New Techniques for Communication-Efficient Secure Comparison Protocols

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

April 24, 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 compared to garbled circuits (GC), but they historically suffer from higher (online) round complexity: while GC-based comparison ends in two rounds, the state-of-the-art SS-based (plaintext) comparison protocol requires three rounds (Lu et al., USENIX Security 2025). To break the barrier, in this paper we propose the first SS-based comparison protocol, built upon "round absorption" via multi-fan-in gates, to match the two-round complexity of GC with online bit complexity $O(n \log n)$ significantly lower than GC-based $O(\lambda n)$. We also propose the second two-round protocol, built upon a new optimization technique for multiplication, that addresses the drawback of $O(n^3)$ offline bit complexity in our first protocol and reduces it to $O(n^2)$ at the cost of increasing the online bit complexity to also $O(n^2)$.

Furthermore, for the purpose of optimization in bandwidth, we propose the third (not constant-round) protocol with asymptotically $6n$ online bit complexity, which is significantly lower than asymptotically $8n$ bits of the state-of-the-art protocol (Couteau, ACNS 2018). Our protocol adopts the framework based on ternary trees and quaternary integers of CrypTFlow2 (ACM CCS 2020) and its followers, but departs from their oblivious-transfer-based computation at each input digit. Instead, we use a Boolean-circuit-based approach driven by a new custom-tailored formula for processing quaternary integers and a specialized multiplication protocol. The latter technique of "multiplication involving an auxiliary input held by a single party" may be of independent interest.

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 [25]. 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 [23], reported that operations involving comparison (specifically ReLU and SoftMax) accounted for 42–73% of the total execution time (Table 5 in [23]). While numerous studies have focused on improving the efficiency of secure comparison protocols (cf. Table 1 below), it continues to be a dominant performance bottleneck in practical privacy-preserving applications. In particular, while 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), they suffer from higher round complexity than approaches based on other cryptographic primitives such as garbled circuits (GC). Specifically, although the state-of-the-art SS-based plaintext comparison protocol of Lu et al. [14] has achieved a constant

three-round online complexity (as demonstrated in [14], such comparison over plaintexts can be extended to secret-shared inputs by sacrificing a single bit of the plaintext space), it is still higher than the GC-based two-round complexity. It has been a long-standing open problem to realize SS-based comparison achieving the same two-round complexity as GC.

1.1 Our Contribution

We propose the first SS-based comparison protocol with a constant two-round online complexity, solving the aforementioned open problem of achieving the same round complexity as GC-based protocols. Our protocol also maintains the $O(n \log n)$ online bit complexity of Lu et al.'s three-round protocol [14], which is significantly lower than the GC-based $O(\lambda n)$. A drawback of our protocol is the $O(n^3)$ offline bit complexity (in comparison to e.g., GC-based $O(\lambda n)$): to address the issue, we propose the second two-round protocol with offline bit complexity $O(n^2)$, while it increases the online bit complexity to also $O(n^2)$.

Furthermore, we also propose the third (not constant, approximately $(\log_3 n)$-round) protocol with asymptotically only $6n$-bit online communication. Except for protocols based on function secret sharing (FSS) [11, 2], which incur exponential offline costs, the state-of-the-art protocol from the viewpoint of online bit complexity (for which both offline and online phases are fully specified[1]) is the asymptotically $8n$-bit protocol by Couteau [5]. Our protocol reduces the asymptotic online bit complexity to 75% of Couteau's protocol.

A summary of these protocols alongside existing works is provided in Table 1. We also compare the concrete efficiency of our proposed protocols with the existing protocols by Lu et al. [14] and by Couteau [5] for the case of 64-bit inputs, where the parameters are set to $\lambda = 128$ and $\mu = 512$ (see Table 1 for the meaning of these parameters). See Table 2 for the result.

1.2 Technical Overview

Protocol 1 (Section 4): Achieving Two-Round Complexity via "Round Absorption". We first give an auxiliary generic construction of a three-round comparison protocol for n-bit inputs $x = (x_{n-1} \cdots x_1 x_0)_2$ and $y = (x_{n-1} \cdots x_1 x_0)_2$ using any one-round equality testing protocol for $O(\log n)$-bit inputs (having the same three-round complexity as Lu et al.'s protocol [14] but with a different structure) 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 digit, 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 $\mathbf{1}\{x < y\} = \mathsf{lt}_{n-1} \oplus \bigoplus_{i=0}^{n-2} (Z_{i+1} \wedge \mathsf{lt}_i)$.

Then we instantiate the internal one-round equality testing protocol as in Section 3.1. In this instantiation, a share of each Z_{i+1} is computed in Round 2 by an n_0-fan-in AND with $n_0 = \lceil \log_2 n \rceil$. Consequently, the AND operation $Z_{i+1} \wedge \mathsf{lt}_i$ in Round 3 can be absorbed into Round 2 using an $(n_0 + 1)$-fan-in AND, resulting in a two-round protocol. We emphasize that such a "round absorption" technique is not applicable to Lu et al.'s protocol due to its different structure.

Protocol 2 (Section 5): Balancing Offline/Online Bit Complexity via New Multiplication Technique for Trivial Shares. In order to address the issue of the higher offline bit complexity in Protocol 1 due to the use of larger fan-in ANDs, here we instantiate the internal equality testing protocol in our

[1] [8] claims an online bit complexity of less than $7n$. However, the manuscript does not provide a clear breakdown of this complexity, and the configuration of the corresponding offline phase is omitted, making it difficult to conduct a fair and comprehensive evaluation. Consequently, we have excluded this protocol from Table 1 to ensure a comparison based on fully specified protocols. Note that even when taking their protocol into account, our protocol remains more efficient.

Table 1: Comparison with the previous plaintext comparison protocols for n-bit integers; λ denotes the security parameter; μ denotes the bit length of a group element satisfying a suitable computational assumption (e.g., CDH assumption); $\log_2^*$ represents the iterated logarithm with base 2; (†) under correlated keys generation scheme which performs $O(2^n)$ times Hash locally; (††) under a trusted third-party dealer; and the values (and remarks) for previous protocols are quoted from [14, Table 1].

Approach	Protocol	Offline	Online	
		Communication	Communication	# of rounds
GC-based	Yao [24, 26]	$2n\lambda$	$2n\lambda$	2
HE-based	GSV [9]	–	$18\mu n + 8\mu$	9
FSS-based	Half-Tree [11]	$(n+2)\lambda$ (†)	$2n$	1
	DCF [2]	$4n(\lambda+1)+\lambda+n$ (††)	$2n$	1
OT-based	Couteau [5]	$6\lambda n$	$8n + 2\log_2 n$	$4\log_2^*\lambda + 5$
	CrypTFlow2 [22]	–	$\lambda n + 14n$	$\log_2 n + 4$
	Lu et al. [14]	$\approx 2n\lambda\log_2(2n) + n\log_2^2 n$	$(2n+3)(\log_2 n + 2)$	3
Generic 2PC	ABY [6]	$6\lambda n + 17\lambda + n$	$2\lambda n + 20n$	$\log_2 n + 5$
	ABY2.0 [21]	$5\lambda n + 17\lambda + 2n$	$\lambda n + 9n$	$\log_2 n + 4$
	Ours $\Pi_{\mathsf{cmp},1}$ (§ 4)	$\approx 4n^3 + 2\mu + \lambda$ $+ (\mu+\lambda)(\log_2 n + \log_2^2 n)$	$2(n-1)\log_2 n$ $+ 6n - 4$	2
	Ours $\Pi_{\mathsf{cmp},2}$ (§ 5)	$\approx 3n^2 + 2\mu + \lambda$ $+ 2(n+\mu+\lambda)\log_2 n$	$2n^2 - 2$	2
	Ours $\Pi_{\mathsf{cmp},3}$ (§ 6)	$\approx 44n + 4\mu + 3\lambda$ $+ (\mu+\lambda)\log_2 n$	$3n + 12\lceil (n-2)/4 \rceil$ $- 2\log_3(n/2)$ $\sim 6n$	$\log_3(n/2) + 1$

generic construction as in Section 3.2 (instead of Section 3.1). Now each bit Z_{i+1} is computed as XOR of plain ANDs, say $Z_{i+1} = \bigoplus_j X_{i+1,j} \wedge Y_{i+1,j}$. Consequently, a naive application of the same technique as in Protocol 1 already achieves two-round complexity using 3-fan-in ANDs $X_{i+1,j} \wedge Y_{i+1,j} \wedge \mathsf{lt}_i$ instead of (n_0+1)-fan-in ANDs.

For more efficiency improvement, here 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. This removal of larger fan-in ANDs reduces the offline bit complexity in contrast to Protocol 1, at the cost of increasing the online bit complexity due to the larger number of AND operations.

Protocol 3 (Section 6): Reducing Online Bit Complexity via Custom-Tailored Formula for Quaternary Integers. This protocol builds upon the quaternary integer representation introduced by CrypTFlow2 [22] and the ternary-tree architecture introduced by Feng et al. [8], with the following two properties:

- Similarly to [8, 12, 22], the inputs x and y are parsed in quaternary (instead of binary) expressions as $x = (x_{\lceil n/2 \rceil - 1} \cdots x_1 x_0)_4$ and $y = (y_{\lceil n/2 \rceil - 1} \cdots y_1 y_0)_4$.
- Similarly to [8], the partial (shared) results $\mathsf{lt}_i = \mathbf{1}\{x_i < y_i\}$ and $\mathsf{eq}_i = \mathbf{1}\{x_i = y_i\}$ are combined by using a ternary (instead of binary) tree.

Table 2: Efficiency comparison between our protocols and (selected) existing protocols. The parameters (see Table 1) are set to $n = 64$, $\lambda = 128$, and $\mu = 512$.

Protocol	Offline	Online	
	Communication	Communication	# of rounds
Lu et al. [14]	14.3 KiB	131 B	3
Ours $\Pi_{\mathsf{cmp},1}$ (§ 4)	131.4 KiB	142 B	**2**
Ours $\Pi_{\mathsf{cmp},2}$ (§ 5)	2.7 KiB	1024 B	**2**
Couteau [5]	6.0 KiB	62.5 B	33
Ours $\Pi_{\mathsf{cmp},3}$ (§ 6)	1.1 KiB	**47.3 B**	5

To compute lt_i and eq_i, the protocols of [8, 12, 22] used oblivious transfer (OT), which would incur an inevitable lower bound (input number) $\times$ (input length) for *online* bit complexity. To move the dominant cost to the *offline* phase (generating Beaver triples (BTs)), here we adopt a more traditional Boolean circuit-based approach. However, a naive construction of a circuit for, say the comparison for quaternary integers such as $\mathbf{1}\{x_i < y_i\} = \mathbf{1}\{x_i = 0\} \wedge \mathbf{1}\{y_i \in \{1,2,3\}\} \oplus \mathbf{1}\{x_i = 1\} \wedge \mathbf{1}\{y_i \in \{2,3\}\} \oplus \mathbf{1}\{x_i = 2\} \wedge \mathbf{1}\{y_i = 3\}$ will result in too large online overhead due to the many AND operations that are not reusable. To address this challenge, we develop a custom-tailored Boolean circuit by searching for decompositions of truth tables for comparison/equality of quaternary integers involving as many common (reusable) factors as possible (Theorem 1). To implement it, we also develop a new correlated AND operation using specialized BTs (referred to as 'IV-AND' in Section 2.4). Then we can compute lt_i and eq_i with only 6-bit online communication. We note that an OT-based computation requires $2 \cdot (4 + 1) = 10$ bits, and the technique of [13] using one-time truth tables (with 2-bit inputs) requires $2 \cdot (2 + 2) = 8$ bits. We also note that our new correlated AND operation involves an AND operation on share values held by only one party. The fact that such a technique is effective in MPC is of independent interest.

1.3 Related Work

Constant-Round Protocols. Generally, garbled circuits (GC) [24, 26] provide the baseline for constant (two)-round comparison but incur high bandwidth costs $O(\lambda n)$. Constant-round comparison protocols based on (fully) homomorphic encryption (HE) are also proposed [9, 19, 4], 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 [16] based on the client-aided model [15]) or expensive pre-computation (e.g., the use of multi-fan-in gates [20, 21, 7] or function secret sharing (FSS) [11, 2]). Recently, Lu et al. [14] established the state-of-the-art for SS-based comparison, achieving three rounds. However, matching the two-round efficiency of GC within the SS framework has remained an elusive goal. Our Protocols 1 and 2 directly address this gap.

Bandwidth-Optimized Protocols. Couteau [5] proposed an asymptotically $8n$-bit protocol, though with logarithmic rounds. More recent works like CrypTFlow2 [22] and Panther [8] leverage tree structures and higher-radix representations (e.g., quaternary); however, these methods typically rely on OT for per-digit computation, which dictates a lower bound on online communication. In contrast, our Protocol 3 adopts a circuit-based approach for the quaternary integers, pushing the online complexity down to asymptotically $6n$ bits.

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

Table 3: Functionalities and costs for share conversion.

Protocol	Input		Output	Online communication	
	Main	Auxiliary		Rounds	Bits
$\mathsf{bit2T}$	$[b]^B$	$[\beta]^B$ for some $\beta \in \{0,1\}$	$\langle b \rangle$ consistent with $[\beta]^B$	1	2
$\mathsf{T2A}$	$\langle b \rangle = ([\delta_b]_0^B, [\delta_b]_1^B; \Delta_b)$	$[\delta_b]^A$	$[b]^A$	0	0
$\mathsf{bit2A} = \mathsf{T2A} \circ \mathsf{bit2T}$	$[b]^B$	$[\beta]^B$, $[\beta]^A$ for some $\beta \in \{0,1\}$	$[b]^A$	1	2

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, and write $[k] := \{0, 1, \ldots, k-1\}$ for an integer $k \geq 1$. For an integer $M \geq 2$, let $(a_{\ell-1} \cdots a_1 a_0)_M$ denote the M-ary expression of an integer $\sum_{j=0}^{\ell-1} a_j M^j$.

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 purely 2PC protocols, unlike the client-aided model [15] where the offline phase is supposed to be executed by a trusted client other than the two parties. Our main aim is to reduce the communication complexity (the numbers of communication rounds and communication bits) for the online phase, while we also require the offline phase to still have polynomial computational/communication complexity (unlike FSS-based approaches [11] using exponential offline computation).

We assume that the two parties are *semi-honest*, and follow the standard security notion in [10]. In this setting, owing to Composition Theorem [10, 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. Most of the protocols proposed in this paper can be easily proven secure in such a hybrid model.

2.2 Secret Sharing

We use the following three kinds of 2-out-of-2 secret sharing (SS) schemes. Some of these shares can be converted into other kinds of shares by using protocols as in [21]; see Table 3 for their functionality and complexity (where the 'T' stands for "triplets"). We note that the auxiliary inputs $[\beta]^B$ and $[\beta]^A$ for $\mathsf{bit2A}$ can be generated in the offline phase by a standard technique using oblivious transfer, specifically $\binom{2}{1}$-$\mathsf{OT}_{\lceil \log_2 N \rceil}$ (see Section 2.3 below for the notation) where the plaintext space for $[\beta]^A$ is $\mathbb{Z}_N$. See Appendix A for details.

2.2.1 $[]^B$-Sharing

It is the ordinary additive SS scheme for bits: for a share ($[]^B$*-share*) $[b]^B := ([b]_0^B, [b]_1^B)$ for $b \in \{0,1\}$, we have $b = [b]_0^B \oplus [b]_1^B$. we assume that $[b]_0^B$ is held by party P_0 only and $[b]_1^B$ 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.

2.2.2 $[]^A$-Sharing

It is the ordinary additive SS scheme for integers in $\mathbb{Z}_N$ for a suitably chosen integer N: for a share ($[]^A$-*share*) $[x]^A := ([x]_0^A, [x]_1^A)$ for $x \in \mathbb{Z}_N$, we have $x = [x]_0^A + [x]_1^A \bmod N$. We assume that $[x]_0^A$ is held by party P_0 only and $[x]_1^A$ is held by party P_1 only.

2.2.3 $\langle\rangle$-Sharing ([21])

For $b \in \{0,1\}$, we let $[\delta_b]_0^B$, $[\delta_b]_1^B$, and Δ_b be a triplet of bits satisfying the relation $[\delta_b]_0^B \oplus [\delta_b]_1^B \oplus \Delta_b = b$, and let $[\delta_b]_0^B$ be held by party P_0 only, $[\delta_b]_1^B$ be held by party P_1 only, and Δ_b be held by both parties. We write the share for b as $\langle b\rangle = ([\delta_b]_0^B, [\delta_b]_1^B; \Delta_b)$, and call it a $\langle\rangle$-*share* for b. We note that the pair ($[\delta_b]_0^B$, $[\delta_b]_1^B$) can be viewed as a $[]^B$-share for the bit $\delta_b := [\delta_b]_0^B \oplus [\delta_b]_1^B$. Then for a $[]^B$-share $[\beta]^B$, we say that a $\langle\rangle$-share $\langle b\rangle$ is *consistent with* $[\beta]^B$ if $[\delta_b]_0^B = [\beta]_0^B$ and $[\delta_b]_1^B = [\beta]_1^B$.

2.3 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 is denoted by $\binom{k}{1}$-OT_ℓ^n. We sometimes omit the superscript 'n' 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 [17] based on a vector oblivious linear evaluation (VOLE) correlation, which is generated according to [3] where the internal small OTs are executed with the method of [18]. 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 B for details.

2.4 (Modified) Beaver Triples and AND Operations (for $[]^B$-Shares)

We use (plain and modified) AND operations of four types for shared bits: I-AND, II-AND, III-AND, and our proposed IV-AND, where the associated (modified) Beaver triples (BTs) are denoted by I-BT, II-BT, III-BT, and IV-BT, respectively. The ideal functionality to generate a T-BT uniformly at random is denoted by $\mathcal{F}_{\text{T-BT}}$. The functionalities and the costs of those (modified) AND operations are given in Table 4. We may write a I-AND operation and a II-AND operation simply by using the binary operation symbol $\wedge$.

A I-AND, i.e., the plain AND for shared bits, can be realized by using the following ordinary BT [1]: $[a]^B$, $[b]^B$, and $[c]^B$, where $c = a \wedge b$. A II-AND can be realized by using "arithmetic blinding" [20] with a modified BT of the form $[a]^B = (a, 0)$, $[b]^B = (0, b)$, and $[c]^B$, where $c = a \wedge b$. A III-AND can be realized as described in [22, Section 3.1.1] with a modified BT of the form $[a]^B$, $[b]^B$, $[c]^B$, $[d]^B$, and $[e]^B$, where $d = a \wedge b$ and $e = a \wedge c$. These (modified) BTs can be generated by a standard technique using OT as specified in Table 4: see Appendix C for details.

As for IV-AND, we use the following modified BT, which can be generated by a standard technique using OT as specified in Table 4 (see Appendix C for details):

$$[a]^B\,,\ [b]^B\,,\ [c]^B\,,\ [d]^B\,,\ [e]^B\,,\ [f]^B\,,\ \text{where } d = a \wedge b,\ e = a \wedge c,\ f = b \wedge [b]_0^B.$$

Then IV-AND can be executed as follows:

1. P_0 computes $X_0' \leftarrow [X]_0^B \oplus [a]_0^B$, $Y_0' \leftarrow [Y]_0^B \oplus [b]_0^B$, and $Z_0' \leftarrow [Z]_0^B \oplus [c]_0^B$, and sends them to P_1. On the other hand, P_1 computes $X_1' \leftarrow [X]_1^B \oplus [a]_1^B$, $Y_1' \leftarrow [Y]_1^B \oplus [b]_1^B$, and $Z_1' \leftarrow [Z]_1^B \oplus [c]_1^B$, and sends them to P_0.

Table 4: Functionalities and costs of (modified) AND operations.

Operation	Input	Output	Offline cost	Online communication	
				Rounds	Bits
I-AND (plain AND)	$[X]^B$, $[Y]^B$	$[X \wedge Y]^B$	$\binom{4}{1}$-OT_1	1	4
II-AND	$[X]^B = (X, 0)$, $[Y]^B = (0, Y)$	$[X \wedge Y]^B$	$\binom{2}{1}$-OT_1	1	2
III-AND	$[X]^B$, $[Y]^B$, $[Z]^B$	$[X \wedge Y]^B$, $[X \wedge Z]^B$	$\binom{8}{1}$-OT_2	1	6
IV-AND (our proposed)	$[X]^B$, $[Y]^B$, $[Z]^B$	$[X \wedge Y]^B$, $[X \wedge Z]^B$, $[Y \wedge y_0]^B$ $(y_0 := [Y]_0^B)$	$\binom{8}{1}$-OT_3	1	6

2. The parties locally compute $X' \leftarrow X'_0 \oplus X'_1$, $Y' \leftarrow Y'_0 \oplus Y'_1$, and $Z' \leftarrow Z'_0 \oplus Z'_1$.
3. P_0 outputs the following, where $y_0 := [Y]_0^B$:

$$\begin{aligned}
[X \wedge Y]_0^B &\leftarrow ([X]_0^B \oplus X'_1) \wedge Y' \oplus X' \wedge [b]_0^B \oplus [d]_0^B \ , \\
[X \wedge Z]_0^B &\leftarrow ([X]_0^B \oplus X'_1) \wedge Z' \oplus X' \wedge [c]_0^B \oplus [e]_0^B \ , \\
[Y \wedge y_0]_0^B &\leftarrow ([Y]_0^B \oplus Y'_1) \wedge Y'_0 \oplus Y' \wedge [b]_0^B \oplus [f]_0^B \ ,
\end{aligned}$$

and P_1 outputs the following:

$$\begin{aligned}
[X \wedge Y]_1^B &\leftarrow [a]_1^B \wedge Y' \oplus X' \wedge [b]_1^B \oplus [d]_1^B \ , \\
[X \wedge Z]_1^B &\leftarrow [a]_1^B \wedge Z' \oplus X' \wedge [c]_1^B \oplus [e]_1^B \ , \\
[Y \wedge y_0]_1^B &\leftarrow [b]_1^B \wedge Y'_0 \oplus [f]_1^B \ .
\end{aligned}$$

The correctness can be verified by a straightforward calculation, and the security can be proved in the $\mathcal{F}_{\text{IV-BT}}$-hybrid model via a standard proof technique (see Appendix D).

2.5 (Correlated) Multi-fan-in AND (for $\langle\rangle$-Shares)

We use an N-fan-in AND operation for $\langle\rangle$-shares $\langle X_0\rangle, \langle X_1\rangle, \ldots, \langle X_{N-1}\rangle$ described in [21, Section 3.1.4], which outputs a $[]^B$-share of the bit $X_0 \wedge X_1 \wedge \cdots \wedge X_{N-1}$. Its auxiliary input is a so-called Beaver triple extension [20], N-BTE in short, which is a collection of $[]^B$-shares of the form $([a_S]^B)_{\emptyset \neq S \subseteq [N]}$ with the relations $a_S = \bigwedge_{i \in S} a_{\{i\}}$ for each non-empty subset $S \subseteq [N]$. Moreover, it should satisfy that for each $i = 0, 1, \ldots, N-1$, $\langle X_i \rangle$ is consistent with $[a_{\{i\}}]^B$ (see Section 2.2 for the terminology). Given such $\langle\rangle$-shares and an N-BTE, the protocol requires no online communication. On the other hand, an N-BTE can be generated by a standard technique using OT, specifically $\binom{2^N}{1}$-OT_{2^N-N-1}. See Appendix E for details. We note that if the main input is a collection of $[]^B$-shares, then one-round online communication is required to convert the $[]^B$-shares to $\langle\rangle$-shares consistent with a given BTE.

2.5.1 New Correlated 3-fan-in AND

In order to efficiently manipulate multiple 3-fan-in ANDs with partly common inputs, we introduce the following correlated 3-fan-in AND: the main input is $\langle\rangle$-shares $\langle X\rangle$, $\langle Y\rangle$, $\langle Z\rangle$, and $\langle W\rangle$, and the protocol outputs $[]^B$-shares $[X \wedge Y \wedge Z]^B$ and $[X \wedge Y \wedge W]^B$ simultaneously. The auxiliary input is two 3-BTE's $([a_S]^B)_{\emptyset \neq S \subseteq [3]}$ and $([b_S]^B)_{\emptyset \neq S \subseteq [3]}$ that are correlated in a way that $[a_{\{0\}}]^B = [b_{\{0\}}]^B$, $[a_{\{1\}}]^B = [b_{\{1\}}]^B$, and $[a_{\{0,1\}}]^B = [b_{\{0,1\}}]^B$; such correlated 3-BTE's can be generated by a standard technique using OT,

specifically $\binom{16}{1}$-OT_7 (see Appendix E.1 for details). It should also be satisfied that $\langle X\rangle$, $\langle Y\rangle$, $\langle Z\rangle$, and $\langle W\rangle$ are consistent with $[a_{\{0\}}]^B$, $[a_{\{1\}}]^B$, $[a_{\{2\}}]^B$, and $[b_{\{2\}}]^B$, respectively. The protocol is similar to the ordinary 3-fan-in AND where $([a_S]^B)_{\emptyset\neq S\subseteq[3]}$ is used for computing $[X\wedge Y\wedge Z]^B$ and $([b_S]^B)_{\emptyset\neq S\subseteq[3]}$ is used for computing $[X\wedge Y\wedge W]^B$. We note that if the computations of $[X\wedge Y\wedge Z]^B$ and $[X\wedge Y\wedge W]^B$ are performed separately, then a given $[]^B$-share $[X]^B$ should be converted to two individual $\langle\rangle$-shares $\langle X\rangle$ beforehand (and similarly for $\langle Y\rangle$), which results in larger online bit complexity.

3 Auxiliary One-Round Equality Testing Protocols

In this section, we give two auxiliary equality testing protocols for plaintexts with one online communication round, which will be used in our first and second comparison protocols, respectively. Although the protocols here for n_0-bit inputs require either an n_0-fan-in AND or $\Theta(2^{n_0})$-bit communication, it will not affect the polynomial complexity of the main protocols with n-bit inputs because these equality testing protocols are 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 $O(\log n)$.

3.1 Our First Protocol

Given P_0's input $x = (x_{n_0-1}\cdots x_1x_0)_2$, P_1's input $y = (y_{n_0-1}\cdots y_1y_0)_2$, and an n_0-BTE $([a_S]^B)_{\emptyset\neq S\subseteq[n_0]}$, the first protocol proceeds as follows:

1. For $i = 0, 1, \ldots, n_0 - 1$, P_0 sets $[\mathsf{eq}_i]_0^B \leftarrow x_i \oplus 1$; P_1 sets $[\mathsf{eq}_i]_1^B \leftarrow y_i$; and the parties compute $\langle\mathsf{eq}_i\rangle \leftarrow \mathsf{bit2T}([\mathsf{eq}_i]^B; [a_{\{i\}}]^B)$.

2. The parties locally compute an n_0-fan-in AND $[\mathsf{eq}]^B \leftarrow \bigwedge_{i=0}^{n_0-1}\langle\mathsf{eq}_i\rangle$ by using the BTE $([a_S]^B)_{\emptyset\neq S\subseteq[n_0]}$, and output $[\mathsf{eq}]^B$.

The correctness follows from the fact that we have $\mathsf{eq}_i = \mathbf{1}\{x_i = y_i\}$ for every i and we have $x = y$ if and only if $x_i = y_i$ for every i. The protocol requires 1 round for the first step (for $\mathsf{bit2T}$) and no communication for the second step (cf. Section 2.5), therefore 1 online round in total. The number of online communication bits is $2n_0$ bits, and the number of offline communication bits is $2^{n_0}(2^{n_0} - n_0 - 1) + n_0 = O((2^{n_0})^2)$ bits (except for the setup of a VOLE correlation) for executing $\binom{2^{n_0}}{1}$-$\mathsf{OT}_{2^{n_0}-n_0-1}$ in generating the BTE.

3.2 Our Second Protocol

Let the parties' inputs be $x \in \mathbb{Z}_N$ and $y \in \mathbb{Z}_N$ for some parameter N. Then the second 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 II-AND (see Section 2.4), and output $[\mathsf{eq}]^B$.

The protocol requires 1 online round (for II-AND in the second step) in total. For the correctness, we first 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. The number of online communication bits is $2N = O(2^{n_0})$ bits for computing N II-ANDs in the second step, 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 II-BT's.

4 Our First Proposed Comparison Protocol $\Pi_{\mathsf{cmp},1}$

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 $y = (y_{n-1} \cdots y_1 y_0)_2$. Then 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 $[\mathsf{lt}_i]^B \leftarrow [\neg x_i]^B \wedge [y_i]^B$ (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 $\mathsf{lt} = \mathbf{1}\{x < y\}$.

4.2 Our Proposed Two-Round Protocol

When instantiating the protocol above with the equality testing protocol in Section 3.1, the second round becomes an n_0-fan-in AND with $n_0 := \lceil \log_2 n \rceil$ to compute each $[Z_i]^B$. Now the AND operation $[Z_{i+1}]^B \wedge [\mathsf{lt}_i]^B$ in the third round can be absorbed into the former n_0-fan-in AND, reducing one communication round. This results in a two-round comparison protocol $\Pi_{\mathsf{cmp},1}$ as in Protocol 1.

As for the online communication in Protocol 1, Line 15 where $(n-1)n_0$ $\mathsf{bit2T}$s (each requiring 2-bit communication) are executed in total is dominant. Hence the online bit complexity of Protocol 1 is $O(nn_0) = O(n \log n)$ bits. On the other hand, as for the offline communication, generating $n-1$ (n_0+1)-BTEs (each requiring approximately $(2^{n_0+1})^2$-bit communication) is dominant. Hence the offline bit complexity of Protocol 1 is $O(n(2^{n_0+1})^2) = O(n^3)$ bits. A more precise evaluation as in Table 1 is explained in Appendix F.

5 Our Second Proposed Comparison Protocol $\Pi_{\mathsf{cmp},2}$

Our second proposed protocol is also based on the construction in Section 4.1, but we use the internal equality testing protocol in Section 3.2. 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 $[Z_{i+1}]^B \leftarrow \bigoplus_{j=0}^{n-1} ([\mathsf{eq}_{i+1,0,j}]^B \wedge [\mathsf{eq}_{i+1,1,j}]^B)$.

Protocol 1 Our First Two-Round Plaintext Comparison Protocol $\Pi_{\mathsf{cmp},1}$

Input: P_0 has input $x = (x_{n-1} \cdots x_1 x_0)_2$, and P_1 has input $y = (y_{n-1} \cdots y_1 y_0)_2$
(Parameter: $n_0 := \lceil \log_2 n \rceil$)
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: Locally set $[\mathsf{neq}_i]_0^B \leftarrow x_i$ and $[\mathsf{neq}_i]_1^B \leftarrow y_i$
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: **for** $i = n-1, \ldots, 1, 0$ **do**
6: Locally set $[\neg x_i]_0^B \leftarrow x_i \oplus 1$, $[\neg x_i]_1^B \leftarrow 0$, $[y_i]_0^B \leftarrow 0$, and $[y_i]_1^B \leftarrow y_i$; and compute $[\mathsf{lt}_i]^B \leftarrow [\neg x_i]^B \wedge [y_i]^B$ by II-AND (see Section 2.4)
7: **end for**
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: P_0 parses $[\mathsf{neq}_{[n-1,i]}]_0^A$ as $(\nu_{i,0,n_0-1} \cdots \nu_{i,0,1}\nu_{i,0,0})_2$, and P_1 parses $(n - [\mathsf{neq}_{[n-1,i]}]_1^A) \bmod n$ as $(\nu_{i,1,n_0-1} \cdots \nu_{i,1,1}\nu_{i,1,0})_2$
11: **end for**
12: **for** $i = n-2, \ldots, 1, 0$ **do**
13: Prepare (in the offline phase) an (n_0+1)-BTE $([a_{i,S}]^B)_{\emptyset \neq S \subseteq [n_0+1]}$ (see Section 2.5)
14: **for** $j = n_0 - 1, \ldots, 1, 0$ **do**
15: Locally set $[\mathsf{eq}_{i+1,j}]_0^B \leftarrow \nu_{i+1,0,j} \oplus 1$ and $[\mathsf{eq}_{i+1,j}]_1^B \leftarrow \nu_{i+1,1,j}$; and compute $\langle \mathsf{eq}_{i+1,j} \rangle \leftarrow \mathsf{bit2T}([\mathsf{eq}_{i+1,j}]^B; [a_{i,\{j\}}]^B)$
16: **end for**
17: Compute $\langle \mathsf{lt}_i \rangle \leftarrow \mathsf{bit2T}([\mathsf{lt}_i]^B; [a_{i,\{n_0\}}]^B)$
18: **end for**
19: By using (n_0+1)-fan-in AND with the BTE, locally compute $[\mathsf{lt}]^B \leftarrow [\mathsf{lt}_{n-1}]^B \oplus \bigoplus_{i=0}^{n-2} \Big(\bigwedge_{j=0}^{n_0-1} \langle \mathsf{eq}_{i+1,j} \rangle \wedge \langle \mathsf{lt}_i \rangle \Big)$
20: **output** $[\mathsf{lt}]^B$

Now 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 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) II-AND:

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

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 second two-round protocol $\Pi_{\mathsf{cmp},2}$ as in Protocol 2.

As for the online communication in Protocol 2, Line 14 where $(n-1)n$ II-ANDs (each requiring 2-bit communication) are executed in total is dominant. Hence the online bit complexity of Protocol 2 is $O(n^2)$ bits. On the other hand, as for the offline communication, generating $(n-1)n$ II-BTs for the II-ANDs in Line 14 (each requiring approximately 3-bit communication) is dominant. Hence the offline bit complexity of Protocol 2 is $O(n^2)$ bits. A more precise evaluation as in Table 1 is explained in Appendix G.

Protocol 2 Our Second Two-Round Plaintext Comparison Protocol $\Pi_{\mathsf{cmp},2}$

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 $\mathbf{1}\{x < y\}$

for $i = n-1, \ldots, 2, 1$ **do**
  Locally set $[\mathsf{neq}_i]_0^B \leftarrow x_i$ and $[\mathsf{neq}_i]_1^B \leftarrow y_i$
  Compute $[\mathsf{neq}_i]^A \leftarrow \mathsf{bit2A}([\mathsf{neq}_i]^B)$ where the plaintext space for $[\mathsf{neq}_i]^A$ is set to $\mathbb{Z}_n$
end for
for $i = n-1, \ldots, 2, 1$ **do**
  Locally compute $[\mathsf{neq}_{[n-1,i]}]^A \leftarrow \sum_{j=i}^{n-1} [\mathsf{neq}_j]^A$
end for
for $i = n-2, \ldots, 1, 0$ **do**
  for $j = 0, 1, \ldots, n-1$ **do**
    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 $[\varepsilon_{i,0,j}]_1^B \leftarrow 0$
    P_0 sets $[\varepsilon_{i,1,j}]_0^B \leftarrow 0$, and P_1 sets $[\varepsilon_{i,1,j}]_1^B \leftarrow \mathbf{1}\{(n - [\mathsf{neq}_{[n-1,i+1]}]_1^A) \bmod n = j\} \wedge y_i$
  end for
end for
By using II-AND, compute $[\mathsf{lt}]^B \leftarrow [\mathsf{lt}_{n-1}]^B \oplus \bigoplus_{i=0}^{n-2} \bigoplus_{j=0}^{n-1} [\varepsilon_{i,0,j}]^B \wedge [\varepsilon_{i,1,j}]^B$
output $[\mathsf{lt}]^B$

6 Our Third Proposed Comparison Protocol $\Pi_{\mathsf{cmp},3}$

Our third proposed protocol is based on a traditional tree-based construction, where the inputs x and y are parsed in quaternary (instead of binary) expressions as $x = (x_{\lceil n/2 \rceil - 1} \cdots x_1 x_0)_4$ and $y = (y_{\lceil n/2 \rceil - 1} \cdots y_1 y_0)_4$, and the partial results $[\mathsf{lt}_i]^B = [\mathbf{1}\{x_i < y_i\}]^B$ and $[\mathsf{eq}_i]^B = [\mathbf{1}\{x_i = y_i\}]^B$ are combined by using a ternary (instead of binary) tree. To compute $[\mathsf{lt}_i]^B$ and $[\mathsf{eq}_i]^B$ more efficiently than the previous OT-based protocols [8, 12, 22], we develop the following custom-tailored Boolean circuits by searching for decompositions of truth tables for comparison/equality of quaternary integers involving as many common (reusable) factors as possible. The correctness can be checked for all cases of x_i and y_i as in Fig. 1.

Theorem 1. *Let $x_i, y_i \in \{0, 1, 2, 3\}$, and put*

$$
\begin{aligned}
[X]^B &= ([X]_0^B, [X]_1^B) := (\mathbf{1}\{x_i \in \{0, 2\}\}, \mathbf{1}\{y_i \in \{0, 2\}\}) \ , \\
[Y]^B &= ([Y]_0^B, [Y]_1^B) := (\mathbf{1}\{x_i \in \{0, 1\}\}, \mathbf{1}\{y_i \in \{0, 1\}\}) \ , \\
[Z]^B &= ([Z]_0^B, [Z]_1^B) := (\mathbf{1}\{x_i = 2\}, \mathbf{1}\{y_i = 1\}) \ .
\end{aligned}
$$

Then, by putting

$$
\mathsf{tmp}_1 := X \wedge Y \ , \ \mathsf{tmp}_2 := X \wedge Z \ , \ \mathsf{tmp}_3 := Y \wedge [Y]_0^B \ ,
$$

we have

$$
\begin{aligned}
\mathbf{1}\{x_i = y_i\} &= \mathsf{tmp}_1 \oplus \mathbf{1}\{x_i \in \{0, 3\}\} \oplus \mathbf{1}\{y_i \in \{1, 2\}\} \ , \\
\mathbf{1}\{x_i < y_i\} &= \mathsf{tmp}_2 \oplus \mathsf{tmp}_3 \ .
\end{aligned}
$$

Hence

$$
\begin{aligned}
[\mathsf{eq}[v_i]]^B &:= [\mathsf{tmp}_1]^B \oplus (\mathbf{1}\{x_i \in \{0, 3\}\}, \mathbf{1}\{y_i \in \{1, 2\}\}) \ , \\
[\mathsf{lt}[v_i]]^B &:= [\mathsf{tmp}_2]^B \oplus [\mathsf{tmp}_3]^B
\end{aligned}
$$

are correct $[]^B$-shares of bits $\mathbf{1}\{x_i = y_i\}$ and $\mathbf{1}\{x_i < y_i\}$, respectively.

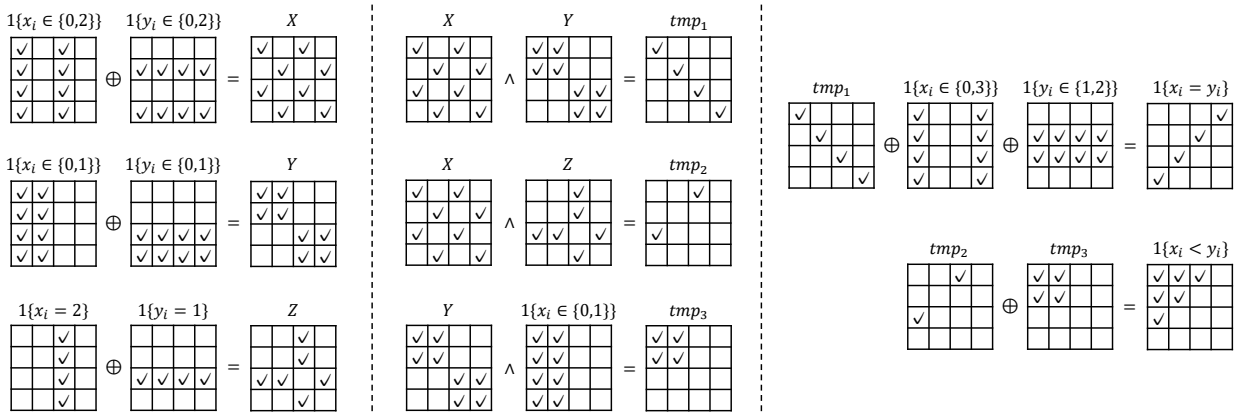

Figure 1: A visualized proof of Theorem 1: for the tables, the cell at the a-th column from the left ($0 \le a \le 3$) and the b-th row from the bottom ($0 \le b \le 3$) represents the bit value for the case $(x_i, y_i) = (a, b)$, and '✓' means 1 and the empty cell means 0.

Protocol 3 Our Third $\sim 6n$-bit Plaintext Comparison Protocol $\Pi_{\mathsf{cmp},3}$

Input: P_0 has input $x = (x_{q-1} \cdots x_1 x_0)_4$, and P_1 has input $y = (y_{q-1} \cdots y_1 y_0)_4$, where $q := \lceil n/2 \rceil$. Moreover, the parties have a ternary tree T with q leaves $v_{q-1}, \ldots, v_1, v_0$ (from left to right) and $\lceil \log_3 q \rceil + 1$ levels as described in the text.

Output: $[]^B$-share $[\mathbf{1}\{x < y\}]^B$ of the bit $\mathbf{1}\{x < y\}$

1: **for** $i = q-1, \ldots, 1, 0$ **do**
2: Compute $[\mathsf{lt}[v_i]]^B$ and $[\mathsf{eq}[v_i]]^B$ (or only $[\mathsf{lt}[v_i]]^B$, when $i = 0$) by subprotocol $\Pi_{\mathsf{sub},1}$ with inputs i, x_i, y_i, n, and q
3: **end for**
4: **for** $j = \lceil \log_3 q \rceil - 1, \ldots, 1, 0$, and each internal node v of T at the j-th level **do**
5: Compute $[\mathsf{lt}[v]]^B$ and $[\mathsf{eq}[v]]^B$ (or only $[\mathsf{lt}[v]]^B$, when v is the rightmost node at the same level) by subprotocol $\Pi_{\mathsf{sub},2}$ with inputs T, v, and $[\mathsf{lt}[u]]^B$ and $[\mathsf{eq}[u]]^B$ (if it exists) for each child node u of v
6: **end for**
7: **output** $[\mathsf{lt}[v_*]]^B$, where v_* is the root of T

Now we obtain our third proposed protocol as in Protocol 3, where the part of digit-wise comparison/equality using Theorem 1 is given as subprotocol $\Pi_{\mathsf{sub},1}$ (Protocol 4) and the internal computation (involving comparison by lexicographic ordering) at the part of the tree-shaped aggregation is given as subprotocol $\Pi_{\mathsf{sub},2}$ (Protocol 5). Here, let T be a ternary tree with $\lceil \log_3 q \rceil + 1$ levels and leaves $v_{q-1}, \ldots, v_1, v_0$ from left to right, where $q := \lceil n/2 \rceil$. We suppose that the lowest (i.e., $\lceil \log_3 q \rceil$-th) level consists of leaves v_h with smaller indices h, and each node at the 0-th to $(\lceil \log_3 q \rceil - 2)$-th levels has three child nodes, while at the $(\lceil \log_3 q \rceil - 1)$-th level, there is at most one node that has precisely two child nodes. See Fig. 2 for an example.

Correctness. The correctness of Protocol 3 is verified once we notice that $\Pi_{\mathsf{sub},1}$ correctly computes $\mathsf{lt}[v_i] = \mathbf{1}\{x_i < y_i\}$ and $\mathsf{eq}[v_i] = \mathbf{1}\{x_i = y_i\}$ (owing to Theorem 1) and $\Pi_{\mathsf{sub},2}$ correctly implements the comparison by the lexicographic ordering and the equality testing by looking at all digits. See Appendix H for details.

Round/Bit Complexity. As for the round complexity, the subprotocol $\Pi_{\mathsf{sub},1}$ requires 1 online round in any case, and the subprotocol $\Pi_{\mathsf{sub},2}$ (called $\lceil \log_3 q \rceil$ times in total) requires 1 online round as well (for $\mathsf{bit2Ts}$ and AND operations in parallel). Hence Protocol 3 requires $\lceil \log_3 q \rceil + 1$ online rounds in total.

Protocol 4 Subprotocol $\Pi_{\mathsf{sub},1}$ for Protocol 3

Input: Index i, integers $x_i, y_i \in \{0,1,2,3\}$, and parameters n and q
Output: $[]^B$-shares $[\mathsf{lt}[v_i]]^B$ and $[\mathsf{eq}[v_i]]^B$ (or only $[\mathsf{lt}[v_i]]^B$, when $i = 0$)

1: **if** n is odd and $i = q - 1$ **then**
2: Locally set $[\neg x_i]_0^B \leftarrow x_i \oplus 1$, $[\neg x_i]_1^B \leftarrow 0$, $[y_i]_0^B \leftarrow 0$, and $[y_i]_1^B \leftarrow y_i$; and compute $[\mathsf{lt}[v_i]]^B \leftarrow [\neg x_i]^B \wedge [y_i]^B$ by II-AND
3: Locally set $[\mathsf{eq}[v_i]]_0^B \leftarrow x_i \oplus 1$ and $[\mathsf{eq}[v_i]]_1^B \leftarrow y_i$
4: **output** $[\mathsf{lt}[v_i]]^B$ and $[\mathsf{eq}[v_i]]^B$
5: **else**
6: Locally set

$$[X]_0^B \leftarrow \mathbf{1}\{x_i \in \{0,2\}\}\,,\ [X]_1^B \leftarrow \mathbf{1}\{y_i \in \{0,2\}\}\ ,$$
$$y_0 := [Y]_0^B \leftarrow \mathbf{1}\{x_i \in \{0,1\}\}\,,\ y_1 := [Y]_1^B \leftarrow \mathbf{1}\{y_i \in \{0,1\}\}\ ,$$
$$[Z]_0^B \leftarrow \mathbf{1}\{x_i = 2\}\,,\ [Z]_1^B \leftarrow \mathbf{1}\{y_i = 1\}\ .$$

7: **if** $i > 0$ **then**
8: Compute $[\mathsf{tmp}_1]^B = [X \wedge Y]^B$, $[\mathsf{tmp}_2]^B = [X \wedge Z]^B$, and $[\mathsf{tmp}_3]^B = [Y \wedge y_0]^B$ by IV-AND (see Section 2.4)
9: P_0 computes $[\mathsf{eq}[v_i]]_0^B \leftarrow [\mathsf{tmp}_1]_0^B \oplus \mathbf{1}\{x_i \in \{0,3\}\}$; P_1 computes $[\mathsf{eq}[v_i]]_1^B \leftarrow [\mathsf{tmp}_1]_1^B \oplus \mathbf{1}\{y_i \in \{1,2\}\}$; and the parties locally compute $[\mathsf{lt}[v_i]]^B \leftarrow [\mathsf{tmp}_2]^B \oplus [\mathsf{tmp}_3]^B$
10: **output** $[\mathsf{lt}[v_i]]^B$ and $[\mathsf{eq}[v_i]]^B$
11: **else**
12: Compute $[\mathsf{tmp}_1]^B = [X]^B \wedge [Z]^B$ by I-AND
13: Locally set $[y_0]_0^B \leftarrow y_0$, $[y_0]_1^B \leftarrow 0$, $[y_1]_0^B \leftarrow 0$, and $[y_1]_1^B \leftarrow y_1$; and compute $[\mathsf{tmp}_2]^B = [y_0]^B \wedge [y_1]^B$ by II-AND
14: Locally set $[\mathsf{lt}[v_i]]^B \leftarrow [\mathsf{tmp}_1]^B \oplus [\mathsf{tmp}_2]^B \oplus [y_0]^B$
15: **output** $[\mathsf{lt}[v_i]]^B$
16: **end if**
17: **end if**

As for the online bit complexity, the subprotocol $\Pi_{\mathsf{sub},1}$ for each $i = 0, 1, \ldots, q-1$ requires at most 6-bit communication, while the subprotocol $\Pi_{\mathsf{sub},2}$ for each of the $\lceil (q-1)/2 \rceil \sim q/2$ internal nodes of the tree T requires at most $4 \cdot 2 + 4 = 12$-bit communication (for four bit2Ts and one I-AND). Hence the total online bit complexity is asymptotically $6 \cdot q + 12 \cdot q/2 = 12q \sim 6n$ bits. On the other hand, as for the offline bit complexity, both of the cost for the subprotocol $\Pi_{\mathsf{sub},1}$ for each $i = 0, 1, \ldots, q-1$ and the cost for the subprotocol $\Pi_{\mathsf{sub},2}$ for each of the $\lceil (q-1)/2 \rceil \sim q/2$ internal nodes of the tree T are constant (independent of n), therefore the total offline bit complexity is $O(q) = O(n)$ bits. A more precise evaluation as in Table 1 is explained in Appendix I.

7 Conclusion

In this work, we presented three novel secure comparison protocols over plaintexts, achieving either a constant two-round or online bandwidth of asymptotically $6n$ bits. While our contributions significantly advance the state-of-the-art, two challenging directions remain for future research: (1) further reducing the round complexity, and (2) further reducing the online communication complexity. Both tasks appear non-trivial. Regarding the first challenge, our two-round constructions (Protocols 1 and 2) fundamentally rely 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. Regarding the second challenge, while exploring different radix representations (e.g., $N > 4$) or alternative tree structures remains

Protocol 5 Subprotocol $\Pi_{\mathsf{sub},2}$ for Protocol 3

Input: The tree T, a node v, and $[]^B$-shares $[\mathsf{lt}[u]]^B$ and $[\mathsf{eq}[u]]^B$ (if it exists) for each child node u of v
Output: $[]^B$-shares $[\mathsf{lt}[v]]^B$ and $[\mathsf{eq}[v]]^B$ (or only $[\mathsf{lt}[v]]^B$, when v is the rightmost node at the same level)

if v is not the rightmost node at the same level **then**
  if v has three child nodes, say u_2, u_1, and u_0 from left to right **then**
    Prepare (in the offline phase) a correlated 3-BTE $([a_{v,S}]^B, [b_{v,S}]^B)_{\emptyset \neq S \subseteq [3]}$ (see Section 2.5.1)
    Compute $\langle \mathsf{eq}[u_2] \rangle \leftarrow \mathsf{bit2T}([\mathsf{eq}[u_2]]^B; [a_{v,\{0\}}]^B)$, $\langle \mathsf{eq}[u_1] \rangle \leftarrow \mathsf{bit2T}([\mathsf{eq}[u_1]]^B; [a_{v,\{1\}}]^B)$, $\langle \mathsf{lt}[u_0] \rangle \leftarrow \mathsf{bit2T}([\mathsf{lt}[v_0]]^B; [a_{v,\{2\}}]^B)$, and $\langle \mathsf{eq}[u_0] \rangle \leftarrow \mathsf{bit2T}([\mathsf{eq}[v_0]]^B; [b_{v,\{2\}}]^B)$
    By using I-AND and correlated 3-fan-in-AND (see Section 2.5.1), compute

$$[\mathsf{lt}[v]]^B \leftarrow [\mathsf{lt}[u_2]]^B \oplus [\mathsf{eq}[u_2]]^B \wedge [\mathsf{lt}[u_1]]^B \oplus \langle \mathsf{eq}[u_2] \rangle \wedge \langle \mathsf{eq}[u_1] \rangle \wedge \langle \mathsf{lt}[u_0] \rangle \ ,$$

$$[\mathsf{eq}[v]]^B \leftarrow \langle \mathsf{eq}[u_2] \rangle \wedge \langle \mathsf{eq}[u_1] \rangle \wedge \langle \mathsf{eq}[u_0] \rangle$$

  else
    Let the two child nodes of v be u_1 and u_0 from left to right
    By using III-AND, compute $[\mathsf{lt}[v]]^B \leftarrow [\mathsf{lt}[u_1]]^B \oplus [\mathsf{eq}[u_1]]^B \wedge [\mathsf{lt}[u_0]]^B$ and $[\mathsf{eq}[v]]^B \leftarrow [\mathsf{eq}[u_1]]^B \wedge [\mathsf{eq}[u_0]]^B$
  end if
  output $[\mathsf{lt}[v]]^B$ and $[\mathsf{eq}[v]]^B$
else
  if v has three child nodes, say u_2, u_1, and u_0 from left to right **then**
    Prepare (in the offline phase) a 3-BTE $([a_{v,S}]^B)_{\emptyset \neq S \subseteq [3]}$
    Compute $\langle \mathsf{eq}[u_2] \rangle \leftarrow \mathsf{bit2T}([\mathsf{eq}[u_2]]^B; [a_{v,\{0\}}]^B)$, $\langle \mathsf{eq}[u_1] \rangle \leftarrow \mathsf{bit2T}([\mathsf{eq}[u_1]]^B; [a_{v,\{1\}}]^B)$, and $\langle \mathsf{lt}[u_0] \rangle \leftarrow \mathsf{bit2T}([\mathsf{lt}[v_0]]^B; [a_{v,\{2\}}]^B)$
    By using I-AND and 3-fan-in AND, compute

$$[\mathsf{lt}[v]]^B \leftarrow [\mathsf{lt}[u_2]]^B \oplus [\mathsf{eq}[u_2]]^B \wedge [\mathsf{lt}[u_1]]^B \oplus \langle \mathsf{eq}[u_2] \rangle \wedge \langle \mathsf{eq}[u_1] \rangle \wedge \langle \mathsf{lt}[u_0] \rangle$$

  else
    Let the two child nodes of v be u_1 and u_0 from left to right
    By using I-AND, compute $[\mathsf{lt}[v]]^B \leftarrow [\mathsf{lt}[u_1]]^B \oplus [\mathsf{eq}[u_1]]^B \wedge [\mathsf{lt}[u_0]]^B$
  end if
  output $[\mathsf{lt}[v]]^B$
end if

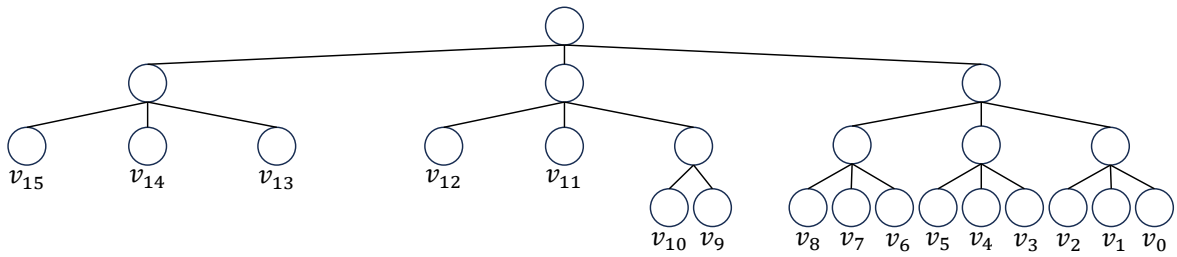

Figure 2: Example of the ternary tree T in Protocol 3, where $q = 16$.

a possibility, our $6n$-bit protocol (Protocol 3) already incorporates significant optimizations via custom-tailored circuit formulas and specialized multiplication gates. Substantial further reductions would likely require the development of entirely new encoding techniques or cryptographic primitives to overcome the current efficiency plateau.

References

[1] 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.

[2] 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.

[3] 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.

[4] 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.

[5] Geoffroy Couteau. New protocols for secure equality test and comparison. In Bart Preneel and Frederik Vercauteren, editors, *ACNS 2018: 16th International Conference on Applied Cryptography and Network Security*, volume 10892 of *Lecture Notes in Computer Science*, pages 303–320, Leuven, Belgium, July 2–4, 2018. Springer, Cham, Switzerland.

[6] Daniel Demmler, Thomas Schneider, and Michael Zohner. ABY - A framework for efficient mixed-protocol secure two-party computation. In *ISOC Network and Distributed System Security Symposium – NDSS 2015*, San Diego, CA, USA, February 8–11, 2015. The Internet Society.

[7] 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.

[8] Jun Feng, Yefan Wu, Hong Sun, Shunli Zhang, and Debin Liu. Panther: Practical secure two-party neural network inference. *IEEE Trans. Inf. Forensics Secur.*, 20:1149–1162, 2025.

[9] 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.

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

[11] 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.

[12] Zhicong Huang, Wen-jie Lu, Cheng Hong, and Jiansheng Ding. Cheetah: Lean and fast secure two-party deep neural network inference. In Kevin R. B. Butler and Kurt Thomas, editors, *31st USENIX Security Symposium, USENIX Security 2022, Boston, MA, USA, August 10-12, 2022*, pages 809–826. USENIX Association, 2022.

[13] Yuval Ishai, Eyal Kushilevitz, Sigurd Meldgaard, Claudio Orlandi, and Anat Paskin-Cherniavsky. On the power of correlated randomness in secure computation. In Amit Sahai, editor, *TCC 2013: 10th Theory of Cryptography Conference*, volume 7785 of *Lecture Notes in Computer Science*, pages 600–620, Tokyo, Japan, March 3–6, 2013. Springer Berlin Heidelberg, Germany.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] Deevashwer Rathee, Mayank Rathee, Nishant Kumar, Nishanth Chandran, Divya Gupta, Aseem Rastogi, and Rahul Sharma. CrypTFlow2: Practical 2-party secure inference. In Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna, editors, *CCS '20: 2020 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, USA, November 9-13, 2020*, pages 325–342. ACM, 2020.

[23] 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.

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

[25] 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.

[26] 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 Share Conversion Protocols

Here we describe the protocols for converting three types of shares given in [21].

$\mathsf{bit2T}$ Given a $[]^B$-share $[b]^B$ and an auxiliary $[]^B$-share $[\beta]^B$, we can consider the following protocol of converting $[b]^B$ to a $\langle\rangle$-share $\langle b\rangle$ that is consistent with $[\beta]^B$ (in the sense of Section 2.2):

1. For $i = 0, 1$, P_i sends $\alpha_i := [b]_i^B \oplus [\beta]_i^B$ to P_{1-i}.
2. For $i = 0, 1$, P_i locally sets $[\delta_b]_i^B \leftarrow [\beta]_i^B$ and $\Delta_b \leftarrow \alpha_0 \oplus \alpha_1$.

We write the protocol as $\mathsf{bit2T}([b]^B; [\beta]^B)$, or simply $\mathsf{bit2T}([b]^B)$ in which case it is implicitly assumed that some fresh $[]^B$-share $[\beta]^B$ is used as an auxiliary input.

$\mathsf{T2A}$ Given a $\langle\rangle$-share $\langle b\rangle = ([\delta_b]_0^B, [\delta_b]_1^B; \Delta_b)$ for a bit b and an auxiliary $[]^A$-share $[\delta_b]^A$ for the value δ_b viewed as an integer, we can consider the following protocol of converting $\langle b\rangle$ to a $[]^A$-share $[b]^A$ for the same value b viewed as an integer:

1. P_0 locally sets $[b]_0^A \leftarrow [\delta_b]_0^A - 2\Delta_b \cdot [\delta_b]_0^A$, and P_1 locally sets $[b]_1^A \leftarrow [\delta_b]_1^A + \Delta_b - 2\Delta_b \cdot [\delta_b]_1^A$.

We write the protocol as $\mathsf{T2A}(\langle b\rangle; [\delta_b]^A)$, or simply $\mathsf{T2A}(\langle b\rangle)$ in which case it is implicitly assumed that some fresh $[]^A$-share $[\delta_b]^A$ is used as an auxiliary input.

$\mathsf{bit2A}$ Given a $[]^B$-share $[b]^B$ for a bit b and auxiliary $[]^B$-share $[\beta]^B$ and $[]^A$-share $[\beta]^A$ for some common value $\beta \in \{0,1\}$, we can convert $[b]^B$ to a $[]^A$-share $[b]^A$ for the same value b viewed as an integer, by composing the protocols $\mathsf{bit2T}$ and $\mathsf{T2A}$. We write the protocol as $\mathsf{bit2A}([b]^B; [\beta]^B, [\beta]^A)$, or simply $\mathsf{bit2A}([b]^B)$. Precisely, the protocol is given as follows:

1. The parties compute $\langle b \rangle \leftarrow \mathsf{bit2T}([b]^B; [\beta]^B)$.
2. The parties locally compute $[b]^A \leftarrow \mathsf{T2A}(\langle b \rangle; [\beta]^A)$.

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 by using $\binom{2}{1}$-$\mathsf{OT}_{\lceil \log_2 N \rceil}$ (see Section 2.3 for the notation) as follows:

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, given P_1's output z of the OT, P_1 sets $[\beta]_1^A \leftarrow z$.

B 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 [17]):

1. Let $x_0, x_1, \ldots, x_{k-1} \in \{0,1\}^\ell$ be the inputs of the sender, and let $r \in [k]$ 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 [3], 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 [18], 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.

C Generating (Modified) BTs

Here we explain how to generate (modified) BTs of types I, II, III, and IV via OT.

I-BT A I-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 $([a]_1^B \mid\mid [b]_1^B)$-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 $[a]_1^B \mid\mid [b]_1^B$. Then P_1 writes the output of the OT as $[c]_1^B$.

II-BT Similarly to the case of I-BT mentioned above, a II-BT $([a]^B, [b]^B, [c]^B)$ where $[a]^B = (a, 0)$ and $[b]^B = (0, b)$ can be generated via $\binom{2}{1}$-OT_1 where the sender's b-th input is a value corresponding to $[c]_1^B$.

III-BT Similarly to the case of I-BT, a III-BT $([a]^B, [b]^B, [c]^B, [d]^B, [e]^B)$ can be generated via $\binom{8}{1}$-OT_2 where the sender's $([a]_1^B \mid\mid [b]_1^B \mid\mid [c]_1^B)$-th input is a value corresponding to $[d]_1^B \mid\mid [e]_1^B$.

IV-BT Similarly to the case of I-BT, a IV-BT can be generated via $\binom{8}{1}$-OT_3 where the sender's $([a]_1^B \mid\mid [b]_1^B \mid\mid [c]_1^B)$-th input is a value corresponding to $[d]_1^B \mid\mid [e]_1^B \mid\mid [f]_1^B$.

D Security Proof for IV-AND Protocol

Here we give a proof for the security of the protocol for IV-AND operation (in the $\mathcal{F}_{\text{IV-BT}}$-hybrid model) described in Section 2.4.

For the security against P_0, the view for P_0 consists of $[a]_0^B$, $[b]_0^B$, $[c]_0^B$, $[d]_0^B$, $[e]_0^B$, $[f]_0^B$, X_1', Y_1', and Z_1'. It suffices to show that, given any inputs $[X]^B$, $[Y]^B$, $[Z]^B$ as well as the values $\alpha_1, \ldots, \alpha_9$ for the nine components of the view (in the order above), there is precisely one choice of a IV-BT consistent with those values. First, among the components of IV-BT, the values of $[a]_0^B, \ldots, [f]_0^B$ are uniquely determined as $[a]_0^B = \alpha_1, \ldots, [f]_0^B = \alpha_6$. Secondly, the values of $[a]_1^B$, $[b]_1^B$, and $[c]_1^B$ are uniquely determined as $[a]_1^B = [X]_1^B \oplus \alpha_7$, $[b]_1^B = [Y]_1^B \oplus \alpha_8$, and $[c]_1^B = [Z]_1^B \oplus \alpha_9$. Finally, the values of $[d]_1^B$, $[e]_1^B$, and $[f]_1^B$ are uniquely determined as $[d]_1^B = ([a]_0^B \oplus [a]_1^B) \wedge ([b]_0^B \oplus [b]_1^B) \oplus [d]_0^B$, $[e]_1^B = ([a]_0^B \oplus [a]_1^B) \wedge ([c]_0^B \oplus [c]_1^B) \oplus [e]_0^B$, and $[f]_1^B = ([b]_0^B \oplus [b]_1^B) \wedge [b]_0^B \oplus [f]_0^B$. Hence it has been shown that any view for P_0 emerges with the same probability, therefore such a view can be simulated simply by a random choice.

The security against P_1 can be shown in a similar manner. This concludes the proof.

E (Correlated) Multi-fan-in AND and BTE

We suppose that N $\langle\rangle$-shares $\langle X_0\rangle, \langle X_1\rangle, \ldots, \langle X_{N-1}\rangle$ are given as the main input, an N-BTE $([a_S]^B)_{\emptyset \neq S \subseteq [N]}$ is given as an auxiliary input with the relations $a_S = \bigwedge_{i \in S} a_{\{i\}}$ for each non-empty subset $S \subseteq [N]$, and for each $i = 0, 1, \ldots, N-1$, $\langle X_i\rangle$ is consistent with $[a_{\{i\}}]^B$ (see Section 2.2 for the terminology). In this setting, by letting P_0 compute

$$[Z]_0^B \leftarrow \bigoplus_{\emptyset \neq S \subseteq [N]} \Big([a_S]_0^B \wedge \bigwedge_{i \in [N] \setminus S} \Delta_{X_i} \Big) ,$$

and letting P_1 compute

$$[Z]_1^B \leftarrow \Big(\bigwedge_{i \in [N]} \Delta_{X_i} \Big) \oplus \bigoplus_{\emptyset \neq S \subseteq [N]} \Big([a_S]_1^B \wedge \bigwedge_{i \in [N] \setminus S} \Delta_{X_i} \Big) ,$$

it can be verified by a straightforward calculation that $[Z]^B := ([Z]_0^B, [Z]_1^B)$ is a correct $[]^B$-share of the bit $X_0 \wedge X_1 \wedge \cdots \wedge X_{N-1}$. Given $\langle\rangle$-shares satisfying the condition above as the main input, the protocol above itself requires no online communication.

As for the offline cost, in a way similar to generating (modified) BTs as in Appendix C, an N-BTE $([a_S]^B)_{\emptyset \neq S \subseteq [N]}$ can be generated via $\binom{2^N}{1}$-OT_{2^N-N-1}, where the input of P_1 as a receiver is $[a_{\{0\}}]_1^B \,||\, [a_{\{1\}}]_1^B \,||\, \cdots \,||\, [a_{\{N-1\}}]_1^B$ and the $([a_{\{0\}}]_1^B \,||\, [a_{\{1\}}]_1^B \,||\, \cdots \,||\, [a_{\{N-1\}}]_1^B)$-th input of P_0 as a sender is a collection of the bits corresponding to $[a_S]_1^B$ over the $2^N - N - 1$ subsets $S \subseteq [N]$ with $|S| \geq 2$.

E.1 BTE for Correlated 3-fan-in AND

As for the offline cost of our new correlated 3-fan-in AND described in Section 2.5.1, the correlated 3-BTE used as an auxiliary input can be generated via $\binom{16}{1}$-OT_7, where the input of P_1 as a receiver is $[a_{\{0\}}]_1^B \,||\, [a_{\{1\}}]_1^B \,||\, [a_{\{2\}}]_1^B \,||\, [b_{\{2\}}]_1^B$ and the $([a_{\{0\}}]_1^B \,||\, [a_{\{1\}}]_1^B \,||\, [a_{\{2\}}]_1^B \,||\, [b_{\{2\}}]_1^B)$-th input of P_0 as a sender is a collection of the bits corresponding to $[a_{\{0,1\}}]^B$, $[a_{\{0,2\}}]^B$, $[a_{\{1,2\}}]^B$, $[a_{\{0,1,2\}}]^B$, $[b_{\{0,2\}}]^B$, $[b_{\{1,2\}}]^B$, and $[b_{\{0,1,2\}}]^B$.

F Bit Complexity for Protocol 1

Here we describe the bit complexity of Protocol 1.

F.1 Online Bit Complexity

In Protocol 1, online communication occurs only in Lines 3, 6, 15, and 17. As for the number of online communication bits, we have the following:

- For Line 3, each bit2A requires 2-bit communication, therefore the total number of communication bits is $2(n-1)$ bits.
- For Line 6, each II-AND requires 2-bit communication, therefore the total number of communication bits is $2n$ bits.
- For Line 15, each bit2T requires 2-bit communication, therefore the total number of communication bits is $2(n-1)n_0$ bits.
- For Line 17, each bit2T requires 2-bit communication, therefore the total number of communication bits is $2(n-1)$ bits.

Summarizing, the total number of online communication bits is

$$\begin{aligned} 2(n-1) + 2n + 2(n-1)n_0 + 2(n-1) &= 2(n-1)n_0 + 6n - 4 \\ &= 2(n-1)\lceil \log_2 n \rceil + 6n - 4 \\ &\sim 2n \log_2 n \text{ (bits)}. \end{aligned}$$

F.2 Offline Bit Complexity

As for the offline communication cost, we have the following:

- For Line 3, each bit2A requires $\binom{2}{1}$-$\mathsf{OT}_{n_0}^1$ in the offline phase. Therefore the total offline cost is $\binom{2}{1}$-$\mathsf{OT}_{n_0}^{n-1}$.
- For Line 6, each II-AND requires $\binom{2}{1}$-OT_1^1 in the offline phase. Therefore the total offline cost is $\binom{2}{1}$-OT_1^n.

- For each $i = n-2, \ldots, 1, 0$, to generate an (n_0+1)-BTE $([a_{i,S}]^B)_{\emptyset \neq S \subseteq [n_0+1]}$, it requires $\binom{2^{n_0+1}}{1}$-$\mathsf{OT}^1_{2^{n_0+1}-n_0-2}$ in the offline phase. Therefore the total offline cost is $\binom{2^{n_0+1}}{1}$-$\mathsf{OT}^{n-1}_{2^{n_0+1}-n_0-2}$.

Summarizing, the total offline communication cost is $\binom{2}{1}$-$\mathsf{OT}^{n-1}_{n_0} + \binom{2}{1}$-$\mathsf{OT}^n_1 + \binom{2^{n_0+1}}{1}$-$\mathsf{OT}^{n-1}_{2^{n_0+1}-n_0-2}$, which is (according to Section 2.3) equivalent to communicating the following number of bits:

$$\begin{aligned}
&(n-1)(2n_0+1) + n \cdot 3 + (n-1)(2^{n_0+1}(2^{n_0+1} - n_0 - 2) + n_0 + 1) \\
&= (n-1)(3n_0+2) + 3n + (n-1)2^{2n_0+2} - (n-1)2^{n_0+1}(n_0+2) \\
&\approx (n-1)(3\log_2 n + 2) + 3n + 4(n-1)n^2 - 2(n-1)n(\log_2 n + 2) \\
&= 4n^3 - 4n^2 + 3n - (n-1)(2n\log_2 n + 4n - 3\log_2 n - 2) \\
&< 4n^3 \text{ (bits)}
\end{aligned}$$

and setting up a VOLE correlation of dimension

$$d := (n-1) \cdot 1 + n \cdot 1 + (n-1)(n_0+1) = n(n_0+3) - n_0 - 2 \lessapprox n(\log_2 n + 3) \ .$$

Hence the total number of offline communication bits is

$$\approx 4n^3 + 2\mu + \lambda + (\mu+\lambda)\log_2 d \approx 4n^3 + 2\mu + \lambda + (\mu+\lambda)(\log_2 n + \log_2^2 n) \text{ (bits)}.$$

G Bit Complexity for Protocol 2

Here we describe the bit complexity of Protocol 2.

G.1 Online Bit Complexity

In Protocol 2, online communication occurs only in Lines 3 and 14. As for the number of online communication bits, we have the following:

- For Line 3, each $\mathsf{bit2A}$ requires 2-bit communication, therefore the total number of communication bits is $2(n-1)$ bits.
- For Line 14, each Π-AND requires 2-bit communication, therefore the total number of communication bits is $2(n-1)n$ bits.

Summarizing, the total number of online communication bits is

$$2(n-1) + 2(n-1)n = 2(n-1)(n+1) = 2n^2 - 2 \sim 2n^2 \text{ (bits)}.$$

G.2 Offline Bit Complexity

As for the offline communication cost, we have the following:

- For Line 3, each $\mathsf{bit2A}$ requires $\binom{2}{1}$-$\mathsf{OT}^1_{n_0}$ (where $n_0 := \lceil \log_2 n \rceil$) in the offline phase. Therefore the total offline cost is $\binom{2}{1}$-$\mathsf{OT}^{n-1}_{n_0}$.
- For Line 14, each Π-AND requires $\binom{2}{1}$-OT^1_1 in the offline phase. Therefore the total offline cost is $\binom{2}{1}$-$\mathsf{OT}^{(n-1)n}_1$.

Summarizing, the total offline communication cost is $\binom{2}{1}$-$\mathsf{OT}^{n-1}_{n_0} + \binom{2}{1}$-$\mathsf{OT}^{(n-1)n}_1$, which is (according to Section 2.3) equivalent to communicating the following number of bits:

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

and setting up a VOLE correlation of dimension $d := (n-1)\cdot 1 + (n-1)n \cdot 1 = n^2 - 1 < n^2$. Hence the total number of offline communication bits is

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

H Correctness of Protocol 3

Here we prove the correctness of Protocol 3. First we consider the subprotocol $\Pi_{\mathsf{sub},1}$ and show that $\mathsf{lt}[v_i] = \mathbf{1}\{x_i < y_i\}$ and $\mathsf{eq}[v_i] = \mathbf{1}\{x_i = y_i\}$. When n is odd and $i = q-1$, we actually have $x_i, y_i \in \{0,1\}$, therefore the result is correct. For the other case, if $i > 0$, then the result is correct by Theorem 1. On the other hand, if $i = 0$, then we have $\mathbf{1}\{x_i < y_i\} = X \wedge Z \oplus Y \wedge [Y]_0^B$ as in Theorem 1, and this is moreover equal to

$$X \wedge Z \oplus ([Y]_0^B \oplus [Y]_1^B) \wedge [Y]_0^B = X \wedge Z \oplus [Y]_0^B \oplus [Y]_0^B \wedge [Y]_1^B \ .$$

Hence the result is correct in any case.

Next, we show recursively (from the leaves to the root of the tree T) that for each node v of T, we have $\mathsf{lt}[v] = \mathbf{1}\{x[v] < y[v]\}$ and $\mathsf{eq}[v] = \mathbf{1}\{x[v] = y[v]\}$ where $x[v] := (x_{i_{\max}} \cdots x_{i_{\min}+1} x_{i_{\min}})_4$, $y[v] := (y_{i_{\max}} \cdots y_{i_{\min}+1} y_{i_{\min}})_4$ and $i_{\max}, \ldots, i_{\min}+1, i_{\min}$ are the indices of the leaves v_i that is covered by v (i.e., the path from v_i to the root involves v). The base case of leaves v for the recursion has been shown in the previous paragraph. In the subprotocol $\Pi_{\mathsf{sub},2}$, suppose that this property is satisfied by every child node u of v. It suffices to consider the case that v is not the rightmost node at the same level; the argument in the other case is a subset of the former. Now we consider the case that v has three child nodes u_2, u_1, and u_0 from left to right; the other case is similar (and easier). Note that $x[v] = x[u_2] \,||\, x[u_1] \,||\, x[u_0]$ and $y[v] = y[u_2] \,||\, y[u_1] \,||\, y[u_0]$. Then we divide the argument into the following four cases:

- If $x[u_2] \neq y[u_2]$, then we have $x[v] \neq y[v]$, $\mathbf{1}\{x[v] < y[v]\} = \mathbf{1}\{x[u_2] < y[u_2]\} = \mathsf{lt}[u_2]$ and $\mathsf{eq}[u_2] = 0$, therefore $\mathsf{lt}[v] = \mathsf{lt}[u_2] = \mathbf{1}\{x[v] < y[v]\}$ and $\mathsf{eq}[v] = 0 = \mathbf{1}\{x[v] = y[v]\}$.
- If $x[u_2] = y[u_2]$ and $x[u_1] \neq y[u_1]$, then we have $x[v] \neq y[v]$, $\mathbf{1}\{x[v] < y[v]\} = \mathbf{1}\{x[u_1] < y[u_1]\} = \mathsf{lt}[u_1]$, $\mathsf{lt}[u_2] = 0$, $\mathsf{eq}[u_2] = 1$ and $\mathsf{eq}[u_1] = 0$, therefore $\mathsf{lt}[v] = \mathsf{lt}[u_1] = \mathbf{1}\{x[v] < y[v]\}$ and $\mathsf{eq}[v] = 0 = \mathbf{1}\{x[v] = y[v]\}$.
- If $x[u_2] = y[u_2]$, $x[u_1] = y[u_1]$ and $x[u_0] \neq y[u_0]$, then we have $x[v] \neq y[v]$, $\mathbf{1}\{x[v] < y[v]\} = \mathbf{1}\{x[u_0] < y[u_0]\} = \mathsf{lt}[u_0]$, $\mathsf{lt}[u_2] = \mathsf{lt}[u_1] = 0$, $\mathsf{eq}[u_2] = \mathsf{eq}[u_1] = 1$ and $\mathsf{eq}[u_0] = 0$, therefore $\mathsf{lt}[v] = \mathsf{lt}[u_0] = \mathbf{1}\{x[v] < y[v]\}$ and $\mathsf{eq}[v] = 0 = \mathbf{1}\{x[v] = y[v]\}$.
- Otherwise, i.e., if $x[u_2] = y[u_2]$, $x[u_1] = y[u_1]$ and $x[u_0] = y[u_0]$, then we have $x[v] = y[v]$, $\mathsf{lt}[u_2] = \mathsf{lt}[u_1] = \mathsf{lt}[u_0] = 0$ and $\mathsf{eq}[u_2] = \mathsf{eq}[u_1] = \mathsf{eq}[u_0] = 1$, therefore $\mathsf{lt}[v] = 0 = \mathbf{1}\{x[v] < y[v]\}$ and $\mathsf{eq}[v] = 1 = \mathbf{1}\{x[v] = y[v]\}$.

Hence the desired property holds in any case. Therefore, for the root v_* of T, we finally have $\mathsf{lt}[v_*] = \mathbf{1}\{x[v_*] < y[v_*]\} = \mathbf{1}\{x < y\}$ and $\mathsf{eq}[v_*] = \mathbf{1}\{x[v_*] = y[v_*]\} = \mathbf{1}\{x = y\}$ as $x[v_*] = x$ and $y[v_*] = y$. This shows that Protocol 3 is correct.

I Bit Complexity for Protocol 3

Here we describe the bit complexity of Protocol 3.

I.1 Online Bit Complexity

In this and the next subsections, we assume for the sake of simplicity that n is even (the communication cost is slightly decreased when n is odd). As for the subprotocol $\Pi_{\mathsf{sub},1}$, it requires one IV-AND if $i > 0$ (6-bit online communication), and one I-AND and one II-AND if $i = 0$ ($4+2 = 6$-bit online communication). Hence the total online bit complexity of Line 2 in Protocol 3 is $6q$ bits.

As for Line 5 in Protocol 3, let N_1, N_2, N_3, and N_4 denote the numbers of internal nodes v of T satisfying the following: (for N_1) v is not the rightmost node at the same level and v has three child nodes; (for N_2) v is not the rightmost node at the same level and v has two child nodes; (for N_3) v is the rightmost node at the same level and v has three child nodes; (for N_4) v is the rightmost node at the same level and v has two child nodes. For each case, the number of online communication bits of the subprotocol $\Pi_{\mathsf{sub},2}$ is as follows:

- For the first case (N_1 nodes): It requires four bit2T's (2 bits each) and one I-AND (4 bits), therefore $4 \cdot 2 + 4 = 12$ bits in total.
- For the second case (N_2 nodes): It requires one III-AND, therefore 6 bits in total.
- For the third case (N_3 nodes): It requires three bit2T's (2 bits each) and one I-AND (4 bits), therefore $3 \cdot 2 + 4 = 10$ bits in total.
- For the fourth case (N_4 nodes): It requires one I-AND, therefore 4 bits in total.

By summing them up, the total number of online communication bits of Line 5 is $12N_1 + 6N_2 + 10N_3 + 4N_4$ bits. For the sake of simplicity, we bound the value from above as

$$\begin{aligned} 12N_1 + 6N_2 + 10N_3 + 4N_4 &\leq 12N_1 + 12N_2 + 10N_3 + 10N_4 \\ &= 12(N_1 + N_2 + N_3 + N_4) - 2(N_3 + N_4) \ . \end{aligned}$$

Now we note that $N_1 + N_2 + N_3 + N_4$ in the right-hand side is the total number of internal nodes of the ternary tree T. Based on the argument of counting "losers in a knockout tournament", there are $\lceil (q-1)/2 \rceil$ internal nodes of T. On the other hand, $N_3 + N_4$ in the right-hand side is the total number of the rightmost nodes at levels 0 to $\lceil \log_3 q \rceil - 1$, which is $\lceil \log_3 q \rceil$. Hence, the total online bit complexity of Line 5 is upper bounded by $12\lceil (q-1)/2 \rceil - 2\lceil \log_3 q \rceil$ bits.

Summarizing, the online bit complexity of Protocol 3 is upper bounded by

$$6q + 12\lceil (q-1)/2 \rceil - 2\lceil \log_3 q \rceil = 3n + 12\lceil (n-2)/4 \rceil - 2\lceil \log_3 (n/2) \rceil \sim 6n \text{ bits.}$$

I.2 Offline Bit Complexity

As for the subprotocol $\Pi_{\mathsf{sub},1}$, when $i > 0$, it uses one IV-BT (generated via $\binom{8}{1}$-OT_3), and when $i = 0$, it uses one I-BT (generated via $\binom{4}{1}$-OT_1) and one II-BT (generated via $\binom{2}{1}$-OT_1). Hence the total offline communication cost for Line 2 is $\binom{8}{1}$-$\mathsf{OT}_3^{q-1} + \binom{4}{1}$-$\mathsf{OT}_1^1 + \binom{2}{1}$-$\mathsf{OT}_1^1$.

As for the subprotocol $\Pi_{\mathsf{sub},2}$, for the four types of internal nodes of the tree T mentioned in Appendix I.1, the offline communication cost is as follows:

- For the first case (N_1 nodes): It uses one correlated 3-BTE (generated via $\binom{16}{1}$-OT_7) and one I-BT (generated via $\binom{4}{1}$-OT_1).
- For the second case (N_2 nodes): It uses one III-BT (generated via $\binom{8}{1}$-OT_2).
- For the third case (N_3 nodes): It uses one 3-BTE (generated via $\binom{8}{1}$-OT_4) and one I-BT (generated via $\binom{4}{1}$-OT_1).
- For the fourth case (N_4 nodes): It uses one I-BT (generated via $\binom{4}{1}$-OT_1).

By comparing them, the first case is more expensive than the second case, and the third case is more expensive than the fourth case. For the sake of simplicity, we bound the cost from above by including the second case to the first case and including the fourth case to the third case. Then the total offline communication cost of Line 5 is upper bounded by

$$\begin{aligned}&\binom{16}{1}\text{-}\mathsf{OT}_7^{N_1+N_2} + \binom{4}{1}\text{-}\mathsf{OT}_1^{N_1+N_2} + \binom{8}{1}\text{-}\mathsf{OT}_4^{N_3+N_4} + \binom{4}{1}\text{-}\mathsf{OT}_1^{N_3+N_4}\\ &= \binom{16}{1}\text{-}\mathsf{OT}_7^{N_1+N_2} + \binom{4}{1}\text{-}\mathsf{OT}_1^{N_1+N_2+N_3+N_4} + \binom{8}{1}\text{-}\mathsf{OT}_4^{N_3+N_4} \ .\end{aligned}$$

Moreover, we have $N_1 + N_2 + N_3 + N_4 = \lceil (q-1)/2 \rceil$ and $N_3 + N_4 = \lceil \log_3 q \rceil$ as shown in Appendix I.1, therefore the total offline communication cost of Line 5 is upper bounded by

$$\binom{16}{1}\text{-}\mathsf{OT}_7^{\lceil (q-1)/2 \rceil - \lceil \log_3 q \rceil} + \binom{4}{1}\text{-}\mathsf{OT}_1^{\lceil (q-1)/2 \rceil} + \binom{8}{1}\text{-}\mathsf{OT}_4^{\lceil \log_3 q \rceil} \ .$$

Summarizing, the total offline communication cost of Protocol 3 is upper bounded by

$$\begin{aligned}&\binom{8}{1}\text{-}\mathsf{OT}_3^{q-1} + \binom{4}{1}\text{-}\mathsf{OT}_1^{1} + \binom{2}{1}\text{-}\mathsf{OT}_1^{1}\\ &+ \binom{16}{1}\text{-}\mathsf{OT}_7^{\lceil (q-1)/2 \rceil - \lceil \log_3 q \rceil} + \binom{4}{1}\text{-}\mathsf{OT}_1^{\lceil (q-1)/2 \rceil} + \binom{8}{1}\text{-}\mathsf{OT}_4^{\lceil \log_3 q \rceil} \ .\end{aligned}$$

This is (according to Section 2.3) equivalent to communicating the following number of bits:

$$\begin{aligned}&(q-1)\cdot(8\cdot 3+3) + 1\cdot(4\cdot 1+2) + 1\cdot(2\cdot 1+1)\\ &+ (\lceil (q-1)/2 \rceil - \lceil \log_3 q \rceil)\cdot(16\cdot 7+4)\\ &+ \lceil (q-1)/2 \rceil \cdot (4\cdot 1+2) + \lceil \log_3 q \rceil \cdot (8\cdot 4+3)\\ &= 27(q-1) + 6 + 3 + 116(\lceil (q-1)/2 \rceil - \lceil \log_3 q \rceil) + 6\lceil (q-1)/2 \rceil + 35\lceil \log_3 q \rceil\\ &= 27q - 18 + 122\lceil (q-1)/2 \rceil - 81\lceil \log_3 q \rceil\\ &\lessapprox 27q + 122(q-1)/2\\ &< 88q \text{ (bits)}\end{aligned}$$

and setting up a VOLE correlation of dimension

$$\begin{aligned}d &:= (q-1)\cdot 3 + 1\cdot 2 + 1\cdot 1\\ &\quad + (\lceil (q-1)/2 \rceil - \lceil \log_3 q \rceil)\cdot 4 + \lceil (q-1)/2 \rceil \cdot 2 + \lceil \log_3 q \rceil \cdot 3\\ &= 3q + 6\lceil (q-1)/2 \rceil - \lceil \log_3 q \rceil\\ &\lessapprox 3q + 6(q-1)/2\\ &< 6q \ .\end{aligned}$$

Hence the total offline bit complexity of Protocol 3 is upper bounded by

$$\begin{aligned}\approx 88q + 2\mu + \lambda + (\mu+\lambda)\log_2 d &\lessapprox 88q + 2\mu + \lambda + (\mu+\lambda)\log_2(6q)\\ &= 44n + 2\mu + \lambda + (\mu+\lambda)\log_2(3n)\\ &= 44n + 2\mu + \lambda + (\mu+\lambda)(\log_2 n + \log_2 3)\\ &< 44n + 4\mu + 3\lambda + (\mu+\lambda)\log_2 n \text{ bits.}\end{aligned}$$