep}_{32\to 2}\\
&= 6\cdot\binom{7}{1}\text{-}\mathsf{OT}^{7}_{5} + 60\cdot\binom{15}{1}\text{-}\mathsf{OT}^{15}_{5} + 36\cdot\binom{32}{1}\text{-}\mathsf{OT}^{32}_{1}\\
&= \binom{7}{1}\text{-}\mathsf{OT}^{42}_{5} + \binom{15}{1}\text{-}\mathsf{OT}^{900}_{5} + \binom{32}{1}\text{-}\mathsf{OT}^{1152}_{1}\ .
\end{aligned}$$

As described in Section 2.2, the cost of the latter OTs is

$$42\cdot(2^3\cdot 5+3) + 900\cdot(2^4\cdot 5+4) + 1152\cdot(2^5\cdot 1+5) = 120030$$

communication bits and setting up a VOLE correlation of dimension

$$42\cdot 3 + 900\cdot 4 + 1152\cdot 5 = 9486\ .$$

Hence the total number of offline communication bits is

$$\begin{aligned}
&\approx 120030 + 2\mu + \lambda + (\mu+\lambda)\log_2 9486\\
&= 120030 + 15.21\mu + 14.21\lambda\\
&= 129636.4 \text{ (bits)}\\
&= 15.82 \text{ (KiB)}
\end{aligned}$$

where we used $\lambda = 128$ and $\mu = 512$.

6 Conclusion

In this work, we presented a new secure comparison protocol over plaintexts, achieving a constant two rounds in the plain model (without a trusted third party in the offline phase) with an improved total bit complexity. While our contribution significantly advances the state-of-the-art, a challenging direction remains for future research; that is, further reducing the round complexity. This task appears non-trivial. Our two-round construction (Protocol 1) fundamentally relies on a two-round equality testing framework. Achieving a one-round comparison protocol would likely necessitate a one-round equality testing primitive—a feat that surpasses the inherent two-round barrier of GC—or a complete departure from the "leftmost differing bit" strategy employed in this work.

References

[1] Nuttapong Attrapadung, Hiraku Morita, Kazuma Ohara, Jacob C. N. Schuldt, and Kazunari Tozawa. Memory and round-efficient MPC primitives in the pre-processing model from unit vectorization. In Yuji Suga, Kouichi Sakurai, Xuhua Ding, and Kazue Sako, editors, *ASIACCS 22: 17th ACM Symposium on Information, Computer and Communications Security*, pages 858–872, Nagasaki, Japan, May 30 – June 3, 2022. ACM Press.

[2] Donald Beaver. Efficient multiparty protocols using circuit randomization. In Joan Feigenbaum, editor, *Advances in Cryptology - CRYPTO '91, 11th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1991, Proceedings*, volume 576 of *Lecture Notes in Computer Science*, pages 420–432. Springer, 1991.

[3] Elette Boyle, Nishanth Chandran, Niv Gilboa, Divya Gupta, Yuval Ishai, Nishant Kumar, and Mayank Rathee. Function secret sharing for mixed-mode and fixed-point secure computation. In Anne Canteaut and François-Xavier Standaert, editors, *Advances in Cryptology – EUROCRYPT 2021, Part II*, volume 12697 of *Lecture Notes in Computer Science*, pages 871–900, Zagreb, Croatia, October 17–21, 2021. Springer, Cham, Switzerland.

[4] Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, Peter Rindal, and Peter Scholl. Efficient two-round OT extension and silent non-interactive secure computation. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS 2019, London, UK, November 11-15, 2019*, pages 291–308. ACM, 2019.

[5] Jung Hee Cheon, Dongwoo Kim, and Duhyeong Kim. Efficient homomorphic comparison methods with optimal complexity. In Shiho Moriai and Huaxiong Wang, editors, *Advances in Cryptology - ASIACRYPT 2020 - 26th International Conference on the Theory and Application of Cryptology and Information Security, Daejeon, South Korea, December 7-11, 2020, Proceedings, Part II*, volume 12492 of *Lecture Notes in Computer Science*, pages 221–256. Springer, 2020.

[6] Ye Dong, Xudong Chen, Xiangfu Song, Yaxi Yang, Wen-Jie Lu, Tianwei Zhang, Jianying Zhou, and Jin-Song Dong. Alkaid: Accelerating three-party boolean circuits by mixing correlations and redundancy. *IEEE Transactions on Information Forensics and Security*, 21:640–654, 2026.

[7] Juan A. Garay, Berry Schoenmakers, and José Villegas. Practical and secure solutions for integer comparison. In Tatsuaki Okamoto and Xiaoyun Wang, editors, *PKC 2007: 10th International Conference on Theory and Practice of Public Key Cryptography*, volume 4450 of *Lecture Notes in Computer Science*, pages 330–342, Beijing, China, April 16–20, 2007. Springer Berlin Heidelberg, Germany.

[8] Oded Goldreich. *The Foundations of Cryptography - Volume 2: Basic Applications*. Cambridge University Press, 2004.

[9] Xiaojie Guo, Kang Yang, Xiao Wang, Wenhao Zhang, Xiang Xie, Jiang Zhang, and Zheli Liu. Half-tree: Halving the cost of tree expansion in COT and DPF. In Carmit Hazay and Martijn Stam, editors, *Advances in Cryptology – EUROCRYPT 2023, Part I*, volume 14004 of *Lecture Notes in Computer Science*, pages 330–362, Lyon, France, April 23–27, 2023. Springer, Cham, Switzerland.

[10] Tianpei Lu, Xin Kang, Bingsheng Zhang, Zhuo Ma, Xiaoyuan Zhang, Yang Liu, Kui Ren, and Chun Chen. Efficient 2pc for constant round secure equality testing and comparison. In Lujo Bauer and Giancarlo Pellegrino, editors, *34th USENIX Security Symposium, USENIX Security 2025, Seattle, WA, USA, August 13-15, 2025*, pages 1985–2004. USENIX Association, 2025.

[11] Payman Mohassel and Yupeng Zhang. SecureML: A system for scalable privacy-preserving machine learning. In *2017 IEEE Symposium on Security and Privacy, SP 2017, San Jose, CA, USA, May 22-26, 2017*, pages 19–38. IEEE Computer Society, 2017.

[12] Hiraku Morita, Nuttapong Attrapadung, Tadanori Teruya, Satsuya Ohata, Koji Nuida, and Goichiro Hanaoka. Constant-round client-aided secure comparison protocol. In Javier López, Jianying Zhou, and Miguel Soriano, editors, *Computer Security - 23rd European Symposium on Research in Computer Security, ESORICS 2018, Barcelona, Spain, September 3-7, 2018, Proceedings, Part II*, volume 11099 of *Lecture Notes in Computer Science*, pages 395–415. Springer, 2018.

[13] Moni Naor and Benny Pinkas. Oblivious transfer and polynomial evaluation. In Jeffrey Scott Vitter, Lawrence L. Larmore, and Frank Thomson Leighton, editors, *Proceedings of the Thirty-First Annual ACM Symposium on Theory of Computing, May 1-4, 1999, Atlanta, Georgia, USA*, pages 245–254. ACM, 1999.

[14] Moni Naor and Benny Pinkas. Efficient oblivious transfer protocols. In S. Rao Kosaraju, editor, *Proceedings of the Twelfth Annual Symposium on Discrete Algorithms, January 7-9, 2001, Washington, DC, USA*, pages 448–457. ACM/SIAM, 2001.

[15] Takashi Nishide and Kazuo Ohta. Multiparty computation for interval, equality, and comparison without bit-decomposition protocol. In Tatsuaki Okamoto and Xiaoyun Wang, editors, *Public Key Cryptography - PKC 2007, 10th International Conference on Practice and Theory in Public-Key Cryptography, Beijing, China, April 16-20, 2007, Proceedings*, volume 4450 of *Lecture Notes in Computer Science*, pages 343–360. Springer, 2007.

[16] Satsuya Ohata and Koji Nuida. Communication-efficient (client-aided) secure two-party protocols and its application. In Joseph Bonneau and Nadia Heninger, editors, *Financial Cryptography and Data Security - 24th International Conference, FC 2020, Kota Kinabalu, Malaysia, February 10-14, 2020 Revised Selected Papers*, volume 12059 of *Lecture Notes in Computer Science*, pages 369–385. Springer, 2020.

[17] Arpita Patra, Thomas Schneider, Ajith Suresh, and Hossein Yalame. ABY2.0: improved mixed-protocol secure two-party computation. In Michael D. Bailey and Rachel Greenstadt, editors, *30th USENIX Security Symposium, USENIX Security 2021, August 11-13, 2021*, pages 2165–2182. USENIX Association, 2021.

[18] Sijun Tan, Brian Knott, Yuan Tian, and David J. Wu. CryptGPU: Fast privacy-preserving machine learning on the GPU. In *42nd IEEE Symposium on Security and Privacy, SP 2021, San Francisco, CA, USA, 24-27 May 2021*, pages 1021–1038. IEEE, 2021.

[19] Xiao Wang, Alex J. Malozemoff, and Jonathan Katz. EMP-toolkit: Efficient MultiParty computation toolkit. `https://github.com/emp-toolkit`, 2016.

[20] Andrew Chi-Chih Yao. Protocols for secure computations (extended abstract). In *23rd Annual Symposium on Foundations of Computer Science, Chicago, Illinois, USA, 3-5 November 1982*, pages 160–164. IEEE Computer Society, 1982.

[21] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *27th Annual Symposium on Foundations of Computer Science, Toronto, Canada, 27-29 October 1986*, pages 162–167. IEEE Computer Society, 1986.

A OT from VOLE Correlations

By a VOLE correlation with parameters p (prime), m, and d (positive integers), we mean random vectors $\vec{\rho} = (\rho_0, \ldots, \rho_{d-1}) \in (\mathbb{F}_p)^d$ and $\vec{w} = (w_0, \ldots, w_{d-1}), \vec{w'} = (w'_0, \ldots, w'_{d-1}) \in (\mathbb{F}_{p^m})^d$ and a random scalar $\sigma \in \mathbb{F}_{p^m}$ with the relation $\vec{w'} = \sigma \cdot \vec{\rho} + \vec{w}$, where one party holds $(\sigma, \vec{w'})$ and the other party holds $(\vec{\rho}, \vec{w})$. Here we typically consider the case of $p = 2$ and $m = \lambda$, while the dimension d should depend on the number of required OT executions.

Given such a VOLE correlation, a $\binom{k}{1}$-OT_ℓ can be executed as follows, where the sender holds $(\sigma, \vec{w'})$, the receiver holds $(\vec{\rho}, \vec{w})$, and ct denotes a counter held by both parties (a variant of the method in [13]):

1. Let $x_0, x_1, \ldots, x_{k-1} \in \{0,1\}^\ell$ be the inputs of the sender, and let $r \in \{0, 1, \ldots, k-1\}$ be the input of the receiver. Set $\nu \leftarrow \lceil \log_2 k \rceil$. If $k < 2^\nu$, then the sender sets $x_i \leftarrow \vec{0}$ for each $i = k, k+1, \ldots, 2^\nu - 1$.
2. For $i = 0, 1, \ldots, \nu - 1$:
   (a) The sender sets $v_{i,0} \leftarrow w'_{\mathsf{ct}}$ and $v_{i,1} \leftarrow v_{i,0} - \sigma$. The receiver sets $\rho'_i \leftarrow \rho_{\mathsf{ct}}$ and $v'_i \leftarrow w_{\mathsf{ct}}$. (Note that now $v'_i = v_{i,\rho'_i}$.)
   (b) Set $\mathsf{ct} \leftarrow \mathsf{ct} + 1$.
3. The receiver sets $\rho' \leftarrow (\rho'_{\nu-1} \cdots \rho'_1 \rho'_0)_2$ and $r' \leftarrow \rho' - r \bmod 2^\nu$, and sends r' to the sender.
4. For $i = 0, 1, \ldots, 2^\nu - 1$:
   (a) The sender parses $r' + i \bmod 2^\nu$ as $(j_{\nu-1} \cdots j_1 j_0)_2$.
   (b) The sender computes $K_i \leftarrow H(v_{\nu-1,j_{\nu-1}} \,||\, \cdots \,||\, v_{1,j_1} \,||\, v_{0,j_0} \,||\, i) \in \{0,1\}^\ell$ where H is a cryptographic hash function (modeled as a random oracle), sets $x'_i \leftarrow K_i \oplus x_i$, and sends x'_i to the receiver.
5. The receiver outputs $H(v'_{\nu-1} \,||\, \cdots \,||\, v'_1 \,||\, v'_0 \,||\, r) \oplus x'_r$.

The number of communication bits of this protocol (given a VOLE correlation) is $2^{\lceil \log_2 k \rceil}\ell + \lceil \log_2 k \rceil$ bits, while it spends $\lceil \log_2 k \rceil$ dimensions for a VOLE correlation. On the other hand, to generate a VOLE correlation of dimension d according to [4], the communication cost is $\lceil \log_2(cd) \rceil = \lceil \log_2 d + \log_2 c \rceil \approx \log_2 d$ executions of $\binom{2}{1}$-OT_λ for a small constant $c > 1$ and λ-bit communication. When the latter $\binom{2}{1}$-OT_λ's are executed with the method of [14], the number of communication bits for the $\log_2 d$ executions is $2\mu + (\mu + \lambda)\log_2 d$ where $\mu = \mu_\lambda$ denotes the bit length of an element of a group for which the CDH assumption holds. Hence the number of communication bits for generating a VOLE correlation is approximately $2\mu + \lambda + (\mu + \lambda)\log_2 d$ bits.