Non-Interactive Secure Multiparty Computation*

Amos Beimel
Dept. of Computer Science
Ben Gurion University
Beer Sheva, Israel
amos.beimel@gmail.com

Ariel Gabizon Yuval Ishai Eyal Kushilevitz
Dept. of Computer Science
Technion
Haifa, Israel
arielga,yuvali,eyalk@cs.technion.ac.il

Sigurd Meldgaard
Google Aarhus
Denmark
stm@cs.au.dk

Anat Paskin-Cherniavsky
Dept. of Computer Science and Mathematics
Ariel University
Ariel, Israel
anps83@gmail.com

November 24, 2014

Abstract

We introduce and study the notion of *non-interactive secure multiparty computation* (NIMPC). An NIMPC protocol for a function $f(x_1, \ldots, x_n)$ is specified by a joint probability distribution $R = (R_1, \ldots, R_n)$ and local encoding functions $\mathrm{Enc}_i(x_i, R_i)$, $1 \leq i \leq n$. Given correlated randomness $(R_1, \ldots, R_n) \in_R R$, each party P_i, using its input x_i and its randomness R_i, computes the message $m_i = \mathrm{Enc}_i(x_i, R_i)$. The messages $m_1, \ldots, m_n$ can be used to decode $f(x_1, \ldots, x_n)$. For a set $T \subseteq [n]$, the protocol is said to be T-*robust* if revealing the messages $(\mathrm{Enc}_i(x_i, R_i))_{i \notin T}$ together with the randomness $(R_i)_{i \in T}$ gives the same information about $(x_i)_{i \notin T}$ as an oracle access to the function f restricted to these input values. Namely, a coalition T can learn no more than the restriction of f fixing the inputs of uncorrupted parties, which, in this non-interactive setting, one cannot hope to hide. For $0 \leq t \leq n$, the protocol is *t-robust* if it is T-robust for every T of size at most t and it is *fully robust* if it is n-robust. A 0-robust NIMPC protocol for f coincides with a protocol in the private simultaneous messages model of Feige et al. (STOC 1994).

In the setting of *computational* (indistinguishability-based) security, fully robust NIMPC is implied by multi-input functional encryption, a notion that was recently introduced by Goldwasser et al. (Eurocrypt 2014) and realized using indistinguishability obfuscation. We consider NIMPC in the *information-theoretic* setting and obtain unconditional positive results for some special cases of interest:

- **Group products.** For every (possibly non-abelian) finite group G, the iterated group product function $f(x_1, \ldots, x_n) = x_1 x_2 \ldots x_n$ admits an efficient, fully robust NIMPC protocol.
- **Small functions.** Every function f admits a fully robust NIMPC protocol whose complexity is polynomial in the size of the input domain (i.e., exponential in the total bit-length of the inputs).

*Research by the first three authors and the fifth author received funding from the European Union's Tenth Framework Programme (FP10/2010-2016) under grant agreement no. 259426 ERC-CaC. The first author was also supported by the Frankel center for computer science. Research by the second author received funding from the European Union's Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 257575. The third and fourth authors were supported by ISF grant 1361/10 and BSF grant 2012378.

- **Symmetric functions.** Every symmetric function $f : \mathcal{X}^n \to \mathcal{Y}$, where $\mathcal{X}$ is an input domain of constant size, admits a t-robust NIMPC protocol of complexity $n^{O(t)}$. For the case where f is a w-out-of-n threshold function, we get a fully robust protocol of complexity $n^{O(w)}$.

On the negative side, we show that natural attempts to realize NIMPC using private simultaneous messages protocols and garbling schemes from the literature fail to achieve even 1-robustness.

Keywords: secure multiparty computation, obfuscation, private simultaneous messages protocols, randomized encoding of functions, garbling schemes, multi-input functional encryption.

1 Introduction

We introduce and study the notion of *non-interactive secure multiparty computation* (NIMPC). This notion can be viewed as a common generalization of several previous notions from the literature, including obfuscation, private simultaneous messages protocols, and garbling schemes. It can also be viewed as a simpler and weaker variant of the recently introduced notion of multi-input functional encryption. Before we define the new notion and discuss its relations with these previous notions, we start with a motivating example.

Consider the following non-interactive scenario for secure multiparty computation. Suppose that each of n "honest but curious" parties holds an input $x_i \in \{0,1\}$, and the parties wish to conduct a vote by computing the majority value of their inputs. Moreover, the parties want to minimize interaction by each *independently* sending only a *single* message to each other party.[1] It is clear that in this scenario, without any setup, no meaningful notion of security can be achieved: each party can efficiently extract the input x_i from the message of the corresponding party by just simulating incoming messages from all other parties on inputs x_j such that $\sum_{j \neq i} x_j = \lfloor n/2 \rfloor$.

The question we ask is whether it is possible to get better security by allowing a *correlated randomness* setup. That is, the parties get correlated random strings $(R_1, \ldots, R_n)$ that are drawn from some predetermined distribution. Such a setup is motivated by the possibility of securely realizing it during an offline preprocessing phase, which takes place before the inputs are known (see, e.g. [27], for further motivation). The above attack fails in this model, since a party can no longer simulate messages coming from the other parties without knowing their randomness. On the other hand, it is still impossible to prevent the following generic attack: any set of parties T can simulate the messages that originate from parties in T on any given inputs. This allows the parties of T to learn the output on any set of inputs that is consistent with the other parties' inputs. In the case of computing majority, this effectively means that the parties in T must learn the sum of the other inputs whenever it is in the interval $[\lfloor n/2 \rfloor - |T|, \lfloor n/2 + 1 \rfloor]$. When T is small, this would still leave other parties with a good level of security. Hence, our goal is to obtain protocols that realize this "best possible" security while completely avoiding interaction.

The above discussion motivates the following notion of *non-interactive secure multiparty computation* (NIMPC). An NIMPC protocol for a function $f(x_1, \ldots, x_n)$ is defined by a joint probability distribution $R = (R_1, \ldots, R_n)$ and local encoding functions $\mathrm{Enc}_i(x_i, R_i)$, where $1 \leq i \leq n$. For a set $T \subseteq [n]$, the protocol is said to be *T-robust* (with respect to f) if revealing the messages $(\mathrm{Enc}_i(x_i, R_i))_{i \notin T}$ together with the randomness $(R_i)_{i \in T}$, where $(R_1, \ldots, R_n)$ is sampled from R, gives the same information about $(x_i)_{i \notin T}$ as an oracle access to the function f restricted to these input values. For $0 \leq t \leq n$, the protocol is said to be *t-robust* if it is T-robust for every T of size *at most* t, and it is said to be *fully robust* if it is n-robust.

[1] Alternative motivating scenarios include each party broadcasting a single message, posting it on a public bulletin board such as a Facebook account, or sending a single message to an external referee, who should learn the output.

Recent work on multi-input functional encryption [14] implies that the existence of a fully robust NIMPC protocol for general functions, with indistinguishability based security, is equivalent to indistinguishability obfuscation (assuming the existence of one-way functions). Combined with the recent breakthrough on the latter problem [12], this gives candidate NIMPC protocols for arbitrary polynomial-time computable functions (see Section 1.2 for discussion of these and other related works). The above positive result leaves much to be desired in terms of the underlying intractability assumptions and the potential for being efficient enough for practical use. Motivated by these limitations, we consider the goal of realizing NIMPC protocols with *information-theoretic security* for special cases of interest.

1.1 Our results

We obtain the following unconditional positive results on NIMPC.

GROUP PRODUCTS. For every (possibly non-abelian) finite group G, the iterated group product function $f_G(x_1, \ldots, x_n) = x_1 x_2 \ldots x_n$ admits an efficient, fully robust NIMPC protocol. The construction makes a simple use of Kilian's randomization technique for iterated group products [29]. While the security analysis in the case of abelian groups is straightforward (see Example 2.7), the analysis for the general case turns out to be more involved and is deferred to Section 5. We note that this result *cannot* be combined with Barrington's Theorem [5] to yield NIMPC for NC^1. For this, one would need to assign multiple group elements to each party and enforce nontrivial restrictions on the choice of these elements. In fact, efficient information-theoretic NIMPC for NC^1 is impossible, even with indistinguishability-based security, unless the polynomial-time hierarchy collapses [16] (see Section 1.2).

SMALL FUNCTIONS. We show that *every* function f admits a fully robust NIMPC protocol whose complexity is polynomial in the size of the input domain (i.e., exponential in the total bit-length of the inputs). This result can provide a light-weight solution for functions defined over an input domain of a feasible size. This result is described in Section 3. The technique used for obtaining this result also yields *efficient* protocols for computing OR of n bits and, more generally, w-out-of-n threshold functions where either w or $n - w$ are constant.

SYMMETRIC FUNCTIONS. Finally, we show that every symmetric function $h : \mathcal{X}^n \to \mathcal{Y}$, where $\mathcal{X}$ is an input domain of constant size, admits a t-robust NIMPC of complexity $n^{O(t)}$. Thus, we get a polynomial-time protocol for any constant t. More generally, our solution applies to any branching program over an abelian group G, that is, a function $h : \mathcal{X}_1 \times \cdots \times \mathcal{X}_n \to \mathcal{Y}$ of the form $h(x_1, \ldots, x_n) = f(\sum_{i=1}^{n} x_i)$ for an arbitrary function $f : G \to \mathcal{Y}$ (the complexity in this case is $|G|^{O(t)}$). Useful special cases include the above voting example, its generalization to multi-candidate voting (where the output is a partially ordered list such as "$A > B = C > D$"), as well as natural bidding mechanisms. We note that while this construction is only t-robust, larger adversarial sets T can only learn the sum $\sum_{i \notin T} x_i$ (e.g., the sum of all honest votes in the majority voting example) as opposed to all the inputs of honest parties. This construction is more technically involved than the previous constructions. A high level overview of a special case of the construction and a formal treatment of the general case appear in Section 4. A more efficient variant of the construction for a constant t is described in Section 7. As a byproduct of our more efficient protocol, we get private simultaneous messages protocols (i.e., 0-robust NIMPC protocols) for symmetric functions in which each party sends just $O(n \log n)$ bits.

Inadequacy of existing techniques. On the negative side, in Section 6 we show that natural attempts to realize NIMPC using PSM protocols or garbling schemes from the literature fail to achieve even 1-robustness. This holds even for simple function classes such as symmetric functions.

Applications. Our main motivating application is for scenarios involving secure computations without interaction, such as the one described above. While in the motivating discussion we assumed the parties to be honest-but-curious, offering protection against malicious parties in the above model is in some sense easier than in the standard MPC model. Indeed, malicious parties pose no additional risk to the *privacy* of the honest parties because of the non-interactive nature of the protocol. Moreover, a reasonable level of *correctness* against malicious parties can be achieved via the use of pairwise authentication (e.g., in the case of binary inputs, the correlated randomness setup may give each party MAC-signatures on each of its two possible messages with respect to the verification key of each other party). In the case where multiple parties receive an output, adversarial parties can use their rushing capabilities to make their inputs depend on the information learned on other inputs, unless some simultaneous broadcast mechanism is employed. For many natural functions (such as the majority function) this type of rushing capability in the ideal model is typically quite harmless, especially when T is small. Moreover, this issue does not arise at all in the case where only one party (such as an external server) receives an output.

The goal of eliminating simultaneous interaction in secure MPC protocols was put forward by Halevi, Lindell, and Pinkas (HLP) [21, 18]. In contrast to the HLP model, which requires the parties to sequentially interact with a central server, our protocols are completely non-interactive and may be applied with or without a central server. While HLP assume a standard PKI and settle for computational security, we allow general correlated randomness which, in turn, also allows for information-theoretic security.

The NIMPC primitive can also be motivated by the goal of obtaining garbling schemes [33, 6] or randomized encodings of functions [25, 1] that are robust to leakage of secret randomness. Indeed, in Yao's garbled circuit construction, the secrecy of the input completely breaks down if a pair of input keys is revealed. In Section 6 we show that this is also the case for other garbling schemes and randomized encoding techniques from the literature. The use of t-robust NIMPC can give implementations of garbled circuits and related primitives that are resilient to up to t fully compromised pairs of input keys.

While we did not attempt to optimize the concrete efficiency of our constructions, they seem to be reasonably practical for some natural application scenarios. To give a rough idea of practical feasibility, consider a setting of non-interactive MPC where there are n clients, each holding a single input bit, who send messages to a central server that computes the output. For $n = 20$, our fully robust solution for small functions requires each client to send roughly 6MB of data and store a comparable amount of correlated randomness. In the case of computing a symmetric function, such as the majority function from the above motivating example, one can use an optimized protocol, which appears in Section 7, to get a 1-robust solution with the same message size for $n \approx 1100$ clients (offering full protection against the server and single client and partial protection against larger collusions). In contrast to the above, solutions that rely on general obfuscation techniques are currently quite far from being efficient enough for practical use. We leave open the question of obtaining broader or stronger positive results for NIMPC, either in the information-theoretic setting or in the computational setting without resorting to general-purpose obfuscation techniques.

1.2 Related work

In the following, we discuss connections between NIMPC and several related notions from the literature.

Relation with obfuscation. As was recently observed in the related context of multi-input functional encryption (see below), NIMPC generalizes the notion of obfuscation. The goal of obfuscation is to provide an efficient randomized mapping that converts a circuit (or "program") from a given class into a functionally equivalent circuit that hides all information about the original circuit except its input-output relation. An obfuscation for a given circuit class $\mathcal{C}$ reduces to a fully robust NIMPC for a *universal function* $U_{\mathcal{C}}$ for $\mathcal{C}$.

Concretely, $U_{\mathcal{C}}$ takes two types of inputs: input bits specifying a circuit $C \in \mathcal{C}$, and input bits specifying an input to this circuit. An NIMPC protocol for $U_{\mathcal{C}}$, in which each bit is assigned to a different party, gives rise to the following obfuscation scheme. The obfuscation of a circuit C consists of the message of each party holding a bit of C, together with the randomness of the parties holding the input bits for C. By extending the notion of NIMPC to apply to a class of functions (more accurately, function representations), as we do in the technical sections, it provides a more direct generalization of obfuscation that supports an independent local restriction of each input bit.

In contrast to obfuscation, NIMPC is meaningful and nontrivial to realize even when applied to a single function f (rather than a class of circuits), and even when applied to efficiently learnable functions (in particular, finite functions). Indeed, the requirement of hiding the inputs of uncorrupted parties is hard to satisfy even in such cases.

The relation with obfuscation implies limitations on the type of results on NIMPC one can hope to achieve, as it rules out fully robust protocols with simulation-based security for sufficiently expressive circuit classes [4]. Moreover, it follows from the results of [16] that some functions in NC^1 (in fact, even some families of CNF formulas) do not admit an efficient and fully robust information-theoretic NIMPC protocol, even under an indistinguishability-based definition, unless the polynomial-time hierarchy collapses. However, these negative results on obfuscation do not rule out general solutions with indistinguishability-based security or with a small robustness threshold t, nor do they rule out fully robust solutions with simulation-based security for simple but useful function classes.

Multi-input functional encryption. NIMPC can be viewed as a simplified and restricted form of *multi-input functional encryption*, a generalization of functional encryption [32, 19, 31, 7] that was very recently studied in [14]. Multi-input functional encryption is stronger than NIMPC in several ways, the most important of which is that it requires the correlated randomness to be *reusable* for polynomially many function evaluations. It was shown in [14] that multi-input functional encryption for general circuits can be obtained from indistinguishability obfuscation and a one-way function. Combined with the recent breakthrough on obfuscation [12], this gives plausible candidates for indistinguishability-based multi-input functional encryption, and hence also fully robust NIMPC, for general circuits. This general positive result can only achieve computational security under strong assumptions. In contrast, by only requiring a one-time use of the correlated randomness, the notion of NIMPC becomes meaningful even in the information-theoretic setting considered in this work.

Private simultaneous messages protocols. A 0-robust NIMPC protocol for f coincides with a protocol for f in the private simultaneous messages (PSM) model of Feige, Kilian, and Naor [11, 24]. In this model for non-interactive secure computation, the n parties share a *common* source of randomness that is unknown to an external referee, and they wish to communicate $f(x_1, \ldots, x_n)$ to the referee by sending simultaneous messages depending on their inputs and common randomness. From the messages it received, the referee should be able to recover the correct output but should learn no additional information about the inputs. (PSM protocols in which each party has a single input bit are also referred to as *decomposable randomized encodings* [28] or *projective garbling schemes* [6].) While standard PSM protocols are inherently insecure when the referee colludes with even a single party, allowing general correlated randomness (rather than common randomness) gets around this limitation. A natural approach for obtaining NIMPC protocols from PSM protocols is to let the correlated randomness of each party include only the valid messages on its possible inputs. In Section 6, we show that applying this methodology to different PSM protocols and garbling schemes from the literature typically fails to offer even 1-robustness. In Section 5 we show a case where this methodology does work – using Kilian's PSM protocol for computing the iterated group product [29] yields a fully robust protocol. Finally, we note that our work leaves a very big gap between

the class of boolean functions known to have efficient information-theoretic PSM protocols (or 0-robust NIMPC protocols) and those known to have efficient 1-robust NIMPC protocols. The former class contains the circuit complexity class NC^1 as well as different types of log-space classes [11, 24], whereas the latter only covers a very restricted subclass containing symmetric functions and other special types of functions.

Bounded-collusion functional encryption. In the related context of (single-input) functional encryption, Gorbunov et al. [17] have shown how to achieve security against bounded collusions by combining MPC protocols and randomized encoding techniques. Similarly, bounded-collusion identity-based encryption is easier to construct than full-fledged identify-based encryption [10, 15]. We do not know how to apply similar techniques for realizing t-robust NIMPC. The difference is likely to be inherent: while the positive results in [17, 10, 15] rely on standard intractability assumptions and apply even for collusion bounds t that are bigger than the security parameter, a similar general result for NIMPC would suffice to imply general (indistinguishability) obfuscation.

2 Preliminaries

Notation 2.1. *For a set $\mathcal{X} = \mathcal{X}_1 \times \cdots \times \mathcal{X}_n$ and $T \subseteq [n]$ we denote $\mathcal{X}_T \triangleq \prod_{i \in T} \mathcal{X}_i$. For $x \in \mathcal{X}$, we denote by x_T the restriction of x to $\mathcal{X}_T$, and for a function $h : \mathcal{X} \to \Omega$, a subset $T \subseteq [n]$, and $x_{\overline{T}} \in \mathcal{X}_{\overline{T}}$, we denote by $h|_{\overline{T}, x_{\overline{T}}} : \mathcal{X}_T \to \Omega$ the function h where the inputs in $\mathcal{X}_{\overline{T}}$ are fixed to $x_{\overline{T}}$.*

An NIMPC protocol for a family of functions $\mathcal{H}$ is defined by three algorithms: (1) a randomness generation algorithm Gen, which given a description of a function $h \in \mathcal{H}$ generates n correlated random inputs $R_1, \ldots, R_n$, (2) a local encoding function Enc_i (where $1 \leq i \leq n$), which takes an input x_i and a random input R_i and outputs a message, and (3) a decoding algorithm Dec that reconstructs $h(x_1, \ldots, x_n)$ from the n messages. Formally:

Definition 2.2 (NIMPC: Syntax and Correctness)**.** *Let $\mathcal{X}_1, \ldots, \mathcal{X}_n, \mathcal{R}_1, \ldots, \mathcal{R}_n$, $\mathcal{M}_1, \ldots, \mathcal{M}_n$ and Ω be finite domains. Let $\mathcal{X} \triangleq \mathcal{X}_1 \times \cdots \times \mathcal{X}_n$ and let $\mathcal{H}$ be a family of functions $h : \mathcal{X} \to \Omega$. A* non-interactive secure multiparty computation (NIMPC) protocol for $\mathcal{H}$ *is a triplet $\Pi = (\mathrm{Gen}, \mathrm{Enc}, \mathrm{Dec})$ where*

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

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

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

The communication complexity *of Π is the maximum of $\log |\mathcal{R}_1|, \ldots, \log |\mathcal{R}_n|, \log |\mathcal{M}_1|, \ldots, \log |\mathcal{M}_n|$.*

We next define the notion of t-robustness for NIMPC, which informally states that every t parties can only learn the information they should. Note that in our setting, a coalition T of size t can compute many outputs from the messages of $\overline{T}$, namely, they can repeatedly encode any inputs for the coalition T and decode h with the new encoded inputs and the original encoded inputs of $\overline{T}$. In other words, they have oracle access to $h|_{\overline{T}, x_{\overline{T}}}$ (as defined in Notation 2.1). Robustness requires that they learn no other information.

Definition 2.3 (NIMPC: Robustness)**.** *For a subset $T \subseteq [n]$, we say that an* NIMPC *protocol Π for $\mathcal{H}$ is* T-robust *if there exists a randomized function Sim_T (a "simulator") such that, for every $h \in \mathcal{H}$ and $x_{\overline{T}} \in \mathcal{X}_{\overline{T}}$, we have $\mathrm{Sim}_T(h|_{\overline{T},x_{\overline{T}}}) \equiv (M_{\overline{T}}, R_T)$, where R and M are the joint randomness and messages defined by $R \leftarrow \mathrm{Gen}(h)$ and $M_i \leftarrow \mathrm{Enc}_i(x_i, R_i)$.*

For an integer $0 \leq t \leq n$, we say that Π is t-robust *if it is T-robust for every $T \subseteq [n]$ of size $|T| \leq t$. We say that Π is* fully robust *(or simply refer to Π as an* NIMPC *for $\mathcal{H}$) if Π is n-robust. Finally, given a concrete function $h : \mathcal{X} \to \Omega$, we say that Π is a (t-robust)* NIMPC protocol for h *if it is a (t-robust)* NIMPC *for $\mathcal{H} = \{h\}$.*

As the same simulator Sim_T is used for every $h \in \mathcal{H}$ and the simulator has only access to $h|_{\overline{T},x_{\overline{T}}}$, NIMPC hides both h and the inputs of $\overline{T}$ (to the extent possible).

Remark 2.4 (0-robust NIMPC and PSM)**.** *An* NIMPC *protocol Π is* 0-robust *if it is $\emptyset$-robust. In this case, the only requirement is that the messages $(M_1, \ldots, M_n)$ reveal $h(x)$ and nothing else. A* 0*-robust* NIMPC *for h corresponds to a* private simultaneous messages *(PSM) protocol in the model of [11]. Note that in a 0-robust* NIMPC *one can assume, without loss of generality, that the n outputs of Gen are identical. In contrast, it is easy to see that in a 1-robust* NIMPC *of a nontrivial h more general correlations are required.*

While the above definitions treat functions h as finite objects and do not refer to computational complexity, our constructions are computationally efficient in the sense that the total computational complexity is polynomial in the communication complexity. Furthermore, with the exception of the protocol from Lemma 3.2, the same holds for the efficiency of the simulator Sim_T (viewing the latter as an algorithm having *oracle access* to $h|_{\overline{T},x_{\overline{T}}}$). When taking computational complexity into account, the function Gen should be allowed to depend not only on h itself but also on its specific representation (such as a branching program computing h).

Remark 2.5 (Statistical and computational variants)**.** *In this work, we consider* NIMPC *protocols with perfect security, as captured by Definition 2.3. However, one could easily adapt the above definitions to capture statistical security and computational security. In the statistical case, we let Gen receive a security parameter κ as an additional input, and require that the two distributions in Definition 2.3 be $(2^{-\kappa})$-close in statistical distance, rather than identical. In the computational case, we have two main variants corresponding to the two main notions of obfuscation from the literature. In both cases, we require that the two distributions in Definition 2.3 be computationally indistinguishable. The difference is in the power of the simulator. If the simulator is unbounded, we get an indistinguishability-based* NIMPC *for which a general construction is implied by indistinguishability obfuscation [12, 14]. If the simulator is restricted to probabilistic polynomial time, we get the stronger "virtual black-box" variant to which the impossibility results from [4] apply and one can only hope to get general positive results in a generic model [8, 3] or using tamper-proof hardware [20]. We note, however, that the latter impossibility results only apply to function classes that are rich enough to implement "cryptographically nontrivial" functions such as pseudo-random functions. In particular, they do not apply to efficiently learnable classes for which obfuscation is trivial. In contrast,* NIMPC *is meaningful and nontrivial even in the latter case.*

Remark 2.6 (Offline communication)**.** *In our model for NIMPC, we assume all communication happens after the inputs are known. However, it is sometimes useful to separate between offline communication, that can take place after the function h is known but before the inputs x_i are known, and online communication that takes place once the inputs are known. This offline-online model for NIMPC can be captured by modifying the output of Gen to include an additional entry R_0, which captures the offline communication.*

The value of R_0 is given as an additional input to the decoder and should be correctly simulated by Sim *(namely, the random variable R_T should be redefined to always include R_0). An example for the usefulness of the offline-online model is in the context of computationally 0-robust NIMPC protocols, where the offline communication of a garbled circuit can be used to make the online communication complexity close to the input length, regardless of the complexity of h [33, 2].*

As a simple example, we present an NIMPC protocol for summation in an abelian group.

Example 2.7 (NIMPC for sum)**.** *Let G be an abelian group, and define $h : G^n \to G$ by $h(X_1, \ldots, X_n) = X_1 + \cdots + X_n$ (where the sum is in G). We next define a fully robust* NIMPC *for h. Algorithm* Gen *chooses $n-1$ random elements $R_1, \ldots, R_{n-1}$ in G, where each element is chosen independently with uniform distribution, and computes $R_n = -\sum_{i=1}^{n-1} R_i$. The output of* Gen *is $(R_1, \ldots, R_n)$. Algorithm* Enc *computes $\mathrm{Enc}_i(x_i, R_i) = x_i + R_i \triangleq M_i$. Algorithm* Dec *simply sums the n outputs of* Enc*, that is, computes $\sum_{i=1}^{n} M_i$.*

As $\sum_{i=1}^{n} M_i = \sum_{i=1}^{n} x_i + \sum_{i=1}^{n} R_i$, and $\sum_{i=1}^{n} R_i = 0$, correctness follows. We next show that this construction is fully robust. Fix a set $T \subseteq [n]$ and define the simulator Sim_T for T. On inputs x_T, it queries $h|_{\overline{T}, x_{\overline{T}}}(0^{|T|})$ and gets $\mathrm{sum} = \sum_{i \in \overline{T}} x_i$. The simulator then chooses $n-1$ random elements $\rho_1, \ldots, \rho_{n-1}$ in G, each element is chosen independently with uniform distribution, and computes $\rho_n = \mathrm{sum} - \sum_{i=1}^{n-1} \rho_i$. The output of the simulator is $((\rho_i)_{i \in \overline{T}}, (\rho_i)_{i \in T})$.

The following easily verifiable claim states that for functions outputting more than one bit, we can compute each output bit separately. Thus, from now on we will mainly focus on boolean functions.

Claim 2.8. *Let $\mathcal{X} \triangleq \mathcal{X}_1 \times \cdots \times \mathcal{X}_n$, where $\mathcal{X}_1, \ldots, \mathcal{X}_n$ are some finite domains. Fix an integer $m > 1$. Suppose $\mathcal{H}$ is a family of boolean functions $h : \mathcal{X} \to \{0,1\}$ admitting an* NIMPC *protocol with communication complexity S. Then, the family of functions $\mathcal{H}^m = \{h : \mathcal{X} \to \{0,1\}^m \mid h = h_1 \circ \ldots \circ h_m,\ h_i \in \mathcal{H}\}$ admits an* NIMPC *protocol with communication complexity $S \cdot m$.*

2.1 NIMPC with an output server

While an NIMPC protocol Π as defined above can be viewed as an abstract primitive, in the following it will be convenient to describe our constructions in the language of protocols. Such a protocol involves n players $P_1, \ldots, P_n$, each holding an input $x_i \in \mathcal{X}_i$, and an external "output server," a player P_0 with no input. The protocol may have an additional input, a function $h \in \mathcal{H}$. We will let $\mathrm{P}(\Pi)$ denote a protocol that proceeds as follows.

Protocol $\mathrm{P}(\Pi)(h)$

- **Offline preprocessing:** Each player P_i, $1 \le i \le n$, receives the random input $R_i \triangleq \mathrm{Gen}(h)_i \in \mathcal{R}_i$.
- **Online messages:** On input R_i, each player P_i, $1 \le i \le n$, sends the message $M_i \triangleq \mathrm{Enc}_i(x_i, R_i) \in \mathcal{M}_i$ to P_0.
- **Output:** P_0 computes and outputs $\mathrm{Dec}(M_1, \ldots, M_n)$.

We informally note the relevant properties of protocol $\mathrm{P}(\Pi)$:

- For any $h \in \mathcal{H}$ and $x \in \mathcal{X}$, the output server P_0 outputs, with probability 1, the value $h(x_1, \ldots, x_n)$.

- Fix $T \subseteq [n]$. Then, Π is T-robust if in $\mathrm{P}(\Pi)$ the set of players $\{P_i\}_{i\in T} \cup \{P_0\}$ can simulate their view of the protocol (i.e., the random inputs $\{R_i\}_{i\in T}$ and the messages $\{M_i\}_{i\in\overline{T}}$) given oracle access to the function h restricted by the other inputs (i.e., $h|_{\overline{T},x_{\overline{T}}}$).
- Π is 0-robust if and only if in $\mathrm{P}(\Pi)$ the output server P_0 learns nothing but $h(x_1,\ldots,x_n)$.

In Appendix A we give a more general treatment of non-interactive MPC, including security definitions and extensions to the case where multiple parties may have different outputs and to the case of security against malicious parties.

3 An inefficient NIMPC for arbitrary functions

The main purpose of this section is to present an NIMPC protocol for the set of all functions. The communication complexity of the protocol is polynomial in the size of the input domain, namely it is exponential in the bit-length of the inputs. It will be useful to first present an NIMPC protocol for *indicator* functions. For reasons to be clarified later on, it will be convenient to include the zero-function.

Definition 3.1. *Let $\mathcal{X}$ be a finite domain. For n-tuple $a = (a_1,\ldots,a_n) \in \mathcal{X}$, let $h_a : \mathcal{X} \to \{0,1\}$ be the function defined by $h_a(a) = 1$, and $h_a(x) = 0$ for all $a \neq x \in \mathcal{X}$. Let $h_0 : \mathcal{X} \to \{0,1\}$ be the function that is identically zero on $\mathcal{X}$. Let $\mathcal{H}_{\mathrm{ind}} \triangleq \{h_a\}_{a\in\mathcal{X}} \cup \{h_0\}$ be the set of all indicator functions together with h_0.*

Note that every function $h : \mathcal{X} \leftarrow \{0,1\}$ can be expressed as the sum of indicator functions, namely, $h = \sum_{a\in\mathcal{X},h(a)=1} h_a$.

Lemma 3.2. *Fix finite domains $\mathcal{X}_1,\ldots,\mathcal{X}_n$ such that $|\mathcal{X}_i| \leq d$ for all $1 \leq i \leq n$ and let $\mathcal{X} \triangleq \mathcal{X}_1 \times \cdots \times \mathcal{X}_n$. Then, there is an* NIMPC *protocol Π_{ind} for $\mathcal{H}_{\mathrm{ind}}$ with communication complexity at most $d^2 \cdot n$.*

Proof. For $i \in [n]$, denote $|\mathcal{X}_i| = d_i$. Let $s = \sum_{i=1}^n d_i$. Fix a function $h \in \mathcal{H}$ that we want to compute. The NIMPC $\Pi_{\mathrm{ind}}(h)$ is as follows.

Correlated randomness generation: If $h = h_0$, then choose s linearly independent random vectors $\{m_{i,b}\}_{i\in[n],b\in\mathcal{X}_i}$ in $\mathbb{F}_2^s$. If $h = h_a$ for some $a = (a_1,\ldots,a_n) \in \mathcal{X}$, choose s random vectors $\{m_{i,b}\}_{i\in[n],b\in\mathcal{X}_i}$ in $\mathbb{F}_2^s$ under the constraint that $\sum_{i=1}^n m_{i,a_i} = 0$ and there are no other linear relations between them (that is, choose all the vectors $m_{i,b}$, except m_{n,a_n}, as random linear independent vectors and set $m_{n,a_n} = -\sum_{i=1}^{n-1} m_{i,a_i}$). Now define $\mathrm{Gen}_{\mathrm{ind}}(h) = R = (R_1,\ldots,R_n)$, where $R_i = \{m_{i,b}\}_{b\in\mathcal{X}_i}$.

Encoding: $\mathrm{Enc}_{\mathrm{ind}}(x,r) = (M_1,\ldots,M_n)$, where $M_i \triangleq m_{i,x_i}$.

Decoding $h(x_1,\ldots,x_n)$: $\mathrm{Dec}_{\mathrm{ind}}(M_1,\ldots,M_n) = 1$ if $\sum_{i=1}^n M_i = \mathbf{0}$.

For the correctness, note that $\sum_{i=1}^n M_i = \sum_{i=1}^n m_{i,x_i}$. If $h = h_a$, for $a \in \mathcal{X}$, this sum equals 0 if and only if $x = a$. If $h = h_0$, this sum is never zero, as all vectors were chosen to be linearly independent in this case.

To prove robustness, fix a subset $T \subsetneq [n]$ and $x_{\overline{T}} \in \mathcal{X}_{\overline{T}}$. The encodings $M_{\overline{T}}$ of $\overline{T}$ consist of the vectors $\{m_{i,x_i}\}_{i\in\overline{T}}$. The randomness R_T consists of the vectors $\{m_{i,b}\}_{i\in T,b\in\mathcal{X}_i}$. If $h|_{\overline{T},x_{\overline{T}}} \equiv 0$, then these vectors are uniformly distributed in $\mathbb{F}_2^s$ under the constraint that they are linearly independent. If $h|_{\overline{T},x_{\overline{T}}}(x_T) = 1$, for some $x_T \in \mathcal{X}_T$, then $\sum_{i=1}^n m_{i,x_i} = 0$ and there are no other linear relations between them. Formally, to prove robustness, we describe a simulator Sim_T: the simulator queries $h|_{\overline{T},x_{\overline{T}}}$ on all possible inputs in $\mathcal{X}_T$. If all answers are zero, the simulator generates random independent vectors. Otherwise, there is an $x_T \in \mathcal{X}_T$

such that $h|_{\overline{T},x_{\overline{T}}}(x_T) = 1$, and the simulator outputs random vectors under the cor
that is, all vectors are independent with the exception that $\sum_{i=1}^{n} m_{i,x_i} = 0$.

As for communication complexity, the correlated randomness is composed of
length $s \leq dn$ and the encoding is one of them. Hence, the communication comple

We next present an NIMPC for all boolean functions with domain $\mathcal{X} = \mathcal{X}_1 \times$
express any $h : \mathcal{X} \to \{0, 1\}$ as a sum of indicator functions, that is, $h = \sum_{a \in \mathcal{X}}$
an NIMPC for h by using the NIMPC protocols for each h_a. A naive implementa
problems. First, it will disclose information on how many 1's the function h has. To
we define $h'_a = h_a$ if $h(a) = 1$ and $h'_a = h_0$ otherwise (this was the motivation c
Thus, $h = \sum_{a \in \mathcal{X}} h'_a$. The second problem is that if, for example, $h(x) = 1$ an
$h'_a(x) = 1$, then the coalition learns that $x = a$

We next construct a simulator Sim_T for the protocol $\mathrm{P}(\Pi)$ on function h. Simulator Sim_T uses the simulator $\mathrm{Sim}_T^{\Pi_{\mathrm{ind}}}$ – the simulator for the set T for the NIMPC Π_{ind} of Lemma 3.2. The simulator Sim_T first queries $h|_{\overline{T},x_{\overline{T}}}(x_T)$ for every $x_T \in \mathcal{X}_T$. Let $I' \subseteq \mathcal{X}_T$ be the set of ones of $h|_{\overline{T},x_{\overline{T}}}$. For every $x_T \in I'$, the simulator Sim_T computes $S_{x_T} = \mathrm{Sim}_T^{\Pi_{\mathrm{ind}}}(h_{x_T})$ (where $h_{x_T} : \mathcal{X}_T \to \{0,1\}$ is such that $h_{x_T}(x) = 1$ if and only if $x = x_T$). Finally, Sim_T samples $\mathrm{Sim}_T^{\Pi_{\mathrm{ind}}}(h_0)$ for $|\mathcal{X}| - |I'|$ times (where $h_0 : \mathcal{X}_T \to \{0,1\}$ such that $h_0(x) = 0$ for every $x \in \mathcal{X}_T$). Altogether, it obtains $|\mathcal{X}|$ outputs of the simulator $\mathrm{Sim}_T^{\Pi_{\mathrm{ind}}}$. It randomly permutes the order of these outputs, and returns the permuted outputs. The T-robustness of Sim_T follows from the T-robustness of $\mathrm{Sim}_T^{\Pi_{\mathrm{ind}}}$ and Observation 3.4. □

Remark 3.5. *In the above proof, instead of looking at the set of all functions, we could have looked at the set of functions that are OR's of a fixed subset $\mathcal{H}' \subset \mathcal{H}_{\mathrm{ind}}$ of indicator functions. For this set of functions, we would get an* NIMPC *with communication complexity $|\mathcal{H}'| \cdot m \cdot \mathrm{poly}(d,n)$ (rather than the $|\mathcal{X}| \cdot m \cdot \mathrm{poly}(d,n)$ communication complexity above). We could also look at a particular function of this form. Take, for example, $\mathcal{X} = \{0,1\}^n$ and $\mathcal{H}'$ to be the set of indicator functions of vectors of weight w. Then, we get an* NIMPC *for the w-out-of-n threshold function with communication complexity $n^{O(w)}$.*

4 A t-robust NIMPC for abelian programs

In this section, we present an NIMPC for symmetric functions. In fact, this result is a corollary of a more general result on NIMPC for *abelian group programs*. In Section 4.1, we define abelian programs and symmetric functions and formally state our results. In Section 4.2, we give a high level overview of the construction. In Section 4.3, we supply a detailed description of our construction and prove its correctness.

4.1 Our results

Definition 4.1. *Let G be an abelian group, $S_1, \ldots, S_n$ be subsets of G, and $\mathcal{H}^G_{S_1,\ldots,S_n}$ be the set of functions $h : S_1 \times \cdots \times S_n \to \{0,1\}$ of the form $h(X_1, \ldots, X_n) = f(\sum_{i=1}^n X_i)$, for some $f : G \to \{0,1\}$. We call such functions h abelian programs.*

Definition 4.2. *A function $h : [d]^n \to \{0,1\}$ is* symmetric *if for every $(x_1, \ldots, x_n) \in [d]^n$ and every permutation $\pi : [n] \to [n]$ the following equality holds $h(x_1, \ldots, x_n) = h(x_{\pi(1)}, \ldots, x_{\pi(n)})$.*

The main positive result in this section is an efficient t-robust NIMPC for $\mathcal{H}^G_{S_1,\ldots,S_n}$ whenever G is abelian of $\mathrm{poly}(n)$-size and t is constant.

Theorem 4.3. *Let t be a positive integer and G an abelian group of size m. Let $S_1, \ldots, S_n$ be subsets of G. Then, there is a t-robust* NIMPC *protocol for $\mathcal{H}^G_{S_1,\ldots,S_n}$ with communication complexity $O(d^{t+2} \cdot n^{t+2} \cdot m^3)$, where $d \triangleq \max_{i\in[n]} |S_i|$. If $G = \mathbb{Z}_{n+1}$ and $S_1 = \cdots = S_t = \{0,1\}$, then the communication complexity of the* NIMPC *is $O(2^t \cdot n^{t+4})$.*

Corollary 4.4. *Let d, t and n be positive integers. Let $\mathcal{H}$ be the set of symmetric functions $h : [d]^n \to \{0,1\}$. There is a t-robust* NIMPC *protocol for $\mathcal{H}$ with communication complexity $O(d^{t+2} \cdot n^{t+3d-1})$. For the class of boolean symmetric functions $h : \{0,1\}^n \to \{0,1\}$, the communication complexity is $O(2^t \cdot n^{t+4})$.*

Proof. The corollary is proved by showing that every symmetric function can be computed by an abelian program over $\mathbb{Z}^d_{n+1}$. Specifically, given an input $x = (x_1, \ldots, x_n) \in [d]^n$, the abelian program first computes a vector $z = (z_1, \ldots, z_d) \in \mathbb{Z}^d_{n+1}$ such that $z_j = |\{i : x_i = j\}|$ (i.e., z_j is the number of appearances

of j in x). As $0 \leq z_j \leq n$, we consider it as an element in $\mathbb{Z}_{n+1}$. By definition of symmetric functions, the output of h can be computed from z.

Specifically, define $S_1 = \cdots = S_n = \{e_1, \ldots, e_d\} \subseteq \mathbb{Z}_{n+1}^d$, where $e_i \in \mathbb{Z}_{n+1}^d$ is the i'th unit vector. Given a symmetric function $h : [d]^n \to \{0,1\}$, we consider a symmetric function h' whose domain is $\{e_1, \ldots, e_d\}^n$. That is, for an input $j \in [d]$, define $\phi(j) \triangleq e_j$ and for an input $(x_1 \ldots, x_n) \in [d]^n$, look at the sum $z \triangleq \sum_{i=1}^n \phi(x_i) \in \mathbb{Z}_{n+1}^d$. For any symmetric function $h : [d]^n \to \{0,1\}$, there exists a function $f : \mathbb{Z}_{n+1}^d \to \{0,1\}$ such that $h(x_1, \ldots, x_n) = f(\sum_{i=1}^n \phi(x_i))$. Define $h' : S_1 \times \cdots \times S_n \to \{0,1\}$ by $h'(y_1, \ldots, y_n) \triangleq f(\sum_{i=1}^n y_i)$. Note that $h' \in \mathcal{H}_{S_1,\ldots,S_n}$.

Thus, by first mapping the inputs x_i to $\phi(x_i)$, we get a t-robust NIMPC of $\mathcal{H}$ using a t-robust NIMPC of $\mathcal{H}_{S_1,\ldots,S_n}$. Theorem 4.3 gives us a t-robust NIMPC of $\mathcal{H}_{S_1,\ldots,S_n}$ of communication complexity $O(d^{t+2} \cdot n^t \cdot |\mathbb{Z}_{n+1}^d|^3) = O(d^{t+2} \cdot n^t \cdot n^{3d})$.

We can optimize the above construction by noting that in the above definition of z, it must be that $\sum_{\ell=1}^d z_j = n$. Therefore, the above function f can ignore the last coordinate of $\sum_{i=1}^n \phi(x_i)$, and we can define the abelian program over the group $\mathbb{Z}_{n+1}^{d-1}$, resulting in an NIMPC of communication complexity $O(d^{t+2} \cdot n^t \cdot |\mathbb{Z}_{n+1}^{d-1}|^3) = O(d^{t+2} \cdot n^t \cdot n^{3d-3}) = O(d^{t+2} \cdot n^{t+3d-3})$ for $d > 2$ and communication complexity $O(2^t n^{t+4})$ for $d = 2$ (i.e., for boolean symmetric functions). □

Note that Corollary 4.4 is a special case of the following: Whenever we can "embed" a function h (or a set of functions) into an abelian group program, we can get an NIMPC for h whose communication complexity depends polynomially on the size of the group.

Corollary 4.5. *Fix integers $t \leq n$, finite domains $\Omega_1, \ldots \Omega_n$, an abelian group G and $S_1, \ldots, S_n \subseteq G$. Let $d = \max_{i \in [n]} |S_i|$. Fix a function $h : \Omega_1 \times \cdots \times \Omega_n \to \{0,1\}$. Suppose we have mappings $\phi_i : \Omega_i \to S_i$, for $1 \leq i \leq n$, and a function $f : G \to \{0,1\}$ such that for every $(x_1, \ldots, x_n) \in \Omega_1 \times \cdots \times \Omega_n$ $h(x_1, \ldots, x_n) = f(\sum_{i=1}^n \phi_i(x_i))$. Then, there is a t-robust* NIMPC *of h with communication complexity $O(d^{t+2} \cdot n^{t+2} \cdot |G|^3)$.*

4.2 Overview of the construction

In this section we give a high level overview of our t-robust protocol for abelian programs. Before outlining our construction, we start with a simple approach for adding some robustness to an arbitrary PSM protocol. (Recall that a PSM protocol is a $\emptyset$-robust NIMPC protocol.) Consider a PSM protocol for a function $h(x_1, \ldots, x_n)$. Suppose that we would like to make this protocol $\{i\}$-robust, in addition to being $\emptyset$-robust. To this end, the following transformation can be used. For each possible value v of x_i, all parties except P_i run an instance of the protocol in which the common randomness is unknown to P_i, and some fixed party (other than P_i) sends the message of P_i corresponding to the input $x_i = v$. In this PSM, P_0 cannot compute the correct output; this problem will be fixed later.

The above protocol is $\{i\}$-robust, since P_0 cannot and P_i together learn no more than the values of h on the inputs of the honest parties with each possible value of x_i. However, the protocol is not $\emptyset$-robust. To make it $\emptyset$-robust without compromising $\{i\}$-robustness, we apply the following masking technique. First, the instances of the PSM protocol are permuted by a random permutation σ that is known to all parties, including P_i, but not to P_0. The pointer $\sigma(x_i)$ to the correct instance is sent by P_i. This allows P_0 to learn the correct output without allowing P_0 to directly learn the correct value of x_i. However, this protocol is not $\emptyset$-robust, since P_0 can still learn the outputs of h on the other values of x_i. To eliminate this information, all parties share a mask r_v for each possible value of v of x_i, where this mask is used to hide the v-instance

of the PSM protocol from P_0. The correct mask r_{x_i} is then sent by P_i in order to reveal only the correct instance to P_0.

The above technique can be generalized to add T_0-robustness to any PSM protocol, for any fixed set $T_0 \subseteq [n]$. However, such a protocol does not necessarily provide any security guarantee against other sets $T \neq T_0$. Our high level approach is to obtain t-robustness by combining T-robust protocols for all T of size at most t. While there are standard MPC-based techniques for *combining* instances of secure computation protocols of which a strict minority are faulty [23, 22, 17], in our case a majority of the instances will be insecure with respect to any fixed T_0. This makes the problem of securely combining such instances more challenging.

To describe our solution idea, we focus here on the special case of symmetric functions h with robustness threshold $t = 1$. The general case is similar. (Recall that a boolean function $h : \{0,1\}^n \to \{0,1\}$ is symmetric if there exists a function $f : \{0,\ldots,n\} \to \{0,1\}$ such that $h(x_1,\ldots,x_n) = f(\sum_{i=1}^n x_i)$.) Our protocol for h *additively* secret-shares h into random symmetric functions $h_i : \{0,1\}^n \to \{0,1\}$, $1 \leq i \leq n$, whose mod-2 sum is h. It then invokes for each i an NIMPC protocol Π_i with robustness against both $\emptyset$ and $\{i\}$, where this protocol hides (to the extent possible) not only the input x but also the function h_i. Furthermore, it is required that for each $i' \in [n] \setminus \{i\}$, a collusion of $P_{i'}$ and P_0 can only learn from their view in Π_i limited information about (h_i, x). Concretely, they should only learn the length-$(n+1)$ "truth-table" of h_i, cyclically shifted by $\sum_{j\in[n]\setminus\{i'\}} x_j$. (The truth-table of h_i contains the outputs of h_i on inputs of weight $0, 1, \ldots, n$ in this order.)

Overall, P_0 and P_i learn from the invocation of Π_i only the two outputs of h_i they are allowed to learn (corresponding to the two possible values of x_i) and no additional information on h_i and x. From the invocations of Π_j, $j \in [n] \setminus \{i\}$, they can also learn, in addition to these two entries of h_j, all other entries of the (shifted) truth-table of h_j. However, by the use of n-out-of-n secret sharing and the fact that the corresponding entries in the truth-table of h_i are hidden, these additional entries reveal no additional information on h and x.

A protocol Π_i as above is constructed in two steps. The first step is an ad-hoc construction (based on a PSM protocol for branching programs from the literature) of a PSM protocol that reveals to *any* set T only the shifted truth-table of h_i. The second step adds robustness against $T = \{i\}$, namely it blocks P_0, P_i from learning any other entry in the shifted truth-table beyond the two entries they are supposed to learn.

The following more detailed exposition of our construction consists of three steps, where the first two steps obtain the protocol Π_i for evaluating h_i and the third steps combines the protocols Π_i into a single protocol for h. We note that while the second step is general and can apply to any class of functions, both the first and the third steps rely on special properties of symmetric functions (or, more generally, abelian programs). The following three subsections describe the three steps of the construction.

4.2.1 An NIMPC for abelian programs with extended input domain

We start by describing the first step of our construction, which considers a variant of abelian programs where the players are allowed to choose their inputs from the entire group. Concretely, for the case of boolean symmetric functions on which we focus in this overview, we allow each party P_i to have an input $x_i \in \{0,\ldots,n\}$ (rather than $x_i \in \{0,1\}$), which can be thought of as an element of the group $G \triangleq \mathbb{Z}_{n+1}$. The fact that dishonest parties can learn the "shifted truth-table" of the function is captured by allowing them to pick their inputs from the entire group.

More concretely, we extend a boolean symmetric function of the form $f(\sum_{i=1}^n x_i)$ to a function $h : G^n \to \{0,1\}$ defined by $h(x_1,\ldots,x_n) = f(\sum_{i=1}^n x_i)$, where the sum is taken in G. The first step of our construction is a fully robust NIMPC protocol for the class $\mathcal{H}$ of all functions h as above, namely the group

extensions of all symmetric functions. Note that here it is crucial to hide both the function h and the inputs x_i to the extent possible, namely up to revealing the truth table of f shifted by the sum of the inputs of the honest parties. In our final protocol for symmetric functions, honest parties will invoke this protocol with inputs $x_i \in \{0,1\}$. Thus, the following description of the protocol assumes that the input of each honest party is from $\{0,1\}$.

The NIMPC protocol for $\mathcal{H}$ is based on the PSM protocol for branching programs from [24], which provides an efficient solution for symmetric functions. This protocol is defined using a *branching program* representation of h. While this protocol is secure when only P_0 is corrupted, it does not remain secure when even a single other party is corrupted. Luckily, there is a simple characterization of the information available to an adversary corrupting P_0 and a set T of other parties P_i: these players learn no more than the graph of the branching program restricted to the inputs $x_{\bar{T}}$. That is, the adversary can learn the labels of all edges it owns (e.g., that such an edge is labeled by the literal $\bar{x}_i$ or the constant 1), as well as the *values* of edges it does not own (e.g., that such an edge evaluates to 1) but not their labels. If we apply the protocol to a standard branching program for h, this information will typically reveal to the adversary both the function h and the inputs $x_{\bar{T}}$.

The key idea for realizing $\mathcal{H}$ is to randomize the branching program before applying the protocol from [24]. That is, we start with a standard layered branching program for the symmetric function h, and then (during preprocessing) we randomize it by applying a random cyclic shift to the nodes in each layer. The protocol from [24] is applied to the randomized branching program. With this randomization in place, revealing the branching program restricted by $x_{\bar{T}}$ leaks nothing about $(h, x_{\bar{T}})$ except what must be learned.

4.2.2 Limiting the inputs of one player

The previous subsection gives an NIMPC protocol for the class of (extended) symmetric functions h, with the caveat that the players may use any input in $G = \mathbb{Z}_{n+1}$, rather than just $\{0,1\}$. Let us call this protocol $\Pi_0(h)$.

As mentioned, we need to limit the parties to inputs from $\{0,1\}$. Note that for NIMPC this is relevant also in the honest-but-curious model since the robustness requirement for the extended function allows an adversary, controlling a set T, to compute $h|_{\overline{T},x_{\overline{T}}}$ on the domain $G^{|T|}$, while for the original function we only allow the adversary to compute $h|_{\overline{T},x_{\overline{T}}}$ on the domain $\{0,1\}^{|T|}$.

Example 4.6. *Consider the majority function* $\mathrm{maj} : \{0,1\}^n \to \{0,1\}$*, where* $\mathrm{maj}(x_1,\ldots,x_n) = 1$ *if and only if* $\sum_{i=1}^n x_i \geq n/2$*. In an* NIMPC *for majority,* P_1 *is allowed to learn if* $\sum_{i=2}^n x_i \geq n/2$ *and if* $1 + \sum_{i=2}^n x_i \geq n/2$*. If we extend the domain to* $\mathbb{Z}_{n+1}$*, party* P_1 *is allowed to learn if* $a + \sum_{i=2}^n x_i \geq n/2$ *for every* $a \in \mathbb{Z}_{n+1}$*, that is,* P_1 *can learn* $\sum_{i=2}^n x_i$*.*

In this section, as an intermediate step, we construct a protocol where a specific player, say P_1, is limited to inputs in $\{0,1\}$. The other players, $P_2,\ldots,P_n$, can still choose any inputs in G. Let h_0 and h_1 denote the function h where the first input is fixed to 0 and 1, respectively, that is, $h_i(X_2,\ldots,X_n) \triangleq h(i,X_2,\ldots,X_n)$, for $i \in \{0,1\}$. Consider the following protocol: $P_2,\ldots,P_n$ run the protocols $\Pi_0(h_0)$ and $\Pi_0(h_1)$. At the end of this protocol, the coalition $\{P_0,P_1\}$ – seeing the messages of $\Pi_0(h_0)$ and $\Pi_0(h_1)$ – knows *exactly* what it is supposed to know: the values $h(0,x_2,\ldots,x_n)$ and $h(1,x_2,\ldots,x_n)$. However, there are two evident problems.

1. On one hand, P_0 alone knows "too much": the same two values $h(0,x_2,\ldots,x_n)$ and $h(1,x_2,\ldots,x_n)$.
2. On the other hand, P_0 does not know which of these two values is the correct one, i.e., $h(x_1,\ldots,x_n)$.

A possible "solution" to the second problem is for P_1 to send its input x_1 to P_0. This is, of course, insecure. Instead, we run $\Pi_0(h_0)$ and $\Pi_0(h_1)$ in a random order, known only to P_1 and given to it in the preprocessing stage (note that $P_2, \ldots, P_n$ need not know which of the two protocols is running to participate). Party P_1 will then send a message stating which one corresponds to its input.

A solution to the first problem is as follows: The (symmetric) functions h_0 and h_1 (which can be though of as $(n+1)$-bit strings representing their truth tables) are "masked" by ($(n+1)$-bit) random functions r_0 and r_1 (where $r_b : G \to \{0,1\}$). Let us call these masked versions g_0 and g_1. Specifically, $g_j(X_2, \ldots, X_n) \triangleq h_j(X_2, \ldots, X_n) \oplus r_j(\sum_{i=2}^n X_i)$, for $j \in \{0,1\}$. In the preprocessing stage, we give the masking functions r_0 and r_1 to P_1. Now $P_0, P_2, \ldots, P_n$ run $\Pi_0(g_0)$ and $\Pi_0(g_1)$ (in a random order). Then, P_1 sends to P_0 only the masking r_i corresponding to its input. In terms of security, the problem has been solved: the protocol not corresponding to P_1's input, i.e., $\Pi_0(g_{1-x_1})$, does not reveal any information to P_0, as g_{1-x_1} is a masked version of h, where the mask has not been revealed. However, can P_0 now compute $h(x_1, \ldots, x_n)$? From seeing the messages of $\Pi_0(g_{x_1})$, it knows $g_{x_1}(x_2, \ldots, x_n) = h(x_1, \ldots, x_n) \oplus r_{x_1}(\sum_{i=2}^n x_i)$. It also knows r_{x_1}, which was sent by P_1. So now, to "unmask" $h(x_1, \ldots, x_n)$ using r_{x_1} it needs the value $\sum_{i=2}^n x_i$, which is more information than we want to give it.

Further randomization techniques are needed to solve this problem, combined with the solutions to the two problems above. Specifically, let $h(x_1, \ldots, x_n) = f(\sum_{i=1}^n x_i)$ be the function we want to compute. In the preprocessing stage we choose a random element $s \in G$ as well as two random strings r_0, r_1 and a bit ρ. The strings r_0, r_1 and the bit ρ are given to P_1. We next define two masked functions $g_j(X_2, \ldots, X_n) \triangleq f((j \oplus \rho) + \sum_{i=2}^n X_i) \oplus r_{j \oplus \rho}(s + \sum_{i=2}^n X_i)$ for $j \in \{0,1\}$. Notice that g_0 and g_1 are abelian programs. We next execute three NIMPCs with $P_2, \ldots, P_n$: the NIMPC $\Pi_0(g_0)$, the NIMPC $\Pi_0(g_1)$, and an NIMPC for $s + \sum_{i=2}^n X_i$ (this shifted sum gives no information to P_0). In addition, P_1 sends $z = \rho \oplus x_1$ and r_{x_i} to P_0. Party P_0 now decodes

$$\begin{aligned} g_z(x_2, \ldots, x_n) &= f\left(((\rho \oplus x_1) \oplus \rho) + \sum_{i=2}^n x_i\right) \oplus r_{(\rho \oplus x_1) \oplus \rho}(s + \sum_{i=2}^n x_i) \\ &= h(x_1, \ldots, x_n) \oplus r_{x_1}(s + \sum_{i=2}^n x_i). \end{aligned}$$

Party P_0, which knows r_{x_1} and $s + \sum_{i=2}^n x_i$, can, therefore, reconstruct $h(x_1, \ldots, x_n)$.

4.2.3 A combiner using secret sharing

The previous section described a protocol where, for a certain fixed $j \in [n]$, the coalition $\{P_0, P_j\}$ does not learn "too much" – specifically, it could evaluate the function h only on inputs in $\{0,1\}$ (while a coalition of P_0 with one of the other players is still not restricted to inputs in $\{0,1\}$). Call this protocol Π_1. Note that h is of the form $h(X_1, \ldots, X_n) = f(\sum_{i=1}^n X_i)$ for a function $f : G \to \{0,1\}$. We now bootstrap the protocol Π_1 to create one in which *all* players can evaluate h only on inputs in $\{0,1\}$. For this, we use an additive secret sharing of f. Namely, we choose n random functions $f_1, \ldots, f_n : G \to \{0,1\}$, such that $\sum_{i=1}^n f_i = f$, where the sum is a xor of $|G|$-bit vectors. For $1 \le i \le n$, define $h_i(X_1, \ldots, X_n) \triangleq f_i(\sum_{j=1}^n X_j)$. Note that for any $x_1, \ldots, x_n \in G$, we have $h(x_1, \ldots, x_n) = \sum_{i=1}^n h_i(x_1, \ldots, x_n)$. For $1 \le i \le n$, we run Π_1 on the function h_i with P_i chosen to be the player that can only use inputs in $\{0,1\}$. After these protocols are run we have that, on the one hand, P_0 knows $h_1(x_1, \ldots, x_n), \ldots, h_n(x_1, \ldots, x_n)$ and can compute $h(x_1, \ldots, x_n) = \sum_{i=1}^n h_i(x_1, \ldots, x_n)$. On the other hand, for any $i \in [n]$ and $a \in G \setminus \{0,1\}$, parties P_0 and P_i have no information on $h_i(x_1, \ldots, x_{i-1}, a, x_{i+1}, \ldots, x_n)$ and hence no information on $h(x_1, \ldots, x_{i-1}, a, x_{i+1}, \ldots, x_n)$. Thus, they can compute $h|_{\overline{\{1\}}, x_2, \ldots, x_n}$ only on 0 and 1 as required.

4.3 The construction and its proof

In the rest of this section we prove our main positive result – Theorem 4.3. We begin by introducing some notation that will be convenient for the proof.

Notation 4.7. *Fix $x = (x_1, \ldots, x_n) \in S_1 \times \cdots \times S_n$. We denote $\mathrm{sum}(x) \triangleq \sum_{i=1}^{n} x_i$, and for a subset $T \subseteq [n]$ we denote $\mathrm{sum}(x_T) \triangleq \sum_{i \in T} x_i$ (if $T = \emptyset$, then $\mathrm{sum}(x_T) \triangleq 0$). We also use the notation $S_T \triangleq \prod_{i \in T} S_i$.*

The proof of Theorem 4.3 follows the outline described in Section 4.2, and is divided into subsections in a similar way. Section 4.3.1 describes an NIMPC for the case that $S_i = G$ for every i. Section 4.3.2 describes an NIMPC where a fixed subset of a players is limited to smaller input domains. Finally, Section 4.3.3 gives the desired protocol for proving Theorem 4.3, i.e., an NIMPC where every subset of players of bounded size is limited to small input domains.

4.3.1 An NIMPC for abelian programs with extended input domain

We next describe an NIMPC for the case that $S_i = G$ for every i (recall that in the general case $S_i \subseteq G$). This NIMPC uses the PSM protocol of [24], which, in turn, uses branching programs.

Definition 4.8. *A deterministic branching program B over a group G on n inputs is a directed acyclic graph, containing a source vertex s and a target vertex t. Each edge has a label of the form $X_i = a$ for some $i \in [n]$ and $a \in G$. For any vertex v there exits a unique $i \in [n]$ such that every outgoing edge from v is labeled $X_i = a$ for some $a \in G$, and, furthermore, for $a \in G$, there is at most one outgoing edge from v labeled $X_i = a$. For an input $x = (x_1, \ldots, x_n)$, let B_x be the subgraph of B containing precisely the edges labeled $X_i = x_i$ for all $i \in [n]$. The branching program B computes a function $f : G^n \to \{0, 1\}$, where $f(x_1, \ldots, x_n) = 1$ if and only if $B_{(x_1,\ldots,x_n)}$ contains a path from s to t.*

We assume, w.l.o.g., that s has no incoming edges and t has no outgoing edges, and fix a topological order on the vertices of the branching program, where s is the first and t is the last. We represent the branching program by $n \cdot |G|$ matrices, where for every $i \in [n]$ and $a \in G$, we define ${B'}_{i,a}$ to be the adjacency matrix of the subgraph of B containing only the edges labeled $X_i = a$, that is, we label the rows and columns of the matrix by the vertices of the branching program according to the fixed topological order and entry i, j of the matrix is one if there is an edges from v_i to v_j labeled by $X_i = a$ and the entry is zero otherwise. For every $i \in [n-1]$ and $a \in G$ we define $B_{i,a}$ as the matrix obtained from ${B'}_{i,a}$ by removing the first column (labeled by s) and the last row (labeled by t). For every $a \in G$, we define $B_{n,a}$ as the matrix obtained from ${B'}_{n,a} - I$ by removing the first column and the last row.

The "strange" definition of $B_{n,a}$ follows from its use in [24]. Specifically, using this notation, for every input $x_1, \ldots, x_n$: if $f(x_1, \ldots, x_n) = 1$ then the matrix $\sum_{i=1}^{n} B_{i,x_i}$ has full rank, and otherwise the matrix has rank $N - 1$ (where the matrix is an $N \times N$ matrix). The following theorem follows directly from [24].

Theorem 4.9 (Ishai and Kushilevitz [24])**.** *Fix an abelian group G, and an integer n. Let $\mathcal{H}$ be the set of functions computable by a branching program over G on n inputs with $N + 1$ vertices. The* NIMPC *$\Pi_{IK} = (\mathrm{Gen}_{IK}, \mathrm{Enc}_{IK}, \mathrm{Dec}_{IK})$ described in Figure 1 is a 0-robust* NIMPC *of $\mathcal{H}$ with communication complexity $O(|G| \cdot N^2)$.*

The NIMPC Π_{IK} is only 0-robust as, for example, R_i contains information on the branching program. For branching programs for symmetric functions, we will show that randomizing the branching program and applying Π_{IK} achieves *full* robustness, i.e., robustness against any set. Recall that we are interested in

Correlated randomness generation:

- Choose random non-singular $N \times N$ matrices P and U over $\mathbb{F}_2$.
- Choose random $N \times N$ matrices $Z_1, \ldots, Z_n$ over $\mathbb{F}_2$ such that $\sum_{i=1}^{n} Z_i = 0$.
- $\mathrm{Gen}_{IK}(B) = R = (R_1, \ldots, R_n)$, where $R_i = (P, U, Z_i, \{B_{i,a}\}_{a \in G})$ for every $i \in [n]$.

Encoding:

- For $i \in [n]$ and $a \in G$, we consider the $N \times N$ matrix $B_{i,a}$ as a matrix over $\mathbb{F}_2$. Let $\mathrm{Enc}_{IK}(x, R) = (M_1, \ldots, M_n)$, where for $i \in [n]$, the encoding M_i is the $N \times N$ matrix $P \cdot B_{i,x_i} \cdot U + Z_i$.

Decoding:

- $\mathrm{Dec}_{IK}(M_1, \ldots, M_n) = 1$ if and if $\sum_{i=1}^{n} M_i$ does not have a full rank.

Figure 1: The 0-robust NIMPC $\Pi_{IK}(B)$ of [24] computing a branching program B.

functions $h : G^n \to \{0,1\}$ of the form $h(x_1, \ldots, x_n) = f(\sum_{i=1}^{n} x_i) = f(\mathrm{sum}(x))$ for some $f : G \to \{0,1\}$. We use a "randomization" of a branching program B computing such a function h. For this, the following definition and claim are useful.

Definition 4.10. *Fix an abelian group G. For a function $f : G \to \{0,1\}$, and elements $r_1, \ldots, r_{n-1} \in G$, we define a branching program $B_f^{r_1,\ldots,r_{n-1}}$ as follows.*

- *The program contains $n+1$ layers $\{0, \ldots, n\}$.*
- *For $0 \leq i \leq n-1$, the i'th layer contains $|G|$ vertices that we identify with the elements of G. The vertex 0 (identified with the zero of G) in the 0'th layer is the start vertex s. The n'th layer contains only the end vertex t.*
- *There are edges only between adjacent layers:*
  1. *For $1 \leq i \leq n-1$, and $a, b \in G$, the vertex b in the $(i-1)$'th layer is connected to vertex $b + a + r_i$ in the $i'th$ layer with an edge labeled $X_i = a$.*
  2. *Define $r_n \triangleq -\sum_{i=1}^{n-1} r_i$. For $a, b \in G$, if $f(b + a + r_n) = 1$, then the vertex b in the $(n-1)$'th layer is connected to the end vertex t with an edge labeled $X_n = a$.*

The "natural" branching program for $h(X_1, \ldots, X_n) = f(\sum_{i=1}^{n} X_n)$ is obtained by taking $r_1 = \cdots = r_n = 0$ in the branching program described in Definition 4.10. Roughly speaking, this natural branching program first computes $\sum_{i=1}^{n} X_n$, and based on this sum decides to accept or reject. Taking arbitrary $r_1, \ldots, r_{n-1}$ does not affect the correctness of the branching program, and, as we will see below, if we choose them at random then we obtain n-robustness. We summarize some properties of the branching program in the following claim.

Correlated randomness generation:

- Choose random elements $r_1, \ldots, r_{n-1} \in G$. Let $r_n \triangleq -\sum_{i=1}^{n-1} r_i$. Now, define $\mathrm{Gen}_0(h) = R = (R_1, \ldots, R_n) \triangleq \mathrm{Gen}_{IK}(B = B_f^{r_1,\ldots,r_{n-1}})$. Note that R_i consists of the $N \times N$ matrices P, U, Z_i and the matrices $\{B_{i,a}\}_{a \in G}$ (as defined in Definition 4.8). Here $N + 1 = O(n \cdot |G|)$ is the number of vertices in B.

Encoding:

- Define $\mathrm{Enc}_{0,i}(x_i, R_i) = M_i$, where $M_i = \mathrm{Enc}_{IK,i}(x_i, R_i) = P \cdot B_{i,x_i} \cdot U + Z_i$ for $i \in [n]$.

Decoding h:

- $\mathrm{Dec}_0(M_1, \ldots, M_n) = \mathrm{Dec}_{IK}(M_1, \ldots, M_n)$.

Figure 2: The NIMPC Π_0 for functions $h : G^n \to \{0,1\}$ of the form $h(X_1, \ldots, X_n) = f(\sum_{i=1}^n X_i)$.

Claim 4.11. *Fix any $f : G \to \{0,1\}$, and elements $r_1, \ldots, r_{n-1} \in G$, and let $r_n \triangleq -\sum_{i=1}^{n-1} r_i$ and $B \triangleq B_f^{r_1,\ldots,r_{n-1}}$.*

- *B computes the function $h(X_1, \ldots, X_n) = f(\mathrm{sum}(X))$.*
- *For $1 \le i \le n-1$ and $a \in G$, the matrix (i.e., the subgraph) $B_{i,a}$ is determined by $a + r_i$ and is independent of f. For $i = n$ and $a \in G$, the matrix $B_{n,a}$ is determined by the shifted function $f'(j) \triangleq f(j + r_n)$.*

Proof. For the first item, note that $B_{(x_1,\ldots,x_n)}$ contains a path from s to the vertex $\sum_{i=1}^{n-1}(x_i + r_i)$ in the $(n-1)$'th level, and this is the only vertex in the $(n-1)$'th level that is reachable from s. There is an edge from vertex $\sum_{i=1}^{n-1}(x_i + r_i)$ in the $(n-1)$'th level to t iff

$$1 = f\left(\sum_{i=1}^{n-1}(x_i + r_i) + x_n + r_n\right) = f\left(\sum_{i=1}^{n} x_i\right)$$

(the latter equality follows from the fact that $\sum_{i=1}^n r_i = 0$). Thus, $f(\sum_{i=1}^n x_i) = 1$ iff there is a path from s to t in $B_{(x_1,\ldots,x_n)}$.

To see that the second item in the claim holds, note that for $1 \le i \le n-1$, the sum $a + r_i$ determines all edges between the $(i-1)$'th and i'th layer labeled $X_i = a$. Similarly, there is an edge labeled $X_n = a$ from vertex b in the $(n-1)$'th layer to t, iff $f(a + b + r_n) = f'(a + b) = 1$. □

Theorem 4.12. *The* NIMPC $\Pi_0 = (\mathrm{Gen}_0, \mathrm{Enc}_0, \mathrm{Dec}_0)$ *described in Figure 2 is a fully robust* NIMPC *for $\mathcal{H}^G_{G,\ldots,G}$ with communication complexity $O(|G|^2 \cdot n^2)$.*

Proof. For the correctness of the NIMPC Π_0, note that, by Claim 4.11, the branching program $B_f^{r_1,\ldots,r_{n-1}}$ computes $h(X_1, \ldots, X_n) = f(\mathrm{sum}(X))$. Thus, by Theorem 4.9,

$$\mathrm{Dec}_0(M_1, \ldots, M_n) = \mathrm{Dec}_{IK}(M_1, \ldots, M_n) = f(\mathrm{sum}(X)).$$

- Fix some $i_0 \in T$.
- Define a function $f' : G \to \{0,1\}$ as follows. For every $a \in G$ do:
  1. Define x_T, where $x_{i_0} = a$ and $x_i = 0$ for every $i \in T \setminus \{i_0\}$.
  2. Let $f'(a) = h|_{\overline{T},x_{\overline{T}}}(x_T)$.

  $(* \quad f'(a) = f(\mathrm{sum}(x_{\overline{T}}) + a) \quad *)$
- Let $h'(X_1, \ldots, X_n) = f'(\sum_{i=1}^n X_i)$.
- Let $R_1, \ldots, R_n = \mathrm{Gen}_0(h')$.
- For every $i \in \overline{T}$, let $M_i = \mathrm{Enc}_0(0, R_i)$.
- Return $\{M_i\}_{i\in\overline{T}}, \{R_i\}_{i\in T}$.

Figure 3: The simulator for the NIMPC Π_0 for a set $T \neq \emptyset$.

The 0-robustness of Π_0 follows from the 0-robustness of Π_{IK}. We next prove the T-robustness of Π_0 for a non-empty subset T, that is, we construct a simulator $\mathrm{Sim}_T(h|_{\overline{T},x_{\overline{T}}})$ whose output distribution is as required. For the construction, we first identify which functions h, h' and inputs $x_{\overline{T}}$ satisfy $h|_{\overline{T},x_{\overline{T}}}(X_T) = h'|_{\overline{T},0_{\overline{T}}}(X_T)$, that is, when the simulator gets the same answers to its queries. For such pairs $h, x_{\overline{T}}$ and $h', 0_{\overline{T}}$, we show that the messages of $\overline{T}$ and the correlated randomness of T are equally distributed in Π_0. Thus, the simulator will find h' such that $h|_{\overline{T},x_{\overline{T}}} = h'|_{\overline{T},0_{\overline{T}}}$ and execute Π_0 on h' and $x_{\overline{T}} = 0_{\overline{T}}$.

Observation 4.13. *Let $h(X_1, \ldots, X_n) = f(\sum_{i=1}^n X_i)$ and $h'(X_1, \ldots, X_n) = f'(\sum_{i=1}^n X_i)$. Then,*

1. *The restricted function $h|_{\overline{T},x_{\overline{T}}}$ is equivalent to the function f restricted to inputs in $\mathcal{X}_T$ and shifted by $s \triangleq \mathrm{sum}(x_{\overline{T}})$, that is, $h|_{\overline{T},x_{\overline{T}}}(X_T) = f(s + \mathrm{sum}(X_T))$.*
2. *Fix a set T and inputs $x_{\overline{T}} \in G^{|\overline{T}|}$. Then, $h|_{\overline{T},x_{\overline{T}}}(X_T) = h'|_{\overline{T},0_{\overline{T}}}(X_T)$ iff $f(a) = f'(a - \mathrm{sum}(x_{\overline{T}}))$ for every $a \in G$, that is, f' is a shift of f by $\mathrm{sum}(x_{\overline{T}})$.*

The simulator for the NIMPC Π_0 is described in Figure 3. Let $h(X) = f(\mathrm{sum}(X))$ and $X_{\overline{T}}$ be the function and the inputs of $\overline{T}$ in the execution of the simulator. The simulator, which has only access to $h|_{\overline{T},x_{\overline{T}}}$, chooses inputs $x'_{\overline{T}} = 0$ and an h' such that, by Observation 4.13, $h|_{\overline{T},x_{\overline{T}}}(X_T) = h'|_{\overline{T},0_{\overline{T}}}(X_T)$. It then executes the randomness generation of Π_0 with h' and the encoding in Π_0 with $x'_{\overline{T}} = 0$. We next prove that the messages of $\overline{T}$ and the randomness of T in the execution of Π_0 on h and $x_{\overline{T}}$ are identically distributed as the output distribution of the simulator, namely, as the messages of $\overline{T}$ and the randomness of T in the execution of Π_0 on h' and $0_{\overline{T}}$ (we will refer to these executions as the first and the second executions). The randomness in both distributions is only the randomness of Gen_0.

To show the equivalence of the distributions, we show a bijection between the random strings used by Gen_0 on h and the random strings used by Gen_0 on h' such that the messages of $\overline{T}$ and the joint randomness

of T are the same under this bijection. The randomness of Gen_0 is composed of $r_1, \ldots, r_{n-1}$ used to construct the branching program and the matrices $P, U, Z_1, \ldots, Z_n$ used by $\mathrm{Gen}_{\Pi_{IK}}$. The bijection maps this randomness to $r'_1, \ldots, r'_{n-1}, P, U, Z_1, \ldots, Z_n$, where

$$r'_i = \begin{cases} r_i & \text{iff } i \in T, \\ x_i + r_i & \text{iff } i \in \overline{T}. \end{cases}$$

Note that for every $i \in T \cap [n-1]$, the joint randomness R_i is the same in both executions of Gen_0 as R_i only depends on P, U, Z_i, r_i which are the same in both executions. We next show that for every $i \in \overline{T} \cap [n-1]$ the encoding M_i is the same in both executions:

- In the first execution, $M_i = \mathrm{Enc}_{0,i}(x_i, r_i) = P \cdot B_{i,x_i} \cdot U + Z_i$, where B_{i,x_i} contains edges between every vertex b in the $(i-1)$'th level of the branching program to vertex $b + x_i + r_i$ in the i'th level.
- In the second execution, $M'_i = \mathrm{Enc}_{0,i}(0, r'_i) = P \cdot B'_{i,0} \cdot U + Z_i$, where $B'_{i,0}$ contains edges between every vertex b in the $(i-1)$'th level of the branching program to vertex $b + 0 + r'_i = b + x_i + r_i$ in the i'th level.

Thus, $B_{i,x_i} = B'_{i,0}$ and the encodings M_i, M'_i are equal.

To complete the proof of the T-robustness, we need to consider two cases: $n \in T$ and $n \in \overline{T}$. In the first case, we need to show that R_n is equal in the two executions, that is, $B_{n,a} = B'_{n,a}$ for every $a \in G$.

- In the first execution with randomness r_n and f, there exists an edge labeled by $X_n = a$ between a vertex b in the $(n-1)$'th level of the branching program to vertex t iff $f(b + a + r_n) = 1$, where $r_n = -\sum_{i=1}^{n-1} r_i$.
- In the second execution with randomness r'_n and f', there exists an edge labeled by $X_n = a$ between a vertex b in the $(n-1)$'th level of the branching program to vertex t iff $f'(b + a + r'_n) = 1$, where

$$r'_n = -\sum_{i=1}^{n-1} r'_i = -\sum_{i=1}^{n-1} r_i - \sum_{i \in \overline{T}} x_i = r_n - \mathrm{sum}(x_{\overline{T}}).$$

  Recall that $f'(a) = f(a + \mathrm{sum}(x_{\overline{T}}))$, thus, $f'(b + a + r'_n) = f(b + a + r'_n + \mathrm{sum}(x_T)) = f(b + a + r_n)$.

Thus, in this case the joint randomness R_n is the same in both executions.

In the second case (where $n \in \overline{T}$), we need to show that M_n is equal in the two executions, that is, B_{n,x_n} is equal $B'_{n,0}$. This follNws by similar arguments as above, noting that in this case $r'_n = r_n - \mathrm{sum}(x_{\overline{T}}) + x_n$.

Next we analyze the communication complexity of the NIMPC Π_0. This NIMPC uses the NIMPC Π_{IK} with a branching program of size $N = O(n|G|)$, thus, by Theorem 4.9, the communication complexity of Π_0 is $O(|G|N^2) = O(|G|^3 n^2)$. However, using Claim 4.11, we can optimize the communication complexity by a slightly better implementation: the string R_i, for $1 \le i \le n-1$ contains U, P, Z_i, and r_i (instead of $\{U \cdot B_{i,a} \cdot P + Z_i\}$ for every $a \in G$). Similarly, R_n contains the U, P, Z_i, and function f' defined in Claim 4.11. Notice that in the proof of T-robustness we allow R_i to contain this additional information. After these changes, each encoding and correlated random string contains only a *constant* number of $N \times N$ matrices, rather than $O(|G|)$ such matrices. Hence, the communication complexity is reduced to $O(N^2) = O(|G|^2 n^2)$. □

4.3.2 Limiting the inputs of one set

In this section we limit the inputs of one (special) subset of variables. We start with an NIMPC for the family of shifted sums. We will use this NIMPC in our constructions.

Lemma 4.14. *Fix an abelian group G and an integer n. For $s \in G$, let $h_s : G^n \to G$ be the function $h_s(X_1, \ldots, X_n) \triangleq \sum_{i=1}^{n} X_i + s = \mathrm{sum}(X) + s$, and let $\Delta_n \triangleq \{h_s\}_{s \in G}$. There is a fully robust* NIMPC $\Pi_{\Delta_n} = (\mathrm{Gen}_{\Delta_n}, \mathrm{Enc}_{\Delta_n}, \mathrm{Dec}_{\Delta_n})$ *of Δ_n with communication complexity* $\log |G|$.

The NIMPC Π_{Δ_n} for Δ_n is a generalization of the NIMPC for the sum function described in Example 2.7, where in Gen_{Δ_n}, we compute $r_n = s - \sum_{i=1}^{n-1} r_i$.

In Figure 4, we describe the NIMPC Π_1, in which the inputs of one set are limited. This NIMPC follows the intuition presented in Section 4.2.2, where some adaptations are needed as (1) the NIMPC presented here is t-robust (compared to 1-robust in Section 4.2.2), and (2) we limit the inputs to sets S_i, which are not binary.

In Π_1, it will be convenient to denote the special set, whose inputs are limited, by $\{1, \ldots, t\}$ and the corresponding variables by $Y_1, \ldots, Y_t$. Furthermore, we denote the rest of the inputs by $X_{t+1}, \ldots, X_n$. For $i \in [t]$, we denote the set of legal inputs of Y_i by $S_i \subseteq G$. We assume for simplicity that $0 \in S_i$ for every i. As in the previous section, the set of legal inputs of any variable X_i is G.

Theorem 4.15. *The* NIMPC $\Pi_1^{S_1,\ldots,S_t}$ *described in Figure 4 is a fully robust* NIMPC *for $\mathcal{H}^G_{S_1,\ldots,S_t,G,\ldots,G}$. The communication complexity of $\Pi_1^{S_1,\ldots,S_t}$ is $O(|G| \cdot t \cdot d^{t+2} + |G|^3 \cdot n^2) = O(d^{t+2} \cdot n^2 \cdot |G|^3)$, where $d \triangleq \max_{i \in [t]} |S_i|$.*

Proof. The correctness of the decoding follows from (1). The communication complexity follows since in the NIMPC Π_1 there are:

- One execution of the NIMPC Π_Γ for the function ℓ with complexity $O(|G| \cdot t \cdot d^{t+2})$,
- Two executions of the NIMPC for shifted sum with complexity $O(\log |G|)$, and
- $|G|$ executions of the NIMPC Π_0 for g_b, each with complexity $O(|G|^2 n^2)$.

We next prove the robustness. In Figure 5, we describe a simulator for a subset $T = T_1 \cup T_2$, where $T_1 \subseteq \{1, \ldots, t\}$ and $T_2 \subseteq \{t+1, \ldots, n\}$ (where these sets can be empty), and the inputs of $\overline{T}$ are $y_{\overline{T}_1} \in S_{\overline{T}_1}$ and $x_{\overline{T}_2} \in G^{|\overline{T}_2|}$. The simulator for Π_1 uses the simulators for the various NIMPC protocols used in Π_1. The important task in constructing the simulator is how to answer the queries made by the simulators of these NIMPC protocols.

Inspection of Figure 5 shows that up to Step 5 the simulator only requires random values it generates by itself. For Step 5, it requires the values

$$f(a + \mathrm{sum}(y_{\overline{T}_1}) + \mathrm{sum}(x_{\overline{T}_2})),$$

where $a \in A_{T_1} = \{a : \exists y_{T_1} \in S_{T_1} \text{ s.t. } \mathrm{sum}(y_{T_1}) = a\}$ if $T_2 = \emptyset$, and $a \in G$ otherwise. In the first case, it is clear that it knows these values given the restricted function $h|_{\overline{T}, x_{\overline{T}}}(Y_{T_1}) = f(\mathrm{sum}(Y_{T_1}) + \mathrm{sum}(y_{\overline{T}_1}) + \mathrm{sum}(x_{\overline{T}_2}))$. This is also clear in the second case recalling that an input belonging to T_2 can range over all of G.

Correlated randomness generation:

1. Choose random $\rho, s \in G$ and for all $b \in G$ choose a random string $r_b \in \{0,1\}^{|G|}$ (we think of r_b as a function from G to $\{0,1\}$).
2. Define the function $\ell : S_1 \times \cdots \times S_t \to \{0,1\}^{|G|}$ by $\ell(a_1, \ldots, a_t) = r_{\sum_{i=1}^{t} a_i}$.
3. For every $a \in G$, define the function $g_{a+\rho} : G^{n-t} \to \{0,1\}$ by

$$g_{a+\rho}(X_{t+1}, \ldots, X_n) \triangleq f(a + \sum_{i=t+1}^{n} X_i) \oplus r_a(s + \sum_{i=t+1}^{n} X_i).$$

4. Let $R^\rho = (R_1^\rho, \ldots, R_t^\rho) \triangleq \mathrm{Gen}_{\Delta_t}(h_\rho)$ (where $h_\rho(Y_1, \ldots, Y_t) = \rho + \sum_{i=1}^{t} Y_i$).
5. Let $R^\ell = (R_1^\ell, \ldots, R_t^\ell) \triangleq \mathrm{Gen}_\Gamma(\ell)$ (where Gen_Γ is defined in Theorem 3.3).
6. Let $R^{\mathrm{s}} = (R_{t+1}^{\mathrm{s}}, \ldots, R_n^{\mathrm{s}}) \triangleq \mathrm{Gen}_{\Delta_{n-t}}(h_s)$ (where $h_s(X_{t+1}, \ldots, X_n) = s + \sum_{i=t+1}^{n} X_i$).
7. For $b \in G$, let $R^b = (R_{t+1}^b, \ldots, R_n^b) \triangleq \mathrm{Gen}_0(g_b)$.
8. Define $\mathrm{Gen}_1(h) = (R_1, \ldots, R_n)$, where $R_i = (R_i^\rho, R_i^\ell)$ if $i \in \{1, \ldots, t\}$ and $R_i = (R_i^{\mathrm{s}}, (R_i^b)_{b \in G})$ if $i \in \{t+1, \ldots, n\}$.

Encoding:

9. For $i \in \{1, \ldots, t\}$: Let $M_i^\rho = \mathrm{Enc}_{\Delta_t, i}(y_i, R_i^\rho)$ and $M_i^\ell = \mathrm{Enc}_{\Gamma, i}(y_i, R_i^\ell)$, and let $\mathrm{Enc}_1(y_i, R_i) = M_i = (M_i^\rho, M_i^\ell)$.
10. For $i \in \{t+1, \ldots, n\}$: Let $M_i^{\mathrm{s}} = \mathrm{Enc}_{\Delta_n, i}(x_i, R_i^{\mathrm{s}})$ and $M_i^b = \mathrm{Enc}_{0,i}(x_i, R_i^b)$ for every $b \in G$, and let $\mathrm{Enc}_{1,i}(x_i, R_i) = M_i = (M_i^\rho, (M_i^b)_{b \in G})$.

Decoding h :

11. $z = \mathrm{Dec}_{\Delta_t}(M_1^\rho, \ldots, M_t^\rho) \quad (* \quad z = \mathrm{sum}(y) + \rho \quad *)$
12. $u = \mathrm{Dec}_{\Delta_{n-t}}(M_{t+1}^{\mathrm{s}}, \ldots, M_n^{\mathrm{s}}) \quad (* \quad u = \mathrm{sum}(x) + s \quad *)$
13. $v = \mathrm{Dec}_\Gamma(M_1^\ell, \ldots, M_t^\ell) \quad (* \quad v = \ell(y_1, \ldots, y_t) = r_{\mathrm{sum}(y)} \quad *)$
14. $w = \mathrm{Dec}_0(M_{t+1}^z, \ldots, M_n^z)$

$$\begin{aligned} (* \quad w &= g_{\mathrm{sum}(y)+\rho}(x_{t+1}, \ldots, x_n) \\ &= f(\mathrm{sum}(y) + \mathrm{sum}(x)) \oplus r_{\mathrm{sum}(y)}(s + \mathrm{sum}(x)) \\ &= h(y_1, \ldots, y_t, x_{t+1}, \ldots, x_n) \oplus v(u) \quad *) \end{aligned} \tag{1}$$

15. The output is: $w \oplus v(u)$.

Figure 4: The NIMPC $\Pi_1^{(S_1,\ldots,S_t)}(h)$, where the inputs of the subset $\{1, \ldots, t\}$ are limited.

1. Define $A_{T_1} \triangleq \{a : \exists y_{T_1} \in S_{T_1} \text{ s.t. } \mathrm{sum}(y_{T_1}) = a\}$.
2. Simulating $(R^{\rho}_{T_1}, M^{\rho}_{\overline{T}_1})$: Choose a random element $\rho' \in G$ and execute the simulator for Π_{Δ_t} for the set T_1 with the restricted function $h'(Y_{T_1}) \triangleq \rho' + \mathrm{sum}(Y_{T_1})$.
3. Simulating $(R^{\ell}_{T_1}, M^{\ell}_{\overline{T}_1})$: Choose random strings $(r'_a)_{a \in A_{T_1}}$ (where $r'_a \in \{0,1\}^{|G|}$) and execute the simulator for $\Pi_{\Gamma}(\ell)$ for the set T_1 with restricted function $\ell'(Y_{T_1}) \triangleq r'_{\mathrm{sum}(Y_{T_1})}$ for $y_{T_1} \in S_{T_1}$ (in particular, $\mathrm{sum}(y_{T_1}) \in A_{T_1}$).
4. Simulating $(R^{\mathrm{s}}_{T_2}, M^{\mathrm{s}}_{\overline{T}_2})$: Choose a random element $s' \in G$ and execute the simulator for $\Pi_{\Delta_{n-t}}$ for the set T_2 with the restricted function $h'_1(X_{T_2}) \triangleq s' + \mathrm{sum}(X_{T_2})$.
5. Simulating $(R^{b}_{T_2}, M^{b}_{\overline{T}_2})$: Let $a = b - \rho'$. Execute the simulator for $\Pi_0(g_b)$ for the set T_2, with the restricted function g'_b defined as follows:
   (a) If $a \in A_{T_1}$: Then
   $$g'_b(X_{T_2}) \triangleq f\left(a + \mathrm{sum}(y_{\overline{T}_1}) + \mathrm{sum}(x_{\overline{T}_2}) + \mathrm{sum}(X_{T_2})\right) \oplus r'_a(s' + \mathrm{sum}(X_{T_2})).$$
   (b) If $a \notin A_{T_1}$: Choose a random string $r'_a \in \{0,1\}^{|G|}$ and let
   $$g'_b(x_{T_2}) \triangleq r'_a(\mathrm{sum}(x_{T_2})).$$

Figure 5: The simulator for the NIMPC Π_1 for $T = T_1 \cup T_2$, where $T_1 \subseteq \{1, \ldots, t\}$ and $T_2 \subseteq \{t+1, \ldots, n\}$.

We consider an execution of the NIMPC with $\rho, s, \{r_a\}_{a \in G}$ and an execution of the simulator with $\rho', s', \{r'_a\}_{a \in G}$, where:

$$\begin{aligned}
\rho' &= \rho + \mathrm{sum}(y_{\overline{T}_1}), && (2)\\
r'_a &= r_{a+\mathrm{sum}(y_{\overline{T}_1})} \text{ for every } a \in A_{T_1}, && (3)\\
s' &= s + \mathrm{sum}(x_{\overline{T}_2}), && (4)\\
r'_a(c) &= f(a + \mathrm{sum}(y_{\overline{T}_1}) + \mathrm{sum}(x_{\overline{T}_2}) + c) \oplus r_{a+\mathrm{sum}(y_{\overline{T}_1})}(s + \mathrm{sum}(x_{\overline{T}_2}) + c) \\
&\quad \text{for every } a \notin A_{T_1} \text{ and every } c \in G. && (5)
\end{aligned}$$

We show below that, given any fixing of the values $\rho, s, \{r_a\}_{a \in G}$,$\rho', s', \{r'_a\}_{a \in G}$ as above, the protocol and simulation generate exactly the same distribution of the relevant random variables. Namely,

$$(R^{\ell}_{T_1}, M^{\ell}_{\overline{T}_1}), (R^{\rho}_{T_1}, M^{\rho}_{\overline{T}_1}), (R^{\mathrm{s}}_{T_2}, M^{\mathrm{s}}_{\overline{T}_2}), \text{ and } (R^{b}_{T_2}, M^{b}_{\overline{T}_2})_{b \in G}. \quad (6)$$

We first argue this suffices for proving the theorem: In the protocol, ρ, s and $\{r_a\}_{a \in G}$ are chosen uniformly. Similarly, in the simulation ρ', s' and $\{r'_a\}_{a \in G}$ are chosen uniformly. Moreover, the choice of the latter given the former described above is a bijection.

We now go over the four items above one by one, and show the simulator generates the correct distribution for them, even when conditioned on the values of previous items. We will use the notation from Figure 5.

1. $(R^{\rho}_{T_1}, M^{\rho}_{\overline{T}_1})$: By (2), for any $y_{T_1} \in G$ we have

$$h'(y_{T_1}) = \rho' + \mathrm{sum}(y_{T_1}) = \rho + \mathrm{sum}(y_{\overline{T}_1}) + \mathrm{sum}(y_{T_1}) = h_\rho|_{\overline{T_1}, y_{\overline{T}_1}}(y_{T_1}).$$

   Thus, the restricted function that the simulator uses in the simulation of Π_{Δ_t} is equal to $h_\rho|_{\overline{T_1}, y_{\overline{T}_1}}$. Hence, by the robustness of Π_{Δ_t}, these random variables are equally distributed in the NIMPC and in the simulation.

2. $(R^{\ell}_{T_1}, M^{\ell}_{\overline{T}_1})$: By (3), for any $y_{T_1} \in A_{T_1}$ we have

$$\ell'(y_{T_1}) = r'_{\mathrm{sum}(y_{T_1})} = r_{\mathrm{sum}(y_{T_1})+\mathrm{sum}(y_{\overline{T}_1})} = \ell|_{\overline{T_1}, y_{\overline{T}_1}}(y_{T_1}).$$

   It follows that the restricted function used in the simulation of Π_Γ is equal to $\ell|_{\overline{T_1}, y_{\overline{T}_1}}$. Thus, by the robustness of Π_Γ, these random variables are equally distributed in the NIMPC and in the simulation.

3. $(R^{\mathrm{s}}_{T_2}, M^{\mathrm{s}}_{\overline{T}_2})$: By (4), using a similar calculation to that in the first item, the restricted function that the simulator uses in the simulation of $\Pi_{\Delta_{n-t}}$ is equal to $h_s|_{\overline{T_2}, x_{\overline{T}_2}}$. Thus, by the robustness of $\Pi_{\Delta_{n-t}}$, these random variables are equally distributed in the NIMPC and in the simulation.

4. $(R^{b}_{T_2}, M^{b}_{\overline{T}_2})$: Let us first look at the restricted function used in the execution of the NIMPC and denote $a = b - \rho - \mathrm{sum}(y_{\overline{T}_1})$. Thus,

$$\begin{aligned} g_b|_{\overline{T_2}, x_{\overline{T}_2}}(X_{T_2}) &= g_{(a+\mathrm{sum}(y_{\overline{T}_1}))+\rho}|_{\overline{T_2}, x_{\overline{T}_2}}(X_{T_2}) \\ &= f\left(a + \mathrm{sum}(y_{\overline{T}_1}) + \mathrm{sum}(x_{\overline{T}_2}) + \mathrm{sum}(X_{T_2})\right) \\ &\quad \oplus r_{a+\mathrm{sum}(y_{\overline{T}_1})}\left(s + \mathrm{sum}(x_{\overline{T}_2}) + \mathrm{sum}(X_{T_2})\right). \end{aligned} \tag{7}$$

   Furthermore, $a = b - \rho - \mathrm{sum}(y_{\overline{T}_1}) = b - \rho'$.

   Now consider two cases:

   (a) If $a = b - \rho' \in A_{T_1}$, then for any $x_{T_2} \in G^{|T_2|}$

$$g'_b(x_{T_2}) = f\left(a + \mathrm{sum}(y_{\overline{T}_1}) + \mathrm{sum}(x_{\overline{T}_2}) + \mathrm{sum}(x_{T_2})\right) \oplus r'_a(s' + \mathrm{sum}(x_{T_2})) = g_b|_{\overline{T_2}, x_{\overline{T}_2}}(x_{T_2}).$$

   That is, the restricted function used by the simulator in the simulation of $\Pi_0(g_b)$ in this case is equal to $g_b|_{\overline{T_2}, x_{\overline{T}_2}}$. Thus, by the robustness of Π_0, these random variables are equally distributed in the NIMPC and in the simulation.

   (b) If $a = b - \rho' \notin A_{T_1}$, then in the NIMPC $\Pi_\Gamma(\ell)$, the string $r_{a+\mathrm{sum}(y_{\overline{T}_1})}$ and $(R^{\ell}_{T_1}, M^{\ell}_{\overline{T}_1})$ are statistically independent, thus, Equation (5) is consistent with the view from the execution of $\Pi_\Gamma(\ell)$. Furthermore, by (5), the answers that the simulator uses in the simulation of $\Pi_0(g_b)$ in this case are equal to $g_b|_{\overline{T_2}, x_{\overline{T}_2}}$. Thus, by the robustness of Π_0, also in this case, these random variables are equally distributed in the NIMPC and in the simulation.

□

We next show that for our application for boolean symmetric functions, we can improve the communication complexity of $\Pi_1^{S_1,\ldots,S_t}$.

Corollary 4.16. *Let $S_1 = \cdots = S_t = \{0,1\}$ and $G = \mathbb{Z}_{n+1}$. A variant of the* NIMPC $\Pi_1^{S_1,\ldots,S_t}$ *described in Figure 4 is a fully robust* NIMPC *for $\mathcal{H}^G_{\{0,1\},\ldots,\{0,1\},G,\ldots,G}$ with communication complexity $O(n^2 \cdot 2^t + tn^4) = O(2^t \cdot n^4)$.*

Proof. We change $\Pi_1^{S_1,\ldots,S_t}$ such that it executes protocol Π_0 only $t+1$ times. Specifically,

- In Step (1) in Figure 4, choose a random $\rho \in \mathbb{Z}_{t+1}$ (instead of $\rho \in \mathbb{Z}_{n+1}$). In Step (4), the NIMPC Δ_t is executed over the group $\mathbb{Z}_{t+1}$.
- In Step (3) in Figure 4, define for every $a \in \mathbb{Z}_{t+1}$ the function $g_{a+\rho} : G^{n-t} \to \{0,1\}$ by

$$g_{(a+\rho) \bmod (t+1)}(X_{t+1}, \ldots, X_n) \triangleq f(a + \sum_{i=t+1}^{n} X_i) \oplus r_a(s + \sum_{i=t+1}^{n} X_i).$$

  That is, only $t+1$ functions are defined (instead of $|G| = n+1$ functions). In Step (7) and throughout the NIMPC, the NIMPC Π_0 is executed only $t+1$ times (for every $b \in \mathbb{Z}_{t+1}$).

The correctness and robustness of the modified NIMPC follows by the same arguments as in the original NIMPC (noticing that the sum of the inputs of the special set $\{1,\ldots,t\}$ is at most t). □

4.3.3 A combiner using secret sharing

We finally present the NIMPC $\Pi_2^{(S_1,\ldots,S_n)}$ for h, where *all* variables are limited to inputs from S_i. In the next theorem, Theorem 4.17, we summarize the properties of NIMPC $\Pi_2^{(S_1,\ldots,S_n)}$. Theorem 4.3 follows from Theorem 4.17.

Theorem 4.17. *Let G be an abelian group of size m and $S_1,\ldots,S_n$ be subsets of G. The* NIMPC $\Pi_2^{(S_1,\ldots,S_n)}$ *described in Figure 6 is a t-robust* NIMPC *for $\mathcal{H}^G_{S_1,\ldots,S_n}$ with communication complexity $\binom{n}{t} \cdot O(d^{t+2} \cdot n^2 \cdot |G|^3) = O(d^{t+2} \cdot n^{t+2} \cdot |G|^3)$, where $d \triangleq \max_{i\in[n]} |S_i|$. If $G = \mathbb{Z}_{n+1}$ and $S_1 = \cdots = S_t = \{0,1\}$, then the communication complexity of the* NIMPC *is $\binom{n}{t} \cdot O(2^t \cdot n^4) = O(2^t \cdot n^{t+4})$.*

Proof. The correctness of decoding of $\Pi_2^{(S_1,\ldots,S_n)}$ follows from the correctness of the decoding of each $\Pi_1^{(S_j)_{j\in T_i}}(h_i)$ and from (8). The 0-robustness of $\Pi_2^{(S_1,\ldots,S_n)}$ follows from the 0-robustness of each $\Pi_1^{(S_j)_{j\in T_i}}(h_i)$. We next prove the robustness against a non-empty set $T \triangleq T_i$ of size $0 < |T_i| \leq t$. In Figure 7, we describe the simulator for T, which has access to $h|_{\overline{T},x_{\overline{T}}}$ for the fixed h and $x_{\overline{T}}$ (unknown to the simulator).

We next prove the correctness of the simulator. In the NIMPC $\Pi_2^{(S_1,\ldots,S_n)}$, we can view the choice of $f_1,\ldots,f_e$ as first independently choosing each function f_j, for $j \neq i$, at random, then defining $f_i = f \oplus \bigoplus_{j\neq i} f_j$, and defining $h_j|_{\overline{T},x_{\overline{T}}}(X) = f_j(\mathrm{sum}(X_T) + \mathrm{sum}(x_{\overline{T}}))$ for $j \in [e]$.

The simulator independently chooses each function f'_j, for $j \neq i$, at random. It then defines $h'_j|_{\overline{T},x_{\overline{T}}}(X) = f'_j(\mathrm{sum}(X))$ for $j \neq i$. Thus, the first difference between the protocol and the simulator is that the simulator does not shift f'_j by $\mathrm{sum}(x_{\overline{T}})$. However, as f'_j are random, $h_j|_{\overline{T},x_{\overline{T}}}(X)$ and $h'_j|_{\overline{T},x_{\overline{T}}}(X)$ are equally

Correlated randomness generation and encoding:

- Denote $|G| = m$ and let $e = \sum_{i=1}^{t} \binom{n}{i} \leq n^t$.
- Choose random $f_1, \ldots, f_e \in \{0,1\}^m$ under the constraint $\bigoplus_{i=1}^{e} f_i = f$ (we consider f and each f_i both as an m-bit string and as a function $f, f_i : G \to \{0,1\}$). Define $h_i(X_1, \ldots, X_n) = f_i(\sum_{i=1}^{n} X_i)$.
- Identify each $i \in \{1, \ldots, e\}$ with a non-empty subset $T_i \subset \{P_1, \ldots, P_n\}$ of size at most t. Think of h_i as a function $h_i(Y_1, \ldots, Y_{|T_i|}, X_1, \ldots, X_{n-|T_i|})$, where the Y-inputs are the inputs of T_i.
- For each $i \in [e]$, execute the correlated randomness generation and encoding of NIMPC $\Pi_1^{(S_j)_{j \in T_i}}(h_i)$.

Decoding h: For each $i \in [e]$, decode $h_i(x) = f_i(\mathrm{sum}(x))$ and compute the output

$$\bigoplus_{i=1}^{e} f_i(\mathrm{sum}(x)) = f(\mathrm{sum}(x)) = h(x). \tag{8}$$

Figure 6: The NIMPC $\Pi_2^{(S_1,\ldots,S_n)}(h)$, where $h : S_1 \times \cdots \times S_n \to \{0,1\}$ such that $h(X_1 \ldots, X_n) = f(\mathrm{sum}(X))$ for some $f : G \to \{0,1\}$.

distributed for $j \neq i$. The second difference is that in the simulator, $h_i'|_{\overline{T}, x_{\overline{T}}} = h|_{\overline{T}, x_{\overline{T}}} \oplus \bigoplus_{j \neq i} h'|_{\overline{T}, x'_{\overline{T}}}$. Nevertheless,

$$\begin{aligned} h_i(X) &= f_i(\mathrm{sum}(X)) \\ &= f(\mathrm{sum}(X)) \oplus \bigoplus_{j \neq i} f_j(\mathrm{sum}(X)) \\ &= h(X) \oplus \bigoplus_{j \neq i} h_j(X). \end{aligned}$$

Thus, $h_i|_{\overline{T}, x_{\overline{T}}} = h(X)|_{\overline{T}, x_{\overline{T}}} \oplus \bigoplus_{j \neq i} h_j|_{\overline{T}, x_{\overline{T}}}(X)$, which is exactly the way that $h_i'|_{\overline{T}, x_{\overline{T}}}$ is chosen. By the robustness of $\Pi_1^{(S_k)_{k \in T_i}}(h_j)$, the output of its simulator is distributed exactly as the encodings of $\overline{T_i}$ and the joint randomness of T_i for h_j and $x_{\overline{T}}$ for every $j \in [e]$. Observe that the simulator only accesses $h|_{\overline{T}, x_{\overline{T}}}$ on points in $\prod_{j \in T} S_j$ as required. □

5 An NIMPC for iterated product over general groups

Fix some finite group G, and integer n denoting the number of elements in the product. The iterated product is defined as $\mathrm{prod}(X) \triangleq \prod_{i \in [n]} X_i$. We show that a variant of Kilian's PSM protocol [29] for the iterated product is a fully robust NIMPC for this function. This result easily follows for abelian groups, but is more involved for general groups.

1. Let $e = \sum_{i=1}^{t} \binom{n}{i}$.
2. Choose $e-1$ random functions $f'_j : G \to \{0,1\}$, for $j \neq i$, and define $h'_j|_{\overline{T_i}, x_{\overline{T_i}}}(X_{T_i}) = f'_j(\mathrm{sum}(X_{T_i}))$ (notice that the domain of h'_j is $G^{|T_i|}$).
3. Define $h'_i|_{\overline{T_i}, x_{\overline{T_i}}}(X_{T_i})$, where the domain of this function is S_{T_i}, as follows:

$$h'_i|_{\overline{T_i}, x_{\overline{T_i}}}(X_{T_i}) = h|_{\overline{T_i}, x_{\overline{T_i}}}(X_{T_i}) \oplus \bigoplus_{j \neq i} h'_j|_{\overline{T_i}, x_{\overline{T_i}}}(X_{T_i}).$$

4. For every $i \in [e]$, execute the simulator of the NIMPC $\Pi_1^{(S_k)_{k \in T_i}}(h'_j)$, where it uses the above defined oracle $h'_j|_{\overline{T_i}, x_{\overline{T_i}}}(X_{T_i})$.

Figure 7: The simulator for the NIMPC $\Pi_2^{(S_1,\ldots,S_n)}$ for a subset T_i such that $0 < |T_i| \leq t$.

5.1 Result statement and discussion

For abelian groups, Kilian's approach leads to a fully robust NIMPC protocol as described in Example 2.7. That is, let the correlated randomness be a sequence $r_1, \ldots, r_n$ of random elements such that $\prod_{i \in [n]} r_i = e$, and $\mathrm{Gen}_i(\mathrm{prod}) = r_i$. Next, let $\mathrm{Enc}_i(x_i, r_i) = M_i = x_i \cdot r_i$, and the output is reconstructed by multiplying all the encodings. This approach does not work for the general case: the r_i's in the product $\prod_{i \in [n]}(x_i \cdot r_i)$ (in that order) do not necessarily cancel out. In Figure 8 we present an NIMPC $\Pi_{\mathrm{Kil}} = (\mathrm{Gen}_{\mathrm{kil}}, \mathrm{Enc}_{\mathrm{kil}}, \mathrm{Dec}_{\mathrm{kil}})$, which is based on Kilian's randomization method that is fully robust even for non-abelian groups.

Correlated randomness generation:

- Choose random group elements $r_1, \ldots, r_{n-1}$.
- For every $a \in G$, let $r_{1,a} = a \cdot r_1$ and $r_{n,a} = r_{n-1}^{-1} \cdot a$.
- For every $a \in G$ and $2 \leq i \leq n-1$, let $r_{i,a} = r_{i-1}^{-1} \cdot a \cdot r_i$.
- Let $\mathrm{Gen}_{\mathrm{kil},i}(\mathrm{prod}) = R_i = (r_{i,a})_{a \in G}$,

Encoding: $\mathrm{Enc}_{\mathrm{kil},i}(x_i, R_i) = M_i = r_{i,x_i}$.

Decoding prod**:** $\mathrm{Dec}_{\mathrm{kil}}(M_1, \ldots, M_n) = \prod_{i \in [n]} M_i$.

Figure 8: The NIMPC Π_{Kil} for the iterated product.

We will show that in the general case, an oracle to the restricted product function, namely, $\mathrm{prod}\,|_{\overline{T}, x_{\overline{T}}}$, may reveal quite a lot of information about $x_{\overline{T}}$ (that is, even in the ideal model a lot is learned). Roughly

speaking, it reveals products of stretches of honest parties' inputs (between two corrupted parties), up to an equivalence relation determined by the *center* of the group. We next recall this notion, which is central to our proof.

Definition 5.1. *The* center *of a group G, denoted* $\mathrm{Center}(G)$*, is the set of elements in G that commute with all elements in G. Formally,* $\mathrm{Center}(G) \triangleq \{c \in G : c \cdot x = x \cdot c \ , \forall x \in G\}$*. It is easy to see that* $\mathrm{Center}(G)$ *is a subgroup of G.*

For example, if G is abelian, then its center is the entire group. Clearly, the group identity is always in its center. There are infinitely many examples of groups whose center contains only the identity, for instance, all dihedral groups D_n for odd n have a trivial center.

When the center contains only the identity element, the adversary in the ideal world learns the most information. For example, consider an odd n and a set $T = \{1, 3, \ldots, n\}$. An adversary, having access to $\mathrm{prod}|_{\overline{T}, x_{\overline{T}}}$, can learn $x_{\overline{T}}$ when the center of the group is trivial. On the other extreme, when G is abelian, for any T, an access to $\mathrm{prod}|_{\overline{T}, x_{\overline{T}}}$ implies learning $\prod_{i \in \overline{T}} x_i$ and no additional information. In fact, in this case NIMPC Π_{Kil} is just a redundant formulation of the protocol for the abelian case. We show that no matter what the size of the center of the group is, the view of T can always be simulated from $\mathrm{prod}|_{\overline{T}, x_{\overline{T}}}$.

Theorem 5.2. *For any positive integer n and a finite group G, the* NIMPC Π_{Kil} *is a fully robust* NIMPC *protocol for the function* $\mathrm{prod}(X_1, \ldots, X_n) = \prod_{i \in [n]} X_i$ *with communication complexity* $|G| \log |G|$*. Furthermore, the simulator running time is* $\mathrm{poly}(|G|, n)$*.*

Notice that the communication complexity and the simulation are efficient when $|G| = \mathrm{poly}(n)$.

5.2 Warm-up

As a warm-up we prove the claim in the case $n = 3$ and $T = \{2\}$. Afterwards we give a formal proof of the general setting, but this case essentially captures what goes on in the general setting. The main tool in the proof will be the following lemma. Intuitively, it says that knowing a restricted product function determines the unknown inputs up to a multiple of an element from $\mathrm{Center}(G)$. Lemma 5.3 is used also in the proof of the general case, when $n \geq 3$.

Lemma 5.3. *Consider the following set* Eq *of equations in the variables Y_1 and Y_2.*

$$\mathrm{Eq} \triangleq \{Y_1 \cdot a \cdot Y_2 = b_a\}_{a \in G},$$

where $b_a \in G$ for every $a \in G$. Suppose there exists a solution $(Y_1, Y_2) = (y_1, y_2) \in G^2$ to Eq*. Then the set of solutions* $\mathrm{SOL} \subseteq G^2$ *to* Eq *is exactly equal to* $\{(y_1 \cdot c, c^{-1} \cdot y_2) : c \in \mathrm{Center}(G)\}$.

Proof. Suppose that $(y_1, y_2) \in \mathrm{SOL}$. That is, for every $a \in G$,

$$y_1 \cdot a \cdot y_2 = b_a.$$

Fix any $c \in \mathrm{Center}(G)$. Then, clearly, for any $a \in G$,

$$(y_1 \cdot c) \cdot a \cdot (c^{-1} \cdot y_2) = y_1 \cdot a \cdot y_2 = b_a.$$

Therefore, $(y_1 \cdot c, c^{-1} \cdot y_2) \in \mathrm{SOL}$.

Now, fix any $(y_1', y_2') \in \mathrm{SOL}$. Write $y_1' = y_1 \cdot c_1$ and $y_2' = c_2 \cdot y_2$ for some $c_1, c_2 \in G$. As $(y_1', y_2') \in \mathrm{SOL}$, we have for every $a \in G$

$$b_a = y_1' \cdot a \cdot y_2' = y_1 \cdot c_1 \cdot a \cdot c_2 \cdot y_2 = y_1 \cdot a \cdot y_2.$$

From the rightmost equation, we get for every $a \in G$

$$c_1 \cdot a = a \cdot c_2^{-1}.$$

In particular, for $a = 1$ the above equation implies $c_2^{-1} = c_1$. Thus, we have for every $a \in G$

$$c_1 \cdot a = a \cdot c_1,$$

which implies $c_1 \in \mathrm{Center}(G)$. □

Suppose $n = 3$, $T = \{2\}$, and $X_1 = x_1, X_3 = x_3 \in G$. We next describe a simulator for T in Π_{Kil} and sketch its correctness. The simulator should generate the distribution $(M \triangleq (M_1, M_3), R \triangleq (R_2))$. Let us recall that the randomness is $R = \{r_a \triangleq r_1^{-1} \cdot a \cdot r_2\}_{a \in G}$ for randomly chosen $r_1, r_2 \in G$. and the encodings M is the pair $(x_1 \cdot r_1, r_2^{-1} \cdot x_3)$.

For every $a \in G$, he simulator queries the oracle $\mathrm{prod}|_{\{1,3\},(x_1,x_3)}(a)$, denote its answer by b_a. The simulator now solves the system of equations $\{Y_1 \cdot a \cdot Y_2 = b_a\}$ by searching over all possibilities in G^2. Let x_1', x_3' be a solution. The simulator now executes the joint randomness generation of NIMPC Π_{Kil} and its encoding for $1, 3$ with inputs x_1', x_3' respectively. The simulator returns $(M_1, M_3), R_2$.

By Lemma 5.3 there exits a $c \in \mathrm{Center}(G)$ such that $x_1' = x_1 \cdot c$ and $x_3' = c^{-1} \cdot x_3$. Note that the output of the NIMPC with inputs x_1, x_3 and randomness r_1, r_2 and the output of the simulator with with inputs x_1', x_3' and randomness $r_1' = c^{-1} \cdot r_1, r_2' = r_2 \cdot c^{-1}$ are equal as

$$\mathrm{Enc}_{\mathrm{kil},1}(x_1', R_1') = x_1' \cdot r_1' = (x_1 \cdot c) \cdot (c^{-1} \cdot r_1) = \mathrm{Enc}_{\mathrm{kil},1}(x_1, R_1).$$

Similarly,

$$\mathrm{Enc}_{\mathrm{kil},3}(x_3', R_3') = (r_2')^{-1} \cdot x_3' = (c \cdot r_2^{-1}) \cdot (c^{-1} \cdot x_3) = \mathrm{Enc}_{\mathrm{kil},3}(x_3, R_3).$$

Furthermore,

$$R_2' = \{(r_1')^{-1} \cdot a \cdot r_2'\}_{a \in G} = \{r_1^{-1} \cdot c \cdot a \cdot r_2 \cdot c^{-1}\}_{a \in G} = R_2.$$

Thus, the output distributions of the NIMPC with x_1, x_3 and the simulator with x_1', x_3' are identical.

5.3 Proof of the general case

We proceed with the formal proof of Theorem 5.2.

5.3.1 A key technical claim

The main tool in the proof is a generalization of Lemma 5.3 describing the set of solutions to a certain set of equations in many variables. For its statement and proof, the following definition will be convenient.

Definition 5.4. *Fix a sequence $y = (y_1, \ldots, y_t) \in G^t$. We say a sequence $y' = (y_1', \ldots, y_t') \in G^t$ is* $\mathrm{Center}(G)$-close *to y if there exist $c_1, \ldots, c_t \in \mathrm{Center}(G)$ such that $y_i' = y_i \cdot c_i$ for every $1 \le i \le t$ and $c_t = (c_1 \cdots c_{t-1})^{-1}$.*

Lemma 5.5. *Consider the following set* Eq *of equations in the variables* $Y_1, \ldots, Y_t$.

$$\mathrm{Eq} \triangleq \{Y_1 \cdot a_1 \cdots Y_{t-1} \cdot a_{t-1} \cdot Y_t = b_{a_1,\ldots,a_{t-1}}\}_{(a_1,\ldots,a_{t-1}) \in G^{t-1}},$$

where $b_{a_1,\ldots,a_{t-1}} \in G$ *for every* $a_1, \ldots, a_{t-1} \in G^{t-1}$. *Suppose there exists a solution* $(Y_1, \ldots, Y_t) = (y_1, \ldots, y_t) \in G^t$ *to* Eq. *Then the set of solutions* $\mathrm{SOL} \subseteq G^t$ *to* Eq *is exactly equal to the set of sequences* $(y'_1, \ldots, y'_t) \in G^t$ *that are* $\mathrm{Center}(G)$*-close to* $(y_1, \ldots, y_t)$.

Proof. Fix any $y' = (y'_1 = y_1 \cdot c_1, \ldots, y'_t = y_t \cdot c_t)$ that is $\mathrm{Center}(G)$-close to y. We have for any $(a_1, \ldots, a_{t-1}) \in G^{t-1}$

$$y'_1 \cdot a_1 \cdot \ldots \cdot y'_{t-1} \cdot a_{t-1} \cdot y'_t = y_1 \cdot a_1 \cdot \ldots \cdot y_{t-1} \cdot a_{t-1} \cdot y_t = b_{a_1,\ldots,a_{t-1}}.$$

Hence, y' is also a solution to Eq.

We next prove by induction on t that every solution to Eq is $\mathrm{Center}(G)$-close to y. Note that the base case, i.e., $t = 2$, is precisely the statement of Lemma 5.3 from the warm-up. Assume that the claim holds for $t-1$. Fix a solution $y' = (y'_1, \ldots, y'_t)$ to Eq. We wish to show that y' is $\mathrm{Center}(G)$-close to y.

By ranging over the values of a_1, while fixing $a_2, \ldots, a_{t-1}$ to one, and using the fact that y and y' are solutions to Eq, we get that

$$y_1 \cdot a_1 \cdot y_2 \cdots y_t = y'_1 \cdot a_1 \cdot y'_2 \cdots y'_t = b_{a_1,1,\ldots,1},$$

for every $a_1 \in G$. In other words, $(y_1, y_2 \cdots y_t)$ and $(y'_1, y'_2 \cdots y'_t)$ both satisfy a set of equations in *two* variables of the form considered in Lemma 5.3 (or the base case $t = 2$ of this lemma). It follows, in particular, that $y'_1 = y_1 \cdot c_1$ for $c_1 \in \mathrm{Center}(G)$.

On the other hand, by fixing $a_1 = 1$, and ranging over the values of $a_2, \ldots, a_{t-1}$ we now have

$$y_1 \cdot y_2 \cdot a_2 \cdots y_{t-1} \cdot a_{t-1} \cdot y_t = y_1 \cdot c_1 \cdot y'_2 \cdot a_2 \cdots y'_{t-1} \cdot a_{t-1} \cdot y'_t = b_{1,a_2,\ldots,a_t},$$

for every $(a_2, \ldots, a_{t-1}) \in G^{t-2}$. Canceling out y_1 and writing $y''_2 = y'_2 \cdot c_1$ we get

$$y_2 \cdot a_2 \cdots y_{t-1} \cdot a_{t-1} \cdot y_t = y''_2 \cdot a_2 \cdots y'_{t-1} \cdot a_{t-1} \cdot y'_t,$$

for every $(a_2, \ldots, a_{t-1}) \in G^{t-2}$. In other words, $(y_2, y_3, \ldots y_t)$ and $(y''_2, y'_3, \ldots, y'_t)$ both satisfy a set of equations in $t-1$ variables of the form covered by the induction hypothesis. It follows that

1. $y''_2 = y_2 \cdot c'_2$, for $c'_2 \in \mathrm{Center}(G)$, and, therefore, $y'_2 = y''_2 \cdot c_1^{-1} = y_2 \cdot c'_2 \cdot c_1^{-1} = y_2 \cdot c_2$ for $c_2 \triangleq c_1^{-1} \cdot c'_2 \in \mathrm{Center}(G)$.
2. For $3 \leq i \leq t$, $y'_i = y_i \cdot c_i$ for $c_i \in \mathrm{Center}(G)$. Moreover, $c_t = (c'_2 \cdot c_3 \cdots c_{t-1})^{-1}$.

As $c_1^{-1} \cdot c_2^{-1} = (c'_2)^{-1}$, we have $c_t = (c_1 \cdot c_2 \cdots c_{t-1})^{-1}$, as required. Thus, y' is $\mathrm{Center}(G)$-close to y and we are done. □

5.3.2 Showing that Π_{Kil} is fully robust

Fix a set $T \subseteq [n]$. We assume for now that

1. $1, n \notin T$. Informally, this means that "the players at the ends are honest".

2. For every $1 \leq i \leq n-1$, either $i \in T$ or $i+1 \in T$ (or both). Informally, this means that "there are no consecutive honest players".

We will prove that Π_{Kil} is T-robust by constructing a simulator for T. Afterwards, we argue that simple adjustments can handle a general $T \subseteq [n]$.

The structure of the construction of the simulator and its proof are as follows. We first identify for which inputs $x_{\overline{T}}$ and $x'_{\overline{T}}$ the restricted functions $\text{prod}|_{\overline{T},x_{\overline{T}}}$ and $\text{prod}|_{\overline{T},x'_{\overline{T}}}$ are identical. We then show that when the restricted functions are identical the outputs of the NIMPC are identical. Thus, we construct the simulator by first finding inputs $x'_{\overline{T}}$ such that $\text{prod}|_{\overline{T},x_{\overline{T}}} = \text{prod}|_{\overline{T},x'_{\overline{T}}}$ and then executing the correlated random generation of Π_{Kil} and the encoding of Π_{Kil} for $\overline{T}$ with inputs $x'_{\overline{T}}$.

Fix inputs $x_{\overline{T}}$ and $x'_{\overline{T}}$ for the variables outside of T such that the restricted product functions $h|_{\overline{T},x_{\overline{T}}}$ and $h|_{\overline{T},x'_{\overline{T}}}$ are identical. We will show a bijection $\phi : G^{n-1} \to G^{n-1}$ with the property that the view $(M_{\overline{T}}, R_T)$ of the NIMPC with inputs $x_{\overline{T}}$ and correlated randomness $(r_1, \ldots, r_{n-1})$ is identical to the view $(M'_{\overline{T}}, R'_T)$ of the NIMPC with inputs $x'_{\overline{T}}$ and correlated randomness $\phi(r_1, \ldots, r_{n-1})$.

Denote $\overline{T} = \{i_1 < \cdots < i_t\}$, and, for simplicity, $x_{\overline{T}} = (x_1, \cdots, x_t)$ and $x'_{\overline{T}} = (x'_1, \ldots, x'_t)$. Here comes our crucial use of Lemma 5.5. Note that the fact that the restricted functions are the equal, together with the assumption 2 above, imply that whatever values $a_1, \ldots, a_{t-1}$ we insert in between the inputs $x_{\overline{T}}$ and in between the inputs $x'_{\overline{T}}$, the products remain equal. That is, for any $(a_1, \ldots, a_{t-1}) \in G^{t-1}$,

$$x_1 \cdot a_1 \cdot \ldots \cdot x_{t-1} \cdot a_{t-1} \cdot x_t = x'_1 \cdot a_1 \cdot \ldots \cdot x'_{t-1} \cdot a_{t-1} \cdot x'_t.$$

By Lemma 5.5, $x'_{\overline{T}} = (x_1 \cdot c_1, \ldots, x_t \cdot c_t)$ is $\text{Center}(G)$-close to $x_{\overline{T}}$.

Before defining ϕ, we define elements $d_1, \ldots, d_{n-1} \in G$ as follows. For $1 \leq i \leq n-1$, d_i will be the (inverse of the) product of the c_j's corresponding to elements of $\overline{T}$ appearing up to i. Formally,

$$d_i = \left(\prod_{1 \leq j \leq t \text{ s.t } i_j \leq i} c_j \right)^{-1}.$$

Now define the bijection ϕ by $\phi(r_1, \ldots, r_{n-1}) \triangleq (r'_1, \ldots, r'_{n-1})$, where $r'_i = r_i \cdot d_i$ for every $1 \leq i \leq n-1$.

Note first, that ϕ is indeed clearly a bijection. It is left to show that the views $(M_{\overline{T}}, R_T)$ and $(M'_{\overline{T}}, R'_T)$ are identical. It will be convenient to denote, for $i \in \{1, \ldots, n\}$, the "i'th index" of the view $(M_{\overline{T}}, R_T)$ by Z_i. That is, $Z_i = M_i$ when $i \in \overline{T}$, and $Z_i = R_i$ when $i \in T$. We define Z'_i for $i \in \{1, \ldots, n\}$ similarly.

We prove that $Z_i = Z'_i$ for every i. We frequently use the fact that the elements $\{c_i\}$ and $\{d_i\}$ are in $\text{Center}(G)$.

$i = 1$**:** We have

$$Z'_1 = M'_1 = x'_1 \cdot r'_1 = x_1 \cdot c_1 \cdot r_1 \cdot c_1^{-1} = x_1 \cdot r_1 = M_1 = Z_1.$$

$i = n$**:** We have

$$Z'_n = M'_n = r_{n-1}^{-1} \cdot d_{n-1}^{-1} \cdot c_n \cdot x_n = r_{n-1}^{-1} \cdot (c_1 \cdots c_{n-1}) \cdot x_n \cdot (c_1 \cdots c_{n-1})^{-1} = r_{n-1}^{-1} \cdot x_n = Z_n.$$

$i_j \in \overline{T} \setminus \{1, n\}$**:** We have

$$\begin{aligned} Z'_{i_j} &= M'_{i_j} \\ &= \phi(r_{i_j-1})^{-1} \cdot x'_j \cdot \phi(r_{i_j}) \\ &= r_{i_j-1}^{-1} \cdot d_{i_j-1}^{-1} \cdot x_j \cdot c_j \cdot d_{i_j} \cdot r_{i_j} \\ &= r_{i_j-1}^{-1} \cdot (c_1 \cdots c_{j-1}) \cdot x_j \cdot c_j \cdot (c_1 \cdots c_j)^{-1} \cdot r_{i_j} \\ &= r_{i_j-1}^{-1} \cdot x_j \cdot r_{i_j} = Z_{i_j}. \end{aligned}$$

$i \in T$**:** In this case, $Z'_i = \{r_{i-1}^{-1} \cdot d_{i-1}^{-1} \cdot a \cdot d_i \cdot r_i\}_{a \in G} = \{r_{i-1}^{-1} \cdot a \cdot r_i\}_{a \in G} = Z_i$, since $d_{i-1} = d_i$ when $i \in T$.

Removing assumptions about T. We sketch how to remove the two assumptions made about T. First, suppose that the "edges" are *not* part of $\overline{T}$. That is $1 < i_1$ and/or $i_t < n$. In the range $i_1 \leq j < i_t$, we will define r'_j in the same way as before. Outside this range, we simply define $r'_j = r_j$. It is easy to check that $\phi(r_1, \ldots, r_n) \triangleq (r'_1, \ldots, r'_n)$ defined this way is still the desired bijection.

Second, suppose that there *are* continuous blocks of honest players, i.e., maximal sequences of indices $\{i_1, i_1 + 1, \ldots, i_1 + e\} \subseteq \overline{T}$ of length greater than one.

In this case, rather than looking at $M_{\overline{T}}$, we look at $N_{\overline{T}}$, defined as the sequence of *products* of blocks of messages of honest players (with inputs $x_{\overline{T}}$ and randomness $(r_1, \ldots, r_{n-1})$). We think of each such product as one message of a new player, i.e. we think of 'collapsing' the honest players into fewer players representing these blocks. We also define $r_0 = (r_{a_1}, \ldots, r_{a_\ell})$ to be a subsequence of the randomness $(r_1, \ldots, r_{n-1})$ by taking only indices that are 'not inside a block of $\overline{T}$' - i.e., indices $i \in \overline{T}$ such that either $i - 1 \in T$ or $i + 1 \in T$. Note that the view $(N_{\overline{T}}, r_T)$ depends only on the input $x_{\overline{T}}$ and the subsequence r_0. In fact it is exactly a view of the same protocol Π_{Kil} where we have fewer players - as the honest players of a block have 'collapsed' into one player.

Using the above arguments in the same way, we can now show a bijection mapping the subsequence r_0 to a subsequence r'_0 that, together with the input $x'_{\overline{T}}$, will generate an identical view $(N'_{\overline{T}}, R'_T)$. Now to extend the mapping, choose the rest of the r_i's in some way. Going over these choices will cause the view $(M_{\overline{T}}, r_T)$ to range over all sequences satisfying the constraint that the product of the messages in the i'th block of honest players is N_i. The same will happen with $(M'_{\overline{T}}, R'_T)$ when ranging over the values of the r'_i's that are not part of the sequence r_0.

Efficiency of the simulator. It is left to address the claim regarding the efficiency of the simulation. Specifically, we want to argue that given $h|_{\overline{T}, x_{\overline{T}}}$, we can generate in time poly$(|G|, n)$ the distribution $(M_{\overline{T}}, R_T)$. Note that if we could obtain input values $x'_{\overline{T}}$ such that $h|_{\overline{T}, x'_{\overline{T}}} = h|_{\overline{T}, x_{\overline{T}}}$, this would suffice. Given $h|_{\overline{T}, x_{\overline{T}}}$, obtaining $x'_{\overline{T}}$ boils down to finding a solution to an equation system of the type discussed in Lemma 5.5 in $|\overline{T}| \leq n$ variables.[2]

Examining the proof of Lemma 5.5, one can show that such a solution can be found in $O(n \cdot |G|^3)$ time when it exists, assuming multiplication in G is considered an $O(1)$ operation. More specifically, the inductive step of the proof shows that finding a solution to such a system in m variables, reduces to solving two systems also of the form considered in Lemma 5.5, the first in two variables, and the second in $m - 1$

[2] Again, this is not precise in the case that 1 or n belong to T, but this case is easily treated.

variables. The first system can be solved by exhaustive search in $O(|G|^3)$ time. Now, using an inductive argument, we get that the system can be solved in time $O(m \cdot |G|^3)$.

6 Counter-examples to constructions from the literature

In this section, we demonstrate how natural adaptations of standard PSM protocols (i.e., 0-robust NIMPC) from the literature to our setting fail to provide even 1-robustness. For simplicity, we focus on functions $h : \{0,1\}^n \to \{0,1\}$.

Leaking the function h. An NIMPC for a class of functions $\mathcal{H}$ should make sure that h itself does not leak beyond what is revealed by $h|_{\overline{T},x_{\overline{T}}}$. In fact, most NIMPC protocols from the literature do not inherently hide h (even against $T = \emptyset$). In the counter-examples for the known PSM protocols, we will consider an NIMPC for a single function h and show that even for sets T or cardinality 1 they leak information about other inputs.

6.1 A generic attack on the NIMPC protocols of Kilian and Yao

We consider 0-robust NIMPC protocols from the literature that are "read once based" – they are constructed in the following two steps: (1) construct an NIMPC protocol for functions represented as read-once formula, and (2) to construct an NIMPC protocol for functions represented as general formulas, first treat each appearance of a variable as a different variable, thus, transforming the formula to a read-once formula, next execute the NIMPC protocol for the resulting read-once formula, and the correlated randomness R_i contains the correlated randomness of all appearance of variable X_i in the original formula. This causes no problems for 0-robust NIMPC protocols (in the honest-but-curious model). However, for sets $T = \{i\}$ for some i, this already is problematic even in the honest-but-curious model, since $\{i\}$ can compute $h|_{\overline{\{i\}},x_{\overline{\{i\}}}}$ for "non-consistent assignments", i.e., for assignments in which two appearances of X_i get different values.

Example 6.1. Consider the formula $h(X_1, X_2, X_3) = X_1X_2 \vee X_1X_3$. The set $\{1\}$ can only learn $x_2 \vee x_2$ from $h|_{\{2,3\},(x_2,x_3)}$. However, by assigning the first appearance of X_1 true and the second false, the set $\{1\}$ can learn x_2 from the "read-once based" NIMPC (and, similarly, is can learn x_3).

We present a stronger attack on "read-once based" NIMPC: We show below that there exists an efficient compiler transforming any formula C into a "logically equivalent" formula C', with only a polynomial blowup in size (and a constant blowup in depth), that allows every singleton $T = \{i\}$ to learn $x_{\overline{T}}$ completely given $R_i, M_{[n]\setminus i}$. This is done by carefully choosing one or more inconsistent partial assignments $x' = (x'_T, x_{\overline{T}})$ and learning $C'(x')$. Thus, a bad implementation C' of h, as above, completely reveals $x_{\overline{T}}$ even for $|T| = 1$. Since $|C'|$ is not much bigger than the size of the best C fhr f, the possibility of using such C' (or other kinds of "leaky" C') is a real issue. In contrast, the candidates 0-robust protocols that we consider work for any implementation C of h.

We first demonstrate that many 0-robust NIMPC protocols from the literature "are read-once based". By our result, these NIMPC break completely even against a singleton, revealing all other inputs (at least, unless special care is taken to use less vulnerable implementation C).

Yao's Garbled Circuit NIMPC**.** The first NIMPC candidates that we consider are various Yao-based NIMPC protocols [33, 30, 26]. In Yao-based information-theoretic NIMPC protocols [26], the function

is represented by a formula and a pair of fresh keys (k_0, k_1) is given in the correlated randomness R_i for each appearance of the variable X_i. Additionally, the correlated randomness contains a garbled circuit encoding a (public) function h to be evaluated. In the encoding stage, the encoding M_j is the key k_{x_j} for each appearance of x_j in the formula. The decryption proceeds gate by gate starting from the leafs. For each gate, a key representing its output is computed from the two keys known for its children. Yao-based protocols have the property that we need, since a fresh pair of keys (k_0, k_1) is assigned to every appearance of a variable and a set $\{i\}$ can evaluate the formula using the key k_0 for one appearance of X_i and the key k_1 for a different appearance of X_i.

Kilian's NIMPC**.** Barrignton's seminal paper [5] shows how to transform a formula (over the basis AND, NOT) into a permutation branching program over S_5 (equivalently, a constant width "standard" branching program). That is, for each of the ℓ_i appearance of a variable X_i in the formula, there is a pair of permutations $(p^1_{i,0}, p^1_{i,1}), \ \ldots \ , p^{\ell_i}_i = (p^{\ell_i}_{i,0}, p^{\ell_i}_{i,1})$. To evaluate the branching program on an input x, pick $m^j_i = p^j_{i,x_i}$ out of every p^j_i corresponding to an appearance of X_i in the formula. Then, the permutations m^j_i's are multiplied in a certain (fixed, publicly known) order. The output of the multiplication is of two fixed permutations $o \neq e$, corresponding to the outputs of 0 and 1, respectively. The 0-robust NIMPC presented by Killian [29] proceeds as follows. In the correlated randomness generation, a formula C is transformed into a permutation branching program as in [5]. Then, a 0-robust NIMPC for iterated group product is used, using the p^j_i's as the inputs. See Section 5 for the description of the NIMPC for iterated group product of [29]. Notice that also in this NIMPC protocol, a singleton $\{i\}$ can choose inconsistent permutations, e.g., $p^1_{i,0}$ and $p^2_{i,1}$, in the computation of the iterated product and obtain the output of the branching program on inconsistent inputs.

6.1.1 Compiling formulas into a bad equivalent formula

We now sketch the compiler that takes a formula C and transforms it into a "highly vulnerable" formula C' for the same function, where every singleton $\{i\}$ for can learn $x_{[n]\setminus\{i\}}$. For simplicity assume that n is a power of two. Given a formula $C(X_1, \ldots, X_n)$, consider the formula C' which takes an input $X_{1,1}, \ldots, X_{1,\log n}, \ldots, X_{n,1}, \ldots, X_{n,\log n}$. Given an assignment $x_{1,1}, \ldots, x_{1,\log n}, \ldots, x_{n,1}, \ldots, x_{n,\log n}$, if for each i, $x_{i,1} = \cdots = x_{i,\log n} = x_i$ for some value x_i, then output $C(x_1, \ldots, x_n)$. Otherwise, consider the first i for which this is not the case and output x_s, where $s - 1 = (x_{i,1}, \ldots, x_{i,\log n})$. Finally, C' is modified by replacing each appearance of $X_{i,j}$ by X_i each appearance of $\overline{X}_{i,j}$ by $\overline{X}_i$. Clearly, the formula is equivalent to C. However, every corrupted singleton $\{i\}$, can learn x_s (for every s) by substituting all appearances of the type $X_{i,j}$ according to s.

6.2 An attack against the 0-robust Ishai-Kushilevitz NIMPC

The above attack strategy does not apply to the NIMPC protocols of Ishai and Kushilevitz [24], in which all appearances of each input (in the original program, say formula or branching program) are bundled together by the NIMPC, and there is no obvious way of substituting them independently. Nevertheless, we show a different type of attack that allows a singleton to learn inputs that are not accessible in the ideal world. We present a simple case of this attack; however, it can be generalized to general functions.

The NIMPC protocols of [24] represent functions as counting branching programs. However, here we only consider the special case of deterministic branching programs as defined in Definition 4.8 restricted to binary alphabet (i.e., to the group $\mathbb{Z}_2$). Furthermore, we consider the NIMPC Π_{IK} described in Figure 1 with the modification that the matrices are over a field $\mathbb{F}_p$ for a prime $p > 2$. Consider the branching program with $n = 2$, four vertices $\{s, v_1, v_2, t\}$, and 3 edges: edge (s, v_1) labeled by $X_1 = 1$, edge (v_1, v_2) labeled

by $X_1 = 0$, and edge (v_2, t) labeled by $X_2 = 1$. Clearly, this branching program computes the all-zero function. The matrices $B_{i,a}$ for this branching program are:

$$B_{1,0} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \qquad B_{1,1} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

$$B_{2,0} = \begin{pmatrix} 0 & 0 & 0 \\ -1 & 0 & 0 \\ 0 & -1 & 0 \end{pmatrix} \quad B_{2,1} = \begin{pmatrix} 0 & 0 & 0 \\ -1 & 0 & 0 \\ 0 & -1 & 1 \end{pmatrix}.$$

The view of $T = \{1\}$ in the NIMPC Π_{IK} is the randomness $R_1 = (M_{1,0} = P \cdot B_{1,0} \cdot V + Z_1, M_{1,1} = P \cdot B_{1,1} \cdot V + Z_1)$ and the encoding $M_2 = P \cdot B_{2,x_2} \cdot V + Z_2$. We can now take a linear combination of $M_{1,0}, M_{1,1}$, and M_2 and the set $T = \{1\}$ can learn x_2 as follows:

$$M = 2M_{1,1} - M_{1,0} + M_2 = P \cdot (2B_{1,1} - B_{1,0} + B_{2,x_2}) \cdot V$$

(as $Z_1 + Z_2 = 0$), where P, U have full rank. However,

$$2B_{1,1} - B_{1,0} + B_{2,x_2} \quad = \quad \begin{pmatrix} 2 & 0 & 0 \\ -1 & -1 & 0 \\ 0 & -1 & x_2 \end{pmatrix}.$$

Thus, M has full rank if and only if $x_2 = 1$. This shows that $\{1\}$ can learn x_2.

7 Improving the efficiency of the NIMPC for symmetric boolean functions

In this section, we give a more efficient t-robust NIMPC for symmetric functions when the domain of each party is boolean.

Theorem 7.1. *Let $\mathcal{H}$ be the set of symmetric boolean functions $f : \{0,1\}^n \to \{0,1\}$. For every t, there is a t-robust* NIMPC *protocol for $\mathcal{H}$ with communication complexity $O(tn^{t+1} \cdot (\log n + 2^t))$. In particular, there is a 1-robust* NIMPC *protocol for $\mathcal{H}$ with communication complexity $O(n^2 \cdot \log n)$.*

In this section, G always denotes the group $\mathbb{Z}_{n+1}$. The above theorem follows by replacing the protocol Π_0 of Theorem 4.12, which is fully robust for $\mathcal{H}^G_{G,\ldots,G}$, by a more efficient fully robust NIMPC Π''_0 for $\mathcal{H}^G_{G,\ldots,G}$ from Theorem 7.2 in the NIMPC Π_1 described in Section 4.3. More precisely, we replace Π_0 in Theorem 4.15, when $G = \mathbb{Z}_{n+1}$. This replacement does not work when $G \neq \mathbb{Z}_{n+1}$, e.g., for symmetric functions when the domain of each party is not boolean.

The communication complexity of the resulting NIMPC Π_1 is recalculate as follows (using the optimization of Corollary 4.16):

- One execution of the NIMPC Π_Γ for the function ℓ with complexity $O(n \cdot t \cdot 2^t)$,
- Two executions of the NIMPC for shifted sum with complexity $O(\log n)$, and
- t executions of the NIMPC Π''_0 for g_b, each with complexity $O(n \log n)$.

Finally, plugging the resulting Π_1 into Π_2 in Theorem 4.17, and recalculating the communication complexity, we get the result of Theorem 7.1.

Theorem 7.2. *Let G be the group $\mathbb{Z}_{n+1}$. There is a fully robust* NIMPC Π''_0 *for $\mathcal{H}^G_{G,\ldots,G}$ with communication complexity $O(n \cdot \log n)$.*

Notation 7.3. *For two functions $\tau_1, \tau_2 : G \to G$ we let $\tau = \tau_1 \cdot \tau_2$ be the function $\tau : G \to G$, where $\tau(a) = \tau_2(\tau_1(a))$. In particular, $\tau^i = \underbrace{\tau \cdot \ldots \cdot \tau}_{i \text{ times}}$. When τ is a permutation, we use the convention that τ^0 is the identity permutation. Let $\mathcal{M}_S$ be the set of permutations on $\mathcal{M}$. Let σ be the permutation on $\mathcal{M} = \mathbb{Z}_{n+1}$ defined by $\sigma(j) = j + 1$ for every $j \in G$.*

In Figure 9, we present NIMPC Π'_0 for the set of functions $\mathcal{H}_{G,\ldots,G}$. This NIMPC is correct, but satisfies a weaker notion of robustness, that is, for $T = \emptyset$ it leaks the size of $f^{-1}(0)$. This said, it is an NIMPC for every single function in $\mathcal{H}_{G,\ldots,G}$ (e.g., for the majority function). In Figure 10, we slightly modify Π'_0 and achieve full robustness for the entire class $\mathcal{H}_{G,\ldots,G}$. The two protocols are a tweak of Kilian's protocol Π_{Kil} for group product from Section 6.1 using the group of permutations. The basic idea is to replace the messages of the first and last player that are permutations in Π_{Kil} by one that is a (non-injective) function. This has the effect of reducing the information the messages convey. As the set of functions from G to G is still a semi-group with the function composition operation, we can still multiply the different messages as we do in Π_{Kil} to obtain the desired output.

7.1 Simplified protocol for a specific function

We first present a simpler protocol that exposes to an outside observer additional information on the function f, specifically the quantity $|f^{-1}(0)|$. We will still be able to show that any *non-empty* set of players $T \subseteq [n]$ learns only the restricted function $h|_{\overline{T}, x_{\overline{T}}}$.

Theorem 7.4. *Fix any function $h : G^n \to \{0,1\}$ such that $h(X_1, \ldots, X_n) = f(\sum_{i=1}^n X_i)$ for some function $f : G \to \{0,1\}$. The* NIMPC Π'_0*, described in Figure 9, is fully robust for h.*

Proof. The proof is similar to that of Theorem 5.2. Fix a subset $T \subseteq [n]$. For brevity, we skip the case $T = \emptyset$ here. It is discussed in the proof of Theorem 7.6 in the next subsection for the more involved protocol. Fix any inputs $x_{\overline{T}}, x'_{\overline{T}}$ for the players outside of T such that $h|_{\overline{T}, x_{\overline{T}}} \equiv h|_{\overline{T}, x'_{\overline{T}}}$, that is, $f(a + \text{sum}(x_{\overline{T}})) = f(a + \text{sum}(x'_{\overline{T}}))$ for every $a \in G$. Denote by $(R_T, M_T)(x_{\overline{T}}, r)$ the view of the set T in the protocol when the inputs $x_{\overline{T}}$ are used, and the vector of permutations chosen in the preprocessing stage is $r = (r_1, \ldots, r_{n-1}) \in G_S^{n-1}$. Observe that the view (R_T, M_T) is indeed a deterministic function of $x_{\overline{T}}$ and r.

We present a bijection $\phi : G_S^{n-1} \to G_S^{n-1}$ such that for all $r \in G_S^{n-1}$,

$$(R_T, M_T)(x_{\overline{T}}, r) = (R_T, M_T)(x'_{\overline{T}}, \phi(r)).$$

This suffices to prove the theorem. Define $a \triangleq \sum_{j \in \overline{T}} (x_j - x'_j)$. A crucial point is that $\sigma^a \cdot f = f$ as for any $b \in G$,

$$f(b) = f(\sum_{i \in \overline{T}} x'_i + (b - \sum_{i \in \overline{T}} x'_i)) = f(\sum_{i \in \overline{T}} x_i + (b - \sum_{i \in \overline{T}} x'_i)) = f(a + b) = f(\sigma^a(b)) = (\sigma^a \cdot f)(b).$$

This implies that $f = \sigma^{-a} \cdot f$. For $1 \leq i \leq n-1$, define

$$d_i \triangleq \sum_{j > i, j \in \overline{T}} (x'_j - x_j) + a$$

Correlated randomness generation:

- Choose random permutations $r_1, \ldots, r_{n-1} \in G_S$.
- Let $R_1 = (R_{1,0}, R_{1,1}) = (L \cdot r_1, L \cdot \sigma \cdot r_1)$, where $L : G \to G$ is the identically zero function.
- For $1 \leq i \leq n-1$, let $R_i = (R_{i,0}, R_{i,1}) = (r_{i-1}^{-1} \cdot r_i, r_{i-1}^{-1} \cdot \sigma \cdot r_i)$.
- Let $R_n = (R_{n,0}, R_{n,1}) = (r_{n-1}^{-1} \cdot f, r_{n-1}^{-1} \cdot \sigma \cdot f)$.

Encoding an input $x_i \in G$:

- For $1 \leq i \leq n$, $\mathrm{Enc}_i'(x_i, R_i) = M_i = R_{i,x_i}$ for $x_i \in \{0,1\}$.

Decoding h: Compute the product of the encodings, that is, $h(x_1, \cdots, x_n) = M_1 \cdot M_2 \cdot \ldots \cdot M_n$

$$\begin{array}{lrcl} (* & M_1 \cdot M_2 \cdot \ldots \cdot M_n & = & L \cdot \sigma^{x_1} \cdot r_1 \cdot r_1^{-1} \cdot \sigma^{x_2} \cdot r_2 \cdot \ldots \cdot r_{n-1}^{-1} \cdot \sigma^{x_n} \cdot f \\ & & = & L \cdot \sigma^{\sum_{i=1}^n x_i} \cdot f \quad = \quad f(\sigma^{\sum_{i=1}^n x_i}(0)) \quad = \quad f(\sum_{i=1}^n x_i) \qquad *) \end{array}$$

Figure 9: The NIMPC Π_0' for a fixed function $h : G^n \to \{0,1\}$ such that $h(X_1, \ldots, X_n) = f(\sum_{i=1}^n X_i)$.

and $c_i \triangleq \sigma^{d_i}$. Furthermore, define $r' = \phi(r)$ by

$$r' \triangleq (c_1 \cdot r_1, \ldots, c_{n-1} \cdot r_{n-1}).$$

It is clear that ϕ defined this way is a bijection. For a fixed r let us use the abbreviated notation $(M_{\overline{T}}, R_T) \triangleq (M_{\overline{T}}, R_T)(x_{\overline{T}}, r_T)$ and $(M'_{\overline{T}}, R'_T) \triangleq (M_{\overline{T}}, R_T)(x'_{\overline{T}}, r'_T)$. It is left to show that under this definition $M_i = M'_i$ for every $i \in \overline{T}$, and $R_i = R'_i$ for every $i \in T$.

- $i = n$:
  - If $n \in T$, then $c_{n-1} = \sigma^a$. Therefore, $R'_{n,0} = (r_{n-1})^{-1} \cdot c_{n-1}^{-1} \cdot f = (r_{n-1})^{-1} \cdot \sigma^{-a} \cdot f = (r_{n-1})^{-1} \cdot f = R_{n,0}$ and $R'_{n,1} = (r_{n-1})^{-1} \cdot \sigma^{-a+1} \cdot f = (r_{n-1})^{-1} \cdot \sigma \cdot f = R_{n,1}$.
  - If $n \in \overline{T}$, we have $c_{n-1} = \sigma^{x'_n - x_n + a}$. Therefore, $M'_n = (r_{n-1})^{-1} \cdot c_{n-1}^{-1} \cdot \sigma^{x'_n} \cdot f = (r_{n-1})^{-1} \cdot \sigma^{x_n - a} \cdot f = (r_{n-1})^{-1} \cdot \sigma^{x_n} \cdot f = M_n$.
- $1 < i < n$:
  - If $i \in T$, we have $c_i = c_{i-1}$. Therefore, $R'_{i,0} = (r_{i-1})^{-1} \cdot c_i \cdot c_{i-1}^{-1} \cdot r_i = (r_{i-1})^{-1} \cdot r_i = R_{i,0}$. Similarly, $R'_{i,1} = R_{i,1}$.
  - If $i \in \overline{T}$, we have $c_i = \sigma^{x_i - x'_i} c_{i-1}$. Therefore, $M'_i = (r_{i-1})^{-1} \cdot c_{i-1}^{-1} \cdot \sigma^{x'_i} \cdot c_i \cdot r_i = (r_{i-1})^{-1} \cdot \sigma^{x_i} \cdot r_i = M_i$.
- $i = 1$:

– If $1 \in T$, we have $c_1 = \sigma^0$. Therefore, $R'_{1,0} = L \cdot r_1 = R_{1,0}$ and $R'_{1,1} = R_{1,1}$ by a similar calculation.
– If $1 \in \overline{T}$, we have $c_1 = \sigma^{\sum_{j \in \overline{T} \setminus \{1\}}(x'_j - x_j)+a} = \sigma^{\sum_{j \in \overline{T}}(x'_j - x_j)-(x'_1 - x_1)+a} = \sigma^{x_1 - x'_1}$. Therefore, $M'_1 = L \cdot \sigma^{x'_1} \cdot r'_1 = L \cdot \sigma^{x'_1} \cdot c_1 \cdot r_1 = L \cdot \sigma^{x_1} \cdot r_1 = M_1$.

□

Remark 7.5. *In* NIMPC Π'_0*, each input can either be* 0 *or* 1*. However, in Theorem 7.4 we claim that this protocol computes a function* $h : G^n \to \{0,1\}$*. This reflects the fact that even if the honest parties have boolean inputs, a coalition* T *can compute* $h|_{\overline{T}, x_{\overline{T}}}$ *on any point in* $G^{|T|}$*. We can fix* NIMPC Π'_0 *to enable non-boolean inputs as follows. First, for* $2 \leq i \leq n-1$ *the encoding* M_i *of an input* x_i *is computed as follows:* $M_i = (R_{i,1} \cdot R_{i,1}^{-1})^{x_i} \cdot R_{i,0} = (r_{i-1}^{-1} \cdot \sigma \cdot r_{i-1})^{x_i} \cdot r_{i-1}^{-1} \cdot r_i = r_{i-1}^{-1} \cdot \sigma^{x_i} \cdot r_i$*. Second, the encoding* M_1 *of an input* x_1 *is* $L \cdot \sigma^{x_i} r_1$*. To be able to compute this encoding, the correlated randomness* R_1 *is* r_1*. Finally,* $M_n = r_{n-1}^{-1} \cdot \sigma^{x_i} \cdot f$*. To be able to compute this encoding, the correlated randomness* R_n *is* $r_{n-1}^{-1} \cdot r_n, r_{n-1}^{-1} \cdot \sigma \cdot r_n$*, and* $r_n \cdot f$*, where* r_n *is a random permutation in* G_S*. It can be checked that Theorem 7.4 remains valid under these changes.*

7.2 The protocol for $\mathcal{H}^G_{G,\ldots,G}$

Let us begin by clarifying why NIMPC Π'_0 is not a good NIMPC for $\mathcal{H}^G_{G,\ldots,G}$. The encoding of x_n is of the form $M_n = r_{n-1} \cdot \sigma^{x_n} \cdot f$. Observe that M_n is a function $M_n : G \to \{0,1\}$ with $|M_n^{-1}(0)| = |f^{-1}(0)|$. Therefore, the protocol is not 0-robust: An outside observer will learn in addition to $f(\mathrm{sum}(x))$ the value $|f^{-1}(0)|$. Let $G' \triangleq \mathbb{Z}_{2n+2}$. We fix this by artificially extending a function $f : G \to \{0,1\}$ to a function $f_2 : G' \to \{0,1\}$ with $|f_2^{-1}(0)| = n+1$, e.g., by mapping the first $n+1-|f^{-1}(0)|$ elements of $\{n+1, \ldots, 2n+1\}$ to zero, and the rest to one. In this case we cannot replace the permutation σ by a cyclic shift on G' as this would allow for example a non-empty set T of size 1 to compute the values of $h|_{\overline{T}, x_{\overline{T}}}$ on the point $n+1$. To overcome this problem, we replace the permutation σ by the permutation $\sigma_2 \in G'_S$ defined by $\sigma_2(i) = i+1 \pmod{(n+1)}$ for $1 \leq i \leq n$ and $\sigma_2(i) = i$ for $n+1 \leq i \leq 2n+1$ That is, σ_2 cyclically shifts the elements $\{0, \ldots, n\}$ as σ does, and fixes the elements $\{n+1, \ldots, 2n+1\}$. The important point is that σ_2, and therefore also any power of σ_2, cannot move an element $i \in \{0, \ldots, n\}$ into the range $\{n+1, \ldots, 2n+1\}$.

Theorem 7.6. *The* NIMPC Π''_0*, described in Figure 10, is fully robust for* $\mathcal{H}^G_{G,\ldots,G}$*.*

Proof. The proof is very similar to the previous proof of Theorem 5.2. Fix a subset $T \subseteq [n]$. Fix any two functions $h, h' \in \mathcal{H}^G_{G,\ldots,G}$ where $h(X_1, \ldots, X_n) = f(\sum_{i=1}^n X_i)$ and $h'(X_1, \ldots, X_n) = f'(\sum_{i=1}^n X_i)$ for some functions $f, f' : G \to \{0,1\}$. Fix any inputs $x_{\overline{T}}, x'_{\overline{T}}$ for the variables outside of T such that $h|_{\overline{T}, x_{\overline{T}}} \equiv h'|_{\overline{T}, x'_{\overline{T}}}$. Denote by $(R_T, M_T)(r)$ the view of the set T in the protocol when the inputs $x_{\overline{T}}$ and the function h are used, and the vector of permutations chosen in the preprocessing stage is $r = (r_1, \ldots, r_{n-1}) \in {G'_S}^{n-1}$. Similarly, denote by $(R'_T, M'_T)(r')$ the view of the set T in the protocol when the inputs $x'_{\overline{T}}$ and the function h' are used, and the vector of permutations chosen in the preprocessing stage is $r' = (r'_1, \ldots, r'_{n-1}) \in {G'_S}^{n-1}$.

We present a bijection $\phi : {G'_S}^{n-1} \to {G'_S}^{n-1}$ such that for all $r \in {G'_S}^{n-1}$,

$$(R_T, M_T)(r) = (R'_T, M'_T)(\phi(r)).$$

Correlated randomness generation:

- Choose random permutations $r_1, \ldots, r_{n-1} \in G'_S$.
- Let $R_1 = (R_{1,0}, R_{1,1}) = (L \cdot r_1, L \cdot \sigma_2 \cdot r_1)$, where $L : G' \to G'$ is the identically zero function.
- For $1 \leq i \leq n-1$, let $R_i = (R_{i,0}, R_{i,1}) = (r_{i-1}^{-1} \cdot r_i, r_{i-1}^{-1} \cdot \sigma_2 \cdot r_i)$.
- Let $R_n = (R_{n,0}, R_{n,1}) = (r_{n-1}^{-1} \cdot f_2, r_{n-1}^{-1} \cdot \sigma_2 \cdot f_2)$.

Encoding an input $x_i \in G$:

- For $1 \leq i \leq n$, $\mathrm{Enc}'_i(x_i, R_i) = M_i = R_{i,x_i}$ for $x_i \in \{0,1\}$.

Decoding h: Compute the product of the encodings, that is, $h(x_1, \cdots, x_n) = M_1 \cdot M_2 \cdot \ldots \cdot M_n$

$$\begin{array}{rcl} (* \quad M_1 \cdot M_2 \cdot \ldots \cdot M_n & = & L \cdot \sigma_2^{x_1} \cdot r_1 \cdot r_1^{-1} \cdot \sigma_2^{x_2} \cdot r_2 \cdot \ldots \cdot r_{n-1}^{-1} \cdot \sigma_2^{x_n} \cdot f_2 \\ & = & L \cdot \sigma_2^{\sum_{i=1}^n x_i} \cdot f_2 \quad = \quad f_2(\sigma_2^{\sum_{i=1}^n x_i}(0)) \quad = \quad f_2(\sum_{i=1}^n x_i) \qquad *) \end{array}$$

Figure 10: The NIMPC Π''_0 for the class $\mathcal{H}^G_{G,\ldots,G}$.

This suffices to prove the theorem.

We first deal separately with the case that T is empty, i.e., 0-robustness: Let $\tau \in G'_S$ be a permutation such that $\tau \cdot f_2 = f'_2$ and $\tau(\mathrm{sum}(x')) = \mathrm{sum}(x)$. Such τ exists as $|f_2^{-1}(0)| = |f_2'^{-1}(0)|$ and $f_2(\mathrm{sum}(x)) = f'_2(\mathrm{sum}(x'))$. For $1 \leq i \leq n-1$, we define a permutation $c_i \in G'_S$ by

$$c_i \triangleq \sigma_2^{\sum_{j>i} x'_j} \cdot \tau \cdot \sigma_2^{-\sum_{j>i} x_j}.$$

We define $r' = \phi(r)$ by

$$r' \triangleq (c_1 \cdot r_1, c_2 \cdot r_2, \ldots, c_{n-1} \cdot r_{n-1}).$$

We need to show that under this defintion $M_i(r) = M'_i(r')$ for every $i \in [n]$ and $r \in {G'_S}^{n-1}$. We fix $r \in {G'_S}^{n-1}$ and use the abbreviated notation $M_i \triangleq M_i(r)$ and $M'_i \triangleq M'_i(r')$.

- $i = n$: $c_{n-1} = \sigma_2^{x'_n} \cdot \tau \cdot \sigma_2^{-x_n}$. Therefore,

  $$M'_n = (r'_{n-1})^{-1} \cdot \sigma_2^{x'_n} \cdot f'_2 = (r_{n-1})^{-1} \cdot c_{n-1}^{-1} \cdot \sigma_2^{x'_n} f'_2 = (r_{n-1})^{-1} \cdot \sigma_2^{x_n} \cdot \tau^{-1} \cdot f'_2 = (r_{n-1})^{-1} \cdot \sigma_2^{x_n} \cdot f_2 = M_n.$$

- $1 < i < n$: We have

  $$c_{i-1}^{-1} \cdot \sigma_2^{x'_i} \cdot c_i = \sigma_2^{\sum_{j>i-1} x_j} \cdot \tau^{-1} \cdot \sigma_2^{-\sum_{j>i-1} x'_j} \cdot \sigma_2^{x'_i} \cdot \sigma_2^{\sum_{j>i} x'_j} \cdot \tau \cdot \sigma_2^{-\sum_{j>i} x_j} = \sigma_2^{x_i}.$$

  Therefore, $M'_i = (r'_{i-1})^{-1} \cdot \sigma_2^{x'_i} \cdot r'_i = (r_{i-1})^{-1} \cdot \sigma_2^{x_i} \cdot r_i = M_i$.

- $i = 1$: $L \cdot \sigma_2^{x'_1} \cdot c_1 = L \cdot \sigma_2^{\sum_j x'_j} \cdot \tau \cdot \sigma_2^{-\sum_{j>1} x_j}$. As $(\sigma_2^{\sum_j x'_j} \cdot \tau)(0) = \sigma_2^{\sum_j x_j}(0) = \sum_j x_j$, and $L \equiv 0$, $L \cdot \sigma_2^{x'_1} \cdot c_1$ is the constant function x_1. Therefore, as functions $L \cdot \sigma_2^{x'_1} \cdot c_1 = L \cdot \sigma_2^{x_1}$. We thus have $M'_1 = L \cdot \sigma_2^{x'_1} \cdot r'_1 = L \cdot \sigma_2^{x'_1} \cdot c_1 \cdot r_1 = L \cdot \sigma_2^{x_1} \cdot r_1 = M_1$.

We now deal with the case $T \neq \emptyset$. In this case, $|f^{-1}(0)| = |f'^{-1}(0)|$ – otherwise the functions $h|_{\overline{T}, x_{\overline{T}}}$ and $h|_{\overline{T}, x'_{\overline{T}}}$ would differ. Therefore, for $b \in \{n+1, \ldots, 2n\}$, $f_2(b) = f'_2(b)$, as we defined f_2 on the range $\{n+1, \ldots, 2n\}$ in a way that depends only on $|f^{-1}(0)|$. We stress that in what follows $a_1 + a_2$ for $a_1, a_2 \in \{0, \ldots, n\}$ refers to addition in G, i.e., $\bmod\, n+1$ rather than addition in G'. Define

$$a \triangleq \sum_{j \in \overline{T}} (x_j - x'_j).$$

A crucial point is that $f_2 = \sigma_2^a \cdot f'_2$ and $f'_2 = \sigma_2^{-a} \cdot f'_2$. This is true for any $b \in \{0, \ldots, n\}$

$$f_2(b) = f_2(\sum_{i \in \overline{T}} x'_i + (b - \sum_{i \in \overline{T}} x'_i)) = f'_2(\sum_{i \in \overline{T}} x_i + (b - \sum_{i \in \overline{T}} x'_i)) = f'_2(a+b) = f'_2(\sigma_2^a(b)) = (\sigma_2^a \cdot f'_2)(b).$$

On the other hand, for $b \in \{n+1, \ldots, 2n+1\}$, $f'_2(b) = f_2(b) = \sigma_2^a \cdot f_2(b)$.

For $1 < i \leq n-1$, we define

$$d_i \triangleq \sum_{j > i, j \in \overline{T}} (x'_j - x_j) + a,$$

and $c_i \triangleq \sigma_2^{d_i}$. We define $r' = \phi(r)$ by

$$r' \triangleq (c_1 \cdot r_1, \ldots, c_{n-1} \cdot r_{n-1}).$$

It is clear that ϕ defined this way is a bijection. It is left to show that under this defintion $M_i = M'_i$ for every $i \in \overline{T}$, and $R_i = R'_i$ for every $i \in T$.

- $i = n$:
  - If $n \in T$, then $c_{n-1} = \sigma_2^a$. Therefore, $R'_{n,0} = (r_{n-1})^{-1} \cdot c_{n-1}^{-1} \cdot f'_2 = (r_{n-1})^{-1} \cdot \sigma_2^{-a} \cdot f'_2 = (r_{n-1})^{-1} \cdot f_2 = R_{n,0}$ and $R'_{n,1} = (r_{n-1})^{-1} \cdot \sigma_2^{1-a} \cdot f'_2 = (r_{n-1})^{-1} \cdot \sigma_2 \cdot f_2 = R_{n,1}$.
  - If $n \in \overline{T}$, then $c_{n-1} = \sigma_2^{x'_n - x_n + a}$. Therefore, $M'_n = (r_{n-1})^{-1} \cdot c_{n-1}^{-1} \cdot \sigma_2^{x'_n} \cdot f'_2 = (r_{n-1})^{-1} \cdot \sigma_2^{x_n - a} \cdot f'_2 = r_{n-1}^{-1} \cdot \sigma_2^{x_n} \cdot f_2 = M_n$.
- $1 < i < n$:
  - If $i \in T$, then $c_i = c_{i-1}$. Therefore, $R'_{i,0} = (r_{i-1})^{-1} \cdot c_{i-1}^{-1} \cdot c_i \cdot r_i = (r_{i-1})^{-1} \cdot r_i = R_{i,0}$ and $R'_{i,1} = R_{i,1}$ by a similar calculation.
  - If $i \in \overline{T}$, then $c_{i-1} = \sigma_2^{x'_i - x_i} \cdot c_i$. Therefore, $M'_i = (r_{i-1})^{-1} \cdot c_{i-1}^{-1} \cdot \sigma_2^{x'_i} \cdot c_i \cdot r_i = (r_{i-1})^{-1} \cdot \sigma_2^{x_i} \cdot r_i = M_i$.
- $i = 1$:
  - If $1 \in T$, then $c_1 = \sigma_2^0$. Therefore, $R'_{1,0} = L \cdot r_1 = R_{1,0}$ and similarly $R'_{1,1} = R_{1,1}$.
  - If $1 \in \overline{T}$, then $c_1 = \sigma_2^{x_1 - x'_1}$. Therefore, $M'_1 = L \cdot \sigma_2^{x'_1} c_1 \cdot r_1 = L \cdot \sigma_2^{x_1} \cdot r_1 = M_1$.

□

References

[1] B. Applebaum, Y. Ishai and E. Kushilevitz. Cryptography in NC^0. In *Proc. FOCS 2004*, pp. 166-175.

[2] B. Applebaum, Y. Ishai, E. Kushilevitz, and B. Waters. Encoding Functions with Constant Online Rate or How to Compress Garbled Circuits Keys. In *Proc. CRYPTO 2013 (2)*, pp. 166-184.

[3] B. Barak, S. Garg, Y. Tauman Kalai, O. Paneth, and A. Sahai. Protecting obfuscation against algebraic attacks. In *Proc. EUROCRYPT 2014*, pp. 221-238.

[4] B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S. P. Vadhan, and Ke Yang On the (im)possibility of obfuscating programs. In *Proc. CRYPTO 2001*, pp. 1-18.

[5] D.M Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in NC^1. In *Proc. STOC '86*, pp. 1-5.

[6] M. Bellare, V.T. Hoang, and P. Rogaway. Foundations of garbled circuits. In *Proc. ACM CCS 2012*, pp. 784-796.

[7] D. Boneh, A. Sahai, and B. Waters. Functional encryption: Definitions and challenges. In *Proc. TCC 2011*, pp. 253-273.

[8] Z. Brakerski and G. N. Rothblum. Virtual black-box obfuscation for all circuits via generic graded encoding. In *Proc. TCC 2014*, pp. 1-25.

[9] R. Canetti. Security and composition of multiparty cryptographic protocols. *Journal of Cryptology*, 13(1):143-202, 2000.

[10] Y. Dodis, J. Katz, S. Xu, and M. Yung. Key-insulated public key cryptosystems. In *Proc. EUROCRYPT 2002*, pp. 65-82.

[11] U. Feige, J. Kilian, M. Naor. A minimal model for secure computation. In *Proc. STOC 1994*, pp. 554-563.

[12] S. Garg, C. Gentry, S. Halevi, M. Raykova, A. Sahai, and B. Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *Proc. FOCS 2013*, pp. 40-49.

[13] O. Goldreich. **Foundations of Cryptography** - *Volume 2*. Cambridge University Press, 2004.

[14] S. Goldwasser, S. D. Gordon, V. Goyal, A. Jain, J. Katz, F.-H. Liu, A. Sahai, E. Shi, and H.-S. Zhou. Multi-input functional encryption. In *Proc. Eurocrypt 2014*, pp. 578-602.

[15] S. Goldwasser, A. B. Lewko, and D. A. Wilson. Bounded-collusion IBE from key homomorphism. In *Proc. TCC 2012*, pp. 564-581.

[16] S. Goldwasser and G. N. Rothblum. On best-possible obfuscation. In *Proc. TCC 2007*, pp. 194-213.

[17] S. Gorbunov, V. Vaikuntanathan, and H. Wee. Functional encryption with bounded collusions via multi-party computation. In *Proc. CRYPTO 2012,* pp. 162-179.

[18] S. D. Gordon, T. Malkin, M. Rosulek, H. Wee. Multi-party computation of polynomials and branching programs without simultaneous interaction. In *EUROCRYPT 2013*, pp. 575-591.

[19] V. Goyal, O. Pandey, A. Sahai, and B. Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *Proc. ACM CCS 2006*, pp. 89-98.

[20] V. Goyal, Y. Ishai, A. Sahai, R. Venkatesan, and A. Wadia. Founding cryptography on tamper-proof hardware tokens. In *Proc. TCC 2010*, pp. 308-326.

[21] S. Halevi, Y. Lindell, and B. Pinkas. Secure computation on the web: Computing without simultaneous interaction. In *Proc. CRYPTO 2011*, pp. 132-150.

[22] D. Harnik, Y. Ishai, E. Kushilevitz, and J. B. Nielsen. OT-Combiners via Secure Computation. In *Proc. TCC 2008*, pp. 393-411.

[23] D. Harnik, J. Kilian, M. Naor, O. Reingold, and A. Rosen. On Robust Combiners for Oblivious Transfer and Other Primitives. In *Proc. EUROCRYPT 2005*, pp. 96-113.

[24] Y. Ishai and E. Kushilevitz. Private simultaneous messages protocols with applications. In *ISTCS 1997*, pp. 174-184.

[25] Y. Ishai and E. Kushilevitz. Randomizing polynomials: A new representation with applications to round-efficient secure computation. In *FOCS 2000*, pp. 294-304.

[26] Y. Ishai, and E. Kushilevitz. Perfect constant-round secure computation via perfect randomizing polynomials. In *Proc. ICALP 2002*, pp. 244-256.

[27] Y. Ishai, E. Kushilevitz, S. Meldgaard, C. Orlandi, A. Paskin-Cherniavsky. On the power of correlated randomness in secure computation. In *Proc. TCC 2013*, pp. 600-620.

[28] Y. Ishai, E. Kushilevitz, R. Ostrovsky, and A. Sahai. Cryptography with constant computational overhead. In *Proc. STOC 2008*, pp. 433-442.

[29] J. Kilian. Founding cryptography on oblivious transfer. In *Proc. STOC 1988*, pp. 20-31.

[30] M. Naor, B. Pinkas, and R. Sumner. Privacy preserving auctions and mechanism design. In *Proc. ACM EC 1999*, pp 129-139.

[31] A. O'Neill. Definitional issues in functional encryption. *IACR Cryptology ePrint Archive* 2010: 556.

[32] A. Sahai and B. Waters. Fuzzy identity-based encryption. In *Proc. EUROCRYPT 2005*, pp. 457-473.

[33] A.C.C. Yao. How to generate and exchange secrets. In *Proc.* 27*th FOCS 1986*, pp 162-167.

A General non-interactive MPC

In this section we extend the treatment of NIMPC to functionalities that may deliver different outputs to different parties as well as to the case of security against malicious parties.

We consider protocols involving n parties, $P_1, \ldots, P_n$, with a correlated randomness setup. That is, we assume an offline preprocessing phase that provides each party P_i with a random input r_i. (This preprocessing can be implemented either using a trusted dealer, by an interactive offline protocol involving the parties themselves, or by an interactive MPC protocol involving a smaller number of specialized servers.) In the online phase, each party P_i, on input (x_i, r_i), may send a single message $m_{i,j}$ to each party P_j. (There is

no need to assume secure or authenticated channels, as these can be easily implemented using a correlated randomness setup.)

Let f be a deterministic functionality mapping inputs $(x_1, \ldots, x_n)$ to outputs $(y_1, \ldots, y_n)$. We define security of an NIMPC protocol for such f using the standard "real vs. ideal" paradigm (cf. [9, 13]), except that the ideal model is relaxed to capture the best achievable security in the non-interactive setting.

Concretely, for NIMPC in the semi-honest security model we relax the standard ideal model for evaluating f by first requiring all parties to send their inputs to the functionality f, then having f deliver the outputs to the honest parties, and finally allowing the adversary to make repeated oracle queries to f with the same fixed honest inputs. (Similar relaxations of the ideal model were previously considered in other contexts, such as fairness and concurrent or resettable security.) In the malicious security model, one should further relax the ideal model in order to additionally take into account the adversary's capability of rushing[3] (namely, correlating its messages with the messages obtained from honest parties). Furthermore, if the communication model only allows for point-to-point communication (with no broadcast), the ideal model should account for the fact that the adversary can use different inputs when sending messages to different honest parties. In the relaxed ideal model, first the honest parties send their inputs to f, then the adversary can repeatedly make oracle calls as above, and finally the adversary can decide on the actual inputs to f (a set of inputs for each honest party) that determine the outputs of honest parties.

Given a t-robust NIMPC protocol (according to Definition 2.2) for each of the n outputs of f, a t-secure protocol for f can be obtained in a straightforward way. In the honest-but-curious model, it suffices to run n independent instances of the protocol described in Section 2.1, where in the i-th instance P_i acts both as a standard party and as the external server P_0. In the malicious model, the correlated randomness setup uses an unconditional one-time MAC to authenticate each of the possible messages sent from P_i to P_j. This is feasible when the input domain of each party is small. In the general case, we can make use of an NIMPC protocol for a functionality f' with a bigger number of parties which is identical to f except for taking a single input bit from each party. Such a functionality f' can be securely realized by a protocol Π' as described above, and then f can be realized by a protocol Π in which each party emulates the corresponding parties in Π'.

[3] If some mechanism is available for ensuring that the adversary's messages are independent of the honest parties' messages, this relaxation is not needed.