Non-Interactive Secure Multiparty Computation for Symmetric Functions, Revisited: More Efficient Constructions and Extensions

Reo Eriguchi[1,2], Kazuma Ohara[2], Shota Yamada[2], and Koji Nuida[2,3]

[1] The University of Tokyo, Tokyo, Japan
reo-eriguchi@g.ecc.u-tokyo.ac.jp
[2] National Institute of Advanced Industrial Science and Technology, Tokyo, Japan
{ohara.kazuma,yamada-shota}@aist.go.jp
[3] Kyushu University, Fukuoka, Japan
nuida@imi.kyushu-u.ac.jp

Abstract. Non-interactive secure multiparty computation (NIMPC) is a variant of secure computation which allows each of n players to send only a single message depending on his input and correlated randomness. Abelian programs, which can realize any symmetric function, are defined as functions on the sum of the players' inputs over an abelian group and provide useful functionalities for real-world applications. We improve and extend the previous results in the following ways:

- We present NIMPC protocols for abelian programs that improve the best known communication complexity. If inputs take any value of an abelian group $\mathbb{G}$, our protocol achieves the communication complexity $O(|\mathbb{G}|(\log|\mathbb{G}|)^2)$ improving $O(|\mathbb{G}|^2n^2)$ of Beimel et al. (Crypto 2014). If players are limited to inputs from subsets of size at most d, our protocol achieves $|\mathbb{G}|(\log|\mathbb{G}|)^2(\max\{n,d\})^{(1+o(1))t}$ where t is a corruption threshold. This result improves $|\mathbb{G}|^3(nd)^{(1+o(1))t}$ of Beimel et al. (Crypto 2014), and even $|\mathbb{G}|^{\log n+O(1)}n$ of Benhamouda et al. (Crypto 2017) if $t=o(\log n)$ and $|\mathbb{G}|=n^{\Theta(1)}$.
- We propose for the first time NIMPC protocols for linear classifiers that are more efficient than those obtained from the generic construction.
- We revisit a known transformation of Benhamouda et al. (Crypto 2017) from Private Simultaneous Messages (PSM) to NIMPC, which we repeatedly use in the above results. We reveal that a sub-protocol used in the transformation does not satisfy the specified security. We also fix their protocol with only constant overhead in the communication complexity. As a byproduct, we obtain an NIMPC protocol for indicator functions with asymptotically optimal communication complexity with respect to the input length.

1 Introduction

Secure multiparty computation enables n players P_i ($i \in [n] := \{1,2,\ldots,n\}$), each holding an input $x_i \in \mathcal{X}_0$, to jointly compute a function while keeping

their inputs as secret as possible. NIMPC (Non-Interactive secure Multi-Party Computation) [2, 3] is a variant of secure computation with a restricted interaction pattern, which assumes an external output player called an evaluator and allows each player to send only a single message depending on his input and pre-distributed correlated randomness to the evaluator.

Since it requires no interaction between the players, this model is especially well suited to a situation where the players cannot simultaneously participate in a protocol due to physical limitations. In spite of its limitations, NIMPC still provides useful functionalities in real-world scenarios such as voting, auctions, and statistical surveys using histograms. NIMPC is also of theoretical interest due to its various applications to other important models of secure computation [10, 4, 11].

To define the notion of security, an adversary is supposed to collude with a set of players $C \subseteq [n]$ as well as the evaluator. In NIMPC for a function h, it is impossible to prevent him from computing h on all possible inputs of the corrupted players combined with the inputs of the honest players. More formally, for the inputs of the honest players $x_{\overline{C}} = (x_i)_{i \in \overline{C}}$, the adversary can always evaluate the function $h|_{\overline{C}, x_{\overline{C}}}(x_C) = h(x_C, x_{\overline{C}})$ for all possible $x_C = (x_i)_{i \in C}$, which is called the residual function [12]. Thus, the security requirement of NIMPC is that the adversary learns the residual function $h|_{\overline{C}, x_{\overline{C}}}$ and nothing more. An NIMPC protocol is called t-robust if it can withstand collusion of at most t players and the evaluator. If $t = n$, we say that it is fully robust. The efficiency of NIMPC protocols is measured by the communication complexity defined as the maximum bit length of randomness and messages.

Fully robust NIMPC for the class of all the functions with input domain $\mathcal{X}_0^n$ is known to be possible [3, 17, 15, 1]. However, due to the lower bound [17], it necessarily has the communication complexity proportional to $|\mathcal{X}_0|^n$, which is very inefficient when n is large. Therefore, it is important to construct efficient NIMPC protocols for specific functions of practical use. The aim of this paper is reducing the communication complexity as much as possible and specifically, making it as close as possible to the lower bound [17].

Above all, symmetric functions realize useful functionalities including voting and statistical surveys using histograms. The notion of abelian programs is a generalization of symmetric functions introduced in [3]. Technically, an abelian program h takes n elements from an abelian group $\mathbb{G}$ as inputs and outputs $h(x_1, \ldots, x_n) = f(\sum_{i \in [n]} x_i)$ for some function $f : \mathbb{G} \to \{0, 1\}$.

The authors of [3] propose a fully robust NIMPC protocol with communication complexity $O(|\mathbb{G}|^2 n^2)$ for abelian programs allowing inputs to take any value of $\mathbb{G}$. Since NIMPC protocols do not satisfy the same level of robustness in general if players are limited to inputs from smaller domains, they also propose a t-robust protocol with communication complexity $|\mathbb{G}|^3 (nd)^{t+O(1)}$ for abelian programs with input domains of size at most d. Benhamouda, Krawczyk, and Rabin [6] construct a fully robust protocol with communication complexity $|\mathbb{G}|^{\log n + O(1)} n$ when input spaces are arbitrary subsets. However, from the viewpoint of the lower bound $n^{-1}|\mathbb{G}|$ [17], there is still room for improvement espe-

cially in the exponent with respect to $|\mathbb{G}|$. To evaluate a histogram for m intervals, for example, we have to choose $d = m$ and the direct product $\mathbb{G} = (\mathbb{Z}_{n+1})^m$ of m copies of the cyclic group of size $n + 1$ [3], which is of size $n^{(1+o(1))m}$. Hence, reducing the exponent with respect to $|\mathbb{G}|$ will have a large effect on the communication complexity.

Abelian programs are also applicable to linear classifiers, which perform classification based on a weighted sum of inputs and cover popular methods such as support vector machines [7, 8] and logistic regression. Indeed, if the weights are public, the players can locally multiply their inputs by the weights and then execute a protocol for a certain abelian program. However, the weights are often kept private to protect the intellectual property of learned models in practice. To the best of our knowledge, there is no NIMPC protocol for linear classifiers that does not reveal weights to players other than those obtained from the generic construction.

1.1 Our Results

The contributions of this paper are threefold. First, we present efficient NIMPC protocols for abelian programs that improve the best known communication complexity. Secondly, we propose for the first time NIMPC protocols for linear classifiers that are more efficient than those obtained from the generic construction. Thirdly, we revisit a known transformation [6] (hereinafter referred to as the BKR transformation), which transforms any 0-robust NIMPC protocol, also known as PSM (Private Simultaneous Messages) protocol [9, 13], into a t-robust one. The transformation is repeatedly used in the above two results to limit players to inputs from smaller domains. We reveal that their NIMPC protocol used in the transformation does not satisfy even 1-robustness and we also fix their protocol.

Efficient NIMPC Protocols for Abelian Programs. We propose a fully robust NIMPC protocol with communication complexity $O(|\mathbb{G}|(\log|\mathbb{G}|)^2)$ for abelian programs allowing inputs to take any value of $\mathbb{G}$. Our protocol improves the previous result $O(|\mathbb{G}|^2 n^2)$ [3]. Note that it is impossible to cut down on the exponent with respect to $|\mathbb{G}|$ anymore due to the lower bound [17]. For abelian programs with limited input domains, we apply the BKR transformation to our protocol with the extended input domain in a non-straightforward way. This is the first time that the BKR transformation, which originally aims at lifting the level of robustness, has been used to restrict input domains. As a result, we obtain a t-robust protocol with communication complexity $|\mathbb{G}|(\log|\mathbb{G}|)^2 p^{t+O(1)}$ if input domains are of size at most d, where p is the smallest prime power such that $p \geq \max\{n, d\}$. This protocol is more efficient than the previous protocol of [3] and even than that of [6] if $t = o(\log n)$ and $|\mathbb{G}| = n^{\Theta(1)}$.

New NIMPC Protocols for Linear Classifiers. We define the class of linear classifiers as functions computing $f(\sum_{i\in[n]} w_i x_i)$ on a weighted sum of

Table 1. Comparison of the existing t-robust NIMPC protocols for abelian programs. Let n be the number of players, $\mathbb{G}$ be an abelian group, d be a positive integer at most $|\mathbb{G}|$, and p be the smallest prime power such that $p \geq \max\{n, d\}$. We suppose $t = n$ if the symbol t does not appear in the complexity.

Reference	Input domain	Communication complexity
[3]	$\mathbb{G}^n$	$O(\lvert\mathbb{G}\rvert^2 n^2)$
Ours (Theorem 1)		$O(\lvert\mathbb{G}\rvert(\log \lvert\mathbb{G}\rvert)^2)$
[3]	$\prod_{i\in[n]} S_i$, where $S_i \subseteq \mathbb{G}$ and $\lvert S_i\rvert \leq d$	$\lvert\mathbb{G}\rvert^3 (nd)^{t+O(1)}$
[6]		$\lvert\mathbb{G}\rvert^{\log n + O(1)} n$
Ours (Corollary 1)		$\lvert\mathbb{G}\rvert(\log \lvert\mathbb{G}\rvert)^2 p^{t+O(1)}$

inputs for some weights $\boldsymbol{w} = (w_i)_{i\in[n]}$ over a finite field $\mathbb{F}_q$ and some function $f : \mathbb{F}_q \to \{0, 1\}$. Our definition can be naturally extended to functions outputting many bits and can also deal with real-valued inputs by choosing a sufficiently large prime q. We propose a fully robust NIMPC protocol with communication complexity $O(q \log q)$ for this class when inputs take any value of $\mathbb{F}_q$. Note that the multiplicative factor of q is unavoidable due to the lower bound from [17]. In our protocol, no information on the weights is leaked other than what is implied by the residual function. Applying the BKR transformation, we also obtain a t-robust protocol with communication complexity $p^{t+O(1)} q \log q$ for linear classifiers with input domains of size at most d, where p is the smallest prime power such that $p \geq \max\{n, d\}$.

Revisiting the BKR Transformation. We have used the BKR transformation in the above two results to restrict input domains. However, we revisit the transformation and reveal that their fully robust NIMPC protocol for what they call outputting-message functions, which is used as a building block in the transformation, does not satisfy even 1-robustness. We also fix their protocol with only constant overhead in the communication complexity. Therefore, the statements of [6] still hold true but it is necessary to use our modified protocol for outputting-message functions when applying the BKR transformation. As a byproduct of that modification, we obtain a fully robust NIMPC protocol for the class of indicator functions. An indicator function decides whether a tuple of inputs $x \in \mathcal{X}_0^n$ is equal to some fixed $a \in \mathcal{X}_0^n$, where $\mathcal{X}_0$ is a fixed domain. Our protocol has communication complexity $O((\log |\mathcal{X}_0|)n)$ improving the best known result $O((\log |\mathcal{X}_0|)^2 n)$ [17] and is asymptotically optimal with respect to the input length according to the lower bound [17].

1.2 Related Work

It is known that NIMPC for indicator functions is used as a building block to construct protocols for any given class of functions [3]. For the class of all the functions from $\mathcal{X}_0^n$ to a finite set $\mathcal{Z}$, our result on indicator functions implies a fully

robust protocol with communication complexity $O(|\mathcal{X}_0|^n(\log|\mathcal{X}_0|)(\log|\mathcal{Z}|)n)$ improving the previous results [3, 17, 15]. Recently, however, the authors of [1] propose an asymptotically optimal protocol achieving $O(|\mathcal{X}_0|^n(\log|\mathcal{Z}|))$ without using indicator functions. For the class consisting only of a single function $f : \mathcal{X}_0^n \to \{0,1\}$, a t-robust protocol has been proposed in [5] by applying the BKR transformation to an efficient PSM protocol for f. If $t < n/2$, it is more efficient than the protocol obtained from indicator functions. Nevertheless, it makes sense to construct efficient protocols for indicator functions if we aim at an intermediate class of functions rather than the above two extreme ones.

For boolean symmetric functions, the authors of [6] devise a more efficient t-robust NIMPC protocol with communication complexity $n^{\log\log n+\log t+O(1)}$ than those obtained from abelian programs.

2 Technical Overview

In this section, we provide an overview of our NIMPC protocols. We give more detailed descriptions and security proofs in the following sections.

2.1 Efficient NIMPC Protocols for Abelian Programs

An abelian program h takes n inputs from an abelian group $\mathbb{G}$ and outputs $h(x_1,\ldots,x_n) = f(\sum_{i\in[n]} x_i)$ for some function $f : \mathbb{G} \to \{0,1\}$. We start by explaining how to construct an efficient fully robust NIMPC protocol for the class of abelian programs in which inputs take any value of $\mathbb{G}$. Our protocol is in part based on the result of Beimel et al. [3, Theorem 7.2], which has presented a protocol tailored to abelian programs over the cyclic group $\mathbb{Z}_{n+1} = \{0,1,\ldots,n\}$ of size $n+1$. They consider a special map $\sigma : x \mapsto x+1 \mod (n+1)$ from $\mathbb{Z}_{n+1}$ to itself and view every element $g \in \mathbb{Z}_{n+1}$ as the g-th iteration of σ, i.e., $\sigma^g := \sigma \circ \cdots \circ \sigma$ (g times). They then reduce computing abelian programs to composing the maps σ^{x_i} corresponding to the players' inputs x_i. To hide the inputs and achieve robustness, they randomize the operation of that composition by using Kilian's technique [14]. However, there is no such map as σ in a general abelian group, which is why the previous result is only applicable to $\mathbb{Z}_{n+1}$.

Construction Based on the Regular Representation. To represent elements of an abelian group $\mathbb{G}$, we make the most use of the regular representation of $\mathbb{G}$. Observe that the g-th iteration of σ is equivalent to the map $x \mapsto x+g \mod (n+1)$. Generalizing it, we view an element $g \in \mathbb{G}$ as a permutation $\sigma_g : \mathbb{G} \ni x \mapsto x+g \in \mathbb{G}$, which is further viewed as a linear map from the $|\mathbb{G}|$-dimensional vector space over $\mathbb{F}_2 = \{0,1\}$ to itself translating every basis $\boldsymbol{e}_x$ to $\boldsymbol{e}_{x+g}$, where $\boldsymbol{e}_x \in \mathbb{F}_2^{|\mathbb{G}|}$ is the unit vector such that the entry indexed by $x \in \mathbb{G}$ is one. Then, the summation of inputs corresponds to the composition of the associated linear maps. Furthermore, if we appropriately represent the associated function $f : \mathbb{G} \to \{0,1\}$ as a vector, we can express the whole computation of the abelian program as a certain matrix-vector product. As in [3],

we use the randomization technique [14] to securely perform that linear algebra operation. However, there still remain two problems in the above protocol: (1) the resultant communication complexity is $O(|\mathbb{G}|^2 \log |\mathbb{G}|)$ since the protocol has to communicate $O(|\mathbb{G}|)$ permutations over $\mathbb{G}$, each of which is expressed as $O(|\mathbb{G}| \log |\mathbb{G}|)$ bits and (2) it only works for the specific abelian program since it reveals partial information on the truth table, e.g., $|f^{-1}(1)|$. Regarding the first problem, we cut down the number of matrices with the help of the fundamental theorem of finite abelian groups. Since it implies that all the group elements are generated by $O(\log |\mathbb{G}|)$ elements, players can compute their messages from only $O(\log |\mathbb{G}|)$ permutations corresponding to the generators. To hide the value of $|f^{-1}(1)|$, we carefully choose a group extension $\mathbb{H} \supsetneq \mathbb{G}$ and extend $f : \mathbb{G} \to \{0,1\}$ to $\widetilde{f} : \mathbb{H} \to \{0,1\}$ so that $|\widetilde{f}^{-1}(1)|$ is constant regardless of f. The details are given in Section 4.

Limiting Inputs. To construct a protocol for abelian programs with limited input domains, we use the BKR transformation in a non-straightforward way. Note that it has been originally devised to obtain t-robust NIMPC protocols from 0-robust ones. Clearly, the above fully robust protocol for abelian programs with the extended input domain satisfies 0-robustness. Since 0-robustness is not affected by what the input domain is, that protocol is itself a 0-robust protocol for abelian programs with *limited* input domain. We then apply the BKR transformation and lift the level of robustness to $t > 0$ with some overhead in communication complexity.

2.2 New NIMPC Protocols for Linear Classifiers

We formally define linear classifiers as the class of all the functions of the form $h_{f,\boldsymbol{w}} : \mathbb{F}_q^n \ni (x_i)_{i\in[n]} \mapsto f(\sum_{i\in[n]} w_i x_i) \in \{0,1\}$, where $\mathbb{F}_q$ is the fixed finite field of size q, $f : \mathbb{F}_q \to \{0,1\}$, and $\boldsymbol{w} = (w_i)_{i\in[n]} \in \mathbb{F}_q^n$.

To begin with, we fix f and treat $k = |f^{-1}(1)|$ as public information. We show a construction of a fully robust protocol for the class consisting only of the specific function $\{h_{f,\boldsymbol{w}}\}$. Let $f^{-1}(1) = \{u_1, \ldots, u_k\}$ and set $\boldsymbol{u} = (u_j)_{j\in[k]} \in \mathbb{F}_q^k$. The main idea of our construction is to use the fact that $f(\sum_{i\in[n]} w_i x_i) = 1$ if and only if at least one entry of $\boldsymbol{u}_0 = (u_j - \sum_{i\in[n]} w_i x_i)_{j\in[k]}$ is zero. We must ensure that the evaluator learns the number of zeros in $\boldsymbol{u}_0$ and nothing more. We randomly choose a permutation π over $[k]$ and k non-zero elements $r_i \in \mathbb{F}_q \setminus \{0\}$ $(i \in [k])$ and define $\widetilde{\boldsymbol{u}} = (r_j u_{\pi(j)})_{j\in[k]}$. We then send each P_i the vector $\widetilde{\boldsymbol{u}}_i = (r_j w_i)_{j\in[k]}$ along with a random vector $\boldsymbol{s}_i \in \mathbb{F}_q^k$ for masking his input, who in turn sends his message $\widetilde{\boldsymbol{u}}_i x_i + \boldsymbol{s}_i$ to the evaluator. We let the evaluator receive $\widetilde{\boldsymbol{u}} + \sum_{i\in[n]} \boldsymbol{s}_i$ in advance and after receiving the messages, he outputs $f(\sum_{i\in[n]} w_i x_i)$ according to the number of zeros in $\widetilde{\boldsymbol{u}} - \sum_{i\in[n]} \widetilde{\boldsymbol{u}}_i x_i$.

However, as mentioned above, this protocol assumes the dimension $k = |f^{-1}(1)|$ of the vectors is public, which is why it does not work for the class of all the linear classifiers. Our solution to hide k is *padding* the vector $\boldsymbol{u}$ with certain $q - k$ elements u_j $(k < j \leq q)$ to ensure that its dimension should be q.

Specifically, we carefully select these $q-k$ elements from an extension field of $\mathbb{F}_q$ so that the protocol satisfies correctness. We also propose a protocol for linear classifiers with limited input domains by applying the BKR transformation as in the case of abelian programs. The details are given in Section 5.

2.3 Revisiting the BKR Transformation

As a building block for the BKR transformation, the authors of [6] propose a fully robust NIMPC protocol for what they call outputting-message functions. We reveal that their protocol does not satisfy even 1-robustness. Technically, they define an outputting-message function for a message m, a vector $\boldsymbol{u}$, and a matrix $\boldsymbol{A} = [\boldsymbol{a}_1, \ldots, \boldsymbol{a}_n]$ as the function outputting m if $\boldsymbol{u} = \boldsymbol{A}[x_1, \ldots, x_n]^\top$ holds and $\perp$ otherwise.

The main issue is in their procedures for securely testing the equality $\boldsymbol{u} = \boldsymbol{A}[x_1, \ldots, x_n]^\top$. In their protocol, each P_i receives a random vector $\boldsymbol{s}_i$ and sends $\boldsymbol{\nu}_i := \boldsymbol{a}_i x_i + \boldsymbol{s}_i$ as part of his message. The evaluator receives $\boldsymbol{\nu}_0 := \boldsymbol{u} + \sum_{i\in[n]} \boldsymbol{s}_i$ in advance and then tests whether $\boldsymbol{\nu}_0 = \sum_{i\in[n]} \boldsymbol{\nu}_i$ holds. However, consider the collusion of the player P_1 and the evaluator. If $\boldsymbol{d} := \boldsymbol{u} - \sum_{i\neq 1} \boldsymbol{a}_i x_i$ and $\boldsymbol{a}_1$ are linearly independent, they should learn nothing at all since the residual function outputs nothing but $\perp$. Nevertheless, they actually obtain the vector $\boldsymbol{d} = \boldsymbol{\nu}_0 - \sum_{i\neq 1} \boldsymbol{\nu}_i - \boldsymbol{s}_1$.

We fix their protocol with only constant overhead in the communication complexity. Our main idea is randomizing $\boldsymbol{d}$ to ensure for P_1 and the evaluator not to learn more than the linear independence relation between $\boldsymbol{d}$ and $\boldsymbol{a}_1$, which is the only information revealed by the residual function. Specifically, we choose an invertible matrix $\boldsymbol{T}$ uniformly at random and redefine $\boldsymbol{\nu}_0 = \boldsymbol{T}\boldsymbol{u} + \sum_{i\in[n]} \boldsymbol{s}_i$. We additionally give $\boldsymbol{T}\boldsymbol{a}_i$ to each P_i as randomness. Now, P_1 and the evaluator only learns $\boldsymbol{\nu}_0 - \sum_{i\neq 1} \boldsymbol{\nu}_i - \boldsymbol{s}_1 = \boldsymbol{T}\boldsymbol{d}$, which does not reveal more than the linear independence relation between $\boldsymbol{d}$ and $\boldsymbol{a}_1$ due to the randomness of $\boldsymbol{T}$. Note that our modification increases the communication complexity of their protocol only by a constant factor. We present the formal statements in Section 6.

Finally, as a byproduct of that modification, we obtain a fully robust protocol for indicator functions with asymptotically optimal communication complexity with respect to the input length. Roughly speaking, we embed every input domain to a subset of a fixed finite field and translate the condition $(x_i)_{i\in[n]} = (a_i)_{i\in[n]}$ to the equality test $\boldsymbol{u} = \boldsymbol{I}_n[x_1, \ldots, x_n]^\top$, where $\boldsymbol{u} = [a_1, \ldots, a_n]^\top$ and $\boldsymbol{I}_n$ is the identity matrix of size n. We show that an NIMPC protocol for that equality test is obtained from our modified protocol for outputting-message functions.

3 Preliminaries

Notations. For a set $\mathcal{X} = \mathcal{X}_1 \times \cdots \times \mathcal{X}_n$ and $C \subseteq [n]$, we define $\mathcal{X}_C = \prod_{i\in C} \mathcal{X}_i$. For $x \in \mathcal{X}$, we define x_C as the restriction $(x_i)_{i\in C}$ of x to $\mathcal{X}_C$. Let $\overline{C}$ be the complement of $C \subseteq [n]$ and $x_{\overline{C}} \in \mathcal{X}_{\overline{C}}$. For a function $h : \mathcal{X} \to \mathcal{Z}$, we define

the residual function $h|_{\overline{C},x_{\overline{C}}} : \mathcal{X}_C \to \mathcal{Z}$ of h for $\overline{C}$ and $x_{\overline{C}}$ as the sub-function of h obtained by restricting the input variables indexed by $\overline{C}$ to $x_{\overline{C}}$, that is, $h|_{\overline{C},x_{\overline{C}}}(x_C) = h(x_C, x_{\overline{C}})$.

For a finite set S, we write $s \leftarrow\!\!\$\, S$ if we choose a uniformly random element s from S. For two distributions $\mathcal{D}, \mathcal{D}'$ on S, we write $\mathcal{D} \equiv \mathcal{D}'$ if they are perfectly identical to each other. Define $\mathfrak{S}_S$ as the set of all the permutations over S. We simply write $\mathfrak{S}_N$ if S is clear from the context, where $N = |S|$. For $\pi \in \mathfrak{S}_S$ and a finite field $\mathbb{K}$, we define a permutation matrix $\boldsymbol{U}_\pi$ as the square matrix over $\mathbb{K}$ of size $|S|$ whose (i,j)-th entry is 1 if $j = \pi(i)$ and 0 otherwise, where we assume that the sets indexing the rows and columns are both S. It holds that $\boldsymbol{U}_\pi^{-1} = \boldsymbol{U}_{\pi^{-1}} = \boldsymbol{U}_\pi^\top$ and $\boldsymbol{U}_\pi \boldsymbol{U}_\tau = \boldsymbol{U}_{\tau\circ\pi}$, where π^{-1} is the inverse of π and $\tau \circ \pi$ is the composition of $\pi, \tau \in \mathfrak{S}_S$, i.e., $(\tau \circ \pi)(i) = \tau(\pi(i))$ for all $i \in S$. For $i \in S$, let $\boldsymbol{e}_i = (a_j)_{j\in S} \in \mathbb{K}^N$ denote the i-th unit vector, i.e., $a_i = 1$ and $a_j = 0$ for all $j \neq i$.

For $r_i \in \mathbb{K}$ $(i \in [N])$, let $\mathrm{diag}(r_1, \ldots, r_N) \in \mathbb{K}^{N\times N}$ denote a diagonal matrix whose (i,i)-th entry is r_i for $i \in [N]$. Throughout the paper, all vectors are column vectors unless otherwise indicated. For a tuple of vectors $(\boldsymbol{v}_i)_{i\in[N]}$ where $\boldsymbol{v}_i \in \mathbb{K}^k$, we define $\mathrm{Ker}((\boldsymbol{v}_i)_{i\in[N]}) = \{(x_i)_{i\in[N]} \in \mathbb{K}^N \mid \sum_{i\in[N]} \boldsymbol{v}_i x_i = \boldsymbol{0}\}$. For a subset $\mathcal{S} \subseteq \mathbb{K}^N$, we denote by $\mathcal{S}^\perp$ the orthogonal complement of $\mathcal{S}$, i.e., $\mathcal{S}^\perp = \{(\delta_i)_{i\in[N]} \in \mathbb{K}^N \mid \forall (x_i)_{i\in[N]} \in \mathcal{S},\ \sum_{i\in[N]} \delta_i x_i = 0\}$. We define $\mathrm{GL}_k(\mathbb{K})$ as the set of all the invertible k-by-k matrices, i.e., $\mathrm{GL}_k(\mathbb{K}) = \{\boldsymbol{T} \in \mathbb{K}^{k\times k} \mid \det(\boldsymbol{T}) \neq 0\}$.

3.1 Non-Interactive Secure Multiparty Computation

In NIMPC, we consider n players P_i $(i \in [n])$, each holding an input, and an external output player P_0 called an evaluator. In this paper, we focus on NIMPC with correlated randomness, in which each player locally computes a message from his input and randomness and then sends it to the evaluator.

Definition 1 (NIMPC: syntax and correctness). *Let $\mathcal{X}_i$ $(i \in [n])$ and $\mathcal{Z}$ be finite sets. Let $\mathcal{X} = \prod_{i\in[n]} \mathcal{X}_i$ and $\mathcal{H}$ be a class of functions from $\mathcal{X}$ to $\mathcal{Z}$. Let $\mathcal{R}_i$ $(i \in \{0\} \cup [n])$ and $\mathcal{M}_i$ $(i \in [n])$ be finite sets. An NIMPC protocol for $\mathcal{H}$ is a triplet $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$, where:*

- $\mathsf{Gen} : \mathcal{H} \to \mathcal{R}_0 \times \mathcal{R}_1 \times \cdots \times \mathcal{R}_n$ *is a randomized function;*
- Enc *is an n-tuple of deterministic functions $(\mathsf{Enc}_1, \ldots, \mathsf{Enc}_n)$, where $\mathsf{Enc}_i : \mathcal{X}_i \times \mathcal{R}_i \to \mathcal{M}_i$;*
- $\mathsf{Dec} : \mathcal{R}_0 \times \mathcal{M}_1 \times \cdots \times \mathcal{M}_n \to \mathcal{Z}$ *is a deterministic function satisfying the following correctness requirement: for any $x = (x_1, \ldots, x_n) \in \mathcal{X}$ and any $h \in \mathcal{H}$, it holds that*

$$\Pr[(R_0, R_1, \ldots, R_n) \leftarrow \mathsf{Gen}(h) : \mathsf{Dec}(R_0, \mathsf{Enc}(x, R)) = h(x)] = 1,$$

where $\mathsf{Enc}(x, R) = (\mathsf{Enc}_1(x_1, R_1), \ldots, \mathsf{Enc}_n(x_n, R_n))$.

The online communication complexity $\mathsf{CC}_{\mathsf{on}}(\Pi)$ *of* Π *is the maximum of* $\log|\mathcal{M}_1|,\ldots,\log|\mathcal{M}_n|$. *The offline communication complexity* $\mathsf{CC}_{\mathsf{off}}(\Pi)$ *of* Π *is* $\log|\mathcal{R}_0|,\log|\mathcal{R}_1|,\ldots,\log|\mathcal{R}_n|$. *The communication complexity* $\mathsf{CC}(\Pi)$ *of* Π *is defined as the maximum of* $\mathsf{CC}_{\mathsf{on}}(\Pi)$ *and* $\mathsf{CC}_{\mathsf{off}}(\Pi)$.

To define the security requirements of NIMPC, we consider an adversary who colludes with a set of players $C \subseteq [n]$ as well as the evaluator. In this setting, it is impossible to prevent the adversary from learning the residual function $h|_{\overline{C},x_{\overline{C}}}$ for the inputs $x_{\overline{C}}$ of the honest players. Indeed, he is allowed to compute $h(x_C, x_{\overline{C}})$ for every input x_C from the correlated randomness of C and the messages of $\overline{C}$. We say that an NIMPC protocol is C-robust if the adversary's view is perfectly simulated by some simulator with oracle access to the residual function.

Definition 2 (NIMPC: robustness). *For a subset* $C \subseteq [n]$, *we say that an NIMPC protocol* Π *for* $\mathcal{H}$ *is* C*-robust if there exists a simulator* Sim *with oracle access to a residual function such that, for every* $h \in \mathcal{H}$ *and* $x_{\overline{C}} \in \mathcal{X}_{\overline{C}}$, *we have* $\mathsf{Sim}^{h|_{\overline{C},x_{\overline{C}}}}(C) \equiv (R_0, R_C = (R_i)_{i\in C}, M_{\overline{C}} = (M_i)_{i\in\overline{C}})$, *where* $(R_0, R_1, \ldots, R_n) \leftarrow \mathsf{Gen}(h)$ *and* $M_i = \mathsf{Enc}_i(x_i, R_i)$.

For an integer $0 \le t \le n$, *we say that* Π *is* t*-robust if it is* C*-robust for every* $C \subseteq [n]$ *of size at most* t. *We say that* Π *is fully robust if it is* n*-robust.*

3.2 Abelian Programs

Let $\mathbb{G}$ be a finite abelian group and $S_1,\ldots,S_n$ be subsets of $\mathbb{G}$. Let $\mathcal{X}_i = S_i$ for $i \in [n]$ and $\mathcal{X} = \mathcal{X}_1 \times \cdots \times \mathcal{X}_n$. Define the abelian program $h_f : \mathcal{X} \to \{0,1\}$ associated with $f : \mathbb{G} \to \{0,1\}$ as $h_f(x_1,\ldots,x_n) = f(\sum_{i\in[n]} x_i)$. We then define $\mathcal{A}_{\mathbb{G}}^{S_1,\ldots,S_n}$ be the class of all the abelian programs, that is, $\mathcal{A}_{\mathbb{G}}^{S_1,\ldots,S_n} = \{h_f : \mathcal{X} \to \{0,1\} \mid f : \mathbb{G} \to \{0,1\}\}$. We simply write $\mathcal{A}_{\mathbb{G}}$ if $S_1 = \cdots = S_n = \mathbb{G}$.

The class of abelian programs with limited input domains includes symmetric functions. A function $h : [d]^n \to \{0,1\}$ is called symmetric if $h(x_{\pi(1)},\ldots,x_{\pi(n)}) = h(x_1,\ldots,x_n)$ for all $(x_1,\ldots,x_n) \in [d]^n$ and $\pi \in \mathfrak{S}_n$. Following [3], let $\mathbb{G} = (\mathbb{Z}_{n+1})^d$ and $S_1 = \cdots = S_n = \{\boldsymbol{e}_1,\ldots,\boldsymbol{e}_d\} \subseteq \mathbb{G}$, where $\boldsymbol{e}_i \in (\mathbb{Z}_{n+1})^d$ is the i-th unit vector. Then, we can see that $\mathcal{A}_{\mathbb{G}}^{S_1,\ldots,S_n}$ is equivalent to the class of all symmetric functions over $[d]^n$ identifying $x_i \in [d]$ with $\boldsymbol{e}_{x_i} \in \mathbb{G}$ since the outputs $h(x_1,\ldots,x_n)$ of symmetric functions only depend on $\sum_{i\in[n]} \boldsymbol{e}_{x_i}$.

Although assuming above that abelian programs output only one bit, it is possible to extend them to the ones outputting m bits by computing each output bit separately [3]. Specifically, given an NIMPC protocol Π for $\mathcal{A}_{\mathbb{G}}^{S_1,\ldots,S_n}$, we can construct a protocol Π_m for $\mathcal{H}_m := \{h : \mathcal{X} \to \{0,1\}^m \mid h = (h_1,\ldots,h_m),\ h_i \in \mathcal{A}_{\mathbb{G}}^{S_1,\ldots,S_n}\}$ with m times higher communication complexity by running Π for each h_i separately.

We note that limiting players to inputs from smaller domains is not a straightforward task for NIMPC. For example, a t-robust NIMPC protocol for $\mathcal{A}_{\mathbb{G}}$ does not directly imply a t-robust protocol for $\mathcal{A}_{\mathbb{G}}^{S_1,\ldots,S_n}$. This is because in a 1-robust protocol for $h \in \mathcal{A}_{\mathbb{G}}$, any player P_i colluding with the evaluator learns the value

of h on the honest inputs and every possible choice of x_i from $\mathbb{G}$ while in a protocol for $h \in \mathcal{A}_{\mathbb{G}}^{S_1,\ldots,S_n}$, P_i is allowed to evaluate the residual function only on x_i from S_i.

4 Efficient NIMPC Protocols for Abelian Programs

4.1 The Design of Our Protocol

First, we design an efficient fully robust NIMPC protocol for the class $\mathcal{A}_{\mathbb{G}}$ of abelian programs in which inputs take any value of $\mathbb{G}$. We have already explained a high-level idea of our construction in Section 2.1.

Construction Based on the Regular Representation. Recall that via the regular representation of $\mathbb{G}$, we identify every element $g \in \mathbb{G}$ with the linear map from $\mathbb{F}_2^N$ to itself translating every basis $\boldsymbol{e}_x \in \mathbb{F}_2^N$ to $\boldsymbol{e}_{x+g} \in \mathbb{F}_2^N$ for $x \in \mathbb{G}$. Here, $N = |\mathbb{G}|$ and we assume that $\mathbb{G}$ is the index set for N-dimensional vectors and for the rows and columns of N-by-N matrices. The linear map is in turn expressed as the permutation matrix $\boldsymbol{A}_g := \boldsymbol{U}_{\sigma_g} \in \mathrm{GL}_N(\mathbb{F}_2)$, where $\sigma_g : \mathbb{G} \ni x \mapsto x + g \in \mathbb{G}$. The summation of inputs x_i is now reduced to the multiplication of the $\boldsymbol{A}_{x_i}$'s. To evaluate a function $f : \mathbb{G} \to \{0,1\}$ on the sum $s = \sum_{i\in[n]} x_i$, we associate f with the vector $\boldsymbol{v}_f := \sum_{x\in f^{-1}(1)} \boldsymbol{e}_x \in \mathbb{F}_2^N$. Then, we can translate the evaluation of f into the matrix-vector product $\boldsymbol{e}_{0_{\mathbb{G}}}^{\top}\boldsymbol{A}_s\boldsymbol{v}_f = f(s)$, where $0_{\mathbb{G}}$ is the identity of $\mathbb{G}$. We use the randomization technique [14] to securely perform these linear algebra operations. Note that all the permutation matrices can be represented by $O(N \log N)$ bits.

To obtain a concrete protocol, suppose that we compute an abelian program h_f associated with $f : \mathbb{G} \to \{0,1\}$. We first randomly select $n-1$ permutations $\pi_1, \ldots, \pi_{n-1}$ over $\mathbb{G}$. Then, we give $\boldsymbol{U}_{\pi_1}$ to the player P_1, $(\boldsymbol{U}_{\pi_{i-1}}^{-1}\boldsymbol{A}_g\boldsymbol{U}_{\pi_i})_{g\in\mathbb{G}}$ to P_i $(1 < i < n)$, and $(\boldsymbol{U}_{\pi_{n-1}}^{-1}\boldsymbol{A}_g\boldsymbol{v}_f)_{g\in\mathbb{G}}$ to P_n. If P_1 sends $\boldsymbol{e}_{0_{\mathbb{G}}}^{\top}\boldsymbol{A}_{x_1}\boldsymbol{U}_{\pi_1}$ and the other players send the matrices corresponding to their inputs, the evaluator can compute $f(\sum_{i\in[n]} x_i) = \boldsymbol{e}_{0_{\mathbb{G}}}^{\top}(\prod_{i\in[n]} \boldsymbol{A}_{x_i})\boldsymbol{v}_f$. However, there remain the following two problems: (1) the communication complexity of this protocol is $O(N^2 \log N)$ since it needs to communicate at most N permutation matrices and (2) it only works for the class $\{h_f\}$ consisting only of the specific abelian program associated with the fixed function f, not for $\mathcal{A}_{\mathbb{G}}$, since the randomness of P_n reveals partial information on the truth table of h_f, i.e., the value of $|f^{-1}(1)|$.

Reducing Communication Complexity and Hiding the Truth Table. To cut down the number of permutation matrices to communicate, we recall the fundamental theorem of finite abelian groups, e.g., [16, Theorem 6.44]. For any abelian group $\mathbb{G}$ of size N, there exists a generating set $\{s_j \mid j \in [m]\}$ of size $m = O(\log N)$. For each $x \in \mathbb{G}$, we fix m integers $\ell_j(x)$ $(j \in [m])$ such that $x = \sum_{j\in[m]} \ell_j(x)s_j$ in $\mathbb{G}$.

We now give $\boldsymbol{U}_{\pi_{i-1}}^{-1}\boldsymbol{U}_{\pi_i}$ and $(\boldsymbol{U}_{\pi_{i-1}}^{-1}\boldsymbol{A}_{s_j}\boldsymbol{U}_{\pi_i})_{j\in[m]}$ to the player P_i with $1 < i < n$. Then, P_i can compute $\boldsymbol{U}_{\pi_{i-1}}^{-1}\boldsymbol{A}_{s_j}\boldsymbol{U}_{\pi_{i-1}}$ for every $j \in [m]$. For an input $x_i \in \mathbb{G}$, he computes $\prod_{j\in[m]}(\boldsymbol{U}_{\pi_{i-1}}^{-1}\boldsymbol{A}_{s_j}\boldsymbol{U}_{\pi_{i-1}})^{\ell_j(x_i)} = \boldsymbol{U}_{\pi_{i-1}}^{-1}\boldsymbol{A}_{x_i}\boldsymbol{U}_{\pi_{i-1}}$. Finally, he obtains $\boldsymbol{U}_{\pi_{i-1}}^{-1}\boldsymbol{A}_{x_i}\boldsymbol{U}_{\pi_i}$ by multiplying $\boldsymbol{U}_{\pi_{i-1}}^{-1}\boldsymbol{U}_{\pi_i}$. Similarly, we give P_n the randomness $\boldsymbol{U}_{\pi_{n-1}}^{-1}\boldsymbol{U}_{\pi_n}$, $(\boldsymbol{U}_{\pi_{n-1}}^{-1}\boldsymbol{A}_{s_j}\boldsymbol{U}_{\pi_n})_{j\in[m]}$, and $\boldsymbol{U}_{\pi_n}^{-1}\boldsymbol{v}_f$, where $\pi_n \leftarrow_{\$} \mathfrak{S}_{\mathbb{G}}$. This protocol only communicates $O(m) = O(\log N)$ permutation matrices and hence achieves the communication complexity $O(N(\log N)^2)$.

Next, to hide the value of $|f^{-1}(1)|$, we augment the vector $\boldsymbol{v}_f$ of Hamming weight $|f^{-1}(1)|$ by a vector of weight $N - |f^{-1}(1)|$ to ensure that the number of ones in the augmented vector is N regardless of f. Specifically, we consider the group extension $\mathbb{H} := \mathbb{G} \times \mathbb{F}_2 = \{(x, b) \mid x \in \mathbb{G},\ b \in \mathbb{F}_2\}$ of $\mathbb{G}$ and replace the $|\mathbb{G}|$-dimensional vectors and matrices introduced above by some $|\mathbb{H}|$-dimensional ones. Since $|\mathbb{H}| = 2N$, the communication complexity is still $O(N(\log N)^2)$.

4.2 Abelian Programs with the Extended Input Domain

Now, we present the formal description of our protocol for $\mathcal{A}_{\mathbb{G}}$. Let $\mathbb{H} = \mathbb{G}\times\mathbb{F}_2$ be the direct product of $\mathbb{G}$ and $\mathbb{F}_2$. Instead of $\boldsymbol{v}_f$, we redefine the vector $\boldsymbol{w}_f \in \mathbb{F}_2^{|\mathbb{H}|}$ representing a function f as $\boldsymbol{w}_f = \sum_{x\in f^{-1}(1)} \boldsymbol{e}_{(x,0)} + \sum_{x\in f^{-1}(0)} \boldsymbol{e}_{(x,1)}$, where $\boldsymbol{e}_{(x,b)} \in \mathbb{F}_2^{|\mathbb{H}|}$ is the unit vector such that the entry indexed by (x, b) is 1. Here, $\mathbb{H}$ is the index set for $2N$-dimensional vectors and for the rows and columns of $2N$-by-$2N$ matrices. It can be seen that the Hamming weight of $\boldsymbol{w}_f$ is now N regardless of f.

According to that modification, we sample each permutation at random from $\mathfrak{S}_{\mathbb{H}}$ rather than $\mathfrak{S}_{\mathbb{G}}$. We also replace the permutation $\sigma_x \in \mathfrak{S}_{\mathbb{G}}$ representing $x \in \mathbb{G}$ with $\tau_x \in \mathfrak{S}_{\mathbb{H}}$ defined as $\tau_x(g, b) = (g + x, b)$ for $(g, b) \in \mathbb{H}$. We define $\boldsymbol{B}_x = \boldsymbol{U}_{\tau_x} \in \mathbb{F}_2^{2N\times 2N}$ instead of $\boldsymbol{A}_x$. Note that the $\boldsymbol{B}_x$'s also satisfy the following homomorphic property: $\boldsymbol{B}_x^{-1} = \boldsymbol{B}_{-x}$ and $\boldsymbol{B}_x\boldsymbol{B}_y = \boldsymbol{B}_y\boldsymbol{B}_x = \boldsymbol{B}_{x+y}$ for $x, y \in \mathbb{G}$.

Theorem 1. *Let $\mathbb{G}$ be a finite abelian group. Let $S \subseteq \mathbb{G}$ be a generating set of $\mathbb{G}$. Let $\mathcal{X}_1 = \cdots = \mathcal{X}_n = \mathbb{G}$ and $\mathcal{X} = \mathcal{X}_1 \times \cdots \times \mathcal{X}_n$. Then, the protocol Π_1 described in Fig. 1 is a fully robust NIMPC protocol for $\mathcal{A}_{\mathbb{G}}$ such that*

$$\mathsf{CC}_{\mathsf{on}}(\Pi_1) = 2|\mathbb{G}| \cdot \lceil \log |\mathbb{G}| + 1 \rceil$$
$$\textit{and } \mathsf{CC}_{\mathsf{off}}(\Pi_1) = 2|\mathbb{G}| \cdot \lceil \log |\mathbb{G}| + 1 \rceil \cdot (|S| + 1) + 2|\mathbb{G}|.$$

In particular, it holds that $\mathsf{CC}(\Pi_1) = O(|\mathbb{G}|(\log |\mathbb{G}|)^2)$.

Proof. **Correctness.** The message of P_1 is $\boldsymbol{e}_{(0_{\mathbb{G}},0)}{}^{\top}\boldsymbol{B}_{x_1}\boldsymbol{U}_{\pi_1}$. The message of P_i with $1 < i < n$ is

$$M_i = \Big(\prod_{j\in[m]} (\boldsymbol{U}_{\pi_{i-1}}^{-1}\boldsymbol{B}_{s_j}\boldsymbol{U}_{\pi_{i-1}})^{\ell_j(x_i)}\Big)(\boldsymbol{U}_{\pi_{i-1}}^{-1}\boldsymbol{U}_{\pi_i}) = \boldsymbol{U}_{\pi_{i-1}}^{-1}\boldsymbol{B}_{x_i}\boldsymbol{U}_{\pi_i}.$$

Protocol Π_1 for $\mathcal{A}_{\mathbb{G}}$

$\mathsf{Gen}(h_f)$**:**

1. Choose $\pi_1, \ldots, \pi_n \leftarrow\!\!\$\, \mathfrak{S}_{\mathbb{H}}$.
2. Output $R_0 = \bot$, $R_1 = \boldsymbol{U}_{\pi_1}$, $R_i = (\boldsymbol{U}_{\pi_{i-1}}^{-1}\boldsymbol{U}_{\pi_i}, (\boldsymbol{U}_{\pi_{i-1}}^{-1}\boldsymbol{B}_{s_j}\boldsymbol{U}_{\pi_i})_{j\in[m]})$ for $1 < i < n$, and $R_n = (\boldsymbol{U}_{\pi_{n-1}}^{-1}\boldsymbol{U}_{\pi_n}, (\boldsymbol{U}_{\pi_{n-1}}^{-1}\boldsymbol{B}_{s_j}\boldsymbol{U}_{\pi_n})_{j\in[m]}, \boldsymbol{U}_{\pi_n}^{-1}\boldsymbol{w}_f)$.

$\mathsf{Enc}_i(x_i, R_i)$**:**

- $i = 1$:
  1. Output $M_1 = \boldsymbol{e}_{(0_{\mathbb{G}},0)}{}^{\top}\boldsymbol{B}_{x_1}\boldsymbol{U}_{\pi_1}$.
- $1 < i < n$:
  1. Parse R_i as $(\boldsymbol{L}_0, (\boldsymbol{L}_j)_{j\in[m]})$, where $\boldsymbol{L}_j \in \mathbb{F}_2^{|\mathbb{H}|\times|\mathbb{H}|}$.
  2. Output $M_i = (\prod_{j\in[m]}(\boldsymbol{L}_j\boldsymbol{L}_0^{-1})^{\ell_j(x_i)})\boldsymbol{L}_0$.
- $i = n$:
  1. Parse R_n as $(\boldsymbol{L}_0, (\boldsymbol{L}_j)_{j\in[m]}, \boldsymbol{r})$, where $\boldsymbol{L}_j \in \mathbb{F}_2^{|\mathbb{H}|\times|\mathbb{H}|}$ and $\boldsymbol{r} \in \mathbb{F}_2^{|\mathbb{H}|}$.
  2. Output $M_n = (\prod_{j\in[m]}(\boldsymbol{L}_j\boldsymbol{L}_0^{-1})^{\ell_j(x_n)})\boldsymbol{L}_0\boldsymbol{r}$.

$\mathsf{Dec}(M_1, \ldots, M_n)$**:**

1. Output $\prod_{i\in[n]} M_i$.

Fig. 1. The NIMPC protocol Π_1 for the class of abelian programs $\mathcal{A}_{\mathbb{G}}$.

The message of P_n is

$$M_i = (\prod_{j\in[m]}(\boldsymbol{U}_{\pi_{n-1}}^{-1}\boldsymbol{B}_{s_j}\boldsymbol{U}_{\pi_{n-1}})^{\ell_j(x_n)})(\boldsymbol{U}_{\pi_{n-1}}^{-1}\boldsymbol{U}_{\pi_n})(\boldsymbol{U}_{\pi_n}^{-1}\boldsymbol{w}_f) = \boldsymbol{U}_{\pi_{n-1}}^{-1}\boldsymbol{B}_{x_i}\boldsymbol{w}_f.$$

Therefore, letting $a = \sum_{i\in[n]} x_i$, we have

$$\begin{aligned}
\prod_{i\in[n]} M_i &= \boldsymbol{e}_{(0_{\mathbb{G}},0)}{}^{\top}\boldsymbol{B}_a\boldsymbol{w}_f \\
&= \boldsymbol{e}_{(0_{\mathbb{G}},0)}{}^{\top}\boldsymbol{B}_a(\sum_{g\in\mathbb{G}}\boldsymbol{e}_{(g,1-f(g))}) \\
&= \boldsymbol{e}_{(0_{\mathbb{G}},0)}{}^{\top}(\sum_{g\in\mathbb{G}}\boldsymbol{e}_{(g-a,1-f(g))}) \\
&= \boldsymbol{e}_{(0_{\mathbb{G}},0)}{}^{\top}\boldsymbol{e}_{(0_{\mathbb{G}},1-f(a))} \\
&= f(a).
\end{aligned}$$

Robustness. Let $C \subseteq [n]$. In the following, we show that for functions $f, f' : \mathbb{G} \to \{0,1\}$ and inputs $x_{\overline{C}}, x'_{\overline{C}}$ such that $h_f|_{\overline{C},x_{\overline{C}}} = h_{f'}|_{\overline{C},x'_{\overline{C}}}$, the messages of $\overline{C}$ and the correlated randomness of C are equally distributed in the protocol Π_1. Based on that observation, we can construct the simulator with oracle access to $h_f|_{\overline{C},x_{\overline{C}}}$ as follows: the simulator finds f' and $x'_{\overline{C}}$ giving the same residual function as $h_f|_{\overline{C},x_{\overline{C}}}$ and then executes Π_1 on $h_{f'}$ and $x'_{\overline{C}}$. Since $h_{f'}|_{\overline{C},x'_{\overline{C}}} =$

Simulator $\mathsf{Sim}^{h|_{\overline{C},x_{\overline{C}}}}(C)$ for Π_1

Input. A set of colluding players C.
Oracle access. The residual function $h|_{\overline{C},x_{\overline{C}}}$ for the inputs $x_{\overline{C}}$ of honest players.
Output. Correlated randomness R_0 and $(R_i)_{i\in C}$ of the colluding players and messages $(M_i)_{i\in\overline{C}}$ of the honest players.
Algorithm.

- If $C = \emptyset$:
  1. Define $f' : \mathbb{G} \to \{0,1\}$ as $f'(a) = h(x_1, \ldots, x_n)$ for all $a \in \mathbb{G}$ using the oracle.
  2. Define $h' : \mathbb{G}^n \to \{0,1\}$ as $h'(a_1, \ldots, a_n) = f'(\sum_{i\in[n]} a_i)$ for all $a_1, \ldots, a_n \in \mathbb{G}$.
  3. Let $(R_0 = \perp, R_1, \ldots, R_n) \leftarrow \mathsf{Gen}(h')$.
  4. Set $M_i = \mathsf{Enc}_i(0_{\mathbb{G}}, R_i)$ for $i \in [n]$.
  5. Output $R_0 = \perp$ and $(M_1, \ldots, M_n)$.
- If $C \neq \emptyset$:
  1. Fix $i_0 \in C$.
  2. Define $f' : \mathbb{G} \to \{0,1\}$ as follows. For every $a \in \mathbb{G}$:
     (a) Define x_C as $x_{i_0} = a$ and $x_i = 0_{\mathbb{G}}$ for $i \in C \setminus \{i_0\}$.
     (b) Let $f'(a) = h|_{\overline{C},x_{\overline{C}}}(x_C)$ using the oracle $h|_{\overline{C},x_{\overline{C}}}$.
  3. Define $h' : \mathbb{G}^n \to \{0,1\}$ as $h'(a_1, \ldots, a_n) = f'(\sum_{i\in[n]} a_i)$ for all $a_1, \ldots, a_n \in \mathbb{G}$.
  4. Let $(R_0 = \perp, R_1, \ldots, R_n) \leftarrow \mathsf{Gen}(h')$.
  5. Set $M_i = \mathsf{Enc}_i(0_{\mathbb{G}}, R_i)$ for $i \in [n]$.
  6. Output $R_0 = \perp$, R_i for $i \in C$, and M_i for $i \in \overline{C}$.

Fig. 2. The simulator for the NIMPC protocol Π_1 for the class of abelian programs $\mathcal{A}_{\mathbb{G}}$.

$h_{f''}|_{\overline{C},0_{\overline{C}}}$ where $f''(x) = f'(x + \sum_{i\in\overline{C}} x'_i)$ and $0_{\overline{C}} = (0_{\mathbb{G}})_{i\in\overline{C}}$, the actual simulator $\mathsf{Sim}^{h|_{\overline{C},x_{\overline{C}}}}(C)$ described in Fig. 2 sets $x'_{\overline{C}} = 0_{\overline{C}}$ for simplicity.

We denote by $(R_C, M_{\overline{C}})(r)$ the joint distribution of the correlated randomness of C and the messages of $\overline{C}$ when Π_1 is executed on f and $x_{\overline{C}}$, where we specify the randomness $r \leftarrow\!\!\$\, \mathfrak{S}_{\mathbb{H}}^n$ used by the protocol. Similarly, we define $(R'_C, M'_{\overline{C}})(r)$ for $r \leftarrow\!\!\$\, \mathfrak{S}_{\mathbb{H}}^n$ as the joint distribution when Π_1 is executed on f' and $x'_{\overline{C}}$. It is sufficient to prove that there is a bijection $\phi : \mathfrak{S}_{\mathbb{H}}^n \to \mathfrak{S}_{\mathbb{H}}^n$ such that $(R_C, M_{\overline{C}})(r) = (R'_C, M'_{\overline{C}})(\phi(r))$ for all $r \in \mathfrak{S}_{\mathbb{H}}^n$.

The case of $C = \emptyset$. Let $a = \sum_{i\in[n]} x_i$ and $a' = \sum_{i\in[n]} x'_i$. Let $d_i = \sum_{j>i} x_j$ and $d'_i = \sum_{j>i} x'_j$ for $i \in [n]$, where we define $d_n = d'_n = 0_{\mathbb{G}}$. Let $S_f = \{(g, 1 - f(g)) \in \mathbb{H} \mid g \in \mathbb{G}\}$ and $S_{f'} = \{(g, 1 - f'(g)) \in \mathbb{H} \mid g \in \mathbb{G}\}$. From the definition, we have that $|S_f| = |S_{f'}| = |\mathbb{G}|$. It also follows from $h_f(x_1, \ldots, x_n) = h_{f'}(x'_1, \ldots, x'_n)$ that $(a, 0) \in S_f$ if and only if $(a', 0) \in S_{f'}$. Therefore, there is a permutation $\rho \in \mathfrak{S}_{\mathbb{H}}$ such that $\rho(a', 0) = (a, 0)$ and $\rho(S_{f'}) = S_f$. Define $\phi : \mathfrak{S}_{\mathbb{H}}^n \to \mathfrak{S}_{\mathbb{H}}^n$ as $\phi(\pi_1, \ldots, \pi_n) = (\pi'_1, \ldots, \pi'_n)$, where $\pi'_i = \pi_i \circ \tau_{-d_i} \circ \rho \circ \tau_{d'_i}$ for $i \in [n-1]$ and $\pi'_n = \pi_n$. Recall that τ_x is defined as $\tau_x(g, b) = (g + x, b)$ for $(g, b) \in \mathbb{H}$. From the definition of $\boldsymbol{B}_x$'s and permutation matrices, we have that $\boldsymbol{U}_{\pi'_i} = \boldsymbol{B}_{d'_i} \boldsymbol{U}_\rho \boldsymbol{B}_{-d_i} \boldsymbol{U}_{\pi_i}$. We fix $r = (\pi_1, \ldots, \pi_n) \in \mathfrak{S}_{\mathbb{H}}^n$ and simply write $M_i = M_i(r)$ and $M'_i = M'_i(\phi(r))$.

Then, we have $M_i = M'_i$ for all $i \in [n]$ from the following:

- $i = n$: From the definition of ρ, we have

$$\begin{aligned}
\sum_{s\in S_f} \boldsymbol{e}_s = \sum_{s'\in S_{f'}} \boldsymbol{e}_{\rho(s')} &\iff \boldsymbol{w}_f = \boldsymbol{U}_\rho^{-1} \boldsymbol{w}_{f'} \\
&\iff \boldsymbol{B}_{x_n} \boldsymbol{w}_f = (\boldsymbol{B}_{x_n} \boldsymbol{U}_\rho^{-1} \boldsymbol{B}_{-x'_n}) \boldsymbol{B}_{x'_n} \boldsymbol{w}_{f'} \\
&\iff \boldsymbol{U}_{\pi_{n-1}}^{-1} \boldsymbol{B}_{x_n} \boldsymbol{w}_f = \boldsymbol{U}_{\pi'_{n-1}}^{-1} \boldsymbol{B}_{x'_n} \boldsymbol{w}_{f'}.
\end{aligned}$$

- $1 < i < n$: We have

$$\begin{aligned}
\boldsymbol{U}_{\pi'_{i-1}}^{-1} \boldsymbol{B}_{x'_i} \boldsymbol{U}_{\pi'_i} &= \boldsymbol{U}_{\pi_{i-1}}^{-1} \boldsymbol{B}_{d_{i-1}} \boldsymbol{U}_\rho^{-1} \boldsymbol{B}_{-d'_{i-1}} \boldsymbol{B}_{x'_i} \boldsymbol{B}_{d'_i} \boldsymbol{U}_\rho \boldsymbol{B}_{-d_i} \boldsymbol{U}_{\pi_i} \\
&= \boldsymbol{U}_{\pi_{i-1}}^{-1} \boldsymbol{B}_{d_{i-1} - d_i} \boldsymbol{U}_{\pi_i} \\
&= \boldsymbol{U}_{\pi_{i-1}}^{-1} \boldsymbol{B}_{x_i} \boldsymbol{U}_{\pi_i}.
\end{aligned}$$

- $i = 1$: From the definition of ρ, we have

$$\begin{aligned}
(a, 0) = \rho(a', 0) &\iff {\boldsymbol{e}_{(a,0)}}^\top = {\boldsymbol{e}_{(a',0)}}^\top \boldsymbol{U}_\rho \\
&\iff {\boldsymbol{e}_{(x_1,0)}}^\top \boldsymbol{B}_{d_1} = {\boldsymbol{e}_{(x'_1,0)}}^\top \boldsymbol{B}_{d'_1} \boldsymbol{U}_\rho \\
&\iff {\boldsymbol{e}_{(x_1,0)}}^\top = {\boldsymbol{e}_{(x'_1,0)}}^\top \boldsymbol{U}_{\pi'_1} \boldsymbol{U}_{\pi_1}^{-1} \\
&\iff {\boldsymbol{e}_{(0_{\mathbb{G}},0)}}^\top \boldsymbol{B}_{x_1} \boldsymbol{U}_{\pi_1} = {\boldsymbol{e}_{(0_{\mathbb{G}},0)}}^\top \boldsymbol{B}_{x'_1} \boldsymbol{U}_{\pi'_1}
\end{aligned}$$

The case of $C \neq \emptyset$. Let $a = \sum_{j\in\overline{C}} (x_j - x'_j) \in \mathbb{G}$, $d_i = \sum_{j>i, j\in\overline{C}} (x'_j - x_j) + a \in \mathbb{G}$ for $i \in [n]$, where we define $d_n = a$. Since the residual functions

$h_f|_{\overline{C},x_{\overline{C}}}, h_{f'}|_{\overline{C},x'_{\overline{C}}}$ are identical to each other, we have

$$\begin{aligned} & f(g' + \sum_{j\in\overline{C}} x_j) = f'(g' + \sum_{j\in\overline{C}} x'_j)\ (\forall g' \in \mathbb{G}) \\ \iff & f(g) = f'(g-a)\ (\forall g \in \mathbb{G}). \end{aligned}$$

Define $\phi : \mathfrak{S}^n_{\mathbb{H}} \to \mathfrak{S}^n_{\mathbb{H}}$ as $\phi(\pi_1, \ldots, \pi_n) = (\pi'_1, \ldots, \pi'_n)$, where $\pi'_i = \pi_i \circ \tau_{d_i}$ for $i \in [n]$. Then, from the above observation we have that $\boldsymbol{B}_a \boldsymbol{w}_f = \boldsymbol{w}_{f'}$ since

$$\begin{aligned} \boldsymbol{B}_a \boldsymbol{w}_f = \boldsymbol{w}_{f'} & \iff \sum_{g\in\mathbb{G}} \boldsymbol{e}_{(g-a,1-f(g))} = \sum_{g'\in\mathbb{G}} \boldsymbol{e}_{(g',1-f'(g'))} \\ & \iff \sum_{g'\in\mathbb{G}} (\boldsymbol{e}_{(g',1-f(g'+a))} - \boldsymbol{e}_{(g',1-f'(g'))}) = 0 \\ & \iff f(g) = f'(g-a)\ (\forall g \in \mathbb{G}). \end{aligned}$$

Observe that for any $1 < i \le n$ and any $x \in \mathbb{G}$,

$$\boldsymbol{U}^{-1}_{\pi'_{i-1}} \boldsymbol{B}_x \boldsymbol{U}_{\pi'_i} = \boldsymbol{U}^{-1}_{\pi_{i-1}} \boldsymbol{B}_{-d_{i-1}} \boldsymbol{B}_x \boldsymbol{B}_{d_i} \boldsymbol{U}_{\pi_i} = \boldsymbol{U}^{-1}_{\pi_{i-1}} \boldsymbol{B}_{x+(d_i-d_{i-1})} \boldsymbol{U}_{\pi_i}. \tag{1}$$

We also have that

$$d_i - d_{i-1} = \begin{cases} x_i - x'_i, & \text{if } i \in \overline{C}, \\ 0_{\mathbb{G}}, & \text{otherwise.} \end{cases}$$

We fix $r = (\pi_1, \ldots, \pi_n) \in \mathfrak{S}^n_{\mathbb{H}}$ and simply write $R_i = R_i(r), R'_i = R'_i(\phi(r))$ for $i \in C$ and $M_i = M_i(r), M'_i = M'_i(\phi(r))$ for $i \in \overline{C}$. Now, we have $R_i = R'_i$ for $i \in C$ and $M_i = M'_i$ for $i \in \overline{C}$ from the following:

- $i = n$:
  - If $n \in C$, then $d_n = d_{n-1}$. Therefore, substituting $x = 0_{\mathbb{G}}$ and $x = s_j$ into (1), we have $\boldsymbol{U}^{-1}_{\pi'_{n-1}} \boldsymbol{U}_{\pi'_n} = \boldsymbol{U}^{-1}_{\pi_{n-1}} \boldsymbol{U}_{\pi_n}$ and $\boldsymbol{U}^{-1}_{\pi'_{n-1}} \boldsymbol{B}_{s_j} \boldsymbol{U}_{\pi'_n} = \boldsymbol{U}^{-1}_{\pi_{n-1}} \boldsymbol{B}_{s_j} \boldsymbol{U}_{\pi_n}$ for $j \in [m]$, respectively. Furthermore, since $d_n = a$, we also have
  $$\boldsymbol{U}^{-1}_{\pi'_n} \boldsymbol{w}_{f'} = \boldsymbol{U}^{-1}_{\pi_n} \boldsymbol{B}_{-d_n} \boldsymbol{w}_{f'} = \boldsymbol{U}^{-1}_{\pi_n} \boldsymbol{w}_f.$$
  - If $n \in \overline{C}$, then $d_n - d_{n-1} = x_n - x'_n$, i.e., $x'_n - d_{n-1} = x_n - a$. Therefore,
  $$\boldsymbol{U}^{-1}_{\pi'_{n-1}} \boldsymbol{B}_{x'_n} \boldsymbol{w}_{f'} = \boldsymbol{U}^{-1}_{\pi_{n-1}} \boldsymbol{B}_{x'_n - d_{n-1}} \boldsymbol{B}_a \boldsymbol{w}_f = \boldsymbol{U}^{-1}_{\pi_{n-1}} \boldsymbol{B}_{x_n} \boldsymbol{w}_f.$$
- $1 < i < n$:
  - If $i \in C$, then $d_i = d_{i-1}$. Therefore, substituting $x = 0_{\mathbb{G}}$ and $x = s_j$ into (1), we have $\boldsymbol{U}^{-1}_{\pi'_{i-1}} \boldsymbol{U}_{\pi'_i} = \boldsymbol{U}^{-1}_{\pi_{i-1}} \boldsymbol{U}_{\pi_i}$ and $\boldsymbol{U}^{-1}_{\pi'_{i-1}} \boldsymbol{B}_{s_j} \boldsymbol{U}_{\pi'_i} = \boldsymbol{U}^{-1}_{\pi_{i-1}} \boldsymbol{B}_{s_j} \boldsymbol{U}_{\pi_i}$ for $j \in [m]$, respectively.
  - If $i \in \overline{C}$, then $d_i - d_{i-1} = x_i - x'_i$. Therefore, substituting $x = x'_i$ into (1), we have $\boldsymbol{U}^{-1}_{\pi'_{i-1}} \boldsymbol{B}_{x'_i} \boldsymbol{U}_{\pi'_i} = \boldsymbol{U}^{-1}_{\pi_{i-1}} \boldsymbol{B}_{x_i} \boldsymbol{U}_{\pi_i}$.
- $i = 1$:

- If $1 \in C$, then $d_1 = 0_{\mathbb{G}}$ and hence $\boldsymbol{U}_{\pi'_1} = \boldsymbol{U}_{\pi_1}$.
- If $1 \in \overline{C}$, then $d_1 = x_1 - x'_1$ and hence

$$\boldsymbol{e}_{(0_{\mathbb{G}},0)}{}^{\top} \boldsymbol{B}_{x'_1} \boldsymbol{U}_{\pi'_1} = \boldsymbol{e}_{(0_{\mathbb{G}},0)}{}^{\top} \boldsymbol{B}_{x'_1} \boldsymbol{B}_{x_1 - x'_1} \boldsymbol{U}_{\pi_1} = \boldsymbol{e}_{(0_{\mathbb{G}},0)}{}^{\top} \boldsymbol{B}_{x_1} \boldsymbol{U}_{\pi_1}.$$

Communication complexity. The maximum component of on-line communication is the messages M_i of the players i with $1 < i < n$, each of which consists of one permutation matrix over $\mathbb{F}_2$ of size $|\mathbb{H}|$. The maximum component of off-line communication is the randomness R_n of the player n, which consists of $|S|+1$ permutation matrices over $\mathbb{F}_2$ of size $|\mathbb{H}|$ and a vector over $\mathbb{F}_2$ of dimension $|\mathbb{H}|$. Note that every permutation matrix can be expressed by $|\mathbb{H}|\lceil \log |\mathbb{H}| \rceil$ bits. Since $|\mathbb{H}| = 2|\mathbb{G}|$, the protocol achieves the communication complexity in the statement. □

4.3 Abelian Programs with Limited Input Domains

Next, we present a t-robust protocol for the class $\mathcal{A}_{\mathbb{G}}^{S_1,\ldots,S_n}$ of abelian programs with limited input domains. As mentioned in Section 3.2, NIMPC protocols for $\mathcal{A}_{\mathbb{G}}$ is not directly applicable to $\mathcal{A}_{\mathbb{G}}^{S_1,\ldots,S_n}$. Nevertheless, it is possible to obtain protocols for $\mathcal{A}_{\mathbb{G}}^{S_1,\ldots,S_n}$ from the ones for $\mathcal{A}_{\mathbb{G}}$ with the help of the BKR transformation [6]. Note that we will show in Section 6 that their NIMPC protocol for outputting-message functions used in the transformation does not satisfy the desired security. Therefore, it is necessary to replace their protocol with our modified protocol given also in Section 6 when actually applying the BKR transformation.

Proposition 1 ([6]). *If there is a 0-robust NIMPC protocol for a class of functions $\mathcal{H}$ with communication complexity α, then for any t, there is a t-robust NIMPC protocol for $\mathcal{H}$ with communication complexity $p^{t+O(1)}\alpha$, where d is the maximum size of the input domains of functions in $\mathcal{H}$ and p is the smallest prime power such that $p \geq \max\{n, d\}$.*

Clearly, the protocol in Theorem 1 satisfies 0-robustness for $\mathcal{A}_{\mathbb{G}}$. Since the simulator to prove 0-robustness only receives the output of the function rather than a residual function, the simulation works regardless of whether the input domains are limited or not. Consequently, the 0-robust protocol also satisfies 0-robustness even for $\mathcal{A}_{\mathbb{G}}^{S_1,\ldots,S_n}$. By applying the BKR transformation to it, we obtain a t-robust protocol for $\mathcal{A}_{\mathbb{G}}^{S_1,\ldots,S_n}$ with some overhead in communication complexity.

Corollary 1. *Let $\mathbb{G}$ be a finite abelian group and $S_1, \ldots, S_n$ be subsets of $\mathbb{G}$. Then, there exists a t-robust NIMPC protocol Π for $\mathcal{A}_{\mathbb{G}}^{S_1,\ldots,S_n}$ such that $\mathsf{CC}(\Pi) = |\mathbb{G}|(\log |\mathbb{G}|)^2 p^{t+O(1)}$, where p is a prime power with $p \geq \max\{n, |S_1|, \ldots, |S_n|\}$.*

The authors of [3] present another method to limit the inputs of players while it only works for abelian programs. However, this method applied to Theorem 1 only provides us with a protocol with communication complexity $|\mathbb{G}|^2(\log |\mathbb{G}|)^2(nd)^{t+O(1)}$ and hence the protocol in Corollary 1 is more efficient.

Example 1. We apply our protocol for $\mathcal{A}_{\mathbb{G}}^{S_1,\ldots,S_n}$ to symmetric functions. We have noted in Section 3.2 that the class of all symmetric functions over $[d]^n$ is equivalent to $\mathcal{A}_{\mathbb{G}}^{S_1,\ldots,S_n}$ for $\mathbb{G} = (\mathbb{Z}_{n+1})^d$ and $S_1 = \cdots = S_n = \{\boldsymbol{e}_1, \ldots, \boldsymbol{e}_d\} \subseteq \mathbb{G}$, where $\boldsymbol{e}_i \in (\mathbb{Z}_{n+1})^d$ is the i-th unit vector. Note that $\mathbb{G}$ can be generated by at most d elements. Therefore, we obtain a t-robust NIMPC protocol Π for the class of all symmetric functions over $[d]^n$ such that

$$\mathsf{CC}(\Pi) = p^{t+O(1)} \times O((n+1)^d d^2 (\log n)) = (\max\{n,d\})^{(1+o(1))t} n^{(1+o(1))d}.$$

Here, p is the smallest prime power such that $p \geq \max\{n,d\}$, which is chosen as $p = O(\max\{n,d\})$. This result is better than $(nd)^{(1+o(1))t} n^{(3+o(1))d}$ [3] and also improves $n^{(\log n + O(1))d}$ [6] if $t = o(\log n)$. In the case of boolean symmetric functions, i.e., $d = 2$, our protocol is more efficient than another t-robust protocol with communication complexity $n^{\log\log n + \log t + O(1)}$ [6] if $t = o(\log\log n)$.

5 New NIMPC Protocols for Linear Classifiers

5.1 Formalization of Linear Classifiers

Let $\mathbb{F}_q$ be a finite field and $S_1, \ldots, S_n$ be subsets of $\mathbb{F}_q$. Let $\mathcal{X}_i = S_i$ for $i \in [n]$ and $\mathcal{X} = \mathcal{X}_1 \times \cdots \times \mathcal{X}_n$. We say that a function $f : \mathbb{F}_q \to \{0,1\}$ is proper if $|f^{-1}(1)| \notin \{0, q\}$. Define the linear classifier $h_{f,\boldsymbol{w}} : \mathcal{X} \to \{0,1\}$ associated with $f : \mathbb{F}_q \to \{0,1\}$ and $\boldsymbol{w} = (w_i)_{i\in[n]} \in \mathbb{F}_q^n$ as $h_{f,\boldsymbol{w}}(x_1, \ldots, x_n) = f(\sum_{i\in[n]} w_i x_i)$. We then define $\mathcal{L}_{\mathbb{F}_q}^{S_1,\ldots,S_n}$ be the class of all the linear classifiers, that is,

$$\mathcal{L}_{\mathbb{F}_q}^{S_1,\ldots,S_n} = \{h_{f,\boldsymbol{w}} \mid f : \mathbb{F}_q \to \{0,1\} \text{ is proper and } \boldsymbol{w} \in \mathbb{F}_q^n\}.$$

We simply write $\mathcal{L}_{\mathbb{F}_q}$ if $S_1 = \cdots = S_n = \mathbb{F}_q$.

We note that focusing on proper functions does not limit the expressive power of linear classifiers. If $|f^{-1}(1)| = 0$, i.e., $f(x) = 0$ for all $x \in \mathbb{F}_q$, then the linear classifier $h_{f,\boldsymbol{w}}$ for any $\boldsymbol{w} \in \mathbb{F}_q^n$ is equivalent to $h_{f_0,\boldsymbol{0}}$, where $f_0(x)$ outputs 0 if and only if $x = 0$. Similarly, any linear classifier $h_{f,\boldsymbol{w}}$ with $|f^{-1}(1)| = q$ is equivalent to a linear classifier associated with some proper function.

In the same manner as abelian programs, we can extend linear classifiers to the ones outputting more than one bits by computing each output bit separately. Specifically, we first extend the definition of proper functions as follows. A function $f : \mathbb{F}_q \to \{0,1\}^m$ is said to be proper if $p_i \circ f : \mathbb{F}_q \to \{0,1\}$ is proper (in the above sense) for every $i \in [m]$, where $p_i : \{0,1\}^m \ni (b_j)_{j\in[m]} \mapsto b_i \in \{0,1\}$ is the i-th projection. Now, for $S_1, \ldots, S_n \subseteq \mathbb{F}_q$, we define the class of linear classifiers outputting m bits as

$$\mathcal{H}_m := \{h_{f,\boldsymbol{w}}^m : \mathcal{X} \to \{0,1\}^m \mid f : \mathbb{F}_q \to \{0,1\}^m \text{ is proper and } \boldsymbol{w} \in \mathbb{F}_q^n\},$$

where $\mathcal{X} = \prod_{i\in[n]} S_i$ and $h_{f,\boldsymbol{w}}^m(x_1, \ldots, x_n) = f(\sum_{i\in[n]} w_i x_i)$. From the definition of proper functions, we have $\mathcal{H}_m = \{h : \mathcal{X} \to \{0,1\}^m \mid h = (h_1, \ldots, h_m),\ h_i \in \mathcal{L}_{\mathbb{F}_q}^{S_1,\ldots,S_n}\}$ for $\mathcal{L}_{\mathbb{F}_q}^{S_1,\ldots,S_n}$. Hence, given an NIMPC protocol Π for $\mathcal{L}_{\mathbb{F}_q}^{S_1,\ldots,S_n}$, we can

construct a protocol Π_m for $\mathcal{H}_m$ with m times higher communication complexity by running Π for each $h_i \in \mathcal{L}_{\mathbb{F}_q}^{S_1,\ldots,S_n}$.

We assume in the above that all the arithmetic operations are performed in a finite field. For real-world applications, it is necessary to deal with weights and inputs expressed as real numbers. We can still use the above linear classifiers by embedding these values into a sufficiently large prime field using a fixed-point number representation.

5.2 NIMPC protocols for Linear Classifiers

We first propose a fully robust protocol for $\mathcal{L}_{\mathbb{F}_q}$ (Theorem 2) and then construct a protocol for $\mathcal{L}_{\mathbb{F}_q}^{S_1,\ldots,S_n}$ by applying the BKR transformation (Corollary 2).

Let $f : \mathbb{F}_q \to \{0,1\}$ be a proper function and $\boldsymbol{w} \in \mathbb{F}_q^n$ be a vector of weights. In Section 2.2, we have already shown a construction of a fully robust protocol for the class consisting only of the specific linear classifier $\{h_{f,\boldsymbol{w}}\} \subseteq \mathcal{L}_{\mathbb{F}_q}$. In that protocol, if $f^{-1}(1) = \{u_1, \ldots, u_k\}$, the evaluator outputs $f(\sum_{i\in[n]} w_i x_i)$ according to the number of zeros of $\boldsymbol{u}_0 = \boldsymbol{u} - \sum_{i\in[n]} \mathbf{1}_k w_i x_i$, where $\boldsymbol{u} = [u_1, \ldots, u_k]^\top \in \mathbb{F}_q^k$ and $\mathbf{1}_N$ denotes the vector of dimension N whose entries are all one. It can be seen that the technique for randomizing $\boldsymbol{u}_0$ is equivalent to multiplying $\boldsymbol{u}_0$ by $\boldsymbol{T} = \mathrm{diag}(r_1, \ldots, r_k)\boldsymbol{U}_\pi \in \mathbb{F}_q^{k\times k}$ for $r_i \leftarrow\!\!\$\, \mathbb{F}_q \setminus \{0\}$ $(i \in [k])$ and $\pi \leftarrow\!\!\$\, \mathfrak{S}_{[k]}$. However, there remains the problem that the dimension $k = |f^{-1}(1)|$ is assumed to be constant, which is why it does not work for the class of all the linear classifiers.

A simple solution to hide k is *padding* the vector $\boldsymbol{u}$ with some $q-k$ elements $u_j \in \mathbb{F}_q$ $(k < j \le q)$ to ensure that its dimension should be q. Accordingly, we redefine $\boldsymbol{u}_0$ as $\boldsymbol{u}_0 = \boldsymbol{u} - \sum_{i\in[n]} \mathbf{1}_q w_i x_i$. In this solution, however, there would exist some inputs x_i such that $\sum_{i\in[n]} w_i x_i \notin f^{-1}(1)$ but $u_{k+1} = \sum_{i\in[n]} w_i x_i$ and then the protocol incorrectly outputs 1. To overcome it, we consider an extension field $\mathbb{K}$ of $\mathbb{F}_q$ and randomly choose u_j $(k < j \le q)$ from $\mathbb{K} \setminus \mathbb{F}_q$. Since $\sum_{i\in[n]} w_i x_i \in \mathbb{F}_q$, the above error never happens. Accordingly, we now uniformly select a permutation π and elements r_i $(i \in [q])$ from $\mathfrak{S}_{[q]}$ and $\mathbb{K}\setminus\{0\}$, respectively, and set $\boldsymbol{T} = \mathrm{diag}(r_1, \ldots, r_q)\boldsymbol{U}_\pi$. We also sample $\boldsymbol{s}_i$ $(i \in [n])$ from $\mathbb{K}^q$. The important point is that regardless of whether $x \in \mathbb{F}_q \setminus \{0\}$ or $x \in \mathbb{K}\setminus\mathbb{F}_q$, the product rx is uniformly distributed over $\mathbb{K} \setminus \{0\}$ if $r \leftarrow\!\!\$\, \mathbb{K} \setminus \{0\}$. This is why we consider an extension *field* of $\mathbb{F}_q$ rather than an extension *ring*. The above modified protocol communicates at most two vectors over $\mathbb{K}$ of dimension q. Since we can choose $\mathbb{K}$ as any extension field of $\mathbb{F}_q$, we may assume that $|\mathbb{K}| = q^2$. Therefore, the communication complexity is at most $O(q \log q)$.

Theorem 2. *Let $\mathbb{F}_q$ be a finite field. Let $\mathcal{X}_1 = \cdots = \mathcal{X}_n = \mathbb{F}_q$ and $\mathcal{X} = \mathcal{X}_1 \times \cdots \times \mathcal{X}_n$. Then, the protocol Π_2 described in Fig. 3 is a fully robust NIMPC protocol for $\mathcal{L}_{\mathbb{F}_q}$ such that $\mathsf{CC}_{\mathsf{on}}(\Pi_2) = 2q\lceil 2\log q\rceil$ and $\mathsf{CC}_{\mathsf{off}}(\Pi_2) = q\lceil 2\log q\rceil$.*

Proof. **Correctness.** Let $h = h_{f,\boldsymbol{w}}$ be a function to compute. Let $S = f^{-1}(1) = \{u_1, \ldots, u_k\}$, where $k = |S|$ and $u_i \in \mathbb{F}_q$ for $i \in [k]$. Correctness follows from the following observation: $h(x_1, \ldots, x_n) = 1$ if and only if the number of 0's in

Protocol Π_2 for $\mathcal{L}_{\mathbb{F}_q}$

$\mathsf{Gen}(h_{f,\boldsymbol{w}})$**:**

1. Set $S = f^{-1}(1) = \{u_1, \ldots, u_k\} \subseteq \mathbb{F}_q$ and choose $u_j \leftarrow_{\$} \mathbb{K} \setminus \mathbb{F}_q$ $(k < j \leq q)$.
2. Set $\boldsymbol{u} = [u_1, \ldots, u_q]^\top \in \mathbb{K}^q$.
3. Choose $r_i \leftarrow_{\$} \mathbb{K} \setminus \{0\}$ $(i \in [q])$, $\pi \leftarrow_{\$} \mathfrak{S}_{[q]}$, and $\boldsymbol{s}_i \leftarrow_{\$} \mathbb{K}^q$ $(i \in [n])$.
4. Set $\boldsymbol{T} = \mathrm{diag}(r_1, \ldots, r_q)\boldsymbol{U}_\pi \in \mathbb{K}^{q \times q}$.
5. Output $R_0 = \boldsymbol{T}\boldsymbol{u} + \sum_{i \in [n]} \boldsymbol{s}_i$ and $R_i = (\boldsymbol{T}\mathbf{1}_q w_i, \boldsymbol{s}_i)$ for $i \in [n]$.

$\mathsf{Enc}_i(x_i, R_i)$**:**

1. Parse R_i as $(\boldsymbol{\rho}_{i1}, \boldsymbol{\rho}_{i2})$, where $\boldsymbol{\rho}_{i1}, \boldsymbol{\rho}_{i2} \in \mathbb{K}^q$.
2. Output $M_i = \boldsymbol{\rho}_{i1} x_i + \boldsymbol{\rho}_{i2}$.

$\mathsf{Dec}(R_0, M_1, \ldots, M_n)$**:**

1. If at least one entry of $R_0 - \sum_{i \in [n]} M_i$ is zero, then output 1 and otherwise output 0.

Fig. 3. The NIMPC protocol Π_2 for the class of linear classifiers $\mathcal{L}_{\mathbb{F}_q}$.

$\boldsymbol{u} - \sum_{i \in [n]} \mathbf{1}_q w_i x_i$ is at least one, which is in turn equivalent to the condition that the number of 0's in $R_0 - \sum_{i \in [n]} M_i = \boldsymbol{T}(\boldsymbol{u} - \sum_{i \in [n]} \mathbf{1}_q w_i x_i)$ is at least one.

Robustness. Let $C \subseteq [n]$. The adversary's view is

$$(\boldsymbol{T}\boldsymbol{u} + \sum_{i \in [n]} \boldsymbol{s}_i; (\boldsymbol{T}\mathbf{1}_q w_i)_{i \in C}, (\boldsymbol{s}_i)_{i \in C}; (\boldsymbol{T}\mathbf{1}_q w_i x_i + \boldsymbol{s}_i)_{i \in \overline{C}}),$$

where $\boldsymbol{T} = \mathrm{diag}(r_1, \ldots, r_q)\boldsymbol{P}_\pi$, $r_i \leftarrow_{\$} \mathbb{K} \setminus \{0\}$ $(i \in [q])$, $\pi \leftarrow_{\$} \mathfrak{S}_q$, $\boldsymbol{u} = [u_1, \ldots, u_q]^\top$, $u_j \leftarrow_{\$} \mathbb{K} \setminus \mathbb{F}_q$ $(k < j \leq q)$, and $\boldsymbol{s}_i \leftarrow_{\$} \mathbb{K}^q$ $(i \in [n])$. It is sufficient to show that the following distribution can be perfectly simulated:

$$(\boldsymbol{T}\boldsymbol{v}; (\boldsymbol{T}\mathbf{1}_q w_i)_{i \in C}, (\boldsymbol{s}_i)_{i \in C}; (\boldsymbol{T}\mathbf{1}_q w_i x_i + \boldsymbol{s}_i)_{i \in \overline{C}}), \tag{2}$$

where we set $\overline{\gamma} = \boldsymbol{w}_{\overline{C}}^\top \boldsymbol{x}_{\overline{C}} = \sum_{i \in \overline{C}} w_i x_i$ and $\boldsymbol{v} = \boldsymbol{u} - \mathbf{1}_q \overline{\gamma}$. This is because the original view can be obtained by computing $\boldsymbol{T}\boldsymbol{u} + \sum_{i \in [n]} \boldsymbol{s}_i = \boldsymbol{T}\boldsymbol{v} + \sum_{i \in C} \boldsymbol{s}_i + \sum_{i \in \overline{C}} (\boldsymbol{T}\mathbf{1}_q w_i x_i + \boldsymbol{s}_i)$.

The case of $C = \emptyset$. We can see below that the distribution of (2) is simulated by the simulator described in Fig. 4:

- If $h(x_1, \ldots, x_n) = 0$, then $u_j \neq \overline{\gamma}$ for all $j \in [k]$. In addition, $u_j \neq \overline{\gamma}$ for all $k < j \leq q$ since the u_j's are selected from $\mathbb{K} \setminus \mathbb{F}_q$ and $\overline{\gamma}$ is an element of $\mathbb{F}_q$. Therefore, $\boldsymbol{v} \in (\mathbb{K} \setminus \{0\})^q$ and hence $\boldsymbol{T}\boldsymbol{v}$ is uniformly distributed over $(\mathbb{K} \setminus \{0\})^q$. Since the $\boldsymbol{s}_i$'s are chosen independent of $\boldsymbol{T}$, we have that $(\boldsymbol{T}\boldsymbol{v}; (\boldsymbol{T}\mathbf{1}_q w_i x_i + \boldsymbol{s}_i)_{i \in [n]}) \equiv (\widetilde{\boldsymbol{v}}; (\widetilde{\boldsymbol{s}}_i)_{i \in [n]})$ for vectors $\widetilde{\boldsymbol{v}}$ and $\widetilde{\boldsymbol{s}}_i$ $(i \in [n])$ sampled by the simulator.
- If $h(x_1, \ldots, x_n) = 1$, then there is the unique index $j \in [k]$ such that $u_j = \overline{\gamma}$. Again, it holds that $u_j \neq \overline{\gamma}$ for all $k < j \leq q$. Therefore, the number of 0's in $\boldsymbol{v}$ is exactly one and $\boldsymbol{T}\boldsymbol{v}$ is uniformly distributed over the set of all the

Simulator $\mathsf{Sim}^{h|_{\overline{C},x_{\overline{C}}}}(C)$ for Π_2

Input. A set of colluding players C.
Oracle access. The residual function $h|_{\overline{C},x_{\overline{C}}}$ for the inputs $x_{\overline{C}}$ of honest players.
Output. Correlated randomness R_0 and $(R_i)_{i\in C}$ of the colluding players and messages $(M_i)_{i\in\overline{C}}$ of the honest players.
Algorithm.

- If $C=\emptyset$:
  1. Choose $(\widetilde{v}_j)_{j\in[q]} \in \mathbb{K}^q$ uniformly at random from $(\mathbb{K}\setminus\{0\})^q$ if $h(x_1,\ldots,x_n)=0$ and otherwise from $\{0\}\times(\mathbb{K}\setminus\{0\})^{q-1}$.
  2. Choose $\pi \leftarrow\!\!\$\, \mathfrak{S}_q$ and set $\widetilde{\boldsymbol{v}}=(\widetilde{v}_{\pi(j)})_{j\in[q]}$.
  3. Choose $\widetilde{\boldsymbol{s}}_i \leftarrow\!\!\$\, \mathbb{K}^q$ $(i\in[n])$.
  4. Output $R_0=\widetilde{\boldsymbol{v}}+\sum_{i\in[n]}\widetilde{\boldsymbol{s}}_i$ and $M_i=\widetilde{\boldsymbol{s}}_i$ for $i\in[n]$.
- If $C\neq\emptyset$:
  1. For $\boldsymbol{x}_C=(x_i)_{i\in C}\in\mathbb{F}_q^{|C|}$, construct a set $\Delta_{\boldsymbol{x}_C}$ using the oracle $h|_{\overline{C},x_{\overline{C}}}$ as
  $$\Delta_{\boldsymbol{x}_C}=\{\alpha\in\mathbb{F}_q \mid h|_{\overline{C},\boldsymbol{x}_{\overline{C}}}(\boldsymbol{x}_C\alpha)=1\}.$$
  2. Construct a set Γ as $\Gamma=\{\boldsymbol{x}_C\in\mathbb{F}_q^{|C|} \mid |\Delta_{\boldsymbol{x}_C}|\in\{0,q\}\}$.
  3. Do the following.
     - If $|\Gamma|=q^{|C|}$:
       (a) Set $\widetilde{\boldsymbol{w}}_C=\boldsymbol{0}$.
       (b) Choose $(\widetilde{v}_j)_{j\in[q]} \in \mathbb{K}^q$ uniformly at random from $(\mathbb{K}\setminus\{0\})^q$ if $|\Delta_{\boldsymbol{x}_C}|=0$ for all $\boldsymbol{x}_C$ and otherwise from $\{0\}\times(\mathbb{K}\setminus\{0\})^{q-1}$.
       (c) Choose $\pi \leftarrow\!\!\$\, \mathfrak{S}_q$ and set $\widetilde{\boldsymbol{v}}=(\widetilde{v}_{\pi(j)})_{j\in[q]}$.
       (d) Choose $\widetilde{\boldsymbol{s}}_i \leftarrow\!\!\$\, \mathbb{K}^q$ $(i\in[n])$.
       (e) Output $R_0=\widetilde{\boldsymbol{v}}+\sum_{i\in[n]}\widetilde{\boldsymbol{s}}_i$, $R_i=(\boldsymbol{0}_q,\widetilde{\boldsymbol{s}}_i)$ for $i\in C$, and $M_i=\widetilde{\boldsymbol{s}}_i$ for $i\in\overline{C}$.
     - If $|\Gamma|\neq q^{|C|}$:
       (a) Fix $\widetilde{\boldsymbol{w}}_C=(\widetilde{w}_i)\in\Gamma^{\perp}\setminus\{\boldsymbol{0}\}$.
       (b) Fix $\boldsymbol{x}_C=(x_i)_{i\in C}\in\mathbb{F}_q^{|C|}\setminus\Gamma$ and write $\Delta_{\boldsymbol{x}_C}=\{\alpha_1,\ldots,\alpha_k\}$, where $k=|\Delta_{\boldsymbol{x}_C}|$.
       (c) Set $\widetilde{v}_j=\alpha_j(\sum_{i\in C}\widetilde{w}_i x_i)$ $(j\in[k])$, choose $\widetilde{v}_j \leftarrow\!\!\$\, \mathbb{K}\setminus\mathbb{F}_q$ $(k<j\le q)$, and set $\widetilde{\boldsymbol{v}}=(\widetilde{v}_j)_{j\in[q]}$.
       (d) Choose $\widetilde{r}_j \leftarrow\!\!\$\, \mathbb{K}\setminus\{0\}$ $(j\in[q])$, $\widetilde{\pi} \leftarrow\!\!\$\, \mathfrak{S}_{[q]}$, and $\widetilde{\boldsymbol{s}}_i \leftarrow\!\!\$\, \mathbb{K}^q$ $(i\in[n])$ and set $\widetilde{\boldsymbol{T}}=\mathrm{diag}(\widetilde{r}_1,\ldots,\widetilde{r}_q)\boldsymbol{U}_{\widetilde{\pi}}$
       (e) Output $R_0=\widetilde{\boldsymbol{T}}\widetilde{\boldsymbol{v}}+\sum_{i\in[n]}\widetilde{\boldsymbol{s}}_i$, $R_i=(\widetilde{\boldsymbol{T}}\boldsymbol{1}_q\widetilde{w}_i,\widetilde{\boldsymbol{s}}_i)$ for $i\in C$, and $M_i=\widetilde{\boldsymbol{s}}_i$ for $i\in\overline{C}$.

Fig. 4. The simulator for the NIMPC protocol Π_2 for the class of linear classifiers $\mathcal{L}_{\mathbb{F}_q}$.

vectors of $\mathbb{K}^q$ of Hamming weight $q-1$. Since the $\boldsymbol{s}_i$'s are chosen independent of $\boldsymbol{T}$, we have that $(\boldsymbol{T}\boldsymbol{v}; (\boldsymbol{T}\boldsymbol{1}_q w_i x_i + \boldsymbol{s}_i)_{i\in[n]}) \equiv (\widetilde{\boldsymbol{v}}; (\widetilde{\boldsymbol{s}}_i)_{i\in[n]})$ for vectors $\widetilde{\boldsymbol{v}}$ and $\widetilde{\boldsymbol{s}}_i$ $(i \in [n])$ sampled by the simulator.

The case of $C \neq \emptyset$. To begin with, observe that $\Delta_{\boldsymbol{x}_C}$ constructed by the simulator described in Fig. 4 satisfies $|\Delta_{\boldsymbol{x}_C}| \in \{0, q\}$ if and only if $\sum_{i\in C} w_i x_i = 0$ for any $\boldsymbol{x}_C = (x_i)_{i\in C}$. Indeed, if $\sum_{i\in C} w_i x_i = 0$, then for every $\alpha \in \mathbb{F}_q$, we have that

$$\begin{aligned} h|_{\overline{C},\boldsymbol{x}_{\overline{C}}}(\boldsymbol{x}_C\alpha) = 1 &\iff \sum_{i\in C} w_i x_i \alpha = u_j - \overline{\gamma}\ (\exists j \in [k]) \\ &\iff 0 = u_j - \overline{\gamma}\ (\exists j \in [k]) \\ &\iff h|_{\overline{C},\boldsymbol{x}_{\overline{C}}}(\boldsymbol{0}) = 1. \end{aligned}$$

Therefore, $h|_{\overline{C},\boldsymbol{x}_{\overline{C}}}(\boldsymbol{x}_C\alpha) = h|_{\overline{C},\boldsymbol{x}_{\overline{C}}}(\boldsymbol{0})$ for all $\alpha \in \mathbb{F}_q$ and hence $|\Delta_{x_C}|$ is either q or 0 depending on whether $h|_{\overline{C},x_{\overline{C}}}(\boldsymbol{0}) = 1$ or not. Conversely, if $\sum_{i\in C} w_i x_i = \delta \neq 0$, then for every $\alpha \in \mathbb{F}_q$, we have that

$$\begin{aligned} h|_{\overline{C},\boldsymbol{x}_{\overline{C}}}(\boldsymbol{x}_C\alpha) = 1 &\iff \sum_{i\in C} w_i x_i \alpha = u_j - \overline{\gamma}\ (\exists j \in [k]) \\ &\iff \alpha = \delta^{-1}(u_j - \overline{\gamma})\ (\exists j \in [k]). \end{aligned}$$

Therefore, $|\Delta_{x_C}| = k = |S| = |f^{-1}(1)| \notin \{0, q\}$.

It is then possible to determine $\boldsymbol{w}_C$ up to a scalar multiple from the size of the set Γ. Indeed, if $\boldsymbol{w}_C = \boldsymbol{0}$, then $|\Delta_{\boldsymbol{x}_C}| \in \{0, q\}$ for all $\boldsymbol{x}_C$ and hence $|\Gamma| = q^{|C|}$. If $\boldsymbol{w}_C \neq \boldsymbol{0}$, then a vector $\boldsymbol{x}_C$ satisfies $|\Delta_{\boldsymbol{x}_C}| \in \{0, q\}$ if and only if it is orthogonal to $\boldsymbol{w}_C$ and hence $|\Gamma| = q^{|C|-1}$. In the latter case, we have that $\Gamma^\perp = \boldsymbol{w}_C \cdot \mathbb{F}_q := \{\boldsymbol{w}_C\beta \mid \beta \in \mathbb{F}_q\}$ and so any non-zero vector in $\Gamma^\perp$ is a salar multiple of $\boldsymbol{w}_C$.

Now, we can see that the simulator in Fig. 4 simulates the distribution of (2):

- If $\Gamma = q^{|C|}$, we surely know that $\boldsymbol{w}_C = \boldsymbol{0}$ and particularly, $\widetilde{\boldsymbol{w}}_C = \boldsymbol{w}_C$. In this case, we should have either $|\Delta_{\boldsymbol{x}_C}| = 0$ for all $\boldsymbol{x}_C$ or $|\Delta_{\boldsymbol{x}_C}| = q$ for all $\boldsymbol{x}_C$ since $|\Delta_{\boldsymbol{x}_C}|$ only depends on $h|_{\overline{C},\boldsymbol{x}_{\overline{C}}}(\boldsymbol{0})$ regardless of $\boldsymbol{x}_C$. If $|\Delta_{\boldsymbol{x}_C}| = 0$ for all $\boldsymbol{x}_C$, then it should hold that $h|_{\overline{C},\boldsymbol{x}_{\overline{C}}}(\boldsymbol{0}) = 0$ and $\overline{\gamma} \notin S$. Then, $\boldsymbol{v} \in (\mathbb{K} \setminus \{0\})^q$ and hence $\boldsymbol{T}\boldsymbol{v}$ is uniformly distributed over $(\mathbb{K} \setminus \{0\})^q$. Otherwise, it should hold that $h|_{\overline{C},\boldsymbol{x}_{\overline{C}}}(\boldsymbol{0}) = 1$ and $\overline{\gamma} \in S$. Then, the number of 0's in $\boldsymbol{v}$ is exactly one and $\boldsymbol{T}\boldsymbol{v}$ is uniformly distributed over the set of all the vectors of $\mathbb{K}^q$ of Hamming weight $q-1$. Since the $\boldsymbol{s}_i$'s are chosen independent of $\boldsymbol{T}$, the distribution of (2) is simulated by $(\widetilde{\boldsymbol{v}}; (\boldsymbol{0}_q)_{i\in C}, (\widetilde{\boldsymbol{s}}_i)_{i\in C}; (\widetilde{\boldsymbol{s}}_i)_{i\in\overline{C}})$ for $\widetilde{\boldsymbol{v}}$ and $\widetilde{\boldsymbol{s}}_i$ $(i \in [n])$ sampled by the simulator.
- If $\Gamma \neq q^{|C|}$, we surely know that $\Gamma^\perp = \boldsymbol{w}_C \cdot \mathbb{F}_q$. The vector $\widetilde{\boldsymbol{w}}_C$ sampled by the simulator can be expressed as $\widetilde{\boldsymbol{w}}_C = \boldsymbol{w}_C \cdot \beta$ for some $\beta \in \mathbb{F}_q \setminus \{0\}$. Let $v_j = u_j - \overline{\gamma}$ for $j \in [k]$. Let $\boldsymbol{x}_C$ be inputs chosen by the simulator such that $\boldsymbol{x}_C \notin \Gamma$ and write $\Delta_{\boldsymbol{x}_C} = \{\alpha_1, \ldots, \alpha_k\}$ for the inputs $\boldsymbol{x}_C$. Let

$\gamma = \sum_{i \in C} \widetilde{w}_i x_i \neq 0$ and $\delta = \sum_{i \in C} w_i x_i$. It holds that $\gamma = \delta\beta$. Note that the simulator knows $k = |S|$, $\widetilde{\boldsymbol{w}}_C$, α_j $(j \in [k])$, and γ.
We have that the values $\widetilde{v}_j := \alpha_j \gamma$ set by the simulator determine the v_j's up to a scalar and a permutation. Specifically, there exists $\tau \in \mathfrak{S}_{[k]} \subseteq \mathfrak{S}_{[q]}$ such that $\widetilde{v}_j = \beta v_{\tau(j)}$ for all $j \in [k]$ since for every $\alpha \in \Delta_{x_C} = \{\alpha_1, \ldots, \alpha_k\}$, it holds that

$$\begin{aligned}
\alpha \in \Delta_{x_C} &\iff \delta\alpha = v_j \ (\exists j \in [k]) \\
&\iff \alpha = \delta^{-1} v_j = \beta\gamma^{-1} v_j \ (\exists j \in [k]) \\
&\iff \alpha\gamma = \beta v_j \ (\exists j \in [k])
\end{aligned}$$

Since $\widetilde{v}_j$ and v_j for $k < j \leq q$ are both uniformly distributed over $\mathbb{K} \setminus \mathbb{F}_q$, we have $(v_j)_{j \in [q]} \equiv (\beta^{-1} \widetilde{v}_{\tau^{-1}(j)})_{j \in [q]}$. We also have $\mathbf{1}_q w_i = \mathbf{1}_q \widetilde{w}_i \beta^{-1}$ for all $i \in C$.
Since the $\boldsymbol{s}_i$'s are chosen independent of $\boldsymbol{T}$, we have the following:

$$\begin{aligned}
&(\boldsymbol{T}\boldsymbol{v}; (\boldsymbol{T}\mathbf{1}_q w_i)_{i \in C}, (\boldsymbol{s}_i)_{i \in C}; (\boldsymbol{T}\mathbf{1}_q w_i x_i + \boldsymbol{s}_i)_{i \in \overline{C}}) \\
&\equiv ((r_j v_{\pi(j)})_{j \in [q]}; ((r_j w_i)_{j \in [q]})_{i \in C}, (\widetilde{\boldsymbol{s}}_i)_{i \in C}; (\widetilde{\boldsymbol{s}}_i)_{i \in \overline{C}}) \\
&\equiv ((r_j \beta^{-1} \widetilde{v}_{(\tau^{-1} \circ \pi)(j)})_{j \in [q]}; ((r_j \beta^{-1} \widetilde{w}_i)_{j \in [q]})_{i \in C}, (\widetilde{\boldsymbol{s}}_i)_{i \in C}; (\widetilde{\boldsymbol{s}}_i)_{i \in \overline{C}}) \\
&\equiv ((\widetilde{r}_j \widetilde{v}_{\widetilde{\pi}(j)})_{j \in [q]}; ((\widetilde{r}_j \widetilde{w}_i)_{j \in [q]})_{i \in C}, (\widetilde{\boldsymbol{s}}_i)_{i \in C}; (\widetilde{\boldsymbol{s}}_i)_{i \in \overline{C}}) \\
&\equiv (\widetilde{\boldsymbol{T}}\widetilde{\boldsymbol{v}}; (\widetilde{\boldsymbol{T}}\mathbf{1}_q \widetilde{w}_i)_{i \in C}, (\widetilde{\boldsymbol{s}}_i)_{i \in C}; (\widetilde{\boldsymbol{s}}_i)_{i \in \overline{C}}),
\end{aligned}$$

where $\widetilde{r}_j$ $(j \in [q])$, $\widetilde{\pi}$, $\widetilde{\boldsymbol{T}} = \mathrm{diag}(\widetilde{r}_1, \ldots, \widetilde{r}_q)\boldsymbol{U}_{\widetilde{\pi}}$, and $\widetilde{\boldsymbol{s}}_i$ $(i \in [n])$ are elements sampled by the simulator.

Communication complexity. The maximum component of on-line and off-line communication is the randomness R_i for $i \in [n]$, which consists of two vectors over $\mathbb{K}$ of dimension q. Since $\mathbb{K}$ can chosen as any extension field of $\mathbb{F}_q$, the protocol achieves the communication complexity in the statement. □

As in the case of abelian programs, NIMPC protocols for $\mathcal{L}_{\mathbb{F}_q}$ is not directly applicable to $\mathcal{L}_{\mathbb{F}_q}^{S_1,\ldots,S_n}$. Nevertheless, from the same reason presented in Corollary 1, it is possible to obtain a protocol for $\mathcal{L}_{\mathbb{F}_q}^{S_1,\ldots,S_n}$ by applying the BKR transformation to the protocol for $\mathcal{L}_{\mathbb{F}_q}$.

Corollary 2. *Let $\mathbb{F}_q$ be a finite field and $S_1, \ldots, S_n$ be subsets of $\mathbb{F}_q$. Then, there exists a t-robust NIMPC protocol Π for $\mathcal{L}_{\mathbb{F}_q}^{S_1,\ldots,S_n}$ such that $\mathsf{CC}(\Pi) = p^{t+O(1)} q \log q$, where p is a prime power with $p \geq \max\{n, |S_1|, \ldots, |S_n|\}$.*

6 Revisiting the BKR Transformation

6.1 Analyzing and Fixing the NIMPC Protocol of [6] for Outputting-Message Functions

To realize the BKR transformation, the authors of [6] introduce the class of outputting-message functions. Let $\mathcal{M}$ be a finite set and let $\perp$ be the special

symbol not in $\mathcal{M}$. Let $\mathcal{X}_1 = \cdots = \mathcal{X}_n = \mathbb{F}_q$ and $\mathcal{X} = \mathcal{X}_1 \times \cdots \times \mathcal{X}_n$. Let $\boldsymbol{A} = [\boldsymbol{a}_1, \ldots, \boldsymbol{a}_n] \in \mathbb{F}_q^{k \times n}$ be a fixed matrix. Define the outputting-message function $h_{\boldsymbol{u},m} : \mathcal{X} \to \mathcal{M} \cup \{\perp\}$ associated with $\boldsymbol{u} \in \mathbb{F}_q^k$ and $m \in \mathcal{M}$ as

$$h_{\boldsymbol{u},m}(x_1, \ldots, x_n) = \begin{cases} m, & \text{if } \boldsymbol{u} = \boldsymbol{A}[x_1, \ldots, x_n]^\top, \\ \perp, & \text{otherwise.} \end{cases}$$

Define $\mathcal{O}_{\mathcal{M},\boldsymbol{A}}$ be the class of all the outputting-message functions, that is, $\mathcal{O}_{\mathcal{M},\boldsymbol{A}} = \{h_{\boldsymbol{u},m} \mid \boldsymbol{u} \in \mathbb{F}_q^k, m \in \mathcal{M}\}$.

The authors of [6] propose a fully robust NIMPC protocol for $\mathcal{O}_{\mathcal{M},\boldsymbol{A}}$ with communication complexity $O(k(\log q)(\log |\mathcal{M}|))$ and use it as a building block for the BKR transformation. However, we show that their protocol does not satisfy even 1-robustness if the matrix $\boldsymbol{A}$ satisfies a certain condition. We first recall their protocol for $\mathcal{O}_{\mathcal{M},\boldsymbol{A}}$. We denote by $\boldsymbol{a}_i \in \mathbb{F}_q^k$ the i-th column vector of $\boldsymbol{A} \in \mathbb{F}_q^{k \times n}$. We may assume $\mathcal{M} = \mathbb{F}_q$ since a protocol for an arbitrary message space $\mathcal{M}$ is obtained by expressing an element of $\mathcal{M}$ as a vector over $\mathbb{F}_q$ of dimension $\lceil \log_q |\mathcal{M}| \rceil$.

Protocol $\Pi_{\mathbf{BKR}}$ for $\mathcal{O}_{\mathcal{M},\boldsymbol{A}}$

$\mathsf{Gen}(h_{\boldsymbol{u},m})$**:**

1. Choose $\boldsymbol{s} \leftarrow\!\!\$\, \mathbb{F}_q^k$, $r_i \leftarrow\!\!\$\, \mathbb{F}_q$ $(i \in [n])$ and set $\mu_0 = m - \boldsymbol{s}^\top \boldsymbol{u} - \sum_{i \in [n]} r_i$.
2. Choose $\boldsymbol{s}_i \leftarrow\!\!\$\, \mathbb{F}_q^k$ $(i \in [n])$ and set $\boldsymbol{\nu}_0 = \boldsymbol{u} + \sum_{i \in [n]} \boldsymbol{s}_i$.
3. Output $R_0 = (\mu_0, \boldsymbol{\nu}_0)$ and $R_i = (\boldsymbol{s}^\top \boldsymbol{a}_i, r_i, \boldsymbol{s}_i)$ for $i \in [n]$.

$\mathsf{Enc}_i(x_i, R_i)$**:**

1. Parse R_i as $(\mu_{i1}, \mu_{i2}, \boldsymbol{\rho}_i)$, where $\mu_{i1}, \mu_{i2} \in \mathbb{F}_q$ and $\boldsymbol{\rho}_i \in \mathbb{F}_q^k$.
2. Output $M_i = (\mu_{i1} x_i + \mu_{i2}, \boldsymbol{a}_i x_i + \boldsymbol{\rho}_i)$.

$\mathsf{Dec}(R_0, M_1, \ldots, M_n)$**:**

1. Parse R_0 as $(\mu_0, \boldsymbol{\nu}_0)$ and M_i as $(\mu_i, \boldsymbol{\nu}_i)$, where $\mu_i \in \mathbb{F}_q$ and $\boldsymbol{\nu}_i \in \mathbb{F}_q^k$.
2. If $\boldsymbol{\nu}_0 = \sum_{i \in [n]} \boldsymbol{\nu}_i$, then output $\mu_0 + \sum_{i \in [n]} \mu_i$ and otherwise output $\perp$.

Fig. 5. The NIMPC protocol Π_{BKR} of [6] for the class of outputting-message functions $\mathcal{O}_{\mathcal{M},\boldsymbol{A}}$.

Proposition 2. *Let $C \subseteq [n]$. If there exists $j \in \overline{C}$ such that $\boldsymbol{a}_j \neq \boldsymbol{0}$ and $\{\boldsymbol{a}_i \mid i \in C \cup \{j\}\}$ does not span $\mathbb{F}_q^k$, then the protocol Π_{BKR} described in Fig. 5 is not C-robust.*

Proof. Let $\boldsymbol{u} \in \mathbb{F}_q^k$ be a vector that is not in the space spanned by $\{\boldsymbol{a}_i \mid i \in C \cup \{j\}\}$. Fix any $m \in \mathbb{F}_q$ and set $h = h_{\boldsymbol{u},m} \in \mathcal{O}_{\mathcal{M},\boldsymbol{A}}$. Let $\boldsymbol{x}_{\overline{C}}, \boldsymbol{y}_{\overline{C}}$ be two inputs such that $x_j \neq y_j$ and $x_i = y_i = 0$ for all $i \in \overline{C} \setminus \{j\}$. Then, it holds that $h|_{\overline{C},x_{\overline{C}}}(z_{\overline{C}}) = h|_{\overline{C},y_{\overline{C}}}(z_{\overline{C}}) = \perp$ for any $z_{\overline{C}} \in \mathcal{X}_C$. Assume that Π_{BKR} is

C-robust and that there is a simulator Sim satisfying Definition 2. Since Sim simulates the adversary's view with oracle access to the same residual function $h|_{\overline{C},x_{\overline{C}}} = h|_{\overline{C},y_{\overline{C}}}$, the views of the execution of Π_{BKR} on $x_{\overline{C}}$ and $y_{\overline{C}}$ should be identical to each other.

On the other hand, the adversary can compute $\boldsymbol{d} := \boldsymbol{\nu}_0 - \sum_{i\in\overline{C}} \boldsymbol{\nu}_i - \sum_{i\in C} \boldsymbol{s}_i$ from his view. If $x_{\overline{C}}$ is inputted to Π_{BKR}, then $\boldsymbol{d} = \boldsymbol{u} - \boldsymbol{a}_j x_j$. If $y_{\overline{C}}$ is inputted, then $\boldsymbol{d} = \boldsymbol{u} - \boldsymbol{a}_j y_j \neq \boldsymbol{u} - \boldsymbol{a}_j x_j$, from which it follows that the two views are different. This is a contradiction. □

We note that the condition of Proposition 2 holds even for the matrix $\boldsymbol{A}$ used in the BKR transformation. Indeed, to transform a 0-robust NIMPC protocol into a t-robust one, it is necessary to choose $k = t+1$ and $\boldsymbol{A} = [\boldsymbol{H}^\top, \boldsymbol{e}_i]^\top$, where $\boldsymbol{H} \in \mathbb{F}_q^{t\times n}$ is a matrix such that every t column vectors is linearly independent. For any set C of size at most $t-1$ and any $j \in \overline{C}$, the column vectors of $\boldsymbol{A}$ indexed by $C\cup\{j\}$ span a subspace of $\mathbb{F}_q^k$ of dimension t, which therefore implies that the condition of Proposition 2 holds.

Next, we fix their protocol Π_{BKR}. The main issue is that the adversary is able to compute $\boldsymbol{d} = \boldsymbol{\nu}_0 - \sum_{i\in\overline{C}} \boldsymbol{\nu}_i - \sum_{i\in C} \boldsymbol{s}_i = \boldsymbol{u} - \sum_{i\in\overline{C}} \boldsymbol{a}_i x_i$ while the only information on $\boldsymbol{d}$ revealed by the residual function is whether it is in the space spanned by $\{\boldsymbol{a}_i \mid i \in C\}$. We therefore randomize the vectors $\boldsymbol{d}$ and $\boldsymbol{a}_i$ $(i \in C)$ to ensure for the adversary not to learn more than their linear independence relation. Specifically, we choose a matrix $\boldsymbol{T}$ uniformly at random from $\mathrm{GL}_k(\mathbb{F}_q)$ and redefine $\boldsymbol{\nu}_0 = \boldsymbol{T}\boldsymbol{u} + \sum_{i\in[n]} \boldsymbol{s}_i$. We additionally give $\boldsymbol{T}\boldsymbol{a}_i$ to each P_i as randomness. Then, the adversary only learns $\boldsymbol{\nu}_0 - \sum_{i\in\overline{C}} \boldsymbol{\nu}_i - \sum_{i\in C} \boldsymbol{s}_i = \boldsymbol{T}(\boldsymbol{u} - \sum_{i\in\overline{C}} \boldsymbol{a}_i x_i)$, which leaks the linear independence relation among $\boldsymbol{d}$ and $\boldsymbol{a}_i$ $(i \in C)$ and nothing more. Note that our modification increases the communication complexity of Π_{BKR} only by a constant factor and hence the statements of [6] still hold true.

Theorem 3. *Let $\mathcal{M}$ be a finite set and $\boldsymbol{A} \in \mathbb{F}_q^{k\times n}$ be a matrix. Let $\mathcal{X}_1 = \cdots = \mathcal{X}_n = \mathbb{F}_q$ and $\mathcal{X} = \mathcal{X}_1 \times \cdots \times \mathcal{X}_n$. Then, the protocol Π^*_{BKR} described in Fig. 6 is a fully robust NIMPC protocol for $\mathcal{O}_{\mathcal{M},\boldsymbol{A}}$ such that*

$$\mathsf{CC}_{\mathsf{on}}(\Pi^*_{BKR}) = (k + \lceil \log_q |\mathcal{M}| \rceil) \cdot \lceil \log q \rceil$$
$$\text{and } \mathsf{CC}_{\mathsf{off}}(\Pi^*_{BKR}) = (2k + 2\lceil \log_q |\mathcal{M}| \rceil) \cdot \lceil \log q \rceil.$$

To begin with, we show a lemma used in the proof of Theorem 3.

Lemma 1. *Let $\boldsymbol{A} = (\boldsymbol{a}_i)_{i\in[\ell]}$ be a tuple of ℓ vectors, where $\boldsymbol{a}_i \in \mathbb{F}_q^k$. Let $\Gamma = Ker(\boldsymbol{A})$. Then, the distribution of $(\boldsymbol{T}\boldsymbol{a}_i)_{i\in[\ell]}$ induced by $\boldsymbol{T} \leftarrow\!\!\$\ GL_k(\mathbb{F}_q)$ is the uniform distribution over $\mathcal{F}_\Gamma = \{(\boldsymbol{v}_i)_{i\in[\ell]} \mid Ker((\boldsymbol{v}_i)_{i\in[\ell]}) = \Gamma\}$.*

Proof. Fix $\boldsymbol{W}_j = [\boldsymbol{w}_{j1}, \ldots, \boldsymbol{w}_{j\ell}] \in \mathcal{F}_\Gamma$ for $j = 1, 2$. Note that $\boldsymbol{T}\boldsymbol{A} \in \mathcal{F}_\Gamma$ for any $\boldsymbol{T} \in \mathrm{GL}_k(\mathbb{F}_q)$. It is sufficient to show that the probabilities of $\boldsymbol{T}\boldsymbol{A}$ being $\boldsymbol{W}_j$ are equal to each other.

Let $I_1 \subseteq [\ell]$ be such that $\{\boldsymbol{w}_{1i} \mid i \in I_1\}$ is a basis of $\boldsymbol{W}_1$, that is, $\{\boldsymbol{w}_{1i} \mid i \in I_1\}$ is linearly independent and for every $j \in [\ell] \setminus I_1$, there exist $c_{ij} \in \mathbb{F}_q$ such

Protocol $\Pi^*_{\mathbf{BKR}}$ for $\mathcal{O}_{\mathcal{M},\boldsymbol{A}}$

$\mathsf{Gen}(h_{\boldsymbol{u},m})$**:**

1. Choose $\boldsymbol{s} \leftarrow\!\!\$\, \mathbb{F}_q^k$, $r_i \leftarrow\!\!\$\, \mathbb{F}_q$ $(i \in [n])$ and set $\mu_0 = m - \boldsymbol{s}^\top \boldsymbol{u} - \sum_{i\in[n]} r_i$.
2. Choose $\boldsymbol{T} \leftarrow\!\!\$\, \mathrm{GL}_k(\mathbb{F}_q)$, $\boldsymbol{s}_i \leftarrow\!\!\$\, \mathbb{F}_q^k$ $(i \in [n])$ and set $\boldsymbol{\nu}_0 = \boldsymbol{T}\boldsymbol{u} + \sum_{i\in[n]} \boldsymbol{s}_i$.
3. Output $R_0 = (\mu_0, \boldsymbol{\nu}_0)$ and $R_i = (\boldsymbol{s}^\top \boldsymbol{a}_i, r_i, \boldsymbol{T}\boldsymbol{a}_i, \boldsymbol{s}_i)$ for $i \in [n]$.

$\mathsf{Enc}_i(x_i, R_i)$**:**

1. Parse R_i as $(\mu_{i1}, \mu_{i2}, \boldsymbol{\rho}_{i1}, \boldsymbol{\rho}_{i2})$, where $\mu_{i1}, \mu_{i2} \in \mathbb{F}_q$ and $\boldsymbol{\rho}_{i1}, \boldsymbol{\rho}_{i2} \in \mathbb{F}_q^k$.
2. Output $M_i = (\mu_{i1}x_i + \mu_{i2}, \boldsymbol{\rho}_{i1}x_i + \boldsymbol{\rho}_{i2})$.

$\mathsf{Dec}(R_0, M_1, \ldots, M_n)$**:**

1. Parse R_0 as $(\mu_0, \boldsymbol{\nu}_0)$ and M_i as $(\mu_i, \boldsymbol{\nu}_i)$, where $\mu_i \in \mathbb{F}_q$ and $\boldsymbol{\nu}_i \in \mathbb{F}_q^k$.
2. If $\boldsymbol{\nu}_0 = \sum_{i\in[n]} \boldsymbol{\nu}_i$, then output $\mu_0 + \sum_{i\in[n]} \mu_i$ and otherwise output $\perp$.

Fig. 6. The NIMPC protocol Π^*_{BKR} for the class of outputting-message functions $\mathcal{O}_{\mathcal{M},\boldsymbol{A}}$.

that $\boldsymbol{w}_{1j} = \sum_{i\in I_1} \boldsymbol{w}_{1i}c_{ij}$. Then, $\{\boldsymbol{w}_{2i} \mid i \in I_1\}$ is a basis of $\boldsymbol{W}_2$. Indeed, if $\sum_{i\in I_1} \boldsymbol{w}_{2i}d_i = \boldsymbol{0}$ for some $d_i \in \mathbb{F}_q$, then $(d_i)_{i\in I_1} \in \Gamma$. Hence $\sum_{i\in I_1} \boldsymbol{w}_{1i}d_i = \boldsymbol{0}$ and $d_i = 0$ for every $i \in I_1$. If $\{\boldsymbol{w}_{2i} \mid i \in I_2\}$ is linearly independent for some $I_2 \supseteq I_1$, then there is no non-zero $\boldsymbol{d} \in \Gamma$ such that $\mathrm{supp}(\boldsymbol{d}) \subseteq I_2$. Then, $\{\boldsymbol{w}_{1i} \mid i \in I_2\}$ is linearly independent and hence $I_2 = I_1$. Note that $\boldsymbol{w}_{2j} = \sum_{i\in I_1} \boldsymbol{w}_{2i}c_{ij}$.

Due to the linear independence, there exists $\boldsymbol{S} \in \mathrm{GL}_k(\mathbb{F}_q)$ such that $\boldsymbol{S}\boldsymbol{w}_{1i} = \boldsymbol{w}_{2i}$ for every $i \in I_1$. Then, $\boldsymbol{S}\boldsymbol{w}_{1j} = \sum_{i\in I_1} \boldsymbol{S}\boldsymbol{w}_{1i}c_{ij} = \sum_{i\in I_1} \boldsymbol{S}\boldsymbol{w}_{2i}c_{ij} = \boldsymbol{w}_{2j}$ for every $j \in [\ell] \setminus I_1$. Therefore,

$$\begin{aligned}\Pr[\boldsymbol{T} \leftarrow\!\!\$\, \mathrm{GL}_k(\mathbb{F}_q) : \boldsymbol{T}\boldsymbol{A} = \boldsymbol{W}_1] &= \Pr[\boldsymbol{T} \leftarrow\!\!\$\, \mathrm{GL}_k(\mathbb{F}_q) : \boldsymbol{S}\boldsymbol{T}\boldsymbol{A} = \boldsymbol{W}_2] \\ &= \Pr[\boldsymbol{T} \leftarrow\!\!\$\, \mathrm{GL}_k(\mathbb{F}_q) : \boldsymbol{T}\boldsymbol{A} = \boldsymbol{W}_2]\,.\end{aligned}$$

□

Proof (of Theorem 3). We assume that $\mathcal{M} = \mathbb{F}_q$. It is possible to construct a protocol for any message space $\mathcal{M}$ in the same manner as [6].

Correctness. Let $h = h_{\boldsymbol{u},m}$ be a function to compute. Correctness follows from the following observation: the protocol outputs a message other than $\perp$ if and only if $\boldsymbol{\nu}_0 - \sum_{i\in[n]} \boldsymbol{\nu}_i = \boldsymbol{T}(\boldsymbol{u} - \sum_{i\in[n]} \boldsymbol{a}_i x_i) = \boldsymbol{0}$, which in turn occurs if and only if $\boldsymbol{u} = \sum_{i\in[n]} \boldsymbol{a}_i x_i$ since $\boldsymbol{T} \in \mathrm{GL}_k(\mathbb{F}_q)$. Then, the output is $\mu_0 + \sum_{i\in[n]} \mu_i = m - \boldsymbol{s}^\top(\boldsymbol{u} - \sum_{i\in[n]} \boldsymbol{a}_i x_i) = m$.

Robustness. Let $C \subseteq [n]$ and $C_0 = C \cup \{0\}$. The adversary's view can be decomposed into the following two parts:

$$(\boldsymbol{T}\boldsymbol{u} + \sum_{i\in[n]} \boldsymbol{s}_i; (\boldsymbol{T}\boldsymbol{a}_i)_{i\in C}, (\boldsymbol{s}_i)_{i\in C}; (\boldsymbol{T}\boldsymbol{a}_i x_i + \boldsymbol{s}_i)_{i\in\overline{C}}),$$

Simulator $\mathsf{Sim}^{h|_{\overline{C},x_{\overline{C}}}}(C)$ for $\Pi^*_{\mathbf{BKR}}$

Input. A set of colluding players C.
Oracle access. The residual function $h|_{\overline{C},x_{\overline{C}}}$ for the inputs $x_{\overline{C}}$ of honest players.
Output. Correlated randomness R_0 and $(R_i)_{i\in C}$ of the colluding players and messages $(M_i)_{i\in\overline{C}}$ of the honest players.
Algorithm.

- If $C=\emptyset$:
  1. Set $\delta_0 = m$ and $\boldsymbol{\Delta}_0 = \mathbf{0}$ if $h(x_1,\ldots,x_n) = m \neq \perp$ and otherwise choose $\delta_0 \leftarrow\!\!\$\, \mathbb{F}_q$, $\boldsymbol{\Delta}_0 \leftarrow\!\!\$\, \mathbb{F}_q^k \setminus \{\mathbf{0}\}$.
  2. Choose $\widetilde{r}_i \leftarrow\!\!\$\, \mathbb{F}_q$ $(i\in[n])$, $\widetilde{\boldsymbol{s}}_i \leftarrow\!\!\$\, \mathbb{F}_q^k$ $(i\in[n])$.
  3. Output $R_0 = (\delta_0 - \sum_{i\in[n]}\widetilde{r}_i, \boldsymbol{\Delta}_0 + \sum_{i\in[n]}\widetilde{\boldsymbol{s}}_i)$ and $M_i = (\widetilde{r}_i, \widetilde{\boldsymbol{s}}_i)$ for $i\in[n]$.
- If $C\neq\emptyset$:
  1. Construct Γ_0 and Γ_0' as
  $$\Gamma_0 = \{\boldsymbol{x}_C \in \mathbb{F}_q^{|C|} \mid \sum_{i\in C}\boldsymbol{a}_i x_i = \mathbf{0}\} \text{ and } \Gamma_0' = \{(0,\boldsymbol{x}_C)\in\mathbb{F}_q^{|C|+1} \mid \boldsymbol{x}_C\in\Gamma_0\}.$$
  2. Construct Γ_1 and Γ_1' using the oracle $h|_{\overline{C},x_{\overline{C}}}$ as
  $$\Gamma_1 = \{\boldsymbol{x}_C \in \mathbb{F}_q^{|C|} \mid h|_{\overline{C},\boldsymbol{x}_{\overline{C}}}(\boldsymbol{x}_C) \neq \perp\}$$
  $$\text{and } \Gamma_1' = \{(x_0, \boldsymbol{x}_C x_0) \in \mathbb{F}_q^{|C|+1} \mid x_0\in\mathbb{F}_q\setminus\{0\},\ \boldsymbol{x}_C\in\Gamma_1\}.$$
  3. Set $\Gamma' = \Gamma_0'\cup\Gamma_1'$ and define
  $$\mathcal{F}_{\Gamma'} = \{(\boldsymbol{\Delta}_i)_{i\in C\cup\{0\}} \in (\mathbb{F}_q^k)^{|C|+1} \mid \mathrm{Ker}((\boldsymbol{\Delta}_i)_{i\in C\cup\{0\}}) = \Gamma'\}.$$
  4. Choose $(\delta_i)_{i\in C\cup\{0\}} \leftarrow\!\!\$\, (\Gamma')^{\perp}$, $(\boldsymbol{\Delta}_i)_{i\in C\cup\{0\}} \leftarrow\!\!\$\, \mathcal{F}_{\Gamma'}$, $\widetilde{r}_i \leftarrow\!\!\$\, \mathbb{F}_q$ $(i\in[n])$, and $\widetilde{\boldsymbol{s}}_i \leftarrow\!\!\$\, \mathbb{F}_q^k$ $(i\in[n])$.
  5. If $\Gamma_1 = \emptyset$, output
  $$R_0 = (\delta_0 - \sum_{i\in[n]}\widetilde{r}_i, \sum_{i\in[n]}\widetilde{\boldsymbol{s}}_i - \boldsymbol{\Delta}_0), R_i = (\delta_i, \widetilde{r}_i, \boldsymbol{\Delta}_i, \widetilde{\boldsymbol{s}}_i) \text{ for } i\in C,$$
  $$\text{and } M_i = (\widetilde{r}_i, \widetilde{\boldsymbol{s}}_i) \text{ for } i\in\overline{C}.$$
  Otherwise, fix $\boldsymbol{x}_C\in\Gamma_1$ and set $m = h|_{\overline{C},\boldsymbol{x}_{\overline{C}}}(\boldsymbol{x}_C)$. Then, output
  $$R_0 = (\delta_0 + m - \sum_{i\in[n]}\widetilde{r}_i, \sum_{i\in[n]}\widetilde{\boldsymbol{s}}_i - \boldsymbol{\Delta}_0), R_i = (\delta_i, \widetilde{r}_i, \boldsymbol{\Delta}_i, \widetilde{\boldsymbol{s}}_i) \text{ for } i\in C,$$
  $$\text{and } M_i = (\widetilde{r}_i, \widetilde{\boldsymbol{s}}_i) \text{ for } i\in\overline{C}.$$

Fig. 7. The simulator for the NIMPC protocol Π^*_{BKR} for the class of outputting-message functions $\mathcal{O}_{\mathcal{M},\boldsymbol{A}}$.

where $\boldsymbol{T} \leftarrow_{\$} \mathrm{GL}_k(\mathbb{F}_q)$ and $\boldsymbol{s}_i \leftarrow_{\$} \mathbb{F}_q^k$ $(i \in [n])$, and

$$(m - \boldsymbol{s}^\top \boldsymbol{u} - \sum_{i\in[n]} r_i; (\boldsymbol{s}^\top \boldsymbol{a}_i)_{i\in C}, (r_i)_{i\in C}; (\boldsymbol{s}^\top \boldsymbol{a}_i x_i + r_i)_{i\in\overline{C}}),$$

where $\boldsymbol{s} \leftarrow_{\$} \mathbb{F}_q^k$ and $r_i \leftarrow_{\$} \mathbb{F}_q$ $(i \in [n])$. We separately show that each of the two distributions is perfectly simulated by the simulator described in Fig. 7.

As for the first part, it is sufficient to show that the distribution of

$$(\boldsymbol{T}\boldsymbol{a}_0; (\boldsymbol{T}\boldsymbol{a}_i)_{i\in C}, (\boldsymbol{s}_i)_{i\in C}; (\boldsymbol{T}\boldsymbol{a}_i x_i + \boldsymbol{s}_i)_{i\in\overline{C}}) \tag{3}$$

is perfectly simulated, where $\boldsymbol{a}_0 = \sum_{i\in\overline{C}} \boldsymbol{a}_i x_i - \boldsymbol{u}$. This is because the original view can be obtained by computing $\boldsymbol{\nu}_0 = -\boldsymbol{T}\boldsymbol{a}_0 + \sum_{i\in C} \boldsymbol{s}_i + \sum_{i\in\overline{C}} (\boldsymbol{T}\boldsymbol{a}_i x_i + \boldsymbol{s}_i)$.

The case of $C = \emptyset$. We have the following:

- If $h(x_1, \ldots, x_n) = \perp$, then $\boldsymbol{a}_0 \neq \boldsymbol{0}$ and $\boldsymbol{T}\boldsymbol{a}_0$ is uniformly distributed over $\mathbb{F}_q^k \setminus \{\boldsymbol{0}\}$. Since the $\boldsymbol{s}_i$'s are chosen independent of $\boldsymbol{T}$, letting $\boldsymbol{\Delta}_0 \leftarrow_{\$} \mathbb{F}_q^k \setminus \{\boldsymbol{0}\}$ and $\widetilde{\boldsymbol{s}}_i \leftarrow_{\$} \mathbb{F}_q^k$ $(i \in [n])$, we have $(\boldsymbol{T}\boldsymbol{a}_0; (\boldsymbol{T}\boldsymbol{a}_i x_i + \boldsymbol{s}_i)_{i\in[n]}) \equiv (\boldsymbol{\Delta}_0; (\widetilde{\boldsymbol{s}}_i)_{i\in[n]})$.
- If $h(x_1, \ldots, x_n) = m \neq \perp$, then $\boldsymbol{a}_0 = \boldsymbol{0}$ and $\boldsymbol{T}\boldsymbol{a}_0 = \boldsymbol{0}$. Since the $\boldsymbol{s}_i$'s are chosen independent of $\boldsymbol{T}$, letting $\boldsymbol{\Delta}_0 = \boldsymbol{0}$ and $\widetilde{\boldsymbol{s}}_i \leftarrow_{\$} \mathbb{F}_q^k$ $(i \in [n])$, we have $(\boldsymbol{T}\boldsymbol{a}_0; (\boldsymbol{T}\boldsymbol{a}_i x_i + \boldsymbol{s}_i)_{i\in[n]}) \equiv (\boldsymbol{\Delta}_0; (\widetilde{\boldsymbol{s}}_i)_{i\in[n]})$.

The case of $C \neq \emptyset$. Let $\boldsymbol{B} = (\boldsymbol{a}_i)_{i\in C_0}$ and $\Gamma = \mathrm{Ker}(\boldsymbol{B})$. We show that Γ'_0 and Γ'_1 constructed by the simulator satisfy $\Gamma'_0 \cup \Gamma'_1 = \Gamma$. Indeed, let $\boldsymbol{x}_{C_0} = (x_0, \boldsymbol{x}_C) \in \Gamma$. If $x_0 = 0$, then $\sum_{i\in C} \boldsymbol{a}_i x_i = \boldsymbol{0}$ and hence $\boldsymbol{x}_{C_0} = (0, \boldsymbol{x}_C) \in \Gamma'_0$. If $x_0 \neq 0$, then $\sum_{i\in\overline{C}} \boldsymbol{a}_i x_i + \sum_{i\in C} \boldsymbol{a}_i (x_i x_0^{-1}) = \boldsymbol{u}$ and $h|_{\overline{C}, \boldsymbol{x}_{\overline{C}}}(\boldsymbol{x}_C x_0^{-1}) = m \neq \perp$. Hence, $\boldsymbol{x}_{C_0} = (x_0, (\boldsymbol{x}_C x_0^{-1}) x_0) \in \Gamma'_1$. Conversely, any $(0, \boldsymbol{x}_C) \in \Gamma'_0$ is clearly an element of Γ. Any element of Γ'_1 can be expressed as $(x_0, \boldsymbol{x}_C x_0)$ for some $x_0 \neq 0$ and $\boldsymbol{x}_C = (x_i)_{i\in C}$ with $h|_{\overline{C}, \boldsymbol{x}_{\overline{C}}}(\boldsymbol{x}_C) \neq \perp$, i.e., $\sum_{i\in\overline{C}} \boldsymbol{a}_i x_i + \sum_{i\in C} \boldsymbol{a}_i x_i = \boldsymbol{u}$. Therefore, it holds that $\boldsymbol{a}_0 x_0 + \sum_{i\in C} \boldsymbol{a}_i (x_i x_0) = \boldsymbol{0}$ and hence $(x_0, \boldsymbol{x}_C x_0) \in \Gamma$.

It follows from Lemma 1 that $(\boldsymbol{T}\boldsymbol{a}_i)_{i\in C_0}$ is uniformly distributed over $\mathcal{F}_\Gamma = \{(\boldsymbol{\Delta}_i)_{i\in C_0} \mid \mathrm{Ker}((\boldsymbol{\Delta}_i)_{i\in C_0}) = \Gamma\}$ when $\boldsymbol{T} \leftarrow_{\$} \mathrm{GL}_k(\mathbb{F}_q)$. Note that the $\boldsymbol{s}_i$'s are chosen independent of $\boldsymbol{T}$. Therefore, letting $(\boldsymbol{\Delta}_i)_{i\in C_0} \leftarrow_{\$} \mathcal{F}_{\Gamma'} = \mathcal{F}_\Gamma$ and $\widetilde{\boldsymbol{s}}_i \leftarrow_{\$} \mathbb{F}_q^k$ $(i \in [n])$, we have

$$(\boldsymbol{T}\boldsymbol{a}_0, (\boldsymbol{T}\boldsymbol{a}_i)_{i\in C}, (\boldsymbol{s}_i)_{i\in C}; (\boldsymbol{T}\boldsymbol{a}_i x_i + \boldsymbol{s}_i)_{i\in\overline{C}}) \equiv (\boldsymbol{\Delta}_0; (\boldsymbol{\Delta}_i)_{i\in C}, (\widetilde{\boldsymbol{s}}_i)_{i\in C}; (\widetilde{\boldsymbol{s}}_i)_{i\in\overline{C}}).$$

As for the second part, it is sufficient to show that the distribution of

$$(m + \boldsymbol{s}^\top \boldsymbol{a}_0; (\boldsymbol{s}^\top \boldsymbol{a}_i)_{i\in C}, (r_i)_{i\in C}; (\boldsymbol{s}^\top \boldsymbol{a}_i x_i + r_i)_{i\in\overline{C}}) \tag{4}$$

is perfectly simulated, where $\boldsymbol{a}_0 = \sum_{i\in\overline{C}} \boldsymbol{a}_i x_i - \boldsymbol{u}$. This is because the original view can be obtained by computing $\mu_0 = (m + \boldsymbol{s}^\top \boldsymbol{a}_0) - \sum_{i\in C} r_i - \sum_{i\in\overline{C}} (\boldsymbol{s}^\top \boldsymbol{a}_i x_i + r_i)$.

The case of $C = \emptyset$. We have the following:

- If $h(x_1, \ldots, x_n) = \perp$, then $\boldsymbol{a}_0 \neq \boldsymbol{0}$. It follows that $\boldsymbol{s}^\top \boldsymbol{a}_0$ and hence $m + \boldsymbol{s}\boldsymbol{a}_0$ are uniformly distributed over $\mathbb{F}_q$. Since the r_i's are chosen independent of $\boldsymbol{s}$, letting $\delta_0 \leftarrow_{\$} \mathbb{F}_q$ and $\widetilde{r}_i \leftarrow_{\$} \mathbb{F}_q$ $(i \in [n])$, we have $(m + \boldsymbol{s}^\top \boldsymbol{a}_0; (\boldsymbol{s}^\top \boldsymbol{a}_i x_i + r_i)_{i\in[n]}) \equiv (\delta_0; (\widetilde{r}_i)_{i\in[n]})$.

– If $h(x_1, \ldots, x_n) = m \neq \perp$, then $\boldsymbol{a}_0 = \boldsymbol{0}$ and $\boldsymbol{s}^\top \boldsymbol{a}_0 = 0$ for all $\boldsymbol{s} \in \mathbb{F}_q^k$. Since the r_i's are chosen independent of $\boldsymbol{s}$, letting $\delta_0 = m$ and $\widetilde{r}_i \leftarrow_\$ \mathbb{F}_q$ $(i \in [n])$, we have $(m + \boldsymbol{s}^\top \boldsymbol{a}_0; (\boldsymbol{s}^\top \boldsymbol{a}_i x_i + r_i)_{i\in[n]}) \equiv (\delta_0; (\widetilde{r}_i)_{i\in[n]})$.

The case of $C \neq \emptyset$. For $\boldsymbol{s} \leftarrow_\$ \mathbb{F}_q^k$, we have that $(m+\boldsymbol{s}^\top \boldsymbol{a}_0, (\boldsymbol{s}^\top \boldsymbol{a}_i)_{i\in C}) \equiv \boldsymbol{s}^\top \boldsymbol{B} + m\boldsymbol{e}_1$. Define $\mathrm{row}(\boldsymbol{B}) = \{\boldsymbol{s}^\top \boldsymbol{B} = (\boldsymbol{s}^\top \boldsymbol{a}_i)_{i\in C_0} \mid \boldsymbol{s} \in \mathbb{F}_q^k\}$. It then holds that $(\Gamma')^\perp = \Gamma^\perp = \mathrm{row}(\boldsymbol{B})$ since $\Gamma' = \Gamma = \mathrm{Ker}(\boldsymbol{B})$.

Now, we can see below that the distribution of (4) is perfectly simulated:

– If $h|_{\overline{C},\boldsymbol{x}_{\overline{C}}}(\boldsymbol{x}_C) = \perp$ for all $\boldsymbol{x}_C$, then $\boldsymbol{a}_0$ is not in the space spanned by $\{\boldsymbol{a}_i \mid i \in C\}$. In particular, there exists a vector $\boldsymbol{s}_0 \in \mathbb{F}_q^k$ such that $\boldsymbol{s}_0^\top \boldsymbol{B} = \boldsymbol{e}_1^\top = [1, 0, \ldots, 0]$. Therefore, $\boldsymbol{s}^\top \boldsymbol{B} + m\boldsymbol{e}_1^\top = (\boldsymbol{s} + \boldsymbol{s}_0 m)^\top \boldsymbol{B}$ is uniformly distributed over $\mathrm{row}(\boldsymbol{B})$ if $\boldsymbol{s} \leftarrow_\$ \mathbb{F}_q^k$. Since the r_i's are chosen independent of $\boldsymbol{s}$, the distribution of (4) is identical to $(\delta_0; (\delta_i)_{i\in C}, (\widetilde{r}_i)_{i\in C}; (\widetilde{r}_i)_{i\in\overline{C}})$, where $(\delta_i)_{i\in C_0} \leftarrow_\$ (\Gamma')^\perp$ and $\widetilde{r}_i \leftarrow_\$ \mathbb{F}_q$ $(i \in [n])$ are sampled by the simulator.
– If $h|_{\overline{C},\boldsymbol{x}_{\overline{C}}}(\boldsymbol{x}_C) \neq \perp$ for some $\boldsymbol{x}_C$, then the simulator actually gets the message $m \in \mathbb{F}_q$. We know that $(\boldsymbol{s}^\top \boldsymbol{a}_i)_{i\in C_0}$ is uniformly distributed over $\mathrm{row}(\boldsymbol{B})$ if $\boldsymbol{s} \leftarrow_\$ \mathbb{F}_q^k$. Therefore, since the r_i's are chosen independent of $\boldsymbol{s}$, the distribution of (4) is identical to $(\delta_0+m; (\delta_i)_{i\in C}, (\widetilde{r}_i)_{i\in C}; (\widetilde{r}_i)_{i\in\overline{C}})$, where $(\delta_i)_{i\in C_0} \leftarrow_\$ (\Gamma')^\perp$ and $\widetilde{r}_i \leftarrow_\$ \mathbb{F}_q$ $(i \in [n])$ are sampled by the simulator.

Communication complexity. The maximum component of on-line and off-line communication is the randomness R_i for $i \in [n]$, which consists of two scalars of $\mathbb{F}_q$ for outputting messages and two vectors over $\mathbb{F}_q$ of dimension k for checking the equality $\boldsymbol{u} = \boldsymbol{A}\boldsymbol{x}$. If elements of $\mathcal{M}$ are expressed as vectors over $\mathbb{F}_q$ of dimension ℓ, the communication complexity of the first two entries of R_i increases ℓ times. Therefore, the protocol achieves the communication complexity in the statement. □

6.2 An Asymptotically Optimal NIMPC Protocol for Indicator Functions

As a byproduct of our modified protocol Π^*_{BKR}, we obtain a fully robust NIMPC protocol for indicator functions with asymptotically optimal communication complexity with respect to the input length.

We recall the definition of indicator functions [3]. Let $\mathcal{X}_i$ $(i \in [n])$ be finite sets and $\mathcal{X} = \mathcal{X}_1 \times \cdots \times \mathcal{X}_n$. Define the indicator function $h_a : \mathcal{X} \to \{0, 1\}$ for $a \in \mathcal{X}$ as $h_a(x) = 1$ if and only if $x = a$ for $x \in \mathcal{X}$. In addition, we define $\mathrm{id}_0 : \mathcal{X} \to \{0, 1\}$ as $\mathrm{id}_0(x) = 0$ for any $x \in \mathcal{X}$. Define $\mathcal{I}_\mathcal{X}$ be the class of all the indicator functions together with id_0, that is, $\mathcal{I}_\mathcal{X} = \{h_a \mid a \in \mathcal{X}\} \cup \{\mathrm{id}_0\}$. Indicator functions are fundamental building blocks to realize NIMPC for an arbitrary class of functions. Actually, based on the fact that any function $h : \mathcal{X} \to \{0, 1\}$ is expressed as $h(x) = \sum_{a\in\mathcal{X}:h(a)=1} h_a(x)$, it is shown that an NIMPC protocol for any given class $\mathcal{H}$ can be obtained from any protocol for $\mathcal{I}_\mathcal{X}$ with a multiplicative overhead of $\max_{h\in\mathcal{H}} |h^{-1}(1)|$ [3]. The reason why we incorporate id_0 in $\mathcal{I}_\mathcal{X}$ is to prevent the equality $h = \sum_{a\in\mathcal{X}:h(a)=1} h_a$ from revealing $|h^{-1}(1)|$.

We show that the protocol Π^*_{BKR} can be used as a protocol for $\mathcal{I}_{\mathcal{X}}$ with a slight modification. Let q be a prime power such that $q > \max_{i\in[n]} |\mathcal{X}_i|$. We identify each $\mathcal{X}_i$ with a subset of $\mathbb{F}_q \setminus \{0\}$. Let $\boldsymbol{A} = [\boldsymbol{e}_1, \ldots, \boldsymbol{e}_n]$ be the identity matrix of size n. Note that computing h_a for $a = (a_i)_{i\in[n]} \in \mathcal{X}$ is reduced to testing the equality $\boldsymbol{u} = \boldsymbol{A}\boldsymbol{x}$, where $\boldsymbol{u} = [a_1, \ldots, a_n]^\top$. We also see that testing $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{0}$ is equivalent to computing id_0 since the inputs are represented as non-zero field elements. Therefore, we can realize NIMPC for $\mathcal{I}_{\mathcal{X}}$ by running the protocol Π^*_{BKR} for $\mathcal{O}_{\mathcal{M},\boldsymbol{A}}$ except that we skip the procedures to reveal messages.

Corollary 3. *Let $\mathcal{X}_1, \ldots, \mathcal{X}_n$ be finite sets and $\mathcal{X} = \mathcal{X}_1 \times \cdots \times \mathcal{X}_n$. Let q be a prime power such that $q > \max_{i\in[n]} |\mathcal{X}_i|$. Then, the protocol Π_3 described in Fig. 8 is a fully robust NIMPC protocol for $\mathcal{I}_{\mathcal{X}}$ such that $\mathsf{CC}_{\mathsf{on}}(\Pi) = n\lceil \log q \rceil$ and $\mathsf{CC}_{\mathsf{off}}(\Pi) = 2n\lceil \log q \rceil$.*

Protocol Π_3 for $\mathcal{I}_{\mathcal{X}}$

$\mathsf{Gen}(h)$**:**
- $h = \mathrm{id}_0$:
  1. Choose $\boldsymbol{T} \leftarrow\!\!\$\, \mathrm{GL}_n(\mathbb{F}_q), \boldsymbol{s}_i \leftarrow\!\!\$\, \mathbb{F}_q^n$ $(i \in [n])$ and set $\boldsymbol{\nu}_0 = \sum_{i\in[n]} \boldsymbol{s}_i$.
  2. Output $R_0 = \boldsymbol{\nu}_0$ and $R_i = (\boldsymbol{T}\boldsymbol{e}_i, \boldsymbol{s}_i)$ for $i \in [n]$.
- $h = h_a$ for $a = (a_1, \ldots, a_n)$:
  1. Set $\boldsymbol{u} = [a_1, \ldots, a_n]^\top \in \mathbb{F}_q^n$.
  2. Choose $\boldsymbol{T} \leftarrow\!\!\$\, \mathrm{GL}_n(\mathbb{F}_q), \boldsymbol{s}_i \leftarrow\!\!\$\, \mathbb{F}_q^n$ $(i \in [n])$ and set $\boldsymbol{\nu}_0 = \boldsymbol{T}\boldsymbol{u} + \sum_{i\in[n]} \boldsymbol{s}_i$.
  3. Output $R_0 = \boldsymbol{\nu}_0$ and $R_i = (\boldsymbol{T}\boldsymbol{e}_i, \boldsymbol{s}_i)$ for $i \in [n]$.

$\mathsf{Enc}_i(x_i, R_i)$**:**
1. Parse R_i as $(\boldsymbol{\rho}_{i1}, \boldsymbol{\rho}_{i2})$, where $\boldsymbol{\rho}_{i1}, \boldsymbol{\rho}_{i2} \in \mathbb{F}_q^n$.
2. Output $M_i = \boldsymbol{\rho}_{i1}x_i + \boldsymbol{\rho}_{i2}$.

$\mathsf{Dec}(R_0, M_1, \ldots, M_n)$**:**
1. If $R_0 = \sum_{i\in[n]} M_i$, then output 1 and otherwise output 0.

Fig. 8. The NIMPC protocol Π_3 for the class of indicator functions $\mathcal{I}_{\mathcal{X}}$.

According to the lower bound [17], any NIMPC protocol for $\mathcal{I}_{\mathcal{X}}$ has communication complexity at least $n^{-1}\sum_{i\in[n]} \log |\mathcal{X}_i|$. Therefore, if the input domains all have the same size, i.e., $|\mathcal{X}_1| = \cdots = |\mathcal{X}_n|$, our protocol is asymptotically optimal with respect to the input length.

Acknowledgements

This research was partially supported by JSPS KAKENHI Grant Numbers JP20J20797 and 19H01109 and JST CREST JPMJCR19F6 and JPMJCR14D6.

References

1. Agarwal, N., Anand, S., Prabhakaran, M.: Uncovering algebraic structures in the MPC landscape. In: Advances in Cryptology – EUROCRYPT 2019, Part II. pp. 381–406 (2019)
2. 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)
3. Beimel, A., Gabizon, A., Ishai, Y., Kushilevitz, E., Meldgaard, S., Paskin-Cherniavsky, A.: Non-interactive secure multiparty computation. Cryptology ePrint Archive, Report 2014/960 (2014), full version of [2]
4. Beimel, A., Ishai, Y., Kushilevitz, E.: Ad hoc PSM protocols: Secure computation without coordination. In: Advances in Cryptology – EUROCRYPT 2017, Part III. pp. 580–608 (2017)
5. Beimel, A., Kushilevitz, E., Nissim, P.: The complexity of multiparty PSM protocols and related models. In: Advances in Cryptology – EUROCRYPT 2018, Part II. pp. 287–318 (2018)
6. Benhamouda, F., Krawczyk, H., Rabin, T.: Robust non-interactive multiparty computation against constant-size collusion. In: Advances in Cryptology – CRYPTO 2017, Part I. pp. 391–419 (2017)
7. Boser, B.E., Guyon, I.M., Vapnik, V.N.: A training algorithm for optimal margin classifiers. In: Proceedings of the Fifth Annual Workshop on Computational Learning Theory. pp. 144–152. COLT '92 (1992)
8. Cortes, C., Vapnik, V.: Support-vector networks. Machine learning **20**(3), 273–297 (1995)
9. 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)
10. 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)
11. Halevi, S., Ishai, Y., Kushilevitz, E., Rabin, T.: Best possible information-theoretic MPC. In: Theory of Cryptography, Part II. pp. 255–281 (2018)
12. 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)
13. 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)
14. Kilian, J.: Founding crytpography on oblivious transfer. In: Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing. pp. 20–31. STOC '88 (1988)
15. Obana, S., Yoshida, M.: An efficient construction of non-interactive secure multiparty computation. In: Cryptology and Network Security. pp. 604–614 (2016)
16. Shoup, V.: A computational introduction to number theory and algebra. Cambridge university press (2009)
17. Yoshida, M., Obana, S.: On the (in)efficiency of non-interactive secure multiparty computation. Designs, Codes and Cryptography **86**(8), 1793–1805 (2018)