Secure Multiparty Computation of Symmetric Functions with Polylogarithmic Bottleneck Complexity and Correlated Randomness*

Reo Eriguchi

National Institute of Advanced Industrial Science and Technology, Tokyo, Japan
eriguchi-reo@aist.go.jp

Abstract. Bottleneck complexity is an efficiency measure of secure multiparty computation (MPC) protocols introduced to achieve load-balancing in large-scale networks, which is defined as the maximum communication complexity required by any one player within the protocol execution. Towards the goal of achieving low bottleneck complexity, prior works proposed MPC protocols for computing symmetric functions in the correlated randomness model, where players are given input-independent correlated randomness in advance. However, the previous protocols with polylogarithmic bottleneck complexity in the number of players n require a large amount of correlated randomness that is linear in n, which limits the per-party efficiency as receiving and storing correlated randomness are the bottleneck for efficiency. In this work, we present for the first time MPC protocols for symmetric functions such that bottleneck complexity and the amount of correlated randomness are both polylogarithmic in n, assuming collusion of size at most $n - o(n)$ players. Furthermore, one of our protocols is even computationally efficient in that each player performs only $\mathrm{polylog}(n)$ arithmetic operations while the computational complexity of the previous protocols is $O(n)$. Technically, our efficiency improvements come from novel protocols based on ramp secret sharing to realize basic functionalities with low bottleneck complexity, which we believe may be of interest beyond their applications to secure computation of symmetric functions.

1 Introduction

Secure multiparty computation (MPC) [49] is a fundamental cryptographic primitive which enables n players to jointly compute a function $f(x_1, \dots, x_n)$ without revealing information on their private inputs x_i to adversaries corrupting at most t players. Due to many important applications, the asymptotic and concrete optimization of MPC protocols has been the subject of a large body of research. In this work, we consider the *dishonest-majority* setting, where the majority of players are corrupted, i.e., $t > n/2$.

* The preliminary version of this paper appears in the proceedings of ITC 2024 [22]. Since we found a security flaw in the original constructions, we show fixed ones in the current version (see Section 1.3 for details).

MPC in the correlated randomness model. A popular approach to designing MPC protocols in the dishonest-majority setting is to employ *correlated randomness*. In this model, players receive correlated randomness from a trusted dealer before inputs are known (the offline phase) and then consume the randomness to perform input-dependent computation (the online phase). It was shown in [1] that the correlated randomness allows us to construct information-theoretically secure protocols in the dishonest-majority setting, while such protocols do not exist in the plain model. Subsequently, many optimizations have been proposed and several of them are even implemented [6, 19, 38, 18, 8, 9]. Two primary efficiency metrics for MPC in this model are the *online communication* cost and the amount of *correlated randomness* received from a trusted dealer [13, 9]. This is because as opposed to local computation, communication and storage costs are usually dominant in MPC protocols and minimizing both costs simultaneously leads to fast and scalable protocols.

Bottleneck complexity. Traditionally, the cost of online communication has been measured by the *total* amount of communication across all n players. On the other hand, for practical applications such as peer-to-peer computations between lightweight devices, the *per-party* cost is a more effective measure than the total cost. For example, several existing protocols (e.g., [17, 14, 30, 23]) require one player to communicate different messages with every other player. Then, while the total communication cost is possibly scalable, the player must bear communication proportional to n and his cost quickly becomes prohibitive in large-scale MPC involving many players. In this work, we focus on a more fine-grained efficiency measure capturing the load-balancing aspect of protocols, called *bottleneck complexity* [10], which is defined as the maximum communication required by any one player during the protocol execution.

To fit large-scale networks, we aim at designing MPC protocols whose bottleneck complexity scales polylogarithmically with n. Unfortunately, there is a negative result that we cannot achieve sublinear bottleneck complexity for *all* functions even without any security considerations [10]. Due to this result, a line of works [44, 40, 21] have studied the problem of constructing protocols with low bottleneck complexity for specific classes of functions. Above all, the class of *symmetric functions*, whose values are the same no matter the order of n inputs, is one of the most fundamental functions including majority, counting, and parity functions. Recently, the authors of [40, 21] constructed information-theoretic protocols for symmetric functions with $O(\log n)$ bottleneck complexity[1]. However, a main drawback of the protocols is that every player needs to receive a large amount of correlated randomness that is linear in n per party. This means that no matter how much bottleneck complexity in the online phase is improved, the protocols do not work efficiently as receiving and storing correlated randomness is the bottleneck for efficiency. Motivated by the above considerations, in this work, we ask:

[1] The authors of [40] considered a related class of functions called *abelian programs*. Their protocol can also compute symmetric functions by setting the underlying abelian group as the ring of integers modulo $n + 1$. See Remark 1 for more details.

Can we construct MPC protocols for symmetric functions keeping both bottleneck complexity and the amount of correlated randomness polylogarithmic in n?

1.1 Our Results

In this work, we answer the above question affirmatively by presenting two different constructions of MPC protocols for symmetric functions, assuming semi-honest adversaries colluding with at most $n - o(n)$ players.

Theorem 1 (Informal). *For a parameter ℓ, there exists an information-theoretic MPC protocol for computing a symmetric function $f : \{0,1\}^n \to \{0,1\}$ that has bottleneck complexity $O(\log n)$, per-party correlated randomness of size $O(\ell \log n)$, and tolerates up to $n - \Theta(n/\ell)$ semi-honest corruptions.*

A typical choice of the parameter ℓ is $\ell = \Theta(\log n)$. Theorem 1 then gives a protocol that has bottleneck complexity $O(\log n)$ and correlated randomness of size $O((\log n)^2)$. Compared to the previous works, the protocol achieves for the first time polylog(n) bottleneck complexity and correlated randomness simultaneously (see Table 1). Furthermore, if we set $\ell \approx 1/\epsilon$ for a constant $0 < \epsilon < 1/2$, then Theorem 1 gives a protocol such that both the bottleneck complexity and the amount of correlated randomness are $O(\log n)$ for a constant fraction of corrupted players (e.g., 99 percent of the parties are corrupted). Although the corruption threshold t is lower than the maximum $n - 1$, our protocol is still secure in the dishonest majority setting $t > n/2$. A more detailed comparison is shown in Table 1. Our protocol even achieves polylog(n) computational complexity since the local computation of each player involves only $O(\log n)$ arithmetic operations in a field of size $O(n)$. As a comparison, the previous protocols in [40, 21] have $O(n)$ computational complexity since every player needs to process vectors or matrices of size $O(n)$.

We also show another construction of protocols for symmetric functions with polylogarithmic bottleneck complexity and correlated randomness. Compared with the first construction, it reduces the amount of correlated randomness excluding correlated randomness required to securely compute the sum of elements in an abelian group. If the secure summation functionality is implemented with the state-of-the-art protocol in [21], the construction derives a protocol with bottleneck complexity $O((\log n)^2)$ and correlated randomness of size $O((\log n)^2)$ but there is no advantage of efficiency compared to the one obtained from the first construction. If there exists a secure summation protocol over an abelian group $\mathbb{G}$ with bottleneck complexity $o(\log |\mathbb{G}|)$ and correlated randomness of size $O(\log |\mathbb{G}|)$, then our second construction implies a protocol that achieves bottleneck complexity $O((\log n)^2)$ and a smaller amount of correlated randomness $o((\log n)^2)$. Unfortunately, it is currently unknown whether such a secure summation protocol exists, which we leave for future work.

Technically, we achieve polylogarithmic bottleneck complexity and correlated randomness with the help of ramp secret sharing [43, 7, 48, 24] (also known as packed secret sharing), a technique to distribute and operate on multiple secrets simultaneously only paying the cost of a single secret. This tool was used to realize certain functionalities in several previous works [14, 23], but they required every player to distribute fresh shares of their local secret, which leads to inefficient protocols in terms of bottleneck complexity. Our technical novelty is carefully designing correlated randomness to avoid such resharing processes and keep polylog(n) bottleneck complexity. See Section 2 for a detailed overview of our techniques.

Table 1. Information-theoretic MPC protocols for computing symmetric functions with sublinear bottleneck complexity in the dishonest-majority setting

Reference	BC	CR	Corruption
[40, 21]	$O(\log n)$	$O(n)$	$n-1$
[21]	$O(\sqrt{n})$	$O(\sqrt{n})$	$n-1$
Ours (Corollary 2)	$O(\log n)$	$O((\log n)^2)$	$n-o(n)$
Ours (Corollary 3)	$O(\log n)$	$O(\log n)$	$(1-\epsilon)n$

"BC" stands for bottleneck complexity and "CR" stands for the amount of correlated randomness per party. ϵ is any constant with $0 < \epsilon < 1/2$.

1.2 Related Work

Boyle et al. [10] constructed a generic compiler from any possibly insecure protocol to a computationally secure protocol (without correlated randomness) preserving bottleneck complexity up to a polynomial factor in a security parameter. However, their compiler is based on fully homomorphic encryption, which can only be instantiated from a narrow class of cryptographic assumptions [26, 47, 27], and the concrete efficiency leaves much to be desired. Orlandi, Ravi, and Scholl [44] constructed a protocol for symmetric functions in the correlated randomness model assuming garbled circuits. However, in addition to not achieving information-theoretic security, players need to receive a garbled circuit with $O(\log n)$ input bits as correlated randomness. Since the minimum size of circuits computing a worst-case function with m input bits is $\Omega(2^m/m)$ [39], the correlated randomness of [44] is $\Omega(\lambda n/\log n)$ in the worst case, which is not polylogarithmic in n. There are maliciously secure protocols with sublinear bottleneck complexity for general tasks [20] and specific tasks [42, 25]. However, these protocols assume the strong honest-majority setting ($t < n/3$) and/or only achieve $\Omega(\sqrt{n})$ bottleneck complexity.

There is a rich line of works studying total communication complexity of MPC, e.g., [28, 4, 11, 45, 15, 36, 35, 17, 2, 16, 19, 5, 37, 12, 31, 32, 41, 30]. However, protocols in all of the above works require *full interaction* among players, that is, each player may send different messages to all the other players in each round of interaction. This feature necessarily results in high bottleneck complexity $\Omega(n)$.

The authors of [34, 33] initiated the study of the communication complexity of MPC with restricted interaction patterns. Halevi et al. [33] studied a chain-based interaction, in which players interact over a simple directed path traversing all players. Protocols on a chain-based interaction possibly achieve low bottleneck complexity since each player communicates with at most two players. However, since the last player on the chain is allowed to evaluate the function on every possible input of his choice, the constructions in [33] cannot achieve the standard security of MPC, which requires that corrupted players learn nothing but the output.

1.3 Publication Note

The preliminary version of this paper appears in the proceedings of ITC 2024 [22]. Although we proposed a protocol to compute the sum of group elements in the preliminary version, we found a security flaw in the protocol and consequently, our main protocols for symmetric functions did not satisfy the specified security (see Appendix A for details). In the current version, we show fixed protocols in which the summation protocol is replaced with a different one in [21]. Our first construction (Theorem 1) still achieves the same asymptotic performance as the original construction. On the other hand, our second construction needs a larger amount of correlated randomness and currently, there is no advantage of efficiency compared to the one obtained from Theorem 1. As we mentioned in Section 1.1, it is still possible that our second construction has an advantage over the first one if a secure summation protocol with a smaller amount of correlated randomness is devised.

2 Technical Overview

In this section, we provide an overview of our techniques. More detailed descriptions and security proofs will be given in the following sections.

2.1 Our First Protocol for Symmetric Functions

To begin with, we recall the protocol computing symmetric functions with $O(\log n)$ bottleneck complexity in [40, 21]. Let $h : \{0,1\}^n \to \{0,1\}$ be a symmetric function. Since the value of $h(x_1, \ldots, x_n)$ depends only on the number of 1's, which is equal to the sum $\sum_{i\in[n]} x_i$, there is the unique function $f : \{0,1,\ldots,n\} \to \{0,1\}$ such that $h(x_1,\ldots,x_n) = f(\sum_{i\in[n]} x_i)$. Roughly speaking, the protocol in [40, 21] proceeds as follows: In the setup, players receive an additive sharing of the truth-table $\mathbf{T}_r \in \{0,1\}^{n+1}$ of f permuted with a random shift $r \in \{0,1,\ldots,n\}$. Simultaneously, they also receive an additive sharing $(r_i)_{i\in[n]}$ of the shift r. In the online phase, players compute x_i+r_i, open $y = \sum_{i\in[n]} x_i+r$, and then open the y-th component of the permuted truth-table $\mathbf{T}_r$, which is $f(y-r) = h(x_1,\ldots,x_n)$.

In this protocol, however, players need to receive additive shares of the $(n+1)$-dimensional vector $\mathbf{T}_r$, which results in correlated randomness of size $O(n)$ per party.

Our starting point to reduce this large correlated randomness is using a *ramp secret sharing scheme* to share the permuted truth-table $\mathbf{T}_r$ of f. Ramp secret sharing [43, 7, 48] is a variant of secret sharing which can share a secret vector of dimension k keeping the share size logarithmic in k and n. One may expect that a ramp secret sharing scheme can compress the $(n+1)$-dimensional vector $\mathbf{T}_r$ into shares each of size logarithmic in n. However, this falls short of achieving our goal since the efficiency of ramp secret sharing schemes comes at the cost of decreasing a privacy threshold t to $n-k$. In our setting, this means that when sharing the $(n+1)$-dimensional vector $\mathbf{T}_r$, we need to set a privacy threshold $t = n - (n+1) < 0$, which guarantees no privacy.

To overcome this, we decompose the permuted truth-table $\mathbf{T}_r$ into ℓ vectors $\mathbf{T}_r = (\mathbf{U}^{(0)}, \mathbf{U}^{(1)}, \ldots, \mathbf{U}^{(\ell-1)})$ each of dimension $k = (n+1)/\ell$. We independently generate shares of each vector $\mathbf{U}^{(j)}$ using a ramp secret sharing scheme. Now, a privacy threshold is $t = n - (n+1)/\ell = n - o(n)$ instead of $t = n - (n+1)$. In the online phase, players write $y = x + r$ as $y = \sigma k + \tau$ for some $\sigma \in \{0, 1, \ldots, \ell-1\}$ and $\tau \in \{0, 1, \ldots, k-1\}$, which implies that the y-th component of $\mathbf{T}_r$ corresponds to the τ-th component of $\mathbf{U}^{(\sigma)}$. Then all players can together reconstruct the output $f(y-r) = h(x_1, \ldots, x_n)$ by opening the τ-th component of $\mathbf{U}^{(\sigma)}$. A typical choice of the parameter ℓ is $\ell = \Theta(\log n)$. Then a privacy threshold is $t = n - \Theta(n/\log n)$ and correlated randomness for each player consists of $O(\ell) = O(\log n)$ shares. Since a ramp secret sharing scheme requires the underlying field to contain $n + k = O(n)$ elements, the size of correlated randomness is $O((\log n)^2)$ in bits. Note that the bottleneck complexity is still $O(\log n)$ since players open only one share in the online phase. On the other hand, if we set $\ell \approx 1/\epsilon$ for a constant $0 < \epsilon < 1/2$, then both the bottleneck complexity and the amount of correlated randomness are $O(\log n)$ while the number of corrupted players should be at most $(1-\epsilon)n$.

2.2 Our Second Protocol for Symmetric Functions

Next, we show another protocol for symmetric functions that achieves polylogarithmic bottleneck complexity and correlated randomness. Compared with the first construction, it reduces the amount of correlated randomness excluding correlated randomness required to securely compute the sum of elements in an abelian group.

Our starting point is a balancing approach in [21] of expressing the truth-table of $f : \{0, 1, \ldots, n\} \to \{0, 1\}$ (induced by a symmetric function h) as a matrix $\mathbf{M}_f$ instead of an $(n+1)$-dimensional vector. More specifically, assume that there are two distinct primes ℓ and k such that $\ell k = n+1$, and fix the one-to-one correspondence ϕ between $\mathbb{Z}_{n+1} = \{0, 1, \ldots, n\}$ and $\mathbb{Z}_\ell \times \mathbb{Z}_k = \{(\sigma, \tau) \in \mathbb{Z}^2 : 0 \le \sigma < \ell,\ 0 \le \tau < k\}$ induced by the Chinese remainder theorem. Then there exists a matrix $\mathbf{M}_f \in \{0,1\}^{\ell \times k}$ such that the computation of $f(\sum_{i \in [n]} x_i)$

can be expressed as the following inner product

$$f(\sum_{i\in[n]} x_i) = \langle \mathbf{e}_\sigma, \mathbf{M}_f \cdot \mathbf{e}_\tau \rangle, \tag{1}$$

where $(\sigma, \tau) = \phi(\sum_{i\in[n]} x_i)$ and $\mathbf{e}_j$ denotes the vector with a 1 in the j-th coordinate and 0's elsewhere. The task is now reduced to secure computation of matrix-vector products of size at most $\max\{\ell, k\}$, which balances bottleneck complexity and the amount of correlated randomness. However, if we naively implement secure computation of the inner product (1) by sharing secret vectors $\mathbf{e}_\sigma$ and $\mathbf{e}_\tau$ in an element-wise way, then the best possible bottleneck complexity is $\Omega(\sqrt{n})$ since the primes ℓ, k should satisfy $\ell k = \Omega(n)$.

To achieve polylogarithmic bottleneck complexity, we use a ramp secret sharing scheme and encode secret vectors $\mathbf{e}_\sigma$ and $\mathbf{e}_\tau$ into small shares. This reduces the secure computation of (1) to constructing protocols for the following functionalities:

Linear transformation. Players obtain ramp shares of an ℓ-dimensional vector $\mathbf{M} \cdot \mathbf{w}$ from shares of a k-dimensional secret vector $\mathbf{w}$, where $\mathbf{M}$ is a public ℓ-by-k matrix.

Inner product. Players obtain $\langle \mathbf{v}, \mathbf{w} \rangle$ from ramp shares of two ℓ-dimensional vectors $\mathbf{v}$ and $\mathbf{w}$.

We note that a protocol for the first functionality was previously considered in [14] but it requires every player to reshare their local shares, which results in $\Omega(n)$ bottleneck complexity. Our technical novelty is carefully designing correlated randomness to avoid such resharing processes and keep bottleneck complexity polylogarithmic in n.

Linear transformation. Ramp secret sharing schemes considered in this paper have linear reconstruction, that is, a secret vector can be expressed as a linear combination of all shares over a field. This implies that given shares of $\mathbf{w}$, every player can locally compute an ℓ-dimensional vector $\mathbf{s}_i$ such that $\mathbf{s}_1 + \cdots + \mathbf{s}_n = \mathbf{M} \cdot \mathbf{w}$. If players were allowed to reshare all the $\mathbf{s}_i$'s, they could securely obtain shares of $\mathbf{M} \cdot \mathbf{w}$. However, the resharing of all the $\mathbf{s}_i$'s results in high bottleneck complexity $\Omega(n)$. Instead, we distribute shares of a randomly chosen ℓ-dimensional vector $\mathbf{r}$ in the offline phase. This enables players to locally compute $\mathbf{x}_i$ such that $\mathbf{x}_1 + \cdots + \mathbf{x}_n = \mathbf{M} \cdot \mathbf{w} + \mathbf{r}$ and jointly reconstruct $\mathbf{M} \cdot \mathbf{w} + \mathbf{r}$, which can be done by communicating $O(\ell)$ field elements. Note that since $\mathbf{r}$ is unknown to any player, $\mathbf{M} \cdot \mathbf{w} + \mathbf{r}$ is just a random vector. It can be done locally to obtain shares of $\mathbf{M} \cdot \mathbf{w} + \mathbf{r}$ from it. Players then convert these shares into the ones of $\mathbf{M} \cdot \mathbf{w}$ by subtracting the shares of $\mathbf{r}$. In our protocol, players communicate only $O(\ell)$ field elements in the online phase and receive a constant number of field elements in the offline phase, excluding correlated randomness required to securely compute the sum of elements.

Inner product. Distributing Beaver triples [1] in the offline phase is a common technique to compute the product vw from shares of two secrets v and

w. Although this technique successfully works when computing the product of *scalars*, a naive generalization does not work if we compute the inner product of *vectors* shared by a ramp scheme. More specifically, a common template using Beaver triples is distributing fresh shares of three secrets a, b and c in the offline phase, where a and b are randomly chosen and $c = ab$. In the online phase, players reconstruct $v - a$ and $w - b$, and then compute shares of vw based on the equation

$$vw = (v - a)(w - b) + a(w - b) + b(v - a) + c.$$

This can be done locally since vw is a linear combination of secrets a, b and c with public coefficients $v-a$ and $w-b$. To generalize this template, we distribute shares of secret vectors $\mathbf{a}$, $\mathbf{b}$ and $\mathbf{c}$, where $\mathbf{a}$ and $\mathbf{b}$ are random and $\mathbf{c} = \mathbf{a} * \mathbf{b}$, where $*$ denotes the element-wise product. As above, players reconstruct $\mathbf{v} - \mathbf{a}$ and $\mathbf{w} - \mathbf{b}$. Naturally, we extends the above equation to vectors:

$$\mathbf{v} * \mathbf{w} = (\mathbf{v} - \mathbf{a}) * (\mathbf{w} - \mathbf{b}) + \mathbf{a} * (\mathbf{w} - \mathbf{b}) + \mathbf{b} * (\mathbf{v} - \mathbf{a}) + \mathbf{c}.$$

It is easy to compute shares of the first term since $\mathbf{v} - \mathbf{a}$ and $\mathbf{w} - \mathbf{b}$ are public. A technical difficulty lies in computing shares of the second and third terms. When we only deal with scalars, players can locally compute shares of $a(w - b)$ from shares of a and a public constant $w - b$ just by multiplying the shares by the constant. However, when a secret vector $\mathbf{a}$ is shared by a ramp scheme, multiplying shares of $\mathbf{a}$ by a constant d results in shares of a vector $d \cdot \mathbf{a}$, whose entries are *all* multiplied by d. To obtain shares of $\mathbf{a} * (\mathbf{w} - \mathbf{b})$, we need to multiply different entries of a secret vector $\mathbf{a}$ by different constants. For that, we rewrite $\mathbf{a} * (\mathbf{w} - \mathbf{b}) = \mathrm{diag}(\mathbf{w} - \mathbf{b}) \cdot \mathbf{a}$ and apply the above protocol for linear transformation with $\mathbf{M} = \mathrm{diag}(\mathbf{w} - \mathbf{b})$, where $\mathrm{diag}(\mathbf{w} - \mathbf{b})$ denotes a diagonal matrix whose (i, i)-th entry is the i-th entry of $\mathbf{w} - \mathbf{b}$. Finally, players obtain shares of $\mathbf{v} * \mathbf{w}$, jointly reconstruct it, and output $\langle \mathbf{1}, \mathbf{v} * \mathbf{w} \rangle = \langle \mathbf{v}, \mathbf{w} \rangle$, where $\mathbf{1}$ is the all-one vector. Since naively reconstructing $\mathbf{v} * \mathbf{w}$ leaks additional information, we let players add shares of a random secret $\mathbf{s}$ such that $\langle \mathbf{1}, \mathbf{s} \rangle = 0$, which does not affect correctness since $\langle \mathbf{1}, \mathbf{v} * \mathbf{w} + \mathbf{s} \rangle = \langle \mathbf{v}, \mathbf{w} \rangle$. In this protocol, players communicate only $O(\ell)$ field elements in the online phase and receive a constant number of field elements in the offline phase, excluding correlated randomness required to securely compute the sum of elements.

Putting it altogether. Similarly to our first protocol, in the offline phase, we distribute additive shares of a random mask $r \in \{0, 1, \dots, n\}$ and ramp shares of vectors $\mathbf{e}_{\sigma_r}$ and $\mathbf{e}_{\tau_r}$, where $\phi(r) = (\sigma_r, \tau_r) \in \mathbb{Z}_\ell \times \mathbb{Z}_k$. In the online phase, players open a masked sum $y = \sum_{i \in [n]} x_i - r$ and compute $\phi(y) = (\sigma_y, \tau_y)$. Note that $(\sigma_y + \sigma_r, \tau_y + \tau_r) = \phi(\sum_{i \in [n]} x_i) = (\sigma, \tau)$. Then, players obtain ramp shares of $\mathbf{e}_\sigma$ by applying the protocol for linear transformation with $\mathbf{w} = \mathbf{e}_{\sigma_r}$ and $\mathbf{M}$ being the linear operation of shifting a vector by σ_y. Similarly, players run the linear transformation protocol on ramp shares of $\mathbf{e}_{\tau_r}$ to obtain shares of $\mathbf{e}_\tau$. Subsequently, they apply the linear transformation protocol setting $\mathbf{w} = \mathbf{e}_\tau$ and $\mathbf{M} = \mathbf{M}_f$ to obtain ramp shares of $\mathbf{M}_f \cdot \mathbf{e}_\tau$. Finally, they run the inner product protocol on input $\mathbf{e}_\sigma$ and $\mathbf{M}_f \cdot \mathbf{e}_\tau$, and obtain $\langle \mathbf{e}_\sigma, \mathbf{M}_f \cdot \mathbf{e}_\tau \rangle = f(\sum_{i\in[n]} x_i) = h(x_1, \ldots, x_n)$. A typical choice of the primes ℓ and k is $\ell = \Theta(\log n)$ and $k = \Theta(n/\log n)$. Since a ramp secret sharing scheme requires a field of size $O(n)$, a field element can be described in $O(\log n)$ bits. Therefore, the bottleneck complexity of our final protocol is $O(\ell \log n) = O((\log n)^2)$ and the per-party correlated randomness is $O(\log n)$ bits, excluding correlated randomness required to securely compute the sum of elements. On the other hand, a privacy threshold is $t = n - \max\{\ell, k\} = n - \Theta(n/\log n)$ since ℓ-dimensional and k-dimensional secret vectors are shared by a ramp scheme.

3 Preliminaries

3.1 Notations

For $m \in \mathbb{N}$, define $[m] = \{1, \ldots, m\}$. Define $\mathbb{Z}_m$ as the ring of integers modulo m. We identify $\mathbb{Z}_m$ (as a set) with $\{z \in \mathbb{Z} : 0 \leq z \leq m-1\}$. For a subset X of a set Y, we define $Y \setminus X = \{y \in Y : y \notin X\}$. We write $u \leftarrow_\$ Y$ if u is chosen uniformly at random from a set Y. For a vector $\mathbf{s} = (s_i)_{i\in\mathbb{Z}_m} \in X^m$ and $r \in \mathbb{Z}_m$, we define $\mathsf{Shift}_r(\mathbf{s})$ as the vector obtained by shifting elements by r. Formally, $\mathbf{u} = (u_i)_{i\in\mathbb{Z}_m} = \mathsf{Shift}_r(\mathbf{s})$ is defined by $u_i = s_{(i-r) \bmod m}$ for all $i \in \mathbb{Z}_m$. If X is a field $\mathbb{F}$, Shift_r can be expressed by a linear operation. Formally, define a permutation matrix $\mathbf{P}_r \in \mathbb{F}^{m\times m}$ as the one whose (i,j)-th entry is 1 if $j = (i-r) \bmod m$ and 0 otherwise, where we identify the sets indexing the rows and columns of the matrix as $\mathbb{Z}_m$. Then it holds that $\mathsf{Shift}_r(\mathbf{s}) = \mathbf{P}_r \cdot \mathbf{s}$. It also holds that $\mathbf{P}_r^{-1} \cdot \mathbf{s} = \mathbf{P}_r^\top \cdot \mathbf{s} = \mathbf{P}_{-r} \cdot \mathbf{s} = \mathsf{Shift}_{-r}(\mathbf{s})$ Let $\mathbf{0}_m$ be the zero vector of dimension m and $\mathbf{1}_m$ be the all-ones vector of dimension m. We simply write $\mathbf{0}$ or $\mathbf{1}$ if the dimension is clear from the context. Let $\mathbf{e}_i$ denote the i-th unit vector whose entry is 1 at position i, and 0 otherwise. For a vector $\mathbf{v}$ of dimension m, we define $\mathrm{diag}(\mathbf{v})$ as a diagonal matrix whose (i,j)-th entry is the i-th entry of $\mathbf{v}$ if $j = i$ and 0 otherwise. For two vectors $\mathbf{u}, \mathbf{v}$ over a ring, we denote the standard inner product of $\mathbf{u}$ and $\mathbf{v}$ by $\langle \mathbf{u}, \mathbf{v} \rangle$. Throughout the paper, we fix the following notations:

- n is the total number of players.
- t is the maximum number of corrupted players (see Section 3.2).
- $\mathbb{K}$ is the minimum finite field such that $|\mathbb{K}| \geq 2n$. Fix $2n$ pairwise distinct elements $\beta_0, \beta_1, \ldots, \beta_{n-1}, \alpha_1, \ldots, \alpha_n \in \mathbb{K}$.

3.2 Secure Multiparty Computation

We denote the set of n players by $\{\mathsf{P}_1, \ldots, \mathsf{P}_n\}$, where P_i is called the i-th player. Assume that each player P_i has a private input x_i from a finite set D. Let $\mathcal{F}(x_1, \ldots, x_n) = (y_1, \ldots, y_n)$ be an n-input/n-output randomized functionality. We assume the *correlated randomness model*, in which there is a trusted dealer who samples $(r_1, \ldots, r_n)$ according to a joint distribution $\mathcal{D}$ over the Cartesian product $R_1 \times \cdots \times R_n$ of n sets, and gives $r_i \in R_i$ to each player P_i

before he decides his input. We assume computationally unbounded adversaries who passively corrupt up to t players. (We do not consider active adversaries whose corrupted players deviate from protocols arbitrarily.) Let Π be a protocol between n players in the correlated randomness model. For a subset $T \subseteq [n]$ of size at most t and any input $\mathbf{x} = (x_i)_{i\in[n]}$, consider the following two processes:

Ideal process. This process is defined with respect to a simulator Sim. Let $(y_1, \ldots, y_n) \leftarrow \mathcal{F}(\mathbf{x})$. The output of this process is

$$\mathsf{Ideal}_{\mathcal{F},\mathsf{Sim}}(T, \mathbf{x}) := (\mathsf{Sim}(T, (x_i, y_i)_{i\in T}), (y_i)_{i\in[n]}).$$

Real process. Suppose that all players each holding an input x_i execute Π honestly. Let $\mathsf{View}_{\Pi,i}(\mathbf{x})$ denote the view of P_i at the end of the protocol execution (which consists of his private input x_i, correlated randomness r_i, and messages that he received or sent during the execution of Π), and let $\mathsf{Output}_{\Pi,i}(\mathbf{x})$ be the output of P_i. The output of this process is

$$\mathsf{Real}_{\Pi}(T, \mathbf{x}) := ((\mathsf{View}_{\Pi,i}(\mathbf{x}))_{i\in T}, (\mathsf{Output}_{\Pi,i}(\mathbf{x}))_{i\in[n]}).$$

We say that Π is a *t-secure MPC protocol for $\mathcal{F}$* if for any subset $T \subseteq [n]$ of size at most t and any input $\mathbf{x} = (x_i)_{i\in[n]}$, the distributions $\mathsf{Ideal}_{\mathcal{F},\mathsf{Sim}}(T, \mathbf{x})$ and $\mathsf{Real}_{\Pi}(T, \mathbf{x})$ are perfectly identical to each other.

Let g be a deterministic function on D^n. We say that Π is a t-secure protocol computing g if it is a t-secure protocol for the functionality that takes $\mathbf{x}$ as input and gives $g(\mathbf{x})$ to every player. Then we have that Π is a t-secure MPC protocol computing g if and only if

Correctness. For any input $\mathbf{x}$ and any $i \in [n]$, it holds with probability 1 that $\mathsf{Output}_{\Pi,i}(\mathbf{x}) = g(\mathbf{x})$.

Privacy. For any set $T \subseteq [n]$ of size at most t and any pair of inputs $\mathbf{x} = (x_i)_{i\in[n]}$, $\mathbf{w} = (w_i)_{i\in[n]}$ such that $(x_i)_{i\in T} = (w_i)_{i\in T}$ and $g(\mathbf{x}) = g(\mathbf{w})$, the distributions $(\mathsf{View}_{\Pi,i}(\mathbf{x}))_{i\in T}$ and $(\mathsf{View}_{\Pi,i}(\mathbf{w}))_{i\in T}$ are perfectly identical to each other.

We denote by $\mathrm{Comm}_i(\Pi)$ the total number of bits sent or received by the i-th player P_i during the execution of a protocol Π with worst-case inputs. We define the bottleneck complexity of Π as $\mathrm{BC}(\Pi) = \max_{i\in[n]}\{\mathrm{Comm}_i(\Pi)\}$. We denote by $\mathrm{Rand}_i(\Pi)$ the size of correlated randomness for P_i, i.e., the total number of bits received by P_i in the setup of Π, and define $\mathrm{CR}(\Pi) = \max_{i\in[n]}\{\mathrm{Rand}_i(\Pi)\}$. We denote by $\mathrm{Round}(\Pi)$ the round complexity of Π, i.e., the number of sequential rounds of interaction.

Let $\mathcal{G}$ be a functionality. We say that a protocol Π is in the *$\mathcal{G}$-hybrid model* if players invoke $\mathcal{G}$ during the execution of Π, that is, a trusted third party receives messages from players and gives them the correct output of $\mathcal{G}$. The composition theorem [29] implies that if a protocol Π securely realizes a functionality $\mathcal{F}$ in the $\mathcal{G}$-hybrid model and a protocol $\Pi_{\mathcal{G}}$ securely realizes $\mathcal{G}$, then the composition of Π and $\Pi_{\mathcal{G}}$, i.e., the protocol obtained by replacing all invocations of $\mathcal{G}$ in Π with $\Pi_{\mathcal{G}}$, also securely realizes $\mathcal{F}$. While the above theorem assumes sequential composition, a set of protocols in the paper can be composed concurrently.

3.3 Basic Algorithms and Protocols

Let $\mathbb{G}$ be an abelian group (e.g., a finite field or a ring of integers modulo m). Define $\mathsf{Additive}_{\mathbb{G}}(s)$ as an algorithm to generate additive shares over $\mathbb{G}$ for a secret $s \in \mathbb{G}$. Formally, on input $s \in \mathbb{G}$, $\mathsf{Additive}_{\mathbb{G}}(s)$ chooses $(s_1, \ldots, s_n) \in \mathbb{G}^n$ uniformly at random conditioned on $s = \sum_{i\in[n]} s_i$, and outputs it.

Broadcast. Let $\mathcal{F}_{\mathsf{Broadcast},i}$ be the functionality which receives an input y from the i-th player and gives y to all players. Since all players are supposed to be semi-honest, a protocol $\Pi_{\mathsf{Broadcast},i}$ realizing $\mathcal{F}_{\mathsf{Broadcast},i}$ with low bottleneck complexity is straightforward (see [21] for a formal description). Roughly speaking, assume that the set of n players is represented by a binary tree whose height is $O(\log n)$ and root is P_i. Each player sends his two children the element that he received from his parent node. The complexity of $\Pi_{\mathsf{Broadcast},i}$ is $\mathrm{CR}(\Pi_{\mathsf{Broadcast},i}) = 0$, $\mathrm{BC}(\Pi_{\mathsf{Broadcast},i}) = O(\ell_y)$, and $\mathrm{Round}(\Pi_{\mathsf{Broadcast},i}) = O(\log n)$, where ℓ_y is the bit-length of y.

Sum. In Fig. 1, we describe the functionality $\mathcal{F}_{\mathsf{Sum},\mathbb{G}}$ which receives group elements $x_1, \ldots, x_n \in \mathbb{G}$, each from P_i, and gives $s := \sum_{i\in[n]} x_i$ to all players. There exists a protocol $\Pi_{\mathsf{Sum},\mathbb{G}}$ realizing $\mathcal{F}_{\mathsf{Sum},\mathbb{G}}$ such that $\mathrm{CR}(\Pi_{\mathsf{Sum},\mathbb{G}}) = O(\log|\mathbb{G}|)$, $\mathrm{BC}(\Pi_{\mathsf{Sum},\mathbb{G}}) = O(\log|\mathbb{G}|)$ and $\mathrm{Round}(\Pi_{\mathsf{Sum},\mathbb{G}}) = O(\log n)$ [21]. If the underlying group $\mathbb{G}$ is clear from the context, we simply write $\mathcal{F}_{\mathsf{Sum}}$ and Π_{Sum} instead of $\mathcal{F}_{\mathsf{Sum},\mathbb{G}}$ and $\Pi_{\mathsf{Sum},\mathbb{G}}$, respectively.

Functionality $\mathcal{F}_{\mathsf{Sum},\mathbb{G}}((x_i)_{i\in[n]})$

Upon receiving a group element $x_i \in \mathbb{G}$ from each player P_i, $\mathcal{F}_{\mathsf{Sum},\mathbb{G}}$ gives every player $s := \sum_{i\in[n]} x_i$.

Fig. 1. The functionality $\mathcal{F}_{\mathsf{Sum},\mathbb{G}}$

3.4 Ramp Secret Sharing

Recall that $\mathbb{K}$ is the minimum finite field such that $|\mathbb{K}| \geq 2n$ and we fix $2n$ pairwise distinct elements $\beta_0, \beta_1, \ldots, \beta_{n-1}, \alpha_1, \ldots, \alpha_n \in \mathbb{K}$. Let ℓ be a positive integer such that $\ell \leq n$. Define $\mathsf{RSS}_\ell(\mathbf{s})$ as an algorithm to generate shares of the (t, ℓ, n)-ramp secret sharing scheme for a secret vector $\mathbf{s} \in \mathbb{K}^\ell$. Formally, for $\mathbf{s} \in \mathbb{K}^\ell$, we define a set $\mathcal{R}_\mathbf{s}$ of polynomials as

$$\mathcal{R}_\mathbf{s} := \{\varphi \in \mathbb{K}[X] : \deg\varphi \leq t + \ell,\ (\varphi(\beta_0), \ldots, \varphi(\beta_{\ell-1})) = \mathbf{s}\}$$

On input $\mathbf{s} \in \mathbb{K}^\ell$, $\mathsf{RSS}_\ell(\mathbf{s})$ chooses a polynomial φ uniformly at random from $\mathcal{R}_\mathbf{s}$, and then outputs $(\varphi(\alpha_1), \ldots, \varphi(\alpha_n))$.

Regarding RSS_ℓ, we recall basic mathematical facts that we will use to construct our protocols in the following lemmas. We defer the proofs to the full version.

Lemma 1. *Let $T \subseteq [n]$ be any set of size at most t and $\mathbf{s} \in \mathbb{K}^\ell$. Then, there is a polynomial $\Delta_\mathbf{s} \in \mathcal{R}_\mathbf{s}$ such that $\Delta_\mathbf{s}(\alpha_i) = 0$ for all $i \in T$.*

Proof. Let $\mathbf{s} = (s_0, \ldots, s_{\ell-1})$ and $T = \{i_1, \ldots, i_{|T|}\}$. Consider the following linear equations with variables $(\delta_0, \ldots, \delta_{t+\ell})$:

$$\begin{pmatrix} 1 & \beta_0 & \cdots & \beta_0^{t+\ell} \\ & & \vdots & \\ 1 & \beta_{\ell-1} & \cdots & \beta_{\ell-1}^{t+\ell} \\ 1 & \alpha_{i_1} & \cdots & \alpha_{i_1}^{t+\ell} \\ & & \vdots & \\ 1 & \alpha_{i_{|T|}} & \cdots & \alpha_{i_{|T|}}^{t+\ell} \end{pmatrix} \begin{pmatrix} \delta_0 \\ \delta_1 \\ \vdots \\ \delta_{t+\ell} \end{pmatrix} = \begin{pmatrix} s_0 \\ \vdots \\ s_{\ell-1} \\ 0 \\ \vdots \\ 0 \end{pmatrix}. \tag{2}$$

Since $\ell + |T| < t + \ell + 1$ and the Vandermonde matrices have full rank, there exists a solution $(\delta_0, \ldots, \delta_{t+\ell})$ to (2). The polynomial $\Delta_\mathbf{s}(X) = \sum_{i=0}^{t+\ell} \delta_i X^i$ is a desired one.

Lemma 2. *Let $\mathbf{s}, \mathbf{u} \in \mathbb{K}^\ell$ and $\varphi_\mathbf{s} \in \mathcal{R}_\mathbf{s}$. If $\varphi_\mathbf{u}$ is uniformly distributed over $\mathcal{R}_\mathbf{u}$, then $\varphi_\mathbf{s} + \varphi_\mathbf{u}$ is uniformly distributed over $\mathcal{R}_{\mathbf{s}+\mathbf{u}}$.*

Proof. Fix any $\varphi_1 \in \mathcal{R}_\mathbf{u}$. Observe that there is a one-to-one correspondence between $\varphi_\mathbf{u} \in \mathcal{R}_\mathbf{u}$ and $\varphi_0 \in \mathcal{R}_\mathbf{0}$ under the relation $\varphi_\mathbf{u} = \varphi_0 + \varphi_1$. In particular, if $\varphi_\mathbf{u}$ is uniformly distributed over $\mathcal{R}_\mathbf{u}$, then the corresponding φ_0 is uniformly distributed over $\mathcal{R}_\mathbf{0}$. We have that $\varphi_\mathbf{s} + \varphi_\mathbf{u} = (\varphi_\mathbf{s} + \varphi_1) + \varphi_0$ and $\varphi_\mathbf{s} + \varphi_1 \in \mathcal{R}_{\mathbf{s}+\mathbf{u}}$. Therefore, $\varphi_\mathbf{s} + \varphi_\mathbf{u}$ is uniformly distributed over $\mathcal{R}_{\mathbf{s}+\mathbf{u}}$.

Lemma 3. *Let $\mathbf{s} = (s_0, \ldots, s_{\ell-1}) \in \mathbb{K}^\ell$. Then, there is an algorithm $\mathsf{Reconst}_\ell$ such that*

$$\sum_{i \in [n]} \mathsf{Reconst}_\ell(j, i; v_i) = s_j, \ \forall j = 0, 1, \ldots, \ell - 1$$

for any possible shares $(v_1, \ldots, v_n) \leftarrow \mathsf{RSS}_\ell(\mathbf{s})$. Furthermore, $\mathsf{Reconst}_\ell$ is linear in the sense that $\mathsf{Reconst}_\ell(j, i; v) + \mathsf{Reconst}_\ell(j, i; v') = \mathsf{Reconst}_\ell(j, i; v + v')$ for any $v, v' \in \mathbb{K}$.

Proof. The standard results on polynomial interpolation imply that for each $j = 0, 1, \ldots, \ell - 1$, there exist constants $L_{1j}, \ldots, L_{nj}$ depending on $\alpha_1, \ldots, \alpha_n$ and β_j only, such that

$$L_{1j} \cdot \varphi(\alpha_1) + \cdots + L_{nj} \cdot \varphi(\alpha_n) = \varphi(\beta_j)$$

for any polynomial φ of degree at most $n - 1$. The statements follow if we define $\mathsf{Reconst}_\ell$ as $\mathsf{Reconst}_\ell(j, i; v) = L_{ij} \cdot v$ for $j \in \{0, 1, \ldots, \ell - 1\}$, $i \in [n]$ and $v \in \mathbb{K}$.

We introduce a deterministic algorithm $\mathsf{FixedShare}_\ell$ that outputs predetermined shares consistent with a given secret vector. Formally, we fix a deterministic algorithm $\mathsf{FixedSample}_\ell$ which on input $\mathbf{s} \in \mathbb{K}^\ell$, computes a polynomial $\psi_\mathbf{s} \in \mathcal{R}_\mathbf{s}$. It can be implemented efficiently, e.g., with Gaussian elimination. Define $\mathsf{FixedShare}_\ell$ as follows: On input $i \in [n]$ and $\mathbf{s} \in \mathbb{K}^\ell$, $\mathsf{FixedShare}_\ell(i, \mathbf{s})$ computes $\psi_\mathbf{s} = \mathsf{FixedSample}_\ell(\mathbf{s})$ and outputs $\psi_\mathbf{s}(\alpha_i)$. Note that $(\mathsf{FixedShare}_\ell(i, \mathbf{s}))_{i\in[n]}$ is a tuple of possible shares of a secret vector $\mathbf{s}$.

4 Our First Protocol for Symmetric Functions

We call a function $h : \{0,1\}^n \to \{0,1\}$ *symmetric* if $h(x_{\sigma(1)}, \dots, x_{\sigma(n)}) = h(x_1, \dots, x_n)$ for any input $(x_1, \dots, x_n) \in \{0,1\}^n$ and any permutation $\sigma : [n] \to [n]$. By definition, the value of a symmetric function h is determined only by the Hamming weight w of the input, i.e., $w := |\{i \in [n] : x_i = 1\}| = \sum_{i\in[n]} x_i$. Thus, there is the unique function $f : \{0, 1, \dots, n\} \to \{0,1\}$ such that $f(x_1 + \cdots + x_n) = h(x_1, \dots, x_n)$ for all $(x_1, \dots, x_n) \in \{0,1\}^n$.

Remark 1. The authors of [44, 40] considered a related class of functions called *abelian programs*. Specifically, a function $\tilde{h} : \mathbb{G}^n \to \{0,1\}$ is called an abelian program over an abelian group $\mathbb{G}$ if there exists a function $f : \mathbb{G} \to \{0,1\}$ such that $\tilde{h}(\tilde{x}_1, \dots, \tilde{x}_n) = f(\tilde{x}_1 + \cdots + \tilde{x}_n)$ for all $(\tilde{x}_1, \dots, \tilde{x}_n) \in \mathbb{G}^n$, where addition is taken over $\mathbb{G}$. As pointed out in [3], abelian programs can compute a symmetric function $h : \{0,1\}^n \to \{0,1\}$ by setting $\mathbb{G} = \mathbb{Z}_{n+1}$ and viewing each input $x_i \in \{0,1\}$ as an element $\tilde{x}_i \in \mathbb{Z}_{n+1}$ (i.e., embed $\{0,1\}$ into $\mathbb{Z}_{n+1}$). The authors of [40] presented an information-theoretic MPC protocol Π for an abelian program $\tilde{h} : \mathbb{G}^n \to \{0,1\}$ such that $\mathrm{CR}(\Pi) = O(|\mathbb{G}|)$ and $\mathrm{BC}(\Pi) = O(\log |\mathbb{G}|)$. Based on the above correspondence, the protocol has $\mathrm{CR}(\Pi) = O(n)$ and $\mathrm{BC}(\Pi) = O(\log n)$ when computing a symmetric function $h : \{0,1\}^n \to \{0,1\}$.

First, for a parameter ℓ, we show an $(n - \Theta(n/\ell))$-secure protocol for any symmetric function h in the $\mathcal{F}_{\mathsf{Sum}}$-hybrid model such that the bottleneck complexity is $O(\log n)$ and the amount of correlated randomness is $O(\ell \log n)$. If we set $\ell = \Theta(\log n)$ and implement $\mathcal{F}_{\mathsf{Sum}}$ with the protocol in [21], then we obtain an $(n - o(n))$-secure protocol such that the bottleneck complexity is $O(\log n)$ and the amount of correlated randomness is $O((\log n)^2)$.

Theorem 2. *Let $h : \{0,1\}^n \to \{0,1\}$ be a symmetric function. Let ℓ be any integer such that $\ell \le n + 1$, and suppose that $t \le n - \lceil (n+1)/\ell \rceil$. The protocol Π_{Sym} described in Fig. 2 is a t-secure MPC protocol computing h in the $(\mathcal{F}_{\mathsf{Sum},\mathbb{Z}_m}, \mathcal{F}_{\mathsf{Sum},\mathbb{K}})$-hybrid model. If $\mathcal{F}_{\mathsf{Sum},\mathbb{G}}$ is implemented by a protocol with bottleneck complexity $b_{\mathsf{Sum},\mathbb{G}}$ and correlated randomness of size $c_{\mathsf{Sum},\mathbb{G}}$ for any abelian group $\mathbb{G}$, then the protocol Π_{Sym} achieves $\mathrm{CR}(\Pi_{\mathsf{Sym}}) = O(\ell \log n + c_{\mathsf{Sum},\mathbb{Z}_m} + c_{\mathsf{Sum},\mathbb{K}})$ and $\mathrm{BC}(\Pi_{\mathsf{Sym}}) = O(b_{\mathsf{Sum},\mathbb{Z}_m} + b_{\mathsf{Sum},\mathbb{K}})$.*

Protocol Π_{Sym}

Notations.

- Let $h : \{0,1\}^n \to \{0,1\}$ be a symmetric function.
- Let $f : \{0,1,\ldots,n\} \to \{0,1\}$ be a function such that $h(x_1,\ldots,x_n) = f(\sum_{i\in[n]} x_i)$ for all $(x_1,\ldots,x_n) \in \{0,1\}^n$.
- Let $\ell \le n+1$, $k := \lceil (n+1)/\ell \rceil$ and $m := \ell k$.
- Define $\mathbf{F} = (F_i)_{i\in\mathbb{Z}_m} \in \mathbb{K}^m$ by $F_i = f(i)$ if $0 \le i \le n$ and $F_i = 0$ otherwise.

Input. Each player P_i has $x_i \in \{0,1\}$.

Output. Every player obtains $z = h(x_1,\ldots,x_n)$.

Setup.

1. Let $r \leftarrow_{\$} \mathbb{Z}_m$ and $(r_i)_{i\in[n]} \leftarrow \mathsf{Additive}_{\mathbb{Z}_m}(r)$.
2. Define $\mathbf{S} \in \mathbb{K}^m$ by $\mathbf{S} = \mathsf{Shift}_r(\mathbf{F})$ and decompose $\mathbf{S}$ into ℓ vectors $\mathbf{U}^{(0)},\ldots,\mathbf{U}^{(\ell-1)}$ of dimension k, i.e., $\mathbf{S} = (\mathbf{U}^{(0)},\ldots,\mathbf{U}^{(\ell-1)})$.
3. For each $j = 0,1,\ldots,\ell-1$, let $(v_i^{(j)})_{i\in[n]} \leftarrow \mathsf{RSS}_k(\mathbf{U}^{(j)})$.
4. Each player P_i receives $(r_i, v_i^{(0)},\ldots,v_i^{(\ell-1)})$.

Protocol.

1. Each player P_i computes $y_i = x_i + r_i \bmod m$.
2. Players obtain $y = \mathcal{F}_{\mathsf{Sum},\mathbb{Z}_m}((y_i)_{i\in[n]})$.
3. Each player computes $(\sigma,\tau) \in \mathbb{Z}_\ell \times \mathbb{Z}_k$ such that $y = \sigma k + \tau$.
4. Each player P_i computes $z_i = \mathsf{Reconst}_k(\tau, i; v_i^{(\sigma)})$.
5. Players obtain $z = \mathcal{F}_{\mathsf{Sum},\mathbb{K}}((z_i)_{i\in[n]})$.
6. Each player P_i outputs z.

Fig. 2. Our first protocol Π_{Sym} for computing a symmetric function

Proof. First, we prove the correctness of Π_{Sym}. Let $\mathbf{x} \in \{0,1\}^n$ be any input. Since $r = \sum_{i\in[n]} r_i$, it holds that $y = r + \sum_{i\in[n]} x_i$. Since $(v_i^{(j)})_{i\in[n]}$ are shares of RSS_k for a secret vector $\mathbf{U}^{(j)}$, it also holds that

$$z = \sum_{i\in[n]} z_i = \sum_{i\in[n]} \mathsf{Reconst}_k(\tau, i; v_i^{(\sigma)}) = (\mathbf{U}^{(\sigma)})_\tau = (\mathbf{S})_{\sigma k+\tau} = (\mathbf{S})_y = F_{(y-r) \bmod m}$$

where $(\mathbf{U}^{(\sigma)})_\tau$ is the τ-th element of $\mathbf{U}^{(\sigma)}$ and $(\mathbf{S})_y$ is the y-th element of $\mathbf{S}$. Therefore, we have that $z = f(\sum_{i\in[n]} x_i) = h(x_1, \ldots, x_n)$.

Next, we prove the privacy of Π_{Sym}. Let $T \subseteq [n]$ be the set of corrupted players. Let $H = [n] \setminus T$ be the set of honest players and fix an honest player $j \in H$. Note that in the $\mathcal{F}_{\mathsf{Sum}}$-hybrid model, corrupted players' view can be simulated from the following elements since the other elements are locally computed from them:

Correlated randomness. $(r_i, v_i^{(0)}, \ldots, v_i^{(\ell-1)})$ for all $i \in T$;
Online messages. $y = \sum_{i\in[n]} x_i + r$ and z.

Let $\mathbf{x} = (x_i)_{i\in[n]}, \widetilde{\mathbf{x}} = (\widetilde{x}_i)_{i\in[n]} \in \{0,1\}^n$ be any pair of inputs such that $x_i = \widetilde{x}_i$ $(\forall i \in T)$ and $h(x_1, \ldots, x_n) = h(\widetilde{x}_1, \ldots, \widetilde{x}_n)$. It is sufficient to prove that the distribution of the above elements during the execution of Π_{Sym} on input $\mathbf{x}$ is identical to that on input $\widetilde{\mathbf{x}}$. To show the equivalence of the distributions, we show a bijection between the random strings used by Π_{Sym} on input $\mathbf{x}$ and the random strings used by Π_{Sym} on input $\widetilde{\mathbf{x}}$ such that the correlated randomness and the online messages received by T are the same under this bijection. The set of all random strings is

$$\mathcal{R} = \left\{ \left((r_i)_{i\in[n]}, \phi^{(0)}, \ldots, \phi^{(\ell-1)} \right) : r_i \in \mathbb{Z}_m,\ \phi^{(j)} \in \mathcal{R}_{\mathbf{U}^{(j)}} \right\},$$

where $r = \sum_{i\in[n]} r_i$ and $(\mathbf{U}^{(0)}, \ldots, \mathbf{U}^{(\ell-1)}) = \mathsf{Shift}_r(\mathbf{F})$. We denote the randomness of Π_{Sym} on input $\mathbf{x}$ by $R = ((r_i)_{i\in[n]}, \phi^{(0)}, \ldots, \phi^{(\ell-1)})$ and that on input $\widetilde{\mathbf{x}}$ by $\widetilde{R} = ((\widetilde{r}_i)_{i\in[n]}, \widetilde{\phi}^{(0)}, \ldots, \widetilde{\phi}^{(\ell-1)})$. We consider a bijection that maps the randomness $R \in \mathcal{R}$ to $\widetilde{R} \in \mathcal{R}$ in such a way that

$$\widetilde{r}_i = \begin{cases} r_i, & \text{if } i \in T, \\ r_i + x_i - \widetilde{x}_i, & \text{if } i \in H, \end{cases}$$
$$\widetilde{\phi}^{(j)} = \phi^{(j)} + \Delta_{\widetilde{\mathbf{U}}^{(j)} - \mathbf{U}^{(j)}}$$

where

$$r := \sum_{i\in[n]} r_i,\ (\mathbf{U}^{(0)}, \ldots, \mathbf{U}^{(\ell-1)}) := \mathsf{Shift}_r(\mathbf{F}),$$
$$\widetilde{r} := \sum_{i\in[n]} \widetilde{r}_i,\ (\widetilde{\mathbf{U}}^{(0)}, \ldots, \widetilde{\mathbf{U}}^{(\ell-1)}) := \mathsf{Shift}_{\widetilde{r}}(\mathbf{F}),$$

and $\Delta_{\widetilde{\mathbf{U}}^{(j)}-\mathbf{U}^{(j)}} \in \mathcal{R}_{\widetilde{\mathbf{U}}^{(j)}-\mathbf{U}^{(j)}}$ is a polynomial such that $\Delta_{\widetilde{\mathbf{U}}^{(j)}-\mathbf{U}^{(j)}}(\alpha_i) = 0$ for all $i \in T$, whose existence is guaranteed by Lemma 1. The image is indeed a consistent random string, i.e., $((\widetilde{r}_i)_{i\in[n]}, \widetilde{\phi}^{(0)}, \dots, \widetilde{\phi}^{(\ell-1)}) \in \mathcal{R}$, since $\phi^{(j)} \in \mathcal{R}_{\mathbf{U}^{(j)}}$ implies that $\widetilde{\phi}^{(j)} = \phi^{(j)} + \Delta_{\widetilde{\mathbf{U}}^{(j)}-\mathbf{U}^{(j)}} \in \mathcal{R}_{\widetilde{\mathbf{U}}^{(j)}}$. The above map is indeed a bijection since it has the inverse

$$r_i = \begin{cases} \widetilde{r}_i, & \text{if } i \in T, \\ \widetilde{r}_i + \widetilde{x}_i - x_i, & \text{if } i \in H, \end{cases}$$
$$\phi^{(j)} = \widetilde{\phi}^{(j)} - \Delta_{\widetilde{\mathbf{U}}^{(j)}-\mathbf{U}^{(j)}}.$$

This bijection does not change the correlated randomness $(r_i, v_i^{(0)}, \dots, v_i^{(\ell-1)})_{i\in T}$ of T since

$$\widetilde{v}_i^{(j)} = \widetilde{\phi}^{(j)}(\alpha_i) = \phi^{(j)}(\alpha_i) + \Delta_{\widetilde{\mathbf{U}}^{(j)}-\mathbf{U}^{(j)}}(\alpha_i) = \phi^{(j)}(\alpha_i) = v_i^{(j)}$$

for all $i \in T$. It can be seen that $\widetilde{x}_i + \widetilde{r}_i = \widetilde{x}_i + (r_i + x_i - \widetilde{x}_i) = x_i + r_i$ for $i \in H$. In particular, the message y is the same in both executions. Since $h(x_1, \dots, x_n) = h(\widetilde{x}_1, \dots, \widetilde{x}_n)$, the message z is also the same in both executions, which implies that the bijection does not change online messages seen by corrupted players.

As for the efficiency of Π_{Sym}, players need to invoke $\mathcal{F}_{\mathsf{Sum},\mathbb{Z}_m}$ and $\mathcal{F}_{\mathsf{Sum},\mathbb{K}}$. Thus, we have $\mathrm{CR}(\Pi_{\mathsf{Sym}}) = O(\log m + \ell \log |\mathbb{K}|) + c_{\mathsf{Sum},\mathbb{Z}_m} + c_{\mathsf{Sum},\mathbb{K}} = O(\ell \log n + c_{\mathsf{Sum},\mathbb{Z}_m} + c_{\mathsf{Sum},\mathbb{K}})$ and $\mathrm{BC}(\Pi_{\mathsf{Sym}}) = O(b_{\mathsf{Sum},\mathbb{Z}_m} + b_{\mathsf{Sum},\mathbb{K}})$.

Recall that the protocol $\Pi_{\mathsf{Sum},\mathbb{G}}$ realizes $\mathcal{F}_{\mathsf{Sum},\mathbb{G}}$ with bottleneck complexity $b_{\mathsf{Sum},\mathbb{G}} = O(\log |\mathbb{G}|)$ and correlated randomness of size $c_{\mathsf{Sum},\mathbb{G}} = O(\log |\mathbb{G}|)$ [21]. Thus, we have the following corollary.

Corollary 1. *Let ℓ be any integer such that $\ell \le n+1$, and suppose that $t \le n - \lceil (n+1)/\ell \rceil$. Then, there exists a t-secure MPC protocol Π computing a symmetric function $h : \{0,1\}^n \to \{0,1\}$ such that $\mathrm{CR}(\Pi) = O(\ell \log n)$ and $\mathrm{BC}(\Pi) = O(\log n)$.*

Setting $\ell = \Theta(\log n)$, we obtain the following corollary.

Corollary 2. *If $t = n - \Theta(n/\log n)$, then there exists a t-secure MPC protocol Π computing a symmetric function $h : \{0,1\}^n \to \{0,1\}$ such that $\mathrm{CR}(\Pi) = O((\log n)^2)$ and $\mathrm{BC}(\Pi) = O(\log n)$.*

Setting $\ell \approx 1/\epsilon$ for a constant $0 < \epsilon < 1/2$, we also obtain a $(1-\epsilon)n$-secure protocol such that both bottleneck complexity and the amount of correlated randomness are $O(\log n)$.

Corollary 3. *For any constant ϵ such that $0 < \epsilon < 1/2$, there exists a $(1-\epsilon)n$-secure MPC protocol Π computing a symmetric function $h : \{0,1\}^n \to \{0,1\}$ such that $\mathrm{CR}(\Pi) = O(\epsilon^{-1} \log n) = O(\log n)$ and $\mathrm{BC}(\Pi) = O(\log n)$.*

Remark 2 (Round and computational complexity). We have $\mathrm{Round}(\Pi_{\mathsf{Sym}}) = 2 \cdot \mathrm{Round}(\Pi)$ if $\mathcal{F}_{\mathsf{Sum}}$ is instantiated with a protocol Π. In particular, if we choose the protocol Π_{Sum} in [21] as Π, we have that $\mathrm{Round}(\Pi_{\mathsf{Sym}}) = O(\log n)$. Each player receives $O(\ell)$ elements in $\mathbb{K}$ and performs a constant number of operations in $\mathbb{K}$. The computational complexity of Π_{Sym} is thus $O(\ell)$ field operations.

5 Our Second Protocol for Symmetric Functions

In this section, we show another construction of protocols for symmetric functions in the $\mathcal{F}_{\mathsf{Sum}}$-hybrid model. Compared with the first construction, it reduces the amount of correlated randomness excluding correlated randomness required to realize $\mathcal{F}_{\mathsf{Sum}}$. First, we construct two building-block protocols with low bottleneck complexity, and then we show our main protocol.

5.1 Additional Building Blocks

For parameters k, ℓ, we consider the following sub-functionalities:

Linear transformation $\mathcal{F}_{\mathsf{LT}}$. Given ramp shares of a k-dimensional secret vector $\mathbf{s}$, players obtain ramp shares of an ℓ-dimensional vector $\mathbf{u} := \mathbf{M} \cdot \mathbf{s}$, where $\mathbf{M}$ is a public ℓ-by-k matrix. The formal description is shown in Fig. 3.

Inner product $\mathcal{F}_{\mathsf{IP}}$. Given ramp shares of two ℓ-dimensional vectors $\mathbf{v}$ and $\mathbf{w}$, players obtain the inner product $\langle \mathbf{v}, \mathbf{w} \rangle$. The formal description is shown in Fig. 4.

We show protocols for $\mathcal{F}_{\mathsf{LT}}$ and $\mathcal{F}_{\mathsf{IP}}$.

Proposition 1. *Let k, ℓ be positive integers with $\ell \leq k \leq n$ and $\mathbf{M}$ be an ℓ-by-k matrix over $\mathbb{K}$. Suppose that $t \leq n - \ell$. Then, the protocol Π_{LT} described in Fig. 3 is a t-secure MPC protocol for $\mathcal{F}_{\mathsf{LT}}$ in the $\mathcal{F}_{\mathsf{Sum},\mathbb{K}}$-hybrid model. If $\mathcal{F}_{\mathsf{Sum},\mathbb{G}}$ is implemented by a protocol with bottleneck complexity $b_{\mathsf{Sum},\mathbb{G}}$ and correlated randomness of size $c_{\mathsf{Sum},\mathbb{G}}$ for any abelian group $\mathbb{G}$, then the protocol Π_{LT} achieves $\mathrm{CR}(\Pi_{\mathsf{LT}}) = O(\log n + \ell \cdot c_{\mathsf{Sum},\mathbb{K}})$ and $\mathrm{BC}(\Pi_{\mathsf{LT}}) = O(\ell \cdot b_{\mathsf{Sum},\mathbb{K}})$.*

Proof. Recall that α_i (resp. β_j) is the point associated with the i-th share (resp. the j-th component of a secret vector) of RSS_ℓ and RSS_k. To simplify notations, we denote $(\varphi(\alpha_i))_{i \in T}$ by $\varphi(\alpha_T)$ for a set $T \subseteq [n]$ and a polynomial φ.

Let T be a subset of size at most t and $(v_i)_{i \in [n]}$ be an input to the protocol $\Pi = \Pi_{\mathsf{LT}}$. Let $\mathbf{s}$ be the secret of RSS_k determined by $(v_i)_{i \in [n]}$ and set $\mathbf{u} = \mathbf{M} \cdot \mathbf{s}$.

Consider the real process. Observe that the a_i's and b_i's can be written as $a_i = A(\alpha_i)$ and $b_i = B(\alpha_i)$ for random polynomials $A \leftarrow_{\$} \mathcal{R}_{\mathbf{r}}$ and $B \leftarrow_{\$} \mathcal{R}_{\mathbf{0}_\ell}$. Also, it holds that

$$
\begin{aligned}
\mathbf{y} &= \sum_{i \in [n]} \mathbf{x}_i \\
&= \mathbf{M} \cdot \sum_{i \in [n]} \begin{pmatrix} \mathsf{Reconst}_k(0, i; v_i) \\ \vdots \\ \mathsf{Reconst}_k(k-1, i; v_i) \end{pmatrix} - \sum_{i \in [n]} \begin{pmatrix} \mathsf{Reconst}_\ell(0, i; a_i) \\ \vdots \\ \mathsf{Reconst}_\ell(\ell-1, i; a_i) \end{pmatrix} \\
&= \mathbf{M} \cdot \mathbf{s} - \mathbf{r} \\
&= \mathbf{u} - \mathbf{r}.
\end{aligned}
$$

Functionality $\mathcal{F}_{\mathsf{LT}}(\mathbf{M}; (v_i)_{i \in [n]})$

1. Players have shares $(v_i)_{i \in [n]}$ of RSS_k for a secret $\mathbf{s} = (s_0, \ldots, s_{k-1})$.
2. $\mathcal{F}_{\mathsf{LT}}$ receives $v_i \in \mathbb{K}$ from each player P_i.
3. $\mathcal{F}_{\mathsf{LT}}$ reconstructs $s_j = \sum_{i \in [n]} \mathsf{Reconst}_k(j, i; v_i)$ for all $j = 0, 1, \ldots, k-1$, and computes $\mathbf{u} = \mathbf{M} \cdot \mathbf{s} \in \mathbb{K}^\ell$.
4. $\mathcal{F}_{\mathsf{LT}}$ computes shares $(w_i)_{i \in [n]} \leftarrow \mathsf{RSS}_\ell(\mathbf{u})$ and gives w_i to each player P_i.

Protocol Π_{LT}

Input. Each player P_i has the i-th share $v_i \in \mathbb{K}$ of RSS_k for a secret $\mathbf{s} = (s_0, \ldots, s_{k-1})$.

Output. Each player P_i obtains w_i, where $(w_i)_{i \in [n]} \leftarrow \mathcal{F}_{\mathsf{LT}}(\mathbf{M}; (v_i)_{i \in [n]})$.

Setup.

1. Let $\mathbf{r} \leftarrow_{\$} \mathbb{K}^\ell$.
2. Let $(a_i)_{i \in [n]} \leftarrow \mathsf{RSS}_\ell(\mathbf{r})$ and $(b_i)_{i \in [n]} \leftarrow \mathsf{RSS}_\ell(\mathbf{0}_\ell)$.
3. Each player P_i receives (a_i, b_i).

Protocol.

1. Each player P_i computes
$$\mathbf{x}_i = \mathbf{M} \cdot \begin{pmatrix} \mathsf{Reconst}_k(0, i; v_i) \\ \vdots \\ \mathsf{Reconst}_k(k-1, i; v_i) \end{pmatrix} - \begin{pmatrix} \mathsf{Reconst}_\ell(0, i; a_i) \\ \vdots \\ \mathsf{Reconst}_\ell(\ell-1, i; a_i) \end{pmatrix}$$
2. Players obtain $\mathbf{y} = \mathcal{F}_{\mathsf{Sum},\mathbb{K}}((\mathbf{x}_i)_{i \in [n]})$, where $\mathcal{F}_{\mathsf{Sum},\mathbb{K}}$ is invoked in an element-wise way.
3. Each player P_i computes $w'_i = \mathsf{FixedShare}_\ell(i, \mathbf{y})$.
4. Each player P_i outputs $w_i = w'_i + a_i + b_i$.

Fig. 3. The functionality $\mathcal{F}_{\mathsf{LT}}$ and a protocol Π_{LT} implementing it

Furthermore, for all $i \in [n]$,

$$w_i = \psi_{\mathbf{y}}(\alpha_i) + A(\alpha_i) + B(\alpha_i),$$

where $\psi_{\mathbf{y}} \in \mathcal{R}_{\mathbf{y}}$ is the polynomial computed by the deterministic algorithm $\mathsf{FixedShare}_\ell$. Thus, the output of the real process in the $\mathcal{F}_{\mathsf{Sum}}$-hybrid model is

$$\begin{aligned}\mathsf{Real}_\Pi(T, (v_i)_{i\in[n]}) &= ((\mathsf{View}_{\Pi,i}((v_i)_{i\in[n]}))_{i\in T}; (\mathsf{Output}_{\Pi,i}((v_i)_{i\in[n]}))_{i\in[n]}) \\ &= ((v_i)_{i\in T}, A(\alpha_T), B(\alpha_T), \mathbf{y}; (\psi_{\mathbf{y}} + A + B)(\alpha_{[n]})),\end{aligned}$$

where $\mathbf{r} \leftarrow_\$ \mathbb{K}^\ell$, $A \leftarrow_\$ \mathcal{R}_{\mathbf{r}}$, $\mathbf{y} = \mathbf{u} - \mathbf{r}$, and $B \leftarrow_\$ \mathcal{R}_{\mathbf{0}_\ell}$. Here, we omit $\mathbf{x}_i$ and w'_i from the view of corrupted players since they are locally computed by the other elements.

Since $t \leq n - \ell$, Lemma 1 ensures that there exists a polynomial $\Delta_{\mathbf{r}} \in \mathcal{R}_{\mathbf{r}}$ such that $\Delta_{\mathbf{r}}(\alpha_i) = 0$ for all $i \in T$. If we set $A' = A - \Delta_{\mathbf{r}}$, then A' is uniformly distributed over $\mathcal{R}_{\mathbf{0}_\ell}$ and $A'(\alpha_i) = A(\alpha_i)$ for all $i \in T$ from Lemma 2. Thus, we have that

$$\mathsf{Real}_\Pi(T, (v_i)_{i\in[n]}) = ((v_i)_{i\in T}, A'(\alpha_T), B(\alpha_T), \mathbf{y}; (\psi_{\mathbf{y}} + A' + \Delta_{\mathbf{r}} + B)(\alpha_{[n]})),$$

where $\mathbf{r} \leftarrow_\$ \mathbb{K}^\ell$, $\mathbf{y} = \mathbf{u} - \mathbf{r}$, and $A', B \leftarrow_\$ \mathcal{R}_{\mathbf{0}_\ell}$. Since $\mathbf{u} - \mathbf{r}$ is uniformly distributed over $\mathbb{K}^\ell$, we have that

$$\mathsf{Real}_\Pi(T, (v_i)_{i\in[n]}) = ((v_i)_{i\in T}, A'(\alpha_T), B(\alpha_T), \mathbf{y}'; (\psi_{\mathbf{y}'} + A' + \Delta_{\mathbf{u}-\mathbf{y}'} + B)(\alpha_{[n]})),$$

where $\mathbf{y}' \leftarrow_\$ \mathbb{K}^\ell$ and $A', B \leftarrow_\$ \mathcal{R}_{\mathbf{0}_\ell}$. Since $\psi_{\mathbf{y}'} \in \mathcal{R}_{\mathbf{y}'}$, $A' \in \mathcal{R}_{\mathbf{0}_k}$ and $\Delta_{\mathbf{u}-\mathbf{y}'} \in \mathcal{R}_{\mathbf{u}-\mathbf{y}'}$, it holds that $\psi_{\mathbf{y}'} + A' + \Delta_{\mathbf{u}-\mathbf{y}'} \in \mathcal{R}_{\mathbf{u}}$. If we set $\phi' := \psi_{\mathbf{y}'} + A' + \Delta_{\mathbf{u}-\mathbf{y}'} + B$, then ϕ' is uniformly distributed over $\mathcal{R}_{\mathbf{u}}$ from Lemma 2. Since $\Delta_{\mathbf{u}-\mathbf{y}'}(\alpha_i) = 0$ for all $i \in T$, we have that

$$\begin{aligned}\mathsf{Real}_\Pi(T, (v_i)_{i\in[n]}) &= ((v_i)_{i\in T}, A'(\alpha_T), (\phi' - \psi_{\mathbf{y}'} - A' - \Delta_{\mathbf{u}-\mathbf{y}'})(\alpha_T), \mathbf{y}'; \phi'(\alpha_{[n]})) \\ &= ((v_i)_{i\in T}, A'(\alpha_T), (\phi' - \psi_{\mathbf{y}'} - A')(\alpha_T), \mathbf{y}'; \phi'(\alpha_{[n]})),\end{aligned}$$

where $\mathbf{y}' \leftarrow_\$ \mathbb{K}^\ell$, $A' \leftarrow_\$ \mathcal{R}_{\mathbf{0}_\ell}$ and $\phi' \leftarrow_\$ \mathcal{R}_{\mathbf{u}}$.

On the other hand, we define a simulator $\mathsf{Sim}(T, (v_i)_{i\in T}, (w_i)_{i\in T})$ as follows: First, it samples $\widetilde{\mathbf{y}} \leftarrow_\$ \mathbb{K}^\ell$ and $\widetilde{A} \leftarrow_\$ \mathcal{R}_{\mathbf{0}_\ell}$, and sets $\widetilde{a}_i = \widetilde{A}(\alpha_i)$ and $\widetilde{b}_i = w_i - \psi_{\widetilde{\mathbf{y}}}(\alpha_i) - \widetilde{A}(\alpha_i)$ for $i \in T$. Then, it outputs

$$\mathsf{Sim}(T, (v_i)_{i\in T}, (w_i)_{i\in T}) = ((v_i)_{i\in T}, (\widetilde{a}_i)_{i\in T}, (\widetilde{b}_i)_{i\in T}, \widetilde{\mathbf{y}}).$$

Note that the functionality $\mathcal{F} = \mathcal{F}_{\mathsf{LT}}$ gives players fresh shares of RSS_ℓ for a secret $\mathbf{u}$. Formally, the i-th player P_i receives $\phi(\alpha_i)$, where $\phi \leftarrow_\$ \mathcal{R}_{\mathbf{u}}$. Then, the output of the ideal process with respect to the functionality $\mathcal{F}$ and the simulator Sim is

$$\mathsf{Ideal}_{\mathcal{F},\mathsf{Sim}}(T, (v_i)_{i\in[n]}) = (\mathsf{Sim}(T, (v_i)_{i\in T}, \widetilde{\phi}(\alpha_T)); \widetilde{\phi}(\alpha_{[n]})),$$

where $\widetilde{\phi} \leftarrow_\$ \mathcal{R}_{\mathbf{u}}$. From the construction of Sim, we have that

$$\mathsf{Ideal}_{\mathcal{F},\mathsf{Sim}}(T,(v_i)_{i\in[n]}) = ((v_i)_{i\in T}, \widetilde{A}(\alpha_T), (\widetilde{\phi} - \psi_{\widetilde{\mathbf{y}}} - \widetilde{A})(\alpha_T), \widetilde{\mathbf{y}}; \widetilde{\phi}(\alpha_{[n]})),$$

where $\widetilde{\mathbf{y}} \leftarrow_\$ \mathbb{K}^\ell$, $\widetilde{A} \leftarrow_\$ \mathcal{R}_{\mathbf{0}_\ell}$, and $\widetilde{\phi} \leftarrow_\$ \mathcal{R}_{\mathbf{u}}$.

Therefore, we conclude that

$$\mathsf{Ideal}_{\mathcal{F},\mathsf{Sim}}(T,(v_i)_{i\in[n]}) = \mathsf{Real}_{\Pi}(T,(v_i)_{i\in[n]}).$$

Since players receive two shares of RSS_ℓ and invoke $\mathcal{F}_{\mathsf{Sum},\mathbb{K}}$ ℓ times, the size of correlated randomness is $\mathrm{CR}(\Pi_{\mathsf{LT}}) = O(\log|\mathbb{K}| + \ell\cdot c_{\mathsf{Sum},\mathbb{K}}) = O(\log n + \ell\cdot c_{\mathsf{Sum},\mathbb{K}})$ and the bottleneck complexity is $\mathrm{BC}(\Pi_{\mathsf{LT}}) = O(\ell\cdot b_{\mathsf{Sum},\mathbb{K}})$. □

Remark 3 (Round and computational complexity). The round complexity of Π_{LT} is $\mathrm{Round}(\Pi_{\mathsf{LT}}) = \mathrm{Round}(\Pi)$ if $\mathcal{F}_{\mathsf{Sum}}$ is instantiated with a protocol Π, since interaction occurs only when players invoke $\mathcal{F}_{\mathsf{Sum}}$. The computationally costly part is Step 1 of the online phase and hence the computational complexity of Π_{LT} is $O(\ell k)$ field operations per party.

Proposition 2. *Let ℓ be a positive integer with $\ell \le n$. Suppose that $t \le n-\ell$. Then, the protocol Π_{IP} described in Fig. 4 is a t-secure MPC protocol for $\mathcal{F}_{\mathsf{IP}}$ in the $(\mathcal{F}_{\mathsf{Sum},\mathbb{K}}, \mathcal{F}_{\mathsf{LT}})$-hybrid model. If $\mathcal{F}_{\mathsf{Sum},\mathbb{G}}$ is implemented by a protocol with bottleneck complexity $b_{\mathsf{Sum},\mathbb{G}}$ and correlated randomness of size $c_{\mathsf{Sum},\mathbb{G}}$ for any abelian group $\mathbb{G}$ then the protocol Π_{IP} achieves $\mathrm{CR}(\Pi_{\mathsf{IP}}) = O(\log n + \ell\cdot c_{\mathsf{Sum},\mathbb{K}})$ and $\mathrm{BC}(\Pi_{\mathsf{IP}}) = O(\ell\cdot b_{\mathsf{Sum},\mathbb{K}})$.*

Let $\mathbf{x} = (x_0,\ldots,x_{\ell-1})$ (resp. $\mathbf{y} = (y_0,\ldots,y_{\ell-1})$) be the secret determined by shares $(v_i)_{i\in[n]}$ (resp. $(w_i)_{i\in[n]}$).

First, we see the correctness of Π_{IP}. The linearity of RSS_ℓ implies that at Step 1 of the protocol, $(v'_i)_{i\in[n]}$ (resp. $(w'_i)_{i\in[n]}$) are shares of a secret $\mathbf{x}-\mathbf{a}$ (resp. $\mathbf{y}-\mathbf{b}$). We thus have that $\mathbf{x}' = \mathbf{x}-\mathbf{a}$, $\mathbf{y}' = \mathbf{y}-\mathbf{b}$ and $\mathbf{z}' = (\mathbf{x}-\mathbf{a})*(\mathbf{y}-\mathbf{b})$ at Steps 3 and 4. On the other hand, the functionality of $\mathcal{F}_{\mathsf{LT}}$ ensures that $(a'_i)_{i\in[n]}$ are shares of a secret

$$\mathbf{a}' := \mathrm{diag}(\mathbf{y}')\cdot\mathbf{a} = (\mathbf{y}-\mathbf{b})*\mathbf{a}$$

and similarly, $(b'_i)_{i\in[n]}$ are shares of a secret $\mathbf{b}' := (\mathbf{x}-\mathbf{a})*\mathbf{b}$. The linearity of RSS_ℓ implies that $d_i = z'_i + a'_i + b'_i + c_i + r_i$ is the i-th share of a secret

$$\begin{aligned}\mathbf{z}'+\mathbf{a}'+\mathbf{b}'+\mathbf{c}+\mathbf{s} &= (\mathbf{x}-\mathbf{a})*(\mathbf{y}-\mathbf{b}) + (\mathbf{y}-\mathbf{b})*\mathbf{a} + (\mathbf{x}-\mathbf{a})*\mathbf{b} + \mathbf{a}*\mathbf{b} + \mathbf{s}\\ &= \mathbf{x}*\mathbf{y}+\mathbf{s}.\end{aligned}$$

Thus it holds that $\mathbf{d} = \mathbf{x}*\mathbf{y}+\mathbf{s}$. The correctness follows from

$$z = \langle \mathbf{1}_\ell, \mathbf{d}\rangle = \langle \mathbf{1}_\ell, \mathbf{x}*\mathbf{y}\rangle + \langle \mathbf{1}_\ell, \mathbf{s}\rangle = \langle \mathbf{x},\mathbf{y}\rangle.$$

We show the privacy of Π_{IP}. Let $T \subseteq [n]$ be the set of corrupted players. Recall that α_i (resp. β_j) is the point associated with the i-th share (resp. the

Functionality $\mathcal{F}_{\mathsf{IP}}((v_i, w_i)_{i\in[n]})$

1. Players have shares $(v_i)_{i\in[n]}$ and $(w_i)_{i\in[n]}$ of RSS_ℓ for secrets $\mathbf{x} = (x_0, \ldots, x_{\ell-1})$ and $\mathbf{y} = (y_0, \ldots, y_{\ell-1})$, respectively.
2. $\mathcal{F}_{\mathsf{IP}}$ receives shares $v_i, w_i \in \mathbb{K}$ from each player P_i.
3. $\mathcal{F}_{\mathsf{IP}}$ reconstructs
$$x_j = \sum_{i\in[n]} \mathsf{Reconst}_\ell(j, i; v_i),\ \ y_j = \sum_{i\in[n]} \mathsf{Reconst}_\ell(j, i; w_i)$$
for all $j = 0, 1, \ldots, \ell-1$, and computes $z = \langle \mathbf{x}, \mathbf{y}\rangle$.
4. $\mathcal{F}_{\mathsf{IP}}$ gives z to every player P_i.

Protocol $\mathit{\Pi}_{\mathsf{IP}}$

Input. Each player P_i has the i-th shares $v_i, w_i \in \mathbb{K}$ of RSS_ℓ for secrets $\mathbf{x} = (x_0, \ldots, x_{\ell-1})$ and $\mathbf{y} = (y_0, \ldots, y_{\ell-1})$, respectively.

Output. Each player P_i obtains $z = \mathcal{F}_{\mathsf{IP}}((v_i, w_i)_{i\in[n]})$.

Setup.
1. Let $\mathbf{a}, \mathbf{b} \leftarrow_\$ \mathbb{K}^\ell$ and $\mathbf{c} = \mathbf{a} * \mathbf{b}$, where $*$ is the element-wise multiplication.
2. Let $(a_i)_{i\in[n]} \leftarrow \mathsf{RSS}_\ell(\mathbf{a})$, $(b_i)_{i\in[n]} \leftarrow \mathsf{RSS}_\ell(\mathbf{b})$ and $(c_i)_{i\in[n]} \leftarrow \mathsf{RSS}_\ell(\mathbf{c})$.
3. Choose a random vector $\mathbf{s} \in \mathbb{K}^\ell$ such that $\langle \mathbf{1}_\ell, \mathbf{s}\rangle = 0$.
4. Let $(r_i)_{i\in[n]} \leftarrow \mathsf{RSS}_\ell(\mathbf{s})$.
5. Each player P_i receives (a_i, b_i, c_i, r_i).

Protocol.
1. Each player P_i computes $v'_i = v_i - a_i$ and $w'_i = w_i - b_i$.
2. Each player P_i computes
$$\mathbf{x}'_i = (\mathsf{Reconst}_\ell(0, i; v'_i), \ldots, \mathsf{Reconst}_\ell(\ell-1, i; v'_i)),$$
$$\mathbf{y}'_i = (\mathsf{Reconst}_\ell(0, i; w'_i), \ldots, \mathsf{Reconst}_\ell(\ell-1, i; w'_i)).$$
3. Players obtain $\mathbf{x}' = \mathcal{F}_{\mathsf{Sum}}((\mathbf{x}'_i)_{i\in[n]})$ and $\mathbf{y}' = \mathcal{F}_{\mathsf{Sum},\mathbb{K}}((\mathbf{y}'_i)_{i\in[n]})$, where $\mathcal{F}_{\mathsf{Sum},\mathbb{K}}$ is invoked in an element-wise way.
4. Each player P_i computes $\mathbf{z}' = \mathbf{x}' * \mathbf{y}'$ and $z'_i = \mathsf{FixedShare}_\ell(i, \mathbf{z}')$.
5. Players obtain
$$(a'_i)_{i\in[n]} \leftarrow \mathcal{F}_{\mathsf{LT}}(\mathbf{N}; (a_i)_{i\in[n]}),\ \ (b'_i)_{i\in[n]} \leftarrow \mathcal{F}_{\mathsf{LT}}(\mathbf{M}; (b_i)_{i\in[n]}),$$
where $\mathbf{M} = \mathrm{diag}(\mathbf{x}')$ and $\mathbf{N} = \mathrm{diag}(\mathbf{y}')$.
6. Each player P_i computes $d_i = z'_i + a'_i + b'_i + c_i + r_i$ and
$$\mathbf{d}_i = (\mathsf{Reconst}_\ell(0, i; d_i), \ldots, \mathsf{Reconst}_\ell(\ell-1, i; d_i)).$$
7. Players obtain $\mathbf{d} = \mathcal{F}_{\mathsf{Sum},\mathbb{K}}((\mathbf{d}_i)_{i\in[n]})$.
8. Every player outputs $z = \langle \mathbf{1}_\ell, \mathbf{d}\rangle$.

Fig. 4. The functionality $\mathcal{F}_{\mathsf{IP}}$ and a protocol $\mathit{\Pi}_{\mathsf{IP}}$ implementing it

j-th component of a secret vector) of RSS_ℓ. To simplify notations, we denote $(\varphi(\alpha_i))_{i\in T}$ by $\varphi(\alpha_T)$ for a polynomial φ. In the $\mathcal{F}_{\mathsf{Sum}}$-hybrid model, corrupted players' view at Steps 3 and 7 (including their correlated randomness for $\mathcal{F}_{\mathsf{Sum}}$) only contains their inputs $(\mathbf{x}'_i, \mathbf{y}'_i, \mathbf{d}_i)_{i\in T}$ to $\mathcal{F}_{\mathsf{Sum}}$ and the outputs $\mathbf{x}', \mathbf{y}', \mathbf{d}$. Also, in the $\mathcal{F}_{\mathsf{LT}}$-hybrid model, corrupted players' view at Step 5 (including their correlated randomness for $\mathcal{F}_{\mathsf{LT}}$) only contains their inputs $(a_i, b_i)_{i\in T}$ to $\mathcal{F}_{\mathsf{LT}}$ and the outputs $(a'_i, b'_i)_{i\in T}$. It is therefore sufficient to show that the joint distribution of the following elements is simulated from $(v_i, w_i)_{i\in T}$ and $z = \mathcal{F}_{\mathsf{IP}}((v_i, w_i)_{i\in[n]})$ since the other elements are locally computed from them:

Correlated randomness. $(a_i, b_i, c_i, r_i)_{i\in T}$;
Online messages. $\mathbf{x}' = \mathbf{x} - \mathbf{a}$, $\mathbf{y}' = \mathbf{y} - \mathbf{b}$, $(a'_i, b'_i)_{i\in T}$ and $\mathbf{d} = \mathbf{x} * \mathbf{y} + \mathbf{s}$.

To analyze the distribution of the above elements, we define

$$\mathsf{View} = ((a_i, b_i, c_i, r_i, a'_i, b'_i)_{i\in T}, \mathbf{x}', \mathbf{y}', \mathbf{d}).$$

Observe that the distribution of View is given by

$$\mathsf{View} = (\phi_a(\alpha_T), \phi_b(\alpha_T), \phi_c(\alpha_T), \phi_s(\alpha_T), \phi_{a'}(\alpha_T), \phi_{b'}(\alpha_T), \mathbf{x} - \mathbf{a}, \mathbf{y} - \mathbf{b}, \mathbf{x} * \mathbf{y} + \mathbf{s}),$$

where

$$\mathbf{a}, \mathbf{b} \leftarrow_\$ \mathbb{K}^\ell,\ \mathbf{s} \leftarrow_\$ V_0 := \{\mathbf{s} \in \mathbb{K}^\ell : \langle \mathbf{1}_\ell, \mathbf{s}\rangle = 0\},\ \phi_a \leftarrow_\$ \mathcal{R}_{\mathbf{a}},\ \phi_b \leftarrow_\$ \mathcal{R}_{\mathbf{b}},$$
$$\phi_c \leftarrow_\$ \mathcal{R}_{\mathbf{a}*\mathbf{b}},\ \phi_s \leftarrow_\$ \mathcal{R}_{\mathbf{s}},\ \phi_{a'} \leftarrow_\$ \mathcal{R}_{(\mathbf{y}-\mathbf{b})*\mathbf{a}},\ \phi_{b'} \leftarrow_\$ \mathcal{R}_{(\mathbf{x}-\mathbf{a})*\mathbf{b}}.$$

Lemma 1 ensures that for any $\mathbf{v} \in \mathbb{K}^\ell$, there is a polynomial $\Delta_{\mathbf{v}} \in \mathcal{R}_{\mathbf{v}}$ such that $\Delta_{\mathbf{v}}(\alpha_i) = 0$ for all $i \in T$. If $\widetilde{\phi}_a$ is uniformly distributed over $\mathcal{R}_{\mathbf{0}_\ell}$, then $\widetilde{\phi}_a + \Delta_{\mathbf{a}}$ is uniformly distributed over $\mathcal{R}_{\mathbf{a}}$ from Lemma 2 and $(\widetilde{\phi}_a + \Delta_{\mathbf{a}})(\alpha_i) = \widetilde{\phi}_a(\alpha_i)$ for all $i \in T$. Similarly, let $\widetilde{\phi}_b, \widetilde{\phi}_c, \widetilde{\phi}_s, \widetilde{\phi}_{a'}, \widetilde{\phi}_{b'} \leftarrow_\$ \mathcal{R}_{\mathbf{0}_\ell}$, and then it holds that

$$\widetilde{\phi}_b + \Delta_{\mathbf{b}} \leftarrow_\$ \mathcal{R}_{\mathbf{b}},\ \widetilde{\phi}_c + \Delta_{\mathbf{a}*\mathbf{b}} \leftarrow_\$ \mathcal{R}_{\mathbf{a}*\mathbf{b}},\ \widetilde{\phi}_s + \Delta_{\mathbf{s}} \leftarrow_\$ \mathcal{R}_{\mathbf{s}},$$
$$\widetilde{\phi}_{a'} + \Delta_{(\mathbf{y}-\mathbf{b})*\mathbf{a}} \leftarrow_\$ \mathcal{R}_{(\mathbf{y}-\mathbf{b})*\mathbf{a}},\ \widetilde{\phi}_{b'} + \Delta_{(\mathbf{x}-\mathbf{a})*\mathbf{b}} \leftarrow_\$ \mathcal{R}_{(\mathbf{x}-\mathbf{a})*\mathbf{b}}.$$

It also holds that

$$\left(\widetilde{\phi}_b + \Delta_{\mathbf{b}}\right)(\alpha_i) = \widetilde{\phi}_b(\alpha_i),\ \left(\widetilde{\phi}_c + \Delta_{\mathbf{a}*\mathbf{b}}\right)(\alpha_i) = \widetilde{\phi}_c(\alpha_i),\ \left(\widetilde{\phi}_s + \Delta_{\mathbf{s}}\right)(\alpha_i) = \widetilde{\phi}_s(\alpha_i),$$
$$\left(\widetilde{\phi}_{a'} + \Delta_{(\mathbf{y}-\mathbf{b})*\mathbf{a}}\right)(\alpha_i) = \widetilde{\phi}_{a'}(\alpha_i),\ \left(\widetilde{\phi}_{b'} + \Delta_{(\mathbf{x}-\mathbf{a})*\mathbf{b}}\right)(\alpha_i) = \widetilde{\phi}_{b'}(\alpha_i)$$

for all $i \in T$. We thus have that

$$\mathsf{View} = (\widetilde{\phi}_a(\alpha_T), \widetilde{\phi}_b(\alpha_T), \widetilde{\phi}_c(\alpha_T), \widetilde{\phi}_s(\alpha_T), \widetilde{\phi}_{a'}(\alpha_T), \widetilde{\phi}_{b'}(\alpha_T), \mathbf{x} - \mathbf{a}, \mathbf{y} - \mathbf{b}, \mathbf{x} * \mathbf{y} + \mathbf{s}),$$

where $\mathbf{a}, \mathbf{b} \leftarrow_\$ \mathbb{K}^\ell$, $\mathbf{s} \leftarrow_\$ V_0$, and $\widetilde{\phi}_b, \widetilde{\phi}_c, \widetilde{\phi}_s, \widetilde{\phi}_{a'}, \widetilde{\phi}_{b'} \leftarrow_\$ \mathcal{R}_{\mathbf{0}_\ell}$. Since $\widetilde{\mathbf{a}} := \mathbf{x} - \mathbf{a}$ and $\widetilde{\mathbf{b}} := \mathbf{y} - \mathbf{b}$ are uniformly distributed over $\mathbb{K}^\ell$, we have that

$$\mathsf{View} = (\widetilde{\phi}_a(\alpha_T), \widetilde{\phi}_b(\alpha_T), \widetilde{\phi}_c(\alpha_T), \widetilde{\phi}_s(\alpha_T), \widetilde{\phi}_{a'}(\alpha_T), \widetilde{\phi}_{b'}(\alpha_T), \widetilde{\mathbf{a}}, \widetilde{\mathbf{b}}, \mathbf{x} * \mathbf{y} + \mathbf{s}),$$

where $\widetilde{\mathbf{a}}, \widetilde{\mathbf{b}} \leftarrow_\$ \mathbb{K}^\ell$, $\mathbf{s} \leftarrow_\$ V_0$, and $\widetilde{\phi}_b, \widetilde{\phi}_c, \widetilde{\phi}_s, \widetilde{\phi}_{a'}, \widetilde{\phi}_{b'} \leftarrow_\$ \mathcal{R}_{\mathbf{0}_\ell}$. Since $z = \mathcal{F}_{\mathsf{IP}}((v_i, w_i)_{i\in[n]}) = \langle \mathbf{x}, \mathbf{y}\rangle$, it holds that $\langle \mathbf{1}_\ell, \mathbf{x}*\mathbf{y} - z\cdot\mathbf{e}_0\rangle = \langle \mathbf{x}, \mathbf{y}\rangle - z = 0$. and hence $\mathbf{s}_0 := \mathbf{x}*\mathbf{y} - z\cdot\mathbf{e}_0 \in V_0$, where $\mathbf{e}_0 = (1, 0, \ldots, 0) \in \mathbb{K}^\ell$. Furthermore, since V_0 is a linear space, if $\mathbf{s}$ is uniformly distributed over V_0, then so is $\mathbf{s} + \mathbf{s}_0$. In particular, if $\mathbf{s}, \widetilde{\mathbf{s}} \leftarrow_\$ V_0$, then $\mathbf{x} * \mathbf{y} + \mathbf{s}$ and $z \cdot \mathbf{e}_0 + \widetilde{\mathbf{s}}$ follow the same distribution. We then have that

$$\mathsf{View} = (\widetilde{\phi}_a(\alpha_T), \widetilde{\phi}_b(\alpha_T), \widetilde{\phi}_c(\alpha_T), \widetilde{\phi}_s(\alpha_T), \widetilde{\phi}_{a'}(\alpha_T), \widetilde{\phi}_{b'}(\alpha_T), \widetilde{\mathbf{a}}, \widetilde{\mathbf{b}}, z \cdot \mathbf{e}_0 + \widetilde{\mathbf{s}}),$$

where $\widetilde{\mathbf{a}}, \widetilde{\mathbf{b}} \leftarrow_\$ \mathbb{K}^\ell$, $\widetilde{\mathbf{s}} \leftarrow_\$ V_0$, and $\widetilde{\phi}_a, \widetilde{\phi}_b, \widetilde{\phi}_c, \widetilde{\phi}_{a'}, \widetilde{\phi}_{b'}, \widetilde{\phi}_s \leftarrow_\$ \mathcal{R}_{\mathbf{0}_\ell}$. Therefore, we conclude that View is simulated from z only.

Since players receive four shares of RSS_ℓ and invoke $\mathcal{F}_{\mathsf{Sum},\mathbb{K}}$ $O(\ell)$ times and $\mathcal{F}_{\mathsf{LT}}$ twice, we have $\mathrm{CR}(\Pi_{\mathsf{IP}}) = O(\log|\mathbb{K}| + \ell \cdot c_{\mathsf{Sum},\mathbb{K}}) = O(\log n + \ell \cdot c_{\mathsf{Sum},\mathbb{K}})$ and $\mathrm{BC}(\Pi_{\mathsf{IP}}) = O(\ell \cdot b_{\mathsf{Sum},\mathbb{K}})$.

Remark 4 (Round and computational complexity). Since interaction occurs only when players invoke $\mathcal{F}_{\mathsf{Sum}}$ and $\mathcal{F}_{\mathsf{LT}}$, the round complexity of Π_{IP} is $\mathrm{Round}(\Pi_{\mathsf{IP}}) = 2\cdot\mathrm{Round}(\Pi) + \mathrm{Round}(\Pi_{\mathsf{LT}}) = 3\cdot\mathrm{Round}(\Pi)$ if $\mathcal{F}_{\mathsf{Sum}}$ is instantiated with a protocol Π. The most computationally costly part of Π_{IP} is executing the protocol Π_{LT} implementing $\mathcal{F}_{\mathsf{LT}}$. The computational complexity is thus $O(\ell^2)$ field operations per party.

5.2 Main Protocol

Now, we show our main protocol.

Theorem 3. *Let $h : \{0,1\}^n \to \{0,1\}$ be a symmetric function. Let ℓ, k be primes such that $\ell < k$ and $n + 1 \le \ell k \le O(n)$, and suppose that $t \le n - k$. The protocol Π'_{Sym} described in Fig. 5 is a t-secure MPC protocol for $\mathcal{F}_h$ in the $(\mathcal{F}_{\mathsf{Sum},\mathbb{Z}_m}, \mathcal{F}_{\mathsf{Sum},\mathbb{K}}, \mathcal{F}_{\mathsf{LT}}, \mathcal{F}_{\mathsf{IP}})$-hybrid model. If $\mathcal{F}_{\mathsf{Sum},\mathbb{G}}$ is implemented by a protocol with bottleneck complexity $b_{\mathsf{Sum},\mathbb{G}}$ and correlated randomness of size $c_{\mathsf{Sum},\mathbb{G}}$ for any abelian group $\mathbb{G}$, then the protocol Π'_{Sym} achieves $\mathrm{CR}(\Pi'_{\mathsf{Sym}}) = O(\log n + c_{\mathsf{Sum},\mathbb{Z}_m} + \ell \cdot c_{\mathsf{Sum},\mathbb{K}})$ and $\mathrm{BC}(\Pi'_{\mathsf{Sym}}) = O(b_{\mathsf{Sum},\mathbb{Z}_m} + \ell \cdot b_{\mathsf{Sum},\mathbb{K}})$.*

Proof. First, we prove the correctness of Π'_{Sym}. Let $\mathbf{x} \in \{0,1\}^n$ be any input. Since $r = \sum_{i\in[n]} r_i$, it holds that $y = r - \sum_{i\in[n]} x_i$. Let $(\sigma', \tau') := (\sigma + u, \tau + v)$. Note that we have $\phi(\sum_{i\in[n]} x_i) = \phi(y) + \phi(r) = (\sigma', \tau')$. Since $(d_i)_{i\in[n]}$ are shares of RSS_ℓ for a secret vector $\mathbf{e}_v$, the functionality of $\mathcal{F}_{\mathsf{LT}}$ implies that $(d'_i)_{i\in[n]}$ are shares of a secret vector $\mathbf{N}_y \cdot \mathbf{e}_v = \mathbf{P}_\sigma^\top \cdot \mathbf{M} \cdot \mathbf{P}_\tau \cdot \mathbf{e}_v = \mathbf{P}_\sigma^\top \cdot \mathbf{M} \cdot \mathbf{e}_{\tau'}$. Furthermore, since $(c_i)_{i\in[n]}$ are shares of RSS_k for a secret vector $\mathbf{e}_u$, the functionality of $\mathcal{F}_{\mathsf{IP}}$ implies that $z = \langle \mathbf{e}_u, \mathbf{P}_\sigma^\top \cdot \mathbf{M} \cdot \mathbf{e}_{\tau'}\rangle = \langle \mathbf{P}_\sigma \cdot \mathbf{e}_u, \mathbf{M} \cdot \mathbf{e}_{\tau'}\rangle = \langle \mathbf{e}_{\sigma'}, \mathbf{M} \cdot \mathbf{e}_{\tau'}\rangle = \mathbf{M}[\sigma', \tau']$ where $\mathbf{M}[\sigma', \tau']$ is the (σ', τ')-th entry of $\mathbf{M}$. Therefore, we have that $z = f(\phi^{-1}(\sigma', \tau')) = f(\sum_{i\in[n]} x_i) = h(x_1, \ldots, x_n)$.

Next, we prove the privacy of Π'_{Sym}. Let $T \subseteq [n]$ be the set of corrupted players. Recall that α_i (resp. β_j) is the point associated with the i-th share (resp. the j-th component of a secret vector) of RSS_ℓ and RSS_k. To simplify notations,

Protocol Π'_{Sym}

Notations.

- Let $h : \{0,1\}^n \to \{0,1\}$ be a symmetric function.
- Let $f : \{0,1,\ldots,n\} \to \{0,1\}$ be a function such that $h(x_1,\ldots,x_n) = f(\sum_{i\in[n]} x_i)$ for all $(x_1,\ldots,x_n) \in \{0,1\}^n$.
- Let ℓ, k be primes such that $\ell < k$ and $n+1 \leq \ell k$, and set $m = \ell k$.
- Let $\phi : \mathbb{Z}_m \to \mathbb{Z}_\ell \times \mathbb{Z}_k$ be the ring isomorphism induced by the Chinese remainder theorem.
- Define a matrix $\mathbf{M} \in \mathbb{K}^{\ell\times k}$ as follows: For $(\sigma,\tau) \in \mathbb{Z}_\ell \times \mathbb{Z}_k$, the (σ,τ)-th entry of $\mathbf{M}$ is $f(\phi^{-1}(\sigma,\tau))$ if $\phi^{-1}(\sigma,\tau) \in \{0,1,\ldots,n\}$, and 0 otherwise, where we identify the sets indexing the rows and columns of $\mathbf{M}$ as $\mathbb{Z}_\ell$ and $\mathbb{Z}_k$, respectively.

Input. Each player P_i has $x_i \in \{0,1\}$.

Output. Every player obtains $z = h(x_1,\ldots,x_n)$.

Setup.

1. Let $r \leftarrow_{\$} \mathbb{Z}_m$, $(r_i)_{i\in[n]} \leftarrow \mathsf{Additive}_{\mathbb{Z}_m}(r)$, and $(u,v) = \phi(r)$.
2. Let $(c_i)_{i\in[n]} \leftarrow \mathsf{RSS}_\ell(\mathbf{e}_u)$ and $(d_i)_{i\in[n]} \leftarrow \mathsf{RSS}_k(\mathbf{e}_v)$, where $\mathbf{e}_u \in \mathbb{K}^\ell$ (resp. $\mathbf{e}_v \in \mathbb{K}^k$) is the unit vector whose entry is 1 at position $u \in \mathbb{Z}_\ell$ (resp. $v \in \mathbb{Z}_k$), and 0 otherwise.
3. Each player P_i receives (r_i, c_i, d_i).

Protocol.

1. Each player P_i computes $y_i = x_i - r_i \bmod m$.
2. Players obtain $y = \mathcal{F}_{\mathsf{Sum}}((y_i)_{i\in[n]})$.
3. Each player computes $(\sigma,\tau) = \phi(y) \in \mathbb{Z}_\ell \times \mathbb{Z}_k$ and $\mathbf{N}_y = \mathbf{P}_\sigma^\top \cdot \mathbf{M} \cdot \mathbf{P}_\tau$.
4. Players obtain $(d'_i)_{i\in[n]} \leftarrow \mathcal{F}_{\mathsf{LT}}(\mathbf{N}_y; (d_i)_{i\in[n]})$.
5. Players obtain $z \leftarrow \mathcal{F}_{\mathsf{IP}}((c_i, d'_i)_{i\in[n]})$.
6. Each player P_i outputs z.

Fig. 5. Our second protocol Π'_{Sym} for computing a symmetric function

we denote $(\varphi(\alpha_i))_{i\in T}$ by $\varphi(\alpha_T)$ for a polynomial φ. In the $\mathcal{F}_{\mathsf{Sum}}$-hybrid model, corrupted players' view at Step 2 only contains their inputs $(y_i)_{i\in T}$ to $\mathcal{F}_{\mathsf{Sum}}$ and the output y. Also, in the $\mathcal{F}_{\mathsf{LT}}$-hybrid model, corrupted players' view at Step 5 (including their correlated randomness for $\mathcal{F}_{\mathsf{LT}}$) only contains their inputs $(d_i)_{i\in T}$ to $\mathcal{F}_{\mathsf{LT}}$ and the outputs $(d'_i)_{i\in T}$. It is sufficient to show that the joint distribution of the following elements is simulated from $(x_i)_{i\in T}$ and $h(x_1,\ldots,x_n)$ since the other elements are locally computed from them:

Correlated randomness. (r_i, c_i, d_i) for all $i \in T$;
Online messages. $y = \sum_{i\in[n]} x_i + r$, $(d'_i)_{i\in T}$, and z.

To analyze the distribution of the above element, we define $\mathsf{View} = ((r_i, c_i, d_i, d'_i)_{i\in T}, y, z)$. Observe that the distribution of View is given by

$$\mathsf{View} = \left((r_i)_{i\in T}, \phi_c(\alpha_T), \phi_d(\alpha_T), \phi_{d'}(\alpha_T), y = \sum_{i\in[n]} x_i + \sum_{i\in[n]} r_i, z\right),$$

where $(r_1,\ldots,r_n) \leftarrow_{\$} \mathbb{Z}_m^n$, $(u,v) = \phi(\sum_{i\in[n]} r_i)$, $\phi_c \leftarrow_{\$} \mathcal{R}_{\mathbf{e}_u}$, $\phi_d \leftarrow_{\$} \mathcal{R}_{\mathbf{e}_v}$, and $\phi_{d'} \leftarrow_{\$} \mathcal{R}_{\mathbf{N}_y\cdot\mathbf{e}_v}$. The correctness of Π'_{Sym} implies that

$$\mathsf{View} = \left((r_i)_{i\in T}, \phi_c(\alpha_T), \phi_d(\alpha_T), \phi_{d'}(\alpha_T), y = \sum_{i\in[n]} x_i + \sum_{i\in[n]} r_i, h(x_1,\ldots,x_n)\right).$$

Lemma 1 ensures that for any $\mathbf{v} \in \mathbb{K}^\ell$, there is a polynomial $\Delta_{\mathbf{v}} \in \mathcal{R}_{\mathbf{v}}$ such that $\Delta_{\mathbf{v}}(\alpha_i) = 0$ for all $i \in T$. If $\widetilde{\phi}_c$ are uniformly distributed over $\mathcal{R}_{\mathbf{0}_\ell}$, then $\widetilde{\phi}_c + \Delta_{\mathbf{e}_u}$ is uniformly distributed over $\mathcal{R}_{\mathbf{e}_u}$ from Lemma 2 and $(\widetilde{\phi}_c + \Delta_{\mathbf{e}_u})(\alpha_i) = 0$ for all $i \in T$. Similarly, if $\widetilde{\phi}_d, \widetilde{\phi}_{d'} \leftarrow_{\$} \mathcal{R}_{\mathbf{0}_k}$, then it holds that $\widetilde{\phi}_d + \Delta_{\mathbf{e}_v} \leftarrow_{\$} \mathcal{R}_{\mathbf{e}_v}$ and $\widetilde{\phi}_{d'} + \Delta_{\mathbf{N}_y\cdot\mathbf{e}_v} \leftarrow_{\$} \mathcal{R}_{\mathbf{N}_y\cdot\mathbf{e}_v}$. It also holds that $(\widetilde{\phi}_d + \Delta_{\mathbf{e}_v})(\alpha_i) = \widetilde{\phi}_d(\alpha_i)$ and $(\widetilde{\phi}_{d'} + \Delta_{\mathbf{N}_y\cdot\mathbf{e}_v})(\alpha_i) = \widetilde{\phi}_{d'}(\alpha_i)$ for all $i \in T$. We thus have that

$$\mathsf{View} = \left((r_i)_{i\in T}, \widetilde{\phi}_c(\alpha_T), \widetilde{\phi}_d(\alpha_T), \widetilde{\phi}_{d'}(\alpha_T), y = \sum_{i\in[n]} x_i + \sum_{i\in[n]} r_i, h(x_1,\ldots,x_n)\right),$$

where $(r_1,\ldots,r_n) \leftarrow_{\$} \mathbb{Z}_m^n$, $\widetilde{\phi}_c \leftarrow_{\$} \mathcal{R}_{\mathbf{0}_\ell}$, $\widetilde{\phi}_d, \widetilde{\phi}_{d'} \leftarrow_{\$} \mathcal{R}_{\mathbf{0}_k}$. Since $T \neq [n]$ and $(r_i)_{i\in[n]}$ are independent and uniformly random elements, the joint distribution of $(r_i)_{i\in T}$ and $y = \sum_{i\in[n]} x_i + \sum_{i\in[n]} r_i$ is the uniform distribution over $\mathbb{Z}_m^{|T|+1}$. We thus have that

$$\mathsf{View} = \left((\widetilde{r}_i)_{i\in T}, \widetilde{\phi}_c(\alpha_T), \widetilde{\phi}_d(\alpha_T), \widetilde{\phi}_{d'}(\alpha_T), \widetilde{y}, h(x_1,\ldots,x_n)\right),$$

where $((\widetilde{r}_i)_{i\in T}, \widetilde{y}) \leftarrow_{\$} \mathbb{Z}_m^{|T|+1}$, $\widetilde{\phi}_c \leftarrow_{\$} \mathcal{R}_{\mathbf{0}_\ell}$, $\widetilde{\phi}_d, \widetilde{\phi}_{d'} \leftarrow_{\$} \mathcal{R}_{\mathbf{0}_k}$. Therefore, we conclude that View is simulated from $h(x_1,\ldots,x_n)$ only.

Finally, since players need to invoke $\mathcal{F}_{\mathsf{Sum},\mathbb{Z}_m}$, $\mathcal{F}_{\mathsf{Sum},\mathbb{K}}$, $\mathcal{F}_{\mathsf{LT}}$ and $\mathcal{F}_{\mathsf{IP}}$, Propositions 1 and 2 imply $\mathrm{CR}(\Pi'_{\mathsf{Sym}}) = O(\log|\mathbb{K}|) + c_{\mathsf{Sum},\mathbb{Z}_m} + O(\log n + \ell\cdot c_{\mathsf{Sum},\mathbb{K}}) = O(\log n + c_{\mathsf{Sum},\mathbb{Z}_m} + \ell\cdot c_{\mathsf{Sum},\mathbb{K}})$ and $\mathrm{BC}(\Pi'_{\mathsf{Sym}}) = O(b_{\mathsf{Sum},\mathbb{Z}_m} + \ell\cdot b_{\mathsf{Sum},\mathbb{K}})$.

Remark 5 (Round and computational complexity). The round complexity of Π'_{Sym} is $\mathrm{Round}(\Pi'_{\mathsf{Sym}}) = \mathrm{Round}(\Pi) + \mathrm{Round}(\Pi_{\mathsf{LT}}) + \mathrm{Round}(\Pi_{\mathsf{IP}}) = 5 \cdot \mathrm{Round}(\Pi)$ if $\mathcal{F}_{\mathsf{Sum}}$ is instantiated with a protocol Π. Since the computation of $\mathbf{N}_y = \mathbf{P}_\sigma^\top \cdot \mathbf{M} \cdot \mathbf{P}_\tau$ is just permuting rows and columns of $\mathbf{M}$, it can be done by $O(\ell k)$ field operations. The computational complexities of Π_{LT} implementing $\mathcal{F}_{\mathsf{LT}}$ and Π_{IP} implementing $\mathcal{F}_{\mathsf{IP}}$ are $O(\ell k)$ and $O(\ell^2)$ field operations, respectively. Since $\ell < k$, the computational complexity of Π'_{Sym} is $O(\ell k) = O(n)$ field operations.

Thanks to Bertrand's postulate [46, Theorem 5.8], we can choose primes k, ℓ such that $k = \Theta(n/\log n)$ and $\ell = \Theta(\log n)$. If we implement $\mathcal{F}_{\mathsf{Sum},\mathbb{G}}$ with the protocol $\Pi_{\mathsf{Sum},\mathbb{G}}$ in [21], we obtain an $(n - O(n/\log n))$-secure protocol for a symmetric function h such that the bottleneck complexity is $O((\log n)^2)$ and the amount of correlated randomness is $O((\log n)^2)$ but it has no advantage of efficiency compared to the one obtained from the first construction (see Corollary 2). If there exists a protocol Π for $\mathcal{F}_{\mathsf{Sum},\mathbb{K}}$ such that $\mathrm{CR}(\Pi) = o(\log |\mathbb{K}|)$ and $\mathrm{BC}(\Pi) = O(\log |\mathbb{K}|)$, then we obtain a protocol that achieves bottleneck complexity $O((\log n)^2)$ and a smaller amount of correlated randomness $o((\log n)^2)$. Unfortunately, it is currently unknown whether such Π exists, which we leave for future work.

Note that setting k and ℓ as primes close to ϵn and $1/\epsilon$ (resp.) leads to a protocol with asymptotically the same complexity as Corollary 3.

6 Conclusion

In this paper, we presented two constructions of $(n - o(n))$-secure n-party protocols that compute symmetric functions achieving polylogarithmic (in n) bottleneck complexity and correlated randomness simultaneously. Our first construction achieves bottleneck complexity $O(\log n)$ and requires correlated randomness of size $O((\log n)^2)$. Our second construction has larger bottleneck complexity $O((\log n)^2)$ but reduces the amount of correlated randomness excluding correlated randomness required to securely compute the sum of elements in an abelian group $\mathbb{G}$. If the secure summation functionality is implemented with the state-of-the-art protocol in [21], the construction derives a protocol with bottleneck complexity $O((\log n)^2)$ and correlated randomness of size $O((\log n)^2)$ but there is no advantage of efficiency compared to the one obtained from the first construction. If a secure summation protocol with bottleneck complexity $o(\log |\mathbb{G}|)$ and correlated randomness of size $O(\log |\mathbb{G}|)$ is devised in the future, then our second construction implies a protocol that achieves bottleneck complexity $O((\log n)^2)$ and a smaller amount of correlated randomness $o((\log n)^2)$. Unfortunately, it is currently unknown whether such a secure summation protocol exists, which we leave for future work. In addition, we found a security flaw in the summation protocol from the conference version of this paper. Currently, we are not able to fix the flaw or to find an alternative protocol achieving the same level of performance, and we also leave it for future work.

Acknowledgements

We would like to thank anonymous reviewers for their insightful comments. We also thank Keitaro Hiwatashi and Takahiro Matsuda for their helpful comments and suggestions. This work was partially supported by JST AIP Acceleration Research JPMJCR22U5, Japan and JST CREST Grant Number JPMJCR22M1, Japan.

References

1. Beaver, D.: Efficient multiparty protocols using circuit randomization. In: Advances in Cryptology – CRYPTO '91. pp. 420–432 (1992)
2. Beerliová-Trubíniová, Z., Hirt, M.: Perfectly-secure mpc with linear communication complexity. In: Theory of Cryptography. pp. 213–230 (2008)
3. Beimel, A., Gabizon, A., Ishai, Y., Kushilevitz, E., Meldgaard, S., Paskin-Cherniavsky, A.: Non-interactive secure multiparty computation. In: Advances in Cryptology – CRYPTO 2014, Part II. pp. 387–404 (2014)
4. Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness theorems for non-cryptographic fault-tolerant distributed computation. In: Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing. pp. 1–10 (1988)
5. Ben-Sasson, E., Fehr, S., Ostrovsky, R.: Near-linear unconditionally-secure multiparty computation with a dishonest minority. In: Advances in Cryptology – CRYPTO 2012. pp. 663–680 (2012)
6. Bendlin, R., Damgård, I., Orlandi, C., Zakarias, S.: Semi-homomorphic encryption and multiparty computation. In: Advances in Cryptology – EUROCRYPT 2011. pp. 169–188 (2011)
7. Blakley, G.R., Meadows, C.: Security of ramp schemes. In: Advances in Cryptology – CRYPTO '84. pp. 242–268 (1985)
8. Boyle, E., Gilboa, N., Ishai, Y.: Secure computation with preprocessing via function secret sharing. In: Theory of Cryptography. pp. 341–371 (2019)
9. Boyle, E., Gilboa, N., Ishai, Y., Nof, A.: Sublinear gmw-style compiler for mpc with preprocessing. In: Advances in Cryptology – CRYPTO 2021. pp. 457–485 (2021)
10. Boyle, E., Jain, A., Prabhakaran, M., Yu, C.H.: The Bottleneck Complexity of Secure Multiparty Computation. In: 45th International Colloquium on Automata, Languages, and Programming (ICALP 2018). Leibniz International Proceedings in Informatics (LIPIcs), vol. 107, pp. 24:1–24:16 (2018)
11. Chaum, D., Crépeau, C., Damgard, I.: Multiparty unconditionally secure protocols. In: Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing. pp. 11–19. STOC '88 (1988)
12. Chida, K., Genkin, D., Hamada, K., Ikarashi, D., Kikuchi, R., Lindell, Y., Nof, A.: Fast large-scale honest-majority MPC for malicious adversaries. In: Advances in Cryptology – CRYPTO 2018, Part III. pp. 34–64 (2018)
13. Couteau, G.: A note on the communication complexity of multiparty computation in the correlated randomness model. In: Advances in Cryptology – EUROCRYPT 2019. pp. 473–503 (2019)
14. Cramer, R., Damgård, I., de Haan, R.: Atomic secure multi-party multiplication with low communication. In: Advances in Cryptology – EUROCRYPT 2007. pp. 329–346 (2007)

15. Cramer, R., Damgård, I., Maurer, U.: General secure multi-party computation from any linear secret-sharing scheme. In: Advances in Cryptology – EUROCRYPT 2000. pp. 316–334 (2000)
16. Damgård, I., Ishai, Y., Krøigaard, M.: Perfectly secure multiparty computation and the computational overhead of cryptography. In: Advances in Cryptology – EUROCRYPT 2010. pp. 445–465 (2010)
17. Damgård, I., Nielsen, J.B.: Scalable and unconditionally secure multiparty computation. In: Advances in Cryptology – CRYPTO 2007. pp. 572–590 (2007)
18. Damgård, I., Nielsen, J.B., Nielsen, M., Ranellucci, S.: The tinytable protocol for 2-party secure computation, or: Gate-scrambling revisited. In: Advances in Cryptology – CRYPTO 2017. pp. 167–187 (2017)
19. Damgård, I., Pastro, V., Smart, N., Zakarias, S.: Multiparty computation from somewhat homomorphic encryption. In: Advances in Cryptology – CRYPTO 2012. pp. 643–662 (2012)
20. Dani, V., King, V., Movahedi, M., Saia, J., Zamani, M.: Secure multi-party computation in large networks. Distributed Computing **30**, 193–229 (2017)
21. Eriguchi, R.: Unconditionally secure multiparty computation for symmetric functions with low bottleneck complexity. In: Advances in Cryptology – ASIACRYPT 2023. pp. 335–368 (2023)
22. Eriguchi, R.: Secure multiparty computation of symmetric functions with polylogarithmic bottleneck complexity and correlated randomness. In: 5th Information-Theoretic Cryptography (ITC) Conference (2024)
23. Escudero, D., Goyal, V., Polychroniadou, A., Song, Y.: TurboPack: Honest majority mpc with constant online communication. In: Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security. pp. 951–964. CCS '22 (2022)
24. Franklin, M., Yung, M.: Communication complexity of secure computation (extended abstract). In: Proceedings of the Twenty-Fourth Annual ACM Symposium on Theory of Computing. pp. 699–710. STOC '92 (1992)
25. Gelles, Y., Komargodski, I.: Optimal load-balanced scalable distributed agreement. Cryptology ePrint Archive, Paper 2023/1139 (2023), https://eprint.iacr.org/2023/1139
26. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing. pp. 169–0178. STOC '09 (2009)
27. Gentry, C., Sahai, A., Waters, B.: Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In: Advances in Cryptology – CRYPTO 2013. pp. 75–92 (2013)
28. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game. In: Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing. pp. 218–229. STOC '87 (1987)
29. Goldreich, O.: Foundations of cryptography: volume 2, basic applications. Cambridge University Press (2009)
30. Goyal, V., Li, H., Ostrovsky, R., Polychroniadou, A., Song, Y.: ATLAS: Efficient and scalable MPC in the honest majority setting. In: Advances in Cryptology – CRYPTO 2021, Part II. pp. 244–274 (2021)
31. Goyal, V., Liu, Y., Song, Y.: Communication-efficient unconditional MPC with guaranteed output delivery. In: Advances in Cryptology – CRYPTO 2019, Part II. pp. 85–114 (2019)

32. Goyal, V., Song, Y., Zhu, C.: Guaranteed output delivery comes free in honest majority MPC. In: Advances in Cryptology – CRYPTO 2020, Part II. pp. 618–646 (2020)
33. Halevi, S., Ishai, Y., Jain, A., Kushilevitz, E., Rabin, T.: Secure multiparty computation with general interaction patterns. In: Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science. pp. 157–168. ITCS '16 (2016)
34. Halevi, S., Lindell, Y., Pinkas, B.: Secure computation on the web: Computing without simultaneous interaction. In: Advances in Cryptology – CRYPTO 2011. pp. 132–150 (2011)
35. Hirt, M., Maurer, U.: Robustness for free in unconditional multi-party computation. In: Advances in Cryptology – CRYPTO 2001. pp. 101–118 (2001)
36. Hirt, M., Maurer, U., Przydatek, B.: Efficient secure multi-party computation. In: Advances in Cryptology – ASIACRYPT 2000. pp. 143–161 (2000)
37. Hirt, M., Tschudi, D.: Efficient general-adversary multi-party computation. In: Advances in Cryptology – ASIACRYPT 2013, Part II. pp. 181–200 (2013)
38. Ishai, Y., Kushilevitz, E., Meldgaard, S., Orlandi, C., Paskin-Cherniavsky, A.: On the power of correlated randomness in secure computation. In: Theory of Cryptography. pp. 600–620 (2013)
39. Jukna, S.: Boolean Function Complexity. Springer, Berlin, Heidelberg, 1 edn. (2012)
40. Keller, H., Orlandi, C., Paskin-Cherniavsky, A., Ravi, D.: MPC with low bottleneck-complexity: Information-theoretic security and more. In: 4th Information-Theoretic Cryptography (ITC) Conference (2023), https://eprint.iacr.org/2023/683
41. Keller, M.: MP-SPDZ: A versatile framework for multi-party computation. In: Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security. pp. 1575–1590. CCS '20 (2020)
42. King, V., Saia, J., Sanwalani, V., Vee, E.: Scalable leader election. In: SODA. vol. 6, pp. 990–999 (2006)
43. McEliece, R.J., Sarwate, D.V.: On sharing secrets and Reed-Solomon codes. Communications of the ACM **24**(9), 583–584 (1981)
44. Orlandi, C., Ravi, D., Scholl, P.: On the bottleneck complexity of mpc with correlated randomness. In: Public-Key Cryptography – PKC 2022, Part I. pp. 194–220 (2022)
45. Rabin, T., Ben-Or, M.: Verifiable secret sharing and multiparty protocols with honest majority. In: Proceedings of the Twenty-First Annual ACM Symposium on Theory of Computing. pp. 73–85. STOC '89 (1989)
46. Shoup, V.: A computational introduction to number theory and algebra. Cambridge university press (2009)
47. van Dijk, M., Gentry, C., Halevi, S., Vaikuntanathan, V.: Fully homomorphic encryption over the integers. In: Advances in Cryptology – EUROCRYPT 2010. pp. 24–43 (2010)
48. Yamamoto, H.: Secret sharing system using (k, L, n) threshold scheme. Electronics and Communications in Japan (Part I: Communications) **69**(9), 46–54 (1986)
49. Yao, A.C.: Protocols for secure computations. In: Proceedings of the 23rd Annual Symposium on Foundations of Computer Science. pp. 160–164. SFCS '82 (1982)

A A Flaw in the Protocol for $\mathcal{F}_{\mathsf{Sum}}$ from ITC 2024

In the preliminary version of this paper [22], we claimed that there exists a protocol Π realizing $\mathcal{F}_{\mathsf{Sum},\mathbb{G}}$ such that $\mathrm{CR}(\Pi) = 0$ and $\mathrm{BC}(\Pi) = O(\log|\mathbb{G}|)$, and used Π as a building block for our main constructions. However, we found a security flaw in the protocol Π and consequently, Π does not securely realize the target functionality $\mathcal{F}_{\mathsf{Sum},\mathbb{G}}$. More specifically, Π is described in Fig. 6. Suppose that an adversary corrupts players P_1 and P_3. Then, the adversary learns $z_2 = z_1 + y_2 = x_1 + r_1 + x_2 + r_2$ at Step 3 and also learns $w_2 = w_1 - r_2$ at Step 6. Since the adversary knows x_1, r_1, and w_1, she can compute $r_2 = w_1 - w_2$ and hence $x_2 = z_2 - x_1 - r_1 - r_2$, which reveals more information than the sum s. Currently, we are not able to fix the flaw in Π or to find an alternative protocol that securely realizes $\mathcal{F}_{\mathsf{Sum},\mathbb{G}}$ with the same level of efficiency.

Protocol Π

Input. Each player P_i has a group element $x_i \in \mathbb{G}$.
Output. Every player obtains $s = \sum_{i\in[n]} x_i$.
Protocol.

1. Each player P_i chooses $r_i \leftarrow_{\$} \mathbb{G}$ and sets $y_i = x_i + r_i$.
2. P_1 sends y_1 to P_2.
3. For each $i = 2, 3, \ldots, n-1$, P_i lets z_{i-1} be the message from P_{i-1}, computes $z_i = z_{i-1} + y_i$, and sends z_i to P_{i+1}.
4. P_n sends $z_n = z_{n-1} + y_n$ to P_1.
5. P_1 sends $w_1 = z_n - r_1$ to P_2.
6. For each $i = 2, 3, \ldots, n-1$, P_i lets w_{i-1} be the message from P_{i-1}, computes $w_i = w_{i-1} - r_i$, and sends w_i to P_{i+1}.
7. P_n computes $s = w_{n-1} - r_n$ and invokes $\mathcal{F}_{\mathsf{Broadcast},n}$ with input s.
8. Each player P_i outputs s.

Fig. 6. The protocol from ITC 2024 [22]