Robust Non-Interactive Multiparty Computation Against Constant-Size Collusion

Fabrice Benhamouda, Hugo Krawczyk, and Tal Rabin

IBM Research, Yorktown Heights, US

Abstract. Non-Interactive Multiparty Computations (Beimel et al., Crypto 2014) is a very powerful notion equivalent (under some corruption model) to garbled circuits, Private Simultaneous Messages protocols, and obfuscation. We present robust solutions to the problem of Non-Interactive Multiparty Computation in the computational and information-theoretic models. Our results include the first efficient and robust protocols to compute any function in NC^1 for constant-size collusions, in the information-theoretic setting and in the computational setting, to compute any function in P for constant-size collusions, assuming the existence of one-way functions. Our constructions start from a Private Simultaneous Messages construction (Feige, Killian Naor, STOC 1994 and Ishai, Kushilevitz, ISTCS 1997) and transform it into a Non-Interactive Multiparty Computation for constant-size collusions.
We also present a new Non-Interactive Multiparty Computation protocol for symmetric functions with significantly better communication complexity compared to the only known one of Beimel et al.

Keywords. Non-interactive multiparty computation, private simultaneous messages

1 Introduction

A non-interactive multiparty computation enables n parties $P_1, \ldots, P_n$, each holding a private input, and a special party P_0, called an evaluator, to compute a joint function of the n parties' inputs so that the evaluator learns the output. The communication structure in this setting is that each party sends a single message to the evaluator. This is a highly desired mode of interaction as the required connectivity between the parties is extremely simple, yet it enables to carry out natural computations such as voting and auctions.

Feige, Kilian, and Naor [4] were first to study such a model, referred to as the *Private Simultaneous Messages (PSM)*[1] model. They considered information-theoretic security, namely, in a PSM protocol for a function f, the evaluator of the function learns the output of the function on the parties' inputs and *nothing else*. Essential to their solutions was the assumption that the evaluator does not

[1] Name given by Ishai and Kushilevitz [9].

collude with any of the n parties. If such collusions were possible, even with a single misbehaving party, their protocols would lose the privacy guarantee.

Beimel, Gabizon, Ishai, Kushilevitz, Meldgaard, and Paskin [3] generalized the PSM model to what they call *Non-Interactive Multiparty Computation (NIMPC)*, by considering the possibility of collusions between parties and the evaluator. In this setting the notion of security needs to be modified as clearly we cannot prevent the evaluator from computing the function on all possible inputs of the colluding parties. Thus, they define the notion of "best possible security" by utilizing the residual function [8] for a set of colluding parties T. The residual function of f is all the values $f(y_1, \ldots, y_n)$ where $y_i = x_i$ if $P_i \notin T$ (x_i being the input of the non-colluding party P_i) and $y_i \in \{0, 1\}$ for $P_i \in T$. A secure protocol would enable the adversary to learn the residual function and nothing more. An NIMPC protocol that can withstand collusions of up to t parties is called *t-robust*. If $t = n$ the protocol is said to be *fully robust*.

Due to their very restricted communication pattern, both PSM and NIMPC require some form of setup arrangement. PSM assumes a common random string shared by the parties while NIMPC allows for a setup phase where parties are provided with *correlated randomness*. The latter models an offline stage run independently of the parties' inputs with the actual computation of the function happening in a later online phase.

We note that while the above notions were introduced in the information-theoretic setting, they apply to the computational case as well. The notion of NIMPC turns out to be extremely powerful both in the computational and information-theoretic setting, and for a wide range of applications. It generalizes such notions as obfuscation and garbling schemes, and is a weaker variant of multi-input functional encryption. At the same time, in more practical settings, NIMPC can be used for voting, auctions, or distributed computations on bulletin boards.

The wide applicability of the NIMPC abstraction is also reflected in the wide range of results (and open questions) for what is computable in this model. In the information-theoretic setting, Feige et al. [4] show that *any* function can be computed with *exponential size* messages sent from parties to evaluator. At the same time, they show that any function in NC^1 can be computed with polynomial-size messages. Ishai and Kushilevitz [9] further expanded the class of functions that can be computed by PSM protocols to log-space language classes such as $\mathrm{mod}_p L$ and to log-space counting classes such as $\#L$.

Not surprisingly, the NIMPC model proves to be more challenging, even for restricted robustness. Beimel et al. [3] prove that some non-trivial functions can be computed with information-theoretic security in this model. They showed that the iterated product function $f(x_1, \ldots, x_n) = x_1 \cdots x_n$ over a group $\mathbb{G}$ can be computed efficiently with a collusion of any size. In addition, for any function f, they exhibit a solution that can tolerate arbitrary collusions but is exponential in the total bit-length of the inputs. Their strongest result shows that symmetric

functions over a domain $\mathcal{X}_1 \times \cdots \times \mathcal{X}_n$ where each $\mathcal{X}_i$ is of constant-size admits a t-robust NIMPC with polynomial complexity for constant t.

Can these information-theoretic NIMPC results be extended to a larger class, e.g., NC^1, as in the PSM case? A negative result follows from Goldwasser and Rothblum [6] implicitly stating that the existence of an efficient protocol for NC^1 that can tolerate a polynomial-size collusion (i.e., of size $t = \Omega(n^\alpha)$, with $\alpha > 0$ being constant) in the information-theoretic setting would imply the collapse of the polynomial-time hierarchy. This still leaves the possibility that robust NIMPC with restricted, say constant-size, collusions are still possible for NC^1. Yet, Beimel et al. show evidence that even achieving 1-robustness, i.e., security against a collusion of one party with the evaluator, may require a new technical approach (they show that natural approaches to realize NIMPC based on known PSM or garbling techniques fail even for $t = 1$). They leave this question open.

In the computational setting the situation is strikingly different. First of all, in the PSM model or the equivalent 0-robust NIMPC, one can compute any polynomial-time computable function with polynomial-size messages under the sole assumption of the existence of one-way functions. Indeed, note that a Yao garbled circuit is a 0-robust NIMPC. At the other end, fully-robust NIMPC for any polynomial function can be constructed using multi-input functional encryption which Goldwasser et al. [5] build on the basis of indistinguishability obfuscation (iO) and one-way functions. Actually, the existence of efficient NIMPC protocols for P that can tolerate a polynomial-size collusion implies iO.

The above results leave two wide gaps in our knowledge regarding the feasibility of constructing robust NIMPC protocols. In the information-theoretic setting, PSM exists for at least all of NC^1 while NIMPC with non-zero robustness is only known for a handful of simple functions [3]. In the computational setting, one-way functions suffice for 0-robust NIMPC for all polynomial functions, and under iO fully-robust NIMPC for all P is possible.

This raises two important questions:

1. Do information-theoretic robust NIMPC protocols exist, even for 1-robustness, for a class of functions covered by PSM, e.g., NC^1?
2. Do computational robust NIMPC protocols exist for P, with restricted robustness, under weaker assumptions than iO?

These are open questions postulated in the work of Beimel et al. [3] and the ones that we set to answer.

1.1 Our results

From PSM to NIMPC. Our main theorem shows an information-theoretic transformation which takes any PSM (or 0-robust NIMPC) construction and transforms it into a t-robust NIMPC protocol. The resultant protocol has complexity that is, roughly, $n^{O(t)}$ times that of the given PSM protocol. Furthermore,

if the original PSM relied on some assumptions the new protocol relies on the same assumptions without needing to introduce any further assumptions.

This single theorem is extremely powerful and its corollaries give an affirmative answer to the two questions raised above.

In the information-theoretic setting we have that for constant t, there exist efficient t-robust protocols for the same class of functions for which efficient PSM protocols exist, in particular for the whole class NC^1 and the classes shown in [9], namely, $\mathrm{mod}_p L$ and log-space counting classes such as $\#L$.

In the computational setting, we achieve robust NIMPC solutions for constant-size collusions for any polynomial-time function, solely based on one way functions. That is, we narrow the gap between the PSM solutions based on one-way functions that tolerate no collusions, and the solutions based on iO, that can tolerate any number of collusions. Recall that robust NIMPC solutions for any polynomial-time function, even for polynomial-size collusions, implies iO.

Design. The idea governing our result was to directly find a solution to the problem identified by Beimel et al. [3, Section 6]. The essence of the problem can be understood by considering a Yao garbled circuit. The circuit is set up so that each input wire i has two possible labels $m'_{i,0}, m'_{i,1}$ one of which will be used by party P_i to convey its input to the evaluator. The problem arises when P_i colludes with the evaluator providing both labels for input wire i. One might hope that this would only enable the evaluator to compute the residual function, i.e., $f(x_1, \ldots, x_{i-1}, 0, x_{i+1}, \ldots, x_n)$ and $f(x_1, \ldots, x_{i-1}, 1, x_{i+1}, \ldots, x_n)$, which is allowed. However, the above paper shows that in fact more is exposed via the knowledge of both labels, thus violating the security of the computation. This problem also arises in similar constructions based on Barrington theorem [2] and Kilian randomization [12], or the Ishai-Kushilevitz protocol in [9].

The issue is that the adversary can learn two different labels $m'_{i,0}$ and $m'_{i,1}$ for the same input wire i, when P_i is colluding. If we could prevent it, this would resolve the problem described above. Yet, this seems challenging as we need to enable P_i to still have a message for a possible input of 0 and a message for a possible input of 1, otherwise it will render the computation impossible. But maybe this counter-intuitive approach can be achieved?

Given a function f, n parties, $P_1, \ldots, P_n$, holding inputs $x_1, \ldots, x_n$ (resp.), an evaluator P_0, and a PSM which computes the function we will do the following. We duplicate the PSM a number of times (this number is a function of the number of colluding parties; we denote it for now by κ), creating the copies $\mathcal{PSM}_1, \ldots, \mathcal{PSM}_\kappa$. Each PSM will have a fresh set of labels for its input wires. Concretely, $\mathcal{PSM}_{\boldsymbol{\sigma}}$ will have labels $m'_{\boldsymbol{\sigma},i,0}, m'_{\boldsymbol{\sigma},i,1}$ for $i = \{1, \ldots, n\}$. On top of these copies of the PSM we will put NIMPC protocols which we call *selectors*. There will be n selectors $\mathcal{S}el_1, \ldots, \mathcal{S}el_n$, one for each party. The input wires for all the selectors will be labeled by $m_{i,0}, m_{i,1}$ for $i = \{1, \ldots, n\}$. The selector $\mathcal{S}el_i$ is expected to output a label $m'_{\boldsymbol{\sigma},i,x_i}$ for exactly one index $\boldsymbol{\sigma}$. Each selector will have one output wire for a total of n output wires for all the selectors combined.

Clearly, the adversary can still utilize both $m_{i,0}, m_{i,1}$ of a colluding party on the input wires to the selectors. But the selectors will be sophisticated. Given a specific set of labels for the inputs wires, they will provide a full set of labels for only one of the copies of the PSM. Given a different set of input wire labels, they will provide a full set of labels for a different PSM. So for the example above, on input the set of labels $m_{1,x_1}, \ldots, m_{i-1,x_{i-1}}, m_{i,0}, m_{i+1,x_{i+1}}, \ldots, m_{n,x_n}$, the adversary will receive the labels for $\mathcal{PSM}_{\boldsymbol{\sigma}}$ and on input the set of labels $m_{1,x_1}, \ldots, m_{i-1,x_{i-1}}, m_{i,1}, , m_{i+1,x_{i+1}}, \ldots, m_{n,x_n}$ it will receive the labels for $\mathcal{PSM}_{\boldsymbol{\sigma}'}$. Thus, effectively disarming the adversary from the ability to learn two different labels for the same input of the *same* PSM. Note, that the adversary can still run the selectors multiple times on different inputs, in fact, on 2^t if there are t colluding parties. But the selectors can "tolerate" such behavior without violating the privacy of the inputs of the non-colluding parties.

Thus we have achieved that the combination of selectors $\mathcal{S}el_i$ and κ copies of the original PSM yield a t-robust NIMPC for the function computed by the PSM. See Fig. 1.

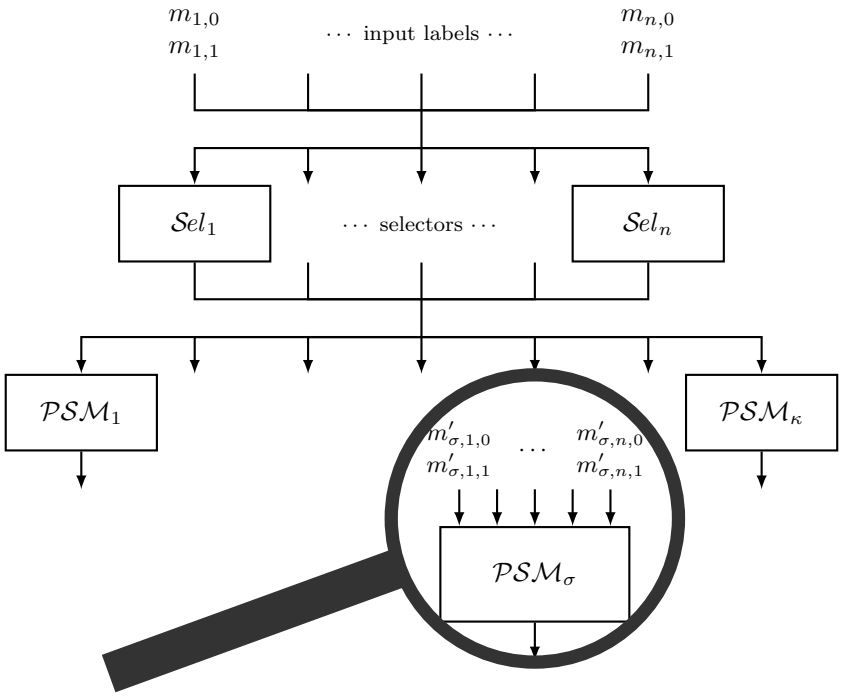
Example for one colluding party. In the following, we give a flavor of the ideas of our protocols in the specific case where only one party is colluding with the evaluator. In this case we would need two copies of the PSM, $\mathcal{PSM}_1$ and $\mathcal{PSM}_2$ with labels $m'_{1,i,0}, m'_{1,i,1}$ and $m'_{2,i,0}, m'_{2,i,1}$ (resp.) for the input wire corresponding to party P_i. Thus, we want the selectors either to provide a full set of labels for one or the other of the two PSMs.

We define $\mathcal{S}el_i$, the algorithm used by the evaluator to derive exactly one of the labels $m'_{1,i,x_i}, m'_{2,i,x_i}$ from the input labels $m_{1,x_1}, \ldots, m_{n,x_n}$, as:

$$\mathcal{S}el_i(m_{1,x_1}, \ldots, m_{n,x_n}) = \begin{cases} m'_{1,i,0} & \text{if } \sum_{j=1}^{n} x_j = 0 \bmod 2 \text{ and } x_i = 0\,, \\ m'_{1,i,1} & \text{if } \sum_{j=1}^{n} x_j = 0 \bmod 2 \text{ and } x_i = 1\,, \\ m'_{2,i,0} & \text{if } \sum_{j=1}^{n} x_j = 1 \bmod 2 \text{ and } x_i = 0\,, \\ m'_{2,i,1} & \text{if } \sum_{j=1}^{n} x_j = 1 \bmod 2 \text{ and } x_i = 1\ . \end{cases}$$

We assume that x_i can be implicitly obtained from m_{i,x_i}. Things will be made more formal later.

Let us examine the output of the selector and see that it works properly. W.l.o.g., assume that P_i is the colluding party and that the evaluator first uses the message corresponding to an input of 0 for P_i and that this sets the sum of all the parties' inputs to 0. In this case the selector for all parties, colluding or not, will output one of the values from the two top rows depending on the party's individual input. These are all labels for $\mathcal{PSM}_1$. Now, if the evaluator flips the input value of the colluding party to be 1, this causes the sum of all the parties' inputs to flip to 1, resulting in the selector outputting a value from the



Given input labels $m_{1,x_1}, \ldots, m_{n,x_n}$, the selectors choose an index σ and output the labels $m'_{\sigma,1,x_1}, \ldots, m'_{\sigma,n,x_n}$ for the instance $\mathcal{PSM}_\sigma$. For different input labels (that a collusion of t users can obtain), a different index σ' is chosen.

Fig. 1: NIMPC transformation

bottom two rows of the function. Those outputs are all labels of $\mathcal{PSM}_2$. Thus, we manage to prevent the evaluator from learning two labels for the same input of the same PSM.

NIMPC for symmetric functions. While our above result and also [3] provide NIMPC protocols for symmetric functions (both of complexity $n^{O(t)}$), here we present a specialized solution that improves significantly the level of robustness it can offer.

A symmetric Boolean function can be seen as a function of the sum of its inputs over $\mathbb{Z}_{n+1}$. Our core idea is to start with an inefficient NIMPC solution based on an information-theoretic implementation of Yao's garbling and then improve its complexity via a "divide-and-conquer" approach that uses the Chinese remainder theorem to create small instances of the problem. The NIMPC protocols on these smaller instances provide much stronger collusion resistance. Using this technique we show that there exists an information-theoretic t-robust NIMPC for symmetric Boolean functions with communication complexity $n^{\log\log n + \log t + O(1)}$,

improving significantly on the best prior protocol in [3, Theorem 4.17] that has communication complexity $\binom{n}{t} \cdot O(2^t \cdot n^4)$.

1.2 Related Work

Halevi, Lindell, and Pinkas [8] wanted to avoid the need for a fresh common or correlated randomness string for each execution. However, their model requires the parties to sequentially interact with the evaluator. They provide solutions for very specific patterns of interaction, assuming a public-key infrastructure (PKI).

Halevi et al. [7] expand the graphs of interaction patterns that can be handled in [8] to directed graphs, chains, and star. They examine which functions can be computed under these communication patterns and show that any interaction pattern can be reduced via an information theoretic protocol to a star, while providing the best possible security that can be achieved. Note that a star communication pattern is equivalent to the pattern presented in NIMPC. Using our new t-robust NIMPC protocols for the star communication pattern can enable a constant number of colluding parties for general communication patterns without relying on strong assumptions such as iO.

In [16] the authors provide an exponential lower bound of the communication complexity of NIMPC protocols for arbitrary functions, and improve the polynomial factors of the communication complexity of the NIMPC protocol for arbitrary functions of Beimel et al. They further extend their result in [14] improved complexity of the previous NIMPC protocol for arbitrary function with multi-bit inputs, yet it still has exponential complexity.

1.3 Organization of the Paper

In Section 2, we start by an extensive overview to provide intuition for the techniques we use in our transformation of PSM into t-robust NIMPC. After some formal preliminaries in Section 3, we present one of the main components of the transformation and of our NIMPC protocol for symmetric functions, namely selectors, in Section 4. In Section 5, we define and construct another component of the transformation, namely admissible linear indexing function. The transformation itself is formally described and proven in Section 6. Finally, in Section 7, we show our new NIMPC protocol for symmetric functions.

2 Overview

In this section we provide an extensive overview of the techniques we use in our transformation from PSM to t-robust NIMPC with emphasis on ideas and intuition at the expense of formalism. For the sake of simplicity, we assume that the inputs of the parties are bits.

2.1 Defining the Indexing Function

In the Introduction we showed that we need selectors that, for every given set of inputs of the parties, choose a different PSM, and then output a consistent set of labels for the input wires of the chosen PSM. In this section, we explain how to choose the PSM. This is done via what we call an indexing function ind.

The indexing function ind takes as input a vector $\boldsymbol{x}$, which reflects the inputs of the parties. The entry for a non-colluding P_i will be set to its actual input, x_i and the rest of the entries are fixed by the adversary. When the adversary controls t parties it can create 2^t distinct vectors $\boldsymbol{x}$, running over all possible inputs of the colluding parties. These 2^t vectors can in fact be reflected in 2^t evaluations of the selectors. Thus, we want to have (at least) 2^t PSM and in return require that the indexing function, ind, will map each one of the possible vectors into a different PSM. We will index the PSM by $\boldsymbol{\sigma}$ in some set $\mathcal{S}$.

We now build the indexing function ind. Let ind be a function that on input $\boldsymbol{x} = (x_1, \ldots, x_n)$ outputs an index $\boldsymbol{\sigma} \in \mathcal{S}$. The function ind should have the property that if a party P_i is colluding then any input $\boldsymbol{x}$ to ind that has $x_i = 0$ should produce a different $\boldsymbol{\sigma}$ than the same $\boldsymbol{x}$ but with $x_i = 1$. In general, no coalition of t colluding parties should be able to choose their inputs so that two different inputs lead to the selection of the same index $\boldsymbol{\sigma}$. Note that this does not mean that ind should be injective but rather that if one fixes the inputs of the non-colluding parties, then any two assignments of the remaining t inputs should result in a different value $\boldsymbol{\sigma}$ output by ind. Going back to the example from the Introduction, for $t = 1$ we implicitly defined $\mathsf{ind}(\boldsymbol{x}) = x_1 + \cdots + x_n$ and obtained the desired property. Indeed, if all the inputs are fixed except the one of a single colluding party P_i, each input x_i of P_i yields a different value $\mathsf{ind}(\boldsymbol{x})$ (note that this property assumes a single colluding party but does not require to know who this party is).

For the general case of t colluding parties we build ind using a linear code. We first observe that for any value $\boldsymbol{\sigma}$ in the range of ind, it should be that the set $\mathsf{ind}^{-1}(\boldsymbol{\sigma})$ forms a code of distance at least $t + 1$. Indeed, assume that two different elements $\boldsymbol{x_1}, \boldsymbol{x_2}$ in the set $\mathsf{ind}^{-1}(\boldsymbol{\sigma})$ have Hamming distance $\leq t$ and let $T \subseteq \{1, \ldots, n\}$ be the set of entries where the two differ. Choosing T as the set of colluding parties, we have that $\boldsymbol{x_1}, \boldsymbol{x_2}$ coincide in all the honest parties' inputs, differ on the colluding parties' inputs, yet they are mapped to the same value $\boldsymbol{\sigma}$. This contradicts our requirement from ind. We can thus define ind via a linear code of distance $t + 1$ over a linear space $\mathbb{F}_q^n$ (for some prime power q) as follows. Let $H \in \mathbb{F}_q^{\ell \times n}$ be the parity-check matrix of such a code, namely, the code is formed by all vectors $\boldsymbol{x} \in \mathbb{F}_q^n$ for which $H \cdot \boldsymbol{x} = 0$. This means that $H^{-1}(0)$ is a code of distance $t + 1$ and the same is also true, by linearity, for $H^{-1}(\boldsymbol{\sigma})$ for any other $\boldsymbol{\sigma}$ in the range of H. Thus, defining $\mathsf{ind}(\boldsymbol{x}) = H \cdot \boldsymbol{x}$ we get the property we needed. See Section 5 for the details.

We note that using such an H, we get that the set of possible values $\boldsymbol{\sigma}$ (i.e., the range of the function ind) is of size q^ℓ ($\ell = t$ in our implementation, for well-chosen prime powers q) and that is also the number of PSMs. This is the

source of exponential complexity in our construction and the reason for why we are polynomial-time only for constant t.

2.2 Reduction of $\mathcal{S}el_i$ to Message-Outputting Protocols

Given an indexing function $\mathsf{ind}\colon \boldsymbol{x} \mapsto H \cdot \boldsymbol{x} \in \mathbb{F}_q^\ell$ as above, our goal is now to construct the selector $\mathcal{S}el_i$ which is an NIMPC protocol for the following functions:

$$h_i\colon \boldsymbol{x} \in \{0,1\}^n \mapsto m'_{\mathsf{ind}(\boldsymbol{x}),i,x_i} \ . \tag{1}$$

where messages $m'_{\boldsymbol{\sigma},i,b}$ are implicit secret parameters of the NIMPC. The message $m'_{\boldsymbol{\sigma},i,b}$ is the message that party P_i would send on input b in the PSM $\mathcal{PSM}_{\boldsymbol{\sigma}}$. We recall that the selector $\mathcal{S}el_i$ ensures that an adversary should not be able to obtain two messages $m'_{\boldsymbol{\sigma},i,b}$ and $m'_{\boldsymbol{\sigma},i,1-b}$ for the same $\boldsymbol{\sigma}$ and i.

We can reduce the construction of such selectors $\mathcal{S}el_i$ to the construction of an NIMPC for the following functions:

$$h_{\boldsymbol{\sigma},i,b}\colon \boldsymbol{x} \in \mathbb{F}_q^n \mapsto \begin{cases} m'_{\boldsymbol{\sigma},i,b} & \text{if } \mathsf{ind}(\boldsymbol{x}) = \boldsymbol{\sigma} \text{ and } x_i = b, \\ \perp & \text{otherwise.} \end{cases} \tag{2}$$

The idea consists in running all the NIMPC protocols for all the functions $h_{\boldsymbol{\sigma},i,b}$, for each $\boldsymbol{\sigma} \in \mathcal{S}$ and each $b \in \{0,1\}$, in parallel, to get the selector $\mathcal{S}el_i$. Exactly one of them will have a non-$\perp$ output. To avoid leaking the value $b = x_i$ and $\boldsymbol{\sigma} = \mathsf{ind}(\boldsymbol{x})$, these protocols are randomly permuted.

As the condition "$\mathsf{ind}(\boldsymbol{x}) = \boldsymbol{\sigma}$ and $x_i = b$" in Eq. (2) is linear, we can rewrite these functions in terms of matrix-vector operations, a representation that will facilitate the design of NIMPC protocols for such functions. We first define the following generic family of functions indexed by a public matrix M in $\mathbb{F}_q^{k\times n}$ (for our constructions, $k = \ell + 1$), a secret vector $\boldsymbol{u}$ in $\mathbb{F}_q^k$, and a secret message $\tilde{m}$ in $\mathbb{F}_q$:

$$h_{M,\boldsymbol{u},\tilde{m}}\colon \boldsymbol{x} \in \mathbb{F}_q^n \mapsto \begin{cases} \tilde{m} \in \mathbb{F}_q & \text{if } \boldsymbol{u} = M \cdot \boldsymbol{x}, \\ \perp & \text{otherwise.} \end{cases}$$

An NIMPC for such a function is called a *message-outputting protocol*.

Now, assuming w.l.o.g. that $m'_{\boldsymbol{\sigma},i,b} \in \mathbb{F}_q$,[2] we can represent the above functions $h_{\boldsymbol{\sigma},i,b}$ as special cases of $h_{M,\boldsymbol{u},\tilde{m}}$ by setting

$$M = M^{(i)} = \begin{pmatrix} H \\ \boldsymbol{e}_i^\intercal \end{pmatrix}, \qquad \boldsymbol{u} = \boldsymbol{u}^{(\boldsymbol{\sigma},b)} = \begin{pmatrix} \boldsymbol{\sigma} \\ b \end{pmatrix}, \qquad \tilde{m} = m'_{\boldsymbol{\sigma},i,b}\,,$$

where $\boldsymbol{\sigma} \in \mathbb{F}_q^\ell$, and $\boldsymbol{e}_i$ is the i-th vector of the canonical basis of $\mathbb{F}_q^n$.

To sum up, we have reduced the task of designing the selectors $\mathcal{S}el_i$ to the task of designing outputting-message protocols, i.e., NIMPC for $h_{M,\boldsymbol{u},\tilde{m}}$. At this point, we can completely ignore the process that lead us to considering these functions $h_{M,\boldsymbol{u},\tilde{m}}$.

[2] We can always represent the message $m'_{\boldsymbol{\sigma},i,b}$ as a tuple of elements in $\mathbb{F}_q$, and use an independent message-outputting protocol for each of these elements.

2.3 Robust Message-Outputting Protocols

Let us now design robust message-outputting protocols, i.e., NIMPC for $h_{M,\boldsymbol{u},\tilde{m}}$. We note that while in our application M is public and $\boldsymbol{u}$ and $\tilde{m}$ are to remain hidden, here our presentation treats $\boldsymbol{u}$ as public. A full description of the NIMPC protocol for $h_{M,\boldsymbol{u},\tilde{m}}$, including addressing the secrecy of $\boldsymbol{u}$, is presented in Section 4.2.

Linear secret sharing scheme. We start by recalling the following linear secret sharing scheme LSSS [11], a variant of which is at the core of our construction.

The scheme is specified for n parties and an access structure defined on the basis of a matrix $M \in \mathbb{F}_q^{k \times n}$ and vector $\boldsymbol{v}$ in $\mathbb{F}_q^k$. Parties P_i, for i in some set $I \subseteq \{1, \ldots, n\}$, can reconstruct the secret message $\tilde{m}$ if and only if:

$$\boldsymbol{v} \in \mathrm{Span}((M_{\cdot,i})_{i \in I}) \,, \tag{3}$$

where $M_{\cdot,i}$ denotes the i-th column of M, in which case, we say that the set I is *authorized*.

The scheme provides each party P_i with a share s'_i defined as:

$$s'_i = \boldsymbol{s}^\intercal \cdot M_{\cdot,i} \,,$$

where $\boldsymbol{s}$ is a randomly chosen vector in F_q^k. In addition, the scheme publishes (or gives to each party)

$$s'_0 = \tilde{m} - \boldsymbol{s}^\intercal \cdot \boldsymbol{v} \,,$$

where $\tilde{m}$ is the secret being shared.

An authorized set I can recover the secret $\tilde{m}$ as follows. Since I is authorized, there exist scalars $\lambda_i \in \mathbb{F}_q$ for $i \in I$ so that $\sum_{i \in I} \lambda_i \cdot M_{\cdot,i} = \boldsymbol{v}$. Thus, parties P_i for $i \in I$ recover the secret $\tilde{m}$ as:

$$\tilde{m} = s'_0 + \sum_{i \in I} \lambda_i \cdot s'_i \ .$$

Conversely, if I is not an authorized set, the values s'_i only define the linear form $\boldsymbol{v} \in \mathbb{F}_q^k \mapsto \boldsymbol{s}^\intercal \cdot \boldsymbol{v}$ for vectors $\boldsymbol{v}$ in the span of the columns $M_{\cdot,i}$, for $i \in I$. As $\boldsymbol{v}$ is not in this span, the value $\boldsymbol{s}^\intercal \cdot \boldsymbol{v}$ is uniformly random from the point of view of the parties, and $\tilde{m}$ is completely masked.

NIMPC for when $h_{M,\boldsymbol{u},\tilde{\boldsymbol{m}}}(\boldsymbol{x}) = \tilde{\boldsymbol{m}}$. Back to our NIMPC construction for the family $h_{M,\boldsymbol{u},\tilde{m}}$, we want that the adversary can reconstruct $\tilde{m}$ if and only if it has access to a vector $\boldsymbol{x} \in \mathbb{F}_q^n$ such that

$$\boldsymbol{u} = M \cdot \boldsymbol{x} \ .$$

More precisely, let $T \subseteq \{1, \ldots, n\}$ be the set of colluding parties. For any vector $\boldsymbol{x} \in \mathbb{F}_q^n$, let $\boldsymbol{x}_T$ and $\boldsymbol{x}_{\bar{T}}$ be the two vectors in $\mathbb{F}_q^n$ defined as:

$$x_{T,i} = \begin{cases} x_i & \text{if } i \in T\,, \\ 0 & \text{otherwise,} \end{cases} \quad \text{and} \quad x_{\bar{T},i} = \begin{cases} 0 & \text{if } i \in T\,, \\ x_i & \text{otherwise.} \end{cases}$$

In other words, $\boldsymbol{x}_T$ corresponds to the inputs that the adversary can control[3] while $\boldsymbol{x}_{\bar{T}}$ corresponds to the inputs fixed by the honest parties. Each vector $\boldsymbol{x}_T$ is related to one value of the residual function that the adversary is allowed to compute. We have:

$$\boldsymbol{x} = \boldsymbol{x}_T + \boldsymbol{x}_{\bar{T}} \ .$$

With this terminology, we have that the adversary should be able to reconstruct $\tilde{m}$ if and only if there exists a vector $\boldsymbol{x}_T$ such that

$$\boldsymbol{u} = M \cdot \boldsymbol{x} = M \cdot (\boldsymbol{x}_T + \boldsymbol{x}_{\bar{T}})\,,$$

or, equivalently,

$$\boldsymbol{u} - M \cdot \boldsymbol{x}_{\bar{T}} \in \mathrm{Span}((M_{\cdot,i})_{i \in T}) \ .$$

This corresponds exactly to the definition of the access structure for the above LSSS scheme where $\boldsymbol{v} = \boldsymbol{u} - M \cdot \boldsymbol{x}_{\bar{T}}$. We adapt this scheme to our NIMPC setting as follows (see Fig. 2 in Section 4.2 for the details).

Recall that an NIMPC protocol starts with a setup phase (a.k.a. offline preprocessing) in order to generate the correlated randomness. It is indeed impossible to achieve any reasonable security notion without correlated randomness in this non-interactive setting.

In this setup phase, we first generate a uniform vector $\boldsymbol{s} \in \mathbb{F}_q^k$ and give to each party P_i the share of the above secret sharing scheme, namely, $s'_i = \boldsymbol{s}^\intercal \cdot M_{\cdot,i}$, as part of its correlated randomness. Assuming we know $\boldsymbol{x}_{\bar{T}}$, we could define the following value

$$s'_0 = \tilde{m} - \boldsymbol{s}^\intercal \cdot \boldsymbol{u} + \boldsymbol{s}^\intercal \cdot M \cdot \boldsymbol{x}_{\bar{T}}$$

that would correspond to the value s'_0 in the secret sharing scheme when $\boldsymbol{v} = \boldsymbol{u} - M \cdot \boldsymbol{x}_{\bar{T}}$. Yet, this value (as well as $\boldsymbol{v}$) depends on the set T and $\boldsymbol{x}_{\bar{T}}$ that is unknown at the time of sharing. Thus, the correlated randomness (and thus the messages sent by the parties) needs to contain additional information to allow authorized reconstruction of s'_0 and, as a result, $\tilde{m}$.

To achieve this, in the setup phase, we also generate independent uniform scalars $r_1, \ldots, r_n \in \mathbb{F}_q$, compute $r_0 = \sum_{i=1}^n r_i$, publish (in lieu of s'_0):

$$\mu_0 = \tilde{m} - \boldsymbol{s}^\intercal \cdot \boldsymbol{u} - r_0\,,$$

[3] Note that while the honest parties' inputs are from $\{0, 1\}$, we cannot control the inputs the adversary uses. The adversary can choose inputs from $\mathbb{F}_q$.

and give to each party P_i the scalar r_i as part of its correlated randomness. Finally, party P_i on input x_i outputs the message:

$$\mu_{i,x_i} = r_i + s'_i \cdot x_i \ .$$

With these values, message $\tilde{m}$ can be reconstructed in case that $M \cdot \boldsymbol{x} = \boldsymbol{u}$ through the following computation:

$$\tilde{m} = \mu_0 + \sum_{i=1}^{n} \mu_{i,x_i} \ .$$

(this equality is obtained by developing the right-hand term as $\mu_0 + r_0 + \sum s'_i x_i = \tilde{m} - \boldsymbol{s}^\intercal \cdot \boldsymbol{u} + \sum s_i M_{\cdot,i} x_i = \tilde{m} - \boldsymbol{s}^\intercal \cdot \boldsymbol{u} + \sum s_i u_i = \tilde{m}$).

The above shows correctness of the NIMPC scheme for the family $h_{M,\boldsymbol{u},\tilde{m}}$. We now argue robustness, namely, only $\tilde{m}$ is disclosed and only in case that $M \cdot \boldsymbol{x} = \boldsymbol{u}$. All other information remains (information-theoretically) hidden.

Note that when the set of colluding parties is T, the adversary's view (in collusion with P_0) consists of:

$$\begin{array}{lll} \mu_0 & = \tilde{m} - \boldsymbol{s}^\intercal \cdot \boldsymbol{u} - r_0 & \\ \mu_{i,x_i} & = r_i + \boldsymbol{s}^\intercal \cdot M_{\cdot,i} \cdot x_i & \text{for } i \in \bar{T} = \{1,\dots,n\} \setminus T \\ r_i & & \text{for } i \in T \\ s'_i & = \boldsymbol{s}^\intercal \cdot M_{\cdot,i} & \text{for } i \in T. \end{array}$$

The proof of robustness follows by showing that all these values can be simulated given only

$$s'_0 = \tilde{m} - \boldsymbol{s}^\intercal \cdot \boldsymbol{u} + \boldsymbol{s}^\intercal \cdot M \cdot \boldsymbol{x}_{\bar{T}} \quad \text{and} \quad s'_i = \boldsymbol{s}^\intercal \cdot M_{\cdot,i} \text{ for } i \in T \ , \tag{4}$$

which correspond to the shares of parties P_i, for $i \in T$, for the access structure defined by Eq. (3) of the LSSS scheme when $\boldsymbol{v} = \boldsymbol{u} - M \cdot \boldsymbol{x}_{\bar{T}}$. This shows that the above view of the adversary contains no more information than the LSSS shares hence implying the secrecy of $\tilde{m}$ in case that the equality $\boldsymbol{u} = M \cdot \boldsymbol{x}$ does not hold.

Detecting when $h_{M,\boldsymbol{u},\tilde{\boldsymbol{m}}}(\boldsymbol{x}) = \perp$. The above NIMPC protocol is almost a protocol for $h_{M,\boldsymbol{u},\tilde{m}}$, except that it always outputs something even when it should output $\perp$. To finish the construction, we need to add a way to detect whether $h_{M,\boldsymbol{u},\tilde{m}}(\boldsymbol{x}) = \perp$ or not, i.e., whether $M \cdot \boldsymbol{x} = \boldsymbol{u}$ or not.

This is simple to achieve: in the setup phase, we just pick uniform independent vectors $\boldsymbol{r}'_1, \dots, \boldsymbol{r}'_n \in \mathbb{F}_q^n$, compute $\boldsymbol{r}'_0 = \sum_{i=1}^n \boldsymbol{r}'_i$, publish $\boldsymbol{\nu_0} = \boldsymbol{u} + \boldsymbol{r}'_0$, and give to each party P_i the vector $\boldsymbol{r}'_i$ as part of its correlated randomness.

Then, on input x_i, party P_i outputs (in addition to μ_{i,x_i}):

$$\boldsymbol{\nu}_{i,x_i} = \boldsymbol{r}'_i + M_{\cdot,i} \cdot x_i \ .$$

In other words, on input x_i, P_i outputs the message $m_{i,x_i} = (\mu_{i,x_i}, \boldsymbol{\nu}_{i,x_i})$.

To check whether $M \cdot \boldsymbol{x} = \boldsymbol{u}$, it is then sufficient to check whether:

$$\boldsymbol{\nu_0} = \sum_{i=1}^{n} \boldsymbol{\nu}_{i,x_i} \ .$$

Robustness and correctness are straightforward.

2.4 Putting it all Together

We now summarize the steps in our transformation from a PSM for a function f to a t-robust NIMPC for the same function f. Full and formal details are presented in Sections 4.2, 5, and 6 (see Fig. 4).

First, we choose a linear code of length n and of distance at least $t+1$, for a well-chosen prime power q. Let $H \in \mathbb{F}_q^{\ell \times n}$ be its parity-check matrix (we can choose q as the smallest prime power greater or equal to n, and $\ell = t$). We define the indexing function as $\mathsf{ind}\colon \boldsymbol{x} \mapsto H \cdot \boldsymbol{x} \in \mathbb{F}_q^{\ell}$.

Second, in the setup phase, we consider q^ℓ copies of the PSM, indexed by elements $\boldsymbol{\sigma}$ of $\mathbb{F}_q^{\ell}$. We generate the correlated randomness of all these PSMs, and denote by $m'_{\boldsymbol{\sigma},i,b}$ the message that party P_i would send on input b in the PSM $\mathcal{PSM}_{\boldsymbol{\sigma}}$ of index $\boldsymbol{\sigma}$, for $\boldsymbol{\sigma} \in \mathbb{F}_q^{\ell}$, $i \in \{1, \ldots, n\}$, and $b \in \{0,1\}$.

Third, we construct the n NIMPC protocols, $\mathcal{S}el_1, \ldots, \mathcal{S}el_n$ (the linear selectors), for the functions $h_1, \ldots, h_n$ (resp.) defined in Eq. (1):

$$h_i\colon \boldsymbol{x} \in \{0,1\}^n \mapsto m'_{\mathsf{ind}(\boldsymbol{x}),i,x_i} \ .$$

As explained in Section 2.2, these selectors are constructed as parallel composition of outputting-message protocols, described in Section 2.3.

The correlated randomness of the resulting t-robust NIMPC protocol just consists in the concatenation of the correlated randomness of $\mathcal{S}el_1, \ldots, \mathcal{S}el_n$. The message that party P_i sends on input x_i is the concatenation of the ones it would send in $\mathcal{S}el_1, \ldots, \mathcal{S}el_n$ on input x_i. To compute the output, the evaluator first simulates the evaluators of $\mathcal{S}el_1, \ldots, \mathcal{S}el_n$ to get $m'_{\boldsymbol{\sigma},i,x_i}$ for all $i \in \{1, \ldots, n\}$ and for $\boldsymbol{\sigma} = \mathsf{ind}(\boldsymbol{x})$. It then simulates the evaluator of the original PSM on these messages to get the output $f(x_1, \ldots, x_n)$.

3 Preliminaries

3.1 NIMPC Definition

We recall the definition of NIMPC protocols from [3]. We first introduce the following notation.

Let $\mathcal{X}_1, \ldots, \mathcal{X}_n$ be non-empty sets and let $\mathcal{X}$ denote their Cartesian product, namely, $\mathcal{X} := \mathcal{X}_1 \times \cdots \times \mathcal{X}_n$. We use vector notation (boldface font) to denote the elements in $\mathcal{X}$, e.g., $\boldsymbol{x} \in \mathcal{X}$ (even though $\mathcal{X}$ is not necessarily a vector space). For a subset $T = \{i_1, \ldots, i_t\} \subseteq \{1, \ldots, n\}$ and $\boldsymbol{x} = (x_1, \ldots, x_n) \in \mathcal{X}$ we denote by $\boldsymbol{x}_T$ the t-coordinate projection vector $(x_{i_1}, \ldots, x_{i_t})$. For a function $f \colon \mathcal{X} \to \Omega$, we denote by $f|_{\bar{T}, \boldsymbol{x}_{\bar{T}}}$ the function f with the inputs corresponding to positions $\bar{T}$ fixed to the entries of vector $\boldsymbol{x}_{\bar{T}}$.[4]

Definition 3.1 (NIMPC Protocol). *Let $\mathcal{F} = (\mathcal{F}_n)_{n \in \mathbb{N}_{>0}}$ be an ensemble of sets $\mathcal{F}_n$ of functions $f \colon \mathcal{X} \to \Omega$, where Ω is a finite set and $\mathcal{X}$ is the Cartesian product of non-empty finite sets $\mathcal{X}_1, \ldots, \mathcal{X}_n$. A* non-interactive secure multiparty computation (NIMPC) protocol *for $\mathcal{F}$ is a tuple of three algorithms $\Pi = (\mathsf{Setup}, \mathsf{Msg}, \mathsf{Rec})$, where:*

- Setup *takes as input unary representations of n and of the security parameter $\mathfrak{K}$, and (a representation of) a function $f \in \mathcal{F}_n$ and outputs a tuple $(\rho_0, \rho_1, \ldots, \rho_n)$;*[5]
- Msg *takes as input a value ρ_i and an input $x_i \in \mathcal{X}_i$, and deterministically outputs a message m_{i,x_i};*
- Rec *takes as input a value ρ_0 and a tuple of n messages $(m_{i,x_i})_{i=1,\ldots,n}$ and outputs an element of Ω.*

satisfying the following property:

- ***Correctness.*** *For any values $n \in \mathbb{N}_{>0}$, security parameter $\mathfrak{K} \in \mathbb{N}$, $f \in \mathcal{F}_n$, $\boldsymbol{x} \in \mathcal{X}$ and $(\rho_0, \ldots, \rho_n) \xleftarrow{R} \mathsf{Setup}(f)$:*

$$\mathsf{Rec}(\rho_0, \mathsf{Msg}(\rho_1, x_1), \ldots, \mathsf{Msg}(\rho_n, x_n)) = f(\boldsymbol{x}) \ .$$

While the previous definition is abstract, in the sequel, we will often view NIMPC protocols as protocols with n parties $P_1, \ldots, P_n$ with respective inputs $x_1, \ldots, x_n$, and an evaluator P_0. This is actually the view adopted in the Introduction and in Section 2. More precisely, an NIMPC $\Pi = (\mathsf{Setup}, \mathsf{Msg}, \mathsf{Rec})$ yields a protocol in three phases as follows:

Offline preprocessing. For the security parameter $\mathfrak{K}$ and the function $f \in \mathcal{F}_n$, a trusted party generates $(\rho_0, \rho_1, \ldots, \rho_n) \xleftarrow{R} \mathsf{Setup}(1^n, 1^{\mathfrak{K}}, f)$ and gives ρ_i to party P_i (for $i \in \{1, \ldots, n\}$) and ρ_0 to the evaluator P_0.

Online messages. On input x_i, party P_i computes $m_{i,x_i} := \mathsf{Msg}(\rho_i, x_i)$ and outputs m_{i,x_i} to the evaluator P_0.

Reconstruction. After receiving m_{i,x_i} from all the parties P_i (for $i \in \{1, \ldots, n\}$), the evaluator P_0 computes and outputs $\mathsf{Rec}(\rho_0, m_{1,x_1}, \ldots, m_{n,x_n})$.

[4] In Section 3.2 we slightly change notation for vectors $\boldsymbol{x}_T$.

[5] One refers to the vector $(\rho_0, \rho_1, \ldots, \rho_n)$ as the *correlated randomness* of the parties, with ρ_0 called *public randomness*.

A *polynomial-time NIMPC protocol* for $\mathcal{F}$ is an NIMPC protocol $(\mathsf{Setup}, \mathsf{Msg}, \mathsf{Rec})$ where Setup, Msg, and Rec run in polynomial time in n and $\mathfrak{K}$. In particular, functions $f \in \mathcal{F}$ should be representable by polynomial-size bit strings.

The *online communication complexity* of Π, $\mathsf{CC}_{\mathsf{on}}(\Pi)$, is defined as the maximum of the size of the messages m_{i,x_i}. The *offline communication complexity* of Π, $\mathsf{CC}_{\mathsf{off}}(\Pi)$, is defined as the maximum of the size of the correlated randomness ρ_i. The *communication complexity* $\mathsf{CC}(\Pi)$ is defined as the maximum of the online communication complexity and of the offline communication complexity.

Robustness. We now recall the notions of robustness for NIMPC protocols. Informally, T-robustness for a set $T \subseteq \{1, \ldots, n\}$ of colluding parties means that if $\boldsymbol{x}_{\bar{T}}$ represents the inputs of the honest parties, then an evaluator colluding with the parties in set T can compute the residual function $f|_{\bar{T},\boldsymbol{x}_{\bar{T}}}$ on any input $\boldsymbol{x}_T$ but cannot learn anything else about the input of the honest parties. This describes the best privacy guarantee attainable in this adversarial setting. The formal definition is stated in terms of a simulator that can generate the view of the adversary (evaluator plus the colluding parties in set T) with sole oracle access to the residual function $f|_{\bar{T},\boldsymbol{x}_{\bar{T}}}$.

All our constructions and transformations are *unconditional*. But when combined with statistically or computationally robust 0-NIMPC protocols, the resulting protocols are only statistically or computationally robust. Therefore, we also need to define statistical and computational variants of robustness.

Definition 3.2 (NIMPC Robustness). *Let $n \in \mathbb{N}_{>0}$ be a positive integer and $T \subseteq \{1, \ldots, n\}$ be a subset. An NIMPC protocol Π is perfectly (resp., statistically, computationally)* T-robust *if there exists a randomized algorithm Sim (called a* simulator*) such that for any $f \in \mathcal{F}_n$ and $\boldsymbol{x}_{\bar{T}} \in \mathcal{X}_{\bar{T}}$, the following distributions are perfectly (resp., statistically, computationally) indistinguishable:*

$$\{\mathsf{Sim}^{f|_{\bar{T},\boldsymbol{x}_{\bar{T}}}}(1^n, 1^{\mathfrak{K}}, T)\} \quad \textit{and} \quad \{\mathsf{View}(1^n, 1^{\mathfrak{K}}, f, T, \boldsymbol{x}_{\bar{T}})\},$$

where $\mathsf{View}(1^n, 1^{\mathfrak{K}}, f, T, \boldsymbol{x}_{\bar{T}})$ is the view of the evaluator P_0 and of the colluding parties P_i (for $i \in T$) from running Π on inputs $\boldsymbol{x}_{\bar{T}}$ for the honest parties P_i (for $i \in \bar{T}$): namely, $((m_{i,x_{\bar{T},i}})_{i\in\bar{T}}, \rho_0, (\rho_i)_{i\in T})$ where $(\rho_0, \ldots, \rho_n) \xleftarrow{R} \mathsf{Setup}(1^n, 1^{\mathfrak{K}}, f)$ and $m_{i,x_{\bar{T},i}} \leftarrow \mathsf{Msg}(\rho_i, x_{\bar{T},i})$ for $i \in \bar{T}$.

Let t be an integer which is a function of n, then an NIMPC protocol Π is perfectly (resp., statistically, computationally) t-robust *if for any $n \in \mathbb{N}_{>0}$ and any subset $T \subseteq \{1, \ldots, n\}$ of size at most $t = t(n)$, Π is perfectly (resp., statistically, computationally) t-robust. It is perfectly (resp., statistically, computationally)* fully robust*, if it is perfectly (resp., statistically, computationally) n-robust.*

Computational robustness is defined non-uniformly to simplify the definition. However, it is also possible to define a uniform version with an explicit distinguisher which first chooses n, f, T, and $\boldsymbol{x}_{\bar{T}}$.

Robustness does not necessarily imply that the simulator Sim is the same for any n and T nor that it runs in polynomial time in n and $\mathfrak{K}$. Our constructions are efficient in the sense that the simulators are polynomial-time (in the communication complexity of the underlying protocols), and our transformations preserve the efficiency of the simulator.

Simplifications. In the sequel, we simplify notations as follows. The security parameter $\mathfrak{K}$ is dropped for all perfectly robust protocols. Furthermore, we suppose all the functions $f \in \mathcal{F}_n$ have the same domain $\mathcal{X}$ and the same number of parties n. The set $\mathcal{F}_n$ is simply denoted $\mathcal{F}$. We will sometimes refer to NIMPC for single functions f, to mean NIMPC for $\mathcal{F} = \{f\}$.

3.2 Group Embedding

While the definition of NIMPC is stated for arbitrary sets $\mathcal{X}_i$, for our treatment it is convenient (but not mandatory) to associate to these sets an addition operation and a neutral element 0. For this, we use the convention that each input set $\mathcal{X}_i$ is embedded (via an arbitrary injective mapping) into a group of cardinality $\geq |\mathcal{X}_i|$ (same group for all $i \in \{1, \ldots, n\}$). Thus, hereafter, we treat the sets $\mathcal{X}_i$ as subsets of a group where these subsets always include the neutral element 0; in our applications the group is typically a field $\mathbb{F}_q$ or a ring.

With this convention we re-define vectors of the form $\boldsymbol{x}_T$ as follows:

$$\mathcal{X}_T \coloneqq \{\boldsymbol{x} \in \mathcal{X} \mid \forall i \in \bar{T},\, x_i = 0\}\,, \qquad \mathcal{X}_{\bar{T}} \coloneqq \{\boldsymbol{x} \in \mathcal{X} \mid \forall i \in T,\, x_i = 0\}\ .$$

Let $\boldsymbol{x} \in \mathcal{X}$ be a vector. We define the vectors $\boldsymbol{x}_T \in \mathcal{X}_T$ and $\boldsymbol{x}_{\bar{T}} \in \mathcal{X}_{\bar{T}}$ to be the only two such vectors so that $\boldsymbol{x} = \boldsymbol{x}_T + \boldsymbol{x}_{\bar{T}}$. In other words, for all $i \in \{1, \ldots, n\}$:

$$x_{T,i} = \begin{cases} x_i & \text{if } i \in T\,, \\ 0 & \text{otherwise,} \end{cases} \quad \text{and} \quad x_{\bar{T},i} = \begin{cases} 0 & \text{if } i \in T\,, \\ x_i & \text{otherwise.} \end{cases}$$

Let $\boldsymbol{x}_{\bar{T}} \in \mathcal{X}_{\bar{T}}$ be a vector. With this notation we re-define the restriction of a function $f\colon \mathcal{X} \to \Omega$ to $\bar{T}, \boldsymbol{x}_{\bar{T}}$, which we denote by $f|_{\bar{T},\boldsymbol{x}_{\bar{T}}}$, as follows:

$$f|_{\bar{T},\boldsymbol{x}_{\bar{T}}}\colon \boldsymbol{x}_T \in \mathcal{X}_T \mapsto f(\boldsymbol{x}_T + \boldsymbol{x}_{\bar{T}}) \in \Omega\ .$$

That is, $f|_{\bar{T},\boldsymbol{x}_{\bar{T}}}$ is the function f for which the inputs x_i are fixed for $i \in \bar{T}$ to $x_{\bar{T},i}$.

Finally, we define the Hamming weight of an element $\boldsymbol{x} \in \mathcal{X}$ as the number of coordinates i for which $x_i \neq 0$, and define Hamming distance between elements $\boldsymbol{x_1}$ and $\boldsymbol{x_2}$ in $\mathcal{X}$ as the Hamming weight of $\boldsymbol{x_1} - \boldsymbol{x_2}$.

4 Selectors

In this section, we introduce the notion of selectors, which are used both in our transformation from PSM to $O(1)$-robust NIMPC and in our construction of NIMPC for symmetric functions. Intuitively, a selector is an NIMPC which selects a given message in a collection of messages depending on the inputs of the parties. In our construction, the collection of messages correspond to various inputs of other PSMs or NIMPCs. In other words, selectors compose easily with other NIMPCs. That is why they play a central role in our constructions.

We start by defining general selectors, before considering and constructing two particular cases: linear selectors and NIMPC for Abelian programs. The former selectors are used in our transformation from PSM to $O(1)$-robust NIMPC, while the latter selectors are used for symmetric functions. Our constructions are perfectly fully robust. An interesting point if that these selectors are also new constructions of fully robust NIMPCs (of which very few are known, even assuming the existence of one-way functions).

4.1 Definitions

General definition. The next definition is the general definition. Definitions of the two interesting particular cases follow.

Definition 4.1. *Let $\mathcal{X}_1, \dots, \mathcal{X}_n, \mathcal{U}, \mathcal{M}$ be finite sets. Let $\mathcal{X} \coloneqq \mathcal{X}_1 \times \cdots \times \mathcal{X}_n$. Let $\mathsf{sel} \colon \mathcal{X} \to \mathcal{U}$ be a function. A* selector *for the function sel and the message set $\mathcal{M}$ is an NIMPC protocol for the following set of functions $\{\mathcal{H}_{\mathsf{sel},\tilde{\boldsymbol{m}}}\}_{\tilde{\boldsymbol{m}} \in \mathcal{M}^{\mathcal{U}}}$, where:*

$$h_{\mathsf{sel},\tilde{\boldsymbol{m}}} \colon \boldsymbol{x} \in \mathcal{X} \mapsto \tilde{m}_{\mathsf{sel}(\boldsymbol{x})} \ .$$

The message set $\mathcal{M}$ is often implicitly defined. We also implicitly assume that elements of $\mathcal{M}$ can be represented by vectors of $\lceil \log_q |\mathcal{M}| \rceil$ elements in $\mathbb{F}_q$. The set $\mathcal{M}^{\mathcal{U}}$ is the set of tuples $\boldsymbol{u} = (\tilde{m}_{\boldsymbol{u}})_{\boldsymbol{u} \in \mathcal{U}}$ of messages in $\mathcal{M}$, indexed by elements in $\mathcal{U}$.

In this paper, we are interested in two specific types of selectors: linear selectors and NIMPC for Abelian program.

Linear selectors. Linear selectors are used for our transformation from PSM to $O(1)$-robust NIMPC and are defined as follows.

Definition 4.2 (linear selector). *Let $\mathbb{F}_q$ be a finite field. Let k and n be positive integers. Let $M \in \mathbb{F}_q^{k \times n}$ be a matrix. A* linear selector *for M is a selector for the function sel defined by:*

$$\mathsf{sel} : \boldsymbol{x} \in \mathbb{F}_q^n \mapsto M \cdot \boldsymbol{x} \in \mathcal{U} \coloneqq \mathbb{F}_q^k \ .$$

In the above definition, $\mathcal{X}_1, \dots, \mathcal{X}_n$ are implicitly defined as $\mathbb{F}_q$. The set of messages $\mathcal{M}$ can be any finite set.

NIMPC for Abelian programs. For our construction of NIMPC for symmetric functions, we need to introduce another type of selectors.

Abelian programs can be seen a generalization of symmetric functions introduced in [3, Section 4]. More precisely, we have the following definition.

Definition 4.3 (Abelian program). *Let $\mathbb{G}$ be a finite Abelian group. Let $\mathcal{X}_1, \ldots, \mathcal{X}_n$ be subsets of $\mathbb{G}$. Let $\mathcal{X} \coloneqq \mathcal{X}_1 \times \cdots \times \mathcal{X}_n \subseteq \mathbb{G}^n$ denote their Cartesian product. Let Ω be some finite set. An* Abelian program *for $\mathbb{G}$, $\mathcal{X}$, and Ω is a function:*

$$\tilde{h}_{\tilde{g}} \colon \boldsymbol{x} \in \mathcal{X} \mapsto \tilde{g}(\sum_{i=1}^{n} x_i) \ ,$$

where $\tilde{g} \colon \mathbb{G} \to \Omega$ is a function.

An NIMPC for Abelian program is just an NIMPC for the class of Abelian programs for a given group $\mathbb{G}$, input set $\mathcal{X}$, and output set Ω. In this paper, we prefer to view NIMPC for Abelian programs as selectors, as follows.

Definition 4.4 (NIMPC for Abelian Programs). *Let $\mathbb{G}$ be a finite (additive) Abelian group. Let $\mathcal{X}_1, \ldots, \mathcal{X}_n$ be subsets of $\mathbb{G}$. Let $\mathcal{X} \coloneqq \mathcal{X}_1 \times \cdots \times \mathcal{X}_n \subseteq \mathbb{G}^n$ denote their Cartesian product. An* NIMPC for Abelian programs *(for $\mathcal{X}$ and $\mathbb{G}$) is a selector for the function* sel *defined by:*

$$\mathsf{sel} : \boldsymbol{x} \in \mathcal{X} \mapsto \sum_{i=1}^{n} x_i \in \mathcal{U} =: \mathbb{G} \ .$$

The message $\mathcal{M}$ corresponds to the set Ω.

We remark that if $\mathbb{G}$ is a finite field $\mathbb{F}_q$ and $\mathcal{X}_1 = \cdots = \mathcal{X}_n = \mathbb{G}$, then an NIMPC for Abelian programs for $\mathcal{X}$ and $\mathbb{G}$ is a exactly a linear selector for the matrix $M = (1, \ldots, 1) \in \mathbb{F}_q^{1 \times n}$. However, for our constructions, the sets $\mathcal{X}_i$ are strictly included in the group $\mathbb{G}$. We therefore need to use completely different techniques for the construction of NIMPC for Abelian programs, compared to the ones used for the construction of linear selectors.

4.2 Construction of Linear Selectors

Let us now show how to construct linear selectors. As explained in Section 2.2, we first define and construct outputting-message NIMPC protocols.

Outputting-message NIMPC.

Definition 4.5 (outputting-message NIMPC). *Let $\mathbb{F}_q$ be a finite field and $\mathcal{M}$ be a finite set. Let $M \in \mathbb{F}_q^{k \times n}$ be a matrix. An outputting-message NIMPC for*

M is a NIMPC protocol for the following set of functions $\mathcal{H}_M \coloneqq \{h_{M,\boldsymbol{u},\tilde{m}}\}_{\boldsymbol{u}\in\mathbb{F}_q^k,\tilde{m}\in\mathcal{M}}$ where:

$$h_{M,\boldsymbol{u},\tilde{m}} \colon \boldsymbol{x} \in \mathbb{F}_q^n \mapsto \begin{cases} \tilde{m} & \text{if } \boldsymbol{u} = M \cdot \boldsymbol{x}\,, \\ \perp & \text{otherwise,} \end{cases}$$

where $\perp$ is a fresh symbol not in $\mathcal{M}$.

As for linear selectors, in the above definition, $\mathcal{X}_1, \ldots, \mathcal{X}_n$ are implicitly defined as $\mathbb{F}_q$.

Theorem 4.6. *Let $\mathbb{F}_q$ be a finite field and $\mathcal{M}$ be a finite message set. Let $M \in \mathbb{F}_q^{k\times n}$ be a matrix. There exists a perfectly fully robust outputting-message NIMPC for M with communication complexities:*

$$\begin{aligned} \mathsf{CC}_{\mathsf{on}}(\Pi) &= (k + \lceil \log_q |\mathcal{M}| \rceil) \cdot \lceil \log q \rceil\,, \\ \mathsf{CC}_{\mathsf{off}}(\Pi) &= (k + 2 \cdot \lceil \log_q |\mathcal{M}| \rceil) \cdot \lceil \log q \rceil\ . \end{aligned}$$

Furthermore, the simulator for t-robustness runs in time $q^{\min(t,k)} \cdot \mathrm{poly}(q, k, n, \log|\mathcal{M}|)$. In particular, when t or k is a constant, the simulator runs in polynomial time in q, k, n, and $\log|\mathcal{M}|$.

The term $q^{\min(t,k)}$ in the simulator running time comes from the following fact. The simulator needs to enumerate all the possible input values $\boldsymbol{x}_T$ of the colluding parties P_i ($i \in T$; there are q^t such values) or all the resulting values $M \cdot \boldsymbol{x}_T$ (there are at most q^k such values) to find whether there exists $\boldsymbol{x}_T \in \mathcal{X}_T$, such that $h_{M,\boldsymbol{u},\tilde{m}}|_{\bar{T},\boldsymbol{x}_{\bar{T}}}(\boldsymbol{x}_T) \neq \perp$.

Proof (Theorem 4.6). Fig. 2 describes the construction of the outputting-message NIMPC $(\mathsf{Setup}_M, \mathsf{Msg}_M, \mathsf{Rec}_M)$ for $M \in \mathbb{F}_q^{k\times n}$, when the message set is $\mathcal{M} = \mathbb{F}_q$. The security proof follows the informal presentation from Section 2.3 and is provided in the full version.

To construct an outputting-message NIMPC for an arbitrary message set $\mathcal{M}$ (instead of $\mathbb{F}_q$), we just split the messages in sub-messages in $\mathbb{F}_q$ (in other words, we represent a message in $\mathcal{M}$ as a vector of $\lceil \log_q |\mathcal{M}| \rceil$ elements of $\mathbb{F}_q$) and using an independent instance of the linear selector for each sub-message. To get the communication complexities of the theorem, we remark that the vectors $\boldsymbol{r}'_i$ can be the same for each sub-message. □

Construction of Linear Selectors. We can now construct linear selectors. More precisely, we have the following theorem.

Offline preprocessing (Setup). For the function $h_{M,\boldsymbol{u},\tilde{m}}$ ($M \in \mathbb{F}_q^{k\times n}$, $\boldsymbol{u} \in \mathbb{F}_q^k$, $\tilde{m} \in \mathbb{F}_q$):

1. Pick a uniform vector $\boldsymbol{s} \xleftarrow{R} \mathbb{F}_q^k$. For each $i \in \{1,\dots,n\}$, pick a uniform scalar $r_i \xleftarrow{R} \mathbb{F}_q$ and a uniform vector $\boldsymbol{r}'_i \xleftarrow{R} \mathbb{F}_q^k$.
2. Compute:
$$\mu_0 \coloneqq \tilde{m} - \boldsymbol{s}^\intercal \cdot \boldsymbol{u} - \sum_{i=1}^{n} r_i \ \in \mathbb{F}_q\,, \qquad \boldsymbol{\nu}_0 \coloneqq \boldsymbol{u} + \sum_{i=1}^{n} \boldsymbol{r}'_i \ \in \mathbb{F}_q^k \ .$$
3. For each $i \in \{1,\dots,n\}$, compute $s'_i \coloneqq \boldsymbol{s}^\intercal \cdot M_{\cdot,i}$.
4. $\mathsf{Setup}_M(h_{M,\boldsymbol{u},\tilde{m}})$ outputs $(\rho_0,\dots,\rho_n)$ where:
$$\rho_0 \coloneqq (\mu_0, \boldsymbol{\nu}_0)\,, \qquad \rho_i \coloneqq (s'_i, r_i, \boldsymbol{r}'_i) \qquad \text{for } i \in \{1,\dots,n\} \ .$$

Online messages. On input x_i, party P_i computes

$$\mu_{i,x_i} \coloneqq r_i + s'_i \cdot x_i \ \in \mathbb{F}_q\,, \qquad \boldsymbol{\nu}_{i,x_i} \coloneqq \boldsymbol{r}'_i + M_{\cdot,i} \cdot x_i \ \in \mathbb{F}_q^k\,,$$

and outputs $\mathsf{Msg}(\rho_i, x_i) \coloneqq m_{i,x_i} \coloneqq (\mu_{i,x_i}, \boldsymbol{\nu}_{i,x_i})$.

Reconstruction. For $(\rho_0, m_{1,x_1},\dots,m_{n,x_n})$, output:

$$\mathsf{Rec}(\rho_0, m_{1,x_1},\dots,m_{n,x_n}) \coloneqq \begin{cases} \mu_0 + \sum_{i=1}^n \mu_{i,x_i} & \text{if } \boldsymbol{\nu}_0 = \sum_{i=1}^n \boldsymbol{\nu}_{i,x_i}\,, \\ \bot & \text{otherwise.} \end{cases}$$

Fig. 2: Outputting-message NIMPC $(\mathsf{Setup}_M, \mathsf{Msg}_M, \mathsf{Rec}_M)$ for $M \in \mathbb{F}_q^{k\times n}$

Theorem 4.7. *Let $\mathbb{F}_q$ be a finite field and $\mathcal{M}$ be a finite message set. Let $M \in \mathbb{F}_q^{k\times n}$ be a matrix. There exists a perfectly fully robust linear selector for M with communication complexities:*

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

Furthermore, the simulator for t-robustness runs in time $q^k \cdot \mathrm{poly}(q, k, n, \log|\mathcal{M}|)$. In particular, when k is a constant, the simulator runs in polynomial time in q, n, and $\log|\mathcal{M}|$.

Proof. Fig. 3 describes the construction of a fully robust linear selector $(\mathsf{Setup}_M, \mathsf{Msg}_M, \mathsf{Rec}_M)$ for $M \in \mathbb{F}_q^{k\times n}$, from an outputting-message NIMPC. Complexities are computed assuming the outputting-message NIMPC is the one from Theorem 4.6. The security proof is provided in the full version. □

4.3 NIMPC for Abelian Programs

In [3], Beimel et al. constructed a t-robust NIMPC for any Abelian program. But the complexity is at least $\binom{n}{t} \cdot |\mathcal{M}|$. Because of the factor $\binom{n}{t} = \omega(n^{\log\log n + \log t})$, this t-robust NIMPC protocol is not useful for our construction.

Offline preprocessing (Setup). For the function $h_{\boldsymbol{x}\mapsto M\cdot\boldsymbol{x},\,\tilde{\boldsymbol{m}}}$ ($M \in \mathbb{F}_q^{k\times n}$, $\tilde{\boldsymbol{m}} \in \mathcal{M}^{\mathcal{U}}$, where $\mathcal{U} = \mathbb{F}_q^k$):

1. For each $\boldsymbol{u} \in \mathbb{F}_q^k$, compute $(\rho'_{\boldsymbol{u},0}, \ldots, \rho'_{\boldsymbol{u},n}) \xleftarrow{R} \mathsf{Setup}'_M(h_{M,\boldsymbol{u},\tilde{m}_{\boldsymbol{u}}})$.
2. Pick a random uniform permutation π of $\mathcal{U}$.
3. $\mathsf{Setup}(1^n, 1^{\hat{\aleph}}, h_{M,\tilde{\boldsymbol{m}}})$ outputs $(\rho_0, \ldots, \rho_n)$ where $\rho_i = \left(\rho'_{\pi(\boldsymbol{u}),i}\right)_{\boldsymbol{u}\in\mathcal{U}}$ for $i \in \{0, \ldots, n\}$.

Online messages. On input x_i, party P_i computes $m'_{\boldsymbol{u},i,x_i} \coloneqq \mathsf{Msg}'_M(\rho'_{\boldsymbol{u},i}, x_i)$ for $\boldsymbol{u} \in \mathcal{U}$, and outputs $\mathsf{Msg}(\rho_i, x_i) \coloneqq m_{i,x_i} \coloneqq (m'_{\boldsymbol{u},i,x_i})_{\boldsymbol{u}\in\mathcal{U}}$.

Reconstruction. For $(\rho_0, m_{1,x_1}, \ldots, m_{n,x_n})$, compute for each $\boldsymbol{u} \in \mathcal{U}$:

$$\tilde{m}'_{\boldsymbol{u}} \coloneqq \mathsf{Rec}'_M(\rho'_{\boldsymbol{u},0}, m'_{\boldsymbol{u},1,x_1}, \ldots, m'_{\boldsymbol{u},n,x_n}) \ .$$

Search for $\boldsymbol{u} \in \mathcal{U}$ such that $\tilde{m}_{\boldsymbol{u}} \neq \perp$. Abort if not exactly one such $\boldsymbol{u}$ is found. Otherwise output $\mathsf{Rec}(\rho_0, m_{1,x_1}, \ldots, m_{n,x_n}) \coloneqq \tilde{m}'_{\boldsymbol{u}}$.

Fig. 3: Linear selector $\Pi_M = (\mathsf{Setup}_M, \mathsf{Msg}_M, \mathsf{Rec}_M)$ for $M \in \mathbb{F}_q^{k\times n}$ from outputting-message NIMPC $\Pi'_M = (\mathsf{Setup}'_M, \mathsf{Msg}'_M, \mathsf{Rec}'_M)$

Instead, we propose a fully robust construction based on an information-theoretic variant of Yao's garbled circuits [10,15] (for a specific circuit with gates over $\mathbb{G}$ instead of classical Boolean gates) with communication complexity $O(n \cdot |\mathbb{G}|^{\log n} \cdot (\log|\Omega| + \log|\mathbb{G}|))$. When $\mathbb{G}$ has logarithmic size in n, the communication complexity is only $n^{O(\log\log n)}$, which is only slightly quasi-polynomial.

More formally, we prove the following theorem in the full version.

Theorem 4.8. *Let $\mathbb{G}$ be an Abelian group and $\Omega = \mathcal{M}$ be a finite message set. Let $\mathcal{X}_1, \ldots, \mathcal{X}_n$ be subsets of $\mathbb{G}$. Let $\mathcal{X} \coloneqq \mathcal{X}_1 \times \cdots \times \mathcal{X}_n \subseteq \mathbb{F}_q^n$ denote their Cartesian product. There exists a perfectly fully robust NIMPC Π for Abelian programs (for $\mathbb{G}$, $\mathcal{X}$, and $\mathcal{M}$), with communication complexities:*

$$\begin{aligned} CC_{on}(\Pi) &\leq |\mathbb{G}|^{\lceil\log n\rceil} \cdot (\log|\Omega| + 2 \cdot \lceil\log|\mathbb{G}|\rceil) \,, \\ CC_{off}(\Pi) &= O(n \cdot |\mathbb{G}|^{\lceil\log n\rceil+2} \cdot (\log|\Omega| + \log|\mathbb{G}|)) \ . \end{aligned}$$

5 Admissible Linear Indexing Functions

We recall that the high level idea behind our transformation from a given PSM (or 0-robust NIMPC) protocol to a t-robust NIMPC, is to create a collection of instances (in the form of messages m_{i,x_i}) of the underlying PSM protocol and then use an indexing function that maps parties' inputs to an index that identifies one and only one of these instances. Here we describe the indexing function we use. An informal presentation of the ideas behind this function and its design are presented in Section 2 (more specifically, Section 2.1).

5.1 Definition

Definition 5.1. *Let $\mathcal{X}_1, \ldots, \mathcal{X}_n$ be subsets of $\mathbb{F}_q$ all containing 0. Let $\mathcal{X} := \mathcal{X}_1 \times \cdots \times \mathcal{X}_n \subseteq \mathbb{F}_q^n$ denote their Cartesian product. Let $\mathcal{S}$ be a finite set and $\mathsf{ind}\colon \mathcal{X} \to \mathcal{S}$ be a function. Let $T \subseteq \{1, \ldots, n\}$ be a subset and $t \in \{0, \ldots, n\}$ be an integer.*

The function ind *is a* T-admissible indexing function *if for any $\boldsymbol{x} \in \mathcal{X}_{\bar{T}}$, the values $\mathsf{ind}(\boldsymbol{x} + \boldsymbol{y})$ for $\boldsymbol{y} \in \mathcal{X}_T$ are all distinct. The function* ind *is a* t-admissible indexing function *if it is T-admissible for every subset $T \subseteq \{1, \ldots, n\}$ of size $|T| \leq t$.*

We want $\mathcal{S}$ to be as small as possible as in our transformation we need to consider $|\mathcal{S}|$ instances of the 0-robust protocol. In particular, to have polynomial communication complexity, we need $|\mathcal{S}|$ to be polynomial in n.

We focus on admissible *linear* indexing functions, of the form

$$\mathsf{ind}\colon \boldsymbol{x} \in \mathbb{F}_q^n \mapsto H \cdot \boldsymbol{x} \in \mathbb{F}_q^\ell\,,$$

where $H \in \mathbb{F}_q^{\ell \times n}$ is a matrix. W.l.o.g., we assume H to be full rank (if not, we replace H with a full rank sub-matrix that spans the same row-subspace of $F_q^{1\times n}$). Note that full-rank matrices minimize the size of the indexing function's range $\mathcal{S}$ which in turn improves on the complexity of our construction.

5.2 Relation with Codes

A q-ary code of length n is a subset C of $\mathbb{F}_q^n$ and the distance δ of C is the smallest Hamming distance of two distinct vectors in C.

We have the following lemma.

Lemma 5.2. *Let $t \in \{1, \ldots, n\}$ be an integer. Let $\mathcal{X}_1, \ldots, \mathcal{X}_n, \mathcal{X}, \mathcal{S}$ be defined as in Definition 5.1. Then a function $\mathsf{ind}\colon \mathcal{X} \to \mathcal{S}$ is a t-admissible indexing function (not necessarily linear) if and only if for any $\boldsymbol{\sigma} \in \mathcal{S}$, $\mathsf{ind}^{-1}(\boldsymbol{\sigma})$ is a code of distance $\delta \geq t + 1$.*

Proof. The proof follows from the fact that two vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ of $\mathcal{X}$ have distance at most t if and only if $\boldsymbol{x}_{\bar{T}} = \boldsymbol{y}_{\bar{T}}$ for a subset $T \subseteq \{1, \ldots, n\}$ of size at most t. □

When we restrict ourselves to linear indexing functions, the corresponding codes are either empty or shifts of the same linear code $\mathsf{ind}^{-1}(\mathbf{0})$. We recall that a q-ary linear code of length n and dimension $k = n - \ell$ is a q-ary code of length n that is also a linear subspace C of $\mathbb{F}_q^n$. It can be defined as the kernel of a full-rank matrix $H \in \mathbb{F}_q^{\ell \times n}$ where H is called the parity-check matrix of the code, namely, $C = \{\boldsymbol{x} \in \mathbb{F}_q^n \mid H \cdot \boldsymbol{x} = \mathbf{0}\}$. A q-ary linear code of length n, of dimension k, and of minimum distance δ is called a $[n, k, \delta]_q$-code.

We have the following lemma which is a specialization of Lemma 5.2 to the linear case.

Lemma 5.3. *Let $t \in \{1, \ldots, n\}$ be an integer. Let $H \in \mathbb{F}_q^{\ell \times n}$ be a full-rank matrix. The function $\mathsf{ind}\colon \boldsymbol{x} \in \mathbb{F}_q^n \mapsto H \cdot \boldsymbol{x}$ is a t-admissible linear indexing function if and only if H is a parity-check matrix of a linear code of distance $\delta \geq t+1$.*

Proof. H is a parity-check matrix of a linear code of distance δ if and only if $\mathsf{ind}^{-1}(\mathbf{0})$, the kernel of matrix H, is a linear code of distance δ, and this holds if and only if $\mathsf{ind}^{-1}(\boldsymbol{\sigma})$ is a code of distance δ for all $\boldsymbol{\sigma} \in \mathbb{F}_q^{\ell}$. By Lemma 5.2 the latter condition holds if and only if ind is a t-admissible linear indexing function. □

5.3 Constructions

Constructions of t-admissible linear indexing functions can be obtained using different error correcting codes, in particular Reed-Solomon codes [13] as stated next.

Lemma 5.4. *Let $t \in \{1, \ldots, n\}$ be an integer. Let $q \geq n$ be a prime power. Let $\ell = t$. Then there exists a t-admissible linear indexing function $\mathsf{ind}\colon \boldsymbol{x} \in \mathbb{F}_q^n \mapsto H \cdot \boldsymbol{x}$, for a matrix $H \in \mathbb{F}_q^{\ell \times n}$. In particular, H can be a parity-check of the Reed-Solomon $[n, n-\ell, \ell+1]_q$-code.*

We remark that between n and $2n$, there always exists a power of 2.[6] Therefore, the above lemma shows the existence of t-admissible linear indexing functions with $|\mathcal{S}| = q^{\ell} \leq (2n)^t$.

In the special case where $t = 1$, there is a more efficient construction using the parity code, i.e.:

$$H = \begin{pmatrix} 1 & \ldots & 1 \end{pmatrix} \in \mathbb{F}_q^{1 \times n} \quad .$$

In that case, the prime power q can be any prime power (it does not need to be at least equal to n).

5.4 Lower Bound (on the Need for Constant t)

Using the relation of t-admissible indexing functions and codes of distance $\delta \geq t+1$ (Lemma 5.2) together with a sphere-packing-like Hamming bound, we get the following lower bound on $|\mathcal{S}|$. It shows that if $t = \omega(1)$ (as a function of $n \to \infty$), $|\mathcal{S}|$ cannot be polynomial in n. It is formally proven in the full version.

Lemma 5.5. *Let $t \in \{1, \ldots, n\}$ be an integer. Let $\mathcal{X}_1, \ldots, \mathcal{X}_n, \mathcal{X}, \mathcal{S}$ be defined as in Definition 5.1. We suppose that for any $i \in \{1, \ldots, n\}$, $|\mathcal{X}_i| \geq q'$ for some integer $q' \geq 2$. (In the case of linear indexing functions, $\mathcal{X}_i = \mathbb{F}_q$ and we can take $q' = q$.) If a function $\mathsf{ind}\colon \mathcal{X} \to \mathcal{S}$ is t-admissible, then:*

$$|\mathcal{S}| \geq \sum_{k=0}^{\lfloor t/2 \rfloor} \binom{n}{k} (q'-1)^k \geq \left(\frac{n}{t}\right)^{\lfloor t/2 \rfloor} \quad .$$

[6] Better bounds for intervals containing a prime (power) exist. See [1].

6 From 0-Robustness to $O(1)$-Robustness

Here we present the main construction and result of the paper, namely, a transformation from any PSM (i.e., 0-robust NIMPC) to a t-robust NIMPC where the latter has polynomial complexity for constant t provided the original protocol is polynomial time and the input set for each party is of polynomial size too. The transformation uses two main tools: the linear selector presented in Section 4.2 and admissible linear indexing functions introduced in Section 5. The main ideas and intuition about these tools and constructions are described in Section 2. The transformation is presented in Fig. 4, but first let us formally define what an NIMPC transformation is.

6.1 Definition of an NIMPC Transformation

An NIMPC transformation is a function $\mathcal{T}$ which takes as input an NIMPC protocol $\Pi' = (\mathsf{Setup}', \mathsf{Msg}', \mathsf{Rec}')$ (usually 0-robust) and outputs a new NIMPC protocol Π (usually t-robust for $t > 0$). We focus on blackbox transformations that use the original algorithms $\mathsf{Setup}', \mathsf{Msg}', \mathsf{Rec}'$ in a blackbox way (i.e., as oracles).

For convenience and without loss of generality we assume that the original NIMPC protocols Π' do not use public randomness, namely, $\rho_0 = \perp$ (indeed, if $\rho_0 \neq \perp$, ρ_0 can be appended to ρ_1 and to all the messages sent by the first party P_1).

Definition 6.1 (NIMPC transformation). *Let $\mathcal{X}_1, \ldots, \mathcal{X}_n, \Omega$ be non-empty finite sets. An NIMPC transformation is a tuple of three algorithms $\mathcal{T} = (\mathsf{Setup}, \mathsf{Msg}, \mathsf{Rec})$, each with oracle access to three other algorithms $\Pi' = (\mathsf{Setup}', \mathsf{Msg}', \mathsf{Rec}')$ satisfying the following property:*

- ***Functionality preservation.*** *If $\Pi' = (\mathsf{Setup}', \mathsf{Msg}', \mathsf{Rec}')$ is an NIMPC protocol for some set $\mathcal{F}$ of functions, then $\Pi \coloneqq \mathcal{T}(\Pi') \coloneqq (\mathsf{Setup}^{\Pi'}, \mathsf{Msg}^{\Pi'}, \mathsf{Msg}^{\Pi'})$[7] is also an NIMPC protocol for the same set $\mathcal{F}$ of functions.*

To be useful, an NIMPC transformation also needs to be robust. We consider a very strong notion of robustness. Informally, a transformation $\mathcal{T}$ is T-robust if $\mathcal{T}(\Pi')$ can be proven T-robust for any 0-robust Π', in a black-box way. More formally, we have the following definition.

Definition 6.2 (robustness). *Let $n \in \mathbb{N}_{>0}$ and $T \subseteq \{1, \ldots, n\}$. An NIMPC transformation $\mathcal{T} = (\mathsf{Setup}, \mathsf{Msg}, \mathsf{Rec})$ is T-robust if there exists a simulator $\widetilde{\mathsf{Sim}}$ with oracle access to four oracles $(\mathsf{Setup}', \mathsf{Msg}', \mathsf{Rec}', O)$ such that: if $\Pi' =$*

[7] The notation $\mathsf{Setup}^{\Pi'}$ is a shortcut for $\mathsf{Setup}^{\mathsf{Setup}', \mathsf{Msg}', \mathsf{Rec}'}$, i.e., Setup with the three oracles $\mathsf{Setup}', \mathsf{Msg}', \mathsf{Rec}'$.

$(\mathsf{Setup}', \mathsf{Msg}', \mathsf{Rec}')$ *is an NIMPC protocol, the following two distributions are indistinguishable:*

$$\{\widetilde{\mathsf{Sim}}^{\Pi', O_{n,\mathfrak{K},f,T}}(1^n, 1^{\mathfrak{K}}, T)\} \quad \textit{and} \quad \{\mathsf{View}(1^n, 1^{\mathfrak{K}}, f, T, \boldsymbol{x}_{\bar{T}})\},$$

where $O_{n,\mathfrak{K},f,T} : \boldsymbol{x}_T \in \mathcal{X}_T \mapsto \mathsf{View}'(1^n, 1^{\mathfrak{K}}, f, \emptyset, \boldsymbol{x}_{\bar{T}} + \boldsymbol{x}_T)$*, and* View *and* View' *are the views from running* $\Pi = \mathcal{T}(\Pi')$ *and* Π' *(resp.), as defined in Definition 3.2.*

Let t *be an integer, then an NIMPC transformation is* t*-robust if it is* T*-robust for any subset* $T \subseteq \{1, \ldots, n\}$ *of size at most* t*.*

The power of a T-robust NIMPC transformation for transforming 0-robustness into t-robustness, is shown in the following lemma whose proof follows directly from the above definition.

Lemma 6.3. *Let* $n \in \mathbb{N}_{>0}$*,* $T \subseteq \{1, \ldots, n\}$*. Let* $\mathcal{T} = (\mathsf{Setup}, \mathsf{Msg}, \mathsf{Rec})$ *be a* T*-robust NIMPC transformation. If* Π' *is a perfectly (resp., statistically, computationally)* 0*-robust NIMPC, then* $\Pi = \mathcal{T}(\Pi')$ *is perfectly (resp., statistically, computationally)* T*-robust, with the simulator* Sim *defined as follows:* $\mathsf{Sim}^{f|_{\bar{T}}, \boldsymbol{x}_{\bar{T}}}(1^n, 1^{\mathfrak{K}}, T) = \widetilde{\mathsf{Sim}}^{\Pi', O'_{n,\mathfrak{K},f,T}}(1^n, 1^{\mathfrak{K}}, T)$*, where* $O'_{n,\mathfrak{K},f,T} : \boldsymbol{x}_T \in \mathcal{X}_T \mapsto \mathsf{Sim}'^{f|_{\emptyset, \boldsymbol{x}_{\bar{T}} + \boldsymbol{x}_T}}(1^n, 1^{\mathfrak{K}}, \emptyset)$ *using notation in Definition 6.2 and where* Sim' *is a simulator for* Π'*).*

6.2 Actual Transformation

The main theorem of the paper is presented next. It proves that the transformation described in Fig. 4 is functionality preserving (Definition 6.1) and robust (Definition 6.2).

Theorem 6.4. *The NIMPC transformation* $\mathcal{T} = (\mathsf{Setup}, \mathsf{Msg}, \mathsf{Rec})$ *depicted in Fig. 4 satisfies:*

1. ***Functionality preservation.*** *For any NIMPC protocol* $\Pi' = (\mathsf{Setup}', \mathsf{Msg}', \mathsf{Rec}')$ *for a set of functions* $\mathcal{F}$ *from* $\mathcal{X} = \mathcal{X}_1 \times \cdots \times \mathcal{X}_n$*, the resulting NIMPC protocol* $\Pi = \mathcal{T}(\Pi')$ *has the following online and offline communication complexities (when the underlying linear selector is the one from Theorem 4.7):*

$$\mathsf{CC}_{\mathsf{on}}(\Pi) \leq q^{\ell+1} \cdot n \cdot (\ell + 1 + \lceil \mathsf{CC}_{\mathsf{on}}(\Pi')/\log q \rceil) \cdot \lceil \log q \rceil,$$
$$\mathsf{CC}_{\mathsf{off}}(\Pi) \leq q^{\ell+1} \cdot n \cdot (\ell + 1 + 2 \cdot \lceil \mathsf{CC}_{\mathsf{on}}(\Pi')/\log q \rceil) \cdot \lceil \log q \rceil,$$

where $q \geq \max(|\mathcal{X}_1|, \ldots, |\mathcal{X}_n|)$ *and* q *is a prime power, and* ℓ *is the dimension of the range of the linear indexing function* $\mathsf{ind} \colon \boldsymbol{x} \in \mathbb{F}_q^n \mapsto H \cdot \boldsymbol{x} \in \mathbb{F}_q^\ell$*.*[8]

[8] We recall that Π' is assumed not to use any public randomness: $\rho_0 = \perp$.

Components.

- PSM protocol $\Pi' = (\mathsf{Setup}', \mathsf{Msg}', \mathsf{Rec}')$ for $\mathcal{F}$.
- A prime power $q \geq \max(|\mathcal{X}_1|, \ldots, |\mathcal{X}_n|)$.
- A t-admissible linear indexing function $\mathsf{ind}\colon \boldsymbol{x} \in \mathbb{F}_q^n \mapsto H \cdot \boldsymbol{x} \in \mathbb{F}_q^\ell$.
- A linear selector $\Pi_M = (\mathsf{Setup}_M, \mathsf{Msg}_M, \mathsf{Rec}_M)$ for $M \in \mathbb{F}_q^{k \times n}$ (with $k = \ell + 1$) and for a large enough message space $\mathcal{M}$ containing at least all the possibles messages of the PSM protocol Π'.
- Matrices $M^{(i)}$ and vectors $\boldsymbol{u}^{(\boldsymbol{\sigma},v)}$ for $i \in \{1, \ldots, n\}$, $\boldsymbol{\sigma} \in \mathbb{F}_q^\ell$, $v \in \mathbb{F}_q$, defined as: $M^{(i)} \coloneqq \left(H^\intercal \; \boldsymbol{e}_i^\intercal\right)^\intercal$ and $\boldsymbol{u}^{(\boldsymbol{\sigma},v)} = \left(\boldsymbol{\sigma}^\intercal \; v\right)^\intercal$, where $\boldsymbol{e}_i$ is the i-th vector of the canonical basis of $\mathbb{F}_q^n$.

Offline preprocessing (Setup). For the function $f \in \mathcal{F}$:

1. For $\boldsymbol{\sigma} \in \mathbb{F}_q^\ell$, $i \in \{1, \ldots, n\}$, $v \in \mathcal{X}_i$, compute $(\bot, \rho'_{\boldsymbol{\sigma},1}, \ldots, \rho'_{\boldsymbol{\sigma},n}) \xleftarrow{R} \mathsf{Setup}'(1^n, 1^\Re, f)$ and $m'_{\boldsymbol{\sigma},i,v} \coloneqq \mathsf{Msg}'(\rho'_{\boldsymbol{\sigma},i}, v)$.
2. For $i \in \{1, \ldots, n\}$, define $\tilde{\boldsymbol{m}}^{(i)} \in \mathcal{M}^{\mathbb{F}_q^{\ell+1}}$ by: $\tilde{m}^{(i)}_{\boldsymbol{u}^{(\boldsymbol{\sigma},v)}} = m'_{\boldsymbol{\sigma},i,v}$ for $\boldsymbol{\sigma} \in \mathbb{F}_q^\ell$ and $v \in \mathcal{X}_i$ ($\tilde{m}^{(i)}_{\boldsymbol{u}}$ is arbitrary if $\boldsymbol{u}$ cannot be written as $\boldsymbol{u} = \boldsymbol{u}^{(\boldsymbol{\sigma},v)}$) and then compute $(\rho_{i,0}, \ldots, \rho_{i,n}) \xleftarrow{R} \mathsf{Setup}_{M^{(i)}}(h_{\boldsymbol{x} \mapsto M^{(i)} \cdot \boldsymbol{x},\, \tilde{\boldsymbol{m}}^{(i)}})$.
3. $\mathsf{Setup}(1^n, 1^\Re, f)$ outputs $(\rho_0, \ldots, \rho_n)$ where $\rho_j \coloneqq (\rho_{i,j})_{i \in \{1,\ldots,n\}}$ for $j \in \{0, \ldots, n\}$.

Online messages. On input x_j, party P_j computes $m_{i,j,x_j} \coloneqq \mathsf{Msg}_{M^{(i)}}(\rho_{i,j}, x_j)$, for $i \in \{1, \ldots, n\}$, and outputs $\mathsf{Msg}(\rho_j, x_j) \coloneqq m_{j,x_j} \coloneqq (m_{i,j,x_j})_{i \in \{1,\ldots,n\}}$.

Reconstruction. For $(\rho_0, m_{1,x_1}, \ldots, m_{n,x_n})$, compute $m'_i \coloneqq \mathsf{Rec}_{M^{(i)}}(\rho_{0,i}, m_{i,1,x_1}, \ldots, m_{i,n,x_n})$ for $i \in \{1, \ldots, n\}$, and output $\mathsf{Rec}'(\bot, m'_1, \ldots, m'_n)$.

Fig. 4: Main NIMPC transformation $\mathcal{T} = (\mathsf{Setup}, \mathsf{Msg}, \mathsf{Rec})$

2. ***T-robustness.*** *For any $T \subseteq \{1, \ldots, n\}$, if ind is a T-admissible indexing function and Π_M is a perfectly T-robust linear selector, the NIMPC transformation from Fig. 4 is T-robust. The corresponding simulator $\widetilde{\mathsf{Sim}}$ runs in polynomial time in $n, \Re, q^\ell, \mathsf{CC}_{\mathsf{on}}(\Pi'), |\mathcal{X}_T|$ and calls its oracle O once for each vector $\boldsymbol{x}_T \in \mathcal{X}_T$, when the underlying linear selector is the one from Theorem 4.7.*

The proof of the theorem appears in the full version. We have the following corollary.

Corollary 6.5. *Let t be a positive integer. Let $\Pi' = (\mathsf{Setup}', \mathsf{Msg}', \mathsf{Rec}')$ be an NIMPC protocol for a set of functions $\mathcal{F}$ from $\mathcal{X} = \mathcal{X}_1 \times \cdots \times \mathcal{X}_n$. Let q be the smallest prime power at least equal to $\max(n, |\mathcal{X}_1|, \ldots, |\mathcal{X}_n|)$. We recall that $q \leq 2\max(n, |\mathcal{X}_1|, \ldots, |\mathcal{X}_n|)$.*[9] *Let $\mathsf{ind}\colon \boldsymbol{x} \in \mathbb{F}_q^n \mapsto H \cdot \boldsymbol{x} \in \mathbb{F}_q^\ell$ be the t-admissible linear indexing function defined in Lemma 5.4 (in particular $\ell = t$).*

The NIMPC protocol $\Pi = \mathcal{T}(\Pi')$ from Fig. 4 is perfectly (resp., statistically, computationally) t-robust, if Π' is perfectly (resp., statistically, computationally)

[9] Better bounds for intervals containing a prime (power) exist. See [1].

0-robust. Furthermore, if $t = O(1)$ and the communication complexity of Π' and the input size $\mathcal{X}_i$ (for all i) are all polynomial in n and $\mathfrak{K}$, then the communication complexity of Π is polynomial in n and $\mathfrak{K}$. If in addition Π' is polynomial-time, so is Π. Similarly, if the simulator for Π' is polynomial-time, so is the simulator for Π.

We point out that the simulator $\widetilde{\mathsf{Sim}}$ is uniform in T and n.

7 NIMPC for Symmetric Functions

In this section, we construct NIMPC protocols for symmetric functions with better asymptotic complexity than with our generic transformation (from an efficient 0-robust NIMPC for symmetric function which exists for any symmetric function) or with [3, Section 4]. The communication complexity of the latter construction is $\binom{n}{t} \cdot O(2^t \cdot n^4)$, while our new construction for symmetric functions achieve a communication complexity of $n^{\log\log n + \log t + O(1)}$. Our construction uses our new fully robust NIMPC for Abelian programs in Section 4.3.

7.1 Symmetric Functions

Let us first recall the definition of a symmetric function. We focus on the case where each input is a bit. But our construction can be generalized.

Definition 7.1. *Let n be a positive integer and Ω be a finite set. A function $f\colon \{0,1\}^n \to \Omega$ is symmetric if and only if for any permutation π of $\{1,\dots,n\}$ and for any $\boldsymbol{x} \in \{0,1\}^n$, $f(x_1,\dots,x_n) = f(x_{\pi(1)},\dots,x_{\pi(n)})$.*

7.2 Overview of the Construction

We remark that symmetric functions $f\colon \{0,1\}^n \to \Omega$ are Abelian programs (Definition 4.3) over any group $\mathbb{G} = \mathbb{Z}_N$ with $N > n$:

$$f\colon \boldsymbol{x} \in \{0,1\}^n \mapsto \tilde{g}(\sum_{i=1}^{n} x_i) \in \Omega\,,$$

where $\tilde{g}\colon \mathbb{Z}_N \to \Omega$ is some function.

If we directly use the construction of NIMPC for Abelian programs in Section 4.3, we would get a fully robust NIMPC for symmetric function with communication complexity $n^{\log n + O(1)} \cdot \log|\Omega|$. This is already an interesting result. However, we would like to go further. For that we use the Chinese Remainder Theorem to decompose the initial function over a large group $\mathbb{Z}_{n+1}$ or $\mathbb{Z}$ into functions over smaller groups.

Decomposition and recombination using CRT. Let $p_1 < \cdots < p_\ell$ be the first ℓ prime numbers, such that $N \coloneqq \prod_{j=1}^{\ell} p_j \geq n+1$. We recall that there is a ring isomorphism $\mathrm{CRT}\colon \prod_{j=1}^{\ell} \mathbb{Z}_{p_j} \to \mathbb{Z}_N$. In particular $\mathrm{CRT}(y_1, \ldots, y_\ell)$ is the only integer y in $\{0, \ldots, N-1\}$ such that $y \bmod p_j = y_j$ for any $j \in \{1, \ldots, \ell\}$. By the prime number theorem, we can choose $p_j = O(\log n)$ for $j \in \{1, \ldots, \ell\}$ (and $\ell = O(\log n)$ too). The main idea is the following: we first compute some well-chosen Abelian programs over each group $\mathbb{Z}_{p_j}$ (over the original inputs $(x_1, \ldots, x_n) \in \mathcal{X}$) and then combine back the intermediate results (corresponding to some function of $\sum_{i=1}^{n} x_i \bmod p_j$) to compute $\tilde{g}(\sum_{i=1}^{n} x_i)$.

We need to combine the results in a robust way. We consider a fully robust NIMPC for the following set of functions (with ℓ parties) $\mathcal{F}' = \{f'_{\tilde{g}}\}_{\tilde{g}}$ indexed by a function $\tilde{g}\colon \mathbb{Z}_{n+1} \to \Omega$, where the function $f'_{\tilde{g}}\colon \prod_{j=1}^{\ell} \mathbb{Z}_{p_j} \to \Omega \cup \{\bot\}$ is defined by:

$$f'_{\tilde{g}}(y_1, \ldots, y_\ell) \mapsto \begin{cases} \tilde{g}(y) & \text{if } y \coloneqq \mathrm{CRT}(y_1, \ldots, y_\ell) \in \{0, \ldots, n\}\,, \\ \bot & \text{otherwise.} \end{cases}$$

We can use the construction in [3, Section 3, Theorem 3.3] to get a fully robust NIMPC for $\mathcal{F}'$ of communication complexity $O(N \cdot p_\ell^2 \cdot \ell \cdot \log|\Omega|) = O(n \cdot \log|\Omega| \cdot \mathrm{polylog}(n))$. Let m'_{j,y_j} be the message that party P_j would send on input y_j in this protocol.

For each $j \in \{1, \ldots, \ell\}$, we can then use our construction for Abelian programs in Section 4.3 in the groups $\mathbb{Z}_{p_j}$ for the input sets $\mathcal{X}_1 = \cdots = \mathcal{X}_n = \{0,1\}$ and the messages $\tilde{\boldsymbol{m}}_j$ defined by $\tilde{m}_{j,v} = m'_{j,v}$ (for each $v \in \mathbb{Z}_{p_j}$) to enable the computation (or selection) of m'_{j,y_j} for $y_j = \sum_{i=1}^{n} x_i \bmod p_j$. The resulting construction would have communication complexity $n^{\log\log n + O(1)} \cdot \log|\Omega|$, as $|\mathbb{Z}_{p_j}|^{\log n} = n^{\log\log n + O(1)}$.

Issues with robustness. Unfortunately, this construction is not t-robust: the adversary might use different values x_i for $i \in T$ as input to each NIMPC for Abelian program. For example, if P_1 is colluding, the adversary can compute for any j: m_{j,y_j} and m_{j,y_j+1}, if we write $y_j = \sum_{i=2}^{n} x_i \bmod p_j$. He can then mix and match them when using them as input of the fully robust protocol for $f'_{\tilde{g}}$. In other words, he can compute:

$$\tilde{g}(\mathrm{CRT}(y_1 + b_1, \ldots, y_\ell + b_\ell))$$

for any $(b_1, \ldots, b_\ell) \in \{0,1\}^\ell$, instead of just:

$$\tilde{g}(\mathrm{CRT}(y_1, \ldots, y_\ell)) \quad \text{and} \quad \tilde{g}(\mathrm{CRT}(y_1 + 1, \ldots, y_\ell + 1))\,,$$

(i.e., $b_1 = \cdots = b_\ell \in \{0,1\}$).

One first solution consists in choosing p_j such that when $b_1, \ldots, b_\ell$ are not the same bit (or more generally not the same integer in $\{0, \ldots, t\}$ when t parties are colluding):

$$\mathrm{CRT}(y_1 + b_1, \ldots, y_\ell + b_\ell) > n\,,$$

so that $\tilde{g}(\mathrm{CRT}(y_1 + b_1, \ldots, y_\ell + b_\ell)) = \perp$. This works but makes parameters cumbersome to compute and non-optimal.

We propose a cleaner solution. Instead of working in $\mathbb{Z}_{p_j}$, we work in $\mathbb{G}_j = \mathbb{Z}_{p_j} \times \mathbb{Z}_{t+1}$. The second part z_j of an element $(y_j, z_j) \in \mathbb{G}_j$ plays a role very similar to indexes $\boldsymbol{\sigma}$ in our transformation from 0-robustness to $O(1)$-robustness. It prevents mix and matching values computed from different inputs. We consider a fully robust NIMPC protocol for the following set of functions (with ℓ parties) $\mathcal{F}' = \{f'_{\tilde{g}}\}_{\tilde{g}}$ indexed by a function $\tilde{g}\colon \mathbb{Z}_{n+1} \to \Omega$, where the function $f'_{\tilde{g}}\colon \prod_{j=1}^{\ell} \mathbb{G}_j \to \Omega \cup \{\perp\}$ is defined by:

$$f'_{\tilde{g}}((y_1, z_1), \ldots, (y_\ell, z_\ell)) = \begin{cases} \tilde{g}(y) & \text{if } y \coloneqq \mathrm{CRT}(y_1, \ldots, y_\ell) \in \{0, \ldots, n\} \\ & \text{and } z_1 = \cdots = z_\ell\,, \\ \perp & \text{otherwise.} \end{cases}$$

Let m'_{j,y_j,z_j} be the message that party P_j would send on input (y_j, z_j) in this protocol.

For each $j \in \{1, \ldots, \ell\}$, we now use our construction for Abelian programs in Section 4.3 in the groups $\mathbb{G}_j$ for the input sets $\mathcal{X}_1 = \cdots = \mathcal{X}_n = \{0, 1\}$ and the messages $\tilde{\boldsymbol{m}}_j$ defined by $\tilde{m}_{j,v} = m'_{j,v}$ (for each $v \in \mathbb{G}_j$), where $1 \in \{0, 1\}$ is identified to $(1, 1) \in \mathbb{G}_j$ and $0 \in \{0, 1\}$ is identified to $(0, 0) \in \mathbb{G}_j$. The communication complexity becomes $n^{\log\log n + \log t + O(1)} \cdot \log |\Omega|$.

7.3 Formal Construction

We formally prove the following theorem in the full version.

Theorem 7.2. *Let $\mathcal{F} = \{f_{\tilde{g}}\}_{\tilde{g}}$ be the set of symmetric functions $f_{\tilde{g}}\colon \boldsymbol{x} \in \{0, 1\}^n \mapsto \tilde{g}(\sum_{i=1}^{n} x_i) \in \Omega$, where $\tilde{g}\colon \mathbb{Z}_{n+1} \to \Omega$ and Ω is some finite set. Let t be an integer. There exists a t-robust NIMPC for $\mathcal{F}$ with communication complexity $n^{\log\log n + \log t + O(1)} \cdot \log |\Omega|$. In particular, if $t = O(\log n)$, the communication complexity if $n^{O(\log\log n)} \cdot \log |\Omega|$.*

Acknowledgments. This work was supported by the Defense Advanced Research Projects Agency (DARPA) and Army Research Office (ARO) under Contract No. W911NF-15-C-0236.

References

1. Baker, R.C., Harman, G., Pintz, J.: The difference between consecutive primes, ii. Proceedings of the London Mathematical Society 83(3), 532–562 (2001), http://www.cs.umd.edu/~gasarch/BLOGPAPERS/BakerHarmanPintz.pdf

2. Barrington, D.A.M.: Bounded-width polynomial-size branching programs recognize exactly those languages in nc^1. In: Hartmanis, J. (ed.) Proceedings of the 18th Annual ACM Symposium on Theory of Computing, May 28-30, 1986, Berkeley, California, USA. pp. 1–5. ACM (1986), http://doi.acm.org/10.1145/12130.12131
3. Beimel, A., Gabizon, A., Ishai, Y., Kushilevitz, E., Meldgaard, S., Paskin-Cherniavsky, A.: Non-interactive secure multiparty computation. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014, Part II. LNCS, vol. 8617, pp. 387–404. Springer, Heidelberg (Aug 2014)
4. Feige, U., Kilian, J., Naor, M.: A minimal model for secure computation (extended abstract). In: 26th ACM STOC. pp. 554–563. ACM Press (May 1994)
5. Goldwasser, S., Gordon, S.D., Goyal, V., Jain, A., Katz, J., Liu, F.H., Sahai, A., Shi, E., Zhou, H.S.: Multi-input functional encryption. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 578–602. Springer, Heidelberg (May 2014)
6. Goldwasser, S., Rothblum, G.N.: On best-possible obfuscation. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 194–213. Springer, Heidelberg (Feb 2007)
7. Halevi, S., Ishai, Y., Jain, A., Kushilevitz, E., Rabin, T.: Secure multiparty computation with general interaction patterns. In: Sudan, M. (ed.) ITCS 2016. pp. 157–168. ACM (Jan 2016)
8. Halevi, S., Lindell, Y., Pinkas, B.: Secure computation on the web: Computing without simultaneous interaction. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 132–150. Springer, Heidelberg (Aug 2011)
9. Ishai, Y., Kushilevitz, E.: Private simultaneous message protocols with applications. In: Proceedings of ISTCS. pp. 174–184 (1997)
10. Ishai, Y., Kushilevitz, E.: Perfect constant-round secure computation via perfect randomizing polynomials. In: Widmayer, P., Ruiz, F.T., Bueno, R.M., Hennessy, M., Eidenbenz, S., Conejo, R. (eds.) Automata, Languages and Programming, 29th International Colloquium, ICALP 2002, Malaga, Spain, July 8-13, 2002, Proceedings. Lecture Notes in Computer Science, vol. 2380, pp. 244–256. Springer (2002), http://dx.doi.org/10.1007/3-540-45465-9_22
11. Karchmer, M., Wigderson, A.: On span programs. In: Proceedings of Structures in Complexity Theory. pp. 102–111 (1993)
12. Kilian, J.: Founding cryptography on oblivious transfer. In: 20th ACM STOC. pp. 20–31. ACM Press (May 1988)
13. MacWilliams, F.J., Sloane, N.J.A.: The theory of error-correcting codes. Elsevier (1977)
14. Obana, S., Yoshida, M.: An efficient construction of non-interactive secure multiparty computation. In: Foresti, S., Persiano, G. (eds.) CANS 16. LNCS, vol. 10052, pp. 604–614. Springer, Heidelberg (Nov 2016)
15. Yao, A.C.C.: How to generate and exchange secrets (extended abstract). In: 27th FOCS. pp. 162–167. IEEE Computer Society Press (Oct 1986)
16. Yoshida, M., Obana, S.: On the (in)efficiency of non-interactive secure multiparty computation. In: Kwon, S., Yun, A. (eds.) ICISC 15. LNCS, vol. 9558, pp. 185–193. Springer, Heidelberg (Nov 2016)