The Complexity of Multiparty PSM Protocols and Related Models[⋆]

Amos Beimel[1], Eyal Kushilevitz[2], and Pnina Nissim[1]

[1] Dept. of Computer Science, Ben Gurion University, Beer Sheva, Israel
amos.beimel@gmail.com,pninani@post.bgu.ac.il
[2] Dept. of Computer Science, Technion, Haifa, Israel
eyalk@cs.technion.ac.il

Abstract. We study the efficiency of computing *arbitrary* k-argument functions in the *Private Simultaneous Messages* (PSM) model of [10, 14]. This question was recently studied by Beimel et al. [6], in the two-party case ($k = 2$). We tackle this question in the general case of PSM protocols for $k \geq 2$ parties. Our motivation is two-fold: On one hand, there are various applications (old and new) of PSM protocols for constructing other cryptographic primitives, where obtaining more efficient PSM protocols imply more efficient primitives. On the other hand, improved PSM protocols are an interesting goal on its own. In particular, we pay a careful attention to the case of small number of parties (e.g., $k = 3, 4, 5$), which may be especially interesting in practice, and optimize our protocols for those cases.

Our new upper bounds include a k-party PSM protocol, for any $k > 2$ and any function $f : [N]^k \to \{0,1\}$, of complexity $O(\text{poly}(k) \cdot N^{k/2})$ (compared to the previous upper bound of $O(\text{poly}(k) \cdot N^{k-1})$), and even better bounds for small values of k; e.g., an $O(N)$ PSM protocol for the case $k = 3$. We also handle the more involved case where different parties have inputs of different sizes, which is useful both in practice and for applications.

As applications, we obtain more efficient Non-Interactive secure Multi-Party (NIMPC) protocols (a variant of PSM, where some of the parties may collude with the referee [5]), improved ad-hoc PSM protocols (another variant of PSM, where the subset of participating parties is not known in advance [4, 7]), secret-sharing schemes for strongly-homogeneous access structures with smaller share size than previously known, and better homogeneous distribution designs [4] (a primitive with many cryptographic applications on its own).

1 Introduction

Private simultaneous messages (PSM) protocols, introduced by Feige, Kilian, and Naor [10] and further studied by Ishai and Kushilevitz [14], are secure multi-party computation (MPC) protocols with a minimal communication pattern. In

[⋆] The first and third authors are supported by ISF grants 544/13 and 152/17. The second author is supported by ISF grant 1709/14, BSF grant 2012378, and NSF-BSF grant 2015782.

a PSM protocol for a function f, there are k parties, each of them holds a common random string r and a private input x_i. Each party computes a message based on its input and the common random string and sends it to a referee. The referee, which gets the k messages but does not know the common random string, should be able to compute $f(x_1, \dots, x_k)$ without learning any additional information about the inputs. This model, beside being interesting for its simplicity, implies many cryptographic primitives e.g., constant round secure multi-party protocols (without a common random string) [14, 15], protocols for conditional disclosure of secrets [6], generalized oblivious transfer protocols [14], and zero-information Arthur-Merlin protocols [1]. Several generalizations of PSM protocols have been studied, e.g., non-interactive (or robust) MPC [5], in which security is guaranteed even when the referee colludes with some parties, and ad-hoc PSM protocols [4, 7], in which only a subset of the parties take part in computing the function. It was shown that PSM protocols imply these generalizations [7, 8].

The common random string is crucial for this model – without it only very simple functions can be securely computed against an unbounded adversary; however, given the common random string, there is a PSM protocol (i.e., without any interaction) for any function [10]. Furthermore, many functions can be computed by PSM protocols with short messages, e.g., functions that have small non-deterministic branching programs (i.e., NL) [10] and even functions that have small modular branching programs [14]. Beimel et al. [6] presented improved upper-bounds for computing *arbitrary* functions in the 2-party PSM model. In this paper, we present improved upper-bounds for computing arbitrary functions in the *k-party* case. Then, we show that these improvements imply better complexity for various other primitives, including homogenous distribution designs, t-robust non-interactive secure multi-party computation protocols, and secret-sharing schemes for, so-called, strongly-homogenous access structures. We elaborate below.

1.1 Our results

Our main contributions are constructions of multi-party PSM protocols for every function. Prior to our work, the length of the messages in the best known PSM protocol for an arbitrary function $f : [N]^k \to \{0, 1\}$ was $O(N^{k-1})$ for $k \geq 3$ [10] and $O(N^{1/2})$ for $k = 2$ [6]. We present, for every $k > 2$, a PSM protocol with messages of length $O(\text{poly}(k)N^{k/2})$; for $k = 3, 4, 5$ we present better protocols (see Fig. 1). Understanding the complexity of secure computation with small number of parties is motivated by practical systems and was done in other secure computation contexts (see, e.g., [13]). We also design PSM protocols for functions in which the inputs have different lengths. For example, for any 3-argument function $f : [N^\alpha] \times [N^\alpha] \times [N] \to [N]$ for some $\alpha \geq 1$ (i.e., a function in which the largest two input domains are the same), we design a PSM protocol whose communication complexity is $O(N^{(2\alpha+1)/3})$ (that is, proportional to the geometric average of the domains).

There are two additional advantages for our protocols for $k \geq 6$:

Num. of parties	Complexity	citation
2	$O(N^{1/2})$	[6]
3	$O(N)$	this paper
4	$O(N^{5/3})$	this paper
5	$O(N^{7/3})$	this paper
$k \geq 6$	$O(k^3 \cdot N^{k/2})$	this paper

Fig. 1. Complexity of PSM protocols for an arbitrary functions.

- They can handle long outputs with the same message length, that is for every function $f : [N]^k \to [N^k]$, we construct a PSM protocol with complexity $O(k^3 \cdot N^{k/2})$.[3]
- By increasing the complexity by a poly$(\log N)$ factor, the protocol can be made *locally computable*; that is, each party, holding an input from $[N]$, can compute $\log N$ messages, where each message depends on a *single* bit of the input. This property is useful when we design PSM protocols for some of the applications (e.g., when constructing NIMPC protocols from our PSM protocols; see below).

Following [6], our protocols use techniques from private information retrieval (PIR) [9]; specifically, we use the cube approach of [9], where a function $f : [N]^k \to \{0,1\}$ is represented by a d-dimensional cube, for some integer d. The 2-party PSM protocol of [6] uses a 4-dimensional cube. For $k = 3, 4, 5$ we use a 3-dimensional cube, and for $k \geq 6$ we use a 2-dimensional cube. These turn out to be the optimal values of d for *our* approach. The fact that we can only use integral values for d results in the "somewhat unnatural" exponents in the Fig. 1. As the number of dimensions in our protocols for $k \geq 4$ is smaller than the number of parties, our protocols have to address a few problems that were not relevant in the 2-party protocol of [6].

We note that, by simulation arguments, if for every N, every function $f : [N]^k \to \{0,1\}$ has a k-party PSM protocol with message length $O(N^\alpha)$ for some constant α, then every function $g : [N]^2 \to \{0,1\}$ has a 2-party PSM protocol with message length $O(N^{\alpha/\lfloor k/2 \rfloor})$. Thus, if one can improve the message length for k-party PSM protocols for an arbitrary function beyond $O(N^{k/4})$ for an even k, then this would yield 2-party PSM protocols with message length $O(N^\alpha)$ for $\alpha < 1/2$. Similarly, any improvement for k-party PSM protocols for $k > 6$, would imply an improvement for 6-party PSM protocols compared to our protocols. Thus, to improve the complexity of k-party PSM protocols for arbitrary functions, one might want to start with designing k-party PSM protocols for small values of k.

[3] As the inputs are from $[N]^k$, we can assume without loss of generality that the size of the domain of the outputs of f is at most N^k.

1.2 Applications

We show that our PSM protocols imply the following constructions.

Non-Interactive secure Multi-Party (NIMPC) protocols. A non-interactive MPC protocol [5] is a PSM protocol in which the security is guaranteed even when the referee colludes with some parties. Specifically, a k-party NIMPC protocol is *t-robust* if it is secure against any coalition of the referee and at most t parties, and it is *fully robust* if it is k-robust. Prior to our work, the length of messages in the best known fully robust NIMPC protocol for an arbitrary function $f : [N]^k \to \{0,1\}$ was $O(\text{poly}(\log N, k) \cdot N^k)$ [19] (improving on [5]). No better t-robust protocols were known for any $t > 0$. We construct t-robust NIMPC protocols for any function $f : [N]^k \to \{0,1\}$ with complexity $\tilde{O}(N^{k/2+t})$; that is, we improve the complexity when $t < k/2$. Our construction is based on an information-theoretic transformation of [8] that takes any PSM (i.e., 0-robust NIMPC) protocol and transforms it into a t-robust NIMPC protocol. An immediate application of this transformation yields a t-robust NIMPC protocol with complexity $\tilde{O}(N^{k/2+t+1})$. We use properties of our protocols and of the transformation of [8] to improve the complexity by a factor of N. For example, we construct a fully-robust 3-party NIMPC protocol with complexity $\tilde{O}(N^{2.5})$ (compared to $O(N^3)$ using the transformation as is). Thus, for 3 parties, we improve the complexity of fully-robust NIMPC protocols compared to [19].

Ad-hoc PSM protocols. A k-out-of-n ad-hoc PSM protocol [4, 7] is a PSM protocol with n parties, where only k parties, whose identity is not known in advance, actually participate in the execution of the protocol. For example, think of an election, where only some of the potential voters will end up voting. Using a transformation, presented in [7], from a t-party PSM protocol for a symmetric function to an ad-hoc PSM protocol for the same function, we construct a k-out-of-n ad-hoc PSM protocol for any symmetric function $f : [N]^k \to \{0,1\}$ with communication complexity $O(e^k \cdot k^6 \cdot \log n \cdot N^{k/2})$.

Distribution designs. The goal of a distribution design, introduced in [4], is to find a joint distribution on N random variables that satisfies a given set of constraints on the marginal distributions. Each constraint can either require that two sets of variables are identically distributed or, alternatively, that two sets of variables have disjoint supports. Distribution design generalizes many cryptographic primitives, such as secret-sharing, PSM protocols, and NIMPC protocols. We consider k-homogeneous sets of constraints, where all sets in the constraints are of size exactly k. In [4], it was shown that every k-homogeneous set of constraints without contradictions can be realized by a distribution design such that the size of the support of each variable is $O(\binom{N}{k} k \log N)$. We show that, for every k-homogeneous set of constraints, we can define a symmetric function $f : [N]^k \to \binom{N}{[k]}$ such that any ad-hoc PSM protocol Π for f can be used to construct a distribution design realizing the constraints, where each variable is a message of a party in Π. Using the 2-party PSM of [6] and the two transformation described above, we get a distribution design for every 2-homogeneous set of constraints in which the size of the support of each variable is $O(\log^2 N \cdot \sqrt{N})$. Using our constructions of PSM protocols, we get a distribution design for every k-homogeneous set of constraints in which the support of each variable is of size $O(k^6 e^k \log N \cdot N^{k/2})$. For $3 \leq k \leq 5$, we get better distribution designs.

Conditional Disclosure of Secrets. In Conditional Disclosure of Secrets (CDS) protocols, introduced in [12], there are k parties, a referee, and a function $f : [N]^k \to \{0,1\}$. As in the PSM model, each party gets a common random string r and an input x_i. In addition, all parties (excluding the referee) have a secret s. Each party P_i sends one message to the referee, based on r, x_i and s. The referee, which in the CDS setting knows the inputs $x_1, \ldots, x_k$, should learn s if and only if $f(x_1, \ldots, x_k) = 1$. It was shown in [2] that every PSM protocol for f implies a CDS protocol for f with the same complexity. Thus, our PSM protocols imply CDS protocols. However, there are direct constructions of CDS protocols that are much more efficient than the known PSM protocols. Liu, Vaikuntanathan, and Wee [16] showed a CDS protocol for an arbitrary 2-party function $f : [N] \times [N] \to \{0,1\}$ with communication complexity $2^{O(\sqrt{\log N \log \log N})} = N^{o(1)}$. Very recently, Liu, Vaikuntanathan, and Wee [17] have shown a construction of k-party CDS protocols for any function $f : [N]^k \to \{0,1\}$ with complexity $2^{\tilde{O}(\sqrt{k \log N})}$.

We show that CDS protocols imply secret-sharing schemes for strongly homogeneous access structures. An access structure is t-homogeneous if all minimal authorized sets are of size t. In such access structures all sets of size less than t are unauthorized and there can also be large sets that are unauthorized. We say that an access structure is strongly t-homogeneous if all minimal authorized sets are of size t or $t+1$ and all sets of size $t+1$ are authorized. Strongly 2-homogeneous access structures are called forbidden graph access structures [18] and were studied in, e.g., [3, 2]. Secret-sharing schemes for forbidden bipartite graph access structures with N parties are equivalent to 2-party CDS protocols for functions $f : [N]^2 \to \{0,1\}$. We show that if every k-party function $f : \{0,1\}^k \to \{0,1\}$ has a CDS protocol with communication complexity $\mathrm{Com}(k)$, then every strongly t-homogeneous access structure with k parties can be realized by a secret-sharing scheme with share size $k \cdot \mathrm{Com}(k)$. Combined with the result of [17], we get that every strongly t-homogeneous access structures with k parties can be realized by a secret-sharing scheme with share size $2^{\tilde{O}(\sqrt{k})}$.

1.3 Discussion

CDS protocols vs. PSM protocols. The models of CDS and PSM look similar except that, in CDS protocols, the referee knows the inputs and should learn the secret if and only if some function of the inputs returns 1, while in PSM protocols the referee should learn the value of the function without learning additional information about the inputs. It was shown in [12, 6] that a PSM protocol for f implies a CDS protocol for f with the same complexity. This similarity and the recent dramatic efficiency improvements for CDS protocols [16, 17] may indicate that better PSM protocols also exist.

There are, however, some differences between the models. In [11], it was shown that a CDS protocol for a function $f : [N] \times [N] \to \{0, 1\}$ can be constructed from a CDS for the index function – a function where P_1 holds a list of length N, party P_2 holds an index i and the referee should reconstruct the secret if and only if the ith element in the list is 1. The CDS protocols of [16] for an arbitrary function build on a construction of a CDS protocol for the index function. The PSM of [10] for an arbitrary function can be seen as, implicitly, constructing a PSM for the index function. However, the correctness of PSM protocols (even without the security requirement) implies that the communication complexity of any PSM protocol for the index function is $\Omega(N)$. Thus, for non-binary functions there is a separation between the CDS and PSM models. It is open if such huge separation exists for Boolean functions. We note that our constructions of PSM protocols use PSM protocols for the (k-dimensional) index function, however with shorter lists.

An alternative approach. We next describe an alternative approach for constructing PSM protocols with communication complexity that is better than that of previously known protocols, but is worse than the complexity of the protocols constructed in this paper. We explain some difficulties in applying this approach. Suppose we want to construct a 4-party PSM protocol for a function $f : [N]^4 \to \{0, 1\}$ using a 2-party PSM protocol. Viewing the function f as a two-argument function with domain $[N^2] \times [N^2]$, by [6], it has a 2-party PSM protocol with complexity $O(N)$, where the first message m_1 depends on the inputs of the first two parties and the second message m_2 depends on the inputs of the other two parties. Thus, the first two parties can execute a PSM protocol for computing m_1. This can be done by $O(N)$ invocations of a (2-party) PSM protocol with a binary output. If the parties could have used the PSM of [6], which has complexity $O(N^{0.5})$, then the resulting PSM protocol would have complexity $O(N^{1.5})$. However, we do not know how to use the PSM protocol of [6] here, since it only applies to deterministic functions and the messages of the 2-party PSM protocol depend on the common randomness.[4] Instead, one can apply the protocol of [10], which can be used for randomized functions as well, but has complexity $O(N)$. This gives a protocol with complexity $O(N^2)$. More generally, for k-party functions this approach results in a PSM with complexity $O(N^{3k/4-1})$. This approach is described in more details in Section 6.

Our protocols can be viewed as a generalization of the above approach. Instead of using the PSM protocol of [10] to compute the message of a 2-party protocol, we use a special purpose PSM protocol to compute the message. For example, our k-party protocol from Section 4 can be viewed as simulating the 2-party PSM protocol from Section 3.2.

[4] We can treat the random string generating the message as a common input (or as an input of, e.g, P_1). However, this increases the length of the inputs, resulting in a non-efficient protocol.

2 Preliminaries

In this section we define PSM protocols and describe two PSM protocols that will be used in this paper.

2.1 Private Simultaneous Messages Protocols

In a PSM protocol, k parties $P_1, \ldots, P_k$ hold a common random string r and inputs $x_1, \ldots, x_k$, respectively; each party P_i sends a single message to a referee, based on r and its x_i, so that the referee learns the value of a function $f(x_1, \ldots, x_k)$ but nothing else. It is formally defined as follows:

Definition 2.1 (PSM protocols – Syntax and correctness). *Let $X_1, \ldots, X_k$, and Z be finite domains. A private simultaneous messages (PSM) protocol $\mathcal{P}$, computing a k-argument function $f : X_1 \times \cdots \times X_k \to Z$, consists of:*

- *A finite domain R of common random inputs, and k finite message domains $M_1, \ldots, M_k$.*
- *Message computation algorithms $\textsc{Enc}_1, \ldots, \textsc{Enc}_k$, where $\textsc{Enc}_i : X_i \times R \to M_i$.*
- *A reconstruction algorithm $\textsc{Dec} : M_1 \times \cdots \times M_k \to Z$.*

We say that the protocol $\mathcal{P}$ is correct (with respect to f) if $\textsc{Dec}(\textsc{Enc}_1(x_1, r), \ldots, \textsc{Enc}_k(x_k, r)) = f(x_1, \ldots, x_k)$, for every input $(x_1, \ldots, x_k) \in X_1 \times \cdots \times X_k$ and every random input $r \in R$. The communication complexity of PSM protocol $\mathcal{P}$ is defined as $\sum_{i=1}^{k} \log |M_i|$. The randomness complexity of PSM protocol $\mathcal{P}$ is defined as $\log |R|$.

The security of a PSM protocol requires that the message distribution seen by the referee on input $x_1, \ldots, x_k$ can be generated by a simulator that has access only to $f(x_1, \ldots, x_k)$; that is, everything that can be learned from the PSM protocol can be learned from the output of f.

Definition 2.2 (PSM protocols – Security). *A PSM protocol $\mathcal{P}$ is secure with respect to f if there exists a randomized algorithm $\textsc{Sim}$ such that, for every input $(x_1, \ldots, x_k) \in X_1 \times \cdots \times X_k$, the distribution of messages $(\textsc{Enc}_1(x_1, r), \ldots, \textsc{Enc}_k(x_k, r))$ induced by uniformly choosing a common random string $r \in R$ and the distribution of the output of $\textsc{Sim}(f(x_1, \ldots, x_k))$ are identical.*

Ishai and Kushilevitz [14] have shown that every function that has a small modular or non-deterministic branching program can be computed by an efficient PSM protocol. We will use their result for a deterministic branching program.

Theorem 2.3 ([10, 14]). *Let $BP = (V, E, \phi, s, t)$ be a deterministic branching program of size $\alpha(k)$ computing a function $f : \{0,1\}^k \to \{0,1\}$. Then, there exist a PSM protocol for f with communication and randomness complexity $O(k \cdot \alpha(k)^2)$.*

Notation. Denote by $[N]$ the set $\{1, 2, \dots, N\}$. For a finite set S, denote choosing a random element i from S with a uniform distribution by $i \in_R S$; similarly, denote choosing a random subset T of S with a uniform distribution by $T \subseteq_R S$.

2.2 A PSM Protocol for the Index Function

In this section, we show a construction of [10] of a PSM protocol for the index function defined below.

Definition 2.4. *We represent a string $D \in \{0,1\}^{N^{k-1}}$ as a $(k-1)$-dimensional cube (array), that is, $D = (D_{x_2,\dots,x_k})_{x_2,\dots,x_k \in [N]}$, where each $D_{x_2,\dots,x_k}$ is a bit. For a $(k-1)$-dimensional cube D, and a position $(i_2, \dots, i_k)$ let $D_{i_2,\dots,i_k}$ denote the value in position $(i_2, \dots, i_k)$ in the cube D. We define a k-argument function $\mathrm{ind}_{N,k} : \{0,1\}^{N^{k-1}} \times [N] \times \cdots \times [N] \to \{0,1\}$, whose inputs are a $(k-1)$-dimensional cube and $k-1$ indices, by $\mathrm{ind}_{N,k}(D, x_2, \dots, x_k) = D_{x_2,\dots,x_k}$; that is, $\mathrm{ind}_{N,k}(D, x_2, \dots, x_k)$ returns the value of D in the position indexed by $x_2, \dots, x_k$. When N, k will be clear from the context, we will write* ind *instead of* $\mathrm{ind}_{N,k}$

For sake of completeness, we next present a PSM protocol for the index function, based on a PSM from [10]; for simplicity, we assume here that N is a power of 2 (and so values in $[N]$ can be represented by $\log N$-bit strings).

Claim 2.5 ([10]). *There is a k-party PSM protocol $\mathcal{P}_{\mathrm{ind}_{N,k}}$ computing the function $\mathrm{ind}_{N,k}$ with communication and randomness complexity is $O(kN^{k-1})$.*

Proof. In a non-secure protocol for $\mathrm{ind}_{N,k}$, party P_1 would send its input D and parties $P_2, \dots, P_k$ would send their inputs, $x_2, \dots, x_k$ respectively, to the referee who could easily compute $D_{x_2,\dots,x_k}$. However, in a secure protocol, the referee should not learn information on D and $x_2, \dots x_k$ except for $D_{x_2,\dots,x_k}$.

To hide the indices $x_2, \dots, x_k$ we permute D: we give $(k-1)$ random strings $r_2, \dots, r_k \in [N]$ to the parties as their common randomness. Now, party P_1 creates a new cube D' such that $D'_{x_2,\dots,x_k} = D_{x_2 \oplus r_2,\dots,x_k \oplus r_k}$ for every $x_2, \dots, x_k \in [N]$. Party P_1 sends to the referee D' and parties $P_2, \dots, P_k$ send to the referee $x_2 \oplus r_2, \dots, x_k \oplus r_k$ respectively. The referee computes $D'_{x_2 \oplus r_2,\dots,x_k \oplus r_k} = D_{(x_2 \oplus r_2) \oplus r_2,\dots,(x_k \oplus r_k) \oplus r_k} = D_{x_2,\dots,x_k}$ as required.

In the above protocol the referee does not learn information on $x_2, \dots x_k$; however, it learns information on the cube D because party P_1 sends D', which is a shift of the cube D. We fix this protocol by masking D' and revealing to the referee only the mask of position $x_2, \dots, x_k$. Specifically, we choose $(k-1)$ random strings $r^2, \dots, r^k \in \{0,1\}^{N^{k-1}}$; each string is viewed as a $(k-1)$-dimensional cube. Party P_1 computes $D'' = D' \oplus r^2 \oplus \cdots \oplus r^k$ and sends D'', which is now a random string. Each party P_j, for $j = 2, \dots, k$, sends $x_j \oplus r_j$ and also $r^j_{i_2,\dots,i_{j-1},x_j \oplus r_j,i_{j+1},\dots,i_k}$ for every $i_2, \dots, i_{j-1}, i_{j+1}, \dots, i_k \in [N]$ (the length of the message of P_j is $\log N + N^{k-2}$). The referee computes $D_{x_2,\dots,x_k}$ as $D''_{x_2 \oplus r_2,\dots,x_k \oplus r_k} \oplus (\bigoplus_{j=2}^{k} r^j_{x_2 \oplus r_2,\dots,x_k \oplus r_k})$. To see that the protocol is secure note that for each entry in the cube the referee gets at most $k-2$ masks (except for the entry $(x_2 \oplus r_2, \ldots, x_k \oplus r_k)$ for which the referee gets all $k-1$ masks). The communication complexity of this protocol is $O(N^{k-1} + k \cdot N^{k-2}) = O(k \cdot N^{k-1})$ and the randomness complexity is $O(k \cdot N^{k-1})$. ☐

Remark: The dominant contribution to the complexity of the above protocol comes from the size of the cube (i.e., N^{k-1}). We will sometimes need a natural extension of $\mathcal{P}_{\text{ind}}$, where the dimensions are not necessarily of the same size. It is not hard to see that the complexity of this variant remains proportional to the size of the cube.

2.3 A PSM Protocol for $S \oplus \{x\}$

For a set S and an element i, let $S \oplus \{i\}$ denote the set $S \setminus \{i\}$ if $i \in S$, and the set $S \cup \{i\}$ otherwise.

Definition 2.6. *Define the function* $\text{Sxor} : \{0,1\}^{N^k} \times [N] \times \cdots \times [N] \to \{0,1\}^{N^k}$ *as the function, whose inputs are a string of length N^k (interpreted as a set contained in $[N^k]$) and k elements from $[N]$, where* Sxor *outputs a string of length N^k (again, interpreted as a set contained in $[N^k]$) such that* $\text{Sxor}(S, x_1, \ldots, x_k) = S \oplus \{x_1 \circ x_2 \circ \cdots \circ x_k\}$ *(where $x_1 \circ x_2 \circ \cdots \circ x_k$ is the concatenation of the k strings, interpreted as an element of $[N^k]$).*

We construct a k-party PSM protocol for Sxor, where P_1 holds S and x_1 and $P_2, \ldots, P_k$ hold $x_2, \ldots, x_k$, respectively.[5]

Claim 2.7. *There exists a PSM protocol $\mathcal{P}_{\text{Sxor}}$ computing the function* Sxor *with communication and randomness complexity $O(k^3 \cdot N^k)$.*

Proof. Let $S' = \text{Sxor}(S, x_1, \ldots, x_k)$ and for every $i_1, \ldots, i_k \in [N]$ denote $S_{i_1,\ldots,i_k}$ and $S'_{i_1,\ldots,i_k}$ as the $i_1 \circ \cdots \circ i_k$th bits of the strings S and S' respectively.

To compute Sxor, for every bit of S' we execute a PSM protocol computing the bit as explained below. For each $(i_1, \ldots, i_k)$ such that $(i_1, \ldots, i_k) \neq (x_1, \ldots, x_k)$, it holds that $S'_{i_1,\ldots,i_k} = S_{i_1,\ldots,i_k}$, and $S'_{x_1,\ldots,x_k} = S_{x_1,\ldots,x_k} \oplus 1$. Let ℓ_j be 1 if $i_j = x_j$ and 0 otherwise. Notice that $S'_{i_1,\ldots,i_k} = S_{i_1,\ldots,i_k} \oplus (\ell_1 \wedge \ell_2 \wedge \ldots \wedge \ell_k)$. Thus, every bit of $S'_{i_1,\ldots,i_k}$ depends only on one bit of S and on $\ell_1, \ldots, \ell_k$, where party P_j can locally compute ℓ_j from x_j and i_j. Define the function $g : \{0,1\}^{k+1} \to \{0,1\}$ such that $g(s, \ell_1, \ldots, \ell_k) = s \oplus (\ell_1 \wedge \ell_2 \wedge \ldots \wedge \ell_k)$ for every $s, \ell_1, \ldots, \ell_k \in \{0,1\}$. We have shown that $S'_{i_1,\ldots,i_k}$ could be computed using N^k copies of a PSM for g.

Next, we show the existence of an efficient k-party PSM protocol $\mathcal{P}_g$ for g, where P_1 holds the inputs s and ℓ_1 and each party P_j, for $2 \leq j \leq k$, holds the input ℓ_j. There is a simple deterministic branching program of size $O(k)$ computing g, thus, by Theorem 2.3, we get that g has an efficient PSM protocol $\mathcal{P}_g$ with complexity $O(k^3)$.

[5] When we use this PSM protocol, all parties know S. We do not use this advantage as it cannot significantly improve the complexity of the protocol.

Protocol $\mathcal{P}_{\text{Sxor}}$ computing Sxor executes the protocol $\mathcal{P}_g$ for every bit of S', namely N^k times. The correctness and privacy of protocol $\mathcal{P}_{\text{Sxor}}$ follows immediately from the correctness and privacy of the PSM protocol $\mathcal{P}_g$. The complexity of $\mathcal{P}_{\text{Sxor}}$ is $O(k^3 N^k)$. □

3 A 3-Party PSM Protocol for an Arbitrary Function

In this section we show that every function $f : [N]^3 \to \{0,1\}$ has a PSM protocol with communication and randomness complexity of $O(N)$. Our construction is inspired by the cubes approach of [9]. We describe this approach in Section 3.1. Next, as a warm-up, we construct a 2-party PSM protocol using this approach in Section 3.2. We describe the 3-party PSM protocol in Section 3.3.

3.1 The Cube approach

We start with a high level description of the cube approach; specifically, for the case of 2-dimensional cubes, we present a PIR protocol with 4 servers from [9]. Recall that in a PIR protocol, a client holds an index x, each server holds a copy of database D, and the goal of the client is to retrieve D_x without disclosing information about x.

The starting point of the cube approach [9] (restricted here to 2 dimensions) is viewing the database D as a 2-dimensional cube containing N^2 bits, that is $D = (D_{i_1,i_2})_{i_1,i_2 \in [N]}$. Correspondingly, the index that the client wishes to retrieve is viewed as a pair $x = (x_1, x_2)$. The protocol starts by the client choosing a random subset for each dimension, i.e. $S_1, S_2 \subseteq_R [N]$. The client then creates 4 queries of the form (T_1, T_2) where each T_j is either S_j itself or $S_j \oplus \{x_j\}$; i.e. (S_1, S_2), $(S_1 \oplus \{x_1\}, S_2)$, $(S_1, S_2 \oplus \{x_2\})$, and $(S_1 \oplus \{x_1\}, S_2 \oplus \{x_2\})$; we denote these 4 queries by $q_{00}, q_{10}, q_{01}, q_{11}$, respectively. The client sends each query to a different server ($2 \cdot N$ bits to each server). A server, which gets a query (T_1, T_2), replies with a single bit which is the XOR of all bits of D in the sub-cube $T_1 \otimes T_2$, i.e. $\bigoplus_{i_1 \in T_1, i_2 \in T_2} D_{i_1,i_2}$. The observation made in [9] is that each element of the cube appears in an even number of those 4 sub-cubes except the entry $x = (x_1, x_2)$ that appears exactly once. Therefore, taking the XOR of the 4 answer bits, all elements of the cube are canceled except for the desired element in position x. Each server gets no information about x from its query. For example, in the query q_{01}, the sets S_1 and $S_2 \oplus \{x_2\}$ are uniformly distributed, independently of x_1, x_2.

The above approach can be generalized to any number of dimensions. Specifically, for 3 dimensions, the client chooses 3 sets $S_1, S_2, S_3 \subseteq [N]$ and generates 8 queries $q_{000}, \dots, q_{111}$.

3.2 A 2-Party PSM Protocol

Given a function $f : [N] \times [N] \to \{0,1\}$, we construct a 2-party PSM protocol $\mathcal{P}_2$ for f using the above approach. This PSM protocol is not as efficient as the

PSM protocol of [6]; we present it to introduce the ideas we use in our PSM protocol for $k > 2$ parties. The protocol is formally described in Fig. 2.

Next, we give an informal description of an insecure protocol, and then we fix it so it will be secure. We associate the function f with an N^2-bit database, viewed as a 2-dimensional cube (that is, the (x, y) entry in the database is $f(x, y)$). The common randomness of the two parties is viewed as two random subsets, one for each dimension, i.e. $S_1, S_2 \subseteq_R [N]$. In the PSM protocol, party P_1 holds x_1, party P_2 holds x_2, and the referee wishes to compute $f(x_1, x_2)$, i.e. the XOR of the answers to the same 4 queries mentioned above. This should be done without learning information about x_1, x_2 (besides what follows from $f(x_1, x_2)$). In the protocol, P_1 computes the answers, denoted a_{00}, a_{10}, to the queries q_{00}, q_{10} (using S_1, S_2 and its input x_1). For example, the answer to the query q_{10} is

$$a_{10} = \oplus_{i_1 \in S_1 \oplus \{x_1\}, i_2 \in S_2} f(i_1, i_2).$$

It then sends these answers to the referee. Similarly, P_2 computes the answer, denoted a_{01}, to the query q_{01} and sends it to the referee. Now, the referee has the answers to 3 of the 4 queries and the only query that remains unanswered is q_{11}. To compute the answer to q_{11}, party P_1 sends to the referee $S_1 \oplus \{x_1\}$ and P_2 sends $S_2 \oplus \{x_2\}$, i.e. the referee gets the query q_{11} and can compute the corresponding answer a_{11} (as it knows the function f). Now, the referee has the answers to all 4 queries and it can compute $f(x_1, x_2) = a_{00} \oplus a_{10} \oplus a_{01} \oplus a_{11}$. The correctness of this protocol follows immediately from the correctness of the cube approach.

This protocol is not secure because the referee learns the answers to all 4 queries and this could leak information about x_1 and x_2. In order to deal with this problem, we add one more random bit to the common randomness of the two parties, i.e., $b \in_R \{0, 1\}$. Party P_1 sends to the referee $a_{00} \oplus a_{10} \oplus b$ instead of a_{00}, a_{10}, and P_2 sends $a_{01} \oplus b$ instead of a_{01}. Now, the referee only gets random bits from the parties (from the randomness of S_1, S_2, b) and it learns only $f(x_1, x_2)$. The communication complexity and randomness complexity of this protocol are $O(N)$.

More formally, to argue that the PSM protocol $\mathcal{P}_2$ described in Fig. 2 is secure, we construct a simulator whose input is $f(x_1, x_2)$ and whose output is two messages distributed identically to the messages in $\mathcal{P}_2$. The simulator first chooses with uniform distribution two sets $T_1, T_2 \subseteq_R [N]$ and a random bit $c \in \{0, 1\}$. It then computes the answer to the query q_{11}, namely, $a_{11} = \oplus_{i_1 \in T_1, i_2 \in T_2} f(i_1, i_2)$. Finally, it outputs $(c, T_1), (c \oplus f(x_1, x_2) \oplus a_{11}, T_2)$. Note that in the messages of $\mathcal{P}_2$ the sets $S_1 \oplus \{x_1\}$ and $S_2 \oplus \{x_2\}$ are distributed independently with uniform distribution. Furthermore, m_1 is uniformly distributed given the two sets, since we mask m_1 with a random bit b. Finally, by the correctness of $\mathcal{P}_2$, given $m_1, S_1 \oplus \{x_1\}, S_2 \oplus \{x_2\}$, and $f(x_1, x_2)$, the value of m_2 is fully determined (as the sets determine a_{11}). Thus, the simulator's output is distributed as the messages generated in protocol $\mathcal{P}_2$ on inputs x_1, x_2.

Protocol $\mathcal{P}_2$

Common randomness: Both parties share uniform random strings:

- $S_1, S_2 \subseteq_R [N]$ and $b \in_R \{0,1\}$.

The protocol:

1. P_1, holding x_1, computes the answers a_{00}, a_{10} to the queries $q_{00} = (S_1, S_2)$ and $q_{10} = (S_1 \oplus \{x_1\}, S_2)$ respectively. It then computes $q^1_{11} = S_1 \oplus \{x_1\}$. Finally, it sends $m_1 = a_{00} \oplus a_{10} \oplus b$ and q^1_{11} to the referee.
2. P_2, holding x_2, computes the answer a_{01} to the query $q_{01} = (S_1, S_2 \oplus \{x_2\})$. It then computes $q^2_{11} = S_2 \oplus \{x_2\}$. Finally, it sends $m_2 = a_{01} \oplus b$ and q^2_{11} to the referee.
3. The referee computes the answer a_{11} to the query $q_{11} = (q^1_{11}, q^2_{11}) = (S_1 \oplus \{x_1\}, S_2 \oplus \{x_2\})$. The referee computes $f(x_1, x_2) = m_1 \oplus m_2 \oplus a_{11}$.

Fig. 2. A 2-party PSM protocol $\mathcal{P}_2$ for a function $f : [N] \times [N] \to \{0,1\}$.

3.3 A 3-Party PSM Protocol

In this section, we show how to construct a PSM protocol $\mathcal{P}_3$ for any function $f : [N]^3 \to \{0,1\}$ with communication and randomness complexity $O(N)$. As in the 2-party case above, our construction is inspired by the cube approach [9], as described in Section 3.1, using 3-dimensional cubes.

Again, we first give an informal description of an insecure protocol, and then we fix it so it will be secure. The protocol $\mathcal{P}_3$ is formally described in Fig. 3. We associate f with an N^3-bit database that is viewed as a 3-dimensional cube. The common randomness of the 3 parties consists of 3 random subsets, one for each dimension, i.e., $S_1, S_2, S_3 \subseteq_R [N]$. The referee wishes to compute $f(x_1, x_2, x_3)$, i.e. the XOR of the answers to 8 queries. Each query is of the form (T_1, T_2, T_3) where each T_j is either S_j or $S_j \oplus \{x_j\}$. Party P_1 computes the answers a_{000}, a_{100} to the queries q_{000}, q_{100} respectively, and sends these answers to the referee. Similarly, party P_2 (resp. P_3) computes the answer a_{010} (resp. a_{001}) to the query q_{010} (resp. q_{001}) and sends the answer to the referee. There is no party that knows the values of two inputs from $\{x_1, x_2, x_3\}$ and therefore, no party can answer queries of weight 2; e.g. q_{110}. However, using an idea of [9], party P_1 can provide the answers to the queries $(S_1 \oplus \{x_1\}, S_2 \oplus \{\ell\}, S_3)$ for all possible values of $\ell \in [N]$. This is a list of length N in which the entry corresponding to $\ell = x_2$ is the desired answer for the query q_{110}. Party P_1 computes the N-bit list and represents it as a 1-dimension cube; party P_2 holds x_2, which is the position of the answer a_{110} in the list. Now, P_1 and P_2 execute the (2-party) PSM protocol $\mathcal{P}_{\text{ind}}$ for the index function, described in Section 2.2, which enables the referee to compute the answer to the query q_{110} without leaking any additional information. Similarly, P_1 and P_3 (resp. P_2 and P_3) execute the PSM protocol $\mathcal{P}_{\text{ind}}$ that enables the referee to compute the answer to the query q_{101} (resp.

Protocol $\mathcal{P}_3$

Common randomness: The three parties share uniform random strings:

- $S_1, S_2, S_3 \subseteq_R [N]$.
- $r_{110}, r_{101}, r_{011} \in_R \{0,1\}^{O(N)}$ required for the PSM protocol $\mathcal{P}_{\text{ind}}$ with a list of length N.
- $b_{100}, b_{010}, b_{001}, b_{110}, b_{101}, b_{011} \in_R \{0,1\}$, $b_{000} = b_{100} \oplus b_{010} \oplus b_{001} \oplus b_{110} \oplus b_{101} \oplus b_{011}$.

The protocol:

1. Party P_1:
   - Computes the answers a_{000}, a_{100} to the queries $q_{000} = (S_1, S_2, S_3)$ and $q_{100} = (S_1 \oplus \{x_1\}, S_2, S_3)$, respectively.
   - Computes the answers to all queries $(S_1 \oplus \{x_1\}, S_2 \oplus \{\ell\}, S_3)$ for all possible values of $\ell \in [N]$ and represents the answers as a 1-dimension database $(a^1_{110}, \dots, a^N_{110})$. Using $D^{110} = (a^1_{110} \oplus b_{110}, \dots, a^N_{110} \oplus b_{110})$ and common randomness r_{110}, party P_1 computes m^1_{110} – the message of the first party in the 2-party PSM protocol $\mathcal{P}_{\text{ind}}$.
   - Computes the answers to all queries $(S_1 \oplus \{x_1\}, S_2, S_3 \oplus \{\ell\})$ for all possible values of $\ell \in [N]$, and represents the answers as a 1-dimension database $(a^1_{101}, \dots, a^N_{101})$. Using $D^{101} = (a^1_{101} \oplus b_{101}, \dots, a^N_{101} \oplus b_{101})$ and common randomness r_{101}, computes m^1_{101} – the message of the first party in the 2-party PSM protocol $\mathcal{P}_{\text{ind}}$.
   - Sends $m_{000} = a_{000} \oplus b_{000}$, $m_{100} = a_{100} \oplus b_{100}$, m^1_{110}, m^1_{101}, and $S_1 \oplus \{x_1\}$ to the referee.
2. Party P_2:
   - Computes the answer a_{010} to the query $q_{010} = (S_1, S_2 \oplus \{x_2\}, S_3)$.
   - Using x_2 and common randomness r_{110}, computes m^2_{110} – the message of the second party in the 2-party PSM protocol $\mathcal{P}_{\text{ind}}$.
   - Computes the answers to all queries $(S_1, S_2 \oplus \{x_2\}, S_3 \oplus \{\ell\})$ for all possible values of $\ell \in [N]$ and represents the answers as a 1-dimension database $(a^1_{011}, \dots, a^N_{011})$. Using $D^{011} = (a^1_{011} \oplus b_{011}, \dots, a^N_{011} \oplus b_{011})$ and common randomness r_{011}, computes m^2_{011} – the message of the first party in the PSM protocol $\mathcal{P}_{\text{ind}}$.
   - Sends $m_{010} = a_{010} \oplus b_{010}$, m^2_{110}, m^2_{011}, and $S_2 \oplus \{x_2\}$ to the referee.
3. Party P_3:
   - Computes the answer a_{001} to the query $q_{001} = (S_1, S_2, S_3 \oplus \{x_3\})$.
   - Using x_3 and common randomness r_{101}, computes m^3_{101} – the message of the second party in the 2-party PSM protocol $\mathcal{P}_{\text{ind}}$.
   - Using x_3 and common randomness r_{011} it computes m^2_{011} – the message of the second party in the 2-party PSM protocol $\mathcal{P}_{\text{ind}}$.
   - Sends $m_{001} = a_{001} \oplus b_{001}$, m^3_{101}, m^3_{011}, and $S_3 \oplus \{x_3\}$ to the referee.
4. The referee:
   - Computes the answer a_{111} to the query $q_{111} = (S_1 \oplus \{x_1\}, S_2 \oplus \{x_2\}, S_3 \oplus \{x_3\})$ (using the sets received from P_1, P_2, P_3 and the truth table of f).
   - Using the PSM messages m^1_{110}, m^2_{110} the referee computes $D^{110}_{x_2}$. Using the PSM messages m^1_{101}, m^3_{101} the referee computes $D^{101}_{x_3}$. Using the PSM messages m^2_{011}, m^3_{011} the referee computes $D^{011}_{x_3}$.
   - Output $f(x) = m_{000} \oplus m_{100} \oplus m_{010} \oplus m_{001} \oplus D^{110}_{x_2} \oplus D^{101}_{x_3} \oplus D^{011}_{x_3} \oplus a_{111}$.

Fig. 3. A 3-party PSM protocol $\mathcal{P}_3$ for a function $f : [N]^3 \to \{0,1\}$.

q_{011}). Finally, party P_1 sends to the referee $S_1 \oplus \{x_1\}$, party P_2 sends $S_2 \oplus \{x_2\}$, and party P_3 sends $S_3 \oplus \{x_3\}$. The referee gets the query q_{111} and computes the answer a_{111}. It now has the answers to all 8 queries and it can compute $f(x_1, x_2, x_3)$ as the XOR of these 8 answers.

As in the 2-party case, this protocol is not secure because the referee learns the answers to all 8 queries, which could leak information about x_1, x_2, and x_3. To deal with that, we mask these answers so that the referee gets random bits from the parties whose sum is $f(x_1, x_2, x_3)$ (the details of applying these masks appear in Fig. 3).

Theorem 3.1. *Let $f : [N]^3 \to \{0, 1\}$ be a function. The protocol $\mathcal{P}_3$ is a secure PSM protocol for f with communication and randomness complexity $O(N)$.*

Proof. First, we argue that the protocol is correct. The output of the referee is

$$\begin{aligned}
& m_{000} \oplus m_{100} \oplus m_{010} \oplus m_{001} \oplus D_{x_2}^{110} \oplus D_{x_3}^{101} \oplus D_{x_3}^{011} \oplus a_{111} \\
&= (a_{000} \oplus b_{000}) \oplus (a_{100} \oplus b_{100}) \oplus (a_{010} \oplus b_{010}) \oplus (a_{001} \oplus b_{001}) \oplus (a_{110}^{x_2} \oplus b_{110}) \\
&\quad \oplus (a_{101}^{x_3} \oplus b_{101}) \oplus (a_{011}^{x_3} \oplus b_{011}) \oplus a_{111} \\
&= a_{000} \oplus a_{100} \oplus a_{010} \oplus a_{001} \oplus a_{110}^{x_2} \oplus a_{101}^{x_3} \oplus a_{011}^{x_3} \oplus a_{111} \qquad (1) \\
&= f(x_1, x_2, x_3), \qquad (2)
\end{aligned}$$

where the equality in (1) follows from the fact that the exclusive or of the b's is zero, and the equality in (2) follows from the correctness of the cube approach.

To argue that protocol $\mathcal{P}_3$ is secure, we construct a simulator whose input is $f(x_1, x_2, x_3)$ and whose output is three messages distributed as the messages in $\mathcal{P}_3$. The simulator on input $f(x_1, x_2, x_3)$ does the following:

1. Chooses three random sets $T_1, T_2, T_3 \subseteq_R [N]$ and 6 random bits $c_{100}, c_{010}, c_{001}, c_{110}, c_{101}, c_{011} \in_R \{0, 1\}$.
2. Computes the answer to the query q_{111}, namely,
$$a_{111} = \oplus_{i_1 \in T_1, i_2 \in T_2, i_3 \in T_3} f(i_1, i_2, i_3).$$
3. Computes $c_{000} = f(x_1, x_2, x_3) \oplus a_{111} \oplus c_{100} \oplus c_{010} \oplus c_{001} \oplus c_{110} \oplus c_{101} \oplus c_{011}$.
4. Invokes the simulator $\textsc{Sim}_{\text{ind}}$ of protocol $\mathcal{P}_{\text{ind}}$ 3 times:
   – $(m_{110}^1, m_{110}^2) \leftarrow \textsc{Sim}_{\text{ind}}(c_{110})$,
   – $(m_{101}^1, m_{101}^3) \leftarrow \textsc{Sim}_{\text{ind}}(c_{101})$,
   – $(m_{011}^2, m_{011}^3) \leftarrow \textsc{Sim}_{\text{ind}}(c_{011})$.
5. Outputs
$$(c_{000}, c_{100}, m_{110}^1, m_{101}^1, T_1), (c_{010}, m_{110}^2, m_{011}^2, T_2), (c_{001}, m_{101}^3, m_{011}^3, T_3).$$

Note that in the messages of protocol $\mathcal{P}_3$ the sets $S_1 \oplus \{x_1\}, S_2 \oplus \{x_2\}$, and $S_3 \oplus \{x_3\}$ are distributed independently with uniform distribution. Furthermore, m_{100}, m_{010}, m_{001}, $D_{x_2}^{110}, D_{x_3}^{101}, D_{x_3}^{011}$ are uniformly distributed given these 3 sets, since we mask them using independent random bits. Since $D_{x_2}^{110}$ and c_{110} are both random bits, the output of $\textsc{Sim}_{\text{ind}}(c_{110})$ is distributed as the messages of

$\mathcal{P}_{\text{ind}}(D^{110}, x_2)$. The same holds for the other two invocations of $\textsc{Sim}_{\text{ind}}$. Finally, given the sets $S_1 \oplus \{x_1\}, S_2 \oplus \{x_2\}, S_3 \oplus \{x_3\}$, the bits $m_{100}, m_{010}, m_{001}, D^{110}_{x_2}, D^{101}_{x_3}, D^{011}_{x_3}, a_{111}$, and $f(x_1, x_2, x_3)$, the value of m_{000} is fully determined (by the correctness of $\mathcal{P}_3$ and by the fact that the above sets determine a_{111}). Thus, the simulator's output is distributed as the messages generated in protocol $\mathcal{P}_3$ on inputs x_1, x_2, x_3.

We next analyze the complexity of $\mathcal{P}_3$. The communication and randomness complexity of each invocation of $\mathcal{P}_{\text{ind}}$ is linear in the length of the list, i.e., it is $O(N)$. In addition, parties P_1, P_2, P_3 send the subsets $S_1 \oplus \{x_1\}, S_2 \oplus \{x_2\}$, $S_3 \oplus \{x_3\}$ respectively, which are of size N, and $O(1)$ bits each for the answers of the queries $q_{000}, q_{100}, q_{010}, q_{001}$. Therefore, the communication complexity and randomness complexity of our PSM protocol $\mathcal{P}_3$ is $O(N)$. $\square$

4 A k-Party PSM Protocol for an Arbitrary Function and Some Extensions

In this section, we show how to construct a k-party PSM protocol $\mathcal{P}_k$ for a function $f : [N]^k \to [N^k]$ with communication and randomness complexity $O(k^3 \cdot N^{k/2})$. The above complexity is achieved even when the output is of length $k \log N$ (as the input length is $k \log N$, we can assume, without loss of generality, that the output length is at most $k \log N$). In general, when the output of f is an L-bit string, one can execute a PSM protocol for every bit of the output, and the complexity of the PSM protocol for f is L times the complexity of a PSM protocol for a Boolean function. In protocol $\mathcal{P}_k$, we do not pay any penalty for long outputs. We also present better protocols for 4 and 5 parties.

We first describe our construction for an even k. Our construction is inspired by our 2-party PSM protocol presented in Section 3.2. The protocol is formally described in Fig. 4. Next, we give an informal description of the protocol for the case that the range of f is Boolean. We associate f with an N^k-bit database that is viewed as a 2-dimensional cube, where each dimension of the cube is of size $N^{k/2}$. Correspondingly, each input $x = (x_1, \ldots, x_k) \in [N]^k$ is viewed as a pair $(y_1, y_2) \in [N^{k/2}]^2$, where $y_1 = (x_1, \ldots, x_{k/2})$ and $y_2 = (x_{k/2+1}, \ldots, x_k)$. The common randomness of the k parties contains two random subsets, one for each dimension, i.e. $S_1, S_2 \subseteq_R [N^{k/2}]$. The referee wishes to compute $f(x_1, \ldots, x_k)$, i.e., the XOR of answers to the same 4 queries of the cube approach described in Section 3.1. Party P_1 can easily compute the answer to the query (S_1, S_2) (i.e., q_{00}) and send the answer to the referee. However, as the inputs y_1, y_2 are distributed among the parties, there are two problems we have to address: (1) how to answer the queries $(S_1 \oplus \{y_1\}, S_2), (S_1, S_2 \oplus \{y_2\})$, and (2) how to send $S_1 \oplus \{y_1\}$ and $S_2 \oplus \{y_2\}$ to the referee.

We first address the first problem. The answer to query $(S_1 \oplus \{y_1\}, S_2)$ depends on y_1, i.e., on inputs $(x_1, \ldots, x_{k/2})$, and there is no party that knows all these inputs. We solve this problem in a similar way to what is done in [9] and in our protocol $\mathcal{P}_3$, that is, by using the PSM protocol $\mathcal{P}_{\text{ind}}$. Although party P_1 does not know the exact value of $y_1 = (x_1, \ldots, x_{k/2})$, it can compute the answers to all queries $(S_1 \oplus \{\ell\}, S_2)$ for $\ell = (x_1, i_2, \ldots, i_{k/2})$ for all possible values $i_2, \ldots, i_{k/2} \in [N]^{k/2-1}$. This is a list of length $N^{k/2-1}$ in which the entry corresponding to $\ell = y_1$ is the desired answer for the query q_{10}. We view this answer as a $(k/2-1)$-dimensional cube such that the answer corresponding to the values $(x_1, i_2, i_3, \ldots, i_{k/2})$ is in position $(i_2, i_3, \ldots, i_{k/2})$. Specifically, the answer to the query $(S_1 \oplus \{y_1\}, S_2)$ is in position $(x_2, \ldots, x_{k/2})$ in the cube. Parties $P_1, \ldots, P_{k/2}$ use the PSM protocol $\mathcal{P}_{\text{ind}}$ for the index function described in Section 2.2, from which the referee learns the answer to the query $(S_1 \oplus \{y_1\}, S_2)$ and nothing else. Similarly, P_k can compute an $N^{k/2-1}$-bit cube, corresponding to all choices of $i_{k/2+1}, \ldots, i_{k-1}$, such that the answer to the query $(S_1, S_2 \oplus \{y_2\})$ is in position $x_{k/2+1}, \ldots, x_{k-1}$ in this cube. Parties $P_k, P_{k/2+1} \ldots, P_{k-1}$ use the PSM protocol $\mathcal{P}_{\text{ind}}$, from which the referee learns the answer to the query $(S_1, S_2 \oplus \{y_2\})$ and nothing else.

The only query that remained unanswered is the query q_{11}. Parties $P_1, \ldots, P_{k/2}$ execute the PSM protocol $\mathcal{P}_{\text{Sxor}}$ described in Section 2.3 that enables the referee to compute $S_1 \oplus \{y_1\}$ without learning any information about y_1. Similarly, parties $P_{k/2+1}, \ldots, P_k$ execute the PSM protocol $\mathcal{P}_{\text{Sxor}}$ that enables the referee to compute $S_2 \oplus \{y_2\}$. The referee learns the query $q_{11} = (S_1 \oplus \{y_1\}, S_2 \oplus \{y_2\})$ and computes the corresponding answer a_{11}. Now, the referee has the answers to all 4 queries and it can XOR them to compute $f(x_1, \ldots, x_k)$.

The main contributions to the communication complexity of the above protocol is the invocations of the PSM protocols $\mathcal{P}_{\text{ind}}$ and $\mathcal{P}_{\text{Sxor}}$. We invoke $\mathcal{P}_{\text{ind}}$ with $k/2$ parties and a database containing $N^{k/2-1}$ bits, thus, the complexity of this protocol is $O(kN^{k/2-1})$. We invoke $\mathcal{P}_{\text{Sxor}}$ with $k/2$ parties and a set contained in $[N^{k/2}]$, thus, its complexity is $O(k^3 \cdot N^{k/2})$.

Note that the complexity of invoking $\mathcal{P}_{\text{Sxor}}$ dominates the complexity of invoking $\mathcal{P}_{\text{ind}}$. We capitalize on this gap and construct a PSM protocol for any function with output range $[N^k]$. Again, we represent $f : [N]^k \to [N^k]$ as a two dimensional cube, where the size of each dimension is $N^{k/2}$, however now every entry in the cube is from $[N^k]$. The protocol proceeds as above, where the only difference is that we invoke $\mathcal{P}_{\text{ind}}$ with a database containing $N^{k/2-1}$ elements from $[N^k]$.[6] Thus, the complexity of this protocol is $O(kN^{k/2-1} \cdot k \log N)$. The resulting PSM protocol has complexity $O(k^3 \cdot N^{k/2})$.

Next, we give an informal description of protocol $\mathcal{P}_k$ for an odd k, i.e., $k = 2t+1$ for some t. Here, we partition the input x_{t+1} of party P_{t+1} to two parts, i.e. $x_{t+1} = (x^1_{t+1}, x^2_{t+1})$ such that $x^1_{t+1}, x^2_{t+1} \in [N^{1/2}]$. Again, we associate f with an N^k-bit database that is viewed as a 2-dimensional cube (i.e., the size of each dimension is $N^{k/2}$). Correspondingly, each input $x = (x_1, \ldots, x_k) \in [N]^k$ is viewed as a pair $(y_1, y_2) \in [N^{k/2}]^2$, where $y_1 = (x_1, \ldots, x_t, x^1_{t+1})$ and $y_2 = (x^2_{t+1}, x_{t+2}, \ldots, x_k)$. The referee needs the answers to the same 4 queries in order to compute $f(x_1, \ldots, x_k)$. The rest of the protocol is similar to the protocol for an even k, just that in this case, party P_{t+1} participates in the PSM protocols for both queries q_{10}, q_{01}, each time only with half of its input, as well as in both

[6] Protocol $\mathcal{P}_{\text{ind}}$ (described in Section 2.3) can deal with L-bit entries and its complexity, for a list of N entries, is $O(L \cdot N)$.

Protocol $\mathcal{P}_k$

Common randomness: The k parties share uniform random strings:

- $S_1, S_2 \subseteq_R [N^{k/2}]$.
- $r_{10}, r_{01} \in_R \{0,1\}^{O(kN^{k/2-1}\log N)}$ required for the PSM protocol $\mathcal{P}_{\text{ind}}$ with a list of length $[N]^{k/2-1}$, where each element is from $[N^k]$.
- $\rho_{10}, \rho_{01} \in_R \{0,1\}^{O(N^{k/2})}$ required for the $k/2$-party PSM protocol $\mathcal{P}_{\text{Sxor}}$ with a set $S \subseteq [N^{k/2}]$.
- $b_{10}, b_{01} \in_R \{0,1\}^{k\log N}$, $b_{00} = b_{10} \oplus b_{01}$.

The protocol:

1. Party P_1:
   - Compute the answer a_{00} to the query $q_{00} = (S_1, S_2)$.
   - Compute the answers to all queries $(S_1 \oplus \{\ell\}, S_2)$ where $\ell = x_1 \circ i_2 \circ \ldots \circ i_{k/2}$, for all possible values $i_2, \ldots, i_{k/2} \in [N]$, and represent the answers as a $(k/2-1)$-dimensional cube $(a_{10}^{i_2,\ldots,i_{k/2}})_{i_2,\ldots,i_{k/2}\in[N]}$. Using $D^{10} = (a_{10}^{i_2,\ldots,i_{k/2}} \oplus b_{10})_{i_2,\ldots,i_{k/2}\in[N]}$ and common randomness r_{10}, compute m_{10}^1 – the message of the first party in $\mathcal{P}_{\text{ind}}$.
   - Using x_1, S_1 and common randomness ρ_{10}, compute s_{10}^1 – the message of the first party in $\mathcal{P}_{\text{Sxor}}$.
   - Send $m_{00} = a_{00} \oplus b_{00}$, m_{10}^1, and s_{10}^1 to the referee.
2. Party P_j, where $j \in \{2, \ldots, k/2\}$:
   - Using x_j and common randomness r_{10}, compute m_{10}^j – the message of the jth party in $\mathcal{P}_{\text{ind}}$.
   - Using x_j and common randomness ρ_{10}, compute s_{10}^j – the message of the jth party in $\mathcal{P}_{\text{Sxor}}$.
   - Send m_{10}^j and s_{10}^j to the referee.
3. Party P_k:
   - Compute the answers to all queries $(S_1, S_2 \oplus \{\ell\})$ where $\ell = i_{k/2+1} \circ \ldots \circ i_{k-1} \circ x_k$, for all possible values $i_{k/2-1}, \ldots, i_{k-1} \in [N]$, and represent the answers as a $(k/2-1)$-dimensional cube $(a_{01}^{i_{k/2+1},\ldots,i_{k-1}})_{i_{k/2+1},\ldots,i_{k-1}\in[N]}$. Using $D^{01} = (a_{01}^{i_{k/2+1},\ldots,i_{k-1}} \oplus b_{01})_{i_{k/2+1},\ldots,i_{k-1}\in[N]}$ and common randomness r_{01}, compute m_{01}^k – the message of the first party in $\mathcal{P}_{\text{ind}}$.
   - Using x_k, S_2 and common randomness ρ_{01}, compute s_{01}^k – the message of the first party in $\mathcal{P}_{\text{Sxor}}$.
   - Send m_{01}^k and s_{01}^k to the referee.
4. Party P_j, where $j \in \{k/2+1, \ldots, k-1\}$:
   - Using x_j and common randomness r_{01}, compute m_{01}^j – the message of the $(j-k/2+1)$th party in $\mathcal{P}_{\text{ind}}$.
   - Using x_j and common randomness ρ_{01}, compute s_{01}^j – the message of the $(j-k/2+1)$th party in $\mathcal{P}_{\text{Sxor}}$.
   - Send m_{01}^j and s_{01}^j to the referee.
5. The referee:
   - Using the messages $s_{10}^1, \ldots, s_{10}^{k/2}$ of $\mathcal{P}_{\text{Sxor}}$, compute $q_{11}^1 = S_1 \oplus \{y_1\}$. Using the messages $s_{01}^k, s_{01}^{k/2+1}, \ldots, s_{01}^{k-1}$ of $\mathcal{P}_{\text{Sxor}}$, compute $q_{11}^2 = S_2 \oplus \{y_2\}$. Compute the answer a_{11} to the query $q_{11} = (q_{11}^1, q_{11}^2) = (S_1 \oplus \{y_1\}, S_2 \oplus \{y_2\})$.
   - Using the messages $m_{10}^1, \ldots, m_{10}^{k/2}$ of $\mathcal{P}_{\text{ind}}$, compute $D^{10}_{x_2,\ldots,x_{k/2}}$. Using the messages $m_{01}^k, m_{01}^{k/2+1}, \ldots, m_{01}^{k-1}$ of $\mathcal{P}_{\text{ind}}$, compute $D^{01}_{x_{k/2+1},\ldots,x_{k-1}}$.
   - Output $f(x) = m_{00} \oplus D^{10}_{x_2,\ldots,x_{k/2}} \oplus D^{01}_{x_{k/2+1},\ldots,x_{k-1}} \oplus a_{11}$.

Fig. 4. A k-party PSM protocol $\mathcal{P}_k$ for a function $f : [N]^k \to [N^k]$ for an *even* k.

PSM protocols for $S_1 \oplus \{y_1\}$ and for $S_2 \oplus \{y_2\}$. The communication complexity and randomness complexity of this protocol are $O(k^3 N^{k/2})$.

Theorem 4.1. *Let $f : [N]^k \to [N^k]$ be a function. The protocol $\mathcal{P}_k$ is a secure PSM protocol for f with communication and randomness complexity $O(k^3 N^{k/2})$.*

Proof. Correctness follows from the cube approach, where $f(x_1, \dots, x_k)$ is the XOR of the answers for the 4 queries $q_{00}, q_{10}, q_{01}, q_{11}$. The referee computes these 4 answers (each answer is a string in $[N^k]$); however, the first 3 answers are masked. Nevertheless, the XOR of the 3 masks b_{00}, b_{10}, b_{01} is zero, so when the referee computes the XOR of the 4 masked answers, the masks are canceled and the referee gets the correct answer.

To argue that the PSM protocol $\mathcal{P}_k$ is secure, we construct a simulator whose input is $f(x_1, \dots, x_k)$ and whose output is k messages distributed as the messages in $\mathcal{P}_k$. To simplify the indices, we only construct a simulator for an even k; however, the simulator (with minor changes) remains valid also for an odd k. The simulator on input $f(x_1, \dots, x_k)$ does the following:

1. Chooses two random sets $T_1, T_2 \subseteq_R [N^{k/2}]$ and two random strings $c_{10}, c_{01} \in_R \{0,1\}^{k \log N}$.
2. Computes the answer to the query q_{11}, namely, $a_{11} = \oplus_{i_1 \in T_1, i_2 \in T_2} f(i_1, i_2)$ (where i_1, i_2 are considered as $k/2$ elements from $[N]$).
3. Computes $c_{00} = f(x_1, \dots, x_k) \oplus a_{11} \oplus c_{10} \oplus c_{01}$.
4. Invokes the simulator $\textsc{Sim}_{\text{ind}}$ of protocol $\mathcal{P}_{\text{ind}}$ twice:
   - $(m_{10}^1, \dots, m_{10}^{k/2}) \leftarrow \textsc{Sim}_{\text{ind}}(c_{10})$,
   - $(m_{01}^k, m_{01}^{k/2+1}, \dots, m_{01}^{k-1}) \leftarrow \textsc{Sim}_{\text{ind}}(c_{01})$.
5. Invokes the simulator $\textsc{Sim}_{\text{Sxor}}$ of protocol $\mathcal{P}_{\text{Sxor}}$ twice:
   - $(s_{10}^1, \dots, s_{10}^{k/2}) \leftarrow \textsc{Sim}_{\text{Sxor}}(T_1)$,
   - $(s_{01}^k, s_{01}^{k/2+1}, \dots, s_{01}^{k-1}) \leftarrow \textsc{Sim}_{\text{Sxor}}(T_2)$.
6. Outputs

$$(c_{00}, m_{10}^1, s_{10}^1), (m_{10}^2, s_{10}^2), \dots, (m_{10}^{k/2}, s_{10}^{k/2}), (m_{01}^{k/2+1}, s_{01}^{k/2+1}), \dots, (m_{01}^k, s_{01}^k).$$

Note that, in the messages of $\mathcal{P}_k$, the sets $S_1 \oplus \{y_1\}, S_2 \oplus \{y_2\}$ are distributed independently with uniform distribution. Furthermore, $D^{10}_{x_2,\dots,x_{k/2}}, D^{10}_{x_{k/2+1},\dots,x_{k-1}}$ are uniformly distributed given these 2 sets, since we mask them using independent random strings. Since $D^{10}_{x_2,\dots,x_{k/2}}$ and c_{10} are both random strings, the output of $\textsc{Sim}_{\text{ind}}(c_{10})$ is distributed as the messages of $\mathcal{P}_{\text{ind}}(D^{10}, x_2, \dots, x_{k/2})$. Since $S_1 \oplus \{y_1\}$ and T_1 are both random sets, the output of $\textsc{Sim}_{\text{Sxor}}(T_1)$ is distributed as the messages in $\mathcal{P}_{\text{Sxor}}((S_1, x_1), x_2, \dots, x_{k/2})$. The same holds for the other invocation of $\textsc{Sim}_{\text{ind}}$ and $\textsc{Sim}_{\text{Sxor}}$. Finally, given the sets $S_1 \oplus \{y_1\}, S_2 \oplus \{y_2\}$, the strings $m_{10}, m_{01}, D^{10}_{x_2,\dots,x_{k/2}}, D^{01}_{x_{k/2+1},\dots,x_{k-1}}$, and $f(x_1, \dots, x_k)$, the value of m_{00} is fully determined (by the correctness of $\mathcal{P}_k$ and by the fact that the above sets determine a_{11}). Thus, the simulator's output is distributed in the same way as the messages generated in protocol $\mathcal{P}_k$ on inputs $x_1, \dots, x_k$.

The communication and randomness complexity of each invocation of the PSM protocol $\mathcal{P}_{\text{ind}}$ are $O(k \cdot N^{k/2-1} \cdot k \log N)$. The communication and randomness complexity of each invocation of the PSM protocol $\mathcal{P}_{\text{Sxor}}$ is $O(k^3 \cdot N^{k/2})$. Party P_1 also sends a string of length $k \log N$ to the referee (that is, m_{00}). Therefore, the communication complexity and randomness complexity of the PSM protocol $\mathcal{P}_k$ are $O(k^3 \cdot N^{k/2})$. □

4.1 PSM Protocols for 4 and 5 Parties

We next show how to use the ideas of our previous protocols to construct more efficient k-party PSM protocols, for $k = 4, 5$.

Theorem 4.2.

– *Let $f : [N]^4 \to \{0,1\}$ be a function. There is a secure 4-party PSM protocol $\mathcal{P}_4$ for f with communication and randomness complexity $O(N^{5/3})$.*
– *Let $f : [N]^5 \to \{0,1\}$ be a function. There is a secure 5-party PSM protocol $\mathcal{P}_5$ for f with communication and randomness complexity $O(N^{7/3})$.*

Proof Sketch. The protocols $\mathcal{P}_4$ and $\mathcal{P}_5$ are similar to protocol $\mathcal{P}_k$, except that we view f as a 3-dimensional cube. Specifically, in $\mathcal{P}_4$ the size of each dimension of the cube is $N^{4/3}$. We partition the inputs x_2, x_3 as follows: $x_2 = (x_2^1, x_2^2)$ and $x_3 = (x_3^2, x_3^3)$ such that $x_2^1, x_3^3 \in [N^{1/3}]$ and $x_2^2, x_3^2 \in [N^{2/3}]$. We view each input (x_1, x_2, x_3, x_4) as a 3-tuple $(y_1, y_2, y_3) \in [N^{4/3}]^3$, where $y_1 = (x_1, x_2^1), y_2 = (x_2^2, x_3^2)$, and $y_3 = (x_3^3, x_4)$. The common randomness of the parties contains 3 random sets $S_1, S_2, S_3 \subseteq_R [N^{4/3}]$, random strings for protocols $\mathcal{P}_{\text{ind}}$ and $\mathcal{P}_{\text{Sxor}}$, and random masks. The referee should get the answers of 8 queries $q_{000}, \dots, q_{111}$. The (masked) answer to q_{000} is computed by P_1 and sent to the referee. The answers to the queries of weight 2 and 3 are computed using protocol $\mathcal{P}_{\text{ind}}$, where the more expensive queries are queries of weight 2. As an example, we explain how to answer q_{110}. The answer to this query requires knowing $(y_1, y_2) \in [N^{4/3}]^2$. As $y_1 = (x_1, x_2^1)$ and party P_1 has x_1, party P_1 can prepare a list of length $N^{5/3}$ (one entry for each possible value of x_2^1, y_2) and P_1, P_2, and P_3 use the 3-party PSM protocol $\mathcal{P}_{\text{ind}}$ to send the answer of q_{110} to the referee. As the list contains $N^{5/3}$ entries (where each entry is a bit), the complexity of invoking $\mathcal{P}_{\text{ind}}$ is $O(N^{5/3})$ (the input of P_2 is from $[N]$ and of P_3 is from $[N^{2/3}]$). Queries q_{101}, q_{011} are dealt in a similar way, where P_1 and P_4, respectively, construct the list. To answer query q_{111}, the sets $S_1 \oplus \{y_1\}, S_2 \oplus \{y_2\}, S_3 \oplus \{y_3\}$ are sent to the referee using $\mathcal{P}_{\text{Sxor}}$. As each set is contained in $[N^{4/3}]$, the complexity of invoking $\mathcal{P}_{\text{Sxor}}$ is $O(N^{4/3})$. The total communication and randomness complexity of $\mathcal{P}_4$ is $O(N^{5/3})$.

In $\mathcal{P}_5$, the size of each dimension of the cube is $N^{5/3}$ and the 5 inputs are partitioned into 3 inputs $y_1, y_2, y_3 \in [N^{5/3}]$. The details are similar to the PSM protocol $\mathcal{P}_4$. As for the complexity analysis, to answer q_{110}, party P_1 needs $(y_1, y_2) \in [N^{5/3}]^2$ and it knows $x_1 \in [N]$ which is part of y_1, thus, it creates a list of length $N^{7/3}$, and the complexity of invoking $\mathcal{P}_{\text{ind}}$ is $O(N^{7/3})$. As each set in $\mathcal{P}_5$ is contained in $[N^{5/3}]$, the complexity of invoking $\mathcal{P}_{\text{Sxor}}$ is $O(N^{5/3})$. The total communication and randomness complexity of $\mathcal{P}_5$ is $O(N^{7/3})$. □

Discussion. Our protocols, and the 2-party PSM protocol of [6], use cubes of different dimensions for different values of k; i.e., in [6], 4 dimensions are used for 2 parties, and in this work 3 dimensions are used for $3, 4$, and 5 parties, and 2 dimensions are used for more than 5 parties. These are the optimal dimensions, when using our approach, as we next explain. If there are k inputs (each from the domain $[N]$) and d dimensions, then the size of at least one dimension is $N^{k/d}$. Thus, communicating each set $S_j \oplus \{y_j\}$ (either directly, as in the 2-party and 3-party protocols, or using the PSM protocol $\mathcal{P}_{\text{Sxor}}$, as done for $k > 3$ parties) requires $\Omega(N^{k/d})$ bits. The parties also need to send the answers to the $2^d - 1$ queries of weight at most $d-1$ using protocol $\mathcal{P}_{\text{ind}}$. The most expensive queries are queries of weight $d-1$, which involve $d-1$ "virtual" inputs y_j, each one is taken from $[N^{k/d}]$. As each party P_j knows only x_j, the parties will invoke the PSM protocol $\mathcal{P}_{\text{ind}}$ with a list of length at least $N^{(k/d)(d-1)-1} = N^{k-k/d-1}$; [7] the cost of this invocation will be at least $N^{k-k/d-1}$. Thus, for a given number of parties k, we need to choose d that will minimize $\max\{k/d, k-k/d-1\}$. If, hypothetically, we could choose a non-integral dimension, we would take $d = 2k/(k-1)$ and the complexity of our protocol would have been $O(N^{(k-1)/2})$. This matches the complexity of the PSM protocols for 2 and 3 parties. For $k > 3$, we achieve a slightly worst complexity, since we need to round $2k/(k-1)$ to the nearest integer.

5 PSM Protocols with Inputs of Different Sizes

In this section, we construct PSM protocols for functions in which the domains of inputs are not necessarily the same. That is, we consider functions $f : [N^{\alpha_1}] \times [N^{\alpha_2}] \times \cdots \times [N^{\alpha_k}] \to \{0,1\}$, for some integer N and positive numbers $\alpha_1, \ldots, \alpha_k$. By reordering the parties and normalization, we can assume that $\alpha_1 \geq \alpha_2 \geq \cdots \geq \alpha_k = 1$.

We first observe that the complexity of the PSM of [10] for an arbitrary function does not depend on the size of the largest domain (i.e., of party P_1).

Claim 5.1. *Let $f : [N^{\alpha_1}] \times [N^{\alpha_2}] \times \cdots \times [N^{\alpha_k}] \to \{0,1\}$ be a function. Then, there is a PSM protocol for f with communication complexity $O(N^{\sum_{i=2}^{k} \alpha_i})$ and randomness complexity $O(k \cdot N^{\sum_{i=2}^{k} \alpha_i})$.*

Proof. We describe a protocol with the desired complexity. Party P_1 prepares a list of length $N^{\sum_{i=2}^{k} \alpha_i}$, which contains $f(x_1, i_2, \ldots, i_k)$ for every $(i_2, i_3, \ldots, i_k) \in [N^{\alpha_2}] \times \cdots \times [N^{\alpha_k}]$. The parties invoke the PSM $\mathcal{P}_{\text{ind}}$, where the input of P_1 is this list and the input of P_j, for $2 \leq j \leq k$, is x_j. □

We next construct more efficient k-party protocols. Again, we use the cube approach, with 2 differences compared to the previous protocols. First, the number of dimensions d will be bigger than in our previous protocols (the number

[7] For the interesting parameters (specifically, when $d \leq k$), there will always be a party whose entire input x_i is part of the query and the length of the list would be exactly $N^{k-k/d-1}$.

of dimensions grows as $\sum_{i=2}^{k} \alpha_i$ grows). Thus, the k inputs are partitioned to d virtual inputs. Unlike previous protocols, each P_i, where $1 \leq i \leq k$, holds part of each virtual input.

Lemma 5.2. *Let $f : [N^{\alpha_1}] \times [N^{\alpha_2}] \times \cdots \times [N^{\alpha_k}] \to \{0,1\}$ be a function where $\alpha_1 \geq \alpha_2 \geq \cdots \geq \alpha_k = 1$. Then, there is a PSM protocol for f with communication and randomness complexity*

$$O(\min\{k \cdot N^{(\sum_{i=2}^{k} \alpha_i)(1-1/\lceil a \rceil)}, k^3 \cdot N^{(\sum_{i=1}^{k} \alpha_i)/\lfloor a \rfloor}\})$$

where $a = \alpha_1/(\sum_{i=2}^{k} \alpha_i) + 2$.

Proof. We view f as a d-dimensional cube, where the size of each dimension is $N^{(\sum_{i=1}^{k} \alpha_i)/d}$ and d will be fixed later. We partition the inputs as follows: $x_j = (x_j^1, x_j^2, ..., x_j^d)$ for $1 \leq j \leq k$, where $x_j^1, x_j^2, ..., x_j^d \in [N^{\alpha_j/d}]$. We define $y_\ell = (x_1^\ell, x_2^\ell, \ldots, x_k^\ell)$ for each $1 \leq \ell \leq d$. The common randomness of the parties contains d random sets $S_1, ..., S_d \subseteq_R [N^{(\sum_{i=1}^{k} \alpha_i)/d}]$, random strings for protocols $\mathcal{P}_{\text{ind}}$ and $\mathcal{P}_{\text{Sxor}}$, and random masks. Using the cube approach, the referee should get the answers to 2^d queries. To answer query $q_{11\ldots1}$ (the query of weight d), the sets $S_1 \oplus \{y_1\}, ..., S_d \oplus \{y_d\}$ are sent to the referee using $\mathcal{P}_{\text{Sxor}}$. As each set is contained in $[N^{(\sum_{i=1}^{k} \alpha_i)/d}]$, the complexity of invoking $\mathcal{P}_{\text{Sxor}}$ is $O(k^3 \cdot N^{(\sum_{i=1}^{k} \alpha_i)/d})$. The (masked) answer to $q_{00\ldots0}$ (the query of weight 0) is computed by P_1 and sent to the referee. The answers to the queries of weight m for any $1 \leq m \leq d-1$ are computed using protocol $\mathcal{P}_{\text{ind}}$, where the more expensive queries are queries of weight $d-1$. To answer the query $q_{11\ldots10}$, party P_1, which has $x_1^1, ..., x_1^{d-1}$, prepares a list of length $N^{(\sum_{i=2}^{k} \alpha_i)(d-1)/d}$ (one entry for each possible value of $(x_2^1, \ldots, x_k^1)$, ..., $(x_2^{d-1}, \ldots, x_k^{d-1})$) and the parties $P_1, \ldots, P_k$ use the PSM protocol $\mathcal{P}_{\text{ind}}$ to send the masked answer of $q_{11\ldots10}$ to the referee. The complexity of invoking $\mathcal{P}_{\text{ind}}$ is $O(k \cdot N^{(\sum_{i=2}^{k} \alpha_i)(d-1)/d})$. All other queries are dealt in a similar way, where P_1 constructs the list. The total communication and randomness complexity of the protocol are

$$O(\max\{k \cdot N^{(\sum_{i=2}^{k} \alpha_i)(d-1)/d}, k^3 \cdot N^{(\sum_{i=1}^{k} \alpha_i)/d}\}).$$

If we could choose a non-integral value for d then the optimal value of d in this approach would be $d = a = \alpha_1/(\sum_{i=2}^{k} \alpha_i) + 2$. Since d must be an integral value the communication and randomness complexity are minimized either on $\lfloor a \rfloor$ or on $\lceil a \rceil$. Thus, we get a protocol with the following complexity.

$$O(\min\{k \cdot N^{(\sum_{i=2}^{k} \alpha_i)(1-1/\lceil a \rceil)}, k^3 \cdot N^{(\sum_{i=1}^{k} \alpha_i)/\lfloor a \rfloor}\}).$$

□

Corollary 5.3. *Let $f : [N^{\alpha_1}] \times [N^{\alpha_2}] \times \cdots \times [N^{\alpha_k}] \to \{0,1\}$ be a function where $\alpha_1 \geq \alpha_2 \geq \cdots \geq \alpha_k = 1$. Then, there is a PSM protocol for f with communication and randomness complexity $O(k \cdot N^{(\sum_{i=2}^{k} \alpha_i) \cdot c})$ where $c < 1$.*

5.1 2-Party PSM Protocols

In this section, we describe results for 2-party PSM protocols summarized in the following lemma.

Lemma 5.4. *Let $f : [N^{\alpha}] \times [N] \to \{0,1\}$, where $\alpha \geq 1$. Then, there is a PSM for f with communication and randomness complexity:*

– *$O(N^{\alpha/2})$ if $1 \leq \alpha \leq 3/2$,*
– *$O(N^{(\alpha+1)/(\lfloor \alpha+2 \rfloor)})$ if $\alpha - \lfloor \alpha \rfloor \leq 1/\lceil \alpha+2 \rceil$ and $\alpha > 3/2$,*
– *$O(N^{1-\frac{1}{\lceil \alpha+2 \rceil}})$ otherwise.*

Proof. For the first item, we use the protocol of [6], which has complexity $O(N^{\alpha/2})$ (where we consider both domains to be of size N^{α}). The second and third items follow from Lemma 5.2. $\square$

5.2 3-Party PSM Protocols

In this section, we consider 3-party PSM protocols and show that, for many values of the parameters α_1, α_2, there is a PSM protocol whose complexity is the geometric mean of the sizes of domains.

Claim 5.5. *Let $f : [N^{\alpha_1}] \times [N^{\alpha_2}] \times [N] \to \{0,1\}$, where $\alpha_1 \geq \alpha_2 \geq 1$ and $\alpha_1 \leq 2\alpha_2 - 1$. Then, there is a PSM for f with communication and randomness complexity $O(N^{(\alpha_1+\alpha_2+1)/3})$.*

Proof. The idea of the protocol is similar to the protocols of Theorem 4.2. We view f as a 3-dimensional cube, where the size of each dimension is $N^{(\alpha_1+\alpha_2+1)/3}$. We partition the inputs $x_1 \in [N^{\alpha_1}], x_2 \in [N^{\alpha_2}], x_3 \in [N]$ to equal size inputs $y_1, y_2, y_3 \in [N^{(\alpha_1+\alpha_2+1)/3}]$. The way we partition the inputs will be described below. The common randomness of the parties contains 3 random sets $S_1, S_2, S_3 \subseteq_R [N^{(\alpha_1+\alpha_2+1)/3}]$, random strings for protocols $\mathcal{P}_{\text{ind}}$ and $\mathcal{P}_{\text{Sxor}}$, and random masks. The referee should get the answers to 8 queries $q_{000}, \ldots, q_{111}$. To answer query q_{111}, the sets $S_1 \oplus \{y_1\}, S_2 \oplus \{y_2\}, S_3 \oplus \{y_3\}$ are sent to the referee using $\mathcal{P}_{\text{Sxor}}$. As each set is contained in $[N^{(\alpha_1+\alpha_2+1)/3}]$, the complexity of invoking $\mathcal{P}_{\text{Sxor}}$ is $O(N^{(\alpha_1+\alpha_2+1)/3})$. The (masked) answer to q_{000} is computed by P_1 and sent to the referee. The answers to the queries of weight 1 and 2 are computed using protocol $\mathcal{P}_{\text{ind}}$, where the more expensive queries are queries of weight 2. The details of how to answer these queries depends on the partition of the inputs into y_1, y_2, y_3.

We partition x_1 and x_2 as follows: $x_1 = (x_1^1, x_1^3)$ and $x_2 = (x_2^2, x_2^3)$, where $x_1^1, x_2^2 \in [N^{(\alpha_1+\alpha_2+1)/3}]$, $x_1^3 \in [N^{(2\alpha_1-\alpha_2-1)/3}]$ (note that $2\alpha_1 - \alpha_2 - 1 \geq 0$ since $\alpha_1 \geq \alpha_2 \geq 1$), and $x_2^3 \in [N^{(2\alpha_2-\alpha_1-1)/3}]$ (note that $2\alpha_2 - \alpha_1 - 1 \geq 0$ by our assumption). We define $y_1 = x_1^1, y_2 = x_2^2$, and $y_3 = (x_3, x_1^3, x_2^3)$. To answer the query q_{110}, party P_1, which has $y_1 = x_1^1$, prepares a list of length $N^{(\alpha_1+\alpha_2+1)/3}$ (one entry for each possible value of y_2) and P_1, P_2 use the 2-party PSM protocol $\mathcal{P}_{\text{ind}}$ to send the masked answer of q_{110} to the referee. The complexity of invoking

$\mathcal{P}_{\text{ind}}$ is $O(N^{(\alpha_1+\alpha_2+1)/3})$. Queries q_{101}, q_{011} are dealt in a similar way, where P_1 and P_2, respectively, construct the list and all 3 parties participate (as each has a part of y_3). The total communication and randomness complexity of the protocol are $O(N^{(\alpha_1+\alpha_2+1)/3})$. □

Claim 5.6. *Let $f : [N^{\alpha_1}] \times [N^{\alpha_2}] \times [N] \to \{0,1\}$, where $\alpha_1 \geq \alpha_2 \geq 1$ and $\alpha_1 \geq 2\alpha_2 - 1$. Then, there is a PSM for f with communication and randomness complexity $O(N^{(\alpha_1+1)/2})$.*

Proof. The idea of the protocol is similar to the protocols of Theorem 4.2. We view f as a 3-dimensional cube, where the size of each dimension is $N^{(\alpha_1+\alpha_2+1)/3}$. We define $x_1 = (x_1^1, x_1^3)$, where $x_1^1 \in [N^{(\alpha_1+1)/2}]$ and $x_1^3 \in [N^{(\alpha_1-1)/2}]$, and $y_1 = x_1^1, y_2 = x_2$, and $y_3 = (x_3, x_1^3)$. The common randomness of the parties contains 3 random sets $S_1, S_3 \subseteq_R [N^{(\alpha_1+1)/2}]$ and $S_2 \subseteq_R [N^{\alpha_2}]$, random strings for protocols $\mathcal{P}_{\text{ind}}$ and $\mathcal{P}_{\text{Sxor}}$, and random masks. The referee should get the answers to 8 queries $q_{000}, \ldots, q_{111}$. To answer query q_{111}, the sets $S_1 \oplus \{y_1\}, S_2 \oplus \{y_2\}, S_3 \oplus \{y_3\}$ are sent to the referee using $\mathcal{P}_{\text{Sxor}}$. The complexity of invoking $\mathcal{P}_{\text{Sxor}}$ is $O(N^{(\alpha_1+1)/2})$ (where, for S_2, we use the assumption that $\alpha_1 \geq 2\alpha_2 - 1$). The (masked) answer to q_{000} is computed by P_1 and sent to the referee. The answers to the queries of weight 2 and 3 are computed using protocol $\mathcal{P}_{\text{ind}}$, where the most expensive query is q_{011}, for which P_2 has to prepare a list of length $N^{(\alpha_1+1)/2}$ (one entry for every possible value of y_3) and parties P_1, P_2, and P_3 use the 3-party PSM protocol $\mathcal{P}_{\text{ind}}$ to send the answer of q_{011} to the referee. The complexity of invoking $\mathcal{P}_{\text{ind}}$ is $O(N^{(\alpha_1+1)/2})$. Queries q_{101}, q_{011} are dealt in a similar way. The total communication and randomness complexity of the protocol are $O(N^{(\alpha_1+1)/2})$. □

The next two lemmas summarize the various cases of 3-argument functions; the first claim deals with the case $\alpha_2 \geq 2$ and the second claim with the case $\alpha_2 < 2$.

Lemma 5.7. *Let $f : [N^{\alpha_1}] \times [N^{\alpha_2}] \times [N] \to \{0,1\}$, where $\alpha_1 \geq \alpha_2 \geq 1$. If $\alpha_2 \geq 2$ then there is a PSM for f with communication and randomness complexity:*

- *$O(N^{(\alpha_1+\alpha_2+1)/3})$ if $\alpha_1 < \alpha_2 + 1$.*
- *$O(\min\{N^{(\alpha_2+1)(1-1/\lceil a \rceil)}, N^{(\alpha_1+\alpha_2+1)/\lfloor a \rfloor}\})$ where $a = (\alpha_1+2\alpha_2+2)/(\alpha_2+1)$ if $\alpha_1 \geq \alpha_2 + 1$.*

Proof. If $\alpha_1 < \alpha_2 + 1$, then $\alpha_1 < 2\alpha_2 - 1$ (since $\alpha_2 \geq 2$). Thus, the first item is implied by Claim 5.5. The second item follows from Lemma 5.2. □

Lemma 5.8. *Let $f : [N^{\alpha_1}] \times [N^{\alpha_2}] \times [N] \to \{0,1\}$, where $\alpha_1 \geq \alpha_2 \geq 1$. Assume $1 \leq \alpha_2 < 2$. Then there is a PSM for f with communication and randomness complexity:*

- *$O(N^{(\alpha_1+\alpha_2+1)/3})$ if $\alpha_2 \leq \alpha_1 \leq 2\alpha_2 - 1$.*
- *$O(N^{(\alpha_1+1)/2})$ if $2\alpha_2 - 1 \leq \alpha_1 \leq (4\alpha_2+1)/3$.*
- *$O(\min\{N^{(\alpha_2+1)(1-1/\lceil a \rceil)}, N^{(\alpha_1+\alpha_2+1)/\lfloor a \rfloor}\})$ where $a = (\alpha_1+2\alpha_2+2)/(\alpha_2+1)$ if $(4\alpha_2+1)/3 \leq \alpha_1$.*

Proof. The first item follows from Claim 5.5. The second item follows from Claim 5.6. The third item follows from Lemma 5.2. □

6 A PSM for k parties from a PSM for t parties

In this section, we show how to construct a k-party PSM protocol for a function from a t-party PSM protocol for a related function where $k > t$. This is a generic transformation, which does not result in better protocols than the protocols presented in this paper. However, it shows that improvements in the complexity of t-party PSM protocols for small values of t will results in better k-party PSM protocols for all values k.

Claim 6.1. *Let k, t, N be integers such that $k > t$ and $N \geq 2^t$ and let $g : [N]^k \rightarrow \{0, 1\}$ be a function. If every function $f : [n]^t \rightarrow \{0, 1\}$ has a t-party PSM protocol with communication and randomness complexity $O(n^\alpha)$ for $n = N^{k/t}$, then there is a PSM protocol for g with communication and randomness complexity $O(k \cdot t \cdot N^{(\alpha+1)k/t-1})$.*

Proof. We construct the following protocol $\mathcal{P}_k$ for g. We partition the inputs $x_1, ..., x_k$ to equal size inputs $y_1, ..., y_t$ where $y_j \in [N^{k/t}]$ for $j = 1, ..., t$. Let $x_i = (x_i^1, ..., x_i^t)$ for all $i = t + 1, ..., k$ where $x_i^1, ..., x_i^t \in [N^{1/t}]$. We define $y_j = (x_j, x_{t+1}^j, ..., x_k^j)$ for $j = 1, ..., t$. Furthermore, we define $g' : [N^{k/t}]^t \rightarrow \{0, 1\}$ such that $g'(y_1, ..., y_t) = g(x_1, ..., x_k)$. By the assumption of the claim, there is a PSM protocol $\mathcal{P}_t$ for g' with communication and randomness complexity $O(n^\alpha) = O(N^{\alpha k/t})$.

In protocol $\mathcal{P}_k$, for every $1 \leq j \leq t$, party P_j together with parties $P_{t+1}, ..., P_k$ simulate the j-th party of protocol $\mathcal{P}_t$ as follows. For this simulation, party P_j prepares a list of length $N^{k/t-1}$ of the possible messages of the j-th party in $\mathcal{P}_t$ with input $y_j = (x_j, x_{t+1}^j, ..., x_k^j)$. As P_j knows x_j and does not know $x_{t+1}^j, ..., x_k^j$ the list contains one entry for every possible value of $(x_{t+1}^j, ..., x_k^j) \in [N^{k/t-1}]$. Each entry in the list is a message taken from $[N^{\alpha k/t}]$. Parties $P_j, P_{t+1}, ..., P_k$ use the PSM protocol $\mathcal{P}_{\text{ind}}$ to send the message corresponding to their inputs to the referee. The referee computes the t messages and then computes $g(x_1, ..., x_k) = g'(y_1, ..., y_t)$ according to the protocol $\mathcal{P}_t$. The communication and randomness complexities of the PSM protocol $\mathcal{P}_{\text{ind}}$ are $O(k \cdot N^{(\alpha+1)k/t-1})$ and the parties $P_{t+1}, ..., P_k$ invoke this protocol t times. Therefore, the total communication and randomness complexity of this protocol are $O(k \cdot t \cdot N^{(\alpha+1)k/t-1})$. $\square$

For example, for $t = 2$ there is a 2-party PSM protocol with complexity $O(N^{0.5})$ [6] (here $\alpha = 0.5$). This implies that any function $f : [N]^k \rightarrow \{0, 1\}$ has a k-party PSM protocol with complexity $O(N^{3k/4-1})$. This is inferior to our protocols from Sections 3 and 4. For $t = 3$, we get that any function $f : [N]^k \rightarrow \{0, 1\}$ has a k-party PSM protocol with complexity $O(N^{2k/3-1})$ (using our 3-party PSM protocol which has complexity $O(N)$, i.e. $\alpha = 1$). In particular, for $k = 4$, we can construct a 4-party PSM protocol with complexity $O(N^{5/3})$ from our 3-party PSM protocol, matching the complexity of our protocol from Section 4.1. Thus, if one can improve the message length of 3-party PSM protocols for an arbitrary function to $o(N)$, then this would yield 4-party PSM protocol with message length $o(N^{5/3})$ improving on our construction.

To conclude, to improve the complexity of k-party PSM protocols for an arbitrary functions, one might want to start with designing k-party PSM protocols for small values of k. For example, if the complexity of the 2-party PSM protocols will be improved to $O(N^{\beta})$ for $\beta < 2/k$, then we will get k-party PSM protocols with complexity better than $O(N^{k/2})$.

7 Applications

In this section, we present some applications of our PSM protocols for several cryptographic primitives.

7.1 *t*-robust NIMPC protocols

A Non-Interactive secure Multi-Party Computation (NIMPC) protocol, defined in [5], is a PSM protocol that is secure even if some parties collude with the referee. Such parties may send the referee their messages for *every* possible input and therefore the referee can always compute the function on many inputs. The model is defined with *correlated* randomness $r_1, \dots, r_k$ between the k parties, rather than common randomness, and the dishonest parties can, alternatively, send the referee their r_i's.

In such setting, one may only hope for a so-called "best possible security"; that is, in a t-robust NIMPC protocol, an adversary controlling at most t parties and seeing all the messages sent by honest parties learns no information that is not implied by the anavoidable information – the restriction of f fixing the inputs of the honest parties.

The communication complexity of the best known fully-robust (i.e., k-robust) NIMPC protocol, for an arbitrary function $f : [N]^k \to \{0,1\}$, was $O(\text{poly}(\log N, k) \cdot N^k)$ [19] (improving on [5]). We show that every function $f : [N]^k \to \{0,1\}$ has a t-robust NIMPC with complexity $\tilde{O}(N^{k/2+t})$, which improves on previous constructions when $t < k/2$. Our construction is based on an information-theoretic transformation of [8], which constructs a t-robust NIMPC protocol for f from any PSM (that is, 0-robust NIMPC) protocol for f.

Theorem 7.1 ([8]). *Let t be a positive integer and $\mathcal{P}$ be a PSM protocol for a Boolean function $f : [N]^k \to \{0,1\}$ with randomness and communication complexity $\alpha(N)$. Then, there exists a t-robust NIMPC protocol for f with randomness and communication complexity $O((2\max\{N,k\})^{t+1}k(t\log(N+k)+\alpha(N)))$.*

Using Theorem 7.1 and our PSM protocol $\mathcal{P}_k$, we get a t-robust NIMPC protocol for $f : [N]^k \to \{0,1\}$ with randomness and communication complexity $O(k^4 2^t N^{k/2+t+1})$ (assuming that $N \geq k$). We next construct a t-robust NIMPC protocol where we improve the complexity by a factor of N. This optimization is more significant when k and t are small. For example, consider a 3-argument function $f : [N]^3 \to \{0,1\}$. By Theorem 3.1, the function f has a PSM protocol with complexity $O(N)$. Thus, by Theorem 7.1, we get a 1-robust PSM for f with complexity $O(N^3)$. Notice that a 1-robust 3-party NIMPC protocol is a fully-robust protocol (as even in an ideal world 2 parties can basically learn the input of the third party). Our optimized fully robust NIMPC protocol for f will have complexity $O(N^{2.5}\log^4 N)$.

Theorem 7.2. *Let t, k, N be positive integers such that $t < k/2$ and $k \le N$ and $f : [N]^k \to \{0,1\}$ be a Boolean function. Then, there exists a t-robust NIMPC protocol for f with randomness and communication complexity $O(k^4 \log^4 N \cdot (2N)^{k/2+t})$.*

Proof. The basic idea is that, instead of viewing f as a k-argument function, where each party has an input from $[N]$, we consider a $(k \log N)$-argument function f', where each party has an input from $\{0,1\}$ and construct a $(t \log N)$-robust protocol for f'. For simplicity of notation we assume that N is a power of 2. (This results in multiplying N by at most 2, yielding the term $2N$ in Theorem 7.2). We define $f' : \{0,1\}^{k \log N} \to \{0,1\}$, where

$$\begin{aligned} &f'(x_{1,1}, \ldots, x_{1,\log N}, \ldots, x_{k,1}, \ldots, x_{k,\log N}) \\ &= f((x_{1,1}, \ldots, x_{1,\log N}), \ldots, (x_{k,1}, \ldots, x_{k,\log N})). \end{aligned}$$

By Theorem 4.1, the function f' has a PSM protocol with communication and randomness complexity $O((k \log N)^3 2^{(k \log N)/2}) = O(k^3 \log^3 N \cdot N^{k/2})$. We cannot apply Theorem 7.1 directly, as it will not result in the desired complexity. We observe that the NIMPC that we construct needs to be robust only against sets T that contain t blocks of size $\log N$ (where a block is a set of $\log N$ parties of f' holding the bits of some party P_j in f). By [8, Thm. 6.4], we need a matrix H' that is T-admissible only for such sets T.[8] If we have such matrix with ℓ rows over a finite field $\mathbb{F}_q$, then we can replace the term $(2\max\{N,k\})^{t+1}$ with $q^{\ell+1}$ in Theorem 7.1.

We next explain how to construct such H'. In [8], it was shown that a $t \times k$ parity-check matrix H of the Reed-Solomon code over $\mathbb{F}_q$, where $q \ge \max\{k, N\}$ is a prime-power, is T-admissible for every set T of size t. We take such matrix H over the field $\mathbb{F}_{2^{\log N}}$ and construct a matrix H' over $\mathbb{F}_2$ with $t \log N$ rows and $k' = k \log N$ columns by replacing each entry $a \in \mathbb{F}_{2^{\log N}}$ with a $\log N \times \log N$ matrix A over $\mathbb{F}_2$ such that for every $b \in \mathbb{F}_{2^{\log N}}$ it holds that ab, viewed as a vector over $\mathbb{F}_2$, is the same as $A(b_0, \ldots, b_{\log N-1})^T$ (when viewing elements of $\mathbb{F}_{2^{\log N}}$ as polynomials, the matrix A simulates the multiplication of the polynomial by the polynomial representing a modulo the irreducible polynomial). The matrix H' is T-admissible for every T that contains at most t blocks.

By [8, Thm. 6.4], the function f' has an NIMPC protocol that is T-robust, for every T that contains at most t blocks, and has complexity

$$O(q^{\ell+1}(k')^4 2^{k'/2})) = O(2^{t \log N+1}(k \log N)^4 \cdot 2^{(k \log N)/2}) = O(k^4 \log^4 N \cdot N^{k/2+t}).$$

If N is not a power of 2, we need to replace N by $2N$ in the complexity. To construct a PSM protocol for f, the parties $P_1, \ldots, P_k$ execute the PSM for f',

[8] A matrix H' is T-admissible if $H'\boldsymbol{u} \neq H'\boldsymbol{v}$ for any two vectors $\boldsymbol{u}, \boldsymbol{v}$ that agree on all entries not indexed by T.

where each party P_j sends the messages of the block of parties holding the bits of its input. $\square$

If we consider a 3-argument function $f : [N]^3 \to \{0,1\}$, we get a 1-private NIMPC protocol with communication and randomness complexity $O(\log^4 N \cdot N^{5/2})$. As discussed above, this protocol is fully robust. This improves on the previously best known 3-party NIMPC protocol of [19] whose complexity is $O(N^3 \log^2 N)$. While the protocol of [19] hides the function f, in our protocol the referee needs to know f in order to reconstruct its output. In [19], it is proved that in any NIMPC protocol for every k-argument function $f : [N]^k \to \{0,1\}$ that hides the function, the size of the common randomness is $\Omega(N^k)$. The proof of this lower bound actually holds for any PSM protocol in which the reconstruction of f's value is independent of f. Thus, in a fully robust 3-party NIMPC protocol with randomness complexity $o(N^3)$ for an arbitrary function $f : [N]^3 \to \{0,1\}$, the referee has to know f in advance.

Example 7.3. For the 3-party case, the construction of the admissible matrix H' is much simpler starting from the 1-admissible matrix $H = (1,1,1)$. The resulting matrix H' is $H' = (I_{\log N} I_{\log N} I_{\log N})$ (that is, H' contains 3 copies of the $(\log N) \times (\log N)$ identity matrix). Consider two vectors $\boldsymbol{u}, \boldsymbol{v}$ that differ only in the first block i.e., $\boldsymbol{u} = (\boldsymbol{u_1}, \boldsymbol{u_2}, \boldsymbol{u_3})$ and $\boldsymbol{v} = (\boldsymbol{v_1}, \boldsymbol{v_2}, \boldsymbol{v_3})$ where $\boldsymbol{u_1} \neq \boldsymbol{v_1}$ while $\boldsymbol{u_2} = \boldsymbol{v_2}$ and $\boldsymbol{u_3} = \boldsymbol{v_3}$. Then, $H'\boldsymbol{u} - H'\boldsymbol{v} = H'(\boldsymbol{u} - \boldsymbol{v}) = I_{\log N}(\boldsymbol{u_1} - \boldsymbol{v_1}) \neq \mathbf{0}$. Thus, H' is T-admissible for the 3 blocks T.

7.2 Ad-hoc PSM Protocols and Homogeneous Distribution Designs

A k-out-of-n ad-hoc PSM protocol is a PSM protocol with n parties, where only k parties, whose identity is not known in advance, actually participate in the protocol. For a formal denition of ad-hoc PSM protocols see [7].

We obtain improved k-out-of-n ad-hoc PSM protocols for symmetric functions[9] $f : [N]^k \to \{0,1\}$ with communication complexity $O(e^k \cdot k^6 \cdot \log n \cdot N^{k/2})$ and randomness complexity $O(e^k \cdot k^8 \cdot \log n \cdot N^{k/2})$, based on a transformation from k-party PSM protocols to ad-hoc PSM protocols from [7] and on our new PSM protocols.

Theorem 7.4 ([7]). *Assume that there is a k-party PSM protocol Π for a symmetric function f with randomness complexity $\mathrm{Rnd}(\Pi)$ and communication complexity $\mathrm{Com}(\Pi)$. Then, there is a k-out-of-n ad-hoc PSM protocol for f with randomness complexity $O(e^k \cdot k^3 \cdot \log n \cdot (\mathrm{Rnd}(\Pi) + k^2 \cdot \max\{\mathrm{Com}(\Pi), \log n\}))$ and communication complexity $O(e^k \cdot k^3 \cdot \log n \max\{\mathrm{Com}(\Pi), \log n\})$.*

Using Theorem 7.4 and our PSM protocol $\mathcal{P}_k$, whose communication and randomness complexity are $O(k^3 N^{k/2})$, we get:

[9] A function f is symmetric if for every input $(x_1, \ldots, x_k) \in [N]^k$ and every permutation $\pi : [k] \to [k]$, it holds that $f(x_1, \ldots, x_k) = f(x_{\pi(1)}, \ldots, x_{\pi(k)})$.

Theorem 7.5. *Let $f : [N]^k \to [N^k]$ be a symmetric function. Then, there is a k-out-of-n ad-hoc PSM protocol for f with communication complexity $O(e^k \cdot k^6 \cdot \log n \cdot N^{k/2})$ and randomness complexity $O(e^k \cdot k^8 \cdot \log n \cdot N^{k/2})$.*

We next show that ad-hoc PSM protocols imply distribution design. The goal of *distribution design*, introduced in [4], is to find a joint distribution on N random variables $(X_1, X_2, \ldots, X_N)$ that satisfies a given set of constraints on the marginal distributions. If such a distribution exists we say that the set of constraints is realizable. Each constraint involves two equal-size sets $\{i_1, \ldots, i_d\}$ and $\{j_1, \ldots, j_d\}$ and can either be an equality constraint of the form $\{i_1, \ldots, i_d\} \equiv \{j_1, \ldots, j_d\}$, in which case the two marginal distributions should be identical, or a disjointness constraint of the form $\{i_1, \ldots, i_d\} \parallel \{j_1, \ldots, j_d\}$, in which case the two marginal distributions should have disjoint supports.[10] A k-homogenous set of constraints is one where all sets in the constraints have the same size k. Borrowing terminology from secret sharing (which is one of the applications of distribution design), we refer to the value of a random variable X_i as the ith share.

We obtain distribution designs for 2-homogeneous sets of constraints with share size $O(\sqrt{N} \cdot \log^2 N)$. Previously, the best known distribution design had share size of $O(N \log N)$. We also obtain distribution designs for k-homogeneous sets of constraints, for $k > 2$, with share size $O(e^k \cdot k^6 \cdot N^{k/2} \cdot \log N)$, improving on the best known distribution design whose share size was $O(\binom{N}{k} \cdot \min\{d \log N, N\})$. Our construction is based on a PSM protocol for a function that represents the constraints. We use the messages of this protocol to define the random variables.

Definition 7.6. *Given a k-homogenous set of constraints $\mathcal{R}$ on N variables, define the following symmetric function $f_{\mathcal{R}}$: Construct an undirected graph, whose vertices are all subsets of size k. Connect two vertices (sets of size k) A and B if and only if "$A \equiv B$" $\in \mathcal{R}$ and let $\mathcal{A}_1, \ldots, \mathcal{A}_\ell$ be the connected components of this graph.[11] Define a symmetric function $f_{\mathcal{R}} : [N]^k \to \binom{N}{k}$ such that, for every set $\{i_1, \ldots, i_k\}$ (i.e. $i_1 < i_2 < \ldots < i_k$), define $f_{\mathcal{R}}(i_1, \ldots, i_k) = j$ where $(i_1, \ldots, i_k) \in \mathcal{A}_j$. To make this function symmetric, define $f_{\mathcal{R}}(\sigma(i_1), \ldots, \sigma(i_k)) = f_{\mathcal{R}}(i_1, \ldots, i_k)$ for every permutation $\sigma : [k] \to [k]$.*

Theorem 7.7. *Let $\mathcal{R}$ be a k-homogenous set of constraints on N variables. Assume that there is a k-out-of-N ad-hoc PSM protocol $\mathcal{P}_{k,N}$ for $f_{\mathcal{R}} : [N]^k \to \binom{N}{k}$ with communication complexity $\mathrm{Com}(\mathcal{P}_{k,N})$. If $\mathcal{R}$ is realizable, then there is a distribution design X realizing $\mathcal{R}$ with share size $\mathrm{Com}(\mathcal{P}_{k,N}) + \log N$.*

[10] We only consider the projective case, in which the constraints are restricted to be on sets of variables; that is, the elements in the sets are sorted. In [4], also constraints for non-sorted elements are considered.

[11] By [4], a distribution design exists if and only if for every constraint "$A \| B$" $\in \mathcal{R}$, the sets A and B are in different connected components. Furthermore, it is enough to construct a distribution design X where $X_A \equiv X_B$ whenever A and B are in the same component $\mathcal{A}_i$, and $X_A \parallel X_B$ whenever A and B are in different components.

Proof. To construct the distribution design, we first choose a random permutation $\pi : [N] \to [N]$. Let $M_{\pi(i)}$ be the messages in $\mathcal{P}_{k,N}$ of party $P_{\pi(i)}$ with input i. We set X_i to be $(M_{\pi(i)}, \pi(i))$ for all $i \in [N]$. The reason that we choose the permutation π is that an ad-hoc PSM protocol does not hide the identities of the parties sending messages. We claim that $X = (X_1, \ldots, X_N)$ realizes $\mathcal{R}$:

Equivalence constraints. Let $A = \{i_1, \ldots, i_k\}$ and let $B = \{j_1, \ldots, j_k\}$ be two sets such that "$(i_1, \ldots, i_k) \equiv (j_1, \ldots, j_k)$" $\in \mathcal{R}$. Vertices A and B are in the same connected component and therefore $f_{\mathcal{R}}(i_1, \ldots, i_k) = f_{\mathcal{R}}(j_1, \ldots, j_k)$. From the construction, $X_A = (X_{i_1}, \ldots, X_{i_k}) = ((M_{\pi(i_1)}, \pi(i_1)), \ldots, (M_{\pi(i_k)}, \pi(i_k)))$, and $X_B = (X_{j_1}, \ldots, X_{j_k}) = ((M_{\pi(j_1)}, \pi(j_1)), \ldots, (M_{\pi(j_k)}, \pi(j_k)))$. We argue that X_A and X_B are equally distributed. First, π is a random permutation, thus $\{\pi(i_1), \ldots, \pi(i_k)\} \equiv \{\pi(j_1), \ldots, \pi(j_k)\}$. We next fix a set $C = \{c_1, ..., c_k\} \subseteq [N]$ of size k and show that the distribution of X_A conditioned on $\pi(i_\ell) = c_\ell$ for every $\ell \in [k]$ is equal to the distribution of X_B conditioned on $\pi(j_\ell) = c_\ell$ for every $\ell \in [k]$. That is, X_A and X_B contain the messages of the same set C with inputs $i_1, \ldots, i_k$ and $j_1, \ldots, j_k$ respectively. Since $f_{\mathcal{R}}(i_1, \ldots, i_k) = f_{\mathcal{R}}(j_1, \ldots, j_k)$, by the security of the ad-hoc PSM protocol, these messages are equally distributed.

Disjointness constraints. Let $A = \{i_1, \ldots, i_k\}$ and let $B = \{j_1, \ldots, j_k\}$ be two subsets such that "$(i_1, \ldots, i_k) \parallel (j_1, \ldots, j_k)$" $\in \mathcal{R}$. Vertices A and B are in different connected components and therefore $f_{\mathcal{R}}(i_1, \ldots, i_k) \neq f_{\mathcal{R}}(j_1, \ldots, j_k)$. Let X'_A and X'_B be the messages in X_A and X_B respectively sorted according to π; i.e. $X'_A = (M_{\pi(i_{t_1})}, \ldots, M_{\pi(i_{t_k})})$ such that $\pi(i_{t_1}) < \pi(i_{t_2}) < \ldots < \pi(i_{t_k})$ and, similarly, $X'_B = (M_{\pi(j_{m_1})}, \ldots, M_{\pi(j_{m_k})})$ such that $\pi(j_{m_1}) < \pi(j_{m_2}) < \ldots < \pi(j_{m_k})$. By the correctness of the ad-hoc PSM protocol $\mathcal{P}_{k,N}$, we can reconstruct $f_{\mathcal{R}}(i_1, \ldots, i_k)$ from the messages $(M_{\pi(i_{t_1})}, \ldots, M_{\pi(i_{t_k})})$ and, similarly, we can reconstruct $f_{\mathcal{R}}(j_1, \ldots, j_k)$ from the messages $(M_{\pi(j_{m_1})}, \ldots, M_{\pi(j_{m_k})})$ and therefore, $X_A \parallel X_B$. □

Corollary 7.8. *Let $\mathcal{R}$ be a k-homogenous set of constraints on N variables. If $\mathcal{R}$ is realizable, then there is a distribution design X realizing $\mathcal{R}$ with share size $O(k^3 N^{k/2})$. If $k = 2$, the size of the shares is $O(N^{0.5})$, if $k = 3$, the size of the shares is $O(N)$, if $k = 4$, the size of the shares is $O(N^{5/3})$, and if $k = 5$, the size of the shares is $O(N^{7/3})$.*

The size of the shares in our distribution design is smaller than the size of the shares in the distribution designs of [6] for homogeneous sets of constraints when the number of constraints is large. However, the distribution designs in [6] have extra properties that can be used to construct distribution designs for non-homogeneous sets of constraints. We do not know how to use our distribution designs to realize non-homogeneous sets of constraints.

7.3 Conditional Disclosure of Secrets and Strongly-Homogeneous Secret-Sharing Schemes

A CDS protocol allows a set of parties $P_1, \ldots, P_k$ to disclose a secret to a referee, subject to a given condition on their inputs. In such a protocol, each party P_i

holds an input $x_i \in [N]$, a joint secret s, and a common random string, and there is a public function $f : [N]^k \to \{0,1\}$. The referee knows $x_1, \ldots, x_k$. The protocol involves only a unidirectional communication from the parties to the referee, which should learn s if and only if $f(x_1, \ldots, x_k) = 1$.

Using the transformations of [6, 12] from PSM protocols to CDS protocols, our PSM protocols from Section 4 imply k-party CDS protocols with complexity $O(k^3 \cdot N^{k/2})$ for an arbitrary function. However, in a very recent result, Liu, Vaikuntanathan, and Wee [17] have constructed much more efficient CDS protocols; they construct a k-party CDS protocols for any function $f : [N]^k \to \{0,1\}$ with complexity $2^{\tilde{O}(\sqrt{k \log N})}$.

We show that a CDS protocol implies a secret-sharing scheme for strongly t-homogenous access structures in which the share size is the communication complexity of the CDS protocols. An access structure is said to be t-homogeneous if the size of every minimal authorized sets is t. An access structure is said to be strongly t-homogeneous if the size of every minimal authorized sets is either t or $t+1$ and all sets of size at least $t+1$ are authorized. That is, every set of size $< t$ is unauthorized, every set of size $> t$ is authorized and some sets of size t are authorized. Our construction realizing a strongly homogeneous access structure takes a CDS protocol for a function $f : \{0,1\}^k \to \{0,1\}$ and transforms it to a secret-sharing scheme realizing a strongly t-homogenous access structure (with k parties).

Definition 7.9. *Let $\mathcal{P} = \{P_1, \ldots, P_k\}$ be a set of parties. We represent a subset of parties $A \subseteq \mathcal{P}$ by its characteristic string $x_A = (x_1, \ldots, x_k) \in \{0,1\}^k$ where for each $i \in [k]$, $x_i = 1$ if and only if $P_i \in A$. For a strongly t-homogenous access structure $\mathcal{A}$, we define a function $f_{\mathcal{A}} : \{0,1\}^k \to \{0,1\}$ such that for every subset of parties $A \subseteq \mathcal{P}$, $f(x_A) = 1$ if and only if $A \in \mathcal{A}$. In particular, if $|A| > t$ then $f(x_A) = 1$.*

Theorem 7.10. *Let $\mathcal{A}$ be a strongly t-homogenous access structure on a set of parties $\mathcal{P} = \{P_1, \ldots, P_k\}$, and let Π be a CDS protocol for the function $f_{\mathcal{A}} : \{0,1\}^k \to \{0,1\}$ with a secret s. If Π has communication complexity $\mathrm{Com}(\Pi)$, then there is a secret-sharing scheme for $\mathcal{A}$ with share size*

$$O(k \cdot \max\{\mathrm{Com}(\Pi), \log k\} + \log k).$$

Proof. Assume the dealer wants to share a secret $s \in \{0,1\}$. The dealer chooses at random a bit $s_1 \in \{0,1\}$ and shares s_1 using Shamir's t-out-of-k secret-sharing scheme. Let $s_2 = s \oplus s_1$. The dealer chooses the randomness r, required for the CDS protocol Π. Let $M_{i,0}, M_{i,1}$ be the message of party P_i on inputs 0 and 1 respectively in the CDS protocol Π with secret s_2 and randomness r. For each $i \in [k]$ the dealer shares $M_{i,0}$ in Shamir's t-out-of-$(k-1)$ threshold secret-sharing scheme among all the parties except for party P_i. Next, the dealer gives to each party P_i the share $M_{i,1}$. The size of $M_{i,b}$ is $\mathrm{Com}(\Pi)$ for each $i \in [k]$, and $b \in \{0,1\}$. The share of each P_i party is $M_{i,1}$ and $k-1$ additional shares created in the t-out-of-$(k-1)$ Shamir's threshold secret-sharing scheme for messages of the CDS, and a share of the bit s_1. Thus, the total share size is $k \cdot \mathrm{Com}(\Pi) + \log k$ (assuming that $\mathrm{Com}(\Pi) \geq \log k$). We claim that this secret-sharing scheme realizes $\mathcal{A}$.

First, let A be an authorized set of parties such that $|A| \geq t$, thus, $f(x_A) = 1$ and the parties in A can reconstruct s_1. By the correctness of Π, if the parties have the messages $M_{i,1}$ for each $P_i \in A$ and $M_{i,0}$ for each $P_i \notin A$, then they can reconstruct the secret s_2. As $|A| \geq t$, the parties in A hold at least t shares of $M_{i,0}$ for every $P_i \notin A$ and therefore, they can reconstruct $M_{i,0}$. The parties in A also have their message on input 1; i.e. $M_{i,1}$ for each $P_i \in A$. Therefore, they can reconstruct the secret s_2, and therefore, reconstruct s.

Second, let A be a set of parties such that $|A| < t$. The parties have no information on s_1, hence no information on s. Finally, let A be an unauthorized subset of parties such that $|A| = t$. From the correctness of Shamir's secret-sharing scheme, the parties in A can reconstruct $M_{i,0}$ for every $P_i \notin A$ and have no information on $M_{i,0}$ for $P_i \in \mathcal{A}$ (since P_i does not get a share of $M_{i,0}$). The parties in A also have their message on input 1; i.e. $M_{i,1}$ for each $P_i \in A$. The set A is unauthorized, thus, $f(x_A) = 0$. From the security of Π, the parties in A do not learn any information on the secret s_2 from the messages on input $f(x_A) = 0$, therefore, they learn no information about s. □

Corollary 7.11. *Let $\mathcal{A}$ be a strongly t-homogeneous access structure with k parties. Then, there exists a secret-sharing scheme realizing $\mathcal{A}$ with shares of size $2^{\tilde{O}(\sqrt{k})}$.*

References

1. B. Applebaum and P. Raykov. From private simultaneous messages to zero-information Arthur-Merlin protocols and back. In *TCC 2016*, volume 9563 of *LNCS*, pages 65–82, 2016.
2. A. Beimel, O. Farràs, Y. Mintz, and N. Peter. Linear secret-sharing schemes for forbidden graph access structures. In *TCC 2017*, volume 10678 of *LNCS*, pages 394–423, 2017.
3. A. Beimel, O. Farràs, and N. Peter. Secret sharing schemes for dense forbidden graphs. In *SCN 2016*, volume 9841 of *LNCS*, pages 509–528. Springer-Verlag, 2016.
4. A. Beimel, A. Gabizon, Y. Ishai, and E. Kushilevitz. Distribution design. In *ITCS 2016*, pages 81–92, 2016.
5. A. Beimel, A. Gabizon, Y. Ishai, E. Kushilevitz, S. Meldgaard, and A. Paskin-Cherniavsky. Non-interactive secure multiparty computation. In *CRYPTO 2014*, volume 8617 of *LNCS*, pages 387–404, 2014.
6. A. Beimel, Y. Ishai, R. Kumaresan, and E. Kushilevitz. On the cryptographic complexity of the worst functions. In *TCC 2014*, volume 8349 of *LNCS*, pages 317–342, 2014.
7. A. Beimel, Y. Ishai, and E. Kushilevitz. Ad hoc PSM protocols: Secure computation without coordination. In *EUROCRYPT 2017*, volume 10212 of *LNCS*, pages 580–608, 2017.
8. F. Benhamouda, H. Krawczyk, and T. Rabin. Robust non-interactive multiparty computation against constant-size collusion. In *CRYPTO 2017*, volume 10401 of *LNCS*, pages 391–419, 2017.

9. B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan. Private information retrieval. *J. of the ACM*, 45:965–981, 1998.
10. U. Feige, J. Kilian, and M. Naor. A minimal model for secure computation. In *26th STOC*, pages 554–563, 1994.
11. R. Gay, I. Kerenidis, and H. Wee. Communication complexity of conditional disclosure of secrets and attribute-based encryption. In *CRYPTO 2015*, LNCS, pages 485–502. Springer-Verlag, 2015.
12. Y. Gertner, Y. Ishai, E. Kushilevitz, and T. Malkin. Protecting data privacy in private information retrieval schemes. *JCSS*, 60(3):592–629, 2000.
13. Y. Ishai, R. Kumaresan, E. Kushilevitz, and A. Paskin-Cherniavsky. Secure computation with minimal interaction, revisited. In *CRYPTO 2015*, volume 9216, pages 359–378, 2015.
14. Y. Ishai and E. Kushilevitz. Private simultaneous messages protocols with applications. In *5th Israel Symp. on Theory of Computing and Systems*, pages 174–183, 1997.
15. Y. Ishai, E. Kushilevitz, and A. Paskin. Secure multiparty computation with minimal interaction. In *CRYPTO 2010*, volume 6223 of *LNCS*, pages 577–594, 2010.
16. T. Liu, V. Vaikuntanathan, and H. Wee. Conditional disclosure of secrets via non-linear reconstruction. In *CRYPTO 2017*, volume 10401 of *LNCS*, pages 758–790, 2017.
17. T. Liu, V. Vaikuntanathan, and H. Wee. Towards breaking the exponential barrier for general secret sharing. In *Eurocrypt 2018*, 2018.
18. H.-M. Sun and S.-P. Shieh. Secret sharing in graph-based prohibited structures. In *INFOCOM '97*, pages 718–724, 1997.
19. M. Yoshida and S. Obana. On the (in)efficiency of non-interactive secure multiparty computation. In *ICISC 2015*, volume 9558 of *LNCS*, pages 185–193, 2015.