Efficient Multiparty Private Simultaneous Messages for Symmetric Functions

Reo Eriguchi[1][0000−0002−0019−6934] and Kazumasa Shinagawa[2,1][0000−0002−5219−1975]

[1] National Institute of Advanced Industrial Science and Technology, Tokyo, Japan
eriguchi-reo@aist.go.jp
[2] University of Tsukuba, Ibaraki, Japan
shinagawa@cs.tsukuba.ac.jp

Abstract. A *Private Simultaneous Messages* (PSM) protocol is a secure multiparty computation protocol with a minimal interaction pattern, which allows input parties sharing common randomness to securely reveal the output of a function by sending messages only once to an external party. Since existing PSM protocols for arbitrary functions have exponentially large communication complexity in the number n of parties, it is important to explore efficient protocols by focusing on special functions of practical use. In this paper, we study the communication efficiency of PSM protocols for *symmetric* functions, which provide many useful functionalities for real-world applications. We present a new n-party PSM protocol for symmetric functions with communication complexity $n^{2d/3+O(1)}$, where d is the size of the input domain of each party. Our protocol improves the currently best known communication complexity of $n^{d+O(1)}$. As applications to other related models, we show that our novel protocol implies improved communication complexity of *ad-hoc* PSM, where only a subset of parties actually send messages, and also leads to a more communication-efficient *robust* PSM protocol, which is secure against collusion of the external party and input parties. The extension to ad-hoc PSM is not a straightforward application of the previous transformation but includes an optimization technique based on the symmetry of functions.

Keywords: Secure multiparty computation · Symmetric functions

1 Introduction

Private Simultaneous Messages (PSM) [12, 15] protocols are secure multiparty computation (MPC) protocols with a minimal interaction pattern. In an n-party PSM protocol for a function f, n parties each have a private input x_i and share common randomness. Each party computes a message based on their input and randomness, and sends it to an external party called a referee, who does not know the randomness. Then, the referee can compute $f(x_0, \ldots, x_{n-1})$ from messages without learning any information on the inputs. Since there is no interaction between the parties, PSM protocols are effective especially when parties cannot

simultaneously participate in the protocols. Furthermore, efficient PSM protocols can be used to obtain many other cryptographic primitives, e.g., MPC with general interaction patterns [13], constant-round MPC [18, 17] and conditional disclosure of secrets [19, 20]. The model of PSM protocols has been generalized into different scenarios such as ad-hoc PSM [4, 7] where only a subset of the parties actually send messages, and robust PSM (also known as non-interactive MPC) [5] where the referee may collude with some parties.

Communication complexity, which is defined as the total length of messages, is considered as the most important metric to measure the efficiency of PSM protocols. For arbitrary functions, Feige et al. [12] presented a 2-party PSM protocol with communication complexity $O(d)$, where the input domain of each party is $[d] := \{0, 1, \ldots, d-1\}$ and the output domain is $\{0, 1\}$. Subsequently, Ishai [16, Section 4.1] presented an n-party PSM protocol with communication complexity $O(d^n)$. Important results were the 2-party PSM protocol with communication complexity $O(\sqrt{d})$ in [6] and the n-party PSM protocol with communication complexity $O(d^{n/2})$ in [8, 2] (omitting factors depending solely on d or n). The importance of these results lies in not just improving communication complexity but breaking the representation-size barrier, which is $\Omega(d^n)$ in the worst-case for arbitrary functions. Note that the prior protocols of [12, 16] used a representation of f in some computation model and consequently, the communication complexity cannot be smaller than the representation size of worst-case functions.

Despite progress, PSM protocols for computing arbitrary functions leave much to be desired as the communication complexity of all prior protocols grows exponentially in the number n of parties. An important research direction is thus to explore communication-efficient PSM protocols by focusing on special functions of practical use. Above all, *symmetric functions*, whose values are the same no matter the order of n inputs, are one of the most fundamental classes of functions with many real-world applications such as election and statistical surveys. Several PSM protocols for symmetric functions were proposed in [5, 9, 11, 23] and the currently best known communication complexity is $n^{d+O(1)}$ [11]. These PSM protocols are more efficient than those for arbitrary functions if the number n of parties is larger than the domain size d (e.g., in the election scenario, the number of voters is much larger than the number of candidates). However, in contrast to the case of arbitrary functions, no prior protocol for symmetric functions was able to break the representation-size barrier, which is $n^{d+\Omega(1)}$ since each function can be expressed by specifying the outputs for all d-bin histograms of inputs. Note that the prior protocols used a representation of symmetric functions based on branching programs or truth tables, and thus a novel approach is needed to break the above barrier. Motivated by this, we ask the following question:

Is it possible to construct an n-party PSM protocol for symmetric functions with communication complexity $O(n^{cd})$ for $c < 1$?

1.1 Our Results

In this paper, we answer the above question affirmatively by presenting an n-party PSM protocol for symmetric functions with communication complexity $n^{2d/3+O(1)}$ (see Table 1). Our protocol is the first PSM protocol that breaks the representation-size barrier for symmetric functions. Furthermore, our efficiency improvement implies better communication complexity for PSM in other related models and we obtain more communication-efficient ad-hoc PSM and robust PSM protocols for symmetric functions than the respective state-of-the-art protocols. We elaborate our results below.

Table 1. Comparison of PSM protocols for symmetric functions

Protocol	Communication	Computation
[8]	$O(d^{n/2}n^3)$	$O(d^{n/2}n^3)$
[2]	$d^{(n-1)/2}2^{O(n)}$	$d^{(n-1)/2}2^{O(n)\dagger}$
[9]	$(n+1)^{d\log n+O(1)}$	N/A‡
[5]	$O((n+1)^{2d+1})$	$O((n+1)^{2d+1})$
[11]	$O((n+1)^{d+1}d^2\log n)$	$O((n+1)^{d+1}d^2\log n)$
Ours (Corollary 1)	$O(n^{2\lfloor d/3\rceil+1})$	$2^{O(n^{2\lfloor d/3\rceil})}$
Ours (Corollary 2)	$O((n+1)^{2\lfloor d/3\rceil+1}d\log n)$	$(n+d)^{O(d)}$

n is the number of parties and d is the size of the input domain of each party. $\lfloor x\rceil$ is the nearest integer to a number x.

† The protocol [2] applies to infinitely many values of n.

‡ The paper [9] does not provide an analysis of computational complexity.

Generic Construction from PSM for Smaller Functions. Our main technical contribution is devising a generic method to construct a PSM protocol for a symmetric function f from any PSM protocol with smaller input domains. Concretely, we decompose a d-bin histogram of n inputs into several histograms with fewer bins, and reduce the computation of f to the computation of several symmetric functions with input domain of size $d' < d$. Due to the privacy requirements, PSM protocols for the smaller functions should satisfy a special property of *universal reconstruction*, which requires that the decoding algorithm run by the referee should be oblivious to the descriptions of the functions. Although the idea of decomposing a target function into smaller ones was used in the previous works [8, 2], our approach is orthogonal to theirs in that the constructions in [8, 2] decomposed the set of n parties into several smaller subsets of size $n' < n$ while we decompose the input domain.

Communication-efficient PSM with Universal Reconstruction. The second step is to realize PSM with universal reconstruction for smaller symmetric

functions with input domain of size d'. Specifically, we propose a protocol whose communication complexity is $O(n^{d'})$. While the prior PSM protocols of [5, 11] also have universal reconstruction, our protocol is more communication-efficient by a multiplicative factor of $O(d^2 n \log n)$. Technically, our construction is based on the number-theoretic fact in [21]. We encode a histogram of inputs into an integer and reduce the computation of the function to the computation of quadratic residues over a sufficiently large prime. Our approach is inspired by the previous construction in [23], but it was only able to deal with boolean inputs while we devise a more general and efficient encoding of histograms of non-boolean inputs. We also show that our protocol achieves almost optimal communication complexity by showing a lower bound $\Omega(n^{d'-1})$ for PSM with universal reconstruction. A possible drawback of the above protocol is its high computational overhead when encoding the function to an array of quadratic residues, resulting in a final protocol with computational complexity $2^{O(n^{2\lfloor d/3 \rceil})} = \exp(n^d)$. This issue can be mitigated by using an optimized version of the protocol [11], reducing the computational overhead to $(n+d)^{O(d)} = \text{poly}(n^d)$, albeit at the cost of slightly increasing communication complexity.

Applications to Related Models. Our efficiency improvements also imply improvements in the state-of-the-art complexity of ad-hoc PSM and robust PSM protocols for symmetric functions. It is possible to directly apply the transformation methods from standard PSM to ad-hoc PSM in [7] or to robust PSM in [9], which already leads to more communication-efficient protocols in the respective models. Besides these direct implications, we further optimize the construction of ad-hoc PSM in [7] by making use of the symmetry of functions, and obtain a more efficient ad-hoc PSM protocol than simply applying the original technique of [7].

We will give more details on our techniques in Section 2.

1.2 Related Works

We briefly provide a historical perspective on the problem of improving the communication complexity of PSM protocols. Feige et al. [12] formalized the notion of PSM protocols and presented a 2-party PSM protocol with communication complexity $O(d)$ where each party has d possible inputs. Ishai and Kushilevitz [15] extended this to the n-party setting and showed an n-party PSM protocol for functions represented by branching programs. These constructions, along with several variants, are summarized in [16], all of which have worst-case communication complexity $O(d^n)$. Subsequently, Beimel et al. [6] successfully devised a 2-party PSM protocol for general functions with communication complexity $O(\sqrt{d})$, which was later generalized and improved by [8, 2], leading to n-party PSM protocols with communication complexity $O(d^{n/2})$ (omitting factors depending only on d or n).

While the above general constructions can be applied to symmetric functions, several "tailor-made" approaches have been proposed to obtain more efficient

protocols [5, 9, 11, 23]. Concretely, these specialized constructions yield PSM protocols for symmetric functions with communication complexity $n^{cd+O(1)}$ for some $c \geq 1$. These protocols significantly improve the communication complexity for arbitrary functions when $n \gg d$. Our work follows this direction and, for the first time, achieves communication complexity of $n^{cd+O(1)}$ with $c < 1$. We note that PSM protocols for boolean symmetric functions (i.e., $d = 2$) were proposed in [23] but they cannot deal with non-boolean inputs.

There are known lower bounds on the communication complexity of PSM. Applebaum et al. [1] proved a lower bound $3 \log d - O(\log \log d)$ for 2-party PSM for a randomly chosen function, and Ball and Randolph [3] showed a lower bound that is roughly $n^2 \log d$ for n-party PSM with $n = \omega(\log \log d)$. Our lower bound (Theorem 2) does not improve the previously known lower bounds since ours is only applied to PSM protocols satisfying the stronger privacy guarantee of universal reconstruction. Tighter upper/lower bounds are known for concrete functions with small input domains and/or a small number of parties, including the n-bit AND function [12, 10, 24], the equality function [24], and the majority function [24].

2 Technical Overview

2.1 Generic Construction from PSM for Smaller Functions

Encoding. Let f be a symmetric function with input domain $[d] = \{0, 1, \ldots, d-1\}$, that is, a function value $f(x_0, \ldots, x_{n-1})$ is independent of the order of inputs $\boldsymbol{x} = (x_i)_{i \in [n]}$. This means that $f(\boldsymbol{x})$ is uniquely determined by the *histogram* $\boldsymbol{h} = (h_0, \ldots, h_{d-1})$ of $\boldsymbol{x}$, where h_j counts the frequency of a value j in $\boldsymbol{x}$, i.e., $h_j = |\{i \in [n] : x_i = j\}|$. We encode the histogram into a non-negative integer $\hat{h}$. We need an encoding such that (1) the correspondence is one-to-one for correctness and (2) the maximum size of corresponding integers is as small as possible for efficiency. A simple encoding is mapping each histogram $\boldsymbol{h} = (h_0, \ldots, h_{d-1})$ into an integer whose base-$(n+1)$ representation is $(h_0, \ldots, h_{d-1})$, i.e., $\hat{h} = \sum_{j=0}^{d-1} h_j (n+1)^j \in O((n+1)^d)$. With optimization, we devise an encoding method such that $\hat{h} \in O(n^d)$ for all histograms $\boldsymbol{h}$. Let Φ_d be the set of the $\hat{h}$'s for all histograms $\boldsymbol{h}$. A symmetric function f can then be described as a one-dimensional array S_f of size $O(n^d)$ such that the $\hat{h}$-th entry of S_f, denoted by $S_f(\hat{h})$, is equal to $f(\boldsymbol{x})$ for any input $\boldsymbol{x}$ and its corresponding histogram $\boldsymbol{h}$.

PSM with Universal Reconstruction. *Universal reconstruction* [6] is a special property of PSM protocols which requires that the decoding algorithm run by a referee is oblivious to the descriptions of functions. In this section, we assume that we are given a PSM protocol Π_d with universal reconstruction for symmetric functions over $[d]^n$. In the next section, we will show a concrete realization of Π_d such that $\mathrm{Comm}(\Pi_d) = O(n^d)$, where $\mathrm{Comm}(\Pi)$ denotes the communication complexity of Π.

Main construction. Our high-level approach is to decompose the computation of a function f into those of smaller functions with input domain $[d']$ for $d' < d$. We first decompose the histogram $\boldsymbol{h}$ of inputs into smaller histograms. Each histogram $\boldsymbol{h} = (h_0, \ldots, h_{d-1})$ is decomposed into three $d/3$-bin histograms $(\boldsymbol{h}^{(0)}, \boldsymbol{h}^{(1)}, \boldsymbol{h}^{(2)})$ such that

$$\boldsymbol{h}^{(0)} = (h_0, \ldots, h_{d/3-1}),\ \boldsymbol{h}^{(1)} = (h_{d/3}, \ldots, h_{2d/3-1}),\ \boldsymbol{h}^{(2)} = (h_{2d/3}, \ldots, h_{d-1}),$$

where we assume that d is a multiple of 3. Accordingly, the integer $\hat{h} \in \Phi_d$ encoding $\boldsymbol{h}$ can be decomposed into three smaller integers $(\hat{h}^{(0)}, \hat{h}^{(1)}, \hat{h}^{(2)})$ where $\hat{h}^{(j)} \in \Phi_{d/3}$ encodes the histogram $\boldsymbol{h}^{(j)}$. Furthermore, we can transform the one-dimensional array S_f representing f to a three-dimensional array T_f:

$$T_f(\hat{h}^{(0)}, \hat{h}^{(1)}, \hat{h}^{(2)}) = f(\boldsymbol{x})$$

for all inputs $\boldsymbol{x}$.

If we do not care about privacy, it is sufficient to send these integers $(\hat{h}^{(0)}, \hat{h}^{(1)}, \hat{h}^{(2)})$ to the referee. To mask $\hat{h}^{(j)}$, we give input parties a uniformly random subset $R_j \subseteq \Phi_{d/3}$ as common randomness, and let the referee obtain a set $R_j \triangle \{\hat{h}^{(j)}\}$, where $A \triangle B$ is the symmetric difference of sets A, B. Concretely, $R_j \triangle \{\hat{h}^{(j)}\} = R_j \setminus \{\hat{h}^{(j)}\}$ if $\hat{h}^{(j)} \in R_j$ and $R_j \triangle \{\hat{h}^{(j)}\} = R_j \cup \{\hat{h}^{(j)}\}$ if $\hat{h}^{(j)} \notin R_j$. Specifically, we consider a function g_j that outputs $R_j \triangle \{\hat{h}^{(j)}\}$, where R_j is hard-wired to the description of g_j. We can see that g_j is symmetric since it depends only on a histogram $\boldsymbol{h}^{(j)}$, and that the size of each input domain is $d/3$ instead of d since $\boldsymbol{h}^{(j)}$ has only $d/3$ bins. Since the range of g_j is $\Phi_{d/3}$, we have a PSM protocol with universal reconstruction for computing g_j whose communication complexity is $|\Phi_{d/3}| \cdot \mathrm{Comm}(\Pi_{d/3}) = O(n^{d/3} \cdot \mathrm{Comm}(\Pi_{d/3}))$. It is important to note that the property of universal reconstruction is essential here: We should not tell the referee the description of the function g_j to keep R_j hidden from him.

Upon receiving $R_j \triangle \{\hat{h}^{(j)}\}$, the referee computes

$$T_f(R_0 \triangle \{\hat{h}^{(0)}\}, R_1 \triangle \{\hat{h}^{(1)}\}, R_2 \triangle \{\hat{h}^{(2)}\}), \tag{1}$$

where we denote $T_f(A, B, C) = \sum_{(j_0, j_1, j_2) \in A \times B \times C} T_f(j_0, j_1, j_2)$. To extract a desired output from (1), we observe that

$$\begin{aligned}
&T_f(R_0 \triangle \{\hat{h}^{(0)}\}, R_1 \triangle \{\hat{h}^{(1)}\}, R_2 \triangle \{\hat{h}^{(2)}\}) \\
&= T_f(\hat{h}^{(0)}, \hat{h}^{(1)}, \hat{h}^{(2)}) \\
&\quad + T_f(\{\hat{h}^{(0)}\}, \{\hat{h}^{(1)}\}, R_2) + T_f(\{\hat{h}^{(0)}\}, R_1, \{\hat{h}^{(2)}\}) + T_f(R_0, \{\hat{h}^{(1)}\}, \{\hat{h}^{(2)}\}) \\
&\quad + T_f(\{\hat{h}^{(0)}\}, R_1, R_2) + T_f(R_0, \{\hat{h}^{(1)}\}, R_2) + T_f(R_0, R_1, \{\hat{h}^{(2)}\}) \\
&\quad + T_f(R_0, R_1, R_2),
\end{aligned} \tag{2}$$

where the addition is modulo 2. The last term in the right-hand side of (2) can be securely revealed to the referee with $O(1)$ communication since it is

independent of inputs and any input party (e.g., the first party) is able to compute it on his own. To reveal the second term, we consider a function $g_{\{0,1\}}$ that outputs $T_f(\{\hat{h}^{(0)}\}, \{\hat{h}^{(1)}\}, R_2)$, where R_2 is hard-wired to the description of $g_{\{0,1\}}$. Note that $g_{\{0,1\}}$ can be viewed as a symmetric function over $[2d/3]^n$ since it depends only on a $2d/3$-bin histogram $(\boldsymbol{h}^{(0)}, \boldsymbol{h}^{(1)})$. Thus, the second term can be securely revealed to the referee with communication complexity $\mathrm{Comm}(\Pi_{2d/3})$ by running a PSM protocol with universal reconstruction for computing $g_{\{0,1\}}$. Again, the property of universal reconstruction is essential to keep the mask R_2 hidden. Similarly, the third and fourth terms can be revealed with the same communication complexity. Each of the rest of the terms depends only on a $d/3$-bin histogram and hence can be securely computed with communication complexity $\mathrm{Comm}(\Pi_{d/3})$. Then, the referee can obtain the target term $T_f(\hat{h}^{(0)}, \hat{h}^{(1)}, \hat{h}^{(2)}) = f(\boldsymbol{x})$. The total communication complexity is $O(n^{d/3}) \cdot \mathrm{Comm}(\Pi_{d/3}) + O(1) \cdot \mathrm{Comm}(\Pi_{2d/3}) = O(n^{2d/3})$.

We note that the above protocol does not achieve full security since the referee learns non-target terms (e.g., $T_f(\{\hat{h}^{(0)}\}, \{\hat{h}^{(1)}\}, R_2)$). These leakages can be compensated by revealing terms masked with additive shares of 0 (e.g., $T_f(\{\hat{h}^{(0)}\}, \{\hat{h}^{(1)}\}, R_2) + s_{\{0,1\}}$) to the referee. These shares are included in common randomness.

It is natural to ask whether the above construction can be optimized by decomposing d-bin histograms into d/k-bin histograms for $k \neq 3$. From a similar argument, we obtain a PSM protocol whose communication complexity is at most

$$k \cdot |\Phi_{d/k}| \cdot \mathrm{Comm}(\Pi_{d/k}) + 2^k \cdot \mathrm{Comm}(\Pi_{d(k-1)/k}) = n^{\delta + O(1)},$$

where $\delta = \max\{2d/k, d(k-1)/k\}$. Thus, the choice of $k = 3$ is optimal.

2.2 Realization of Π_d

We show a PSM protocol Π_d with universal reconstruction for symmetric functions over $[d]^n$. As in the previous section, we encode a histogram $\boldsymbol{h} = (h_0, \ldots, h_{d-1})$ of inputs into a non-negative integer $\hat{h} \in \Phi_d$. For ease of exposition, we here consider the simple encoding $\hat{h} = \sum_{j=0}^{d-1} h_j (n+1)^j$ using the base-$(n+1)$ representation. Recall that each symmetric function f can be described as an $O(n^d)$-sized array S_f, that is, $S_f(\hat{h}) = f(\boldsymbol{x})$ for any input $\boldsymbol{x}$ and its corresponding histogram $\boldsymbol{h}$.

Our construction of Π_d is based on the number-theoretic fact in [21]: For any positive integer ℓ, there exists a prime $p = 2^{O(\ell)}$ such that

($\star$) The sequence of quadratic characters modulo p contains every array S of size ℓ as a sub-sequence.

That is, for every S, there exists an offset $a \in \mathbb{Z}$ such that

$$\left(\frac{a+j}{p}\right) = (-1)^{S(j)}, \ \forall j = 0, 1, \ldots, \ell - 1,$$

where $\left(\frac{x}{p}\right)$ is the Legendre symbol (see Section 3.2 for its definition and basic properties). It is important to note that the choice of p is independent of the array S and the sequence of quadratic residues contains *all* arrays of size ℓ as sub-sequences.

Our PSM protocol works as follows. For $\ell = |\Phi_d|$, we fix a prime p with the property $(\star)$ and find an offset a corresponding to S_f. Let r be a uniformly random quadratic residue and reveal $r(a + \hat{h})$ to the referee. The correctness follows from

$$\left(\frac{r(a+\hat{h})}{p}\right) = \left(\frac{a+\hat{h}}{p}\right) = (-1)^{S_f(\hat{h})}.$$

The privacy follows since the referee only learns $S_f(\hat{h})$ due to the mask r. To securely reveal $r(a + \hat{h})$, each input party encodes x_i into an integer $\hat{x}_i$ whose base-$(n+1)$ representation is $(\underbrace{0, \ldots, 0}_{x_i}, 1, \underbrace{0, \ldots, 0}_{d-x_i-1})$, i.e., $\hat{x}_i = (n+1)^{x_i}$. We define a message of party i as

$$m_i = \begin{cases} r(\hat{x}_0 + a) + r_0, & \text{if } i = 0, \\ r\hat{x}_i + r_i, & \text{if } i \neq 0, \end{cases}$$

where $(r_i)_{i\in[n]}$ is an additive sharing of 0 and multiplication and addition are modulo p. Then, it holds that $\sum_{i\in[n]} m_i = r \sum_{i\in[n]} \hat{x}_i + ra = r(\hat{h} + a)$ and the messages m_i's reveal nothing other than that. The communication complexity is $\mathrm{Comm}(\Pi_d) = O(n \log p) = O(n|\Phi_d|) = O(n^{d+1})$. We can optimize the above construction by using the fact that a histogram $\boldsymbol{h} = (h_0, \ldots, h_{d-1})$ must satisfy the constraint that $\sum_{j\in[d]} h_j = n$. Therefore, we can ignore the last bin of $\boldsymbol{h}$, resulting in a PSM protocol Π_d with communication complexity $O(n^d)$.

We briefly explain why the above protocol has universal reconstruction. The prime p depends only on $|\Phi_d|$, independent of each function f. Furthermore, the algorithm run by the referee is oblivious to the description of f since he only performs the addition modulo p and tests if the sum is a quadratic residue or not.

We can show that our protocol achieves almost optimal communication complexity by showing an almost matching lower bound, which is $\Omega(n^{d-1})$ for $d = O(1)$. The gap is only a multiplicative factor of n. The lower bound is derived by extending the technique of [6, Appendix H] to the multi-party setting (The original technique of [6] was applied to the two-party setting).

A possible drawback of the above protocol is that it is not computationally efficient. When generating common randomness, finding the offset a corresponding to the array S_f results in large computation complexity $O(p) = 2^{O(n^d)}$. (Note that finding the prime p is efficiently done since any sufficiently large prime $p = 2^{\Omega(\ell)}$ satisfies the property $(\star)$ [21].) To compensate this drawback, we also show a computationally efficient PSM protocol with universal reconstruction at

the cost of slightly increasing communication complexity, by optimizing the PSM protocol in [11].

We note that our idea of using the properties of quadratic residues is inspired by [16, 23]. Our technical novelty is a more optimized encoding of histograms $\boldsymbol{h}$ into integers $\hat{h} \in \Phi_d$, which leads to reducing the size of a prime p. Our encoding is different from that of [16], which did not leverage the symmetry of inputs and resulted in a prime of exponential bit-length, i.e., $p = 2^{O(2^n)}$. It is also different from that of [23], which was not applicable to non-binary inputs.

2.3 Applications to Related Models

A k-out-of-n ad-hoc PSM protocol [7] is an n-party PSM protocol where only a subset S of k input parties, where S is not known in advance, actually sends messages and the referee is able to evaluate a k-input function f on the subset of inputs $(x_i)_{i \in S}$. Ad-hoc PSM can be realized trivially by preparing the common randomness of k-party PSM for f independently for each potential subset, but it results in a large communication overhead $\binom{n}{k}$.

Beimel et al. [7] proposed a more efficient transformation for this problem. Roughly speaking, consider a *related* n-input function $g : ([d] \cup \{\perp\})^n \to \{0, 1, \perp\}$ such that $g(x_0, \ldots, x_{n-1}) = f((x_i)_{i \in S})$ if $S = \{i \in [n] : x_i \neq \perp\}$ is of size k and $g(x_0, \ldots, x_{n-1}) = \perp$ otherwise. Then, a k-out-of-n ad-hoc PSM protocol for f can be efficiently constructed from an n-party PSM protocol for g. Their key technique is that when a set S of k parties send messages, the referee is able to extract a message encoding $\perp$ for every party $i \notin S$ who is not participating in the protocol. Then, the referee obtains $g((x_i)_{i \in S}, (\perp)_{i \notin S}) = f((x_i)_{i \in S})$ via the decoding algorithm of the protocol for g.

It is possible to obtain an ad-hoc version of our PSM protocol by using the transformation of [7] but there is still room for improving efficiency since the size of the input domain necessarily increases to deal with an additional input $\perp$, which results in increasing communication complexity by a factor of $n^{2/3}$. Instead, our ad-hoc PSM protocol is based on a different function $g' : [d]^n \to \{0, 1\}$ which involves no additional input. Our technical novelty is showing that if f is symmetric, a similar technique to [7] is still applicable to this function g' and as a result, an ad-hoc PSM protocol for f can be obtained more efficiently than simply applying the original technique of [7].

Technically, fix any input $a \in [d]$ (e.g., $a = 0$) and consider a symmetric function $g' : [d]^n \to \{0, 1\}$ such that if $S' = \{i \in [n] : x_i \neq a\}$ is of size less than or equal to k, then $g'(x_0, \ldots, x_{n-1}) = f((x_i)_{i \in S'}, \underbrace{a, \ldots, a}_{k - |S'|})$ and otherwise, we do not care about what g outputs. The idea is to use messages encoding a instead of those encoding $\perp$ for $n - k$ non-participating parties. That is, the referee is able to extract a message encoding a for every non-participating party. Since f and g' are symmetric, if a set S of k parties actually send messages, the referee

obtains the correct result

$$g'((x_i)_{i\in S}, \underbrace{a,\ldots,a}_{n-k}) = g'((x_i)_{i\in S'}, \underbrace{a,\ldots,a}_{n-|S'|}) = f((x_i)_{i\in S'}, \underbrace{a,\ldots,a}_{k-|S'|}) = f((x_i)_{i\in S}),$$

where $S' = \{i \in S : x_i \neq a\}$. The resulting ad-hoc PSM protocol for f reduces communication complexity by a factor of $n^{2/3}$ compared to simply applying the original technique of [7]. In particular, we obtain an ad-hoc PSM protocol for a boolean symmetric function (i.e., $d = 2$) with communication complexity $O(n^3)$, improving on the best prior protocol in [7] that has communication complexity $O(n^4k^4)$.

Note that our optimization technique is not applicable if f is not symmetric. For example, if out of three parties, parties 0 and 1 actually send messages encoding a and x_1, respectively, then the protocol should output $f(a, x_1)$. On the other hand, if parties 1 and 2 send messages encoding x_1 and a, respectively, then the output should be $f(x_1, a)$. However, the protocol resulting from our technique outputs $g'(a, x_1, a)$ in both cases, which is equal to $f(x_1, a)$ by the definition of g' but may not be equal to $f(a, x_1)$ if f is not symmetric.

3 Preliminaries

Notations. We denote the set of all positive integers by $\mathbb{N}$ and the set of all non-negative integers by $\mathbb{Z}_{\geq 0}$. For a positive integer $\ell \in \mathbb{N}$, we define $[\ell] = \{0, 1, \ldots, \ell - 1\}$. For a real number x, we denote the nearest integer to x by $\lfloor x \rceil$. For a prime p, we denote the ring of integers modulo p by $\mathbb{Z}_p$ and identify it with $\{a \in \mathbb{Z} : 0 \leq a \leq p - 1\}$ as a set. We denote the multiplicative group of integers modulo p by $\mathbb{Z}_p^*$ and identify it with $\{a \in \mathbb{Z} : 1 \leq a \leq p - 1\}$ as a set. We denote the binary field by $\mathbb{F} = \{0, 1\}$, which can be naturally embedded into $\mathbb{Z}$ as a set. For two random variables X, Y, we write $X \equiv Y$ if they are identical with each other. We use bold letters for vectors (e.g., $\boldsymbol{x}$). Unless stated otherwise, we suppose that the indexing set of an ℓ-dimensional vector $\boldsymbol{x}$ is $[\ell]$ and denote the i-th entry by $\boldsymbol{x}(i)$ for $i \in [\ell]$. We denote the Hamming weight of a vector $\boldsymbol{x} \in \{0, 1\}^\ell$ by $\mathrm{wt}(\boldsymbol{x})$, i.e., $\mathrm{wt}(\boldsymbol{x}) = |\{i \in [\ell] : \boldsymbol{x}(i) \neq 0\}|$. We denote the inner product of two ℓ-dimensional vectors $\boldsymbol{x}, \boldsymbol{y}$ by $\langle \boldsymbol{x}, \boldsymbol{y} \rangle = \sum_{i\in[\ell]} \boldsymbol{x}(i)\boldsymbol{y}(i)$. For a subset $S \subseteq \mathbb{Z}$, we denote the largest integer in S by $\max S$.

Throughout the paper, we fix the following notations:

- $n \in \mathbb{N}$ denotes the number of input parties.
- $d \in \mathbb{N}$ denotes the cardinality of the input domain.

3.1 Private Simultaneous Messages

A Private Simultaneous Messages (PSM) protocol allows n input parties each holding an input x_i to reveal $f(x_0, \ldots, x_{n-1})$ to an external party called a referee without revealing any extra information.

Definition 1. *Let $X_0, \ldots, X_{n-1}, Y$ be finite sets. Let $\mathcal{F}$ be a set of functions from $\prod_{i\in[n]} X_i$ to Y. An n-party private simultaneous messages (PSM) protocol Π for $\mathcal{F}$ is a tuple of algorithms $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$, where:*

- $\mathsf{Gen}(f) \to r$*:* Gen *is a randomized algorithm that takes a description of $f \in \mathcal{F}$ as input and outputs common randomness $r \in \{0,1\}^{\mathsf{rand}}$.*
- $\mathsf{Enc}(i, f, x_i, r) \to m_i$*:* Enc *is a deterministic algorithm that takes an index i, a description of $f \in \mathcal{F}$, $x_i \in X_i$, and $r \in \{0,1\}^{\mathsf{rand}}$ as input and outputs a message $m_i \in \{0,1\}^{\mathsf{comm}_i}$.*
- $\mathsf{Dec}(f, (m_i)_{i\in[n]}) \to y$*:* Dec *is a deterministic algorithm that takes a description of $f \in \mathcal{F}$ and $(m_i)_{i\in[n]} \in \prod_{i\in[n]}\{0,1\}^{\mathsf{comm}_i}$ as input and outputs $y \in Y$.*

satisfying the following properties:

Correctness. *For any $f \in \mathcal{F}$ and any $(x_i)_{i\in[n]} \in \prod_{i\in[n]} X_i$, it holds that*

$$\mathsf{Dec}(f, (\mathsf{Enc}(i, f, x_i, r))_{i\in[n]}) = f(x_0, \ldots, x_{n-1}), \tag{3}$$

with probability 1 over the choice of $r \leftarrow \mathsf{Gen}(f)$.

Privacy. *There exists a randomized algorithm* Sim *such that for any $f \in \mathcal{F}$, it holds that*

$$(\mathsf{Enc}(i, f, x_i, r))_{i\in[n]} \equiv \mathsf{Sim}(f, f(x_0, \ldots, x_{n-1})), \tag{4}$$

where the probabilities are taken over $r \leftarrow \mathsf{Gen}(f)$ and the random coins of Sim*.*

The communication complexity of Π is defined as $\mathrm{Comm}(\Pi) = \sum_{i\in[n]} \mathsf{comm}_i$ and the randomness complexity of Π is defined as $\mathrm{Rand}(\Pi) = \mathsf{rand}$. The computational complexity of Π is defined as the running time of one execution of Π with worst-case inputs $(f, (x_i)_{i\in[n]})$, i.e., the running time of the following procedures:

1. Compute $r \leftarrow \mathsf{Gen}(f)$.
2. For each $i \in [n]$, compute $m_i \leftarrow \mathsf{Enc}(i, f, x_i, r)$.
3. Compute $y \leftarrow \mathsf{Dec}(f, (m_i)_{i\in[n]})$.

We see communication and randomness complexity as a primary efficiency measure and computational complexity as a secondary measure.

We say that $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is a PSM protocol for $\mathcal{F}$ *with universal reconstruction* if it satisfies the above requirements except that Dec and Sim are not allowed to take $f \in \mathcal{F}$ as input. Accordingly, the conditions (3) and (4) are replaced with

$$\mathsf{Dec}((\mathsf{Enc}(i, f, x_i, r))_{i\in[n]}) = f(x_0, \ldots, x_{n-1}) \tag{5}$$

and

$$(\mathsf{Enc}(i, f, x_i, r))_{i\in[n]} \equiv \mathsf{Sim}(f(x_0, \ldots, x_{n-1})), \tag{6}$$

respectively.

3.2 Quadratic Residues

Let p be a prime. We say that $a \in \mathbb{Z}_p^*$ is a quadratic residue modulo p, denoted by $\left(\frac{a}{p}\right) = 1$, if there exists $r \in \mathbb{Z}_p^*$ such that $r^2 = a \bmod p$. Otherwise, $a \in \mathbb{Z}_p^*$ is said to be a quadratic non-residue, in which case we denote $\left(\frac{a}{p}\right) = -1$. We denote the sequence of quadratic characters modulo p by $\boldsymbol{Q}_p \in \{0,1\}^{p-1}$. I.e., for any $i \in \mathbb{Z}_p^* = \{1, 2, \ldots, p-1\}$, $\boldsymbol{Q}_p(i) = 0$ if $\left(\frac{i}{p}\right) = 1$ and $\boldsymbol{Q}_p(i) = 1$ if $\left(\frac{i}{p}\right) = -1$. It is a basic fact that $\left(\frac{ab}{p}\right) = \left(\frac{a}{p}\right) \cdot \left(\frac{b}{p}\right)$ for any $a, b \in \mathbb{Z}_p^*$.

The *ℓ-th Peralta prime* p_ℓ [21] is the smallest prime p such that any bit string $\boldsymbol{T}$ of length ℓ appears in $\boldsymbol{Q}_p$, i.e., there exists $a \in \mathbb{Z}_p^*$ such that $\boldsymbol{T}(i) = \boldsymbol{Q}_p(a+i)$ for all $i \in [\ell]$, or equivalently $\left(\frac{a+i}{p}\right) = (-1)^{\boldsymbol{T}(i)}$. We call a an *offset* for $\boldsymbol{T}$ (with respect to p_ℓ). It was shown in [21] that for any $\ell \in \mathbb{N}$, there exists the ℓ-th Peralta prime p_ℓ such that $p_\ell = 2^{O(\ell)}$.

We note that no computational hardness regarding quadratic residues (e.g., the quadratic residuosity problem) is assumed. All of our protocols satisfy unconditional security.

3.3 Tensor Products

Let $\ell_0, \ell_1, \ldots, \ell_{k-1} \in \mathbb{N}$. For two vectors $\boldsymbol{v}_0 \in \mathbb{F}^{\ell_0}, \boldsymbol{v}_1 \in \mathbb{F}^{\ell_1}$, we define the tensor product $\boldsymbol{v}_0 \otimes \boldsymbol{v}_1 \in \mathbb{F}^{\ell_0 \ell_1}$ as $\boldsymbol{v}_0 \otimes \boldsymbol{v}_1 = (\boldsymbol{v}_0(0)\boldsymbol{v}_1, \ldots, \boldsymbol{v}_0(\ell_0 - 1)\boldsymbol{v}_1)$. The tensor product of k vectors $\boldsymbol{v}_0 \in \mathbb{F}^{\ell_0}, \ldots, \boldsymbol{v}_{k-1} \in \mathbb{F}^{\ell_{k-1}}$ is recursively defined as $\boldsymbol{v}_0 \otimes \cdots \otimes \boldsymbol{v}_{k-1} = (\boldsymbol{v}_0 \otimes \cdots \otimes \boldsymbol{v}_{k-2}) \otimes \boldsymbol{v}_{k-1} \in \mathbb{F}^{\ell}$, where $\ell = \ell_0 \cdots \ell_{k-1}$. We denote by $\mathbb{F}^{\ell_0} \otimes \cdots \otimes \mathbb{F}^{\ell_{k-1}}$ the linear space consisting of all linear combinations of tensor products. Note that $\{\boldsymbol{e}_{i_0} \otimes \cdots \otimes \boldsymbol{e}_{i_{k-1}} : i_0 \in [\ell_0], \ldots, i_{k-1} \in [\ell_{k-1}]\}$ is a basis of $\mathbb{F}^{\ell_0} \otimes \cdots \otimes \mathbb{F}^{\ell_{k-1}}$, where $\boldsymbol{e}_i$ denotes the vector whose entry is 1 at position i and 0 elsewhere. Suppose that $a_{i_0,\ldots,i_{k-1}} \in \mathbb{F}$ is given for each $(i_0, \ldots, i_{k-1}) \in [\ell_0] \times \cdots \times [\ell_{k-1}]$. Then, there exists a unique linear map $T : \mathbb{F}^{\ell_0} \otimes \cdots \otimes \mathbb{F}^{\ell_{k-1}} \to \mathbb{F}$ such that

$$T(\boldsymbol{e}_{i_0} \otimes \cdots \otimes \boldsymbol{e}_{i_{k-1}}) = a_{i_0,\ldots,i_{k-1}}, \ \forall (i_0, \ldots, i_{k-1}) \in [\ell_0] \times \cdots \times [\ell_{k-1}].$$

4 Basic Properties of Symmetric Functions

We call a function $f : X^n \to \{0,1\}$ *symmetric* if

$$f(x_{\sigma(0)}, \ldots, x_{\sigma(n-1)}) = f(x_0, \ldots, x_{n-1})$$

for any $(x_0, \ldots, x_{n-1}) \in X^n$ and any permutation σ on $[n]$.

We introduce an equivalent representation of symmetric functions over $[d]^n$. Define $B_d = \{\boldsymbol{x} \in \{0,1\}^{d-1} : \mathrm{wt}(\boldsymbol{x}) \leq 1\}$ and define $\mathcal{S}_{n,d}$ as the set of all

symmetric functions $f : B_d^n \to \{0,1\}$. Note that there exists a one-to-one correspondence between $x \in [d]$ and $\boldsymbol{x} \in B_d$:

$$\boldsymbol{x} = \begin{cases} \boldsymbol{e}_x, & \text{if } x \in \{0,1,\dots,d-2\}, \\ \boldsymbol{0}, & \text{if } x = d-1. \end{cases}$$

This map induces a one-to-one correspondence between $\mathcal{S}_{n,d}$ and the set of all symmetric functions over $[d]^n$. Therefore, a PSM protocol for $\mathcal{S}_{n,d}$ implies a PSM protocol for symmetric functions over $[d]^n$ with the same communication and randomness complexity. From the above observation, we focus on symmetric functions over B_d^n instead of those over $[d]^n$. Since $f \in \mathcal{S}_{n,d}$ assigns a bit to each histogram $\boldsymbol{h} \in \mathbb{Z}^{d-1}$ such that $\boldsymbol{h}(i) \geq 0$ and $\sum_{i\in[d-1]} \boldsymbol{h}(i) \leq n$, f can be expressed as an array of size $\binom{n+d-1}{d-1}$. In particular, it holds that $\log|\mathcal{S}_{n,d}| = \binom{n+d-1}{d-1}$.

We show that the output of $f \in \mathcal{S}_{n,d}$ is uniquely determined by the *histogram* of inputs. Define $\overline{B}_{n,d}$ as

$$\overline{B}_{n,d} = \left\{ \sum_{i\in[n]} \boldsymbol{x}_i \in \mathbb{Z}^{d-1} : \boldsymbol{x}_0, \dots, \boldsymbol{x}_{n-1} \in B_d \right\}.$$

Note that the j-th component of $\sum_{i\in[n]} \boldsymbol{x}_i \in \overline{B}_{n,d}$ counts the frequency of a value j in the corresponding inputs $(x_i)_{i\in[n]} \in [d]^n$. We can omit the last bin of the histogram since the frequency of $j = d-1$ is uniquely recovered by those of other bins.

Proposition 1. *Let $\boldsymbol{x}_0, \dots, \boldsymbol{x}_{n-1}, \boldsymbol{x}'_0, \dots, \boldsymbol{x}'_{n-1} \in B_d$. If $\sum_{k\in[n]} \boldsymbol{x}_k = \sum_{k\in[n]} \boldsymbol{x}'_k$, then there exists a permutation σ on $[n]$ such that $\boldsymbol{x}'_k = \boldsymbol{x}_{\sigma(k)}$ for all k. In particular, for any $f \in \mathcal{S}_{n,d}$, there exists $g : \overline{B}_{n,d} \to \{0,1\}$ such that $g(\boldsymbol{x}_0 + \cdots + \boldsymbol{x}_{n-1}) = f(\boldsymbol{x}_0, \dots, \boldsymbol{x}_{n-1})$ for all $\boldsymbol{x}_0, \dots, \boldsymbol{x}_{n-1} \in B_d$.*

Proof. We show the claim by induction on n. If $n = 1$, the claim is trivial. Let $n > 1$ and assume that the claim holds for $n-1$. Let $\boldsymbol{s} = \sum_{k\in[n]} \boldsymbol{x}_k = \sum_{k\in[n]} \boldsymbol{x}'_k$. If $\boldsymbol{s} = \boldsymbol{0}$, then the statements are trivial since $\boldsymbol{x}'_k = \boldsymbol{x}_k = \boldsymbol{0}$ for all k. We consider the case of $\boldsymbol{s} \neq \boldsymbol{0}$. Let $j \in [d-1]$ be the smallest index such that $\boldsymbol{s}(j) > 0$. Since $\boldsymbol{x}_k, \boldsymbol{x}'_k \in B_d$, there exist $i, i' \in [n]$ such that $\boldsymbol{x}'_{i'} = \boldsymbol{x}_i = \boldsymbol{e}_j$. Since $\sum_{k\neq i} \boldsymbol{x}_k = \sum_{k\neq i'} \boldsymbol{x}'_k$, the claim for $n-1$ ensures the existence of a one-to-one correspondence $\tau : [n]\setminus\{i'\} \to [n]\setminus\{i\}$ such that $\boldsymbol{x}'_k = \boldsymbol{x}_{\tau(k)}$ for all $k \in [n]\setminus\{i'\}$. We define a permutation σ on $[n]$ as follows:

$$\sigma(k) = \begin{cases} i, & \text{if } k = i', \\ \tau(k), & \text{if } k \in \{1,\dots,n\} \setminus \{i'\}. \end{cases}$$

Then, for $k = i'$, it holds that $\boldsymbol{x}_{\sigma(k)} = \boldsymbol{x}_i = \boldsymbol{x}'_{i'} = \boldsymbol{x}'_k$ and for $k \neq i'$, it holds that $\boldsymbol{x}_{\sigma(k)} = \boldsymbol{x}_{\tau(k)} = \boldsymbol{x}'_k$, which implies the claim for n. □

The following proposition shows that every histogram (i.e., a vector of $\overline{B}_{n,d}$) can be encoded into a non-negative integer of size $O(n^{d-1})$. Specifically, for a vector $\boldsymbol{v} \in \mathbb{Z}^{d-1}$, define $I_{\boldsymbol{v}} : \mathbb{Z}^{d-1} \to \mathbb{Z}$ as $I_{\boldsymbol{v}}(\boldsymbol{x}) = \langle \boldsymbol{v}, \boldsymbol{x} \rangle$.

Proposition 2. *There exists $\boldsymbol{v} \in \mathbb{Z}_{\geq 0}^{d-1}$ such that $I_{\boldsymbol{v}}(\boldsymbol{x}) \neq I_{\boldsymbol{v}}(\boldsymbol{x}')$ for any $\boldsymbol{x}, \boldsymbol{x}' \in \overline{B}_{n,d}$ with $\boldsymbol{x} \neq \boldsymbol{x}'$. That is, the restriction of $I_{\boldsymbol{v}}$ to $\overline{B}_{n,d}$ is injective. Furthermore, such $\boldsymbol{v}$ can be chosen so that $\max(I_{\boldsymbol{v}}(\overline{B}_{n,d})) = O(n^{d-1})$.*

Proof. Since the case of $n = 1$ is trivial, we assume that $n > 1$ in the following. We apply the anti-lexicographical order to $\mathbb{Z}^{d-1}$, which can also be applied to the subset $\overline{B}_{n,d}$. That is, we denote by $\boldsymbol{a} < \boldsymbol{b}$ for $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{Z}^{d-1}$ if there exists $i \in [d-1]$ such that $\boldsymbol{a}(i) < \boldsymbol{b}(i)$ and $\boldsymbol{a}(j) = \boldsymbol{b}(j)$ for all $j \in [d-1]$ with $j > i$. We define $\boldsymbol{v} \in \mathbb{Z}_{\geq 0}^{d-1}$ as

$$\boldsymbol{v}(i) = \frac{n^{i+1} - 1}{n - 1} \in \mathbb{Z}_{\geq 0}$$

for all $i \in [d-1]$. We will show that if $\boldsymbol{x} < \boldsymbol{x}'$ then $I_{\boldsymbol{v}}(\boldsymbol{x}) < I_{\boldsymbol{v}}(\boldsymbol{x}')$ for any $\boldsymbol{x}, \boldsymbol{x}' \in \overline{B}_{n,d}$.

Let $\boldsymbol{x}, \boldsymbol{x}' \in \overline{B}_{n,d}$ be such that $\boldsymbol{x} < \boldsymbol{x}'$. Let $i \in [d-1]$ be the unique index such that $\boldsymbol{x}(i) < \boldsymbol{x}'(i)$ and $\boldsymbol{x}(j) = \boldsymbol{x}'(j)$ for all $j \in [d-1]$ with $j > i$. We then have that

$$\begin{aligned} I_{\boldsymbol{v}}(\boldsymbol{x}') - I_{\boldsymbol{v}}(\boldsymbol{x}) &= \sum_{j=0}^{i} (\boldsymbol{x}'(j) - \boldsymbol{x}(j)) \boldsymbol{v}(j) \\ &= \boldsymbol{v}(i) + \sum_{j=0}^{i-1} (\boldsymbol{x}'(j) - \boldsymbol{x}(j)) \boldsymbol{v}(j) \\ &\geq \boldsymbol{v}(i) - \boldsymbol{v}(i-1) \sum_{j=0}^{i-1} \boldsymbol{x}(j) \end{aligned}$$

since $\boldsymbol{x}'(j) \geq 0$ for all $j \in [d-1]$ and $\boldsymbol{v}(j) \leq \boldsymbol{v}(j')$ if $j < j'$, where we assume that $\boldsymbol{v}(-1) = 0$. We here note that $\boldsymbol{x} \in \overline{B}_{n,d}$ can be expressed as $\boldsymbol{x} = \sum_{k=1}^{n} \boldsymbol{x}_k$ for some $\boldsymbol{x}_k \in B_d$, which ensures that

$$\sum_{j=0}^{d-1} \boldsymbol{x}(j) = \sum_{k=1}^{n} \sum_{j=0}^{d-1} \boldsymbol{x}_k(j) = \sum_{k=1}^{n} \mathrm{wt}(\boldsymbol{x}_k) \leq n.$$

We therefore have that

$$I_{\boldsymbol{v}}(\boldsymbol{x}') - I_{\boldsymbol{v}}(\boldsymbol{x}) \geq \boldsymbol{v}(i) - n\boldsymbol{v}(i-1) = \frac{(n^{i+1} - 1) - n(n^i - 1)}{n - 1} > 0.$$

The rest of the statements follows from

$$\max(I_{\boldsymbol{v}}(\overline{B}_{n,d})) \leq \boldsymbol{v}(d-2) \cdot \max_{\boldsymbol{x} \in \overline{B}_{n,d}} \left(\sum_{j=0}^{d-1} \boldsymbol{x}(j) \right) \leq O(n^{d-2}) \cdot n = O(n^{d-1}).$$

□

5 PSM with Universal Reconstruction for Symmetric Functions

5.1 Our Construction

The following theorem shows our PSM protocol with universal reconstruction for $\mathcal{S}_{n,d}$.

Theorem 1. *There exists an n-party PSM protocol Π for $\mathcal{S}_{n,d}$ with universal reconstruction such that* $\mathrm{Comm}(\Pi) = O((n-1)^d)$, $\mathrm{Rand}(\Pi) = O((n-1)^d)$ *and* $\mathrm{Comp}(\Pi) = 2^{O((n-1)^{d-1})}$.

Proof. We describe the protocol Π in Fig. 1.

Correctness. We note that since the restriction of $I_{\boldsymbol{v}}$ to $\overline{B}_{n-1,d}$ is injective and $0 \le I_{\boldsymbol{v}}(\boldsymbol{x}) \le \ell - 1$ for any $\boldsymbol{x} \in \overline{B}_{n-1,d}$, we can indeed find $\boldsymbol{T} \in \{0,1\}^\ell$ satisfying the condition (7). We have that

$$\sum_{i\in[n]} m_i = (as_0 + r_0) + \sum_{i=1}^{n-1}(I_{\boldsymbol{v}}(\boldsymbol{x}_i)s_0 + r_i) = \left(a + \sum_{i=1}^{n-1} I_{\boldsymbol{v}}(\boldsymbol{x}_i)\right)(s_0')^2 \bmod p.$$

We then have that

$$q = \left(\frac{m_0 + \cdots + m_{n-1}}{p}\right) = \left(\frac{a + \sum_{i=1}^{n-1} I_{\boldsymbol{v}}(\boldsymbol{x}_i)}{p}\right)\left(\frac{s_0'}{p}\right)^2 = (-1)^{\boldsymbol{T}(j)},$$

where $j = I_{\boldsymbol{v}}\left(\sum_{i=1}^{n-1} \boldsymbol{x}_i\right)$. The condition (7) implies that

$$\boldsymbol{T}(j) = g\left(\boldsymbol{x}_0 + \sum_{i=1}^{n-1} \boldsymbol{x}_i\right) = f(\boldsymbol{x}_0, \ldots, \boldsymbol{x}_{n-1}),$$

which in turn implies $y = f(\boldsymbol{x}_0, \ldots, \boldsymbol{x}_{n-1})$.

Privacy. Because of the shares $(r_i)_{i\in[n]}$ of 0 and the random quadratic residue s_0, the referee only learns $(-1)^{T(j)}$, which is equivalent to $f((\boldsymbol{x}_i)_{i\in[n]})$. Formally, we construct a simulator as follows.

$\mathsf{Sim}(f((\boldsymbol{x}_i)_{i\in[n]}))$

1. Choose $\tilde{s}_0' \leftarrow_{\$} \mathbb{Z}_p^*$ and set $\tilde{s}_0 = (\tilde{s}_0')^2 \bmod p$.
2. Let $b \in \mathbb{Z}_p^*$ be any quadratic non-residue modulo p.
3. If $f((\boldsymbol{x}_i)_{i\in[n]}) = 0$, set $\tilde{m} = \tilde{s}_0$. If $f((\boldsymbol{x}_i)_{i\in[n]}) = 1$, set $\tilde{m} = \tilde{s}_0 b \bmod p$.
4. Choose $\tilde{m}_0, \ldots, \tilde{m}_{n-1} \in \mathbb{Z}_p$ conditioned on $\sum_{i\in[n]} \tilde{m}_i = \tilde{m} \bmod p$.
5. Output $(\tilde{m}_0, \ldots, \tilde{m}_{n-1})$.

Since

Notations.

- Let $\boldsymbol{v} \in \mathbb{Z}^{d-1}$ be a vector such that the restriction of $I_{\boldsymbol{v}}$ to $\overline{B}_{n-1,d}$ is injective.
- Let $\ell = \max(I_{\boldsymbol{v}}(\overline{B}_{n-1,d})) + 1$.
- Let p be the ℓ-th Peralta prime.

$\mathsf{Gen}(f)$**.**

1. Choose $r_0, \ldots, r_{n-1} \in \mathbb{Z}_p$ uniformly at random conditioned on $\sum_{i=0}^{n-1} r_i = 0 \bmod p$.
2. Choose $s_0' \leftarrow\!\!\$\, \mathbb{Z}_p^*$ and set $s_0 = (s_0')^2 \bmod p$.
3. Output $r = (s_0, r_0, \ldots, r_{n-1})$.

$\mathsf{Enc}(i, f, \boldsymbol{x}_i, r)$**.**

- $i = 0$:
  1. Let $g : \overline{B}_{n,d} \to \{0,1\}$ be such that
  $$g(\boldsymbol{x}_0' + \cdots + \boldsymbol{x}_{n-1}') = f(\boldsymbol{x}_0', \ldots, \boldsymbol{x}_{n-1}')$$
  for all $\boldsymbol{x}_0', \ldots, \boldsymbol{x}_{n-1}' \in B_d$.
  2. Compute $\boldsymbol{T} \in \{0,1\}^{\ell}$ such that
  $$\boldsymbol{T}(I_{\boldsymbol{v}}(\boldsymbol{x})) = g(\boldsymbol{x}_0 + \boldsymbol{x}),\ \forall \boldsymbol{x} \in \overline{B}_{n-1,d}. \tag{7}$$
  3. Find an offset $a \in \mathbb{Z}_p$ for $\boldsymbol{T}$. That is, a is an integer modulo p such that
  $$\left(\frac{a+j}{p}\right) = (-1)^{\boldsymbol{T}(j)}$$
  for all $j \in [\ell]$.
  4. Output $m_0 = as_0 + r_0 \bmod p$.
- $i \neq 0$:
  1. Output $m_i = I_{\boldsymbol{v}}(\boldsymbol{x}_i)s_0 + r_i \bmod p$.

$\mathsf{Dec}(m_0, \ldots, m_{n-1})$**.**

1. Compute
$$q = \left(\frac{m_0 + \cdots + m_{n-1}}{p}\right) \in \{-1, 1\}.$$
2. Output
$$y = \begin{cases} 0, & \text{if } q = 1, \\ 1, & \text{if } q = -1. \end{cases}$$

Fig. 1. A PSM protocol with universal reconstruction for $\mathcal{S}_{n,d}$

$$\left(\frac{a+\sum_{i=1}^{n-1} I_{\boldsymbol{v}}(\boldsymbol{x}_i)}{p}\right) = (-1)^{\boldsymbol{T}(j)} = (-1)^{g(\boldsymbol{x}_0+\boldsymbol{x}_1+\cdots+\boldsymbol{x}_{n-1})},$$

$a+\sum_{i=1}^{n-1} I_{\boldsymbol{v}}(\boldsymbol{x}_i)$ is a quadratic residue modulo p if $f((\boldsymbol{x}_i)_{i\in[n]}) = 0$ and a quadratic non-residue if $f((\boldsymbol{x}_i)_{i\in[n]}) = 1$. Since s_0 is uniformly distributed over the set of all quadratic residues, $m := \left(a+\sum_{i=1}^{n-1} I_{\boldsymbol{v}}(\boldsymbol{x}_i)\right) s_0$ in the real execution is uniformly distributed over the set of all quadratic residues if $f((\boldsymbol{x}_i)_{i\in[n]}) = 0$ and over the set of all quadratic non-residues if $f((\boldsymbol{x}_i)_{i\in[n]}) = 1$. By definition, $\tilde{m}$ in the simulated execution is uniformly distributed over the set of all quadratic residues if $f((\boldsymbol{x}_i)_{i\in[n]}) = 0$ and over the set of all quadratic non-residues if $f((\boldsymbol{x}_i)_{i\in[n]}) = 1$. Therefore, the distributions of m and $\tilde{m}$ are identical with each other.

Since $(r_0,\ldots,r_{n-1})$ are uniformly random elements in $\mathbb{Z}_p$ conditioned on $\sum_{i\in[n]} r_i = 0$, $(m_0,\ldots,m_{n-1})$ in the real execution are uniformly random elements conditioned on $\sum_{i\in[n]} m_i = m$. Since $\tilde{m}$ and m are identically distributed, $(\tilde{m}_0,\ldots,\tilde{m}_{n-1})$ also follows the same distribution as $(m_0,\ldots,m_{n-1})$.

Complexity. Each message outputted by Enc consists of an element from $\mathbb{Z}_p$. Since the ℓ-th Peralta prime satisfies $p = 2^{O(\ell)}$, it follows from Proposition 2 that $\mathrm{Comm}(\Pi) = O(n\log p) = O(n\ell) = O(n(n-1)^{d-1}) = O((n-1)^d)$. Since the common randomness outputted by Gen consists of $n+1$ elements from $\mathbb{Z}_p$, it follows similarly that $\mathrm{Rand}(\Pi) = O((n-1)^d)$. The most computationally costly part of Π is to find an offset a in Step 3 of $\mathsf{Enc}(0, f, \boldsymbol{x}_0, r)$, which requires computational complexity $O(\ell p)$ since it is necessary to search the $O(p)$-bit string $\boldsymbol{Q}_p$ for the ℓ-bit string $\boldsymbol{T}$. Therefore, we have that $\mathrm{Comp}(\Pi) = 2^{O((n-1)^{d-1})}$. □

A possible drawback is that the above protocol is not computationally efficient. We also show a computationally efficient PSM protocol with universal reconstruction for $\mathcal{S}_{n,d}$ at the cost of increasing communication complexity by a factor of $\log n$. The construction is an optimization of the protocol in [11]. We give the detailed descriptions in Appendix A.

Proposition 3. *There exists an n-party PSM protocol Π for $\mathcal{S}_{n,d}$ with universal reconstruction such that* $\mathrm{Comm}(\Pi) = O((n+1)^d d\log n)$, $\mathrm{Rand}(\Pi) = O((n+1)^d d\log n)$ *and* $\mathrm{Comp}(\Pi) = O((n+1)^d d\log n)$.

5.2 Lower Bound

We can show a lower bound on communication complexity of PSM with universal reconstruction for $\mathcal{S}_{n,d}$, almost matching the upper bound given in Theorem 1. The lower bound is derived by extending the technique of [6, Appendix H] to the multi-party setting (The original technique of [6] was applied to the two-party setting).

Theorem 2. *Let $\mathcal{F}$ be a set of functions from $[d]^n$ to a set Y. Let $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be an n-party PSM protocol for $\mathcal{F}$ with universal reconstruction. Then, it holds that* $\mathrm{Comm}(\Pi) \geq (\log|\mathcal{F}|)/d$. *In particular, if $\mathcal{F} = \mathcal{S}_{n,d}$, then it holds that*

$$\mathrm{Comm}(\Pi) \geq \frac{1}{d}\binom{n+d-1}{d-1},$$

which is $\Omega(n^{d-1})$ if $d = O(1)$.

Proof. Let comm_i denote the bit-length of messages outputted by $\mathsf{Enc}(i,\cdot,\cdot,\cdot)$ and let $C = d\sum_{i\in[n]}\mathsf{comm}_i$. Consider an execution of Π where the common randomness r is fixed. We define a map $\mu : \mathcal{F} \to \{0,1\}^C$ as

$$\mu(f) = \mu_0(f)\|\cdots\|\mu_{n-1}(f), \text{ where } \mu_i(f) = \mathsf{Enc}(i,f,0,r)\|\cdots\|\mathsf{Enc}(i,f,d-1,r)$$

and $\|$ denotes concatenation. Then, μ is injective. Indeed, suppose that $\mu(f) = \mu(f')$ for some $f, f' \in \mathcal{F}$. Write

$$\mu(f) = \mu(f') = m_0\|\cdots\|m_{n-1} \text{ and } m_i = m_{i,0}\|\cdots\|m_{i,d-1},$$

where $m_{i,x} \in \{0,1\}^{\mathsf{comm}_i}$. Since Π has universal reconstruction, it holds that for any $(x_0,\ldots,x_{n-1}) \in [d]^n$,

$$f(x_0,\ldots,x_{n-1}) = \mathsf{Dec}(m_{0,x_0},\ldots,m_{n-1,x_{n-1}}) = f'(x_0,\ldots,x_{n-1})$$

and thus $f = f'$. Therefore, we have that $\mathrm{Comm}(\Pi) = C/d \geq (\log|\mathcal{F}|)/d$.

Since $\log|\mathcal{S}_{n,d}| = \binom{n+d-1}{d-1}$, we have that $\mathrm{Comm}(\Pi) \geq (1/d)\binom{n+d-1}{d-1}$ if $\mathcal{F} = \mathcal{S}_{n,d}$. □

6 Efficient PSM for Symmetric Functions

6.1 Decomposition Approach

Our approach is to decompose the computation of a function $f \in \mathcal{S}_{n,d}$ into those of smaller functions in $\mathcal{S}_{n,d'}$ with $d' < d$. First, we show a decomposition of a histogram of inputs into three histograms with fewer bins. Let $d_0, d_1, d_2 \in \mathbb{N}$ be parameters such that $d_0 \leq d_1 \leq d_2$ and $d_0 + d_1 + d_2 = d - 1$. For $A \subseteq [3]$, we define $\overline{d}_A = \sum_{j\in A} d_j$. If $A = [j]$, we simply write $\overline{d}_j = \overline{d}_{[j]}$. We define $\overline{d}_0 = 0$. For $j \in [3]$, we define $P_j : B_d \to \mathbb{Z}^{d_j}$ as a map that extracts a sub-vector of $\boldsymbol{x} \in B_d$ consisting of the entries with indices $\overline{d}_j \leq i < \overline{d}_{j+1}$. Formally,

$$P_j(\boldsymbol{x}) = (\boldsymbol{x}(\overline{d}_j), \boldsymbol{x}(\overline{d}_j+1), \ldots, \boldsymbol{x}(\overline{d}_{j+1}-1)),\ \forall \boldsymbol{x} \in B_d.$$

In particular, it holds that

$$\boldsymbol{x} = (P_0(\boldsymbol{x}), P_1(\boldsymbol{x}), P_2(\boldsymbol{x})),\ \forall \boldsymbol{x} \in B_d.$$

The histogram $\overline{\boldsymbol{x}} = \sum_{i\in[n]} \boldsymbol{x}_i$ of inputs $(\boldsymbol{x}_i)_{i\in[n]} \in B_d^n$ can be decomposed into three histograms

$$\overline{\boldsymbol{x}} = (P_0(\overline{\boldsymbol{x}}), P_1(\overline{\boldsymbol{x}}), P_2(\overline{\boldsymbol{x}})).$$

Note that we have $P_j(\boldsymbol{x}) \in B_{d_j+1}$ for any $\boldsymbol{x} \in B_d$ and hence $P_j(\overline{\boldsymbol{x}}) \in \overline{B}_{n,d_j+1}$.

In Section 4, we have shown an encoding of histograms into integers of size $O(n^d)$. Thanks to the above decomposition, we are able to encode the histograms into smaller integers. Specifically, let $\boldsymbol{v}_j \in \mathbb{Z}_{\geq 0}^{d_j}$ be a vector such that the restriction of $I_{\boldsymbol{v}_j}$ to $\overline{B}_{n,d_j+1}$ is injective. Let $\ell_j = \max(I_{\boldsymbol{v}_j}(\overline{B}_{n,d_j+1})) + 1$, which is $O(n^{d_j})$ from Proposition 2. We define $\phi_j : B_d^n \to [\ell_j]$ as

$$\phi_j(\boldsymbol{x}_0, \ldots, \boldsymbol{x}_{n-1}) = I_{\boldsymbol{v}_j}(P_j(\overline{\boldsymbol{x}})),$$

where $\overline{\boldsymbol{x}} = \sum_{i\in[n]} \boldsymbol{x}_i$. We also define $\phi : B_d^n \to [\ell_0] \times [\ell_1] \times [\ell_2]$ as

$$\phi((\boldsymbol{x}_i)_{i\in[n]}) = (\phi_0((\boldsymbol{x}_i)_{i\in[n]}), \phi_1((\boldsymbol{x}_i)_{i\in[n]}), \phi_2((\boldsymbol{x}_i)_{i\in[n]})).$$

Then, the inputs $(\boldsymbol{x}_i)_{i\in[n]}$ are encoded into three smaller integers each of size $\ell_j = O(n^{d_j})$.

Now, we show a decomposition of $f \in \mathcal{S}_{n,d}$ into smaller functions. For each $(m_0, m_1, m_2) \in [\ell_0] \times [\ell_1] \times [\ell_2]$, we define $a_{m_0,m_1,m_2} \in \mathbb{F}$ as follows: If there exists $\boldsymbol{X} = (\boldsymbol{x}_i)_{i\in[n]} \in B_d^n$ such that $\phi(\boldsymbol{X}) = (m_0, m_1, m_2)$, then $a_{m_0,m_1,m_2} = f(\boldsymbol{X})$; else, $a_{m_0,m_1,m_2} = 0$. To ensure that a_{m_0,m_1,m_2} is well-defined, we show in the following proposition that for any $(\boldsymbol{x}_i)_{i\in[n]}, (\boldsymbol{x}'_i)_{i\in[n]}$ such that $\phi((\boldsymbol{x}_i)_{i\in[n]}) = \phi((\boldsymbol{x}'_i)_{i\in[n]})$, it holds that $f((\boldsymbol{x}_i)_{i\in[n]}) = f((\boldsymbol{x}'_i)_{i\in[n]})$.

Proposition 4. *Let $f \in \mathcal{S}_{n,d}$ and $(\boldsymbol{x}_i)_{i\in[n]}, (\boldsymbol{x}'_i)_{i\in[n]} \in B_d^n$. If $\phi((\boldsymbol{x}_i)_{i\in[n]}) = \phi((\boldsymbol{x}'_i)_{i\in[n]})$, then $f((\boldsymbol{x}_i)_{i\in[n]}) = f((\boldsymbol{x}'_i)_{i\in[n]})$.*

Proof. We have from the assumption that

$$I_{\boldsymbol{v}_j}\left(P_j(\boldsymbol{x}_0) + \cdots + P_j(\boldsymbol{x}_{n-1})\right) = I_{\boldsymbol{v}_j}\left(P_j(\boldsymbol{x}'_0) + \cdots + P_j(\boldsymbol{x}'_{n-1})\right),\ \forall j = 0, 1, 2.$$

Since $\sum_{i\in[n]} P_j(\boldsymbol{x}_i) \in \overline{B}_{n,d_j+1}$ and the restriction of $I_{\boldsymbol{v}_j}$ to $\overline{B}_{n,d_j+1}$ is injective, we have that

$$P_j(\boldsymbol{x}_0) + \cdots + P_j(\boldsymbol{x}_{n-1}) = P_j(\boldsymbol{x}'_0) + \cdots + P_j(\boldsymbol{x}'_{n-1}),\ \forall j = 0, 1, 2.$$

This means that $\boldsymbol{x}_0 + \cdots + \boldsymbol{x}_{n-1} = \boldsymbol{x}'_0 + \cdots + \boldsymbol{x}'_{n-1}$, as $\boldsymbol{a} = (P_0(\boldsymbol{a}), P_1(\boldsymbol{a}), P_2(\boldsymbol{a}))$ for any $\boldsymbol{a} \in B_d$. Since $\boldsymbol{x}_i, \boldsymbol{x}'_i \in B_d$, Proposition 1 ensures a permutation σ on $[n]$ such that $\boldsymbol{x}'_i = \boldsymbol{x}_{\sigma(i)}$. Since f is symmetric, we conclude that $f(\boldsymbol{x}_0, \ldots, \boldsymbol{x}_{n-1}) = f(\boldsymbol{x}'_0, \ldots, \boldsymbol{x}'_{n-1})$. □

Then, we have a unique linear map $T_f : \mathbb{F}^{\ell_0} \otimes \mathbb{F}^{\ell_1} \otimes \mathbb{F}^{\ell_2} \to \mathbb{F}$ such that for any $(m_0, m_1, m_2) \in [\ell_0] \times [\ell_1] \times [\ell_2]$,

$$T_f(\boldsymbol{e}_{m_0} \otimes \boldsymbol{e}_{m_1} \otimes \boldsymbol{e}_{m_2}) = a_{m_0,m_1,m_2}.$$

Let $\boldsymbol{X} = (\boldsymbol{x}_i)_{i\in[n]} \in B_d^n$, $\boldsymbol{r}_0 \in \mathbb{F}^{\ell_0}$, $\boldsymbol{r}_1 \in \mathbb{F}^{\ell_1}$ and $\boldsymbol{r}_2 \in \mathbb{F}^{\ell_2}$. It then holds that

$$\begin{aligned}
&T_f((\boldsymbol{e}_{\phi_0(\boldsymbol{X})} + \boldsymbol{r}_0) \otimes (\boldsymbol{e}_{\phi_1(\boldsymbol{X})} + \boldsymbol{r}_1) \otimes (\boldsymbol{e}_{\phi_2(\boldsymbol{X})} + \boldsymbol{r}_2)) \\
&= T_f(\boldsymbol{e}_{\phi_0(\boldsymbol{X})} \otimes \boldsymbol{e}_{\phi_1(\boldsymbol{X})} \otimes \boldsymbol{e}_{\phi_2(\boldsymbol{X})}) + T_f(\boldsymbol{e}_{\phi_0(\boldsymbol{X})} \otimes \boldsymbol{e}_{\phi_1(\boldsymbol{X})} \otimes \boldsymbol{r}_2) \\
&\quad + T_f(\boldsymbol{e}_{\phi_0(\boldsymbol{X})} \otimes \boldsymbol{r}_1 \otimes \boldsymbol{e}_{\phi_2(\boldsymbol{X})}) + T_f(\boldsymbol{r}_0 \otimes \boldsymbol{e}_{\phi_1(\boldsymbol{X})} \otimes \boldsymbol{e}_{\phi_2(\boldsymbol{X})}) \\
&\quad + T_f(\boldsymbol{e}_{\phi_0(\boldsymbol{X})} \otimes \boldsymbol{r}_1 \otimes \boldsymbol{r}_2) + T_f(\boldsymbol{r}_0 \otimes \boldsymbol{e}_{\phi_1(\boldsymbol{X})} \otimes \boldsymbol{r}_2) \\
&\quad + T_f(\boldsymbol{r}_0 \otimes \boldsymbol{r}_1 \otimes \boldsymbol{e}_{\phi_2(\boldsymbol{X})}) + T_f(\boldsymbol{r}_0 \otimes \boldsymbol{r}_1 \otimes \boldsymbol{r}_2) \\
&= \sum_{A\subseteq[3]} T_f\left(\bigotimes_{j\in A} \boldsymbol{e}_{\phi_j(\boldsymbol{X})} \otimes \bigotimes_{j\notin A} \boldsymbol{r}_j\right),
\end{aligned}$$

where $\bigotimes_{j\in A} \boldsymbol{e}_{\phi_j(\boldsymbol{X})} \otimes \bigotimes_{j\notin A} \boldsymbol{r}_j$ denotes a tensor $\boldsymbol{u}_0 \otimes \boldsymbol{u}_1 \otimes \boldsymbol{u}_2$ such that

$$\boldsymbol{u}_j = \begin{cases} \boldsymbol{e}_{\phi_j(\boldsymbol{X})}, & \text{if } j \in A, \\ \boldsymbol{r}_j, & \text{if } j \notin A. \end{cases}$$

Since $T_f(\boldsymbol{e}_{\phi_0(\boldsymbol{X})} \otimes \boldsymbol{e}_{\phi_1(\boldsymbol{X})} \otimes \boldsymbol{e}_{\phi_2(\boldsymbol{X})}) = f(\boldsymbol{X})$, we have that

$$\begin{aligned}
f(\boldsymbol{X}) = T_f((\boldsymbol{e}_{\phi_0(\boldsymbol{X})} + \boldsymbol{r}_0) \otimes (\boldsymbol{e}_{\phi_1(\boldsymbol{X})} + \boldsymbol{r}_1) \otimes (\boldsymbol{e}_{\phi_2(\boldsymbol{X})} + \boldsymbol{r}_2)) \\
- \sum_{A\neq\emptyset,[3]} T_f\left(\bigotimes_{j\in A} \boldsymbol{e}_{\phi_j(\boldsymbol{X})} \otimes \bigotimes_{j\notin A} \boldsymbol{r}_j\right) - T_f(\boldsymbol{r}_0 \otimes \boldsymbol{r}_1 \otimes \boldsymbol{r}_2). \qquad (8)
\end{aligned}$$

6.2 Our Construction

The following theorem shows our generic construction of a PSM protocol for $\mathcal{S}_{n,d}$ from PSM protocols for smaller functions.

Theorem 3. *Assume that for any $d' \in \mathbb{N}$, there exists an n-party PSM protocol $\Pi_{d'}$ for $\mathcal{S}_{n,d'}$ with universal reconstruction. Let $d_0, d_1, d_2 \in \mathbb{N}$ be parameters such that $d_0 \le d_1 \le d_2$ and $d_0 + d_1 + d_2 = d - 1$. Then, there exists an n-party PSM protocol Π for $\mathcal{S}_{n,d}$ such that*

$$\begin{aligned}
\mathrm{Comm}(\Pi) &= O(n^{d_2} \cdot \mathrm{Comm}(\Pi_{d_2+1}) + \mathrm{Comm}(\Pi_{d-d_0})), \\
\mathrm{Rand}(\Pi) &= O(n^{d_2} \cdot \mathrm{Rand}(\Pi_{d_2+1}) + \mathrm{Rand}(\Pi_{d-d_0})), \\
\mathrm{Comp}(\Pi) &= O(n^{d_2} \cdot \mathrm{Comp}(\Pi_{d_2+1}) + \mathrm{Comp}(\Pi_{d-d_0})) + (n+d)^{O(d)}.
\end{aligned}$$

Proof. We describe the protocol Π in Figs. 2 and 3.

Correctness. For each $j \in [3]$, the correctness of Π_{d_j+1} implies that

$$\boldsymbol{a}_j = g_j(\boldsymbol{x}_{0,j}, \ldots, \boldsymbol{x}_{n-1,j}) = \boldsymbol{e}_{I_{\boldsymbol{v}_j}(P_j(\boldsymbol{x}_0)+\cdots+P_j(\boldsymbol{x}_{n-1}))} + \boldsymbol{r}_j = \boldsymbol{e}_{\phi_j(\boldsymbol{X})} + \boldsymbol{r}_j.$$

Notations.

- Let $d_0, d_1, d_2 \in \mathbb{N}$ be parameters such that $d_0 \le d_1 \le d_2$ and $d_0+d_1+d_2 = d-1$.
- For $A \subseteq [3]$, we define $\overline{d}_A = \sum_{j\in A} d_j$. If $A = [j]$, we simply write $\overline{d}_j = \overline{d}_{[j]}$. We define $\overline{d}_0 = 0$.
- Let $\boldsymbol{v}_j \in \mathbb{Z}_{\ge 0}^{d_j}$ be a vector such that the restriction of $I_{\boldsymbol{v}_j}$ to $\overline{B}_{n,d_j+1}$ is injective.
- Let $\ell_j = \max(I_{\boldsymbol{v}_j}(\overline{B}_{n,d_j+1})) + 1$.
- For $d' \in \mathbb{N}$, let $\Pi_{d'} = (\mathsf{Gen}_{d'}, \mathsf{Enc}_{d'}, \mathsf{Dec}_{d'})$ be an n-party PSM protocol for $\mathcal{S}_{n,d'}$ with universal reconstruction.

$\mathsf{Gen}(f)$**.**

1. For each $j \in [3]$, choose $\boldsymbol{r}_j \leftarrow\!\!\$\, \mathbb{F}^{\ell_j}$.
2. For each $A \in 2^{[3]} \setminus \{\emptyset, [3]\}$, choose $s_A \leftarrow\!\!\$\, \mathbb{F}$, and set $s_\emptyset = -\sum_{A\neq\emptyset,[3]} s_A$.
3. For each $j \in [3]$, define $g_j : B_{d_j+1}^n \to \{0,1\}^{\ell_j}$ as
$$g_j(\boldsymbol{x}_0, \ldots, \boldsymbol{x}_{n-1}) = \boldsymbol{e}_{I_{\boldsymbol{v}_j}(\boldsymbol{x}_0+\cdots+\boldsymbol{x}_{n-1})} + \boldsymbol{r}_j.$$
Note that each component of g_j belongs to $\mathcal{S}_{n,d_j+1}$.
4. For each $A \in 2^{[3]} \setminus \{\emptyset, [3]\}$, define $h_A : B_{\overline{d}_A+1}^n \to \{0,1\}$ as
$$h_A(\boldsymbol{x}_0, \ldots, \boldsymbol{x}_{n-1}) = T_f\left(\bigotimes_{j\in A} \boldsymbol{e}_{I_{\boldsymbol{v}_j}(\boldsymbol{x}_{0,j}+\cdots+\boldsymbol{x}_{n-1,j})} \otimes \bigotimes_{j\notin A} \boldsymbol{r}_j\right) + s_A,$$
where $\boldsymbol{x}_{i,j} \in B_{d_j+1}$ is a sub-vector of $\boldsymbol{x}_i$ such that $\boldsymbol{x}_i = (\boldsymbol{x}_{i,j})_{j\in A}$. Note that $h_A \in \mathcal{S}_{n,\overline{d}_A+1}$.
5. For each $j \in [3]$, compute
$$\boldsymbol{\mu}_j \leftarrow \mathsf{Gen}_{d_j+1}(g_j).$$
Here, we apply Gen_{d_j+1} to the vector-valued function g_j in a component-wise manner.
6. For each $A \in 2^{[3]} \setminus \{\emptyset, [3]\}$, compute
$$\nu_A \leftarrow \mathsf{Gen}_{\overline{d}_A+1}(h_A).$$
7. Output
$$R = \left((\boldsymbol{r}_j)_{j\in[3]}, (s_A)_{A\in 2^{[3]}\setminus\{[3]\}}, (\boldsymbol{\mu}_j)_{j\in[3]}, (\nu_A)_{A\in 2^{[3]}\setminus\{\emptyset,[3]\}}\right).$$

Fig. 2. The randomness generation algorithm of a PSM protocol for $\mathcal{S}_{n,d}$

Notations.

- Let $P_j : B_d \to \mathbb{Z}^{d_j}$ be a map defined as

$$P_j(\boldsymbol{x}) = (\boldsymbol{x}(\overline{d}_j), \boldsymbol{x}(\overline{d}_j + 1), \ldots, \boldsymbol{x}(\overline{d}_{j+1} - 1)), \ \forall \boldsymbol{x} \in B_d.$$

$\mathsf{Enc}(i, f, \boldsymbol{x}_i, R)$**.**

1. Parse $\boldsymbol{x}_i$ as

$$\boldsymbol{x}_i = (P_0(\boldsymbol{x}_i), P_1(\boldsymbol{x}_i), P_2(\boldsymbol{x}_i)) =: (\boldsymbol{x}_{i,0}, \boldsymbol{x}_{i,1}, \boldsymbol{x}_{i,2})$$

2. For each $j \in [3]$, compute

$$\boldsymbol{m}_j(i) = \mathsf{Enc}_{d_j+1}(i, g_j, \boldsymbol{x}_{i,j}, \boldsymbol{\mu}_j).$$

   Here, we apply Enc_{d_j+1} to the vector-valued function g_j in a component-wise manner.
3. For each $A \in 2^{[3]} \setminus \{\emptyset, [3]\}$, compute

$$m_A(i) = \mathsf{Enc}_{\overline{d}_A+1}(i, h_A, (\boldsymbol{x}_{i,j})_{j \in A}, \nu_A).$$

4. Compute $m_\emptyset(i)$ as

$$m_\emptyset(i) = \begin{cases} T_f(\boldsymbol{r}_0 \otimes \boldsymbol{r}_1 \otimes \boldsymbol{r}_2) + s_\emptyset, & \text{if } i = 0, \\ \perp, & \text{otherwise.} \end{cases}$$

5. Output

$$M_i = \left((\boldsymbol{m}_j(i))_{j \in [3]}, (m_A(i))_{A \in 2^{[3]} \setminus \{[3]\}} \right).$$

$\mathsf{Dec}(f, M_0, \ldots, M_{n-1})$**.**

1. For each $j \in [3]$, compute

$$\boldsymbol{a}_j = \mathsf{Dec}_{d_j+1}(\boldsymbol{m}_j(0), \ldots, \boldsymbol{m}_j(n-1)).$$

2. For each $A \in 2^{[3]} \setminus \{\emptyset, [3]\}$, compute

$$b_A = \mathsf{Dec}_{\overline{d}_A+1}(m_A(0), \ldots, m_A(n-1)).$$

   Set $b_\emptyset = m_\emptyset(0)$.
3. Output

$$y = T_f(\boldsymbol{a}_0 \otimes \boldsymbol{a}_1 \otimes \boldsymbol{a}_2) - \sum_{A \in 2^{[3]} \setminus \{[3]\}} b_A.$$

Fig. 3. The encoding and decoding algorithms of a PSM protocol for $\mathcal{S}_{n,d}$

For each $A \in 2^{[3]} \setminus \{\emptyset, [3]\}$, the correctness of $\Pi_{\bar{d}_A+1}$ implies that

$$\begin{aligned} b_A &= h_A((\boldsymbol{x}_{0,j})_{j\in A}, \ldots, (\boldsymbol{x}_{n-1,j})_{j\in A}) \\ &= T_f\left(\bigotimes_{j\in A} \boldsymbol{e}_{I_{\boldsymbol{v}_j}(P_j(\boldsymbol{x}_0)+\cdots+P_j(\boldsymbol{x}_{n-1}))} \otimes \bigotimes_{j\notin A} \boldsymbol{r}_j\right) + s_A \\ &= T_f\left(\bigotimes_{j\in A} \boldsymbol{e}_{\phi_j(\boldsymbol{X})} \otimes \bigotimes_{j\notin A} \boldsymbol{r}_j\right) + s_A. \end{aligned}$$

Since $b_\emptyset = T_f(\boldsymbol{r}_0 \otimes \boldsymbol{r}_1 \otimes \boldsymbol{r}_2) + s_\emptyset$, it holds that

$$\begin{aligned} \sum_{A\neq[3]} b_A &= \sum_{A\neq[3],\emptyset} T_f\left(\bigotimes_{j\in A} \boldsymbol{e}_{\phi_j(\boldsymbol{X})} \otimes \bigotimes_{j\notin A} \boldsymbol{r}_j\right) + T_f(\boldsymbol{r}_0 \otimes \boldsymbol{r}_1 \otimes \boldsymbol{r}_2) + \sum_{A\neq[3]} s_A \\ &= \sum_{A\neq[3],\emptyset} T_f\left(\bigotimes_{j\in A} \boldsymbol{e}_{\phi_j(\boldsymbol{X})} \otimes \bigotimes_{j\notin A} \boldsymbol{r}_j\right) + T_f(\boldsymbol{r}_0 \otimes \boldsymbol{r}_1 \otimes \boldsymbol{r}_2). \end{aligned}$$

In view of Eq. (8), we obtain that

$$\begin{aligned} y &= T_f\left((\boldsymbol{e}_{\phi_0(\boldsymbol{X})} + \boldsymbol{r}_0) \otimes (\boldsymbol{e}_{\phi_1(\boldsymbol{X})} + \boldsymbol{r}_1) \otimes (\boldsymbol{e}_{\phi_2(\boldsymbol{X})} + \boldsymbol{r}_2)\right) \\ &\quad - \sum_{A\neq[3],\emptyset} T_f\left(\bigotimes_{j\in A} \boldsymbol{e}_{\phi_j(\boldsymbol{X})} \otimes \bigotimes_{j\notin A} \boldsymbol{r}_j\right) - T_f(\boldsymbol{r}_0 \otimes \boldsymbol{r}_1 \otimes \boldsymbol{r}_2) \\ &= f(\boldsymbol{x}_0, \ldots, \boldsymbol{x}_{n-1}). \end{aligned}$$

Privacy. For each $j \in [3]$, the privacy of Π_{d_j+1} ensures a simulator Sim_{d_j+1} such that

$$(\boldsymbol{m}_j(0), \ldots, \boldsymbol{m}_j(n-1)) \equiv \mathsf{Sim}_{d_j+1}(g_j(\boldsymbol{x}_{0,j}, \ldots, \boldsymbol{x}_{n-1,j})).$$

For each $A \in 2^{[3]} \setminus \{\emptyset, [3]\}$, the privacy of $\Pi_{\bar{d}_A+1}$ ensures a simulator $\mathsf{Sim}_{\bar{d}_A+1}$ such that

$$(m_A(0), \ldots, m_A(n-1)) \equiv \mathsf{Sim}_{\bar{d}_A+1}\left(h_A((\boldsymbol{x}_{0,j})_{j\in A}, \ldots, (\boldsymbol{x}_{n-1,j})_{j\in A})\right).$$

The distribution of $(m_\emptyset(0), \ldots, m_\emptyset(n-1))$ is simulated from $m_\emptyset(0) = T_f(\boldsymbol{r}_0 \otimes \boldsymbol{r}_1 \otimes \boldsymbol{r}_2) + s_\emptyset$. Since the Π_{d_j+1}'s and $\Pi_{\bar{d}_A+1}$'s are run with independent randomness, the messages $(M_0, \ldots, M_{n-1})$ is simulated from the following elements:

$$(g_j(\boldsymbol{x}_{0,j}, \ldots, \boldsymbol{x}_{n-1,j}))_{j\in[3]},\ (h_A((\boldsymbol{x}_{0,j})_{j\in A}, \ldots, (\boldsymbol{x}_{n-1,j})_{j\in A}))_{A\neq[3],\emptyset},\ m_\emptyset(0). \tag{9}$$

We will see that the distribution of the elements in (9) is the uniform distribution over the following set:

$$\mathcal{R} = \left\{ \left((\boldsymbol{a}_j)_{j\in[3]}, (b_A)_{A\neq[3]}\right) : \begin{array}{c} \boldsymbol{a}_j \in \mathbb{F}^{\ell_j},\ b_A \in \mathbb{F}, \\ T_f(\boldsymbol{a}_0 \otimes \boldsymbol{a}_1 \otimes \boldsymbol{a}_2) - \sum_{A\neq[3]} b_A = f(\boldsymbol{x}_0, \ldots, \boldsymbol{x}_{n-1}) \end{array} \right\}.$$

Indeed, we have that

$$g_j(\boldsymbol{x}_{0,j},\dots,\boldsymbol{x}_{n-1,j}) = \boldsymbol{e}_{I_{\boldsymbol{v}_j}(\boldsymbol{x}_{0,j}+\cdots+\boldsymbol{x}_{n-1,j})} + \boldsymbol{r}_j = \boldsymbol{e}_{\phi_j(\boldsymbol{X})} + \boldsymbol{r}_j.$$

Since the $\boldsymbol{r}_j$'s are independent uniformly random vectors, the $g_j(\boldsymbol{x}_{0,j},\dots,\boldsymbol{x}_{n-1,j})$'s are also independent uniformly random vectors.

Furthermore, we have that

$$\begin{aligned} h_A((\boldsymbol{x}_{0,j})_{j\in A},\dots,(\boldsymbol{x}_{n-1,j})_{j\in A}) &= T_f\left(\bigotimes_{j\in A} \boldsymbol{e}_{I_{\boldsymbol{v}_j}(\boldsymbol{x}_{0,j}+\cdots+\boldsymbol{x}_{n-1,j})} \otimes \bigotimes_{j\notin A} \boldsymbol{r}_j\right) + s_A \\ &= T_f\left(\bigotimes_{j\in A} \boldsymbol{e}_{\phi_j(\boldsymbol{X})} \otimes \bigotimes_{j\notin A} \boldsymbol{r}_j\right) + s_A \end{aligned}$$

Since $(s_A)_{A\neq[3],\emptyset}$ is uniformly distributed, the $h_A((\boldsymbol{x}_{0,j})_{j\in A},\dots,(\boldsymbol{x}_{n-1,j})_{j\in A})$'s are independent uniformly random elements in $\mathbb{F}$.

Due to the constraint of $\mathcal{R}$, $b_\emptyset$ is uniquely determined from the other elements $(\boldsymbol{a}_j)_{j\in[3]}, (b_A)_{A\neq\emptyset,[3]}$ via the following formula:

$$b_\emptyset = T_f(\boldsymbol{a}_0\otimes\boldsymbol{a}_1\otimes\boldsymbol{a}_2) - \sum_{A\neq\emptyset,[3]} b_A - f(\boldsymbol{X}).$$

On the other hand, on view of (8), we have that

$$\begin{aligned} & m_\emptyset(0) \\ &= T_f(\boldsymbol{r}_0\otimes\boldsymbol{r}_1\otimes\boldsymbol{r}_2) + s_\emptyset \\ &= T_f\left(\bigotimes_{j\in[3]} (\boldsymbol{e}_{\phi_j(\boldsymbol{X})} + \boldsymbol{r}_j)\right) - \sum_{A\neq\emptyset,[3]} T_f\left(\bigotimes_{j\in A} \boldsymbol{e}_{\phi_j(\boldsymbol{X})} \otimes \bigotimes_{j\notin A} \boldsymbol{r}_j\right) - f(\boldsymbol{X}) + s_\emptyset \\ &= T_f\left(\bigotimes_{j\in[3]} g_j(\boldsymbol{x}_{0,j},\dots,\boldsymbol{x}_{n-1,j})\right) - \sum_{A\neq\emptyset,[3]} h_A((\boldsymbol{x}_{0,j})_{j\in A},\dots,(\boldsymbol{x}_{n-1,j})_{j\in A}) \\ & \quad -f(\boldsymbol{X}) + \sum_{A\neq[3]} s_A \\ &= T_f\left(\bigotimes_{j\in[3]} g_j(\boldsymbol{x}_{0,j},\dots,\boldsymbol{x}_{n-1,j})\right) - \sum_{A\neq\emptyset,[3]} h_A((\boldsymbol{x}_{0,j})_{j\in A},\dots,(\boldsymbol{x}_{n-1,j})_{j\in A}) \\ & \quad -f(\boldsymbol{X}) \end{aligned}$$

Therefore, $m_\emptyset(0)$ is uniquely determined from the other elements in (9) in the same formula as $b_\emptyset$. We thus conclude that the elements in (9) are uniformly distributed over $\mathcal{R}$.

Since $\mathcal{R}$ depends only on f and $f(\boldsymbol{x}_0,\dots,\boldsymbol{x}_{n-1})$, the elements in (9) and hence the messages $(M_0,\dots,M_{n-1})$ are simulated only from f and $f(\boldsymbol{x}_0,\dots,\boldsymbol{x}_{n-1})$. Below, we provide the overall description of the simulator for Π.

$\mathsf{Sim}(f, f(\boldsymbol{x}_0, \ldots, \boldsymbol{x}_{n-1}))$

1. For each $j \in [3]$, choose $\boldsymbol{a}_j \leftarrow_{\$} \mathbb{F}^{\ell_j}$.
2. For each $A \in 2^{[3]} \setminus \{\emptyset, [3]\}$, choose $b_A \leftarrow_{\$} \mathbb{F}$ and set
$$b_\emptyset = T_f(\boldsymbol{a}_0 \otimes \boldsymbol{a}_1 \otimes \boldsymbol{a}_2) - \sum_{A \neq \emptyset, [3]} b_A - f(\boldsymbol{x}_0, \ldots, \boldsymbol{x}_{n-1})$$
3. For each $j \in [3]$, run the simulator Sim_{d_j+1} of Π_{d_j+1} on input $\boldsymbol{a}_j$ and obtain
$$(\tilde{\boldsymbol{m}}_j(0), \ldots, \tilde{\boldsymbol{m}}_j(n-1)) \leftarrow \mathsf{Sim}_{d_j+1}(\boldsymbol{a}_j).$$
4. For each $A \in 2^{[3]} \setminus \{[3], \emptyset\}$, run the simulator $\mathsf{Sim}_{\overline{d}_A+1}$ of $\Pi_{\overline{d}_A+1}$ on input b_A and obtain
$$(\tilde{m}_A(0), \ldots, \tilde{m}_A(n-1)) \leftarrow \mathsf{Sim}_{\overline{d}_A+1}(b_A).$$
Set
$$\tilde{m}_\emptyset(i) = \begin{cases} b_\emptyset, & \text{if } i = 0, \\ \perp, & \text{otherwise.} \end{cases}$$
5. Set $\tilde{M}_i = ((\tilde{\boldsymbol{m}}_j(i))_{j \in [3]}, (\tilde{m}_A(i))_{A \neq [3]})$ and output $(\tilde{M}_0, \ldots, \tilde{M}_{n-1})$.

Complexity. Note that for any $d', d'' \in \mathbb{N}$ with $d'' \leq d'$, if Π' is a PSM protocol for $\mathcal{S}_{n,d'}$ with universal reconstruction, then it is also a PSM protocol for $\mathcal{S}_{n,d''}$ with universal reconstruction. This is because we can view $B^n_{d''}$ as a subset of $B^n_{d'}$ in a straightforward way. Thus, we may assume that $\mathrm{Comm}(\Pi_{d''}) \leq \mathrm{Comm}(\Pi_{d'})$, $\mathrm{Rand}(\Pi_{d''}) \leq \mathrm{Rand}(\Pi_{d'})$ and $\mathrm{Comp}(\Pi_{d''}) \leq \mathrm{Comp}(\Pi_{d'})$ if $d'' \leq d'$.

We analyze $\mathrm{Rand}(\Pi)$. The $\boldsymbol{r}_j$'s are vectors of dimension $\ell_j = O(n^{d_j})$. The tuple $(s_A)_{A \in 2^{[3]} \setminus \{[3]\}}$ is a bit string of constant length. For each $j \in [3]$, $\boldsymbol{\mu}_j$ is common randomness for ℓ_j invocations of Π_{d_j+1} and hence is of size $\ell_j \cdot \mathrm{Rand}(\Pi_{d_j+1})$. For each $A \neq [3], \emptyset$, ν_A is common randomness for $\Pi_{\overline{d}_A+1}$ and hence is of size $\mathrm{Rand}(\Pi_{\overline{d}_A+1})$. Since $d_0 \leq d_1 \leq d_2$, we have that

$$\begin{aligned} \mathrm{Rand}(\Pi) &= \sum_{j \in [3]} \ell_j + O(1) + \sum_{j \in [3]} \ell_j \cdot \mathrm{Rand}(\Pi_{d_j+1}) + \sum_{A \neq [3], \emptyset} \mathrm{Rand}(\Pi_{\overline{d}_A+1}) \\ &= O(n^{d_2} \cdot \mathrm{Rand}(\Pi_{d_2+1}) + \mathrm{Rand}(\Pi_{d_1+d_2+1})) \\ &= O(n^{d_2} \cdot \mathrm{Rand}(\Pi_{d_2+1}) + \mathrm{Rand}(\Pi_{d-d_0})). \end{aligned}$$

The analysis of $\mathrm{Comm}(\Pi)$ is similar.

We analyze $\mathrm{Comp}(\Pi)$. First, the computation time of Steps 1 and 2 of Gen is $O(\max_j\{\ell_j\}) + O(1) = O(n^d)$ since $\ell_j = O(n^{d_j})$ and $d_j < d$. The size of the truth table of g_j is $\binom{n+d_j}{d_j}$ since each component of g_j belongs to $\mathcal{S}_{n,d_j+1}$.

The complexity of computing each element of the truth table of g_j is $O(\ell_j) + \mathrm{poly}(d_j, \log n) = n^{O(d_j)}$ since computing $I_{\boldsymbol{v}_j}(\boldsymbol{x})$ for $\boldsymbol{x} \in \overline{B}_{n,d_j+1}$ can be done in time $\mathrm{poly}(d_j, \log \ell_j, \log n) = \mathrm{poly}(d_j, \log n)$ and $\boldsymbol{e}_{I_{\boldsymbol{v}_j}(\boldsymbol{x})}$ and $\boldsymbol{r}_j$ are ℓ_j-dimensional vectors. Thus, the computation time of Step 3 of Gen is $\sum_{j\in[3]} \ell_j \cdot \binom{n+d_j}{d_j} \cdot n^{O(d_j)} = (n+d)^{O(d)}$. The size of the truth table of each h_A is $\binom{n+\overline{d}_A}{\overline{d}_A}$ since $h_A \in \mathcal{S}_{n,\overline{d}_A+1}$. Note that if $A = \{0\}$, then

$$
\begin{aligned}
&T_f\left(\bigotimes_{j\in A} \boldsymbol{e}_{I_{\boldsymbol{v}_j}(\boldsymbol{x}_{0,j}+\cdots+\boldsymbol{x}_{n-1,j})} \otimes \bigotimes_{j\notin A} \boldsymbol{r}_j\right) \\
&= T_f\left(\boldsymbol{e}_{I_{\boldsymbol{v}_0}(\boldsymbol{x}_{0,0}+\cdots+\boldsymbol{x}_{n-1,0})} \otimes \sum_{\lambda_1\in[\ell_1],\lambda_2\in[\ell_2]} (\boldsymbol{r}_1(\lambda_1)\boldsymbol{r}_2(\lambda_2))\boldsymbol{e}_{\lambda_1}\otimes\boldsymbol{e}_{\lambda_2}\right) \\
&= \sum_{\lambda_1\in[\ell_1],\lambda_2\in[\ell_2]} (\boldsymbol{r}_1(\lambda_1)\boldsymbol{r}_2(\lambda_2)) \cdot T_f\left(\boldsymbol{e}_{I_{\boldsymbol{v}_0}(\boldsymbol{x}_{0,0}+\cdots+\boldsymbol{x}_{n-1,0})} \otimes \boldsymbol{e}_{\lambda_1}\otimes\boldsymbol{e}_{\lambda_2}\right).
\end{aligned}
$$

Applying a similar argument to other A's, we can see that the complexity of computing a component of the truth table of h_A is $\sum_{j\in A} \mathrm{poly}(d_j, \log n) + \prod_{j\notin A} O(\ell_j) = n^{O(d)}$. Thus, the computation time of Step 4 of Gen is $\sum_A \binom{n+\overline{d}_A}{\overline{d}_A} \cdot n^{O(d)} = (n+d)^{O(d)}$. The total computation time of Gen is $(n+d)^{O(d)} + O(n^{d_2} \cdot \mathrm{Comp}(\Pi_{d_2}) + \mathrm{Comp}(\Pi_{d-d_0}))$.

Second, the computation time of Steps 2 and 3 of Enc is at most $O(n^{d_2} \cdot \mathrm{Comp}(\Pi_{d_2}) + \mathrm{Comp}(\Pi_{d-d_0}))$. The computation time of Step 4 is at most $O(\ell_0\ell_1\ell_2) = n^{O(d)}$ since the most computationally costly part is computing $T_f(\boldsymbol{r}_0 \otimes \boldsymbol{r}_1 \otimes \boldsymbol{r}_2)$. Thus, the total computation time of Enc is $n^{O(d)} + O(n^{d_2} \cdot \mathrm{Comp}(\Pi_{d_2}) + \mathrm{Comp}(\Pi_{d-d_0}))$.

Third, the computation time of Steps 1 and 2 of Dec is at most $O(n^{d_2} \cdot \mathrm{Comp}(\Pi_{d_2}) + \mathrm{Comp}(\Pi_{d-d_0}))$. The computation time of Step 3 is at most $O(\ell_0\ell_1\ell_2) = n^{O(d)}$ since the most computationally costly part is computing $T_f(\boldsymbol{a}_0 \otimes \boldsymbol{a}_1 \otimes \boldsymbol{a}_2)$. Thus, the total computation time of Dec is $n^{O(d)} + O(n^{d_2} \cdot \mathrm{Comp}(\Pi_{d_2}) + \mathrm{Comp}(\Pi_{d-d_0}))$.

Therefore, we conclude that $\mathrm{Comp}(\Pi) = (n+d)^{O(d)} + O(n^{d_2} \cdot \mathrm{Comp}(\Pi_{d_2}) + \mathrm{Comp}(\Pi_{d-d_0}))$. □

By applying Theorem 3 to the PSM protocol with universal reconstruction in Theorem 1, we obtain the following corollary.

Corollary 1. *For any $n, d \in \mathbb{N}$, there exists an n-party PSM protocol Π for $\mathcal{S}_{n,d}$ such that*

$$\mathrm{Comm}(\Pi) = O(n^\delta),\ \mathrm{Rand}(\Pi) = O(n^\delta),\ \mathrm{Comp}(\Pi) = 2^{O(n^{\delta-1})},$$

where $\delta = 2\lfloor d/3 \rceil + 1$.

Proof. We show that there exist parameters (d_0, d_1, d_2) such that $d_0 \le d_1 \le d_2$, $d_0 + d_1 + d_2 = d - 1$, and $\delta' := \max\{2d_2 + 1, d - d_0\} = 2\lfloor d/3 \rceil + 1$. Indeed,

if $d = 3t$ for some $t \in \mathbb{N}$, we set $d_0 = t - 1$ and $d_1 = d_2 = t$. Then $\delta' = 2t + 1 = 2\lfloor d/3 \rceil + 1$. If $d = 3t - 1$, we set $d_0 = t - 2$ and $d_1 = d_2 = t$. Then $\delta' = 2t+1 = 2(d+1)/3+1 = 2\lfloor d/3 \rceil+1$. If $d = 3t-2$, we set $d_0 = d_1 = d_2 = t-1$. Then $\delta' = 2t - 1 = 2(d-1)/3 + 1 = 2\lfloor d/3 \rceil + 1$.

In Theorem 1, for any $d' \in \mathbb{N}$, we have shown an n-party PSM protocol $\Pi_{d'}$ with universal reconstruction for $\mathcal{S}_{n,d'}$ such that $\mathrm{Comm}(\Pi) = O(n^{d'})$, $\mathrm{Rand}(\Pi) = O(n^{d'})$ and $\mathrm{Comp}(\Pi) = 2^{O(n^{d'-1})}$. Then, we have that

$$\begin{aligned} n^{d_2} \cdot \mathrm{Comm}(\Pi_{d_2+1}) = O(n^{2d_2+1}),\ \mathrm{Comm}(\Pi_{d-d_0}) = O(n^{d-d_0}),\\ n^{d_2} \cdot \mathrm{Rand}(\Pi_{d_2+1}) = O(n^{2d_2+1}),\ \mathrm{Rand}(\Pi_{d-d_0}) = O(n^{d-d_0}),\\ n^{d_2} \cdot \mathrm{Comp}(\Pi_{d_2+1}) = 2^{O(n^{d_2})},\ \mathrm{Comp}(\Pi_{d-d_0}) = 2^{O(n^{d-d_0-1})}. \end{aligned}$$

Therefore, applying Theorem 3, we obtain a PSM protocol Π for $\mathcal{S}_{n,d}$ such that $\mathrm{Comm}(\Pi) = O(n^{\delta})$, $\mathrm{Rand}(\Pi) = O(n^{\delta})$ and $\mathrm{Comp}(\Pi) = 2^{O(n^{\delta-1})}$. □

If we apply Theorem 3 to the PSM protocol with universal reconstruction in Proposition 3, we obtain a computationally-efficient PSM protocol for $\mathcal{S}_{n,d}$ at the cost of slightly increasing communication complexity.

Corollary 2. *For any $n, d \in \mathbb{N}$, there exists an n-party PSM protocol Π for $\mathcal{S}_{n,d}$ such that*

$$\begin{aligned} \mathrm{Comm}(\Pi) = O((n+1)^{\delta}(\log n)d),\ \mathrm{Rand}(\Pi) = O((n+1)^{\delta}(\log n)d),\\ \mathrm{Comp}(\Pi) = (n+d)^{O(d)}, \end{aligned}$$

where $\delta = 2\lfloor d/3 \rceil + 1$.

7 Applications to Related Models

7.1 Ad-hoc PSM

A k-out-of-n ad-hoc PSM protocol for a k-input function f is an n-party PSM protocol where only a subset S of k input parties, where S is not known in advance, actually sends messages and the referee is able to evaluate f on the subset of inputs $(x_i)_{i \in S}$. The privacy requirements are that (1) if exactly k parties send messages, then the referee learns $f((x_i)_{i \in S})$ but nothing more, and (2) if less than k parties send messages, then the referee learns no information. See Appendix B or [7] for the formal definition. In this paper, we assume that at most k parties send messages, and do not consider the scenario where more than k parties may participate in the protocol.

As shown in [7], ad-hoc PSM can be realized trivially by preparing the common randomness of a k-party PSM protocol for f independently for each potential subset, but it results in a large communication overhead $\binom{n}{k}$. The following theorem shows that a k-out-of-n ad-hoc PSM protocol Π' for a symmetric function f can be efficiently obtained from an n-out-of-n ad-hoc PSM protocol Π for an n-input symmetric function $\hat{f}$ that is related to f.

Theorem 4. *If there exists an n-out-of-n ad-hoc PSM protocol Π for $\mathcal{S}_{n,d}$, then there exists a k-out-of-n ad-hoc PSM protocol Π' for $\mathcal{S}_{k,d}$ with*

$$\begin{aligned}\mathrm{Comm}(\Pi') &= O(n \cdot \max\{\mathrm{Comm}(\Pi), \log n\}),\\ \mathrm{Rand}(\Pi') &= \mathrm{Rand}(\Pi) + O(n^2 \cdot \max\{\mathrm{Comm}(\Pi), \log n\}).\end{aligned}$$

Proof. First, we define notations. Let $f : B_d^k \to \{0,1\} \in \mathcal{S}_{k,d}$ be a k-input symmetric function. We define an augmented function $\hat{f}$ of f as an n-input symmetric function $\hat{f} : B_d^n \to \{0,1\} \in \mathcal{S}_{n,d}$ such that if there are at most k inputs $\boldsymbol{x}_{i_0}, \ldots, \boldsymbol{x}_{i_{k'-1}} \in B_d^n$ ($k' \leq k$) that are not $\mathbf{0}$, then the function outputs

$$f(\boldsymbol{x}_{i_0}, \ldots, \boldsymbol{x}_{i_{k'-1}}, \underbrace{\mathbf{0}, \ldots, \mathbf{0}}_{k-k'})$$

and if there are more than k inputs that are not $\mathbf{0} \in B_d^n$, then the function is undefined (in the latter case, we do not care about what $\hat{f}$ outputs).

We construct Π' based on $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ as follows.

Randomness generation. Given a k-input symmetric function $f \in \mathcal{S}_{k,d}$,

1. Let $\hat{f} \in \mathcal{S}_{n,d}$ be an augmented function of f.
2. Run $\mathsf{Gen}(\hat{f})$ to generate randomness r for Π.
3. For each $j \in [n]$, let $m_{\mathbf{0},j}$ be the message that P_j sends in Π with randomness r and input $\mathbf{0} \in B_d$. Share $m_{\mathbf{0},j}$ using a k-out-of-n secret sharing scheme; let $m_{\mathbf{0},j,i}$ be the i-th share.
4. Output $r' = (r, (m_{\mathbf{0},j,i})_{i,j\in[n]})$.

Encoding. Given an input $\boldsymbol{x}_i$ and randomness r',

1. Run $\mathsf{Enc}(i, \hat{f}, \boldsymbol{x}_i, r)$ to obtain a message m_i for Π.
2. Output $m'_i = (m_i, (m_{\mathbf{0},j,i})_{j\in[n]\setminus\{i\}})$.

Decoding. Given k messages $(m'_i)_{i\in S}$, where $S \subseteq [n]$ is a k-sized subset,

1. Reconstruct the message $m_{\mathbf{0},j}$ for all $j \notin S$ from k shares $(m_{\mathbf{0},j,i})_{i\in S}$. Note that if $i \in S$ and $j \notin S$, the share $m_{\mathbf{0},j,i}$ is included in m'_i.
2. Output $y = \mathsf{Dec}(\hat{f}, (m_i)_{i\in S}, (m_{\mathbf{0},j})_{j\notin S})$.

The correctness follows since for any k-sized subset S and any k inputs $(\boldsymbol{x}_i)_{i\in S}$, it holds that

$$\begin{aligned}y &= \mathsf{Dec}(\hat{f}, (m_i)_{i\in S}, (m_{\mathbf{0},j})_{j\notin S})\\ &= \mathsf{Dec}(\hat{f}, (\mathsf{Enc}(i, \hat{f}, \boldsymbol{x}_i, r))_{i\in S}, (\mathsf{Enc}(j, \hat{f}, \mathbf{0}, r))_{j\notin S})\\ &= \hat{f}((\boldsymbol{x}_i)_{i\in S}, \mathbf{0}, \ldots, \mathbf{0})\\ &= f((\boldsymbol{x}_i)_{i\in S}).\end{aligned}$$

As for privacy, assume that the referee receives exactly k messages $(m'_i)_{i\in S}$ from parties in a k-sized subset S. He learns k messages $(m_i)_{i\in S}$ for Π, each of which encodes an input $\boldsymbol{x}_i$, and $n-k$ messages $(m_{\mathbf{0},j})_{j\in S}$ for Π, each of which encodes $\mathbf{0}$. It can be seen that the referee learns no information beyond

the above messages. Note that he sees k shares $(m_{\mathbf{0},j,i})_{i\in S}$ of $m_{\mathbf{0},j}$ for each $j \notin S$ and $k-1$ shares $(m_{\mathbf{0},j,i})_{i\in S\setminus\{j\}}$ of $m_{\mathbf{0},j}$ for each $j \in S$. From the security of k-out-of-n secret sharing, the k shares $(m_{\mathbf{0},j,i})_{i\in S}$ can be simulated from $m_{\mathbf{0},j}$ and the $k-1$ shares $(m_{\mathbf{0},j,i})_{i\in S\setminus\{j\}}$ are just random elements. Therefore, the privacy of Π follows since n messages $((m_i)_{i\in S}, (m_{\mathbf{0},j})_{j\in S})$ can be simulated from $\hat{f}((\boldsymbol{x}_i)_{i\in S}, \mathbf{0}, \ldots, \mathbf{0}) = f((\boldsymbol{x}_i)_{i\in S})$.

Next, assume that the referee receives less than k messages $(m'_i)_{i\in S}$, where S is a k'-sized subset for $k' < k$. He learns k' messages $(m_i)_{i\in S}$ for Π, each of which encodes an input $\boldsymbol{x}_i$. In this case, we can see that he learns no information beyond them. Indeed, although he sees shares $(m_{\mathbf{0},j,i})_{i\in S}$ for $j \notin S$ and $(m_{\mathbf{0},j,i})_{i\in S\setminus\{j\}}$ for $j \in S$, it leaks nothing since the number of shares is at most $k' < k$. Therefore, since Π is n-out-of-n ad-hoc PSM, k' messages for Π leaks no information to the referee and hence the privacy of Π follows.

As for complexity, it is known that there is a k-out-of-n secret sharing scheme which has share size $O(\max\{\ell, \log n\})$, where ℓ is the bit-length of secrets [22]. The bit-length of each share $m_{\mathbf{0},j,i}$ is thus $O(\max\{\mathrm{Comm}(\Pi), \log n\})$. Therefore, Π' has the communication and randomness complexity in the statement. □

Applying the above compiler to the protocol in Corollary 1, we obtain an efficient ad-hoc PSM protocol for symmetric functions. Prior to this work, the best known communication complexity was $O((n+1)^{d+2}d^2 \log n)$, which is obtained by applying the original transformation of [7] to the PSM protocol in [11].

Corollary 3. *There exists a k-out-of-n ad-hoc PSM protocol Π for $\mathcal{S}_{k,d}$ such that* $\mathrm{Comm}(\Pi) = O(n^{\delta+1})$ *and* $\mathrm{Rand}(\Pi) = O(n^{\delta+2})$*, where* $\delta = 2\lfloor d/3 \rceil + 1$.

Proof. Let Π_0 be an n-party PSM protocol for $\mathcal{S}_{n,d}$ obtained from Corollary 1. We show that Π_0 is n-out-of-n ad-hoc PSM. This does not trivially follow since the privacy of PSM does not necessarily guarantee that the referee should learn no information if less than n parties actually send messages. From its construction described in Figs. 2 and 3, it is sufficient to show that the underlying PSM protocol with universal reconstruction is n-out-of-n ad-hoc PSM. This is because if so, the referee learns no information from less than n messages $\boldsymbol{m}_j(i)$'s (or $m_A(i)$'s). We can see that the PSM protocol with universal reconstruction obtained in Theorem 1 is n-out-of-n ad-hoc PSM, since each message is masked by a share of zero and hence less than n messages are just uniformly random elements. Finally, apply Theorem 4 to Π_0. □

Beimel et al. [7] presented a construction of k-out-of-n ad-hoc PSM based on *k-party* PSM, while the construction in Theorem 4 derives ad-hoc PSM from *n-party* PSM. If the number of actual parties is much smaller than the total number of potential parties (e.g., $k = o(\log n)$), then the following protocol achieves better communication complexity than the one in Corollary 3.

Corollary 4. *There exists a k-out-of-n ad-hoc PSM protocol Π for $\mathcal{S}_{k,d}$ with* $\mathrm{Comm}(\Pi) = 2^{O(k)}k^{\delta}\log n$ *and* $\mathrm{Rand}(\Pi) = 2^{O(k)}k^{\delta}\log n$*, where* $\delta = 2\lfloor d/3 \rceil + 1$.

Note that the above protocol improves the state-of-the-art communication complexity of ad-hoc PSM for symmetric functions which is $2^{O(k)}d^{k/2}\log n$ [8], if $k \gg d$.

7.2 Robust PSM

Consider a situation where the referee colludes with a subset of input parties. If the referee receives messages from honest parties, then it is impossible to prevent him from evaluating a function f on all possible inputs of the corrupted parties using their randomness for encoding. More formally, let $f : X_0 \times \cdots \times X_{n-1} \to Y$ be an n-input function. Let $H \subseteq [n]$ and $x_H = (x_i)_{i\in H} \in \prod_{i\in H} X_i$. If honest parties send messages encoding inputs x_H, then the referee can learn the function $f|_{H,x_H} : \prod_{i\notin H} X_i \to Y$ (called a residual function [14]) defined as

$$f|_{H,x_H}(x_{\overline{H}}) = f(x_H, x_{\overline{H}}),\ \forall x_{\overline{H}} = (x_i)_{i\in\overline{H}} \in \prod_{i\notin H} X_i.$$

A PSM protocol is said to be t-robust if it ensures that the referee colluding with t parties cannot learn anything beyond the residual function, which is the best possible due to the above generic attack. See Appendi B or [5] for the formal definition. A 0-robust PSM protocol corresponds to a standard PSM protocol.

Benhamouda et al. [9] proposed a compiler from a (non-robust) PSM protocol to a t-robust PSM protocol for the same function. By applying it to our PSM protocol in Corollary 1, we obtain a t-robust PSM protocol for symmetric functions with communication complexity $n^{2d/3+o(d)}(\max\{n,d\})^{(1+o(1))t}$, improving on the state-of-the-art protocol in [11] which has communication complexity $n^{d+o(d)}(\max\{n,d\})^{(1+o(1))t}$.

Corollary 5. *There exists a t-robust n-party PSM protocol Π for $\mathcal{S}_{n,d}$ such that* $\mathrm{Comm}(\Pi) = n^{\delta}p^{t+O(1)}$ *and* $\mathrm{Rand}(\Pi) = n^{\delta}p^{t+O(1)}$*, where $\delta = 2\lfloor d/3\rceil + 1$ and p is the smallest prime power such that $p \geq \max\{n,d\}$.*

Acknowledgements

This work was supported in part by JST CREST Grant Number JPMJCR22M1, JST AIP Acceleration Research JPMJCR22U5, and JSPS KAKENHI Grant Numbers 21K17702, 23H00479, and 24K20775.

References

1. Applebaum, B., Holenstein, T., Mishra, M., Shayevitz, O.: The communication complexity of private simultaneous messages, revisited. J. Cryptol. **33**(3), 917–953 (2020)
2. Assouline, L., Liu, T.: Multi-party psm, revisited: Improved communication and unbalanced communication. In: Theory of Cryptography. pp. 194–223 (2021)

3. Ball, M., Randolph, T.: A note on the complexity of private simultaneous messages with many parties. In: 3rd Conference on Information-Theoretic Cryptography, ITC 2022. pp. 7:1–7:12 (2022)
4. Beimel, A., Gabizon, A., Ishai, Y., Kushilevitz, E.: Distribution design. In: Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science. pp. 81–92. ITCS '16 (2016)
5. 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)
6. Beimel, A., Ishai, Y., Kumaresan, R., Kushilevitz, E.: On the cryptographic complexity of the worst functions. In: Theory of Cryptography. pp. 317–342 (2014)
7. Beimel, A., Ishai, Y., Kushilevitz, E.: Ad hoc PSM protocols: Secure computation without coordination. In: Advances in Cryptology – EUROCRYPT 2017, Part III. pp. 580–608 (2017)
8. Beimel, A., Kushilevitz, E., Nissim, P.: The complexity of multiparty PSM protocols and related models. In: Advances in Cryptology – EUROCRYPT 2018, Part II. pp. 287–318 (2018)
9. Benhamouda, F., Krawczyk, H., Rabin, T.: Robust non-interactive multiparty computation against constant-size collusion. In: Advances in Cryptology – CRYPTO 2017, Part I. pp. 391–419 (2017)
10. Data, D., Prabhakaran, M., Prabhakaran, V.M.: On the communication complexity of secure computation. In: Advances in Cryptology - CRYPTO 2014. pp. 199–216 (2014)
11. Eriguchi, R., Ohara, K., Yamada, S., Nuida, K.: Non-interactive secure multiparty computation for symmetric functions, revisited: More efficient constructions and extensions. In: Advances in Cryptology – CRYPTO 2021. pp. 305–334 (2021)
12. Feige, U., Kilian, J., Naor, M.: A minimal model for secure computation (extended abstract). In: Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing. pp. 554–563. STOC '94 (1994)
13. 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)
14. 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)
15. Ishai, Y., Kushilevitz, E.: Private simultaneous messages protocols with applications. In: Proceedings of the Fifth Israeli Symposium on Theory of Computing and Systems. pp. 174–183 (1997)
16. Ishai, Y.: Randomization techniques for secure computation. Secure Multi-Party Computation **10**, 222–248 (2013)
17. Ishai, Y., Kumaresan, R., Kushilevitz, E., Paskin-Cherniavsky, A.: Secure computation with minimal interaction, revisited. In: Advances in Cryptology – CRYPTO 2015. pp. 359–378 (2015)
18. Ishai, Y., Kushilevitz, E., Paskin, A.: Secure multiparty computation with minimal interaction. In: Advances in Cryptology – CRYPTO 2010. pp. 577–594 (2010)
19. Liu, T., Vaikuntanathan, V., Wee, H.: Conditional disclosure of secrets via non-linear reconstruction. In: Advances in Cryptology – CRYPTO 2017. pp. 758–790 (2017)

20. Liu, T., Vaikuntanathan, V., Wee, H.: Towards breaking the exponential barrier for general secret sharing. In: Advances in Cryptology – EUROCRYPT 2018. pp. 567–596 (2018)
21. Peralta, R.: On the distribution of quadratic residues and nonresidues modulo a prime number. Mathematics of Computation **58**(197), 433–440 (1992)
22. Shamir, A.: How to share a secret. Communications of the ACM **22**(11), 612–613 (1979)
23. Shinagawa, K., Eriguchi, R., Satake, S., Nuida, K.: Private simultaneous messages based on quadratic residues. Designs, Codes and Cryptography **91**(12), 3915–3932 (2023)
24. Shinagawa, K., Nuida, K.: Explicit lower bounds for communication complexity of PSM for concrete functions. In: Progress in Cryptology - INDOCRYPT 2023. pp. 45–61 (2023)

A Optimization of the PSM Protocol from [11]

The protocol of [11] used a different representation of symmetric functions called abelian programs. Let $\mathbb{G}$ be an abelian group. For a function $g : \mathbb{G} \to \{0,1\}$, we define a function $h_g : \mathbb{G}^n \to \{0,1\}$ as

$$h_g(x_0, \ldots, x_{n-1}) = g(x_0 + \cdots + x_{n-1}),\ \forall (x_0, \ldots, x_{n-1}) \in \mathbb{G}^n.$$

We call each h_g an abelian program associated with g and define $\mathcal{A}_{n,\mathbb{G}}$ as the set of all abelian programs. Set $\mathbb{G} = (\mathbb{Z}_{n+1})^{d-1}$, the direct product of $d-1$ copies of $\mathbb{Z}_{n+1}$. Then, a PSM protocol for $\mathcal{A}_{n,\mathbb{G}}$ implies a PSM protocol for $\mathcal{S}_{n,d}$. Indeed, let $f \in \mathcal{S}_{n,d}$. From Proposition 1, there exists $g : \overline{B}_{n,d} \to \{0,1\}$ such that

$$g(\boldsymbol{x}_0 + \cdots + \boldsymbol{x}_{n-1}) = f(\boldsymbol{x}_0, \ldots, \boldsymbol{x}_{n-1}),\ \forall (\boldsymbol{x}_0, \ldots, \boldsymbol{x}_{n-1}) \in B_d^n.$$

We can view B_d and $\overline{B}_{n,d}$ as subsets of $\mathbb{G}$, and no wrap-around occurs in the summation $\boldsymbol{x}_0 + \cdots + \boldsymbol{x}_{n-1}$. Therefore, g can be extended to $g' : \mathbb{G} \to \{0,1\}$ such that

$$h_{g'}(\boldsymbol{x}_0, \ldots, \boldsymbol{x}_{n-1}) = f(\boldsymbol{x}_0, \ldots, \boldsymbol{x}_{n-1}),\ \forall (\boldsymbol{x}_0, \ldots, \boldsymbol{x}_{n-1}) \in B_d^n.$$

Thus, the PSM protocol for $h_{g'}$ implies the one for f.

We show an optimized version of the PSM protocol for $\mathcal{A}_{n,\mathbb{G}}$ in [11] while the original protocol was more complicated since it aimed to guarantee security against collusion between the referee and parties. We prepare some notations. Let $0_{\mathbb{G}}$ denote the identity element of $\mathbb{G}$. Define an abelian group $\overline{\mathbb{G}} = \mathbb{G} \times \{0,1\} = \{(x,y) : x \in \mathbb{G}, y \in \{0,1\}\}$. For a map $g : \mathbb{G} \to \{0,1\}$, we define $\overline{g} : \overline{\mathbb{G}} \to \{0,1\}$ as

$$\overline{g}(x,y) = \begin{cases} g(x), & \text{if } y = 0, \\ 1 - g(x), & \text{if } y = 1. \end{cases}$$

For $a \in \mathbb{G}$, let σ_a be a permutation over $\overline{\mathbb{G}}$ defined as $\sigma_a(x,y) = (x+a, y)$ for all $x \in \mathbb{G}$ and $y \in \{0,1\}$. Let $\sigma\tau$ denote the composition of maps σ, τ over $\overline{\mathbb{G}}$.

The protocol works as follows:

Randomness generation.

1. Choose $n-1$ uniformly random permutations $(\pi_i)_{i \in [n-1]}$ over $\overline{\mathbb{G}}$.
2. Output $r = (\pi_i)_{i \in [n-1]}$.

Encoding. Given a party index i, an abelian program associated with $g : \mathbb{G} \to \{0,1\}$, an input $x_i \in \mathbb{G}$ and randomness r,

- If $i = 0$, output $m_0 = \overline{g}\sigma_{x_0}\pi_0$.
- If $1 \le i \le n-2$, output $m_i = \pi_{i-1}^{-1}\sigma_{x_i}\pi_i$.
- If $i = n-1$, output $m_{n-1} = \pi_{n-2}^{-1}\sigma_{x_{n-1}}(0_{\mathbb{G}}, 0)$.

Decoding. Given n messages $(m_i)_{i \in [n]}$, where $m_0 : \overline{\mathbb{G}} \to \{0,1\}$, $m_1, \ldots, m_{n-2} : \overline{\mathbb{G}} \to \overline{\mathbb{G}}$ and $m_{n-1} \in \overline{\mathbb{G}}$, output $(m_0 m_1 \cdots m_{n-2})(m_{n-1})$.

The correctness is straightforward:

$$\begin{aligned}
&(m_0 m_1 \cdots m_{n-2})(m_{n-1}) \\
&= (\overline{g}\sigma_{x_0}\pi_0)(\pi_0^{-1}\sigma_{x_1}\pi_1)\cdots(\pi_{n-3}^{-1}\sigma_{x_{n-2}}\pi_{n-2})(\pi_{n-2}^{-1}\sigma_{x_{n-1}})(0_{\mathbb{G}}, 0) \\
&= (\overline{g}\sigma_{x_0}\cdots\sigma_{x_{n-1}})(0_{\mathbb{G}}, 0) \\
&= \overline{g}(x_0 + \cdots + x_{n-1}, 0) \\
&= g(x_0 + \cdots + x_{n-1}).
\end{aligned}$$

For privacy, it is sufficient to show that for any pair of inputs $(x_i)_{i\in[n]}, (x'_i)_{i\in[n]} \in \mathbb{G}^n$ and any pair of functions $g, g' : \mathbb{G} \to \{0,1\}$ such that $g(x_0 + \cdots + x_{n-1}) = g'(x'_0 + \cdots + x'_{n-1})$, the distributions of messages are identical to each other. (Given $b := g(x_0 + \cdots + x_{n-1})$, a simulator finds any $(x'_i)_{i\in[n]}$ and any g' such that $g'(x'_0, \ldots, x'_{n-1}) = b$ and simulates messages by executing the protocol on input $(x'_i)_{i\in[n]}$ and the function g'.) We denote by $M(r)$ the distribution of messages when the protocol is executed on input $(x_i)_{i\in[n]}$, the function g, and randomness $r = (\pi_i)_{i\in[n-1]}$. Similarly, we denote by $M'(r)$ the distribution of messages when the protocol is executed on input $(x'_i)_{i\in[n]}$, the function g' and randomness r. It is sufficient to prove that there is a bijection $\phi : \mathcal{R} \to \mathcal{R}$ such that $M(r) = M'(\phi(r))$ for all $r \in \mathcal{R}$, where $\mathcal{R}$ denotes the set of all randomness.

Let $d_i = \sum_{0\le j\le i} x_j$ and $d'_i = \sum_{0\le j\le i} x'_j$ for all $i \in [n]$. Define $N = \{(x, 1 - g(x)) : x \in \mathbb{G}\} = \{(x,y) : \overline{g}(x,y) = 1\}$ and $N' = \{(x, 1 - g'(x)) : x \in \mathbb{G}\} = \{(x,y) : \overline{g}'(x,y) = 1\}$. Note that $|N| = |N'| = |\mathbb{G}|$ and $(d_{n-1}, 0) \in N$ if and only if $(d'_{n-1}, 0) \in N'$ since $g(d_{n-1}) = g'(d_{n-1})$. Thus, there exists a permutation τ over $\overline{\mathbb{G}}$ such that

$$\tau(N) = N' \text{ and } \tau(d_{n-1}, 0) = (d'_{n-1}, 0). \tag{10}$$

Then, we have that $M(r) = M'(\phi(r))$ for all $r = (\pi_i)_{i\in[n-1]}$ from the following observations:

- $\overline{g}'\sigma'_{x_0}\pi'_0 = \overline{g}'\sigma'_{x_0}(\sigma_{-x'_0}\tau\sigma_{x_0}\pi_0) = \overline{g}'\tau\sigma_{x_0}\pi_0 = \overline{g}\sigma_{x_0}\pi_0$ since (10) implies that
$$\overline{g}'\tau(x,y) = 1 \iff \tau(x,y) \in N' \iff (x,y) \in N \iff \overline{g}(x,y) = 1$$
for all $(x,y) \in \overline{\mathbb{G}}$.
- For any $1 \le i \le n-2$, we have
$$\begin{aligned}
&(\pi'_{i-1})^{-1}\sigma_{x'_i}\pi'_i = \pi_{i-1}^{-1}\sigma_{x_i}\pi_i \\
\iff\ &\sigma_{x'_i}(\sigma_{-d'_i}\tau\sigma_{d_i}) = (\sigma_{-d'_{i-1}}\tau\sigma_{d_{i-1}})\sigma_{x_i} \\
\iff\ &\sigma_{x'_i - d'_i}\tau\sigma_{d_i} = \sigma_{-d'_{i-1}}\tau\sigma_{d_{i-1}+x_i},
\end{aligned}$$
which is true since $x'_i - d'_i = -d'_{i-1}$ and $d_i = d_{i-1} + x_i$.
- We have
$$\begin{aligned}
&(\pi'_{n-2})^{-1}\sigma_{x'_{n-1}}(0_{\mathbb{G}}, 0) = \pi_{n-2}^{-1}\sigma_{x_{n-1}}(0_{\mathbb{G}}, 0) \\
\iff\ &\sigma_{x'_{n-1}}(0_{\mathbb{G}}, 0) = (\sigma_{-d'_{n-2}}\tau\sigma_{d_{n-2}})\sigma_{x_{n-1}}(0_{\mathbb{G}}, 0) \\
\iff\ &\sigma_{d'_{n-2}+x'_{n-1}}(0_{\mathbb{G}}, 0) = \tau\sigma_{d_{n-2}+x_{n-1}}(0_{\mathbb{G}}, 0) \\
\iff\ &(d'_{n-1}, 0) = \tau(d_{n-1}, 0),
\end{aligned}$$

which is true from (10).

Regarding complexity, the common randomness of the protocol consists of $O(n)$ permutations over $\overline{\mathbb{G}}$ and a message of each party can be expressed as an $O(|\overline{\mathbb{G}}|)$-dimensional vector or a permutation over $\overline{\mathbb{G}}$. Since $|\overline{\mathbb{G}}| = 2|\mathbb{G}| = O((n+1)^{d-1})$, the communication and randomness complexity are $O((n+1)^d d \log n)$, reducing the complexity of the original protocol [11] by a factor of $O(nd)$. Since the most computationally costly part of the protocol is choosing $O(n)$ random permutations over $\mathbb{G}$, the computational complexity is $O((n+1)^d d \log n)$. This completes the proof of Proposition 3.

B Formal Descriptions of Related Models

B.1 Ad-hoc PSM

Definition 2 (Ad-hoc PSM [7]). *Let X and Y be finite sets. Let $\mathcal{F}$ be a set of k-input functions from X^k to Y. A k-out-of-n ad-hoc PSM protocol Π for $\mathcal{F}$ is a tuple of algorithms $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$, where:*

- $\mathsf{Gen}(f) \to r$*:* Gen *is a randomized algorithm that takes a description of $f \in \mathcal{F}$ as input and outputs common randomness $r \in \{0,1\}^{\mathsf{rand}}$.*
- $\mathsf{Enc}(i, f, x_i, r) \to m_i$*:* Enc *is a deterministic algorithm that takes an index i, a description of $f \in \mathcal{F}$, $x_i \in X$, and $r \in \{0,1\}^{\mathsf{rand}}$ as input and outputs a message $m_i \in \{0,1\}^{\mathsf{comm}_i}$.*
- $\mathsf{Dec}(f, S, (m_i)_{i \in S}) \to y$*:* Dec *is a deterministic algorithm that takes a description of $f \in \mathcal{F}$, a k-sized subset $S \subseteq [n]$ and k messages $(m_i)_{i \in S}$ as input and outputs $y \in Y$.*

satisfying the following properties:

Correctness. *For any $f \in \mathcal{F}$, any k-sized subset $S \subseteq [n]$ and any k inputs $x_S = (x_i)_{i \in S} \in X^k$, it holds that*

$$\mathsf{Dec}(f, S, (\mathsf{Enc}(i, f, x_i, r))_{i \in S}) = f(x_S), \tag{11}$$

with probability 1 over the choice of $r \leftarrow \mathsf{Gen}(f)$.

Privacy.

- *There exists a randomized algorithm* Sim *such that for any $f \in \mathcal{F}$, any k-sized subset $S \subseteq [n]$ and any k inputs $x_S \in X^k$, it holds that*

$$(\mathsf{Enc}(i, f, x_i, r))_{i \in S} \equiv \mathsf{Sim}(f, S, f(x_S)), \tag{12}$$

  where the probabilities are taken over $r \leftarrow \mathsf{Gen}(f)$ and the random coins of Sim.

- *There exists a randomized algorithm* Sim *such that for any $f \in \mathcal{F}$, any $k' < k$, any k'-sized subset $S \subseteq [n]$ and any k' inputs $x_S \in X^{k'}$, it holds that*

$$(\mathsf{Enc}(i, f, x_i, r))_{i \in S} \equiv \mathsf{Sim}(f, S), \tag{13}$$

  where the probabilities are taken over $r \leftarrow \mathsf{Gen}(f)$ and the random coins of Sim.

Proposition 5 ([7]). *If there exists a k-party PSM protocol Π for $\mathcal{S}_{k,d}$, then there exists a k-out-of-n ad-hoc PSM protocol Π' for $\mathcal{S}_{k,d}$ such that*

$$\begin{aligned}\mathrm{Comm}(\Pi') &= 2^{O(k)} \cdot \log n \cdot \max\{\mathrm{Comm}(\Pi), \log n\},\\ \mathrm{Rand}(\Pi') &= 2^{O(k)} \cdot \log n \cdot (\mathrm{Rand}(\Pi) + \max\{\mathrm{Comm}(\Pi), \log n\}).\end{aligned}$$

B.2 Robust PSM

Definition 3 (Robust PSM [5]). *Let $X_0, \ldots, X_{n-1}, Y$ be finite sets. Let $\mathcal{F}$ be a set of functions from $\prod_{i\in[n]} X_i$ to Y. A t-robust n-party PSM protocol Π for $\mathcal{F}$ is a tuple of algorithms $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$, where:*

- $\mathsf{Gen}(f) \to r = (r_i)_{i\in[n]}$*:* Gen *is a randomized algorithm that takes a description of $f \in \mathcal{F}$ as input and outputs correlated randomness $r = (r_i)_{i\in[n]} \in \{0,1\}^{\mathsf{rand}}$.*
- $\mathsf{Enc}(i, f, x_i, r_i) \to m_i$*:* Enc *is a deterministic algorithm that takes an index i, a description of $f \in \mathcal{F}$, $x_i \in X_i$, and r_i as input and outputs a message $m_i \in \{0,1\}^{\mathsf{comm}_i}$.*
- $\mathsf{Dec}(f, (m_i)_{i\in[n]}) \to y$*:* Dec *is a deterministic algorithm that takes a description of $f \in \mathcal{F}$ and $(m_i)_{i\in[n]} \in \prod_{i\in[n]}\{0,1\}^{\mathsf{comm}_i}$ as input and outputs $y \in Y$.*

satisfying the following properties:

Correctness. *For any $f \in \mathcal{F}$ and any $(x_i)_{i\in[n]} \in \prod_{i\in[n]} X_i$, it holds that*

$$\mathsf{Dec}(f, (\mathsf{Enc}(i, f, x_i, r_i))_{i\in[n]}) = f(x_0, \ldots, x_{n-1}), \tag{14}$$

with probability 1 over the choice of $r = (r_i)_{i\in[n]} \leftarrow \mathsf{Gen}(f)$.

Privacy. *For any subset $T \subseteq [n]$ with $|T| \le t$, there exists a randomized algorithm Sim with oracle access to residual functions such that for any $f \in \mathcal{F}$ and any $x_{\overline{T}} \in \prod_{i\in\overline{T}} X_i$, it holds that*

$$\big((\mathsf{Enc}(i, f, x_i, r_i))_{i\in\overline{T}}, (r_i)_{i\in T}\big) \equiv \mathsf{Sim}^{f|_{\overline{T}, x_{\overline{T}}}}(f), \tag{15}$$

where the probabilities are taken over $r = (r_i)_{i\in[n]} \leftarrow \mathsf{Gen}(f)$ and the random coins of Sim.

Proposition 6 ([9]). *If there exists an n-party PSM protocol for a set of functions $\mathcal{F}$ with communication and randomness complexity at most α, then for any t, there exists a t-robust n-party PSM protocol for $\mathcal{F}$ whose communication and randomness complexity are at most $p^{t+O(1)}\alpha$, where d is the maximum size of the input domains of functions in $\mathcal{F}$ and p is the smallest prime power such that $p \ge \max\{n, d\}$.*