Unconditionally Secure Multiparty Computation for Symmetric Functions with Low Bottleneck Complexity

Reo Eriguchi

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

Abstract. Bottleneck complexity is an efficiency measure of secure multiparty computation (MPC) introduced by Boyle et al. (ICALP 2018) to achieve load-balancing. Roughly speaking, it is defined as the maximum communication complexity required by any player within the protocol execution. Since it was shown to be impossible to achieve sublinear bottleneck complexity in the number of players n for all functions, a prior work constructed MPC protocols with low bottleneck complexity for specific functions. However, the previous protocol for symmetric functions needs to assume a computational primitive of garbled circuits and its unconditionally secure variant has exponentially large bottleneck complexity in the depth of an arithmetic formula computing the function, which limits the class of symmetric functions the protocol can compute with sublinear bottleneck complexity in n. In this work, we make the following contributions to unconditionally secure MPC protocols for symmetric functions with sublinear bottleneck complexity in n.

- We propose for the first time unconditionally secure MPC protocols computing *any* symmetric function with sublinear bottleneck complexity in n. Technically, our first protocol is inspired by the one-time truth-table protocol by Ishai et al. (TCC 2013) but our second and third protocols use a novel technique to express the one-time truth-table as an array of two or higher dimensions and achieve better trade-offs.
- We propose an unconditionally secure protocol tailored to the AND function with lower bottleneck complexity. It avoids pseudorandom functions used by the previous protocol for the AND function, preserving bottleneck complexity up to a logarithmic factor in n.
- By combining our protocol for the AND function with Bloom filters, we construct an unconditionally secure protocol for private set intersection (PSI), which computes the intersection of players' private sets. This is the first PSI protocol with sublinear bottleneck complexity in n and to the best of our knowledge, there has been no such protocol even under cryptographic assumptions.

Keywords: Secure multiparty computation · Bottleneck complexity · Unconditional security

1 Introduction

Secure multiparty computation (MPC) [62] is a fundamental cryptographic primitive which enables n players to jointly compute a function $f(x_1, \ldots, x_n)$ without revealing any additional information on their private inputs x_i. Communication complexity, which counts the total number of bits transmitted between players, is considered as the most fundamental metric to measure the efficiency of MPC protocols. A number of works have made significant progresses to minimize communication complexity for various useful functions (e.g., [28, 7, 14, 22, 23, 16, 32, 45, 30]).

However, in practical applications where lightweight devices perform MPC via peer-to-peer communication, the per-party communication cost is a more effective measure than the total cost. For example, consider secure computation on a star interaction pattern, in which a central player interacts with all the other players and computes an output. Then, while total communication cost is possibly scalable (i.e., $O(n)$), the central player must bear communication proportional to the total number of players. In large-scale MPC, these costs quickly become prohibitive. To address these concerns, Boyle et al. [11] introduced a different important efficiency measure, called *bottleneck complexity*. Roughly speaking, the bottleneck complexity of an MPC protocol is defined as the maximum communication required by any player during the execution of the protocol.

To make protocols useful in applications to large-scale secure computation, we aim at designing MPC protocols with sublinear bottleneck complexity in the number of players n. On the negative side, Boyle et al. [11] showed that it is impossible to achieve sublinear bottleneck complexity for all functions — even without any security considerations. On the positive side, they proposed a generic transformation from any (possibly insecure) protocol computing a function f to a secure MPC protocol for f preserving bottleneck complexity (up to polynomial factors in a security parameter). Their results reduce in some sense the above goal to constructing protocols with sublinear bottleneck complexity without any privacy requirements, which is a purely complexity-theoretic question. However, a main drawback of their compiler is that it needs to use fully homomorphic encryption, which can only be instantiated from narrow cryptographic assumptions [26, 60]. Recently, Orlandi, Ravi and Scholl [52] constructed MPC protocols with sublinear bottleneck complexity for specific functions from weaker assumptions of one-way functions or linearly homomorphic encryption. It still remains open whether we can construct *unconditionally secure* MPC protocols with sublinear bottleneck complexity.

1.1 Our Results

In this paper, we propose for the first time unconditionally secure MPC protocols for symmetric functions with sublinear bottleneck complexity in the number of players[1]. We also propose an MPC protocol with lower bottleneck complexity tailored to the functionality of checking if the sum of inputs is zero. As an application, we construct an unconditionally secure protocol for *private set intersection* (PSI), which computes the intersection of players' private sets. This is the first PSI protocol with sublinear bottleneck complexity and to the best of our knowledge, there has been no such protocol even under cryptographic assumptions. Following [52], we assume semi-honest adversaries, who do not deviate from protocols, and the preprocessing model, in which players receive in advance input-independent correlated randomness from a trusted third party. In what follows, we discuss our contributions in more detail.

Protocols for General Symmetric Functions. Orlandi et al. [52] constructed an MPC protocol for a symmetric function $h : \{0,1\}^n \to \{0,1\}$ assuming a garbled circuit [63]. Although their original protocol was computationally secure, it can be made unconditionally secure by replacing the underlying garbled circuit with an information-theoretic one, which is also known as randomized encoding [41][2]. However, the unconditionally secure variant has exponentially large bottleneck complexity in the depth of an arithmetic formula computing the function $f : \{0,1,2,\ldots,n\} \to \{0,1\}$ such that $f(\sum_{i\in[n]} x_i) = h(x_1,\ldots,x_n)$ for all $(x_1,\ldots,x_n) \in \{0,1\}^n$. To achieve bottleneck complexity $O(n^{1-\epsilon})$ for a constant $\epsilon > 0$, the unconditionally secure protocol needs to assume that the related function f is represented by an arithmetic formula of depth $(1-\epsilon)\log n$. Note that such symmetric functions only account for $o(1)$ fraction of all symmetric functions (see Appendix B). We propose three kinds of unconditionally secure protocols with sublinear bottleneck complexity for *any* symmetric function. There are trade-offs between online bottleneck complexity, offline bottleneck complexity, i.e., the amount of correlated randomness per party, and the privacy threshold (see Table 1). The first protocol has online bottleneck complexity $O(\log n)$ and offline bottleneck complexity $O(n)$. The second is more balanced and its online and offline bottleneck complexities are both $O(\sqrt{n})$. The third protocol has lower bottleneck complexity $O(n^{1/d}\log n)$ for any constant d but is only secure against adversaries corrupting less than $n/(d-1)$ players. The numbers of rounds of our protocols are $O(\log n)$ while that of [52] is $O(n)$. Technically, our first protocol is inspired by the one-time truth-table protocol [42] and our second and third protocol use a novel technique to express the truth-table of f as an array of two or higher dimensions (see Section 2 for technical details).

Protocol for Checking Equality to Zero. We propose an unconditionally secure MPC protocol realizing the functionality of checking if the sum of players'

[1] An independent work [44] show an unconditionally secure MPC protocol for symmetric functions but our second and third protocols achieve better offline bottleneck complexity than [44].

[2] Since we have not seen any work that states this fact explicitly, we show a self-contained exposition in Appendix B.

Table 1. Comparison of unconditionally secure MPC protocols for a symmetric function h with sublinear bottleneck complexity in the number of players n.

Reference	Condition on h	Bottleneck complexity Online	 Offline	Corruption
[52]	$D_h \leq (1-\epsilon)\log n$	$O(n^{1-\epsilon})$	$O(n^{1-\epsilon})$	$t < n$
Ours (Theorem 1)	Unnecessary	$O(\log n)$	$O(n)$	$t < n$
Ours (Theorem 2)	Unnecessary	$O(\sqrt{n})$	$O(\sqrt{n})$	$t < n$
Ours (Theorem 3)	Unnecessary	$O(n^{1/d}\log n)$	$O(n^{1/d}\log n)$	$t < n/(d-1)$

The offline bottleneck complexity means the amount of correlated randomness per party and t is the maximum number of players corrupted by adversaries. Let ϵ be a constant and D_h be the minimum depth of arithmetic formulas computing the unique function f such that $f(\sum_{i\in[n]} x_i) = h(x_1,\ldots,x_n)$ for all $(x_1,\ldots,x_n) \in \{0,1\}^n$.

inputs is equal to zero over a finite field $\mathbb{F}$. Since the functionality is symmetric, we can apply the above protocols for general symmetric functions. Our tailored protocol achieves lower bottleneck complexity $O(\max\{\lambda, \log|\mathbb{F}|\})$, where λ is a security parameter. The functionality is a generalization of the AND function if we choose a field whose characteristic is larger than n. As a comparison, the AND protocol in [52] is based on pseudorandom functions and hence only computationally secure. Our protocol avoids their use of pseudorandom functions while preserving bottleneck complexity up to a logarithmic factor in n.

Application to Private Set Intersection. By combining our protocol for checking equality to zero with Bloom filters [9, 10], we obtain an unconditionally secure PSI protocol, that is, an MPC protocol for computing $X_1 \cap \cdots \cap X_n$ from private sets $X_1,\ldots,X_n \subseteq U$ each of size at most s. The offline bottleneck complexity is $O_\lambda(s^2 \log n)$ and the online bottleneck complexity is $O_\lambda(s^2\log n + s\log|U|)$, where we omit a polynomial factor in λ. Note that the bottleneck complexity is sublinear in n. To the best of our knowledge, there has been no such PSI protocol even under cryptographic assumptions (see Section 1.2 for a more detailed comparison). The round complexity of our protocol is $O(\log n)$.

1.2 Related Work

General MPC. Since the introduction of MPC [62], a rich line of works studied communication complexity in various settings and showed feasibility results and optimizations, e.g., [28, 7, 14, 55, 18, 37, 36, 22, 4, 21, 23, 8, 38, 16, 31, 32, 45, 30]. However, protocols in all of the above works require *full interaction* among players, that is, each player may send messages to all the other players in each round of interaction. This feature necessarily results in high bottleneck complexity $\Omega(n)$.

MPC with Restricted Interaction Patterns. Halevi, Lindell and Pinkas [34] initiated the study of MPC which restricts interaction among players. Halevi et al. [33] formalized the notion in the more general setting by representing an interaction pattern as a directed acyclic graph. MPC protocols on a star-based interaction were proposed for general tasks and for specific tasks including symmetric functions [24, 40, 5]. As we mentioned above, protocols on a star-based interaction require a central player to bear communication proportional to n, which results in $\Omega(n)$ bottleneck complexity. Halevi et al. [33] also studied a chain-based interaction, in which players interact over a simple directed path traversing all players. Protocols on a chain-based interaction possibly achieve low bottleneck complexity since each player communicates with at most two players. However, since the last player on the chain is allowed to evaluate the function on every possible input of his choice, the constructions in [33] cannot achieve the standard security of MPC, which requires that corrupted players learn nothing but the output. The weaker security notion of MPC with restricted interaction is formalized as residual security [34, 33].

Private Set Intersection. Private set intersection (PSI) has a wide variety of real-world applications as pointed out in [47]. Although many PSI protocols were constructed unconditionally or based on cryptographic assumptions, to the best of our knowledge, there has been no PSI protocol that achieves sublinear bottleneck complexity in n. Indeed, the protocols in [46, 49, 56, 57, 53, 54, 15, 47, 39, 27, 51] require full interaction and those in [25, 50, 35, 13, 2, 61, 1, 6] assume a star interaction pattern. Thus the bottleneck complexity of these protocols must be at least linear in n. An exception is the protocol in [20], which utilizes a "round table" structure where players are supposed to be nodes in a ring network and each player only communicates with the consecutive players around the table. The bottleneck complexity is thus possibly independent of n. However, their protocol outputs an incorrect intersection with constant probability. Essentially, their protocol securely finds elements $x \in X_n$ on which a polynomial $p(T) := r \cdot \sum_{i\in[n]} f_i(T)$ vanishes, where r is a random element unknown to any player and each $f_i(T)$ is a polynomial which vanishes exactly when evaluated on the elements of the i-th player's input set X_i.[3] It is true that $p(x) = 0$ for all $x \in X_1 \cap \cdots \cap X_n$ but the set of all the roots of $p(T)$ may include elements outside $X_1 \cap \cdots \cap X_n$ since $\sum_{i\in[n]} f_i(x) = 0$ even if x is not a common root of the f_i's. The authors of [20] do not show how to make the probability of this event negligible. This is why we do not consider their protocol for our comparison. Finally, we note that the previous state-of-the-art protocols have better dependency on the maximum size s of players' input sets than ours. Indeed, the best-known communication complexity (e.g., [1]) is linear in s while ours is quadratic in s.

1.3 Difference from the Conference Paper

The preliminary version of this paper appears in the proceedings of ASIACRYPT 2023. In this full version, we replace the original protocols for sum with a more

[3] We here assume that there is an injective map from the universe U containing all the X_i's to some finite field.

round-efficient protocol. As a result, all of our protocols have round complexity $O(\log n)$.

2 Technical Overview

In this section, we provide an overview of our techniques. We give more detailed descriptions and security proofs in the following sections. We call MPC protocols with sublinear bottleneck complexity in the number of players *BC-efficient*.

2.1 BC-Efficient Protocols for General Symmetric Functions

First Protocol. Let $h : \{0,1\}^n \to \{0,1\}$ be a symmetric function. Since the value of h depends only on the sum $\sum_{i\in[n]} x_i$, there is the unique function $f : \{0,1,\ldots,n\} \to \{0,1\}$ such that $h(x_1,\ldots,x_n) = f(\sum_{i\in[n]} x_i)$. Our starting point is the protocol in [52]: In the setup, players receive an additive sharing $(r_i)_{i\in[n]}$ of a random secret r, and a garbled circuit of a circuit computing $g(y) := f(y-r)$ from y, where r is hard-coded. In the online phase, each player broadcasts $y_i = x_i + r_i$, computes $y = \sum_{i\in[n]} y_i$, and evaluates the garbled circuit on y in a threshold manner. Due to the mask r, the value y is random and independent of $\sum_{i\in[n]} x_i$, and the security of the garbled circuit ensures that the evaluation process reveals nothing but the output $g(y) = h(x_1,\ldots,x_n)$. However, to achieve unconditional security, it is necessary to replace the garbled circuit with an information-theoretic one, which results in exponentially large bottleneck complexity in the depth of an arithmetic formula computing g.

Instead, our first protocol uses the idea of one-time truth-tables (OTTT) [42]. Roughly speaking, in an OTTT protocol, players receive in the setup an additive sharing of the truth-table of a (not necessarily symmetric) function h permuted with a random shift $\mathbf{r} = (r_1,\ldots,r_n)$. In the online phase, each player broadcasts $y_i = x_i + r_i$ and sets $\mathbf{y} = (y_1,\ldots,y_n)$. Players recover the $\mathbf{y}$-th component of the permuted truth-table, which is equal to $h(\mathbf{y}-\mathbf{r}) = h(x_1,\ldots,x_n)$. We adapt the OTTT protocol to the setting of h being symmetric. In this case, it is sufficient to prepare an additive sharing of a shifted version of the truth-table of the related function $f : \{0,1,\ldots,n\} \to \{0,1\}$. By receiving an additive sharing of the random shift $r \in \{0,1,\ldots,n\}$ in the setup, players can open $y = \sum_{i\in[n]} x_i + r$ and the y-th component of the table, which is $f(y-r) = f(\sum_{i\in[n]} x_i)$.

Now, remaining problems are how to let players open secrets and broadcast messages in a BC-efficient way. The protocol in [52] used a round-table structure and let each player add his share to the message from the previous player and send the result to the next player around the table. However, the round complexity of this protocol is $O(n)$. We show a more round-efficient protocol that opens secrets with $O(\log n)$ rounds. Assume that players are represented as nodes (including leafs) of a binary tree of depth $O(\log n)$. In the first round, players corresponding to leafs send their inputs to their parents. In each round, players compute the sum of elements sent by their children, add their input to the sum, and send it to their parents. Finally, the player corresponding to the root obtains the aggregation of all inputs. In [52], the broadcast functionality was also realized by a similar round-table structure, in which each player simply relays a message from the previous to the next one. Similarly, we reduce it to $O(\log n)$ by using a tree-based representation: Represent the set of players by a binary tree whose height is $O(\log n)$ and root is the player who broadcasts a message. We let each player relay the message from his parent to his children in the tree.

Second Protocol. Although the first protocol has polylogarithmic online bottleneck complexity in the number of players n, a drawback is that the offline bottleneck complexity is linear in n. We show a novel technique to extend the above OTTT-based approach and obtain a protocol that balances offline and online bottleneck complexities. The core idea of the first protocol is that players securely obtain the component of the truth table $\mathbf{T} = [f(0), f(1), \ldots, f(n)]$ at position $s = \sum_{i\in[n]} x_i$. This can also be interpreted as securely computing the inner product $\langle \mathbf{T}, \mathbf{e}_s \rangle$, where $\mathbf{e}_s$ is the unit vector whose entry is 1 at position s and 0 otherwise (we identify the set indexing entries with $\{0, 1, \ldots, n\}$).

The key idea of our second protocol is to represent vectors $\mathbf{T}$ and $\mathbf{e}_s$ as arrays of two dimension. Assume that $n+1$ is the product of two distinct primes p, q of almost equal size $O(\sqrt{n})$. We then have a one-to-one correspondence ϕ between $\mathbb{Z}_{n+1} := \{0, 1, \ldots, n\}$ and $\mathbb{Z}_p \times \mathbb{Z}_q$ from the Chinese remainder theorem. The truth-table $\mathbf{T}$ then corresponds to a matrix $\mathbf{M} \in \{0,1\}^{p\times q}$ whose (k_1, k_2)-th entry is $f(k)$, where $(k_1, k_2) = \phi(k)$ (we identify the set indexing rows and columns with $\mathbb{Z}_p$ and $\mathbb{Z}_q$, respectively). Furthermore, the equation $\langle \mathbf{T}, \mathbf{e}_s \rangle = f(s)$ is rewritten as

$$\langle \mathbf{e}_{s_1}, \mathbf{M} \cdot \mathbf{e}_{s_2} \rangle = f(s), \text{ where } (s_1, s_2) = \phi(s). \tag{1}$$

Importantly, since $\mathbf{M}$ is public, the computation of $\mathbf{M} \cdot \mathbf{e}_{s_2}$ can be locally done and hence the computation players need to interactively perform is only the inner product of vectors of dimension $O(\sqrt{n})$, instead of $O(n)$.

Specifically, we cannot compute the inner product (1) for $(s_1, s_2) = \phi(s)$ directly since we should not reveal s and can only open $y = s - r$ with a random mask r. To unmask s from y, we give players vectors $\mathbf{e}_{u_1} \in \{0,1\}^p$ and $\mathbf{e}_{u_2} \in \{0,1\}^q$ as correlated randomness, where $(u_1, u_2) = \phi(r)$. Now, as in our first protocol, players open $y = \sum_{i\in[n]} x_i - r$, express it as $\phi(y) = (z_1, z_2)$, and permutes $\mathbf{e}_{u_1}$ and $\mathbf{e}_{u_2}$ with shifts z_1 and z_2, respectively. They obtain $\mathbf{e}_{u_1+z_1} = \mathbf{e}_{s_1}$ and $\mathbf{e}_{u_2+z_2} = \mathbf{e}_{s_2}$, from which $f(s) = h(x_1, \ldots, x_n)$ can be computed via Eq. (1). However, distributing $(\mathbf{e}_{u_1}, \mathbf{e}_{u_2})$ itself reveals r. We instead give players additive shares for $\mathbf{e}_{u_1}$ and $\mathbf{e}_{u_2}$ as correlated randomness. Since permuting vectors with a public shift and multiplying vectors by a constant matrix are linear operations, players can obtain additive shares for $\mathbf{e}_{s_1}$ and $\mathbf{M} \cdot \mathbf{e}_{s_2}$.

A remaining problem is how to compute the inner product of $\mathbf{e}_{s_1}$ and $\mathbf{M} \cdot \mathbf{e}_{s_2}$ in a secret-shared form. Our key observation is that the multiplication protocol based on Beaver triples [3] is BC-efficient. Indeed, assume that players have additive shares of (a, b, c), where a, b are random secrets and $c = ab$. They can compute an additive sharing of xy from those for x and y as follows: (1) players open $u := x - a$ and $v := y - b$; (2) they compute $[\![xy]\!] = uv + v[\![a]\!] + u[\![b]\!] + [\![c]\!]$, where $[\![w]\!]$ means an additive sharing for w. Since the functionality of opening secrets can be realized in a BC-efficient way, this multiplication protocol is also BC-efficient. Therefore, if we additionally distribute $O(\sqrt{n})$ Beaver triples in the setup, we can securely compute $\langle \mathbf{e}_{s_1}, \mathbf{M} \cdot \mathbf{e}_{s_2} \rangle = f(s)$ in a secret-shared form. Although we assume for simplicity that $n + 1$ is the product of two primes of almost equal size, it is straightforward to extend it the general case since we can choose primes such that $\sqrt{n} < p < 2\sqrt{n} < q < 4\sqrt{n}$ thanks to Bertrand's postulate [59].

Third Protocol. In our second protocol, we express the truth-table $\mathbf{T}$ of the function f by a two-dimensional array. We further extend the technique and use an array of higher dimension $d \geq 2$. For simplicity, assume that $n + 1$ is the product of d distinct primes $p_1, p_2, \ldots, p_d$ of almost equal size $O(n^{1/d})$. The general case can be dealt with similarly. From the Chinese remainder theorem, we have a one-to-one correspondence ϕ from $\{0, 1, \ldots, n\}$ to $\mathbb{Z}_{p_1} \times \mathbb{Z}_{p_2} \times \cdots \times \mathbb{Z}_{p_d}$. We can then equivalently express the truth-table $\mathbf{T}$ of f by a p_1-by-q matrix $\mathbf{M}$ over any field $\mathbb{F}$, where $q = p_2 \cdots p_d$. Indeed, fixing a one-to-one correspondence between $k \in \mathbb{Z}_q$ and $(s_2, \ldots, s_d) \in \mathbb{Z}_{p_2} \times \cdots \times \mathbb{Z}_{p_d}$, we embed every value $f(s)$ into the (s_1, k)-th entry of $\mathbf{M}$ for $(s_1, s_2, \ldots, s_d) = \phi(s)$. In other words, we can choose a matrix $\mathbf{M}$ such that the equation $\langle \mathbf{T}, \mathbf{e}_s \rangle = f(s)$ is rewritten as

$$\langle \mathbf{e}_{s_1}, \mathbf{M} \cdot (\mathbf{e}_{s_2} \otimes \cdots \otimes \mathbf{e}_{s_d}) \rangle = f(s), \tag{2}$$

where $(s_1, s_2, \ldots, s_d) = \phi(s)$, $\mathbf{e}_{s_j} \in \mathbb{F}^{p_j}$ is the unit vector whose entry is 1 at position s_j, and $\mathbf{a} \otimes \mathbf{b}$ denotes the Kronecker product of $\mathbf{a}$ and $\mathbf{b}$. The problem is now reduced to securely computing the inner product (2). Again, we should not compute (2) for the sum $s = \sum_{i \in [n]} x_i$ itself since we can only open a masked value $y = s - r$. To remove the mask r, we try to distribute vectors $\mathbf{e}_{u_1}, \mathbf{e}_{u_2}, \ldots, \mathbf{e}_{u_d}$ for $(u_1, u_2, \ldots, u_d) = \phi(r)$. Since this trivially reveals r, we give players shares for them. A major difference from the second protocol is that we use the Shamir secret sharing scheme over $\mathbb{F}$ with a threshold $t < n/(d-1)$ [58] instead of the additive one. By opening $\phi(y) = (z_1, z_2, \ldots, z_d)$ and permuting each share vector with a shift z_j, players can compute a Shamir sharing of $\mathbf{e}_{s_1}, \mathbf{e}_{s_2}, \ldots, \mathbf{e}_{s_d}$ without recovering s. Importantly, the homomorphic property of the Shamir scheme enables players to locally compute a Shamir sharing of $\mathbf{e}_{s_2} \otimes \cdots \otimes \mathbf{e}_{s_d}$ with a threshold $(d-1)t$ since the degree of a function to be evaluated is at most $d - 1$. Furthermore, since $(d-1)t < n$, the shares can be locally converted to an additive sharing of $\mathbf{e}_{s_2} \otimes \cdots \otimes \mathbf{e}_{s_d}$ and to an additive sharing of $\mathbf{M} \cdot (\mathbf{e}_{s_2} \otimes \cdots \otimes \mathbf{e}_{s_d})$. Finally, using $p_1 = O(n^{1/d})$ Beaver triples, players interactively computes the inner product $\langle \mathbf{e}_1, \mathbf{M} \cdot (\mathbf{e}_{s_2} \otimes \cdots \otimes \mathbf{e}_{s_d}) \rangle$ in a secret-shared form in a BC-efficient way. For any constant d, the offline and online bottleneck complexities are both $O(n^{1/d})$ field elements. Since the Shamir scheme needs to assume $|\mathbb{F}| > n$, they are $O(n^{1/d} \log n)$ in bits.

2.2 BC-Efficient Protocol for Checking Equality to Zero

We propose an unconditionally secure BC-efficient protocol for checking if the sum of players' inputs is zero over a finite field $\mathbb{F}$. Since the functionality is symmetric, we can apply our protocols for general symmetric functions. Our tailored protocol shown below achieves lower bottleneck complexity $O(\max\{\lambda, \log |\mathbb{F}|\})$, where λ is a security parameter.

First, we show a special case of our protocol, which computes the OR function of players' inputs. Our starting point is the previous protocol for the OR function in [52][4]. In the protocol, players receive an additive sharing $(r_i)_{i\in[n]}$ of a random secret r in the setup. The i-th player sets $y_i \leftarrow r_i$ if he has $x_i = 0$, and otherwise he chooses y_i uniformly at random. Players then open $y = \sum_{i\in[n]} y_i$, which is equal to r if and (essentially) only if $\mathrm{OR}(x_1, \dots, x_n) = 0$. Note that r should not be revealed since otherwise an adversary can learn the OR of the honest players' inputs by subtracting the inputs and correlated randomness of the corrupted players. It is therefore necessary to check the equality $y = r$ without revealing r. In [52], this is done by using a pseudorandom function (PRF): Players apply the PRF in a nested manner starting from y by using their private keys, and check the final value is equal to the value computed in the same way except that it starts from r. To obtain an unconditionally secure protocol, we propose a different method to check the equality $y = r$. Our key observation is that for two random secrets a and b, the equation $ay + b = ar + b$ is equivalent to $y = r$ except with a small probability that $a = 0$, and the view $(ay + b, ar + b)$ reveals nothing but whether the equality $y = r$ holds in the information-theoretic sense. In our protocol, players receive additive shares $(a_i)_{i\in[n]}$ and $(b_i)_{i\in[n]}$ for a and b, and $r' := ar + b$ in the setup. In the online phase, after opening y, each player computes $y'_i = a_i y + b_i$ and obtains $y' = \sum_{i\in[n]} y'_i = ay + b$. Players then check $y' = r'$ and if so, output 0. The probability of failure can be made negligible in λ if we choose r, a, b and shares for them from sufficiently large sets.

Finally, it is straightforward to extend the above protocol to a protocol for checking whether $\sum_{i\in[n]} x_i = 0$ holds. Let every player compute $y_i = x_i + r_i$ and open $y = \sum_{i\in[n]} x_i + r$. Using the above technique, players can check the equality $y = r$, i.e., $\sum_{i\in[n]} x_i = 0$, without learning any additional information.

2.3 BC-Efficient PSI Protocol

We combine our protocol for checking equality to zero with Bloom filters [9] and construct an unconditionally secure PSI protocol with sublinear bottleneck complexity in n. First, we show a simple protocol whose bottleneck complexity is sublinear in n but linear in the cardinality of the universe $U = \{1, 2, \dots, N\}$ containing all the input sets. In the protocol, each player encodes his input set $X_i \subseteq U$ as the characteristic vector $\mathbf{B}_i \in \{0,1\}^N$ of its complement, i.e., the vector whose entry at position j is 1 if and only if $j \notin X_i$. They compute an

[4] The original protocol in [52] computes the AND function. The functionalities are equivalent since $\mathrm{OR}(x_1, \dots, x_n) = 1 - \mathrm{AND}(1 - x_1, \dots, 1 - x_n)$.

additive sharing of the sum $\mathbf{V} = \sum_{i\in[n]} \mathbf{B}_i$ over a finite field. (We suppose that the characteristic of the field is so large that no wrap-around occurs.) This can be done by giving players an additive sharing $(\mathbf{u}_i)_{i\in[n]}$ of the N-dimensional zero vector in the setup and letting them compute $\mathbf{B}_i + \mathbf{u}_i$. Observe that $x \in X_1 \cap \cdots \cap X_n$ if and only if the entry $\mathbf{V}[x]$ at position x is 0. For each element x of the input set of a designated player, players compute the inner product of $\mathbf{V}$ and $\mathbf{e}_x$ in a secret-shared form using N Beaver triples. While the inner products themselves reveal additional information on elements outside $X_1 \cap \cdots \cap X_n$, players use our BC-efficient protocol for checking equality to zero and learn whether $\langle \mathbf{V}, \mathbf{e}_x \rangle = 0$ and nothing else. Finally, $X_1 \cap \cdots \cap X_n$ is securely obtained if the designated player broadcasts the x's for which $\langle \mathbf{V}, \mathbf{e}_x \rangle = 0$ holds.

Since U has typically large size, we aim to achieve better dependency on $|U|$ by making use of Bloom filters, which provide compact encodings of sets. A Bloom filter encodes a set X of elements as an m-bit vector with respect to λ randomly selected hash functions $H_1, \ldots, H_\lambda$. In a Bloom filter for X, all entries are initialized to 0. To insert an element $x \in X$ to the Bloom filter, the entries at positions $H_1(x), \ldots, H_\lambda(x)$ are all set to 1. We can test whether $y \in X$ by checking if the entries at positions $H_1(y), \ldots, H_\lambda(y)$ are all 1. There is a possibility of false positives but its probability can be made negligible in λ if we choose $m = \Theta(s\lambda)$, where s is the maximum size of input sets [10]. Although our PSI protocol bears some similarities with the previous protocols based on Bloom filters [50, 1], their protocols are not BC-efficient and assume a computational primitive of homomorphic threshold public-key encryption. In our protocol, each player computes a Bloom filter $\mathbf{BF}(X_i)$ of his input set X_i and inverts it, i.e., $\mathbf{B}_i := \mathbf{1}_m - \mathbf{BF}(X_i)$, where $\mathbf{1}_m$ is the m-dimensional all-ones vector. As above, players compute an additive sharing of $\mathbf{V} = \sum_{i\in[n]} \mathbf{B}_i$ using additive sharings of zeros. For any $x \in X_1 \cap \cdots \cap X_n$, the entries of $\mathbf{B}_i$ at positions $H_1(x), \ldots, H_\lambda(x)$ are all zero and hence so are the corresponding entries of $\mathbf{V}$. The converse is not always true but with overwhelming probability, for $x \notin X_1 \cap \cdots \cap X_n$, at least one entry of $\mathbf{V}$ at positions $H_1(x), \ldots, H_\lambda(x)$ is non-zero due to the property of Bloom filters. Based on that observation, for each element x of the input set of a designated player, we let players compute the inner product of $\mathbf{V}$ and a Bloom filter $\mathbf{BF}(\{x\})$ of the singleton $\{x\}$ in a secret-shared form. The rest is similar to the above: Players check whether $\langle \mathbf{V}, \mathbf{BF}(\{x\}) \rangle = 0$ based on our protocol for checking equality to zero, and the designated player broadcasts the x's for which the equality holds. If we implement the functionality of opening secrets based on a round-table structure, the bottleneck complexity is $O(sm) = O_\lambda(s^2)$ field elements plus $O(s \log |U|)$ bits since we have to compute s inner products of m-dimensional vectors and $m = \Theta(s\lambda)$. Since no wrap-around occurs if the characteristic is larger than $n\lambda$, the bottleneck complexity is $O_\lambda(s^2 \log n + s \log |U|)$ in bits, omitting a polynomial factor in λ.

3 Preliminaries

Notations. For $m \in \mathbb{N}$, define $[m] = \{1, \ldots, m\}$ and $[0..m] = [m] \cup \{0\}$. Define $\mathbb{Z}_m$ as the ring of integers modulo m. We identify $\mathbb{Z}_m$ (as a set) with $\{z \in \mathbb{Z} : 0 \leq z \leq m-1\}$. We denote the set of all subsets of X by 2^X. For a subset X of a set Y, we define $Y \setminus X = \{y \in Y : y \notin X\}$ and simply denote it by $\overline{X}$ if Y is clear from the context. We write $u \leftarrow_{\$} Y$ if u is chosen uniformly at random from a set Y. We call a function $f : \mathbb{N} \ni \lambda \mapsto f(\lambda) \in \mathbb{R}$ *negligible* and denote it by $\mathsf{negl}(\lambda)$ if for any $c > 0$, there exists $\lambda_0 \in \mathbb{N}$ such that $0 \leq f(\lambda) < \lambda^{-c}$ for any $\lambda > \lambda_0$. We call f *polynomial* and denote it by $\mathsf{poly}(\lambda)$ if there exists $c > 0$ and $\lambda_0 \in \mathbb{N}$ such that $0 \leq f(\lambda) \leq \lambda^c$ for any $\lambda > \lambda_0$. For two random variables with range U, we define the statistical distance $\mathrm{SD}(X, Y)$ between X and Y as $\mathrm{SD}(X, Y) = (1/2) \sum_{u \in U} |\Pr[X = u] - \Pr[Y = u]|$. For two sequences $X = (X_\lambda)_{\lambda \in \mathbb{N}}$, $Y = (Y_\lambda)_{\lambda \in \mathbb{N}}$ of random variables, we write $X \approx Y$ if a function $f : \lambda \mapsto \mathrm{SD}(X_\lambda, Y_\lambda)$ is negligible in λ. By default, the i-th element of a vector $\mathbf{u}$ is denoted by u_i or $\mathbf{u}[i]$. For a vector $\mathbf{s} = (s_i)_{i \in \mathbb{Z}_m} \in X^m$ and $r \in \mathbb{Z}_m$, we define $\mathsf{Shift}_r(\mathbf{s})$ as the vector obtained by shifting elements by r. Formally, $\mathsf{Shift}_r(\mathbf{s}) = (t_i)_{i \in \mathbb{Z}_m}$ is defined by $t_i = s_{(i-r) \bmod m}$ for all $i \in \mathbb{Z}_m$. Let $\mathbf{0}_m$ be the zero vector of dimension m and $\mathbf{1}_m$ be the all-ones vector of dimension m. We simply write $\mathbf{0}$ or $\mathbf{1}$ if the dimension is clear from the context. Let $\mathbf{I}_m$ denote the m-by-m identity matrix and $\mathbf{e}_i$ denote the i-th unit vector. For two vectors $\mathbf{u}, \mathbf{v}$ over a ring, we define the standard inner product of $\mathbf{u}$ and $\mathbf{v}$ as $\langle \mathbf{u}, \mathbf{v} \rangle = \sum_i \mathbf{u}[i]\mathbf{v}[i]$. For a tuple of m polynomials $\boldsymbol{\varphi} = (\varphi_j)_{j \in [m]}$ over $\mathbb{F}$ and $\alpha \in \mathbb{F}$, we write $\boldsymbol{\varphi}(\alpha) = (\varphi_j(\alpha))_{j \in [m]} \in \mathbb{F}^m$. Let g be a deterministic function on D^n, where D is a set. We denote by $\mathcal{F}_g$ an n-input/n-output functionality that on input $\mathbf{x} \in D^n$, outputs $\mathcal{F}_g(\mathbf{x}) = (g(\mathbf{x}), \ldots, g(\mathbf{x}))$. We call a function $h : \{0,1\}^n \to \{0,1\}$ *symmetric* if $h(x_{\sigma(1)}, \ldots, x_{\sigma(n)}) = h(x_1, \ldots, x_n)$ for input $(x_1, \ldots, x_n) \in \{0,1\}^n$ and any permutation σ on $[n]$. The value of a symmetric function h is determined only by the Hamming weight w of the input, i.e., $w := |\{i \in [n] : x_i = 1\}| = \sum_{i \in [n]} x_i$. Thus, there is the unique function $f : [0..n] \to \{0,1\}$ such that $f(\sum_{i \in [n]} x_i) = h(x_1, \ldots, x_n)$ for all $(x_1, \ldots, x_n) \in \{0,1\}^n$.

3.1 Secure Multiparty Computation

Let n be a polynomial in a security parameter λ. We denote the set of n players by $\{\mathsf{P}_1, \ldots, \mathsf{P}_n\}$, where P_i is the i-th player. Assume that each player P_i has a private input x_i from a finite set D. Let g be a deterministic function on D^n. Consider the functionality $\mathcal{F}_g(\mathbf{x}) = (g(\mathbf{x}), \ldots, g(\mathbf{x}))$ for $\mathbf{x} = (x_1, \ldots, x_n)$. Let Π be a protocol between n players. We assume the *preprocessing model.* That is, a protocol includes a joint distribution $\mathcal{D}$ over the Cartesian product $R_1 \times \cdots \times R_n$ of n sets, and each player P_i receives $r_i \in R_i$ before he decides his input, where $(r_1, \ldots, r_n)$ is sampled from $R_1 \times \cdots \times R_n$ according to $\mathcal{D}$. We assume computationally unbounded adversaries who passively corrupt up to t players. (We do not consider active adversaries whose corrupted players deviate from protocols arbitrarily.) Let $\mathsf{View}_{\Pi,i}(\mathbf{x})$ denote the view of P_i at the end of the protocol execution on input $\mathbf{x}$, and let $\mathsf{Output}_{\Pi,i}(\mathbf{x})$ be the output of P_i. For a security parameter λ, we say that Π is a *t-secure MPC protocol for $\mathcal{F}_g$* if

Correctness. For any input $\mathbf{x}$, $\Pr\left[\exists i \in [n] : \mathsf{Output}_{\Pi,i}(\mathbf{x}) \neq g(\mathbf{x})\right] = \mathsf{negl}(\lambda)$;

Privacy. For any set $T \subseteq [n]$ of size at most t and any pair of inputs $\mathbf{x} = (x_i)_{i\in[n]}$, $\mathbf{w} = (w_i)_{i\in[n]}$ such that $(x_i)_{i\in T} = (w_i)_{i\in T}$ and $g(\mathbf{x}) = g(\mathbf{w})$, it holds that $\{(\mathsf{View}_{\Pi,i}(\mathbf{x}))_{i\in T}\}_{\lambda\in\mathbb{N}} \approx \{(\mathsf{View}_{\Pi,i}(\mathbf{w}))_{i\in T}\}_{\lambda\in\mathbb{N}}$.

We simply say that Π is *fully secure* if it is $(n-1)$-secure.

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

We defer the simulation-based definition of secure MPC for n-input/n-output randomized functionalities to Appendix A. We also argue in Appendix A that a set of protocols in the paper can be composed concurrently.

3.2 Basic Algorithms

Let $\mathbb{G}$ be an abelian group (e.g., a finite field or a ring of integers modulo m). Define $\mathsf{Additive}_{\mathbb{G}}(s)$ as an algorithm to generate additive shares over $\mathbb{G}$ for a secret $s \in \mathbb{G}$. Formally, on input $s \in \mathbb{G}$, $\mathsf{Additive}_{\mathbb{G}}(s)$ chooses $(s_1, \ldots, s_n) \in \mathbb{G}^n$ uniformly at random conditioned on $s = \sum_{i\in[n]} s_i$, and outputs it. For a vector $\mathbf{s} \in \mathbb{G}^m$, we define $\mathsf{Additive}_{\mathbb{G}}(\mathbf{s})$ as $\mathsf{Additive}_{\mathbb{G}}$ being applied to $\mathbf{s}$ in an element-wise way. We simply write $\mathsf{Additive}$ instead of $\mathsf{Additive}_{\mathbb{G}}$ if $\mathbb{G}$ is clear from the context.

Let $\mathbb{F}$ be a finite field. Define $\mathsf{MakeBeaver}_{\mathbb{F}}()$ as an algorithm to generate a Beaver triple [3]. Formally, $\mathsf{MakeBeaver}_{\mathbb{F}}()$ takes no input and does the following: (1) Let $a, b \leftarrow_{\$} \mathbb{F}$ and $c = ab$; (2) Let $(a_i)_{i\in[n]} \leftarrow \mathsf{Additive}_{\mathbb{F}}(a)$, $(b_i)_{i\in[n]} \leftarrow \mathsf{Additive}_{\mathbb{F}}(b)$ and $(c_i)_{i\in[n]} \leftarrow \mathsf{Additive}_{\mathbb{F}}(c)$; (3) Output $(a_i, b_i, c_i)_{i\in[n]}$.

Suppose that $|\mathbb{F}| \geq n+1$ and let $\alpha_1, \ldots, \alpha_n$ be pairwise distinct non-zero elements of $\mathbb{F}$. Define $\mathsf{Shamir}_{\mathbb{F},t}(s)$ as an algorithm to generate shares of the (t,n)-Shamir secret sharing scheme for a secret $s \in \mathbb{F}$. Formally, on input $s \in \mathbb{F}$, $\mathsf{Shamir}_{\mathbb{F},t}(s)$ chooses a random polynomial φ over $\mathbb{F}$ of degree at most t such that $\varphi(0) = s$, and then outputs $(\varphi(\alpha_1), \ldots, \varphi(\alpha_n))$. For a vector $\mathbf{s} \in \mathbb{F}^m$, we define $\mathsf{Shamir}_{\mathbb{F},t}(\mathbf{s})$ as $\mathsf{Shamir}_{\mathbb{F},t}$ being applied to $\mathbf{s}$ in an element-wise way.

We can convert consistent shares of the Shamir scheme into additive shares for the same secret. Indeed, there exist constants $\ell_1, \ldots, \ell_n$, which we call *Lagrange coefficients associated with the α_i's*, such that $\ell_1 \cdot \varphi(\alpha_1) + \cdots + \ell_n \cdot \varphi(\alpha_n) = \varphi(0)$ for any polynomial φ of degree at most $n-1$. It immediately implies that $\sum_{i\in[n]} \ell_i v_i = s$ for $(v_i)_{i\in[n]} \leftarrow \mathsf{Shamir}_{\mathbb{F},t}(s)$, which means that $(\ell_i v_i)_{i\in[n]}$ is a tuple of additive shares for s.

Finally, we recall a simple mathematical fact that also follows from the invertibility of Vandermonde matrices: Let $s \in \mathbb{F}$ and $T \subseteq [n]$ be any set of size at most t. Then, there is a polynomial φ of degree at most t such that $\varphi(0) = s$ and $\varphi(\alpha_i) = 0$ for all $i \in T$.

3.3 Bloom Filters

Bloom filters [9] are probabilistic data structures that provide compact encodings of sets. Formally, let U be a set and $\mathcal{H}$ be a set of k independent uniform hash functions $\mathcal{H} = \{H_1, \dots, H_k\}$ such that each H_i maps elements in U to numbers in $[m]$. To add an element $x \in U$, an algorithm Add takes an m-bit string $\mathbf{BF} = (v^{(h)})_{h\in[m]}$ and an element $x \in U$ as input, and sets $v^{(h)} \leftarrow 1$ for all $h \in \{H_1(x), \dots, H_k(x)\}$. Let $\mathbf{BF}(\emptyset)$ denote the string whose bits are all zero. For $X \in 2^U$, let $\mathbf{BF}(X)$ denote a string obtained by adding the elements of X. That is, if $X = \{x_1, \dots, x_s\}$, first construct $\mathbf{BF}_1, \dots, \mathbf{BF}_s$ as $\mathbf{BF}_j = \mathsf{Add}(\mathbf{BF}_{j-1}, x_j)$ for all $j \in [s]$, where $\mathbf{BF}_0 = \mathbf{BF}(\emptyset)$, and then define $\mathbf{BF}(X) = \mathbf{BF}_s$. Note that the definition of $\mathbf{BF}(X)$ is independent of the order of the elements of X. To query $\mathbf{BF}(X)$ for an element $y \in U$ (i.e., to test whether $y \in X$), a query algorithm Check takes an m-bit string $\mathbf{BF} = (v^{(h)})_{h\in[m]}$ and an element $y \in U$ as input, and checks if $v^{(h)} = 1$ for all $h \in \{H_1(y), \dots, H_k(y)\}$. If so, the algorithm outputs 1 ("yes") and otherwise it outputs 0 ("no").

It is straightforward to see that if y is indeed in X, Check correctly outputs "yes". On the other hand, the converse is not true. Suppose that $y \notin X$. If the bits at positions $H_1(y), \dots, H_k(y)$ have by chance been set to 1 during the insertion of the elements of X, the algorithm outputs "yes" incorrectly, resulting in a false positive. We require a Bloom filter to satisfy that the probability of false positives is negligible. Formally, we say that $\mathcal{BF} = (\mathsf{Add}, \mathsf{Check})$ is a Bloom filter for U with parameters m, k and s if for any $X \in 2^U$ with $|X| = s$ and any $y \in U \setminus X$, the probability that $\mathsf{Check}(\mathbf{BF}(X), y) = 1$ is negligible in k, where the probability is taken over the random choice of $\mathcal{H}$.

We can implement $\mathcal{H}$ with a d-universal hash family, which ensures that the hash values of any d distinct inputs are independent and uniformly at random. According to [17], the false positive probability of a Bloom filter is still negligible even when a universal hash family is used instead of truly random functions. More precisely, the analysis [17, Proposition 3.10] shows that if $\mathcal{H}$ is a d-universal hash family for a constant $d \geq 2$, then the false positive probability of the Bloom filter based on $\mathcal{H}$ with parameters m, k, s is upper bounded by

$$\left(1 - \left(1 - \frac{k}{m}\right)^s + \left(\frac{ks}{m}\right)^d\right)^k \leq \left(1 - \left(1 - \frac{ks}{m}\right) + \left(\frac{ks}{m}\right)^d\right)^k = \left(\frac{ks}{m} + \left(\frac{ks}{m}\right)^d\right)^k$$

If we set $m = 2ks$, then $ks/m + (ks/m)^d = 1/2 + 1/2^d$ is a constant less than 1, and hence the probability is negligible in k. We note that an efficient d-universal hash family can be obtained by letting a hash function be a random polynomial p of degree $d-1$ over a field of size at least m, and letting the hash value of x be $p(x)$. Note that such a function can be described in $O(\log m)$ bits if $d = O(1)$.

4 BC-Efficient Protocols for Basic Functionalities

Broadcast. Let $\mathcal{F}_{\mathsf{Broadcast},i}$ be the functionality which receives a non-private input y from the i-th player and gives y to all players (described in Fig. 1). Since we assume that all players are semi-honest, we immediately obtain a BC-efficient protocol realizing $\mathcal{F}_{\mathsf{Broadcast},i}$. Indeed, we just utilize a round table structure where players are supposed to be nodes in a ring network and each player only communicates with the consecutive players around the table. Such a protocol implicitly appears in [52]. However, the round complexity of this protocol is $O(n)$. To reduce it to $O(\log n)$, assume that the set of n players is represented by a binary tree whose height is $O(\log n)$ and root is P_i. Each player sends his two children the element that he received from his parent node. We show the formal description of the protocol $\mathit{\Pi}_{\mathsf{Broadcast},i}$ in Fig. 1. The complexity of $\mathit{\Pi}_{\mathsf{Broadcast},i}$ is $\mathrm{BC}_{\mathrm{on}}(\mathit{\Pi}_{\mathsf{Broadcast},i}) = O(\ell_y)$ and $\mathrm{Round}(\mathit{\Pi}_{\mathsf{Broadcast},i}) = O(\log n)$, where ℓ_y is the bit-length of y. Note that $\mathrm{BC}_{\mathrm{off}}(\mathit{\Pi}_{\mathsf{Broadcast},i}) = 0$.

Functionality $\mathcal{F}_{\mathsf{Broadcast},i}$

Upon receiving y from the i-th player P_i, $\mathcal{F}_{\mathsf{Broadcast},i}$ gives every player y.

Protocol $\mathit{\Pi}_{\mathsf{Broadcast},i}$

Assumption. The set of n players is represented by a binary tree whose height is $h = O(\log n)$ and root is the i-th player P_i. For each $k \in [n]$, let Parent_k be the parent of P_k if $k \neq 1$, and $\{\mathsf{L}_k, \mathsf{R}_k\}$ be at most two children of P_k.

Non-private input. The i-th player P_i has y.

Output. Every player obtains y.

Protocol.

1. P_i sets $y_i = y$ and sends it to L_i and R_i.
2. For each $j = 1, 2, \ldots, h$, every player P_k at the j-th level sets y_k as the element he received from Parent_k, and sends it to L_k and R_k.
3. Each player P_k outputs y_k.

Fig. 1. The functionality $\mathcal{F}_{\mathsf{Broadcast},i}$ and a protocol $\mathit{\Pi}_{\mathsf{Broadcast},i}$ implementing it

Sum. In Fig. 2, we describe the functionality $\mathcal{F}_{\mathsf{Sum}}$ which receives group elements $x_1, \ldots, x_n \in \mathbb{G}$, each from P_i, and gives $s := \sum_{i \in [n]} x_i$ to all players. We show a BC-efficient protocol $\mathit{\Pi}_{\mathsf{Sum}}$ with $O(\log n)$ rounds. Assume that players are represented as nodes (including leafs) of a binary tree of depth $O(\log n)$. In the first round, players corresponding to leafs send their inputs to their parents.

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

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

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

Assumption. The set of n players is represented by a binary tree whose height is $h = O(\log n)$ and root is the n-th player P_n. For each $k \in [n]$, let Parent_k be the parent of P_k if $k \neq 1$, and $\{\mathsf{L}_k, \mathsf{R}_k\}$ be at most two children of P_k.

Input. Each player P_i has a group element $x_i \in \mathbb{G}$.

Output. Every player obtains $s = \sum_{i\in[n]} x_i$.

Setup.

1. Let $(a_i)_{i\in[n]} \leftarrow \mathsf{Additive}_{\mathbb{G}}(0)$.
2. Each player P_i receives a_i.

Protocol.

1. Each player P_i sets $y_i = x_i + a_i$.
2. Every player P_i at the h-th level sends y_i to Parent_i.
3. For each $j = h-1, h-2, \ldots, 1$, every player P_i at the j-th level sets z_i' and z_i'' as the elements he received from his left child and right child, respectively, and computes $z_i = z_i' + z_i'' + y_i$. If $i \neq n$, he sends z_i to Parent_i. If $i \neq n$, he broadcasts z_n to every player via $\mathcal{F}_{\mathsf{broadcast},n}$.
4. Each player P_i outputs z_n.

Fig. 2. The functionality $\mathcal{F}_{\mathsf{Sum}}$ and a protocol $\mathit{\Pi}_{\mathsf{Sum}}$ implementing it

In each round, players compute the sum of elements sent by their children, add their input to the sum, and send it to their parents. Finally, the player corresponding to the root obtains the aggregation of all inputs. Note that if one naively implements the above procedures, players learn additional information, e.g., a partial sum of inputs. We thus let players mask their inputs with additive shares of 0 in advance. The formal description of $\mathit{\Pi}_{\mathsf{Sum}}$ is shown in Fig. 2. The complexities are $\mathrm{BC}_{\mathrm{off}}(\mathit{\Pi}_{\mathsf{Sum}}) = O(\log|\mathbb{G}|)$, $\mathrm{BC}_{\mathrm{on}}(\mathit{\Pi}_{\mathsf{Sum}}) = O(\log|\mathbb{G}|)$ and $\mathrm{Round}(\mathit{\Pi}_{\mathsf{Sum}}) = O(\log n)$.

Functionality $\mathcal{F}_{\mathsf{Mult}}((x_i, y_i)_{i\in[n]})$

1. $\mathcal{F}_{\mathsf{Mult}}$ receives field elements $x_i, y_i \in \mathbb{F}$ from each player P_i.
2. $\mathcal{F}_{\mathsf{Mult}}$ computes $x = \sum_{i\in[n]} x_i$, $y = \sum_{i\in[n]} y_i$ and $z = xy$.
3. $\mathcal{F}_{\mathsf{Mult}}$ runs $(z_i)_{i\in[n]} \leftarrow \mathsf{Additive}_{\mathbb{F}}(z)$.
4. $\mathcal{F}_{\mathsf{Mult}}$ gives z_i to each player P_i.

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

Input. Each player P_i has $x_i, y_i \in \mathbb{F}$.

Output. Each player P_i obtains $z_i \in \mathbb{F}$, where $(z_i)_{i\in[n]} \leftarrow \mathcal{F}_{\mathsf{Mult}}((x_i, y_i)_{i\in[n]})$.

Setup.

1. Generate a Beaver triple $(a_i, b_i, c_i)_{i\in[n]} \leftarrow \mathsf{MakeBeaver}_{\mathbb{F}}()$.
2. Each player P_i receives (a_i, b_i, c_i).

Protocol.

1. Each player P_i computes $u_i = x_i - a_i$ and $v_i = y_i - b_i$.
2. Players obtain $u = \mathcal{F}_{\mathsf{Sum}}((u_i)_{i\in[n]})$ and $v = \mathcal{F}_{\mathsf{Sum}}((v_i)_{i\in[n]})$.
3. For each $k = 1, 2, \ldots, n$, P_k does the following:
   - If $k = 1$, output $z_1 = ub_1 + a_1v + c_1 + uv$.
   - If $k \neq 1$, output $z_i = ub_i + a_iv + c_i$.

Fig. 3. The functionality $\mathcal{F}_{\mathsf{Mult}}$ and a protocol $\mathit{\Pi}_{\mathsf{Mult}}$ implementing it

Multiplication. In Fig. 3, we describe the functionality $\mathcal{F}_{\mathsf{Mult}}$ which takes additive shares $(x_i)_{i\in[n]}$, $(y_i)_{i\in[n]}$ for x and y (respectively), and gives additive shares $(z_i)_{i\in[n]}$ for $z = xy$. We obtain a BC-efficient protocol $\mathit{\Pi}_{\mathsf{Mult}}$ for $\mathcal{F}_{\mathsf{Mult}}$ based on the above BC-efficient protocols for sum and a Beaver triple. We show its formal description in Fig. 3. The correctness and security follow from [3]. Since players also need to receive correlated randomness for two executions of a protocol implementing $\mathcal{F}_{\mathsf{Sum}}$, the protocol $\mathit{\Pi}_{\mathsf{Mult}}$ has $\mathrm{BC}_{\mathrm{off}}(\mathit{\Pi}_{\mathsf{Mult}}) = O(\log|\mathbb{F}|)$, $\mathrm{BC}_{\mathrm{on}}(\mathit{\Pi}_{\mathsf{Mult}}) = O(\log|\mathbb{F}|)$ and $\mathrm{Round}(\mathit{\Pi}_{\mathsf{Mult}}) = O(\log n)$.

Inner Product. Let $\mathbb{F}$ be a finite field. Define a functionality $\mathcal{F}_{\mathsf{IP}}$ as follows (described in Fig. 4): On input $(\mathbf{x}_i, \mathbf{y}_i)_{i\in[n]}$, where $\mathbf{x}_i, \mathbf{y}_i \in \mathbb{F}^m$, $\mathcal{F}_{\mathsf{IP}}$ gives all players $z = \langle \mathbf{x}, \mathbf{y} \rangle$, where $\mathbf{x} = \sum_{i\in[n]} \mathbf{x}_i$ and $\mathbf{y} = \sum_{i\in[n]} \mathbf{y}_i$. We construct a BC-efficient protocol $\mathit{\Pi}_{\mathsf{IP}}$ for $\mathcal{F}_{\mathsf{IP}}$ based on BC-efficient protocols for sum and multiplication. We show the security proof in Appendix C.1.

Functionality $\mathcal{F}_{\mathsf{IP}}((\mathbf{x}_i, \mathbf{y}_i)_{i\in[n]})$

1. $\mathcal{F}_{\mathsf{IP}}$ receives vectors $\mathbf{x}_i, \mathbf{y}_i \in \mathbb{F}^m$ from each player P_i.
2. $\mathcal{F}_{\mathsf{IP}}$ computes $\mathbf{x} = \sum_{i\in[n]} \mathbf{x}_i$, $\mathbf{y} = \sum_{i\in[n]} \mathbf{y}_i$ and $z = \langle \mathbf{x}, \mathbf{y} \rangle$.
3. $\mathcal{F}_{\mathsf{IP}}$ gives z to every player P_i.

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

Input. Each player P_i has $\mathbf{x}_i = (x_i^{(j)})_{j\in[m]} \in \mathbb{F}^m$ and $\mathbf{y}_i = (y_i^{(j)})_{j\in[m]} \in \mathbb{F}^m$.

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

Protocol.

1. For each $j \in [m]$, players obtain $(w_i^{(j)})_{i\in[n]} \leftarrow \mathcal{F}_{\mathsf{Mult}}((x_i^{(j)}, y_i^{(j)})_{i\in[n]})$.
2. Each player P_i computes $z_i = \sum_{j\in[m]} w_i^{(j)} \in \mathbb{F}$.
3. Players obtain $z = \mathcal{F}_{\mathsf{Sum}}((z_i)_{i\in[n]})$.
4. Each player P_i outputs z.

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

Since players need to receive correlated randomness for m executions of a protocol implementing $\mathcal{F}_{\mathsf{Mult}}$ and one execution of a protocol implementing $\mathcal{F}_{\mathsf{Sum}}$, the offline communication complexity is $\mathrm{BC}_{\mathrm{off}}(\mathit{\Pi}_{\mathsf{IP}}) = O(m \log |\mathbb{F}|)$. In the online phase, the protocol $\mathit{\Pi}_{\mathsf{IP}}$ invokes $\mathcal{F}_{\mathsf{Mult}}$ m times and $\mathcal{F}_{\mathsf{Sum}}$ once and thus has online communication complexity $\mathrm{BC}_{\mathrm{on}}(\mathit{\Pi}_{\mathsf{IP}}) = O(m \log |\mathbb{F}|)$ and round complexity $\mathrm{Round}(\mathit{\Pi}_{\mathsf{IP}}) = O(\log n)$.

We define a variant $\mathcal{F}'_{\mathsf{IP}}$, whose difference from $\mathcal{F}_{\mathsf{IP}}$ is that $\mathcal{F}'_{\mathsf{IP}}$ outputs shares for an inner product z instead of z itself. We can construct a BC-efficient protocol $\mathit{\Pi}'_{\mathsf{IP}}$ for $\mathcal{F}'_{\mathsf{IP}}$ in a similar way. Indeed, $\mathit{\Pi}'_{\mathsf{IP}}$ is the same as $\mathit{\Pi}_{\mathsf{IP}}$ except that it does not reconstruct the inner product z. Thus, the bottleneck communication complexity and round complexity of $\mathit{\Pi}'_{\mathsf{IP}}$ is asymptotically the same as those of $\mathit{\Pi}_{\mathsf{IP}}$. We defer the formal description to Appendix C.1.

5 BC-Efficient Protocols for General Symmetric Functions

5.1 First Protocol

First, we show a fully secure protocol that can achieve low online bottleneck complexity $O(\log n)$. Recall that for a function $h : \{0,1\}^n \to \{0,1\}$, we denote the functionality of giving every player $h(x_1, \dots, x_n)$ by $\mathcal{F}_h$.

Theorem 1. *Let $h : \{0,1\}^n \to \{0,1\}$ be a symmetric function. The protocol $\mathit{\Pi}_{\mathsf{Sym}}$ described in Fig. 5 is a fully secure MPC protocol for $\mathcal{F}_h$ in the $\mathcal{F}_{\mathsf{Sum}}$-hybrid model. Implementing $\mathcal{F}_{\mathsf{Sum}}$, the protocol $\mathit{\Pi}_{\mathsf{Sym}}$ achieves* $\mathrm{BC}_{\mathrm{off}}(\mathit{\Pi}_{\mathsf{Sym}}) = O(n)$, $\mathrm{BC}_{\mathrm{on}}(\mathit{\Pi}_{\mathsf{Sym}}) = O(\log n)$ *and* $\mathrm{Round}(\mathit{\Pi}_{\mathsf{Sym}}) = O(\log n)$.

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

Notations.
- Let $h : \{0,1\}^n \to \{0,1\}$ be a symmetric function.
- Let $f : [0..n] \to \{0,1\}$ be a function such that $h(x_1, \dots, x_n) = f(\sum_{i\in[n]} x_i)$ for all $(x_1, \dots, x_n) \in \{0,1\}^n$.
- Let $\mathbb{F} = \{0,1\}$ be the binary field.

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

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

Setup.
1. Let $r \leftarrow_{\$} \mathbb{Z}_{n+1}$ and $(r_i)_{i\in[n]} \leftarrow \mathsf{Additive}_{\mathbb{Z}_{n+1}}(r)$.
2. Define $\mathbf{T} = (T_i)_{i\in\mathbb{Z}_{n+1}} \in \{0,1\}^{n+1}$ by $T_i = f(i)$ for all $i \in \mathbb{Z}_{n+1}$.
3. Define $\mathbf{S} \in \{0,1\}^{n+1}$ by $\mathbf{S} = \mathsf{Shift}_r(\mathbf{T})$ and let $(\mathbf{S}_i)_{i\in[n]} \leftarrow \mathsf{Additive}_{\mathbb{F}}(\mathbf{S})$.
4. Each player P_i receives $(r_i, \mathbf{S}_i)$.

Protocol.
1. Each player P_i computes $y_i = x_i + r_i \bmod (n+1)$.
2. Players obtain $y = \mathcal{F}_{\mathsf{Sum}}((y_i)_{i\in[n]})$.
3. Each player P_i sets $z_i = (\mathbf{S}_i)_y$, where $(\mathbf{S}_i)_y$ is the y-th element of $\mathbf{S}_i$. Here, we identify the set indexing the entries of $\mathbf{S}_i$ with $\mathbb{Z}_{n+1}$.
4. Players obtain $z = \mathcal{F}_{\mathsf{Sum}}((z_i)_{i\in[n]})$.
5. Each player P_i outputs z.

Fig. 5. The first protocol $\mathit{\Pi}_{\mathsf{Sym}}$ for computing a symmetric function

Proof. First, we prove the correctness of $\mathit{\Pi}_{\mathsf{Sym}}$. Let $\mathbf{x} \in \{0,1\}^n$ be any input. Since $r = \sum_{i\in[n]} r_i$, it holds that $y = r + \sum_{i\in[n]} x_i$. It also holds that

$$z = \sum_{i\in[n]} z_i = \sum_{i\in[n]} (\mathbf{S}_i)_y = (\mathbf{S})_y = T_{(y-r) \bmod (n+1)}$$

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

Next, we prove the privacy of $\mathit{\Pi}_{\mathsf{Sym}}$. Let $T \subseteq [n]$ be the set of corrupted players. Let $H = \overline{T}$ be the set of honest players and fix an honest player $j \in H$. Note that corrupted players' view can be simulated from the following elements:

Correlated randomness. $(r_i, \mathbf{S}_i)$ for all $i \in T$;
Online messages. $y_i = x_i + r_i$ and $z_i = (\mathbf{S}_i)_y$ for all $i \in H$.

Let $\mathbf{x}, \mathbf{x}' \in \{0,1\}^n$ be any pair of inputs such that

$$x_i = x'_i \ (\forall i \in T) \text{ and } h(x_1, \ldots, x_n) = h(x'_1, \ldots, x'_n).$$

It is sufficient to prove that the distribution of the above elements during the execution of $\mathit{\Pi}_{\mathsf{Sym}}$ on input $\mathbf{x}$ is identical to that on input $\mathbf{x}'$. To show the equivalence of the distributions, we show a bijection between the random strings used by $\mathit{\Pi}_{\mathsf{Sym}}$ on input $\mathbf{x}$ and the random strings used by $\mathit{\Pi}_{\mathsf{Sym}}$ on input $\mathbf{x}'$ such that the correlated randomness received by T and the online messages from H are the same under this bijection. The set of all random strings is

$$\mathcal{R} = \left\{ (r_i, \mathbf{S}_i)_{i\in[n]} : \sum_{i\in[n]} \mathbf{S}_i = \mathsf{Shift}_r(\mathbf{T}),\ \text{where } r = \sum_{i\in[n]} r_i \right\}.$$

We denote the randomness of $\mathit{\Pi}_{\mathsf{Sym}}$ on input $\mathbf{x}$ by $(r_i, \mathbf{S}_i)_{i\in[n]}$ and that on input $\mathbf{x}'$ by $(r'_i, \mathbf{S}'_i)_{i\in[n]}$. The bijection maps the randomness $(r_i, \mathbf{S}_i)_{i\in[n]} \in \mathcal{R}$ to $(r'_i, \mathbf{S}'_i)_{i\in[n]} \in \mathcal{R}$ in such a way that

$$r'_i = \begin{cases} r_i, & \text{if } i \in T, \\ r_i + x_i - x'_i, & \text{if } i \in H, \end{cases}$$
$$\mathbf{S}'_i = \begin{cases} \mathbf{S}_i, & \text{if } i \neq j, \\ \mathbf{S}_j + \mathsf{Shift}_{r'}(\mathbf{T}) - \mathsf{Shift}_r(\mathbf{T}), & \text{if } i = j, \end{cases}$$

where $r := \sum_{i\in[n]} r_i$ and $r' := \sum_{i\in[n]} r'_i = r + \sum_{i\in H}(x_i - x'_i)$. The image is indeed a consistent random string (i.e., $(r'_i, \mathbf{S}'_i)_{i\in[n]} \in \mathcal{R}$) since $\sum_{i\in[n]} \mathbf{S}'_i = \mathsf{Shift}_{r'}(\mathbf{T})$ if and only if $\sum_{i\in[n]} \mathbf{S}_i = \mathsf{Shift}_r(\mathbf{T})$. The above map is indeed a bijection since it has the inverse

$$r_i = \begin{cases} r'_i, & \text{if } i \in T, \\ r'_i + x'_i - x_i, & \text{if } i \in H, \end{cases}$$
$$\mathbf{S}_i = \begin{cases} \mathbf{S}'_i, & \text{if } i \neq j, \\ \mathbf{S}'_j + \mathsf{Shift}_r(\mathbf{T}) - \mathsf{Shift}_{r'}(\mathbf{T}), & \text{if } i = j. \end{cases}$$

Clearly, this bijection does not change the correlated randomness $(r_i, \mathbf{S}_i)_{i\in T}$ of T. It can be seen that $x'_i + r'_i = x'_i + (r_i + x_i - x'_i) = x_i + r_i$ for $i \in H$. In particular, the message y is the same in both executions and hence so is $z_i = (\mathbf{S}_i)_y$ for $i \in H \setminus \{j\}$. We see that

$$\begin{aligned}(\mathbf{S}'_j)_y &= (\mathbf{S}_j)_y + (\mathsf{Shift}_{r'}(\mathbf{T}))_y - (\mathsf{Shift}_r(\mathbf{T}))_y \\ &= (\mathbf{S}_j)_y + f(y-r') - f(y-r) \\ &= (\mathbf{S}_j)_y\end{aligned}$$

since $f(y-r') = f(\sum_{i\in[n]} x'_i) = f(\sum_{i\in[n]} x_i) = f(y-r)$.

Finally, since players also need to receive correlated randomness for two executions of a protocol implementing $\mathcal{F}_{\mathsf{Sum}}$, the offline communication complexity is $\mathrm{BC}_{\mathrm{off}}(\mathit{\Pi}_{\mathsf{Sym}}) = O(n)$. Implementing $\mathcal{F}_{\mathsf{Sum}}$, $\mathit{\Pi}_{\mathsf{Sym}}$ has $\mathrm{BC}_{\mathrm{on}}(\mathit{\Pi}_{\mathsf{Sym}}) = O(\log n)$ and $\mathrm{Round}(\mathit{\Pi}_{\mathsf{Sym}}) = O(\log n)$. □

5.2 Second Protocol

The second protocol reduces the offline bottleneck complexity of the first protocol to $O(\sqrt{n})$ at the cost of increasing the online bottleneck complexity.

Theorem 2. *Let $h : \{0,1\}^n \to \{0,1\}$ be a symmetric function. The protocol $\mathit{\Pi}'_{\mathsf{Sym}}$ described in Fig. 6 is a fully secure MPC protocol for $\mathcal{F}_h$ in the $\{\mathcal{F}_{\mathsf{Sum}}, \mathcal{F}_{\mathsf{IP}}\}$-hybrid model. Implementing $\mathcal{F}_{\mathsf{Sum}}$ and $\mathcal{F}_{\mathsf{IP}}$, the protocol $\mathit{\Pi}'_{\mathsf{Sym}}$ achieves* $\mathrm{BC}_{\mathrm{off}}(\mathit{\Pi}'_{\mathsf{Sym}}) = O(\sqrt{n})$, $\mathrm{BC}_{\mathrm{on}}(\mathit{\Pi}'_{\mathsf{Sym}}) = O(\sqrt{n})$ *and* $\mathrm{Round}(\mathit{\Pi}'_{\mathsf{Sym}}) = O(\log n)$

Proof. We prove the correctness of $\mathit{\Pi}'_{\mathsf{Sym}}$. Since the proof of the privacy bears some similarities with that of $\mathit{\Pi}_{\mathsf{Sym}}$, we defer it to Appendix C.2. Let $\mathbf{x} \in \{0,1\}^n$ be any input. Since $r = \sum_{i\in[n]} r_i$, it holds that $y = \sum_{i\in[n]} x_i - r$. Let $\mathbf{a} := \sum_{i\in[n]} \mathbf{a}_i$ and $\mathbf{b} := \sum_{i\in[n]} \mathbf{b}_i$. Since $\sum_{i\in[n]} \mathbf{c}_i = \mathbf{e}_u$ and $\sum_{i\in[n]} \mathbf{d}_i = \mathbf{e}_v$, we have that

$$z = \langle \mathbf{a}, \mathbf{b} \rangle = \langle \mathsf{Shift}_\sigma\left(\mathbf{c}_1 + \cdots + \mathbf{c}_n\right), \mathbf{M} \cdot \mathsf{Shift}_\tau\left(\mathbf{d}_1 + \cdots + \mathbf{d}_n\right)\rangle = \mathbf{M}[u+\sigma, v+\tau],$$

where $\mathbf{M}[u', v']$ is the (u', v')-th entry of $\mathbf{M}$. Since $\phi^{-1}(u+\sigma, v+\tau) = y + r = \sum_{i\in[n]} x_i \in [0..n]$, it is equal to $f\left(y+r\right) = h(x_1, \ldots, x_n)$.

Finally, since players also need to receive correlated randomness for one execution of a protocol implementing $\mathcal{F}_{\mathsf{IP}}$ and for one execution of a protocol implementing $\mathcal{F}_{\mathsf{Sum}}$, the offline bottleneck complexity is

$$\mathrm{BC}_{\mathrm{off}}(\mathit{\Pi}'_{\mathsf{Sym}}) = O(\log pq) + O((p+q)\log|\mathbb{F}|) + O(p \log|\mathbb{F}|) = O(\sqrt{n}).$$

At Step 1 of the online phase, each player sends a constant number of elements in $\mathbb{Z}_{pq}$. The bottleneck complexity of Step 5 is equal to that of a protocol realizing $\mathcal{F}_{\mathsf{IP}}$. The protocol $\mathit{\Pi}'_{\mathsf{Sym}}$ thus has online communication complexity $\mathrm{BC}_{\mathrm{on}}(\mathit{\Pi}'_{\mathsf{Sym}}) = O(\sqrt{n})$ and round complexity $\mathrm{Round}(\mathit{\Pi}'_{\mathsf{Sym}}) = O(\log n)$. □

Protocol Π'_{Sym}

Notations.

- Let $h : \{0,1\}^n \to \{0,1\}$ be a symmetric function.
- Let $f : [0..n] \to \{0,1\}$ be a function such that $h(x_1, \ldots, x_n) = f(\sum_{i\in[n]} x_i)$ for all $(x_1, \ldots, x_n) \in \{0,1\}^n$.
- For a prime p, we identify the set indexing the entries of a vector of dimension p with $\mathbb{Z}_p$.
- Let p, q be primes such that $\sqrt{n} < p < q \leq O(\sqrt{n})$. We identify $[0..n]$ with a subset of $\mathbb{Z}_{pq}$.
- Let $\phi : \mathbb{Z}_{pq} \to \mathbb{Z}_p \times \mathbb{Z}_q$ be the ring isomorphism induced by the Chinese remainder theorem.
- Let $\mathbb{F} = \{0,1\}$ be the binary field.
- Define a matrix $\mathbf{M} \in \mathbb{F}^{p\times q}$ as follows: For $(y, z) \in \mathbb{Z}_p \times \mathbb{Z}_q$, the (y, z)-th entry of $\mathbf{M}$ is $f(\phi^{-1}(y, z))$ if $\phi^{-1}(y, z) \in [0..n]$, and 0 otherwise, where we identify the sets indexing the rows and columns of $\mathbf{M}$ as $\mathbb{Z}_p$ and $\mathbb{Z}_q$, respectively.

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

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

Setup.

1. Let $r \leftarrow_{\$} \mathbb{Z}_{pq}$, $(r_i)_{i\in[n]} \leftarrow \mathsf{Additive}_{\mathbb{Z}_{pq}}(r)$ and $(u, v) = \phi(r)$.
2. Let $(\mathbf{c}_i)_{i\in[n]} \leftarrow \mathsf{Additive}_{\mathbb{F}}(\mathbf{e}_u)$ and $(\mathbf{d}_i)_{i\in[n]} \leftarrow \mathsf{Additive}_{\mathbb{F}}(\mathbf{e}_v)$, where $\mathbf{e}_u \in \mathbb{F}^p$ (resp. $\mathbf{e}_v \in \mathbb{F}^q$) is the vector whose entry is 1 at position $u \in \mathbb{Z}_p$ (resp. $v \in \mathbb{Z}_q$), and 0 otherwise.
3. Each player P_i receives $(r_i, \mathbf{c}_i, \mathbf{d}_i)$.

Protocol.

1. Each player P_i computes $y_i = x_i - r_i \bmod pq$.
2. Players obtain $y = \mathcal{F}_{\mathsf{Sum}}((y_i)_{i\in[n]})$.
3. Each player P_i computes $\phi(y) = (\sigma, \tau) \in \mathbb{Z}_p \times \mathbb{Z}_q$.
4. Each player P_i computes $\mathbf{a}_i = \mathsf{Shift}_\sigma(\mathbf{c}_i) \in \mathbb{F}^p$ and $\mathbf{b}_i = \mathbf{M} \cdot \mathsf{Shift}_\tau(\mathbf{d}_i) \in \mathbb{F}^p$.
5. Players obtain $z = \mathcal{F}_{\mathsf{IP}}((\mathbf{a}_i, \mathbf{b}_i)_{i\in[n]})$.
6. Each player P_i outputs z.

Fig. 6. The second protocol Π'_{Sym} for computing a symmetric function

5.3 Third Protocol

The third protocol achieves bottleneck complexity $O(n^{1/d}\log n)$ for any constant d but is only secure against adversaries corrupting less than $n/(d-1)$ players.

To begin with, we prepare some notations. Let $h:\{0,1\}^n\to\{0,1\}$ be a symmetric function and $f:[0..n]\to\{0,1\}$ be a function such that $h(x_1,\ldots,x_n)=f(\sum_{i\in[n]}x_i)$ for all $(x_1,\ldots,x_n)\in\{0,1\}^n$. Let $d\geq 2$ be any constant and choose d pairwise distinct primes $p_1,\ldots,p_d$ such that $n^{1/d}<p_j\leq 2n^{1/d}$ for all $j\in[d]$. Such primes indeed exist for sufficiently large n since Bertrand's postulate [59, Theorem 5.8] ensures that there are at least $M/(3\log(2M))$ primes between M and $2M$. Set $N:=p_1\cdots p_d$ and $q:=N/p_1=p_2\cdots p_d$. We identify $[0..n]$ with a subset of $\mathbb{Z}_N$. Let $\phi:\mathbb{Z}_N\to\mathbb{Z}_{p_1}\times\mathbb{Z}_q$ and $\psi:\mathbb{Z}_q\to\mathbb{Z}_{p_2}\times\cdots\times\mathbb{Z}_{p_d}$ be the ring isomorphisms induced by the Chinese remainder theorem. Define projection maps $\pi_1:\mathbb{Z}_{p_1}\times\mathbb{Z}_q\to\mathbb{Z}_{p_1}$, $\pi_2:\mathbb{Z}_{p_1}\times\mathbb{Z}_q\to\mathbb{Z}_q$ and $\pi'_j:\mathbb{Z}_{p_2}\times\cdots\times\mathbb{Z}_{p_d}\to\mathbb{Z}_{p_j}$ for $j=2,\ldots,d$. Also, define

$$\phi_1=\pi_1\circ\phi,\ \psi_j=\pi'_j\circ\psi,\ \text{and}\ \phi_j=\psi_j\circ\pi_2\circ\phi, \tag{3}$$

where $\circ$ means the composition of maps. For $j\in[d]$, we identify the set indexing the entries of a vector $\mathbf{v}\in\mathbb{F}^{p_j}$ (resp. $\mathbf{v}\in\mathbb{F}^q$) with $\mathbb{Z}_{p_j}$ (resp. $\mathbb{Z}_q$). Let $\mathbf{e}_u\in\mathbb{F}^{p_j}$ denote the vector whose entry is 1 at position $u\in\mathbb{Z}_{p_j}$ and 0 otherwise. For vectors $\mathbf{v}_2\in\mathbb{F}^{p_2},\ldots,\mathbf{v}_d\in\mathbb{F}^{p_d}$, we define the Kronecker product $\mathbf{v}_2\otimes\cdots\otimes\mathbf{v}_d\in\mathbb{F}^q$ as

$$(\mathbf{v}_2\otimes\cdots\otimes\mathbf{v}_d)[k]=\mathbf{v}_2[\psi_2(k)]\cdots\mathbf{v}_d[\psi_d(k)]$$

for all $k\in\mathbb{Z}_q$. Define a matrix $\mathbf{M}\in\mathbb{F}^{p_1\times q}$ as follows: For each $(j,k)\in\mathbb{Z}_{p_1}\times\mathbb{Z}_q$, the (j,k)-th entry $\mathbf{M}[j,k]$ of $\mathbf{M}$ is

$$\mathbf{M}[j,k]=\begin{cases}f(\phi^{-1}(j,k)), & \text{if } \phi^{-1}(j,k)\in[0..n],\\ 0, & \text{otherwise,}\end{cases}$$

where we identify the sets indexing the rows and columns of $\mathbf{M}$ as $\mathbb{Z}_{p_1}$ and $\mathbb{Z}_q$, respectively. It then holds that for all $x\in[0..n]$,

$$\begin{aligned}&\langle\mathbf{e}_{\phi_1(x)},\mathbf{M}\cdot(\mathbf{e}_{\phi_2(x)}\otimes\cdots\otimes\mathbf{e}_{\phi_d(x)})\rangle\\ &=\sum_{j\in\mathbb{Z}_{p_1}}\mathbf{e}_{\phi_1(x)}[j]\left(\sum_{k\in\mathbb{Z}_q}\mathbf{M}[j,k]\cdot(\mathbf{e}_{\phi_2(x)}\otimes\cdots\otimes\mathbf{e}_{\phi_d(x)})[k]\right)\\ &=\sum_{k\in\mathbb{Z}_q}\mathbf{M}[\phi_1(x),k]\mathbf{e}_{\phi_2(x)}[\psi_2(k)]\cdots\mathbf{e}_{\phi_d(x)}[\psi_d(k)]\\ &=\mathbf{M}[\pi_1\circ\phi(x),\pi_2\circ\phi(x)]\\ &=f(x).\end{aligned} \tag{4}$$

At the third equation, we use the fact that

$$\begin{aligned}\psi_j(k) = \phi_j(x) \ (\forall j = 2, \dots, d) \ &\Rightarrow \ \pi'_j(\psi(k)) = \pi'_j(\psi \circ \pi_2 \circ \phi(x)) \ (\forall j = 2, \dots, d) \\ &\Rightarrow \ \psi(k) = \psi(\pi_2 \circ \phi(x)) \\ &\Rightarrow \ k = \pi_2 \circ \phi(x).\end{aligned}$$

Using the above notations, we show the following theorem.

Theorem 3. *Let $h : \{0,1\}^n \to \{0,1\}$ be a symmetric function. Let $d \geq 2$ be any constant. For any $t < n/(d-1)$, the protocol $\Pi^{(d)}_{\mathsf{Sym}}$ described in Fig. 7is a t-secure MPC protocol $\Pi^{(d)}_{\mathsf{Sym}}$ for $\mathcal{F}_h$ in the $\{\mathcal{F}_{\mathsf{Sum}}, \mathcal{F}_{\mathsf{IP}}\}$-hybrid model. Implementing $\mathcal{F}_{\mathsf{Sum}}$ and $\mathcal{F}_{\mathsf{IP}}$, the protocol $\Pi^{(d)}_{\mathsf{Sym}}$ achieves $\mathrm{BC}_{\mathrm{off}}(\Pi^{(d)}_{\mathsf{Sym}}) = O(n^{1/d}\log n)$, $\mathrm{BC}_{\mathrm{on}}(\Pi^{(d)}_{\mathsf{Sym}}) = O(n^{1/d}\log n)$ and $\mathrm{Round}(\Pi^{(d)}_{\mathsf{Sym}}) = O(\log n)$.*

Proof. First, we prove the correctness of $\Pi^{(d)}_{\mathsf{Sym}}$. Let $\mathbf{x} \in \{0,1\}^n$ be any input. Since $r = \sum_{i\in[n]} r_i$, it holds that $y = \sum_{i\in[n]} x_i - r$. We have that $\phi_j(\sum_{i\in[n]} x_i) = \phi_j(y) + \phi_j(r) = \sigma_j + u_j$ for all $j \in [d]$. At Step 4 of the online phase, we have that for all $j \in [d]$, $(\mathbf{a}_i^{(j)})_{i\in[n]}$ is a tuple of consistent shares whose secret is the unit vector $\mathbf{e}_{u_j+\sigma_j} \in \mathbb{F}^{p_j}$. That is, there is a tuple of degree-t polynomials $(\varphi_k^{(j)})_{k\in\mathbb{Z}_{p_j}}$ such that for all $k \in \mathbb{Z}_{p_j}$,

$$\varphi_k^{(j)}(0) = \mathbf{e}_{u_j+\sigma_j}[k] \text{ and } \varphi_k^{(j)}(\alpha_i) = \mathbf{a}_i^{(j)}[k] \ (\forall i \in [n]).$$

The property of Lagrange coefficients ℓ_i's implies that $(\mathbf{a}_i)_{i\in[n]}$ is a tuple of consistent additive shares for $\mathbf{e}_{u_1+\sigma_1}$. That is,

$$\mathbf{a} := \sum_{i\in[n]} \mathbf{a}_i = \mathbf{e}_{u_1+\sigma_1}.$$

Also, for any $k \in \mathbb{Z}_q$ and any $i \in [n]$, it holds that

$$(\mathbf{a}_i^{(2)} \otimes \cdots \otimes \mathbf{a}_i^{(d)})[k] = \mathbf{a}_i^{(2)}[k_2] \cdots \mathbf{a}_i^{(d)}[k_d] = (\varphi_{k_2}^{(2)} \cdots \varphi_{k_d}^{(d)})(\alpha_i),$$

where $k_j = \psi_j(k) \in \mathbb{Z}_{p_j}$. Since the degree of $\varphi_{k_2}^{(2)} \cdots \varphi_{k_d}^{(d)}$ is at most $t(d-1) \leq n-1$, we have that

$$\begin{aligned}\sum_{i\in[n]} \ell_i \cdot (\mathbf{a}_i^{(2)} \otimes \cdots \otimes \mathbf{a}_i^{(d)})[k] &= (\varphi_{k_2}^{(2)} \cdots \varphi_{k_d}^{(d)})(0) \\ &= \mathbf{e}_{u_2+\sigma_2}[k_2] \cdots \mathbf{e}_{u_d+\sigma_d}[k_d] \\ &= (\mathbf{e}_{u_2+\sigma_2} \otimes \cdots \otimes \mathbf{e}_{u_d+\sigma_d})[k].\end{aligned}$$

Therefore, we have that

$$\mathbf{b} := \sum_{i\in[n]} \mathbf{b}_i = \mathbf{M} \cdot \sum_{i\in[n]} \ell_i (\mathbf{a}_i^{(2)} \otimes \cdots \otimes \mathbf{a}_i^{(d)}) = \mathbf{M} \cdot (\mathbf{e}_{u_2+\sigma_2} \otimes \cdots \otimes \mathbf{e}_{u_d+\sigma_d}).$$

Protocol $\mathit{\Pi}^{(d)}_{\mathsf{Sym}}$

Notations.

- Let $h : \{0,1\}^n \to \{0,1\}$ be a symmetric function.
- Let $f : [0..n] \to \{0,1\}$ be a function such that $h(x_1, \dots, x_n) = f(\sum_{i \in [n]} x_i)$ for all $(x_1, \dots, x_n) \in \{0,1\}^n$.
- Let $p_1, \dots, p_d$ be d pairwise distinct primes such that $n^{1/d} < p_j \leq 2n^{1/d}$ for all $j \in [d]$, and set $N = p_1 \cdots p_d$ and $q = N/p_1 = p_2 \cdots p_d$.
- Let $\phi_j : \mathbb{Z}_N \to \mathbb{Z}_{p_j}$ ($j \in [d]$) be the ring homomorphism defined in Eq. (3).
- Let $\mathbf{M} \in \mathbb{F}^{p_1 \times q}$ be a matrix such that
$$\left\langle \mathbf{e}_{\phi_1(x)}, \mathbf{M} \cdot (\mathbf{e}_{\phi_2(x)} \otimes \cdots \otimes \mathbf{e}_{\phi_d(x)}) \right\rangle = f(x)$$
for all $x \in [0..n]$.
- Let $\mathbb{F}$ be the minimum finite field containing n pairwise distinct non-zero elements $\alpha_1, \dots, \alpha_n$.
- Let $\ell_1, \dots, \ell_n \in \mathbb{F}$ be Lagrange coefficients associated with the α_i's.

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

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

Setup.

1. Let $r \leftarrow_{\$} \mathbb{Z}_N$, $(r_i)_{i \in [n]} \leftarrow \mathsf{Additive}_{\mathbb{Z}_N}(r)$ and $u_j = \phi_j(r)$ for all $j \in [d]$.
2. For each $j \in [d]$, let $(\mathbf{c}_i^{(j)})_{i \in [n]} \leftarrow \mathsf{Shamir}_{\mathbb{F},t}(\mathbf{e}_{u_j})$, where $\mathbf{e}_{u_j} \in \mathbb{F}^{p_j}$ is the vector whose entry is 1 at position $u_j \in \mathbb{Z}_{p_j}$ and 0 otherwise.
3. Each player P_i receives r_i and $(\mathbf{c}_i^{(j)})_{j \in [d]}$.

Protocol.

1. Each player P_i computes $y_i = x_i - r_i \bmod N$.
2. Players obtain $y = \mathcal{F}_{\mathsf{Sum}}((y_i)_{i \in [n]})$.
3. Each player P_i computes $\phi_j(y) = \sigma_j \in \mathbb{Z}_{p_j}$ for all $j \in [d]$.
4. Each player P_i computes $\mathbf{a}_i^{(j)} = \mathsf{Shift}_{\sigma_j}(\mathbf{c}_i^{(j)}) \in \mathbb{F}^{p_j}$ for all $j \in [d]$.
5. Each player P_i sets $\mathbf{a}'_i := \mathbf{a}_i^{(1)}$ and computes $\mathbf{b}'_i := \mathbf{M} \cdot (\mathbf{a}_i^{(2)} \otimes \cdots \otimes \mathbf{a}_i^{(d)}) \in \mathbb{F}^{p_1}$.
6. Each player P_i computes $\mathbf{a}_i = \ell_i \cdot \mathbf{a}'_i$ and $\mathbf{b}_i = \ell_i \cdot \mathbf{b}'_i$.
7. Players obtain $z = \mathcal{F}_{\mathsf{IP}}((\mathbf{a}_i, \mathbf{b}_i)_{i \in [n]})$.
8. Each player P_i outputs z.

Fig. 7. The third protocol $\mathit{\Pi}^{(d)}_{\mathsf{Sym}}$ for computing a symmetric function

We obtain that

$$z = \langle \mathbf{a}, \mathbf{b} \rangle = \langle \mathbf{e}_{u_1+\sigma_1}, \mathbf{M} \cdot (\mathbf{e}_{u_2+\sigma_2} \otimes \cdots \otimes \mathbf{e}_{u_d+\sigma_d}) \rangle = f(\sum_{i\in[n]} x_i) = h(x_1, \ldots, x_n).$$

Here, we use the fact that $u_j + \sigma_j = \phi_j(\sum_{i\in[n]} x_i)$ and Eq. (4).

Next, we prove the privacy of $\mathit{\Pi}^{(d)}_{\mathsf{Sym}}$. Let $T \subseteq [n]$ be the set of t corrupted players. Let $H = \overline{T}$ be the set of honest players and fix an honest player $j \in H$. In the $\mathcal{F}_{\mathsf{IP}}$-hybrid model, corrupted players' view at Step 7 (including their correlated randomness for $\mathcal{F}_{\mathsf{IP}}$) only contains their inputs $(\mathbf{a}_i, \mathbf{b}_i)_{i\in T}$ to $\mathit{\Pi}_{\mathsf{IP}}$ and the output $z = h(x_1, \ldots, x_n)$. Since every $(\mathbf{a}_i, \mathbf{b}_i)$ is locally computed from y and $(\mathbf{c}^{(j)}_i)_{j\in[d]}$ at Steps 3–6, it is sufficient to show that the joint distribution of the following elements can be simulated from $(x_i)_{i\in T}$ and z:

Correlated randomness. r_i and $(\mathbf{c}^{(j)}_i)_{j\in[d]}$ for all $i \in T$;
Online messages. $y_i = x_i - r_i$ for all $i \in H$.

Let $\mathbf{x}, \tilde{\mathbf{x}} \in \{0,1\}^n$ be any pair of inputs such that

$$x_i = \tilde{x}_i \ (\forall i \in T) \text{ and } h(x_1, \ldots, x_n) = h(\tilde{x}_1, \ldots, \tilde{x}_n).$$

It is sufficient to prove that the distribution of the above elements during the execution of $\mathit{\Pi}^{(d)}_{\mathsf{Sym}}$ on input $\mathbf{x}$ is identical to that on input $\tilde{\mathbf{x}}$. To show the equivalence of the distributions, we show a bijection between the random strings used by $\mathit{\Pi}^{(d)}_{\mathsf{Sym}}$ on input $\mathbf{x}$ and the random strings used by $\mathit{\Pi}^{(d)}_{\mathsf{Sym}}$ on input $\tilde{\mathbf{x}}$ such that the above values are the same under this bijection. Note that the randomness of $\mathit{\Pi}^{(d)}_{\mathsf{Sym}}$ on input $\mathbf{x}$ is uniformly distributed over a set $\mathcal{S}$ consisting of all $(r_i, (\mathbf{c}^{(j)}_i)_{j\in[d]})_{i\in[n]}$ such that for each $j \in [d]$, $(\mathbf{c}^{(j)}_i)_{i\in[n]}$ is a tuple of consistent shares of the (t, n)-Shamir scheme for a secret $\mathbf{e}_{u_j}$, where $u_j = \phi_j(\sum_{i\in[n]} r_i)$.

We recall the fact that for any $\mathbf{c} \in \mathbb{F}^p$, there exists a uniquely determined tuple of p polynomials $\boldsymbol{\theta}(X) \in (\mathbb{F}[X])^p$, each of degree at most t, such that

$$\boldsymbol{\theta}(0) = \mathbf{c} \text{ and } \boldsymbol{\theta}(\alpha_i) = \mathbf{0} \ (\forall i \in T)$$

(see Section 3.2). Now, we define a bijection map from the randomness $(r_i, (\mathbf{c}^{(j)}_i)_{j\in[d]})_{i\in[n]}$ of $\mathit{\Pi}^{(d)}_{\mathsf{Sym}}$ on input $\mathbf{x}$ to the randomness $(\tilde{r}_i, (\tilde{\mathbf{c}}^{(j)}_i)_{j\in[d]})_{i\in[n]}$ of $\mathit{\Pi}^{(d)}_{\mathsf{Sym}}$ on input $\tilde{\mathbf{x}}$ in such a way that

$$\tilde{r}_i = \begin{cases} r_i, & \text{if } i \in T, \\ r_i + \tilde{x}_i - x_i, & \text{if } i \in H, \end{cases}$$

$$\tilde{\mathbf{c}}^{(j)}_i = \begin{cases} \mathbf{c}^{(j)}_i, & \text{if } i \in T, \\ \mathbf{c}^{(j)}_i + \boldsymbol{\theta}^{(j)}(\alpha_i), & \text{if } i \in H, \end{cases}$$

where

$$\tilde{r} := \sum_{i\in[n]} \tilde{r}_i, \ r := \sum_{i\in[n]} r_i, \ \tilde{u}_j := \phi_j(\tilde{r}), \ u_j := \phi_j(r),$$

and $\boldsymbol{\theta}^{(j)}$ is the uniquely determined tuple of p_j polynomials, each of degree at most t, such that $\boldsymbol{\theta}^{(j)}(0) = \mathbf{e}_{\tilde{u}_j} - \mathbf{e}_{u_j}$ and $\boldsymbol{\theta}^{(j)}(\alpha_i) = \mathbf{0}$ for all $i \in T$.

We see that the image is indeed a consistent random string, i.e., $(\tilde{r}_i, (\tilde{\mathbf{c}}_i^{(j)})_{j\in[d]})_{i\in[n]} \in \mathcal{S}$. If $(r_i, (\mathbf{c}_i^{(j)})_{j\in[d]})_{i\in[n]} \in \mathcal{S}$, then $(\mathbf{c}_i^{(j)})_{i\in[n]}$ forms a tuple of consistent shares for $\mathbf{e}_{u_j}$, i.e., there is a tuple of p_j polynomials $\boldsymbol{\varphi}^{(j)}$, each of degree at most t, such that

$$\boldsymbol{\varphi}^{(j)}(0) = \mathbf{e}_{u_j} \text{ and } \boldsymbol{\varphi}^{(j)}(\alpha_i) = \mathbf{c}_i^{(j)} \ (\forall i \in [n]).$$

A tuple of polynomials $\tilde{\boldsymbol{\varphi}}^{(j)} := \boldsymbol{\varphi}^{(j)} + \boldsymbol{\theta}^{(j)}$ satisfies that

$$\begin{aligned}\tilde{\boldsymbol{\varphi}}^{(j)}(0) &= \boldsymbol{\varphi}^{(j)}(0) + \boldsymbol{\theta}^{(j)}(0) = \mathbf{e}_{u_j} + (\mathbf{e}_{\tilde{u}_j} - \mathbf{e}_{u_j}) = \mathbf{e}_{\tilde{u}_j},\\ \tilde{\boldsymbol{\varphi}}^{(j)}(\alpha_i) &= \mathbf{c}_i^{(j)} + \boldsymbol{\theta}^{(j)}(\alpha_i) = \tilde{\mathbf{c}}_i^{(j)} \ (\forall i \in [n])\end{aligned}$$

since $\boldsymbol{\theta}^{(j)}(\alpha_i) = \mathbf{0}$ for any $i \in T$. Thus, we have that $(\tilde{r}_i, (\tilde{\mathbf{c}}_i^{(j)})_{j\in[d]})_{i\in[n]} \in \mathcal{S}$.

The above map is indeed a bijection since it has the inverse

$$r_i = \begin{cases}\tilde{r}_i, & \text{if } i \in T,\\ \tilde{r}_i + x_i - \tilde{x}_i, & \text{if } i \in H,\end{cases}$$

$$\mathbf{c}_i = \begin{cases}\tilde{\mathbf{c}}_i, & \text{if } i \in T,\\ \tilde{\mathbf{c}}_j - \boldsymbol{\theta}^{(j)}(\alpha_i), & \text{if } i \in H,\end{cases}$$

Clearly, this bijection does not change the correlated randomness $(r_i, (\mathbf{c}_i^{(j)})_{j\in[d]})_{i\in T}$ of T. Since $x_i' - r_i' = x_i' - (r_i + x_i' - x_i) = x_i - r_i$ for $i \in H$, it does not change the online messages from H.

Finally, since players also need to receive correlated randomness for one execution of a protocol implementing $\mathcal{F}_{\mathsf{IP}}$ and for one execution of a protocol implementing $\mathcal{F}_{\mathsf{Sum}}$, the offline bottleneck complexity is

$$\mathrm{BC}_{\mathsf{off}}(\mathit{\Pi}_{\mathsf{Sym}}^{(d)}) = O(\log N) + \sum_{j\in[d]} O(p_j \log|\mathbb{F}|) + O(p_1 \log|\mathbb{F}|) = O(n^{1/d}\log n).$$

Here, we use the fact that $d \in O(1)$, $p_j \in O(n^{1/d})$ and $|\mathbb{F}| \in O(n)$. At Steps 1 and 2 of the online phase, each player sends a constant number of elements in $\mathbb{Z}_N$. Players perform local computation at Steps 3–6. The bottleneck complexity of Step 7 is equal to that of a protocol realizing $\mathcal{F}_{\mathsf{IP}}$. The protocol $\mathit{\Pi}_{\mathsf{Sym}}^{(d)}$ thus has online communication complexity $\mathrm{BC}_{\mathrm{on}}(\mathit{\Pi}_{\mathsf{Sym}}^{(d)}) = O(n^{1/d}\log n)$ and round complexity $\mathrm{Round}(\mathit{\Pi}_{\mathsf{Sym}}^{(d)}) = O(\log n)$. □

6 BC-Efficient Protocol for Checking Equality to Zero

Let $\mathbb{F}$ be a finite field. Define a functionality $\mathcal{F}_{\mathsf{CheckZero},\mathbb{F}}$ as follows: On input $(x_i)_{i\in[n]} \in \mathbb{F}^n$, $\mathcal{F}_{\mathsf{CheckZero},\mathbb{F}}$ gives all players $b \in \{0,1\}$ such that $b = 0$ if and only if $\sum_{i\in[n]} x_i = 0$ in $\mathbb{F}$. We show a protocol tailored to the functionality that achieves lower bottleneck complexity than our protocols for general symmetric functions. A high-level idea is given in Section 2.2. The formal description and security proof are deferred to Appendix C.3.

Theorem 4. *Let λ be a security parameter. Let $\mathbb{F}$ be a finite field and $\mathbb{K}$ be an extension field of $\mathbb{F}$ such that $|\mathbb{K}| \geq 2^\lambda$. The protocol $\Pi_{\mathsf{CheckZero},\mathbb{K}}$ described in Fig. 10 is a fully secure MPC protocol for $\mathcal{F}_{\mathsf{CheckZero},\mathbb{F}}$ in the $\mathcal{F}_{\mathsf{Sum}}$-hybrid model. Implementing $\mathcal{F}_{\mathsf{Sum}}$, the protocol $\Pi_{\mathsf{CheckZero},\mathbb{K}}$ achieves* $\mathrm{BC}_{\mathrm{off}}(\Pi_{\mathsf{CheckZero},\mathbb{K}}) = O(\max\{\lambda, \log|\mathbb{F}|\})$, $\mathrm{BC}_{\mathrm{on}}(\Pi_{\mathsf{CheckZero},\mathbb{K}}) = O(\max\{\lambda, \log|\mathbb{F}|\})$ *and* $\mathrm{Round}(\Pi_{\mathsf{CheckZero},\mathbb{K}}) = O(\log n)$.

Remark 1. If the characteristic of $\mathbb{F}$ is larger than n, e.g., $\mathbb{F} = \mathbb{Z}_p$ for a prime $p > n$, our protocol $\Pi_{\mathsf{CheckZero},\mathbb{K}}$ implies BC-efficient protocols for computing the AND and OR of players' inputs $x_i \in \{0,1\}$. Indeed, we can compute the OR function due to the fact that if $x_i \in \{0,1\}$, $\sum_{i\in[n]} x_i = 0$ (over $\mathbb{F}$) if and only if $x_i = 0$ for all $i \in [n]$. A protocol for the AND function is immediately follows from the fact that $\mathrm{AND}(x_1,\dots,x_n) = 1 - \mathrm{OR}(1-x_1,\dots,1-x_n)$.

7 BC-Efficient Protocol for Private Set Intersection

In this section, we show a BC-efficient protocol for computing the intersection of players' input sets. For now, we assume that players' input sets have the same size s. We will show later that the protocol is extended to the general case.

To begin with, we define a functionality computing the intersection based on a Bloom filter. Let λ be a security parameter. Let U be a finite set. Assume that there exists a Bloom filter $\mathcal{BF} = (\mathsf{Add}, \mathsf{Check})$ for U with parameters $m = m(\lambda)$, $k = \Theta(\lambda)$ and $s = \mathsf{poly}(\lambda)$. For a subset $X \in 2^U$ of size s, let $\mathbf{BF}(X)$ denote an m-bit string obtained after adding the elements of X with Add. Define an n-input/single-output functionality $\mathcal{F}_{\mathsf{PSI},\mathcal{BF}}$ as follows (described in Fig. 8): On input $(X_i)_{i\in[n]}$ such that X_i is a subset of U of size s, $\mathcal{F}_{\mathsf{PSI},\mathcal{BF}}$ gives all players

$$Z_\lambda(X_1,\dots,X_n) := \{x \in X_n : \forall i \in [n],\ \mathsf{Check}(\mathbf{BF}(X_i), x) = 1\}. \tag{5}$$

Recall that the property of the Bloom filter $\mathcal{BF}$ ensures that for any (x, X) such that $x \notin X$, the probability that $\mathsf{Check}(\mathbf{BF}(X), x) = 1$ is negligible in k (and hence in λ). For any $x \in X_1 \cap \dots \cap X_n$, it holds with probability 1 that $x \in Z_\lambda(X_1,\dots,X_n)$, while for $x \in U \setminus (X_1 \cap \dots \cap X_n)$, the probability that $x \in Z_\lambda(X_1,\dots,X_n)$ is negligible in λ since we suppose $n = \mathsf{poly}(\lambda)$. Thus, the probability that $Z_\lambda(X_1,\dots,X_n) = X_1 \cap \dots \cap X_n$ is at least $1 - |X_n| \cdot \mathsf{negl}(\lambda) = 1 - s \cdot \mathsf{negl}(\lambda) \geq 1 - \mathsf{negl}(\lambda)$ since we assume $s = \mathsf{poly}(\lambda)$. Therefore, for any input $(X_i)_{i\in[n]}$, the statistical distance between $Z_\lambda(X_1,\dots,X_n)$ and $\mathrm{Int}_\lambda(X_1,\dots,X_n)$ is upper bounded by a negligible function. Here, we abuse notation and denote the random variable over 2^U defined in Eq. (5) by $Z_\lambda(X_1,\dots,X_n)$ and the random variable whose outcome is $X_1 \cap \dots \cap X_n$ with probability 1 by $\mathrm{Int}_\lambda(X_1,\dots,X_n)$. In the following, we show a protocol realizing $\mathcal{F}_{\mathsf{PSI},\mathcal{BF}}$. From the above observation, it also securely realizes the functionality that directly computes $\mathrm{Int}_\lambda(X_1,\dots,X_n) = X_1 \cap \dots \cap X_n$.

Functionality $\mathcal{F}_{\mathsf{PSI},\mathcal{BF}}((X_i)_{i\in[n]})$

1. $\mathcal{F}_{\mathsf{PSI},\mathcal{BF}}$ receives a subset $X_i \in 2^U$ of size s from each player P_i.
2. $\mathcal{F}_{\mathsf{PSI},\mathcal{BF}}$ gives every player

$$Z_\lambda(X_1,\ldots,X_n) := \{x \in X_n : \forall i \in [n],\ \mathsf{Check}(\mathbf{BF}(X_i),x) = 1\}.$$

Protocol Π_{PSI}

Notations.

- Let U be a set.
- Let $\mathcal{BF} = (\mathsf{Add},\mathsf{Check})$ be a Bloom filter for U with parameters m, k and s.
- For a subset $X \in 2^U$ of size s, let $\mathbf{BF}(X)$ denote an m-bit string obtained after adding the elements of X with Add.
- Let $\mathbb{F} = \mathbb{Z}_p$ be a prime field such that $p > nk$.

Input. Each player P_i has a subset $X_i \in 2^U$ of size s.

Output. Every player obtains $Z = \mathcal{F}_{\mathsf{PSI},\mathcal{BF}}((X_i)_{i\in[n]})$.

Setup.

1. Let $(\mathbf{u}_i)_{i\in[n]} \leftarrow \mathsf{Additive}_{\mathbb{F}}(\mathbf{0}_m)$ and $(\mathbf{w}_i^{(j)})_{i\in[n]} \leftarrow \mathsf{Additive}_{\mathbb{F}}(\mathbf{0}_m)$ for $j \in [s]$.
2. Each player P_i receives $\mathbf{u}_i$ and $(\mathbf{w}_i^{(j)})_{j\in[s]}$.

Protocol.

1. Each player P_i computes $\mathbf{B}_i = \mathbf{1}_m - \mathbf{BF}(X_i) \in \{0,1\}^m$.
2. Each player P_i computes $\mathbf{V}_i = \mathbf{B}_i + \mathbf{u}_i$, which is the i-th additive share of $\mathbf{V} := \sum_{i\in[n]} \mathbf{B}_i \in \mathbb{F}^m$.
3. P_n permutes the elements of X_n uniformly at random and lets $x^{(1)},\ldots,x^{(s)}$ be the permuted elements of X_n.
4. Each player P_i does the following:
   - If $i \neq n$, P_i sets $\mathbf{W}_i^{(j)} = \mathbf{w}_i^{(j)}$ for all $j \in [s]$.
   - If $i = n$, P_n computes $\mathbf{W}_n^{(j)} = \mathbf{BF}(\{x^{(j)}\}) + \mathbf{w}_n^{(j)}$ for all $j \in [s]$.
5. For each $j \in [s]$, players obtain $(y_i^{(j)})_{i\in[n]} \leftarrow \mathcal{F}'_{\mathsf{IP}}((\mathbf{V}_i, \mathbf{W}_i^{(j)})_{i\in[n]})$.
6. For each $j \in [s]$, players obtain $z^{(j)} = \mathcal{F}_{\mathsf{CheckZero},\mathbb{F}}((y_i^{(j)})_{i\in[n]})$.
7. P_n computes $Z = \{x^{(j)}\}_{j\in[s]:z^{(j)}=0}$ and invoke $\mathcal{F}_{\mathsf{Broadcast},n}$ with input Z.
8. Each player P_i outputs Z.

Fig. 8. The functionality $\mathcal{F}_{\mathsf{PSI},\mathcal{BF}}$ and a protocol Π_{PSI} implementing it

We now show the following theorem. The formal proof is given in Appendix C.4.

Theorem 5. *Let λ be a security parameter. Assume that there exists a Bloom filter $\mathcal{BF}$ for U with parameters $m = m(\lambda)$, $k = \Theta(\lambda)$ and $s = \mathsf{poly}(\lambda)$. Let $\mathbb{F} = \mathbb{Z}_p$ be a prime field such that $p > nk$. The protocol Π_{PSI} described in Fig. 8 is a fully secure MPC protocol for $\mathcal{F}_{\mathsf{PSI},\mathcal{BF}}$ in the $\{\mathcal{F}_{\mathsf{Broadcast},n}, \mathcal{F}_{\mathsf{CheckZero},\mathbb{F}}, \mathcal{F}'_{\mathsf{IP}}\}$-hybrid model. Implementing $\mathcal{F}_{\mathsf{Broadcast},n}$, $\mathcal{F}_{\mathsf{CheckZero},\mathbb{F}}$, and $\mathcal{F}'_{\mathsf{IP}}$, the protocol Π_{PSI} achieves $\mathrm{BC}_{\mathrm{off}}(\Pi_{\mathsf{PSI}}) = O(sm \log(n\lambda) + s\lambda)$, $\mathrm{BC}_{\mathrm{on}}(\Pi_{\mathsf{PSI}}) = O(sm \log(n\lambda) + s\lambda + s \log |U|)$ and $\mathrm{Round}(\Pi_{\mathsf{PSI}}) = O(\log n)$.*

According to the analysis in [17], we can choose a Bloom filter $\mathcal{BF}$ such that $m = \Theta(ks)$. Then, the complexity of Π_{PSI} is $\mathrm{BC}_{\mathrm{off}}(\Pi_{\mathsf{PSI}}) = O(s^2\lambda \log(n\lambda))$, $\mathrm{BC}_{\mathrm{on}}(\Pi_{\mathsf{PSI}}) = O(s^2\lambda \log(n\lambda) + s \log |U|)$, and $\mathrm{Round}(\Pi_{\mathsf{PSI}}) = O(\log n)$.

Finally, we deal with the general case where the sizes of players' input sets are upper bounded by s. Let $V_1, \dots, V_n$ be n sets, each of size s, such that they are pairwise disjoint and also disjoint from U, i.e., $|V_i \cap V_j| = |V_i \cap U| = \emptyset$ for any $i \neq j$. Set $U' = U \cup V_1 \cup \dots \cup V_n$. Each player P_i pads his input set X_i with $s - |X_i|$ elements in V_i if $|X_i| < s$, and lets X'_i be the resulting set of size exactly s. Since $X'_1 \cap \dots \cap X'_n = X_1 \cap \dots \cap X_n$, players can compute the intersection by running Π_{PSI} on input $X'_1, \dots, X'_n \subseteq U'$. Since $|U'| = |U| + ns$, this reduction only incurs an additive factor of $\log(ns)$ to online bottleneck complexity. In particular, the complexity is asymptotically the same as given above.

Acknowledgements

We thank anonymous reviewers of ASIACRYPT 2023 for their helpful comments. Especially, one of the reviewers kindly showed us a more round-efficient protocol for computing the sum, which reduces the round complexity of our original protocols to $O(\log n)$. We really appreciate his or her great solution. We also thank Koji Nuida and Takahiro Matsuda for helpful discussions and suggestions. This work was partially supported by JST AIP Acceleration Research JPMJCR22U5, Japan and JST CREST Grant Number JPMJCR22M1, Japan.

References

1. Bay, A., Erkin, Z., Hoepman, J.H., Samardjiska, S., Vos, J.: Practical multi-party private set intersection protocols. IEEE Transactions on Information Forensics and Security **17**, 1–15 (2022)
2. Bay., A., Erkin., Z., Alishahi., M., Vos., J.: Multi-party private set intersection protocols for practical applications. In: Proceedings of the 18th International Conference on Security and Cryptography – SECRYPT,. pp. 515–522 (2021)
3. Beaver, D.: Efficient multiparty protocols using circuit randomization. In: Advances in Cryptology – CRYPTO '91. pp. 420–432 (1992)
4. Beerliová-Trubíniová, Z., Hirt, M.: Perfectly-secure mpc with linear communication complexity. In: Theory of Cryptography. pp. 213–230 (2008)

5. Beimel, A., Gabizon, A., Ishai, Y., Kushilevitz, E., Meldgaard, S., Paskin-Cherniavsky, A.: Non-interactive secure multiparty computation. In: Advances in Cryptology – CRYPTO 2014, Part II. pp. 387–404 (2014)
6. Ben-Efraim, A., Nissenbaum, O., Omri, E., Paskin-Cherniavsky, A.: Psimple: Practical multiparty maliciously-secure private set intersection. In: Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security. pp. 1098–1112. ASIA CCS '22 (2022)
7. Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness theorems for non-cryptographic fault-tolerant distributed computation. In: Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing. pp. 1–10 (1988)
8. Ben-Sasson, E., Fehr, S., Ostrovsky, R.: Near-linear unconditionally-secure multiparty computation with a dishonest minority. In: Advances in Cryptology – CRYPTO 2012. pp. 663–680 (2012)
9. Bloom, B.H.: Space/time trade-offs in hash coding with allowable errors. Communications of the ACM **13**(7), 422–426 (1970)
10. Bose, P., Guo, H., Kranakis, E., Maheshwari, A., Morin, P., Morrison, J., Smid, M., Tang, Y.: On the false-positive rate of bloom filters. Information Processing Letters **108**(4), 210–213 (2008)
11. Boyle, E., Jain, A., Prabhakaran, M., Yu, C.H.: The Bottleneck Complexity of Secure Multiparty Computation. In: 45th International Colloquium on Automata, Languages, and Programming (ICALP 2018). Leibniz International Proceedings in Informatics (LIPIcs), vol. 107, pp. 24:1–24:16 (2018)
12. Canetti, R.: Universally composable security: a new paradigm for cryptographic protocols. In: Proceedings 42nd IEEE Symposium on Foundations of Computer Science. pp. 136–145 (2001)
13. Chandran, N., Dasgupta, N., Gupta, D., Obbattu, S.L.B., Sekar, S., Shah, A.: Efficient linear multiparty psi and extensions to circuit/quorum psi. In: Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security. pp. 1182–1204. CCS '21 (2021)
14. Chaum, D., Crépeau, C., Damgard, I.: Multiparty unconditionally secure protocols. In: Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing. pp. 11–19. STOC '88 (1988)
15. Cheon, J.H., Jarecki, S., Seo, J.H.: Multi-party privacy-preserving set intersection with quasi-linear complexity. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences **95**(8), 1366–1378 (2012)
16. Chida, K., Genkin, D., Hamada, K., Ikarashi, D., Kikuchi, R., Lindell, Y., Nof, A.: Fast large-scale honest-majority MPC for malicious adversaries. In: Advances in Cryptology – CRYPTO 2018, Part III. pp. 34–64 (2018)
17. Chung, K.M., Mitzenmacher, M., Vadhan, S.: Why simple hash functions work: Exploiting the entropy in a data stream. Theory of Computing **9**(30), 897–945 (2013)
18. Cramer, R., Damgård, I., Maurer, U.: General secure multi-party computation from any linear secret-sharing scheme. In: Advances in Cryptology – EUROCRYPT 2000. pp. 316–334 (2000)
19. Cramer, R., Fehr, S., Ishai, Y., Kushilevitz, E.: Efficient multi-party computation over rings. In: Advances in Cryptology — EUROCRYPT 2003. pp. 596–613 (2003)
20. Dachman-Soled, D., Malkin, T., Raykova, M., Yung, M.: Secure efficient multiparty computing of multivariate polynomials and applications. In: Applied Cryptography and Network Security. pp. 130–146 (2011)

21. Damgård, I., Ishai, Y., Krøigaard, M.: Perfectly secure multiparty computation and the computational overhead of cryptography. In: Advances in Cryptology – EUROCRYPT 2010. pp. 445–465 (2010)
22. Damgård, I., Nielsen, J.B.: Scalable and unconditionally secure multiparty computation. In: Advances in Cryptology – CRYPTO 2007. pp. 572–590 (2007)
23. Damgård, I., Pastro, V., Smart, N., Zakarias, S.: Multiparty computation from somewhat homomorphic encryption. In: Advances in Cryptology – CRYPTO 2012. pp. 643–662 (2012)
24. Feige, U., Kilian, J., Naor, M.: A minimal model for secure computation (extended abstract). In: Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing. pp. 554–563. STOC '94 (1994)
25. Freedman, M.J., Nissim, K., Pinkas, B.: Efficient private matching and set intersection. In: Advances in Cryptology - EUROCRYPT 2004. pp. 1–19 (2004)
26. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing. pp. 169–0178. STOC '09 (2009)
27. Ghosh, S., Nilges, T.: An algebraic approach to maliciously secure private set intersection. In: Advances in Cryptology – EUROCRYPT 2019, Part III. pp. 154–185 (2019)
28. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game. In: Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing. pp. 218–229. STOC '87 (1987)
29. Goldreich, O.: Foundations of cryptography: volume 2, basic applications. Cambridge University Press (2009)
30. Goyal, V., Li, H., Ostrovsky, R., Polychroniadou, A., Song, Y.: ATLAS: Efficient and scalable MPC in the honest majority setting. In: Advances in Cryptology – CRYPTO 2021, Part II. pp. 244–274 (2021)
31. Goyal, V., Liu, Y., Song, Y.: Communication-efficient unconditional MPC with guaranteed output delivery. In: Advances in Cryptology – CRYPTO 2019, Part II. pp. 85–114 (2019)
32. Goyal, V., Song, Y., Zhu, C.: Guaranteed output delivery comes free in honest majority MPC. In: Advances in Cryptology – CRYPTO 2020, Part II. pp. 618–646 (2020)
33. Halevi, S., Ishai, Y., Jain, A., Kushilevitz, E., Rabin, T.: Secure multiparty computation with general interaction patterns. In: Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science. pp. 157–168. ITCS '16 (2016)
34. Halevi, S., Lindell, Y., Pinkas, B.: Secure computation on the web: Computing without simultaneous interaction. In: Advances in Cryptology – CRYPTO 2011. pp. 132–150 (2011)
35. Hazay, C., Venkitasubramaniam, M.: Scalable multi-party private set-intersection. In: Public-Key Cryptography – PKC 2017. pp. 175–203 (2017)
36. Hirt, M., Maurer, U.: Robustness for free in unconditional multi-party computation. In: Advances in Cryptology – CRYPTO 2001. pp. 101–118 (2001)
37. Hirt, M., Maurer, U., Przydatek, B.: Efficient secure multi-party computation. In: Advances in Cryptology – ASIACRYPT 2000. pp. 143–161 (2000)
38. Hirt, M., Tschudi, D.: Efficient general-adversary multi-party computation. In: Advances in Cryptology – ASIACRYPT 2013, Part II. pp. 181–200 (2013)
39. Inbar, R., Omri, E., Pinkas, B.: Efficient scalable multiparty private set-intersection via garbled bloom filters. In: Security and Cryptography for Networks. pp. 235–252 (2018)

40. Ishai, Y., Kushilevitz, E.: Private simultaneous messages protocols with applications. In: Proceedings of the Fifth Israeli Symposium on Theory of Computing and Systems. pp. 174–183 (1997)
41. Ishai, Y., Kushilevitz, E.: Randomizing polynomials: A new representation with applications to round-efficient secure computation. In: Proceedings 41st Annual Symposium on Foundations of Computer Science. pp. 294–304 (2000)
42. Ishai, Y., Kushilevitz, E., Meldgaard, S., Orlandi, C., Paskin-Cherniavsky, A.: On the power of correlated randomness in secure computation. In: Theory of Cryptography. pp. 600–620 (2013)
43. Jukna, S.: Boolean Function Complexity. Springer, Berlin, Heidelberg, 1 edn. (2012)
44. Keller, H., Orlandi, C., Paskin-Cherniavsky, A., Ravi, D.: MPC with low bottleneck-complexity: Information-theoretic security and more. In: 4th Information-Theoretic Cryptography (ITC) Conference (2023), https://eprint.iacr.org/2023/683
45. Keller, M.: MP-SPDZ: A versatile framework for multi-party computation. In: Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security. pp. 1575–1590. CCS '20 (2020)
46. Kissner, L., Song, D.: Privacy-preserving set operations. In: Advances in Cryptology – CRYPTO 2005. pp. 241–257 (2005)
47. Kolesnikov, V., Matania, N., Pinkas, B., Rosulek, M., Trieu, N.: Practical multiparty private set intersection from symmetric-key techniques. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. pp. 1257–1272. CCS '17 (2017)
48. Kushilevitz, E., Lindell, Y., Rabin, T.: Information-theoretically secure protocols and security under composition. SIAM Journal of Computing **39**(5), 2090–2112 (2010)
49. Li, R., Wu, C.: An unconditionally secure protocol for multi-party set intersection. In: Applied Cryptography and Network Security. pp. 226–236 (2007)
50. Miyaji, A., Nishida, S.: A scalable multiparty private set intersection. In: Network and System Security. pp. 376–385 (2015)
51. Nevo, O., Trieu, N., Yanai, A.: Simple, fast malicious multiparty private set intersection. In: Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security. pp. 1151–1165. CCS '21 (2021)
52. Orlandi, C., Ravi, D., Scholl, P.: On the bottleneck complexity of mpc with correlated randomness. In: Public-Key Cryptography – PKC 2022, Part I. pp. 194–220 (2022)
53. Patra, A., Choudhary, A., Rangan, C.P.: Information theoretically secure multi party set intersection re-visited. In: Selected Areas in Cryptography. pp. 71–91 (2009)
54. Patra, A., Choudhary, A., Rangan, C.P.: Round efficient unconditionally secure mpc and multiparty set intersection with optimal resilience. In: Progress in Cryptology – INDOCRYPT 2009. pp. 398–417 (2009)
55. Rabin, T., Ben-Or, M.: Verifiable secret sharing and multiparty protocols with honest majority. In: Proceedings of the Twenty-First Annual ACM Symposium on Theory of Computing. pp. 73–85. STOC '89 (1989)
56. Sang, Y., Shen, H.: Privacy preserving set intersection protocol secure against malicious behaviors. In: Eighth International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT 2007). pp. 461–468 (2007)

57. Sang, Y., Shen, H.: Privacy preserving set intersection based on bilinear groups. In: Proceedings of the Thirty-First Australasian Conference on Computer Science - Volume 74. pp. 47–54. ACSC '08 (2008)
58. Shamir, A.: How to share a secret. Communications of the ACM **22**(11), 612–613 (1979)
59. Shoup, V.: A computational introduction to number theory and algebra. Cambridge university press (2009)
60. van Dijk, M., Gentry, C., Halevi, S., Vaikuntanathan, V.: Fully homomorphic encryption over the integers. In: Advances in Cryptology – EUROCRYPT 2010. pp. 24–43 (2010)
61. Vos, J., Conti, M., Erkin, Z.: Fast multi-party private set operations in the star topology from secure ANDs and ORs. Cryptology ePrint Archive, Paper 2022/721 (2022), https://eprint.iacr.org/2022/721
62. Yao, A.C.: Protocols for secure computations. In: Proceedings of the 23rd Annual Symposium on Foundations of Computer Science. pp. 160–164. SFCS '82 (1982)
63. Yao, A.C.: How to generate and exchange secrets. In: 27th Annual Symposium on Foundations of Computer Science (sfcs 1986). pp. 162–167 (1986)

Supplementary Material

A Secure Multiparty Computation

Definitions. Let n be a polynomial in a security parameter λ. We denote the set of n players by $\{\mathsf{P}_1, \dots, \mathsf{P}_n\}$, where P_i is the i-th player. Assume that each player P_i has a private input x_i from a finite set D. Let $\mathcal{F}(x_1, \dots, x_n) = (y_1, \dots, y_n)$ be an n-input/n-output randomized functionality. Let $\mathit{\Pi}$ be a protocol between n players. We assume the *preprocessing model.* That is, a protocol includes a joint distribution $\mathcal{D}$ over the Cartesian product $R_1 \times \cdots \times R_n$ of n sets, and each player P_i receives $r_i \in R_i$ before he decides his input, where $(r_1, \dots, r_n)$ is sampled from $R_1 \times \cdots \times R_n$ according to $\mathcal{D}$. We assume computationally unbounded adversaries who passively corrupt up to t players. (We do not consider active adversaries whose corrupted players deviate from protocols arbitrarily.) For a security parameter λ, a subset $T \subseteq [n]$ of size at most t and any input $\mathbf{x} = (x_i)_{i \in [n]}$, consider the following two processes:

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

$$\mathsf{Ideal}_{\mathcal{F},\mathsf{Sim}}(1^\lambda, T, \mathbf{x}) := (\mathsf{Sim}(1^\lambda, T, (x_i, y_i)_{i \in T}), (y_i)_{i \in [n]}).$$

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

$$\mathsf{Real}_{\mathit{\Pi}}(1^\lambda, T, \mathbf{x}) := ((\mathsf{View}_{\mathit{\Pi},i}(\mathbf{x}))_{i \in T}, (\mathsf{Output}_{\mathit{\Pi},i}(\mathbf{x}))_{i \in [n]}).$$

We say that $\mathit{\Pi}$ is a *t-secure MPC protocol for $\mathcal{F}$* if for any subset $T \subseteq [n]$ of size at most t and any input $\mathbf{x} = (x_i)_{i \in [n]}$, it holds that

$$\{\mathsf{Ideal}_{\mathcal{F},\mathsf{Sim}}(1^\lambda, T, \mathbf{x})\}_{\lambda \in \mathbb{N}} \approx \{\mathsf{Real}_{\mathit{\Pi}}(1^\lambda, T, \mathbf{x})\}_{\lambda \in \mathbb{N}}.$$

We simply say that $\mathit{\Pi}$ is *fully secure* if it is $(n-1)$-secure.

Let g be a deterministic function on D^n. Recall that we have defined $\mathcal{F}_g$ as a functionality such that $\mathcal{F}_g(\mathbf{x}) = (g(\mathbf{x}), \dots, g(\mathbf{x}))$. Then we have an equivalent definition of $\mathit{\Pi}$ being a secure MPC protocol for $\mathcal{F}_g$: $\mathit{\Pi}$ is a t-secure MPC protocol for $\mathcal{F}_g$ if and only if

Correctness. For any input $\mathbf{x}$, it holds that

$$\Pr\left[\exists i \in [n] : \mathsf{Output}_{\mathit{\Pi},i}(\mathbf{x}) \neq g(\mathbf{x})\right] = \mathsf{negl}(\lambda)\,;$$

Privacy. For any set $T \subseteq [n]$ of size at most t and any pair of inputs $\mathbf{x} = (x_i)_{i \in [n]}$, $\mathbf{w} = (w_i)_{i \in [n]}$ such that $(x_i)_{i \in T} = (w_i)_{i \in T}$ and $g(\mathbf{x}) = g(\mathbf{w})$, it holds that

$$\{(\mathsf{View}_{\mathit{\Pi},i}(\mathbf{x}))_{i \in T}\}_{\lambda \in \mathbb{N}} \approx \{(\mathsf{View}_{\mathit{\Pi},i}(\mathbf{w}))_{i \in T}\}_{\lambda \in \mathbb{N}}.$$

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

The above composition theorem assumes sequential composition, that is, protocols are invoked one after the other. For example, if a protocol Π internally invokes a sub-functionality $\mathcal{G}$ multiple (say, m) times and a protocol $\Pi_{\mathcal{G}}$ implements $\mathcal{G}$, the number of rounds of the composed protocol results in $\mathrm{Round}(\Pi) + m \cdot \mathrm{Round}(\Pi_{\mathcal{G}})$. On the other hand, it is desirable that a protocol is secure even under concurrent general composition since we can achieve better round complexity by running as many sub-protocols as possible in parallel. It is known that a protocol preserves security even under concurrent general composition if (and only if) it satisfies universal composability [12]. Although a protocol that is secure in the above stand-alone model does not satisfy universal composability in general, Kushilevitz, Lindell and Rabin [48] showed that if a protocol is secure in the stand-alone model and has a straight-line black-box simulator[5], then it is secure under concurrent general composition *with fixed inputs*. In the composition with fixed inputs, each player P_i receives a tuple of inputs $(x_i, w_{i1}, \ldots, w_{im})$ before the protocol execution begins, where P_i uses x_i as its input to a main protocol and uses w_{ij} as its input to the j-th sub-protocol. In other words, the inputs to sub-protocols are fixed before the execution of the main protocol begins.

All of the simulators for our protocols simulate messages that corrupted players see only from the input/output of a functionality, and their internal states just by performing the same computation as the real protocol execution. Thus, our simulators are all black-box and straight-line. Furthermore, in all of our protocols, the inputs to sub-protocols to be executed in parallel are independent of each other. Therefore, the concurrent composition of the sub-protocols securely realizes desired functionalities. For that reason, it is sufficient to provide security proofs of our protocols in the stand-alone setting.

B Unconditionally Secure Variant of the Protocol in [52]

In this section, we show an unconditionally secure variant of the protocol for symmetric functions [52]. To this end, we replace a garbled circuit used in their original protocol with an information-theoretically secure garbled circuit, which is also known as randomized encoding [41].

[5] A simulator is *black-box* if it uses only oracle access to a real-world adverasry. Such a simulator is *straight-line* if it interacts with the real-world adversary in essentially the same way as the real protocol execution.

Definition 1 (Randomized encoding). *Let $g : \{0,1\}^m \to \{0,1\}$ be a function. We say that a function $\hat{g} : \{0,1\}^m \times \{0,1\}^\rho \to \{0,1\}^k$ is a randomized encoding for g if it satisfies the following requirements:*

Correctness. *There exists a function $\mathsf{Dec} : \{0,1\}^k \to \{0,1\}$, called a decoder, such that for every $x \in \{0,1\}^m$ and $r \in \{0,1\}^\rho$, we have $\mathsf{Dec}(\hat{g}(x;r)) = g(x)$;*

Privacy. *There exists a randomized function Sim, called a simulator, such that for every $x \in \{0,1\}^m$, the distribution $\mathsf{Sim}(g(x))$ is identical to the distribution of $\hat{g}(x;r)$ induced by a uniformly random choice of r from $\{0,1\}^\rho$.*

We call k the output length of $\hat{g}$.

Let $h : \{0,1\}^n \to \{0,1\}$ be a symmetric function and $f : [0..n] \to \{0,1\}$ be the unique function such that $f(\sum_{i\in[n]} x_i) = h(x_1,\dots,x_n)$ for all $(x_1,\dots,x_n) \in \{0,1\}^n$. In the unconditionally secure variant of the protocol in [52], players first compute a masked sum $y := \sum_{i\in[n]} x_i + r \bmod (n+1)$ and then securely compute $g(Y) := f(Y-r)$ on input $Y = y$ using a randomized encoding for g, where r is a random element unknown to any player. More formally, in the preprocessing phase, a trusted third party chooses $r \leftarrow_{\$} \mathbb{Z}_{n+1}$ and an additive sharing $(r_i)_{i\in[n]}$ for r. In addition, the trusted party defines a function $g(\mathbf{y}) := f(Y-r)$, where $Y \in \mathbb{Z}_{n+1}$ is an input variable and $\mathbf{y} = (y_1,\dots,y_m) \in \{0,1\}^m$ is the binary representation of Y, i.e., $m := \lceil \log(n+1) \rceil$ and $Y = \sum_{j\in[m]} y_j 2^j$. Assume that g can be represented by a (fan-in-2) arithmetic formula of depth D over $\mathbb{F}_2 = \{0,1\}$. Following the construction in [19], the trusted party computes a randomized encoding $\hat{g}(\mathbf{y};R) = (\hat{g}_1(y_1;R),\dots,\hat{g}_m(y_m;R))$ for g, where $R \in \{0,1\}^\rho$ and each $\hat{g}_j(y_j;R)$ can be written as $\alpha_j(R)\cdot y_j + \beta_j(R)$ for some functions $\alpha_j, \beta_j : \{0,1\}^\rho \to \{0,1\}^{s_j}$. The currently best-known upper bound on the output length is $\sum_{j\in[m]} s_j = 2^{D+O(\sqrt{D})}$ [19]. The trusted party chooses a random string $R \in \{0,1\}^\rho$ and generates bit-wise additive shares $(\mathbf{a}_{ji})_{i\in[n]}$ (resp. $(\mathbf{b}_{ji})_{i\in[n]}$) of $\alpha_j(R)$ (resp. $\beta_j(R)$). Finally, the trusted party gives r_i, $(\mathbf{a}_{ji})_{j\in[m]}$, and $(\mathbf{b}_{ji})_{j\in[m]}$ to the i-th player P_i. In the online phase, each player P_i computes $x'_i := x_i + r_i$ and every player then obtains $Y = \sum_{i\in[n]} x'_i = r + \sum_{i\in[n]} x_i$ by using the protocol Π_{Sum}. They compute the binary representation $\mathbf{y} = (y_1,\dots,y_m)$ of Y, and $\mathbf{z}_{ji} := \mathbf{a}_{ji}\cdot y_j + \mathbf{b}_{ji}$ for all $j \in [m]$. Finally, they compute $\hat{g}_j(y_j;R) = \sum_{i\in[n]} \mathbf{z}_{ji}$ for all $j \in [m]$ via Π_{Sum} and obtain an output $g(\mathbf{y}) = f(Y-r) = h(x_1,\dots,x_n)$ from $\hat{g}(Y;R)$. The offline and online bottleneck complexities of the above protocol are $O(\sum_{j\in[m]} s_j) = 2^{D+O(\sqrt{D})}$. Therefore, to achieve bottleneck complexity $O(n^{1-\epsilon})$ for a constant $\epsilon > 0$, the unconditionally secure protocol needs to assume that the related function f is represented by an arithmetic formula of depth at most $(1-\epsilon)\log n$.

We show below that such symmetric functions only account for $o(1)$ fraction of all symmetric functions. Technically, it follows from the following fact:

Proposition 1. *For any $\delta > 0$, there are $(1-\delta)2^{2^m}$ functions $g : \{0,1\}^m \to \{0,1\}$ which require arithmetic formulas of depth at least $\log(2^m - \log\delta^{-1}) - O(\log\log m)$. In particular, setting $\delta = 2^{-m}$, all but $2^{-m} = o(1)$ fraction of* *functions require arithmetic formulas of depth at least $m - O(\log\log m) = (1 - o(1))m$.*

Proof. This proof is inspired by that of [43, Theorem 1.23]. We upper bound the total number of arithmetic formulas with ℓ leaves. For ease of calculation, we assume that a formula is represented by a full binary tree, i.e., every node has either 0 or 2 children, and that every node is assigned m input literals (the x_i's), two types of gates (addition and multiplication), or their negations $(1 - x_i,\ 1 - (a + b)$ or $1 - a \cdot b)$. A full binary tree with ℓ leaves has $2\ell - 1$ nodes and the number of all full binary trees with $2\ell - 1$ nodes is at most $4^{2\ell-1}$. For each such tree, there are at most $(2m + 4)^{2\ell-1}$ possibilities to turn it into an arithmetic formula. Hence, the total number of arithmetic formulas with ℓ leaves is at most $4^{2\ell-1}(2m + 4)^{2\ell-1} \leq (9m)^{2\ell}$ for $m \geq 16$. Set $\ell = \lfloor (2^m - \log\delta^{-1})/(2\log(9m)) \rfloor$. Then, the number of functions that can be computed by arithmetic formulas with leafsize ℓ is at most $(9m)^{2\ell} \leq \delta \cdot 2^{2^m}$. This implies that the $1 - \delta$ fraction of all functions require arithmetic formulas with leafsize at least $\ell \geq (2^m - \log\delta^{-1})/(2\log(9m))$. Therefore, these functions require formulas of depth at least $\log\ell \geq \log(2^m - \log\delta^{-1}) - O(\log\log m)$. □

Since $m = \lceil \log(n + 1) \rceil$ in our setting, for all but $o(1)$ fraction of symmetric functions h, the related functions f require arithmetic formulas of depth at least $(1 - o(1))\log n$. In particular, the above unconditionally secure protocol can only compute $O(n^{-1}) = o(1)$ fraction of all symmetric functions with bottleneck complexity $O(n^{1-\epsilon})$ for a constant ϵ.

C Deferred Proofs

C.1 Protocols for Inner Products

Proposition 2. *The protocol Π_{IP} described in Fig. 4 is a fully secure MPC protocol Π_{IP} for $\mathcal{F}_{\mathsf{IP}}$ in the $\{\mathcal{F}_{\mathsf{Sum}}, \mathcal{F}_{\mathsf{Mult}}\}$-hybrid model.*

Proof. The correctness follows from

$$\begin{aligned} z &= \sum_{i\in[n]} \sum_{j\in[m]} w_i^{(j)} \\ &= \sum_{j\in[m]} \left(x_1^{(j)} + \cdots + x_n^{(j)}\right)\left(y_1^{(j)} + \cdots + y_n^{(j)}\right) \\ &= \sum_{j\in[m]} \mathbf{x}[j] \cdot \mathbf{y}[j] \\ &= \langle \mathbf{x}, \mathbf{y} \rangle \end{aligned}$$

where $\mathbf{x}[j]$ and $\mathbf{y}[j]$ are the j-th elements of $\mathbf{x}$ and $\mathbf{y}$, respectively. The second equality follows from the functionality of $\mathcal{F}_{\mathsf{Mult}}$.

We show the privacy of Π_{IP}. Let $T \subseteq [n]$ be the set of corrupted players. Let $H = \overline{T}$ be the set of honest players. In the $\mathcal{F}_{\mathsf{Mult}}$-hybrid model, corrupted players' view at Step 1 (including their correlated randomness for $\mathcal{F}_{\mathsf{Mult}}$) only contains their inputs $(\mathbf{x}_i, \mathbf{y}_i)_{i \in T}$ and their outputs $(w_i^{(j)})_{i \in T, j \in [m]}$ of $\mathcal{F}_{\mathsf{Mult}}$. We also have that for any $j \in [m]$, $(w_i^{(j)})_{i \in [n]}$ is uniformly distributed over $\mathbb{F}^n$ conditioned on $\sum_{i \in [n]} w_i^{(j)} = (x_1^{(j)} + \cdots + x_n^{(j)})(y_1^{(j)} + \cdots + y_n^{(j)})$. In particular, $(w_i^{(j)})_{i \in T, j \in [m]}$ are independent and identically distributed according to the uniform distribution over $\mathbb{F}$. The corrupted players' views at Step 3 can be simulated from z and $(z_i)_{i \in T}$ by choosing $(z_i)_{i \in H}$ uniformly at random from $\mathbb{F}^{|H|}$ conditioned on $\sum_{i \in H} z_i = z - \sum_{i \in T} z_i$. Since z_i is locally computed from $(w_i^{(j)})_{j \in [m]}$, we conclude that the corrupted players' views are simulated from their inputs $(\mathbf{x}_i, \mathbf{y}_i)_{i \in T}$ and the output z. □

Functionality $\mathcal{F}'_{\mathsf{IP}}((\mathbf{x}_i, \mathbf{y}_i)_{i \in [n]})$

1. $\mathcal{F}'_{\mathsf{IP}}$ receives vectors $\mathbf{x}_i, \mathbf{y}_i \in \mathbb{F}^m$ from each player P_i.
2. $\mathcal{F}'_{\mathsf{IP}}$ computes $\mathbf{x} = \sum_{i \in [n]} \mathbf{x}_i$, $\mathbf{y} = \sum_{i \in [n]} \mathbf{y}_i$ and $z = \langle \mathbf{x}, \mathbf{y} \rangle$.
3. $\mathcal{F}'_{\mathsf{IP}}$ runs $(z_i)_{i \in [n]} \leftarrow \mathsf{Additive}_{\mathbb{F}}(z)$.
4. $\mathcal{F}'_{\mathsf{IP}}$ gives z_i to each player P_i.

Protocol Π'_{IP}

Input. Each player P_i has $\mathbf{x}_i = (x_i^{(j)})_{j \in [m]} \in \mathbb{F}^m$ and $\mathbf{y}_i = (y_i^{(j)})_{j \in [m]} \in \mathbb{F}^m$.

Output. Each player P_i obtains z_i, where $(z_i)_{i \in [n]} \leftarrow \mathcal{F}'_{\mathsf{IP}}((\mathbf{x}_i, \mathbf{y}_i)_{i \in [n]})$.

Protocol.

1. For each $j \in [m]$, players obtain $(w_i^{(j)})_{i \in [n]} \leftarrow \mathcal{F}_{\mathsf{Mult}}((x_i^{(j)}, y_i^{(j)})_{i \in [n]})$.
2. Each player P_i outputs $z_i = \sum_{j \in [m]} w_i^{(j)} \in \mathbb{F}$.

Fig. 9. The functionality $\mathcal{F}'_{\mathsf{IP}}$ and a protocol Π'_{IP} implementing it

Proposition 3. *The protocol Π'_{IP} described in Fig. 9 is a fully secure MPC protocol Π'_{IP} for $\mathcal{F}'_{\mathsf{IP}}$ in the $\mathcal{F}_{\mathsf{Mult}}$-hybrid model.*

Proof. In the $\mathcal{F}_{\mathsf{Mult}}$-hybrid model, it holds that for all $j \in [m]$, $(w_i^{(j)})_{i \in [n]}$ is a vector of random additive shares for $\mathbf{x}[j] \cdot \mathbf{y}[j]$, where $\mathbf{x}[j]$ and $\mathbf{y}[j]$ are the j-th element of $\mathbf{x}$ and $\mathbf{y}$, respectively. Thus, $(z_i)_{i \in [n]}$ is a vector of random additive shares for $\sum_{j \in [m]} \mathbf{x}[j] \cdot \mathbf{y}[j] = z$, which implies that the outputs of Π'_{IP} correctly follows the distribution of the outputs of $\mathcal{F}'_{\mathsf{IP}}$.

Let T be the set of corrupted players. In the $\mathcal{F}_{\mathsf{Mult}}$-hybrid model, the joint view of the corrupted players at Step 1 only contains their inputs $(x_i^{(j)}, y_i^{(j)})_{i \in T}$

and outputs $(w_i^{(j)})_{i\in T}$ of $\mathcal{F}_{\mathsf{Mult}}$. Note that $(w_i^{(j)})_{i\in T}$ are just uniformly random elements. We can therefore construct a simulator for Π'_{IP} in the $\mathcal{F}_{\mathsf{Mult}}$-hybrid model as follows: On input $(\mathbf{x}_i, \mathbf{y}_i)_{i\in T}$ and $(z_i)_{i\in T}$, it chooses $(w_i^{(j)})_{i\in T, j\in[m]}$ uniformly at random conditioned on $z_i = \sum_{j\in[m]} w_i^{(j)}$ for all $i \in T$. It outputs $(w_i^{(j)})_{i\in T, j\in[m]}$, $(\mathbf{x}_i, \mathbf{y}_i)_{i\in T}$ and $(z_i)_{i\in T}$. □

C.2 Proof of Theorem 2

We have shown the correctness. We prove the privacy of Π'_{Sym}. Let $T \subseteq [n]$ be the set of corrupted players. Let $H = \overline{T}$ be the set of honest players and fix an honest player $j \in H$. In the $\mathcal{F}_{\mathsf{IP}}$-hybrid model, corrupted players' view at Step 5 (including their correlated randomness for $\mathcal{F}_{\mathsf{IP}}$) only contains their inputs $(\mathbf{a}_i, \mathbf{b}_i)_{i\in T}$ to $\mathcal{F}_{\mathsf{IP}}$ and the output $z = h(x_1, \ldots, x_n)$. Since every $(\mathbf{a}_i, \mathbf{b}_i)$ is locally computed from y and $(\mathbf{c}_i, \mathbf{d}_i)$, it is sufficient to show that the joint distribution of the following elements is simulated from $(x_i)_{i\in T}$ and z:

Correlated randomness. $(r_i, \mathbf{c}_i, \mathbf{d}_i)$ for all $i \in T$;
Online messages. $y_i = x_i - r_i$ for all $i \in H$.

Let $\mathbf{x}, \mathbf{x}' \in \{0,1\}^n$ be any pair of inputs such that

$$x_i = x'_i \ (\forall i \in T) \text{ and } h(x_1, \ldots, x_n) = h(x'_1, \ldots, x'_n).$$

It is sufficient to prove that the distribution of the above elements during the execution of Π'_{Sym} on input $\mathbf{x}$ is identical to that on input $\mathbf{x}'$. To show the equivalence of the distributions, we show a bijection between the random strings used by Π'_{Sym} on input $\mathbf{x}$ and the random strings used by Π'_{Sym} on input $\mathbf{x}'$ such that the above values are the same under this bijection. Note that the randomness of Π'_{Sym} on input $\mathbf{x}$ is uniformly distributed over a set

$$\mathcal{S} = \left\{ (r_i, \mathbf{c}_i, \mathbf{d}_i)_{i\in[n]} : \sum_{i\in[n]} \mathbf{c}_i = \mathbf{e}_u, \sum_{i\in[n]} \mathbf{d}_i = \mathbf{e}_v, \text{ where } (u,v) = \phi(\sum_{i\in[n]} r_i) \right\}$$

The bijection maps the randomness $(r_i, \mathbf{c}_i, \mathbf{d}_i)_{i\in[n]}$ of Π'_{Sym} on input $\mathbf{x}$ to the randomness $(r'_i, \mathbf{c}'_i, \mathbf{d}'_i)_{i\in[n]}$ of Π'_{Sym} on input $\mathbf{x}'$ in such a way that

$$r'_i = \begin{cases} r_i, & \text{if } i \in T, \\ r_i + x'_i - x_i, & \text{if } i \in H, \end{cases}$$

$$\mathbf{c}'_i = \begin{cases} \mathbf{c}_i, & \text{if } i \neq j, \\ \mathbf{c}_j + \mathbf{e}_{u'} - \mathbf{e}_u, & \text{if } i = j, \end{cases}$$

$$\mathbf{d}'_i = \begin{cases} \mathbf{d}_i, & \text{if } i \neq j, \\ \mathbf{d}_j + \mathbf{e}_{v'} - \mathbf{e}_v, & \text{if } i = j, \end{cases}$$

where we write $r := \sum_{i\in[n]} r_i$ and $r' := \sum_{i\in[n]} r'_i$, and define $(u,v) := \phi(r)$ and $(u',v') := \phi(r')$. We first see that the image is indeed a consistent random string, i.e., $(r'_i, \mathbf{c}'_i, \mathbf{d}'_i)_{i\in[n]} \in \mathcal{S}$. Observe that

$$\sum_{i\in[n]} \mathbf{c}'_i = \sum_{i\in[n]} \mathbf{c}_i + \mathbf{e}_{u'} - \mathbf{e}_u \text{ and } \sum_{i\in[n]} \mathbf{d}'_i = \sum_{i\in[n]} \mathbf{d}_i + \mathbf{e}_{v'} - \mathbf{e}_v.$$

Therefore, $\sum_{i\in[n]} \mathbf{c}'_i = \mathbf{e}_{u'}$ and $\sum_{i\in[n]} \mathbf{d}'_i = \mathbf{e}_{v'}$, where $(u',v') = \phi(r')$, if and only if $\sum_{i\in[n]} \mathbf{c}_i = \mathbf{e}_u$ and $\sum_{i\in[n]} \mathbf{d}_i = \mathbf{e}_v$, where $(u,v) = \phi(r)$. Next, the above map is indeed a bijection since it has the inverse

$$r_i = \begin{cases} r'_i, & \text{if } i \in T, \\ r'_i + x_i - x'_i, & \text{if } i \in H, \end{cases}$$

$$\mathbf{c}_i = \begin{cases} \mathbf{c}'_i, & \text{if } i \neq j, \\ \mathbf{c}'_j + \mathbf{e}_u - \mathbf{e}_{u'}, & \text{if } i = j, \end{cases}$$

$$\mathbf{d}_i = \begin{cases} \mathbf{d}'_i, & \text{if } i \neq j, \\ \mathbf{d}'_j + \mathbf{e}_v - \mathbf{e}_{v'}, & \text{if } i = j. \end{cases}$$

Clearly, this bijection does not change the correlated randomness $(r_i, \mathbf{c}_i, \mathbf{d}_i)_{i\in T}$ of T. Since $x'_i - r'_i = x'_i - (r_i + x'_i - x_i) = x_i - r_i$ for $i \in H$, it does not change the online messages from H.

C.3 Proof of Theorem 4

The protocol $\Pi_{\mathsf{CheckZero},\mathbb{K}}$ is described in Fig. 10. First, we prove the correctness of $\Pi_{\mathsf{CheckZero},\mathbb{K}}$. Let $\mathbf{x} \in \mathbb{F}^n$ be any input. Since $r = \sum_{i\in[n]} r_i$, it holds that $y = r + \sum_{i\in[n]} x_i$. Since $A = \sum_{i\in[n]} A_i$ and $B = \sum_{i\in[n]} B_i$, it also holds that

$$Z = \sum_{i\in[n]} Z_i = Ay + B.$$

Therefore, $Z = S$ holds if and only if $y = r$ (i.e., $\sum_{i\in[n]} x_i = 0$) or $A = 0$. If $\mathcal{F}_{\mathsf{CheckZero},\mathbb{F}}(\mathbf{x}) = 0$, the protocol $\Pi_{\mathsf{CheckZero},\mathbb{K}}$ outputs 0 with probability 1. If $\mathcal{F}_{\mathsf{CheckZero},\mathbb{F}}(\mathbf{x}) = 1$, it outputs 1 except with negligible probability $\Pr[A = 0] = |\mathbb{K}|^{-1} \leq 2^{-\lambda}$.

Next, we prove the privacy of $\Pi_{\mathsf{CheckZero},\mathbb{K}}$. Let $T \subseteq [n]$ be the set of corrupted players and $H = \overline{T}$ be the set of honest players. It is sufficient to show that there exists a simulator which can simulate the joint distribution of the following elements:

Correlated randomness. (A_i, B_i, r_i) for all $i \in T$ and $S = Ar + B$;
Online messages. $y_i = x_i + r_i$ and $Z_i = A_i y + B_i$ for all $i \in H$.

To this end, we analyze the distribution of the above elements during the execution of $\Pi_{\mathsf{CheckZero},\mathbb{K}}$ on input $\mathbf{x} = (x_i)_{i\in[n]}$. We can see that $(A_i, B_i, r_i)_{i\in T}$

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

1. $\mathcal{F}_{\mathsf{CheckZero},\mathbb{F}}$ receives a field element $x_i \in \mathbb{F}$ from each player P_i.
2. $\mathcal{F}_{\mathsf{CheckZero},\mathbb{F}}$ sets $b = 0$ if $\sum_{i\in[n]} x_i = 0$ in $\mathbb{F}$, and $b = 1$ otherwise.
3. $\mathcal{F}_{\mathsf{CheckZero},\mathbb{F}}$ gives b to every player P_i.

Protocol $\Pi_{\mathsf{CheckZero},\mathbb{K}}$

Input. Each player P_i has $x_i \in \mathbb{F}$.

Output. Every player obtains $b = \mathcal{F}_{\mathsf{CheckZero},\mathbb{F}}((x_i)_{i\in[n]})$.

Setup.

1. Let $r \leftarrow\!\!\$\, \mathbb{F}$ and $(r_i)_{i\in[n]} \leftarrow \mathsf{Additive}_{\mathbb{F}}(r)$.
2. Let $A, B \leftarrow\!\!\$\, \mathbb{K}$, $(A_i)_{i\in[n]} \leftarrow \mathsf{Additive}_{\mathbb{K}}(A)$ and $(B_i)_{i\in[n]} \leftarrow \mathsf{Additive}_{\mathbb{K}}(B)$.
3. Set $S = Ar + B \in \mathbb{K}$.
4. Each player P_i receives (A_i, B_i, r_i, S).

Protocol.

1. Each player P_i computes $y_i = x_i + r_i \in \mathbb{F}$.
2. Players obtain $y = \mathcal{F}_{\mathsf{Sum}}((y_i)_{i\in[n]}) \in \mathbb{F}$.
3. Each player P_i computes $Z_i = A_i y + B_i \in \mathbb{K}$.
4. Players obtain $Z = \mathcal{F}_{\mathsf{Sum}}((Z_i)_{i\in[n]}) \in \mathbb{K}$.
5. Each player P_i outputs $b = 0$ if $Z = S$ and outputs $b = 1$ otherwise.

Fig. 10. The functionality $\mathcal{F}_{\mathsf{CheckZero},\mathbb{F}}$ and a protocol $\Pi_{\mathsf{CheckZero},\mathbb{K}}$ implementing it

and $(y_i)_{i\in H}$ are independent and distributed according to the uniform distributions over $(\mathbb{K}\times\mathbb{K}\times\mathbb{F})^{|T|}$ and $\mathbb{F}^{|H|}$, respectively, since $(r_i)_{i\in H}$ are independent of $(A_i, B_i, r_i)_{i\in T}$. Fix any $(A_i, B_i, r_i)_{i\in T}$ and any $(r_i)_{i\in H}$. Then $y = \sum_{i\in[n]}(x_i+r_i)$ and $r = \sum_{i\in[n]} r_i$ are also fixed. The remaining independent random variables are $(A_i, B_i)_{i\in H}$. The distribution of $(Z_i)_{i\in H}$ and S is determined by the following equation:

$$\begin{bmatrix}\mathbf{Z}_H \\ S\end{bmatrix} = \mathbf{M}\begin{bmatrix}\mathbf{A}_H \\ \mathbf{B}_H\end{bmatrix} + \begin{bmatrix}\mathbf{0}_{|H|} \\ W\end{bmatrix},$$

where $\mathbf{Z}_H = (Z_i)_{i\in H}$, $\mathbf{A}_H = (A_i)_{i\in H}$, $\mathbf{B}_H = (B_i)_{i\in H}$, $W := \sum_{i\in T}(A_i r + B_i)$, and

$$\mathbf{M} := \begin{bmatrix} y\cdot\mathbf{I}_{|H|} & \mathbf{I}_{|H|} \\ r\cdot\mathbf{1}_{|H|}^{\top} & \mathbf{1}_{|H|}^{\top}\end{bmatrix} = \begin{bmatrix} y & & & 1 & & \\ & \ddots & & & \ddots & \\ & & y & & & 1 \\ r & \cdots & r & 1 & \cdots & 1\end{bmatrix} \in \mathbb{F}^{(|H|+1)\times(2|H|)}.$$

If $\sum_{i\in[n]} x_i \neq 0$, then $y \neq r$ and $\mathbf{M}$ is of full rank. Hence $(Z_i)_{i\in H}$ and S are independent and uniformly distributed over $\mathbb{K}$. If $\sum_{i\in[n]} x_i = 0$, then $y = r$. In this case, $(Z_i)_{i\in H}$ and S are uniformly distributed over a subspace of $\mathbb{K}^{|H|+1}$ consisting of $(Z'_i)_{i\in H}$ and S' such that $S' = W + \sum_{i\in H} Z'_i$, i.e.,

$$S' = \sum_{i\in T} A_i \sum_{j\in[n]}(x_j + r_j) + \sum_{i\in T} B_i + \sum_{i\in H} Z'_i.$$

From the above observation, the privacy follows from the following simulator Sim: On input corrupted players' inputs $(x_i)_{i\in T} \in \mathbb{F}^{|T|}$ and a bit $b \in \{0,1\}$:

- If $b = 1$, Sim chooses $(A_i, B_i, r_i)_{i\in T}$ and $(y_i)_{i\in H}$ uniformly at random from $(\mathbb{K}\times\mathbb{K}\times\mathbb{F})^{|T|}$ and $\mathbb{F}^{|H|}$, respectively. It also chooses S and $(Z_i)_{i\in H}$ uniformly at random from $\mathbb{K}$ and $\mathbb{K}^{|H|}$, respectively. It then outputs all of them.
- If $b = 0$, Sim chooses $(A_i, B_i, r_i)_{i\in T}$ and $(y_i)_{i\in H}$ uniformly at random from $(\mathbb{K}\times\mathbb{K}\times\mathbb{F})^{|T|}$ and $\mathbb{F}^{|H|}$, respectively. It also chooses $(Z_i)_{i\in H}$ uniformly at random from $\mathbb{K}^{|H|}$ and computes

$$S = \sum_{i\in T} A_i \left(\sum_{j\in T}(x_j + r_j) + \sum_{j\in H} y_j\right) + \sum_{i\in T} B_i + \sum_{i\in H} Z_i.$$

  It then outputs all of them.

Finally, since players also need to receive correlated randomness of two executions of a protocol implementing $\mathcal{F}_{\mathsf{Sum}}$, the offline communication complexity is

$$\mathrm{BC}_{\mathrm{off}}(\mathit{\Pi}_{\mathsf{CheckZero},\mathbb{K}}) = O(\log|\mathbb{K}|) = O(\max\{\lambda, \log|\mathbb{F}|\}).$$

It has online communication complexity $\mathrm{BC}_{\mathrm{on}}(\mathit{\Pi}_{\mathsf{CheckZero},\mathbb{K}}) = O(\max\{\lambda, \log|\mathbb{F}|\})$ and round complexity $\mathrm{Round}(\mathit{\Pi}_{\mathsf{CheckZero},\mathbb{K}}) = O(\log n)$.

C.4 Proof of Theorem 5

First, we prove the correctness of Π_{PSI}. Let $(X_i)_{i\in[n]} \in (2^U)^n$ be any input. Let $h \in [m]$. At Step 1 of the protocol, the h-th entry $\mathbf{B}_i[h]$ of $\mathbf{B}_i$ is 0 if and only if $h \in \{H_1(x), \ldots, H_k(x)\}$ for some $x \in X_i$, where $H_1, \ldots, H_k$ are hash functions associated with the Bloom filter $\mathcal{BF}$. Since the characteristic p of $\mathbb{F}$ is larger than n, at Step 2, the h-th entry $\mathbf{V}[h]$ of $\mathbf{V}$ is 0 if and only if $h \in \bigcup_{x\in X_i}\{H_1(x), \ldots, H_k(x)\}$ for all $i \in [n]$. Note that $\langle \mathbf{V}, \mathbf{BF}(\{x\})\rangle = \sum_{h\in\{H_1(x),\ldots,H_k(x)\}} \mathbf{V}[h]$ for any $x \in U$. Since each $\mathbf{V}[h]$ is an integer between 0 and n and the characteristic p is larger than nk, we have that $\langle \mathbf{V}, \mathbf{BF}(\{x\})\rangle = 0$ if and only if $\mathbf{V}[h] = 0$ for all $h \in \{H_1(x), \ldots, H_k(x)\}$, which is equivalent to the condition that

$$\forall i \in [n] : \ \{H_1(x), \ldots, H_k(x)\} \subseteq \bigcup_{x'\in X_i} \{H_1(x'), \ldots, H_k(x')\}. \tag{6}$$

The condition (6) is then equivalent to

$$\forall i \in [n] : \ \mathsf{Check}(\mathbf{BF}(X_i), x) = 1. \tag{7}$$

At Step 4 of the protocol, players compute additive shares for $\mathbf{BF}(\{x^{(j)}\})$ for all $j \in [s]$, where $X_n = \{x^{(1)}, \ldots, x^{(s)}\}$. At Step 5, $(y_i^{(j)})_{i\in[n]}$ is an additive sharing for $\langle \mathbf{V}, \mathbf{BF}(\{x^{(j)}\})\rangle$. Therefore, for each $j \in [s]$, $z^{(j)} = 0$ if and only if $\langle \mathbf{V}, \mathbf{BF}(\{x^{(j)}\})\rangle = 0$, i.e., $x = x^{(j)}$ satisfies the condition (7). Since the protocol invokes $\mathcal{F}_{\mathsf{CheckZero},\mathbb{F}}$ only $s = \mathsf{poly}(\lambda)$ times, we conclude that $Z = \{x^{(j)}\}_{j\in[s]:z^{(j)}=0}$ is equal to the output of $\mathcal{F}_{\mathsf{PSI},\mathcal{BF}}$ with probability at least $1 - \mathsf{negl}(\lambda)$. The correctness thus holds.

Next, we prove the privacy of Π_{PSI}. Let T be the set of corrupted players. The interaction occurs only at Steps 5–7 in the online phase. In the $\{\mathcal{F}'_{\mathsf{IP}}, \mathcal{F}_{\mathsf{CheckZero},\mathbb{F}}\}$-hybrid model, corrupted players' view (including their correlated randomness) only contains the following elements:

$$(\mathbf{V}_i)_{i\in T}, \ (\mathbf{W}_i^{(j)})_{i\in T, j\in[s]}, \ (y_i^{(j)})_{i\in T, j\in[s]}, \ (z^{(j)})_{j\in[s]}, \ Z.$$

First, the vectors $(\mathbf{V}_i)_{i\in T}$, $(\mathbf{W}_i^{(j)})_{i\in T,j\in[s]}$ are simulated from $(X_i)_{i\in T}$ only since they are obtained by local computation. Secondly, it follows from the functionality of $\mathcal{F}'_{\mathsf{IP}}$ that the elements $(y_i^{(j)})_{i\in T,j\in[s]}$ are independent and identically distributed according to the uniform distribution over $\mathbb{F}$. Thirdly, we have seen that $z^{(j)} = 0$ if and only if $x^{(j)} \in Z$. If P_n is corrupted, i.e., $\mathsf{P}_n \in T$, then $(z^{(j)})_{j\in[s]}$ can be simulated from X_n and Z. If $\mathsf{P}_n \notin T$, then from the corrupted players' viewpoint, $(z^{(j)})_{j\in[s]}$ is a random sequence of s bits such that the number of 1's is equal to $|Z|$, since $(x^{(j)})_{j\in[s]}$ is a random permutation of the elements of X_n. In conclusion, corrupted players' view is simulated from the output Z and their inputs $(X_i)_{i\in T}$ only.

Finally, since players also need to receive correlated randomness for s executions of a protocol implementing $\mathcal{F}'_{\mathsf{IP}}$ and for s executions of a protocol implementing $\mathcal{F}_{\mathsf{CheckZero},\mathbb{F}}$, the offline complexity is

$$\begin{aligned}\mathrm{BC}_{\mathsf{off}}(\mathit{\Pi}_{\mathsf{PSI}}) &= O(m\log|\mathbb{F}|) + O(sm\log|\mathbb{F}|) + s\cdot O(m\log|\mathbb{F}|) + s\cdot O(\log|\mathbb{K}|)\\ &= O(sm\log(nk)) + O(s\log 2^{\lambda})\\ &= O(sm\log(n\lambda) + s\lambda).\end{aligned}$$

Note that interaction occurs only at Steps 5–7 in the online phase. Steps 5 and 6 invokes $\mathcal{F}'_{\mathsf{IP}}$ and $\mathcal{F}_{\mathsf{CheckZero},\mathbb{F}}$ s times, respectively. In Step 7, the n-th player broadcasts at most s elements, each of $O(\log|U|)$ bits. Thus, the protocol $\mathit{\Pi}_{\mathsf{PSI}}$ has online bottleneck complexity

$$\begin{aligned}\mathrm{BC}_{\mathrm{on}}(\mathit{\Pi}_{\mathsf{PSI}}) &= s\cdot O(m\log|\mathbb{F}|) + s\cdot O(\log|\mathbb{K}|) + O(s\log|U|)\\ &= O(sm\log(n\lambda) + s\lambda + s\log|U|);\end{aligned}$$

and round complexity $\mathrm{Round}(\mathit{\Pi}_{\mathsf{PSI}}) = O(\log n)$.