Electronic Preprint for Journal of Information Processing Vol.29

Regular Paper

Client-aided Robust Bit-composition Protocol with Deterministic Cheater Identification in Standard Model

Hikaru Tsuchida[1,2,a),b)] Takashi Nishide[1,c)]

Received: November 30, 2020, Accepted: June 7, 2021

Abstract: Secure multiparty computation (MPC) enables parties to compute an arbitrary function without revealing each party's inputs. A typical MPC is secret-sharing based MPC (SS-MPC). In the SS-MPC, each party distributes its inputs, and the computation proceeds with secret shares that look exactly like random numbers distributed among the parties. In the SS-MPC protocol, the parties can compute any function represented as a circuit by using shares locally and communicating among the parties. In particular, when the parties compute a complex function composed of binary and arithmetic circuits, an efficient share conversion protocol facilitates the computation of it. An important conversion protocol is a bit-composition protocol that converts a k-dimensional vector with shares on $\mathbb{Z}_2$ (i.e., shares of binary sequence) to shares on $\mathbb{Z}_{2^k}$ (i.e., shares of decimal value). Previous studies proposed a maliciously secure bit-composition protocol with robustness, which is a security notion that all parties learn the correct output regardless of the attacker's behaviour. However, its security is dependent on a statistical parameter or proved in the random oracle model. In this paper, we propose a novel bit-composition protocol with robustness independent of a statistical parameter by introducing additional clients generating the pair of shares of random values only in the offline phase (which can be performed without the parties' inputs). Our protocol is based on a maliciously secure four-party protocol with one corruption using replicated secret sharing. The security of our protocol is proved in the standard model (which is a weaker assumption than the random oracle model). Our protocol achieves efficiency and the strongest security simultaneously. We also propose a protocol for the Hamming distance with robustness by modifying our bit-composition protocol. It can achieve a secure iris recognition service via MPC with robustness. Furthermore, we extend our protocol with a constant number of parties and clients to one with an arbitrary number of parties and clients.

Keywords: secure multiparty computation, secret sharing, bit-composition protocol, robustness

1. Introduction

1.1 Background

Secure multiparty computation (MPC) [4], [14], [23] enables parties to compute an arbitrary joint function represented as a circuit without revealing the inputs of the other parties. MPC can compute it securely even if some of the parties are adversaries at a certain rate. MPC protocols guarantee *privacy* (which means that no party knows more than the output of the protocol) and *correctness* (which means that the protocol computes the correct output). These security guarantees provided by the protocols vary depending on the behaviour of the adversaries in the protocols. The typical adversary can be roughly divided into two types: *semi-honest* and *malicious adversary*. The former is a corrupted party that follows the specifications of the protocol and tries to get as much information as possible. The latter is a corrupted party that can deviate arbitrarily from the protocol. For example, it can cheat by sending incorrect values. It can also try to get as much information as possible. Therefore, the malicious adversary is stronger than the semi-honest one.

Even among the MPC schemes that are secure against a malicious adversary, the security notions regarding the correctness achieved by each scheme are different. For example, there are three typical notions: *security with abort* (the protocol is aborted if it detects a malicious adversary's cheating), *fairness* (all parties including a malicious adversary get the correct outputs or nothing) and *guaranteed output delivery* (GOD). GOD is also known as *robustness*. The strongest security notion is GOD, which ensures that all parties including a malicious adversary learn the correct outputs regardless of the attacker's behaviour. There are two main types of GOD: *traditional robustness* [16] and *private robustness* [5], [12]. The former ensures that all parties learn the correct outputs, but there is a possibility that the parties' inputs may be known to a honest party. The latter ensures that all parties learn the correct outputs while keeping information on parties' inputs secret from all parties. Hence, the latter is a stronger security notion than the former. In particular, the existing protocols with private robustness achieve the cheating detection and cheater identification implicitly because the private robustness in

1 University of Tsukuba, Tsukuba, Ibaraki 305–8577, Japan
2 NEC Corporation, Kawasaki, Kanagawa 211–8668, Japan
a) s2030119@s.tsukuba.ac.jp
b) h_tsuchida@nec.com
c) nishide@risk.tsukuba.ac.jp

This manuscript is an extended version of Ref. [22] with additional information and changes. Specifically, we add new related works, SWIFT [16] and Fantastic Four [12], to Table 1 and the body for comparison. We describe the four-party and one-client construction (Protocol 5) explicitly and add its communication costs to Table 1. We propose a new generalized protocol (Protocol 7) and add its communication costs to Table 1. We also propose a new Hamming distance calculation protocol based on the four-party and one-client construction (Protocol 9) and a new generalized protocol (Protocol 10).

these protocols is achieved by detecting the cheating and removing the values sent from the identified cheater without revealing the parties' inputs. For example, Fantastic Four [12] is capable of detecting cheating deterministically, but the identification of the cheater is achieved probabilistically depending on a statistical parameter. FLASH [5] is capable of both detecting cheating and identifying the cheater, deterministically. The performance of the protocols that realize probabilistic cheating detection or probabilistic cheater identification degrades as the statistical parameter is increased. Hence, if strong security is to be achieved, the protocols that realize deterministic cheating detection and deterministic cheater identification are preferable.

The practical advantage of MPC with GOD is that it is secure against a malicious adversary who performs denial of service (DoS) attacks. Real applications or services using MPC schemes that achieve security with abort or fairness are not secure against DoS attacks because it is easy for the adversary to shut them down by sending incorrect values in MPC. In comparison, real applications or services using MPC schemes with GOD are robust against such DoS attacks.

A typical MPC is a *secret sharing based MPC* (*SS-MPC*) [4], [14]. In the SS-MPC, each party distributes its inputs, and the computation proceeds with secret shares that look exactly like random numbers distributed among the parties. In the SS-MPC protocol, each party computes any function represented as a binary, arithmetic or mixed circuit (which is composed of binary and arithmetic circuits) by using shares locally and communicating among the parties. In particular, the secure three-party or four-party protocol [5], [11], [12], [16], [17], [21] has gained attention in recent years because it can achieve a high throughput even when it computes a complex function represented as mixed circuits.

An efficient *share conversion* protocol is required when the SS-MPC protocol computes a complex function represented as mixed circuits. Let $\mathbb{Z}_2$ and $\mathbb{Z}_{2^k}$ be residue ring modulo 2 and 2^k, respectively. For example, the addition of the shares on the arithmetic ring $\mathbb{Z}_{2^k}$ is for free. However, the exclusive OR (XOR) operation of the shares of bit on $\mathbb{Z}_{2^k}$ requires communication. In comparison, the XOR operation for the shares on the binary ring $\mathbb{Z}_2$ is for free. However, the addition of shares on $\mathbb{Z}_2$ requires communication. Therefore, converting shares with changing the modulus according to the type of circuits can reduce the communication cost of the entire computation.

The bit-composition protocol is a protocol that converts a k-dimensional vector with shares of $x_j \in \{0,1\}$ $(j = 0,\ldots,k-1)$ on $\mathbb{Z}_2$ (or a binary field $\mathbb{F}_2$) to shares of $x(= \sum_{j=0}^{k-1} 2^j \cdot x_j)$ on $\mathbb{Z}_{2^k}$ (or a finite field $\mathbb{F}_q$ s.t. q is a prime number). In particular, the bit conversion protocol is a protocol that converts shares of $x \in \{0,1\}$ on $\mathbb{Z}_2$ (or $\mathbb{F}_2$) to shares of x on $\mathbb{Z}_{2^k}$ (or $\mathbb{F}_q$). Note that these conversion protocols that do not require changing the modulus are easily achievable. However, it is hard to construct these protocols by changing the modulus. An existing maliciously secure bit-composition protocol achieves security with abort [1], [17], fairness [11], [21], or GOD [5], [12], [16]. Byali et al. proposed a bit conversion protocol with private robustness independent of a statistical parameter in Ref. [5]. The bit-composition protocol with private robustness independent of a statistical parameter can be achieved by running the bit conversion protocol with private robustness independent of statistical parameter [5] in parallel k times and using shares locally. However, the bit conversion protocol in Ref. [5] uses a commitment scheme that uses a collision-resistant hash function. Therefore, its security is proved only in the random oracle model (ROM). In the ROM, the collision-resistant hash function is replaced by an ideal random function. In comparison, there are no such replacements in the standard model. Therefore, the ROM is a stronger assumption than the standard model.

From a practical viewpoint, an MPC scheme that computes complex functions represented as mixed circuits and achieves private robustness independent of a statistical parameter is preferred. Hence, a bit-composition protocol with private robustness independent of a statistical parameter is required. However, there are no such protocols for the standard model.

1.2 Our Results

In this paper, we propose a client-aided bit-composition protocol with private robustness independent of a statistical parameter that is provably secure in the standard model. The proposed scheme is based on a maliciously secure four-party computation with one corruption [5] with three additional clients who assist the parties in the calculation. Note that these clients provide an auxiliary input only in the preprocessing phase. They do not input any secret values, do not compute any values, and do not learn the output during the actual computation. That is, we propose a maliciously secure seven-party bit-composition protocol with one corruption that achieves private robustness independent of a statistical parameter and is provably secure in the standard model for the first time. Our scheme can improve the efficiency and security of computation for complex functions represented as mixed circuits.

We also propose a secure computation protocol for the Hamming distance by modifying the proposed bit-composition protocol. Our protocol for the Hamming distance can be useful for secure iris recognition applications.

Furthermore, we extend our protocol with a constant number of parties and clients to one with an arbitrary number of parties and clients. In our extended protocol, the parameters (i.e., the number of parties and clients) are flexible so that it fits in many situations.

Table 1 shows a comparison of the communication cost between the proposed scheme and the existing maliciously secure bit-composition protocols. This table shows that only our scheme achieves private robustness composed of deterministic cheating detection and cheater identification in the standard model. Furthermore, it shows that the number of communication rounds of our scheme is the lowest compared with other previous schemes.

1.3 Related Work

1.3.1 Typical Method for Reducing Communication Cost

In MPC protocols, it is important to reduce the communication cost. For example, introducing entities that help with part of the computation [3] is known as a typical methodol-

Table 1 Comparison between existing maliciously secure bit-composition protocols with one corruption and proposed protocol (*Rounds: number of communication rounds, Comm.:* (*amortized*) *communication bits per all parties, k: bit length of modulus, N: number of parties, H: number of clients, t: number of malicious corruptions in protocol, t_p: number of malicious corruptions in parties, t_c: number of malicious corruptions in clients, Std.: standard model, ROM: random oracle model*).

Scheme	Threshold	Property	Deterministic cheating detection	Deterministic cheater identification	Model	Offline phase		Online phase	
						Rounds	Comm.	Rounds	Comm.
ABY^3 [17] + [2]	$(t,N)=(1,3)$	abort	(probabilistic)	(not achievable)	Std.	3	$12k\log_2 k+12k$	$1+\log_2 k$	$9k\log_2 k+9k$
BLAZE [21]	$(t,N)=(1,3)$	fairness	(probabilistic)	(not achievable)	Std.	5	$9k^2$	1	$4k^2$
Trident [11]	$(t,N)=(1,4)$	fairness	✓	(not achievable)	Std.	2	$3k^2+k$	1	$3k$
FLASH [5]	$(t,N)=(1,4)$	private robustness	✓	✓	ROM	2	$4k^2$	3	$10k^2$
SWIFT [16]	$(t,N)=(1,4)$	traditional robustness	✓	(not achievable)	Std.	2	$3k^2+4k$	1	$3k^2$
Fantastic Four [12]	$(t,N)=(1,4)$	private robustness	✓	(probabilistic)	Std.	2	$18k^2$	1	$8k$
This Work (Protocol 4)	$(t,N,H)=(1,4,3)$	private robustness	✓	✓	Std.	1	$24k^2+24k$	1	$8k$
This Work (Protocol 5)	$(t_p,N,H)=(1,4,1)$	private robustness	✓	✓	Std.	1	$12k^2+12k$	1	$8k$
This Work (Protocol 7)	$t_p(2t_p+1)<N$, $2t_c+1<H$	private robustness	✓	✓	Std.	1	$(t_c+1)(2t_p+1)k^2$ $+(t_c+1)(2t_p+1)k$	1	$N(N-(2t_p+1))(t_p+1)k$

ogy for reducing the communication cost. The offline-online paradigm [5], [10], [11], [21] is another method. The offline-online paradigm divides the MPC protocol into an offline phase and online phase. In the offline phase, the protocol processes part of the computation that can be done independently of the parties' inputs. In the online phase, the protocol processes the rest of the computation with the parties' inputs. The offline-online paradigm can reduce the communication cost of the online phase even if it increases the communication cost of the offline phase and the whole computation. The offline-online paradigm is suited to MPC applications that focus on the response time of queries. However, to the best of our knowledge, simultaneously realizing private robustness and the communication cost reduction of the bit-composition protocol by these methodologies has not been proposed.

1.3.2 Bit-composition Protocol

There are two main ways to realize the existing bit-composition protocol: using an adder circuit and executing a bit conversion protocol in parallel. The former approach is used in Ref. [17]. It can change the communication complexity according to the adder circuit used in the protocol. It can reduce the total communication volume, but it is not a constant-round protocol. Therefore, it is not suited to a network with low latency.

The latter one is used in Refs. [1], [5], [11], [12], [21]. It is a constant-round protocol. Hence, it is suited to a network with low latency. In particular, this approach is compatible with the offline-online paradigm. In Refs. [1], [11], [12], [16], the shares of the random bit $r_j \in \{0,1\}$ $(j=0,\ldots,k-1)$ on $\mathbb{Z}_2$ and $\mathbb{Z}_{2^k}$ (or $\mathbb{F}_q$) are prepared in the offline phase. Then, in the online phase, the communication complexity can be reduced to $O(k)$ by using the shares of the random bit generated in the offline phase, even if the bit conversion protocol is executed in parallel. Among the existing works, only the bit conversion protocol in Ref. [5] achieves private robustness independent of a statistical parameter (in ROM). Therefore, to the best of our knowledge, there is no bit-composition protocol that simultaneously achieves private robustness independent of a statistical parameter in the standard model and $O(k)$ communication complexity in the online phase.

1.3.3 MPC with GOD

Most of the existing protocols have used a broadcast channel or an expensive asymmetric-key primitive [6], [7], [15]. In recent years, secure four-party protocols achieving GOD without the broadcast channel or the expensive asymmetric-key primitive are proposed [5], [12], [16]. Byali et al. proposed a secure four-party protocol with GOD (private robustness independent of a statistical parameter) that does not use the broadcast channel or an expensive asymmetric-key primitive in Ref. [5]. However, it uses a commitment scheme of which the security is proved in ROM. Therefore, the security of the whole protocol is also proved in ROM. To the best of our knowledge, there has been no proposed general MPC protocol that achieves private robustness independent of a statistical parameter in the standard model without a broadcast channel or an expensive asymmetric-key primitive nor even a specific protocol that achieves it.

2. Preliminaries

2.1 Notations

Let $\mathbb{Z}_2$ and $\mathbb{Z}_{2^k}$ be residue rings modulo 2 and 2^k, respectively. We set the notation of share vectors and inner-products such that $\vec{x}=(x_0,\ldots,x_{k-1}), \vec{y}=(y_0,\ldots,y_{k-1})$, and $\vec{x}\cdot\vec{y}=\sum_{j=0}^{k-1} x_j\cdot y_j \mod 2^k$, where $x_j, y_j \in \mathbb{Z}_{2^k}$ $(j=0,\ldots,k-1)$. We denote the XOR operator and AND operator as $\oplus$ and $\cdot$, respectively. Note that we also use $\cdot$ as the multiplication operator on $\mathbb{Z}_{2^k}$. Let P_i be the i-th party $(i=0,\ 1,\ 2,\ 3)$. P_i has a collision-resistant hash function. We let $H_{i'}$ $(i'=0,\ 1,\ 2)$ be the clients who are the helper entities. The security parameter is denoted as κ. The κ-bit string is $\{0,1\}^\kappa$. We use the (cryptographically secure) pseudo-random functions $F:\{0,1\}^\kappa\times\{0,1\}^\kappa\to\mathbb{Z}_2$, $F':\{0,1\}^\kappa\times\{0,1\}^\kappa\to\mathbb{Z}_{2^k}$. $H_{i'}$ has $\mathsf{seed}\in\{0,1\}^\kappa$. We use the unique identifier, $\mathsf{vid}_j, \mathsf{vid}_j^{(\sigma)}, \mathsf{vid}_j^{(\sigma_1)}, \mathsf{vid}_j^{(\mu_1)}, \mathsf{vid}_j^{(1)},\ldots,\mathsf{vid}_j^{(N-1)}\in\{0,1\}^\kappa$. These are unique identifiers and function as nonces. They are public identifiers (e.g., counter values). That is, any clients (and parties) can know these values.

We also use collision-resistant hash function $\mathcal{H}$ in the same way as Byali et al. [5] for checking the message consistency (not the commitment scheme). Note that the cryptographic hash function is required to have the following three properties:

(1) **Preimage resistance**: It is (computationally) hard to compute the value x from $\mathcal{H}(x)$.
(2) **Collision resistance**: It is (computationally) hard to find the pair (x,y) such that $\mathcal{H}(x)=\mathcal{H}(y)$ and $x\neq y$.
(3) **Second preimage resistance**: Given the value y, it is (com-

putationally) hard to compute the value x such that $\mathcal{H}(x) = \mathcal{H}(y)$ and $x \neq y$. Note that $\mathcal{H}$ has the second preimage resistance if $\mathcal{H}$ has the collision resistance.

We use the collision-resistant hash function $\mathcal{H}$. (Actually, to prove that our protocols achieve private robustness, it is sufficient for $\mathcal{H}$ to have the second preimage resistance.)

2.2 2-out-of-4 Replicated Secret Sharing Scheme ((2,4)-RSS)

We use the (2,4)-RSS (also known as mirrored sharing) in Ref. [5]. We denote the (2,4)-RSS's shares of x on $\mathbb{Z}_{2^k}$ as $[x]$. Each P_i has share $[x]_i$ as follows, where $\sigma_x, \sigma_x^1, \sigma_x^2 \in \mathbb{Z}_{2^k}$, and $\mu_x, \mu_x^1, \mu_x^2 \in \mathbb{Z}_{2^k}$ such that $\mu_x = x + \sigma_x \bmod 2^k$, $\sigma_x = \sigma_x^1 + \sigma_x^2 \bmod 2^k$, and $\mu_x = \mu_x^1 + \mu_x^2 \bmod 2^k$:

- P_0's share: $[x]_0 = (\sigma_x^1, \mu_x^1, \mu_x^2)$
- P_1's share: $[x]_1 = (\sigma_x^1, \sigma_x^2, \mu_x^1)$
- P_2's share: $[x]_2 = (\sigma_x^2, \mu_x^1, \mu_x^2)$
- P_3's share: $[x]_3 = (\sigma_x^1, \sigma_x^2, \mu_x^2)$

We denote the (2,4)-RSS's shares of x on $\mathbb{Z}_2$ as $[x]^{\mathsf{B}}$. We note that P_0, P_1, P_2 and P_3 correspond to E_1, V_1, E_2 and V_2 in Ref. [5], respectively.

2.3 $(N - 2t_{\mathsf{p}})$-out-of-N Replicated Secret Sharing Scheme ($(N - 2t_{\mathsf{p}}, N)$-RSS)

Protocol 1 $x \leftarrow \pi_{\mathsf{mboOneParty}}([\![x]\!]^{\mathsf{B}}, \ell)$

Input: $[\![x]\!]^{\mathsf{B}}$ (where $x, x_j \in \mathbb{Z}_2$ and $x = x_0 \oplus \cdots \oplus x_{N-1} \bmod 2$)
Output: P_ℓ obtains x.

1: **for** $j = 0, \ldots, \ell - 1, \ell + 2t_{\mathsf{p}} + 1, \ldots, N - 1$ **do in parallel**
2: Each P_i sends x_j as $m_{j,i}$ to P_ℓ for $i = j-(2t_{\mathsf{p}}+1), \ldots, j-t_{\mathsf{p}}$. // 1 round and $t_{\mathsf{p}} + 1$ bits
3: Each P_i sends the hashed value of x_j, $h_{j,i} = \mathcal{H}(x_j) = \mathcal{H}(m_{j,i})$ for $i = j - t_{\mathsf{p}}, \ldots, j$. // 1 round & $t_{\mathsf{p}}|h_{i,j}|$ bits
4: P_ℓ computes the hashed value of $m_{j,i}$, $h_{j,i}$ for $i = j - (2t_{\mathsf{p}} + 1), \ldots, j - t_{\mathsf{p}}$. Then, P_ℓ chooses $m_j \in \{m_{j,i}\}_{i=j-(2t_{\mathsf{p}}+1)}^{j-t_{\mathsf{p}}}$ as the correct value x_j if $t_{\mathsf{p}} + 1$ or more of the hashed values in $\{h_{j,i}\}_{i=0}^{N-1}$ match the hashed value of m_j, $\mathcal{H}(m_j)$.
5: **end for**
6: P_ℓ computes $x = x_0 \oplus \cdots \oplus x_{N-1} \bmod 2$ by using $[\![x]\!]_\ell^{\mathsf{B}}$ and the chosen values as correct values in the previous steps.

Protocol 2 $x \leftarrow \pi_{\mathsf{mbo}}([\![x]\!]^{\mathsf{B}})$

Input: $[\![x]\!]^{\mathsf{B}}$ (where $x, x_i \in \mathbb{Z}_2$ and $x = x_0 \oplus \cdots \oplus x_{N-1} \bmod 2$)
Output: All parties obtain x.

1: **for** $i = 0, \ldots, N - 1$ **do in parallel**
2: All parties run $\pi_{\mathsf{mboOneParty}}([\![x]\!]^{\mathsf{B}}, i)$ and P_i gets x. // 1 round & $N(N - (2t_{\mathsf{p}} + 1))(t_{\mathsf{p}} + 1)$ bits
3: **end for**

Let N and t_{p} be the number of parties and the corruptions in the parties. We use the $(N - 2t_{\mathsf{p}}, N)$-RSS where $t_{\mathsf{p}}(2t_{\mathsf{p}} + 1) < N$ (i.e., at most $O(\sqrt{N})$ corruptions similar to the secure N-party computation protocol proposed by Byali et al. [6] or Chandran et al. [9]). We denote the $(N - 2t_{\mathsf{p}}, N)$-RSS's shares of x on $\mathbb{Z}_{2^k}$ as $[\![x]\!]$. Each P_i has share $[\![x]\!]_i = (x_i, \ldots, x_{i+2t_{\mathsf{p}}})$ where $x = x_0 + \cdots + x_{N-1} \bmod 2^k$. That is, each P_i has $(2t_{\mathsf{p}} + 1)$ of these x_j $(j = 0, \ldots, N - 1)$, ranging from x_i to $x_{i+2t_{\mathsf{p}}}$ as shares. In other words, each x_j $(j = 0, \ldots, N - 1)$ is possessed by $(2t_{\mathsf{p}} + 1)$ parties. Therefore, if $t_{\mathsf{p}}(2t_{\mathsf{p}} + 1) < N$ holds, then the secret value cannot be reconstructed from shares of $(N - 2t_{\mathsf{p}}, N)$-RSS even if t_{p} corruptions occur. We also denote the $(N - 2t_{\mathsf{p}}, N)$-RSS's shares of x on $\mathbb{Z}_2$ as $[\![x]\!]^{\mathsf{B}}$ as in Protocols 1 (multiparty binary shares opening (mbo) protocol for one party, $\pi_{\mathsf{mboOneParty}}$) and 2 (multiparty binary shares opening protocol, π_{mbo}).

Each party P_i (for $i = 0, \ldots, N - 1$) performs the share addition, $[\![x + y]\!] = [\![x]\!] + [\![y]\!]$ by setting $[\![x + y]\!]_i = (x_i + y_i \bmod 2^k, \ldots, x_{i+2t_{\mathsf{p}}} + y_{i+2t_{\mathsf{p}}} \bmod 2^k)$. If parties perform the scalar addition, $[\![c + x]\!] = c + [\![x]\!]$, where $x, x_i \in \mathbb{Z}_{2^k}$, $x = \sum_{i=0}^{N-1} x_i \bmod 2^k$, and $c \in \mathbb{Z}_{2^k}$ is public, the parties who obtain x_0 replace x_0 (in their shares) with $x'_0 = x_0 + c \bmod 2^k$. Each party performs the scalar multiplication, $[\![c \cdot x]\!] = c \cdot [\![x]\!]$, where $c \in \mathbb{Z}_{2^k}$, by setting $[\![c \cdot x]\!]_i = (c \cdot x_i \bmod 2^k, \ldots, c \cdot x_{i+2t_{\mathsf{p}}} \bmod 2^k)$. Therefore, the linearity of shares of $(N - 2t_{\mathsf{p}}, N)$-RSS on $\mathbb{Z}_{2^k}$ holds, i.e., $[\![c_0 \cdot x + c_1 \cdot y]\!] = c_0 \cdot [\![x]\!] + c_1 \cdot [\![y]\!]$, where $c_0, c_1 \in \mathbb{Z}_{2^k}$. The linearity of shares of $(N - 2t_{\mathsf{p}}, N)$-RSS on $\mathbb{Z}_2$ holds in the same way as $(N - 2t_{\mathsf{p}}, N)$-RSS on $\mathbb{Z}_{2^k}$.

We explain how to reconstruct the secret value from shares of $(N - 2t_{\mathsf{p}}, N)$-RSS. We describe the opening protocol of $(N - 2t_{\mathsf{p}}, N)$-RSS on $\mathbb{Z}_2$, Protocol 2, because we use it only on $\mathbb{Z}_2$ (not $\mathbb{Z}_{2^k}$) in our protocol. Protocol 2 is the opening protocol on $\mathbb{Z}_2$ for all parties. It is realized by running the opening protocol on $\mathbb{Z}_2$ for one party (Protocol 1) in parallel. Protocols 1 and 2 achieve private robustness independent of a statistical parameter because of the majority voting in Line 4 of Protocol 1 [*1].

Protocol 1 requires 1 round and $(N-(2t_{\mathsf{p}}+1))(t_{\mathsf{p}}+1+t_{\mathsf{p}}|h_{i,j}|)$ bits as communication bits. Note that we can ignore the communication cost of Step 3 of Protocol 1 when running multiple instances. Hence, Protocol 1 requires 1 round and $(N - (2t_{\mathsf{p}} + 1))(t_{\mathsf{p}} + 1)$ bits as amortized communication bits. Therefore, Protocol 2 requires 1 round and $N(N - (2t_{\mathsf{p}} + 1))(t_{\mathsf{p}} + 1)$ bits as amortized communication bits.

Note that we do not use the sharing protocol of $(N - 2t_{\mathsf{p}}, N)$-RSS in our protocol. Hence, we describe it in Appendix A.1 for completeness.

2.4 Secure Four-party Computation with One Corruption

We use the same addition of shares as Ref. [5]. We denote the addition of shares as $[x] + [y]$. We also use the same scalar addition and scalar multiplication of shares as Ref. [5]. We denote the scalar addition and scalar multiplication of shares as $c + [x]$ and $c \cdot [x]$, where $c \in \mathbb{Z}_{2^k}$. We denote the operations on $\mathbb{Z}_2$ in the same way as on $\mathbb{Z}_{2^k}$. Note that we use the same notation for operations of scalars and those of the shares to keep the description simple.

We also use the same four-party binary shares opening (bo) protocol (i.e., output computation protocol) as Ref. [5]. The bo protocol of Ref. [5] (not the bit conversion protocol of Ref. [5]) achieves private robustness independent of a statistical parameter in the standard model because it uses the collision-resistant hash

[*1] At Line 4 in Protocol 1, the security is not compromised even if P_i $(i = j - t_{\mathsf{p}}, \ldots, j)$ sends the raw value, i.e., $m_{j,i}$. The reason why P_i $(i = j - t_{\mathsf{p}}, \ldots, j)$ sends the hashed value, $\mathcal{H}(m_{j,i})$, is just to reduce the cost of communication volume.

function only for reducing the communication cost but does not use the commitment scheme based on the collision-resistant hash function. We denote the bo protocol on $\mathbb{Z}_2$ as π_{bo}. π_{bo} takes a share $[x]^{\mathsf{B}}$ as input and outputs $x \in \mathbb{Z}_2$. We denote $x \leftarrow \pi_{\mathsf{bo}}([x]^{\mathsf{B}})$ when it is called. Protocol π_{bo} needs one round and 8 bits as the (amortized) communication cost, respectively.

2.5 Definition of Security

Let $\mathsf{view}_i^{\pi}(\vec{x})$ be the view of P_i while running protocol π on inputs $\vec{x}$. It consists of its inputs $\vec{x}$, all the messages received by P_i, and an internal random coin. Let $\mathsf{output}^{\pi}(\vec{x})$ be the outputs of all parties while running protocol π on inputs $\vec{x}$.

Definition 1 (Perfect Security in the Presence of One Malicious Corrupted Party)**.** *Let $f : (\{0,1\}^*)^7 \to (\{0,1\}^*)^7$ be a deterministic 7-ary functionality and let π be a protocol. We say that π computes f with perfect security in the presence of one malicious corrupted party for f if there exists probabilistic polynomial-time algorithm $\mathcal{S}$ such that for every corrupted party $i \in \{0,1,2,3,4,5,6\}$, and every $\vec{x} \in (\{0,1\}^*)^7$ where $|x_0| = |x_1| = |x_2| = |x_3| = |x_4| = |x_5| = |x_6|$:*

$$\{(\mathcal{S}(x_i,\ f_i(\vec{x})),\ f(\vec{x}))\} \equiv \{(\mathsf{view}_i^{\pi}(\vec{x}), \mathsf{output}^{\pi}(\vec{x}))\} \qquad (1)$$

We note that Definition 1 is modified from the security definition of Ref. [13] to adapt the proposed seven-party (i.e., four parties P_0, P_1, P_2 and P_3 and three helper $H_0(= P_4)$, $H_1(= P_5)$ and $H_2(= P_6)$) bit-composition protocol with private robustness independent of a statistical parameter. If the above Eq. (1) holds with computational indistinguishability, then we say that π computes f with computational security in the presence of one malicious corrupted party.

We use the hybrid model [8] to prove the security of the proposed scheme. Let g be the subfunctionality. We say that protocol π is secure in the g-hybrid model. We denote protocol π secure in the g-hybrid model as π^g.

3. Proposal

We propose the new client-aided bit composition protocol with private robustness independent of statistical parameter in Section 3.1. The bit-composition protocol is useful as a subprotocol, but not a useful application. Hence, by modifying our bit-composition protocol, we also propose a new client-aided secure Hamming distance calculation protocol with private robustness independent of statistical parameter as a useful application in Section 3.2. It is useful for the secure iris recognition as a biometric authentication service.

For more details, in Section 3.1, we propose a three bit-composition protocols. In Section 3.1.1, we propose a four-party and three-clients construction with one malicious corruption. We prove that it is secure in Section 3.1.2. We also propose a four-party and one-client construction with one malicious corruption by modifying the four-party and three-clients construction to reduce the number of clients (in Section 3.1.3). In order to allow flexibility in the number of parties and clients, we propose a multi-party and multi-client construction with malicious corrupted parties and clients (in Section 3.1.4).

In Section 3.2, we propose the secure Hamming distance calculation protocol (in Section 3.2.1). We explain how to use our protocol in the secure iris recognition in Section 3.2.2. Then, we modify and extend our secure Hamming distance calculation protocol to reduce the number of clients and allow flexibility in the number of parties and clients in Section 3.2.3.

3.1 Client-aided Bit-composition Protocol with Private Robustness Independent of Statistical Parameter

Protocol 3 $\{[r_j]^{\mathsf{B}},\ [r_j]\}_{j=0}^{n-1} \leftarrow \pi_{\mathsf{rRndGen}}(F, F', \mathcal{H}, \mathsf{seed}, \{\mathsf{vid}_j, \mathsf{vid}_j^{(\sigma)}, \mathsf{vid}_j^{(\sigma_1)}, \mathsf{vid}_j^{(\mu_1)}\}_{j=0}^{n-1})$

Input: pseudo-random function $F : \{0,1\}^{\kappa} \times \{0,1\}^{\kappa} \to \mathbb{Z}_2$ and $F' : \{0,1\}^{\kappa} \times \{0,1\}^{\kappa} \to \mathbb{Z}_{2^k}$, collision-resistant hash function $\mathcal{H}$, shared seed by clients seed, unique identifiers $\{\mathsf{vid}_j, \mathsf{vid}_j^{(\sigma)}, \mathsf{vid}_j^{(\sigma_1)}, \mathsf{vid}_j^{(\mu_1)}\}_{j=0}^{n-1}$ *where* $\mathsf{seed}, \mathsf{vid}_j, \mathsf{vid}_j^{(\sigma)}, \mathsf{vid}_j^{(\sigma_1)}, \mathsf{vid}_j^{(\mu_1)} \in \{0,1\}^{\kappa}$ (n is a positive integer)

Output: $\{[r_j]^{\mathsf{B}},\ [r_j]\}_{j=0}^{n-1}$ *s.t.* $r_j \in \mathbb{Z}_2$

1: **for** $j = 0, \ldots, n-1$ **do**
2: Each $H_{i'}$ ($i' = 0, 1, 2$) computes $r_j = F(\mathsf{seed},\ \mathsf{vid}_j)$.
3: Each $H_{i'}$ computes $\sigma_{r_j} = F(\mathsf{seed},\ \mathsf{vid}_j^{(\sigma)})$, $\sigma^1_{r_j} = F(\mathsf{seed},\ \mathsf{vid}_j^{(\sigma_1)})$, $\sigma^2_{r_j} = \sigma_{r_j} \oplus \sigma^1_{r_j} \bmod 2$.
4: Each $H_{i'}$ computes $\mu_{r_j} = r_j \oplus \sigma_{r_j} \bmod 2$, $\mu^1_{r_j} = F(\mathsf{seed},\ \mathsf{vid}_j^{(\mu_1)})$, $\mu^2_{r_j} = \mu_{r_j} \oplus \mu^1_{r_j} \bmod 2$.
5: Each $H_{i'}$ computes $\sigma'_{r_j} = F'(\mathsf{seed},\ \mathsf{vid}_j^{(\sigma)})$, $\sigma'^1_{r_j} = F'(\mathsf{seed}, \mathsf{vid}_j^{(\sigma_1)})$, $\sigma'^2_{r_j} = \sigma'_{r_j} - \sigma'^1_{r_j} \bmod 2^k$.
6: Each $H_{i'}$ computes $\mu'_{r_j} = r_j + \sigma'_{r_j} \bmod 2^k$, $\mu'^1_{r_j} = F'(\mathsf{seed}, \mathsf{vid}_j^{(\mu_1)})$, $\mu'^2_{r_j} = \mu'_{r_j} - \mu'^1_{r_j} \bmod 2^k$.
7: Each $H_{i'}$ sets $[r_j]_0^{\mathsf{B}} = (\sigma^1_{r_j},\ \mu^1_{r_j},\ \mu^2_{r_j})$, $[r_j]_1^{\mathsf{B}} = (\sigma^1_{r_j},\ \sigma^2_{r_j},\ \mu^1_{r_j})$, $[r_j]_2^{\mathsf{B}} = (\sigma^2_{r_j},\ \mu^1_{r_j},\ \mu^1_{r_j})$, $[r_j]_3^{\mathsf{B}} = (\sigma^1_{r_j},\ \sigma^2_{r_j},\ \mu^2_{r_j})$.
8: Each $H_{i'}$ sets $[r_j]_0 = (\sigma'^1_{r_j},\ \mu'^1_{r_j},\ \mu'^2_{r_j})$, $[r_j]_1 = (\sigma'^1_{r_j},\ \sigma'^2_{r_j},\ \mu'^1_{r_j})$, $[r_j]_2 = (\sigma'^2_{r_j},\ \mu'^1_{r_j},\ \mu'^1_{r_j})$, $[r_j]_3 = (\sigma'^1_{r_j},\ \sigma'^2_{r_j},\ \mu'^2_{r_j})$.
9: **end for**
10: H_0 and H_1 send the set of shares $\{[r_j]_i^{\mathsf{B}},\ [r_j]_i\}_{j=0}^{n-1}$ to each P_i ($i = 0, 1, 2, 3$). H_2 computes the hashed value of $\{[r_j]_i^{\mathsf{B}},\ [r_j]_i\}_{j=0}^{n-1}, \mathcal{H}(R_{2,i})$, and send it to each P_i. // 1 round & $24nk + 24n$ bits
11: Let each $R_{0,i}$ and $R_{1,i}$ be the set of shares $\{[r_j]_i^{\mathsf{B}},\ [r_j]_i\}_{j=0}^{n-1}$ that is sent to P_i from H_0 and H_1, respectively. Each P_i computes the hashed value of $R_{0,i}$ and $R_{1,i}$, $\mathcal{H}(R_{0,i})$ and $\mathcal{H}(R_{1,i})$. If $\mathcal{H}(R_{0,i}) = \mathcal{H}(R_{1,i})$ or $\mathcal{H}(R_{0,i}) = \mathcal{H}(R_{2,i})$, each P_i outputs $R_{0,i}$. If $\mathcal{H}(R_{1,i}) = \mathcal{H}(R_{2,i})$, each P_i outputs $R_{1,i}$.

Protocol 4 $[x] \leftarrow \pi^{\mathcal{F}_{\mathsf{rRndGen}}}_{\mathsf{rBitComp}}(\{[x_j]^{\mathsf{B}}\}_{j=0}^{k-1})$

Input: $\{[x_j]^{\mathsf{B}}\}_{j=0}^{k-1}$ (where $x_j \in \mathbb{Z}_2$)

Output: $[x]$ *s.t.* $x = \sum_{j=0}^{k-1} 2^j \cdot x_j \mod 2^k$

1: (Offline phase)
2: P_0, P_1, P_2, P_3, H_0, H_1 and H_2 invoke $\mathcal{F}_{\mathsf{rRndGen}}$ where $n = k$, then get $\{[r_j]^{\mathsf{B}},\ [r_j]\}_{j=0}^{k-1}$. // π_{rRndGen} requires 1 round & $24k^2 + 24k$ bits
3: (Online phase)
4: Each P_i ($i = 0, 1, 2, 3$) computes $[c_j]_i^{\mathsf{B}} = [x_j \oplus r_j]_i^{\mathsf{B}} = [x_j]_i^{\mathsf{B}} \oplus [r_j]_i^{\mathsf{B}}$ for $j = 0, \ldots, k-1$.
5: Each P_i gets the value $c_j \leftarrow \pi_{\mathsf{bo}}([c_j]^{\mathsf{B}})$ for $j = 0, \ldots, k-1$ in parallel. // 1 round & $8k$ bits

FUNCTIONALITY 1 ($\mathcal{F}_{\mathsf{rRndGen}}$ - generating two types of random shares).
(1) $\mathcal{F}_{\mathsf{rRndGen}}$ receives message (gen, n) from clients H_0, H_1 and H_2.
(2) $\mathcal{F}_{\mathsf{rRndGen}}$ generates $\{[r_j]^{\mathsf{B}}, [r_j]\}_{j=0}^{n-1}$ randomly where $r_j \in \{0, 1\}$.
(3) $\mathcal{F}_{\mathsf{rRndGen}}$ sends $\{[r_j]^{\mathsf{B}}, [r_j]\}_{j=0}^{n-1}$ to parties P_0, P_1, P_2, and P_3.

FUNCTIONALITY 2 ($\mathcal{F}_{\mathsf{rBitComp}}$ - converting binary share vector on $\mathbb{Z}_2$ to a share on $\mathbb{Z}_{2^k}$).
(1) $\mathcal{F}_{\mathsf{rBitComp}}$ receives shares $\{[x_j]^{\mathsf{B}}\}_{j=0}^{k-1}$ from parties P_0, P_1, P_2 and P_3.
(2) $\mathcal{F}_{\mathsf{rBitComp}}$ reconstructs x_j and computes $x = \sum_{j=0}^{k-1} 2^j \cdot x_j \bmod 2^k$. Then, $\mathcal{F}_{\mathsf{rBitComp}}$ computes $[x]$ and sends it to parties P_0, P_1, P_2, and P_3.

6: $[x] = \sum_{j=0}^{k-1} 2^j \cdot (c_j + [r_j] - 2 \cdot c_j \cdot [r_j]) \bmod 2^k$.

3.1.1 Four-party and Three-clients Construction with One Malicious Corruption

Our scheme is divided into three steps. In the first step (at Lines 1 and 2 in Protocol 4), three clients generate the shares of random bits $r_j \in \mathbb{Z}_2$ ($j = 0, \ldots, k-1$) on $\mathbb{Z}_2$ and $\mathbb{Z}_{2^k}$, i.e., $\{[r_j]^{\mathsf{B}}, [r_j]\}_{j=0}^{k-1}$ by running ideal functionality $\mathcal{F}_{\mathsf{rRndGen}}$ (where $n = k$). In the actual protocol, ideal functionality $\mathcal{F}_{\mathsf{rRndGen}}$ is replaced with protocol π_{rRndGen} in Protocol 3. In π_{rRndGen}, three clients generate random values by using the same seed, the unique identifiers $\{\mathsf{vid}_j, \mathsf{vid}_j^{(\sigma)}, \mathsf{vid}_j^{(\sigma_1)}, \mathsf{vid}_j^{(\mu_1)}\}_{j=0}^{n-1}$, and pseudo-random functions F and F' (from Line 1 to 6 in Protocol 3). Then, they set the random values as shares $\{[r_j]^{\mathsf{B}}, [r_j]\}_{j=0}^{k-1}$ (from Line 7 to 8 in Protocol 3) and send $\{[r_j]_i^{\mathsf{B}}, [r_j]_i\}_{j=0}^{k-1}$ or the hashed value of them to P_i ($i = 0, 1, 2, 3$) (at Line 10 in Protocol 3) [*2]. Each P_i selects the set of shares that matches two or more matching sets of shares sent from H_0, H_1, and H_2 as correct outputs (at Line 11 in Protocol 3). Each P_i can always get correct random shares $\{[r_j]_i^{\mathsf{B}}, [r_j]_i\}_{j=0}^{k-1}$ because there can be at most one corrupted client.

In the second step (at Lines 4 and 5 in Protocol 4), each P_i ($i = 0, 1, 2, 3$) computes $[c_j]_i^{\mathsf{B}} = [x_j \oplus r_j]_i^{\mathsf{B}} = [x_j]_i^{\mathsf{B}} \oplus [r_j]_i^{\mathsf{B}}$ for $j = 0, \ldots, k-1$ and gets the masked values c_j by running the four-party binary shares opening protocol π_{bo}, which is proposed in Ref. [5].

In the third step (at Line 6 in Protocol 4), the parties remove mask r_j from c_j by computing $[x_j] = [c_j \oplus r_j] = (c_j - [r_j])^2 = c_j + [r_j] - 2 \cdot c_j \cdot [r_j] \bmod 2^k$. We note that $(c_j)^2 = c_j$ and $(r_j)^2 = r_j$ where $c_j, r_j \in \{0, 1\}$. Then, the parties output $[x] = \sum_{j=0}^{k-1} 2^j \cdot (c_j + [r_j] - 2 \cdot c_j \cdot [r_j])$.

3.1.2 Security Proof Sketch of Protocol 4

The bit-composition protocol $\pi_{\mathsf{rBitComp}}^{\mathcal{F}_{\mathsf{rRndGen}}}$ in Protocol 4 computes $\mathcal{F}_{\mathsf{rBitComp}}$ with computational security in the presence of one malicious corrupted party because $\pi_{\mathsf{rBitComp}}^{\mathcal{F}_{\mathsf{rRndGen}}}$ consists of $\mathcal{F}_{\mathsf{rRndGen}}$, π_{bo} and local computations. The security of π_{bo} with private robustness independent of a statistical parameter in the standard model is proved in Ref. [5] because it uses majority voting and does not use a commitment protocol. Therefore, if we prove that π_{rRndGen} computes $\mathcal{F}_{\mathsf{rRndGen}}$ with computational security in the presence of one malicious corrupted party, we can also prove the security of $\pi_{\mathsf{rBitComp}}^{\mathcal{F}_{\mathsf{rRndGen}}}$ with private robustness independent of a statistical parameter in the standard model.

We prove that protocol π_{rRndGen} in Protocol 3 computes $\mathcal{F}_{\mathsf{rRndGen}}$ with computational security in the presence of one malicious corrupted party. We assume that at most one of $H_{i'}$ ($i' = 0, 1, 2$) is corrupted by a malicious adversary. In this case, $H_{i'}$ does not learn any information about secret inputs. Therefore, privacy is achieved. The correctness is also achieved with private robustness independent of a statistical parameter by majority voting (i.e., a selection of two or more matching the set of shares which is received) because there is at most one corrupted party. More specifically, either $R_{0,i}$ (that is sent to P_i from H_0) or $R_{1,i}$ (that is sent to P_i from H_1) is always the correct value because there is at most one corrupted client of $H_{i'}$ ($i' = 0, 1, 2$). Then, at Line 11 in Protocol 3, each P_i can choose either $R_{0,i}$ or $R_{1,i}$ as the correct value by knowing $\mathcal{H}(R_{2,i})$ (that is sent to P_i from H_2) and comparing $\mathcal{H}(R_{0,i})$, $\mathcal{H}(R_{1,i})$ and $\mathcal{H}(R_{2,i})$. Hence, the simulator $\mathcal{S}$ (i.e., the polynomial-time algorithm $\mathcal{S}$ in Definition 1) can be composed.

In another case, we assume that at most one of P_i ($i = 0, 1, 2, 3$) is corrupted by a malicious adversary. $\mathcal{S}$ can generate random bits $r'_j \in \mathbb{Z}_2$ ($j = 0, \ldots, k-1$). If the outputs of F and F' are indistinguishable from the random values with computational security, $\mathcal{S}$ can also generate random values $\sigma_{r'_j}, \sigma^1_{r'_j}, \mu^1_{r'_j} \in \mathbb{Z}_2$ and $\sigma'_{r'_j}, \sigma'^1_{r'_j}, \mu'^1_{r'_j} \in \mathbb{Z}_{2^k}$, then set $\sigma^2_{r'_j} = \sigma_{r'_j} \oplus \sigma^1_{r'_j} \bmod 2$, $\mu_{r'_j} = r'_j \oplus \sigma'_{r'_j} \bmod 2$, $\mu^2_{r'_j} = \mu_{r'_j} \oplus \mu^2_{r'_j} \bmod 2$, $\sigma'^2_{r'_j} = \sigma'_{r'_j} - \sigma'^1_{r'_j} \bmod 2^k$, $\mu'_{r'_j} = r'_j + \sigma'_{r'_j} \bmod 2^k$, and $\mu^2_{r'_j} = \mu_{r'_j} - \mu^2_{r'_j} \bmod 2^k$. Then, $\mathcal{S}$ can set $\{[r'_j]^{\mathsf{B}}, [r'_j]\}_{j=0}^{k-1}$ by using these random values in the same way as at Lines 7 and 8 in Protocol 3. Therefore, privacy is achieved. Correctness with private robustness independent of a statistical parameter in the standard model is also achieved because each P_i ($i = 0, 1, 2, 3$) sends no messages and cannot cheat. Hence, $\mathcal{S}$ can be composed.

To prove that π_{rRndGen} is secure with private robustness independent of a statistical parameter in the standard model, we prove that the corrupted party or client cannot break private robustness independent of a statistical parameter if $\mathcal{H}$ has the second preimage resistance. In π_{rRndGen}, only the corrupted client can break private robustness because the parties send no messages in π_{rRndGen}. Let the corrupted client be H_{c} ($\mathsf{c} = 0, 1, 2$). If H_{c} would like to cheat and break private robustness, he/she needs to find $R'_{\mathsf{c},i}$ such that $R_{\mathsf{c}-1,i} \neq R'_{\mathsf{c},i}$, $R_{\mathsf{c}+1,i} \neq R'_{\mathsf{c},i}$, $\mathcal{H}(R_{\mathsf{c}-1,i}) = \mathcal{H}(R'_{\mathsf{c},i})$, and $\mathcal{H}(R_{\mathsf{c}+1,i}) = \mathcal{H}(R'_{\mathsf{c},i})$ because breaking private robustness requires that the majority voting does not work. However, if H_{c} (knowing $R_{\mathsf{c}-1,i}$ and $R_{\mathsf{c}+1,i}$) finds such $R'_{\mathsf{c},i}$, H_{c} breaks the second preimage resistance of $\mathcal{H}$. Hence, π_{rRndGen} is secure with private robustness independent of a statistical parameter in the standard model if $\mathcal{H}$ has the second preimage resistance. We note that we can prove that π_{bo} is secure with the private robustness independent of a statistical parameter in the standard model if $\mathcal{H}$ has the second preimage resistance similarly. On the other hand, we emphasize that FLASH [5] needs the collision-resistant hash function that has not only the second preimage resistance but also the preimage resistance and collision resistance because the collision-resistant hash function in FLASH [5] is used as the commitment scheme that achieves the hiding and binding proper-

[*2] At Line 10 in Protocol 3, the security is not compromised even if H_2 sends the raw value, i.e., R_2. The reason why H_2 sends the hashed value, $\mathcal{H}(R_2)$, is just to reduce the cost of communication volume.

ties. To use a hash function as a commitment scheme, the security of FLASH needs to be proven in the ROM.

Therefore, π_{rRndGen} satisfies Definition 1 and computes $\mathcal{F}_{\mathsf{rRndGen}}$ with computational security in the presence of one malicious corrupted party. We can prove the whole security of $\pi_{\mathsf{rBitComp}}^{\mathcal{F}_{\mathsf{rRndGen}}}$ with private robustness independent of a statistical parameter in the standard model.

3.1.3 Four-party and One-client Construction with One Malicious Corrupted Party

Protocol 5 $[x] \leftarrow \pi_{\mathsf{rBitComp5}}(\{[x_j]^{\mathsf{B}}\}_{j=0}^{k-1})$

Input: $\{[x_j]^{\mathsf{B}}\}_{j=0}^{k-1}$ (where $x_j \in \mathbb{Z}_2$)
Output: $[x]$ *s.t.* $x = \sum_{j=0}^{k-1} 2^j \cdot x_j \mod 2^k$

1: (Offline phase)
2: H_0 generates $\{[r_j]^{\mathsf{B}}, [r_j]\}_{j=0}^{k-1}$ randomly and distributes it to the parties. // 1 round & $12k^2 + 12k$ bits
3: (Online phase)
4: Each P_i $(i = 0, 1, 2, 3)$ computes $[c_j]_i^{\mathsf{B}} = [x_j \oplus r_j]_i^{\mathsf{B}} = [x_j]_i^{\mathsf{B}} \oplus [r_j]_i^{\mathsf{B}}$ for $j = 0, \ldots, k-1$.
5: Each P_i gets the value $c_j \leftarrow \pi_{\mathsf{bo}}([c_j]^{\mathsf{B}})$ for $j = 0, \ldots, k-1$ in parallel. // 1 round & $8k$ bits
6: $[x] = \sum_{j=0}^{k-1} 2^j \cdot (c_j + [r_j] - 2 \cdot c_j \cdot [r_j]) \mod 2^k$.

As a natural modification of Protocol 4, we note that H_1 and H_2 are not required if it is certain that H_0 is a semi-honest client. One client is more natural in the setting than three clients. However, we stress that this modified protocol places a stronger assumption on the clients than Protocol 4 does.

The modified protocol is Protocol 5. It requires 1 round and $12k^2 + 12k$ bits in the offline phase and 1 round and $8k$ bits in the online phase as the (amortized) communication cost. Hence, the (amortized) communication cost of Protocol 5 is lower than that of Protocol 4.

3.1.4 *N*-party and *H*-client Construction with Malicious Corrupted Parties and Clients

Protocol 6 $\{[\![r_j]\!]^{\mathsf{B}}, [\![r_j]\!]\}_{j=0}^{n-1} \leftarrow \pi_{\mathsf{rMultiRndGen}}(F, F', \mathcal{H}, \mathsf{seed}, \{\mathsf{vid}_j, \mathsf{vid}_j^{(1)}, \ldots, \mathsf{vid}_j^{(N-1)}\}_{j=0}^{n-1})$

Input: pseudo-random function $F : \{0,1\}^\kappa \times \{0,1\}^\kappa \to \mathbb{Z}_2$ and $F' : \{0,1\}^\kappa \times \{0,1\}^\kappa \to \mathbb{Z}_{2^k}$, collision-resistant hash function $\mathcal{H}$, shared seed by clients seed, unique identifiers $\{\mathsf{vid}_j, \mathsf{vid}_j^{(1)}, \ldots, \mathsf{vid}_j^{(N-1)}\}_{j=0}^{n-1}$ *s.t.* $\mathsf{seed}, \mathsf{vid}_j, \mathsf{vid}_j^{(1)}, \ldots, \mathsf{vid}_j^{(N-1)} \in \{0,1\}^\kappa$ (n is a positive integer)
Output: $\{[\![r_j]\!]^{\mathsf{B}}, [\![r_j]\!]\}_{j=0}^{n-1}$ *s.t.* $r_j \in \mathbb{Z}_2$

1: **for** $j = 0, \ldots, n-1$ **do**
2: Each $H_{i'}$ $(i' = 0, \ldots, H-1)$ computes $r_j = F(\mathsf{seed}, \mathsf{vid}_j)$.
3: Each $H_{i'}$ computes $r_{j,1} = F(\mathsf{seed}, \mathsf{vid}_j^{(1)}), \ldots r_{j,N-1} = F(\mathsf{seed}, \mathsf{vid}_j^{(N-1)})$.
4: Each $H_{i'}$ sets $r_{j,0} = r_j \oplus r_{j,1} \oplus \cdots \oplus r_{j,N-1} \mod 2$.
5: Each $H_{i'}$ computes $r'_{j,1} = F'(\mathsf{seed}, \mathsf{vid}_j^{(1)}), \ldots r'_{j,N-1} = F'(\mathsf{seed}, \mathsf{vid}_j^{(N-1)})$.
6: Each $H_{i'}$ sets $r'_{j,0} = r_j - \sum_{\ell=1}^{N-1} r'_{j,\ell} \mod 2^k$.
7: Each $H_{i'}$ sets $[\![r_j]\!]_i^{\mathsf{B}} = (r_{j,i}, \ldots, r_{j,i+2t_{\mathsf{p}}})$ for $i = 0, \ldots, N-1$.
8: Each $H_{i'}$ sets $[\![r_j]\!]_i = (r'_{j,i}, \ldots, r'_{j,i+2t_{\mathsf{p}}})$ for $i = 0, \ldots, N-1$.
9: **end for**

FUNCTIONALITY 3 ($\mathcal{F}_{\mathsf{rMultiRndGen}}$ - generating two types of random shares of $(t_{\mathsf{p}}+1, N)$-RSS).
(1) $\mathcal{F}_{\mathsf{rMultiRndGen}}$ receives message (gen, n) from clients $H_{i'}$ for $i' = 0, \ldots, H-1$.
(2) $\mathcal{F}_{\mathsf{rMultiRndGen}}$ generates $\{[\![r_j]\!]^{\mathsf{B}}, [\![r_j]\!]\}_{j=0}^{n-1}$ randomly, where $r_j \in \{0,1\}$.
(3) $\mathcal{F}_{\mathsf{rMultiRndGen}}$ sends $\{[\![r_j]\!]^{\mathsf{B}}, [\![r_j]\!]\}_{j=0}^{n-1}$ to parties P_i for $i = 0, \ldots, N-1$.

FUNCTIONALITY 4 ($\mathcal{F}_{\mathsf{rMultiBitComp}}$ - converting the binary share vector of $(t_{\mathsf{p}}+1, N)$-RSS on $\mathbb{Z}_2$ to the share of $(t_{\mathsf{p}}+1, N)$-RSS on $\mathbb{Z}_{2^k}$).
(1) $\mathcal{F}_{\mathsf{rMultiBitComp}}$ receives shares $\{[\![x_j]\!]^{\mathsf{B}}\}_{j=0}^{k-1}$ from parties P_i for $i = 0, \ldots, N-1$.
(2) $\mathcal{F}_{\mathsf{rMultiBitComp}}$ reconstructs x_j and computes $x = \sum_{j=0}^{k-1} 2^j \cdot x_j \mod 2^k$. Then, $\mathcal{F}_{\mathsf{rMultiBitComp}}$ computes $[\![x]\!]$ and sends it to parties P_i for $i = 0, \ldots, N-1$.

10: $H_0 \ldots, H_{t_{\mathsf{c}}}$ send the set of shares $\{[\![r_j]\!]_i^{\mathsf{B}}, [\![r_j]\!]_i\}_{j=0}^{n-1}$ to each P_i $(i = 0, \ldots, N-1)$. $H_{i'''}$ $(i''' = t_{\mathsf{c}}+1, \ldots, 2t_{\mathsf{c}})$ computes the hashed values of $\{[\![r_j]\!]_i^{\mathsf{B}}, [\![r_j]\!]_i\}_{j=0}^{n-1}, h_{i''',i}$ $(i''' = t_{\mathsf{c}}+1, \ldots, 2t_{\mathsf{c}})$, and send it to each P_i. // 1 round & $(t_{\mathsf{c}}+1)(2t_{\mathsf{p}}+1)nk + (t_{\mathsf{c}}+1)(2t_{\mathsf{p}}+1)n$ bits
11: Let each $R_{i'',i}$ $(i'' = 0, \ldots, t_{\mathsf{c}})$ be the set of shares $\{[\![r_j]\!]_i^{\mathsf{B}}, [\![r_j]\!]_i\}_{j=0}^{n-1}$ that is sent to P_i from $H_{i''}$. Each P_i computes the hashed value of $R_{i'',i}, h_{i'',i}$.
12: Each P_i outputs $R_i \in \{R_{i'',i}\}_{i''=0}^{t_{\mathsf{c}}}$ as the correct shares $\{[\![r_j]\!]_i^{\mathsf{B}}, [\![r_j]\!]_i\}_{j=0}^{n-1}$ if $t_{\mathsf{c}}+1$ or more of the hashed values in $\{h_{i',i}\}_{i'=0}^{H-1}$ match the hashed value of $R_{i',i}$ $(i' = 0, \ldots, H-1)$.

Protocol 7 $[\![x]\!] \leftarrow \pi_{\mathsf{rMultiBitComp}}^{\mathcal{F}_{\mathsf{rMultiRndGen}}}(\{[\![x_j]\!]^{\mathsf{B}}\}_{j=0}^{k-1})$

Input: $\{[\![x_j]\!]^{\mathsf{B}}\}_{j=0}^{k-1}$ (where $x_j \in \mathbb{Z}_2$)
Output: $[\![x]\!]$ *s.t.* $x = \sum_{j=0}^{k-1} 2^j \cdot x_j \mod 2^k$

1: (Offline phase)
2: All the parties and clients invoke $\mathcal{F}_{\mathsf{rMultiRndGen}}$ where $n = k$, then get $\{[\![r_j]\!]^{\mathsf{B}}, [\![r_j]\!]\}_{j=0}^{k-1}$. // $\pi_{\mathsf{rMultiRndGen}}$ requires 1 round & $(t_{\mathsf{c}}+1)(2t_{\mathsf{p}}+1)k^2 + (t_{\mathsf{c}}+1)(2t_{\mathsf{p}}+1)k$ bits
3: (Online phase)
4: Each P_i $(i = 0, \ldots, N-1)$ computes $[\![c_j]\!]_i^{\mathsf{B}} = [\![x_j \oplus r_j]\!]_i^{\mathsf{B}} = [\![x_j]\!]_i^{\mathsf{B}} \oplus [\![r_j]\!]_i^{\mathsf{B}}$ for $j = 0, \ldots, k-1$.
5: Each P_i $(i = 0, \ldots, N-1)$ gets the value $c_j \leftarrow \pi_{\mathsf{mbo}}([\![c_j]\!]^{\mathsf{B}})$ for $j = 0, \ldots, k-1$ in parallel. // 1 round & $N(N-(2t_{\mathsf{p}}+1))(t_{\mathsf{p}}+1)k$ bits
6: $[\![x]\!] = \sum_{j=0}^{k-1} 2^j \cdot (c_j + [\![r_j]\!] - 2 \cdot c_j \cdot [\![r_j]\!]) \mod 2^k$.

Let H and t_{c} be the number of clients and the corruption in the clients. As a modification of Protocol 4, we propose a modified construction with an arbitrary number of parties and clients, Protocol 7, where $t_{\mathsf{p}}(2t_{\mathsf{p}}+1) < N$ and $2t_{\mathsf{c}}+1 < H$. We assume that all clients have seed. We also assume that all clients do not collude with a party as well as the existing client-aided protocols [18], [19], [20].

Since there are no restrictions on the number of parties and clients, Protocol 7 is available for a wider range of situations than Protocol 4.

The computation strategy of Protocol 7 is almost the same as Protocol 4 except that it relies on $(N-2t_{\mathsf{p}}, N)$-RSS, not $(2,4)$-RSS. The proof strategy is also the same as Section 4. Therefore,

Protocol 7 computes $\mathcal{F}_{\mathsf{rMultiBitComp}}$ with computational security if no client colludes with parties and it holds that $t_{\mathsf{p}}(2t_{\mathsf{p}}+1) < N$ and $2t_{\mathsf{c}}+1 < H$.

3.2 Client-aided Secure Hamming Distance Calculation Protocol with Private Robustness Independent of Statistical Parameter

3.2.1 Protocol

Protocol 8 $[\sum_{j=0}^{m-1}(x_j \oplus y_j)] \leftarrow \pi_{\mathsf{rHD}}^{\mathcal{F}_{\mathsf{rRndGen}}}(\{[x_j]^{\mathsf{B}}, [y_j]^{\mathsf{B}}\}_{j=0}^{m-1})$

Input: $\{[x_j]^{\mathsf{B}}, [y_j]^{\mathsf{B}}\}_{j=0}^{m-1}$ (where $x_j, y_j \in \mathbb{Z}_2$ and $m < 2^k$)
Output: $[\mathsf{dist}]$ *s.t.* $\mathsf{dist} = \sum_{j=0}^{m-1}(x_j \oplus y_j) \mod 2^k$

1: (Offline phase)
2: P_0, P_1, P_2, P_3, H_0, H_1 and H_2 invoke $\mathcal{F}_{\mathsf{rRndGen}}$ where $n = m$, then get $\{[r_j]^{\mathsf{B}}, [r_j]\}_{j=0}^{m-1}$. // π_{rRndGen} `requires` 1 `round &` $24mk + 24m$ `bits`
3: (Online phase)
4: Each P_i computes $[z_j]_i^{\mathsf{B}} = [x_j \oplus y_j]_i^{\mathsf{B}} = [x_j]_i^{\mathsf{B}} \oplus [y_j]_i^{\mathsf{B}}$ for $j = 0, \ldots, m-1$.
5: Each P_i computes $[c_j]_i^{\mathsf{B}} = [z_j \oplus r_j]_i^{\mathsf{B}} = [z_j]_i^{\mathsf{B}} \oplus [r_j]_i^{\mathsf{B}}$ for $j = 0, \ldots, m-1$.
6: Each P_i gets value $c_j \leftarrow \pi_{\mathsf{bo}}([c_j]^{\mathsf{B}})$ for $j = 0, \ldots, m-1$ in parallel. // 1 `round &` $8m$ `bits`
7: $[\mathsf{dist}] = \sum_{j=0}^{m-1}(c_j + [r_j] - 2 \cdot c_j \cdot [r_j]) \mod 2^k$.

FUNCTIONALITY 5 ($\mathcal{F}_{\mathsf{rHD}}$ - computing the share of the Hamming distance between $(x_0, \ldots, x_{m-1})$ and $(y_0, \ldots, y_{m-1})$ from $\{[x_j]^{\mathsf{B}}, [y_j]^{\mathsf{B}}\}_{j=0}^{m-1}$, where $x_j, y_j \in \{0,1\}$ and $m < 2^k$).
(1) $\mathcal{F}_{\mathsf{rHD}}$ receives shares $\{[x_j]^{\mathsf{B}}, [y_j]^{\mathsf{B}}\}_{j=0}^{m-1}$ from parties P_0, P_1, P_2, and P_3.
(2) $\mathcal{F}_{\mathsf{rHD}}$ reconstructs $x_j, y_j (j = 0, \ldots, m)$ and computes $\mathsf{dist} = \sum_{j=0}^{m-1}(x_j \oplus y_j) \mod 2^k$. Then, $\mathcal{F}_{\mathsf{rHD}}$ computes $[\mathsf{dist}]$ and sends it to parties P_0, P_1, P_2 and P_3.

Our secure Hamming distance calculation protocol $\pi_{\mathsf{rHD}}^{\mathcal{F}_{\mathsf{rRndGen}}}$ in Protocol 8 is based on $\pi_{\mathsf{rBitComp}}^{\mathcal{F}_{\mathsf{rRndGen}}}$. In the offline phase, $\mathcal{F}_{\mathsf{rRndGen}}$ (where $n = m$) is invoked in the same way as $\pi_{\mathsf{rBitComp}}^{\mathcal{F}_{\mathsf{rRndGen}}}$ (at Line 2 in Protocol 8). In the online phase, each P_i computes $[z_j]_i^{\mathsf{B}} = [x_j \oplus y_j]_i^{\mathsf{B}}$ (at Line 4). Then, each P_i gets masked value $c_j (\in \{0,1\}) = z_j \oplus r_j$ $(j = 0, \ldots, m-1)$ by using shares $\{[r_j]^{\mathsf{B}}\}_{j=0}^{m-1}$ and π_{bo} (at Lines 5 and 6 in Protocol 8). Finally, the parties remove mask r_j from c_j by computing $[x_j] = [c_j \oplus r_j] = (c_j - [r_j])^2 = c_j + [r_j] - 2 \cdot c_j \cdot [r_j] \mod 2^k$. Then, the parties output $[\mathsf{dist}] = [\sum_{j=0}^{m-1} z_j] = \sum_{j=0}^{m-1}(c_j + [r_j] - 2 \cdot c_j \cdot [r_j])$ (at Line 7 in Protocol 8). The security of our protocol is proved in the same way as for $\pi_{\mathsf{rBitComp}}^{\mathcal{F}_{\mathsf{rRndGen}}}$.

3.2.2 Application Setting

In biometric authentication services that use iris recognition, a user has the biometric template, the binary vector $\{x_j\}_{j=0}^{m-1}$. The servicer has the registered template, the binary vector $\{y_j\}_{j=0}^{m-1}$. We assume that the authentication of the user is successful if the Hamming distance $\sum_{j=0}^{m-1} x_j \oplus y_j$ is smaller than the decision threshold value d, which the servicer has. Note that the user and servicer do not want to reveal the vector to each other to ensure privacy and prevent information leakage.

We assume that the (honest) user and the (honest) servicer compute $\{[x_j]^{\mathsf{B}}\}_{j=0}^{m-1}$ and $\{[y_j]^{\mathsf{B}}\}_{j=0}^{m-1}$ and send them to the MPC servers (P_0, P_1, P_2 and P_3) running Protocol 8, respectively. Then, all the MPC servers send their share of the Hamming distance $[\mathsf{dist}]$ to the servicer. Then, the servicer reconstructs dist and selects two or more matching values as the correct output. Finally, the servicer checks whether dist is smaller than d and sends the result of the authentication to the user. In this way, our scheme can provide a secure authentication service that is robust against DoS attacks in the standard model.

3.2.3 Modification and Extension of Protocol

Protocol 9 $[\sum_{j=0}^{m-1}(x_j \oplus y_j)] \leftarrow \pi_{\mathsf{rHD5}}(\{[x_j]^{\mathsf{B}}, [y_j]^{\mathsf{B}}\}_{j=0}^{m-1})$

Input: $\{[x_j]^{\mathsf{B}}, [y_j]^{\mathsf{B}}\}_{j=0}^{m-1}$ (where $x_j, y_j \in \mathbb{Z}_2$ and $m < 2^k$)
Output: $[\mathsf{dist}]$ *s.t.* $\mathsf{dist} = \sum_{j=0}^{m-1}(x_j \oplus y_j) \mod 2^k$

1: (Offline phase)
2: H_0 generates $\{[r_j]^{\mathsf{B}}, [r_j]\}_{j=0}^{m-1}$ randomly and distributes it to the parties. // 1 `round &` $12mk + 12m$ `bits`
3: (Online phase)
4: Each P_i $(i = 0, \ldots, 3)$ computes $[z_j]_i^{\mathsf{B}} = [x_j \oplus y_j]_i^{\mathsf{B}} = [x_j]_i^{\mathsf{B}} \oplus [y_j]_i^{\mathsf{B}}$ for $j = 0, \ldots, m-1$.
5: Each P_i computes $[c_j]_i^{\mathsf{B}} = [z_j \oplus r_j]_i^{\mathsf{B}} = [z_j]_i^{\mathsf{B}} \oplus [r_j]_i^{\mathsf{B}}$ for $j = 0, \ldots, m-1$.
6: Each P_i gets value $c_j \leftarrow \pi_{\mathsf{bo}}([c_j]^{\mathsf{B}})$ for $j = 0, \ldots, m-1$ in parallel. // 1 `round &` $8m$ `bits`
7: $[\mathsf{dist}] = \sum_{j=0}^{m-1}(c_j + [r_j] - 2 \cdot c_j \cdot [r_j]) \mod 2^k$.

Protocol 10 $[\![\sum_{j=0}^{m-1}(x_j \oplus y_j)]\!] \leftarrow \pi_{\mathsf{rMultiHD}}^{\mathcal{F}_{\mathsf{rMultiRndGen}}}(\{[\![x_j]\!]^{\mathsf{B}}, [\![y_j]\!]^{\mathsf{B}}\}_{j=0}^{m-1})$

Input: $\{[\![x_j]\!]^{\mathsf{B}}, [\![y_j]\!]^{\mathsf{B}}\}_{j=0}^{m-1}$ (where $x_j, y_j \in \mathbb{Z}_2$ and $m < 2^k$)
Output: $[\![\mathsf{dist}]\!]$ *s.t.* $\mathsf{dist} = \sum_{j=0}^{m-1}(x_j \oplus y_j) \mod 2^k$

1: (Offline phase)
2: All the parties and clients invoke $\mathcal{F}_{\mathsf{rMultiRndGen}}$ where $n = m$, then get $\{[\![r_j]\!]^{\mathsf{B}}, [\![r_j]\!]\}_{j=0}^{m-1}$. // $\pi_{\mathsf{rMultiRndGen}}$ `requires` 1 `round &` $(t_{\mathsf{c}}+1)(2t_{\mathsf{p}}+1)mk + (t_{\mathsf{c}}+1)(2t_{\mathsf{p}}+1)m$ `bits`
3: (Online phase)
4: Parties compute $[\![z_j]\!]^{\mathsf{B}} = [\![x_j \oplus y_j]\!]^{\mathsf{B}} = [\![x_j]\!]^{\mathsf{B}} \oplus [\![y_j]\!]^{\mathsf{B}}$ for $j = 0, \ldots, m-1$.
5: Parties compute $[\![c_j]\!]^{\mathsf{B}} = [\![z_j \oplus r_j]\!]^{\mathsf{B}} = [\![z_j]\!]^{\mathsf{B}} \oplus [\![r_j]\!]^{\mathsf{B}}$ for $j = 0, \ldots, m-1$.
6: Each party gets value $c_j \leftarrow \pi_{\mathsf{mbo}}([\![c_j]\!]^{\mathsf{B}})$ for $j = 0, \ldots, m-1$ in parallel. // 1 `round &` $N(N-(2t_{\mathsf{p}}+1))(t_{\mathsf{p}}+1)m$ `bits`
7: $[\![\mathsf{dist}]\!] = \sum_{j=0}^{m-1}(c_j + [r_j] - 2 \cdot c_j \cdot [\![r_j]\!]) \mod 2^k$.

We can modify and extend Protocol 8 by using Protocols 5 and 7. Protocol 9 (i.e., the modified Hamming distance calculation protocol based on Protocol 5) requires only one client. Hence, the financial cost of the service with Protocol 9 is lower than with Protocol 8. In Protocol 10 (i.e., the extended protocol based on Protocol 7), there can be an arbitrary number of parties and clients. Therefore, Protocol 10 is useful for improving the system redundancy and high availability of services.

4. Conclusion

In this paper, we proposed the client-aided maliciously secure bit-composition protocol with GOD (private robustness independent of a statistical parameter) in the standard model. Our scheme simultaneously improves the efficiency of computing complex

functions and the security. We also proposed the secure Hamming distance protocol with GOD (private robustness independent of a statistical parameter) in the standard model by modifying our bit-composition protocol. Our Hamming distance protocol can help provide a secure iris recognition service that is robust against DoS attacks.

Acknowledgments This work was supported in part by JSPS KAKENHI Grant Number 20K11807.

References

[1] Aly, A., Orsini, E., Rotaru, D., Smart, N.P. and Wood, T.: Zaphod: Efficiently Combining LSSS and Garbled Circuits in SCALE, pp.33–44, ACM (2019).

[2] Araki, T., Barak, A., Furukawa, J., Lichter, T., Lindell, Y., Nof, A., Ohara, K., Watzman, A. and Weinstein, O.: Optimized Honest-Majority MPC for Malicious Adversaries - Breaking the 1 Billion-Gate Per Second Barrier, *IEEE Symposium on Security and Privacy*, pp.843–862, IEEE Computer Society (2017).

[3] Beaver, D.: Commodity-Based Cryptography (Extended Abstract), *STOC*, pp.446–455, ACM (1997).

[4] Ben-Or, M., Goldwasser, S. and Wigderson, A.: Completeness Theorems for Non-Cryptographic Fault-Tolerant Distributed Computation (Extended Abstract), *STOC*, pp.1–10, ACM (1988).

[5] Byali, M., Chaudhari, H., Patra, A. and Suresh, A.: FLASH: Fast and Robust Framework for Privacy-preserving Machine Learning, *Proc. Priv. Enhancing Technol.*, Vol.2020, No.2, pp.459–480 (2020).

[6] Byali, M., Hazay, C., Patra, A. and Singla, S.: Fast Actively Secure Five-Party Computation with Security Beyond Abort, *ACM Conference on Computer and Communications Security*, pp.1573–1590, ACM (2019).

[7] Byali, M., Joseph, A., Patra, A. and Ravi, D.: Fast Secure Computation for Small Population over the Internet, *ACM Conference on Computer and Communications Security*, pp.677–694, ACM (2018).

[8] Canetti, R.: Universally Composable Security: A New Paradigm for Cryptographic Protocols, *FOCS*, pp.136–145, IEEE Computer Society (2001).

[9] Chandran, N., Garay, J.A., Mohassel, P. and Vusirikala, S.: Efficient, Constant-Round and Actively Secure MPC: Beyond the Three-Party Case, *CCS*, pp.277–294, ACM (2017).

[10] Chaudhari, H., Choudhury, A., Patra, A. and Suresh, A.: ASTRA: High Throughput 3PC over Rings with Application to Secure Prediction, *CCSW@CCS*, pp.81–92, ACM (2019).

[11] Chaudhari, H., Rachuri, R. and Suresh, A.: Trident: Efficient 4PC Framework for Privacy Preserving Machine Learning, *NDSS*, The Internet Society (2020).

[12] Dalskov, A., Escudero, D. and Keller, M.: Fantastic Four: Honest-Majority Four-Party Secure Computation With Malicious Security, Cryptology ePrint Archive, Report 2020/1330 (2020).

[13] Furukawa, J., Lindell, Y., Nof, A. and Weinstein, O.: High-Throughput Secure Three-Party Computation for Malicious Adversaries and an Honest Majority, *EUROCRYPT* (*2*), Lecture Notes in Computer Science, Vol.10211, pp.225–255 (2017).

[14] Goldreich, O., Micali, S. and Wigderson, A.: How to Play any Mental Game or A Completeness Theorem for Protocols with Honest Majority, *STOC*, pp.218–229, ACM (1987).

[15] Gordon, S.D., Ranellucci, S. and Wang, X.: Secure Computation with Low Communication from Cross-Checking, *ASIACRYPT* (*3*), Lecture Notes in Computer Science, Vol.11274, pp.59–85, Springer (2018).

[16] Koti, N., Pancholi, M., Patra, A. and Suresh, A.: SWIFT: Super-fast and Robust Privacy-Preserving Machine Learning, Cryptology ePrint Archive, Report 2020/592 (2020).

[17] Mohassel, P. and Rindal, P.: ABY^3: A Mixed Protocol Framework for Machine Learning, *ACM Conference on Computer and Communications Security*, pp.35–52, ACM (2018).

[18] Mohassel, P. and Zhang, Y.: SecureML: A System for Scalable Privacy-Preserving Machine Learning, *IEEE Symposium on Security and Privacy*, pp.19–38, IEEE Computer Society (2017).

[19] Morita, H., Attrapadung, N., Teruya, T., Ohata, S., Nuida, K. and Hanaoka, G.: Constant-Round Client-Aided Secure Comparison Protocol, *ESORICS* (*2*), Lecture Notes in Computer Science, Vol.11099, pp.395–415, Springer (2018).

[20] Ohata, S. and Nuida, K.: Communication-Efficient (Client-Aided) Secure Two-Party Protocols and Its Application, *Financial Cryptography*, Lecture Notes in Computer Science, Vol.12059, pp.369–385, Springer (2020).

[21] Patra, A. and Suresh, A.: BLAZE: Blazing Fast Privacy-Preserving Machine Learning, *NDSS*, The Internet Society (2020).

[22] Tsuchida, H. and Nishide, T.: Client-Aided Bit-Composition Protocol with Guaranteed Output Delivery, *2020 International Symposium on Information Theory and Its Applications* (*ISITA*), IEEE (2020).

[23] Yao, A.C.: How to Generate and Exchange Secrets (Extended Abstract), *FOCS*, pp.162–167, IEEE Computer Society (1986).

Appendix

A.1 Sharing Protocol of $(N - 2t_{\mathsf{p}}, N)$-RSS

Protocol 11 $[\![x]\!] \leftarrow \pi_{\mathsf{share}}(x,\ P_\ell,\ \mathsf{vid}, F)$

Input: Input value $x \in \mathbb{Z}_2$, input dealer P_ℓ, unique identifier vid, pseudo-random function $F : \{0,1\}^\kappa \times \{0,1\}^\kappa \to \mathbb{Z}_2$

Output: $[\![x]\!]$

1: **for** $i = 0, \ldots, N-1$ **do in parallel**
2: Each party P_i computes $r_j = F(\mathsf{seed}_j,\ \mathsf{vid})$ where $r_j \in \mathbb{Z}_2$ for $j = i, \ldots, i + 2t_{\mathsf{p}}$.
3: Each party P_i sets $[\![r]\!]^{\mathsf{B}}_i = (r_j, \ldots, r_{j+2t_{\mathsf{p}}})$ where $r = r_0 \oplus \cdots \oplus r_{N-1} \mod 2$ and $j = i, \ldots, i + 2t_{\mathsf{p}}$.
4: **end for**
5: An input dealer P_ℓ generates random values $x_j \in \mathbb{Z}_2$ for $j = 1, \ldots, N-1$.
6: P_ℓ sets $x_0 = x \oplus x_1 \oplus \cdots \oplus x_{N-1} \mod 2$.
7: **for** $i = 0, \ldots, N-1$ **do in parallel**
8: P_ℓ sends $[\![x]\!]^{\mathsf{B}}_i = (x_i, \ldots, x_{i+2t_{\mathsf{p}}})$ to P_i `// 1 round & (N − 1)(2t_p + 1) bits`
9: **end for**
10: $[\![x \oplus r]\!]^{\mathsf{B}} = [\![x]\!]^{\mathsf{B}} \oplus [\![r]\!]^{\mathsf{B}}$
11: All parties run $\pi_{\mathsf{mbo}}([\![x \oplus r]\!]^{\mathsf{B}})$ and gets $x \oplus r$. `// 1 round & N(N − (2t_p + 1))(t_p + 1) bits`
12: All parties exchange their $(x \oplus r)$ and choose the correct $(x \oplus r)$ by the majority voting. If it is not possible to determine the correct value by majority voting, then P_ℓ and its initial input values are removed.
13: $[\![x]\!]^{\mathsf{B}} = (x \oplus r) \oplus [\![r]\!]^{\mathsf{B}}$

We assume that each P_i $(i = 0, \ldots, N-1)$ obtains $(\mathsf{seed}_i, \ldots, \mathsf{seed}_{i+2t_{\mathsf{p}}})$ where $\mathsf{seed}_i \in \{0,1\}^\kappa$. We also assume that $(\mathsf{seed}_i, \ldots, \mathsf{seed}_{i+2t_{\mathsf{p}}})$ are given to each P_i only once by a trusted third party or MPC-based random value generation during the initialization process.

We describe the sharing protocol of $(N - 2t_{\mathsf{p}}, N)$-RSS on $\mathbb{Z}_2$ in Protocol 11. From Lines 1 to 4 in Protocol 11, each party computes the shares of random bit r, $[\![r]\!]^{\mathsf{B}}$ to mask the input value. Then, the input dealer P_ℓ computes the shares of input value $[\![x]\!]^{\mathsf{B}}$ and sends $[\![x]\!]^{\mathsf{B}}_i$ to P_i from Lines 5 to 9. After that, all parties compute $[\![x \oplus r]\!]^{\mathsf{B}}$ at Line 10 and get $(x \oplus r)$ by π_{mbo} at Line 11. At Line 12, all parties exchange $(x \oplus r)$ and choose the correct $(x \oplus r)$ by majority voting. However, if P_ℓ is the corrupted party, P_ℓ can cause so much corruptions that $t_{\mathsf{p}}(2t_{\mathsf{p}} + 1) < N$ does not hold, indirectly, by sending the incorrect different values to each party in the previous Lines. Hence, if the majority voting does not work well, the parties except P_ℓ identify P_ℓ as the corrupted party and remove it and its initial inputs. Finally, all parties compute $[\![x]\!]^{\mathsf{B}} = (x \oplus r) \oplus [\![r]\!]^{\mathsf{B}}$ at Line 13 and get the shares of the input value $[\![x]\!]^{\mathsf{B}}$.

Hikaru Tsuchida received B.S. degree from the Tokyo University of Science in 2014 and M.S. degree from the University of Tsukuba in 2016. From 2016, he works as a researcher at NEC Corporation. From 2020, he is a Ph.D. student at University of Tsukuba. His research is in the areas of cryptography and information security.

Takashi Nishide received B.S. degree from the University of Tokyo in 1997, M.S. degree from the University of Southern California in 2003, and Dr.E. degree from the University of Electro-Communications in 2008. From 1997 to 2009, he had worked at Hitachi Software Engineering Co., Ltd. developing security products. From 2009 to 2013, he had been an assistant professor at Kyushu University and from 2013 he is an associate professor at University of Tsukuba. His research is in the areas of cryptography and information security.