Non-Interactive Secure Multiparty Computation*

Amos Beimel[1], Ariel Gabizon[2], Yuval Ishai[2], Eyal Kushilevitz[2], Sigurd Meldgaard[3], and Anat Paskin-Cherniavsky[4]

[1] Dept. of Computer Science, Ben Gurion University, Beer Sheva, Israel
amos.beimel@gmail.com
[2] Computer Science Department, Technion, Haifa, Israel
{arielga,yuvali,eyalk}@cs.technion.ac.il
[3] Google Aarhus, Denmark
stm@cs.au.dk
[4] Computer Science Department, UCLA, Los Angeles, CA, USA
anps83@gmail.com

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:

* 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.

J.A. Garay and R. Gennaro (Eds.): CRYPTO 2014, Part II, LNCS 8617, pp. 387–404, 2014.

– **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).
– **Symmetric functions.** Every symmetric function $f : X^n \to Y$, where 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. [24], for further motivation). The above attack fails in this model, since a party can no

[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.

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.

Recent work on multi-input functional encryption [13] 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 [11], 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 [26]. While the security analysis in the case of abelian groups is straightforward (see Example 6), the analysis for the general case turns out to be more involved and is deferred to the full version of this paper. We note that this result *cannot* be combined with Barrington's Theorem [4] 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 [15] (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 $h : X^n \to Y$, where 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 $h : X_1 \times \cdots \times X_n \to Y$ of the form $h(x_1, \ldots, x_n) = f(\sum_{i=1}^{n} x_i)$ for an arbitrary function $f : G \to Y$ (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 is given in Section 4, and a formal treatment of the general case appears in the full version of this paper. In the full version we also describe a more efficient variant of the construction for the case $t = 1$.

Inadequacy of Existing Techniques. On the negative side, in the full version 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) [20,17]. 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 [30,5] or randomized encodings of functions [22,1] 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 the full version of this paper, 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 the full version of this paper, to get a 1-robust solution with the same message size for $n \approx 1400$ 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 [3]. Moreover, it follows from the results of [15] 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 [29,18,28,6] that was very recently studied in [13] 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 [13] 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 [11], 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 [10,21]. 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 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* [25] or *projective garbling schemes* [5].) 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 the full version of this paper, we show that applying this methodology to different PSM protocols and garbling schemes from the literature typically fails to offer even 1-robustness. We also show a case where this methodology does work – using Kilian's PSM protocol for computing the iterated group product [26] yields a fully robust protocol.

Bounded-Collusion Functional Encryption. In the related context of (single-input) functional encryption, Gorbunov et al. [16] 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 [9,14]. 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 [16,9,14] 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 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 $(1 \leq i \leq n)$, 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 (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 1). Robustness requires that they learn no other information.

Definition 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 $0 \le t \le n$*, we say that* Π *is* t-robust *if it is* T*-robust for every* $T \subseteq [n]$ *of size* $|T| \le t$*. 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 4. 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 [10,21]. 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 9, 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 5. **(Statistical and computational variants.)** In this work, we consider NIMPC protocols with perfect security, as captured by Definition 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 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 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 [11,13]. If the simulator is restricted to probabilistic polynomial time, we get the stronger "virtual black-box" variant to which the impossibility results from [3] apply and one can only hope to get general positive results in a generic model [7,2] or using tamper-proof hardware [19]. We note, however, that the latter impossibility results only apply to function classes that are rich enough to implement pseudo-random functions. In particular, they do not apply to efficiently learnable classes for which obfuscation is trivial. NIMPC is meaningful and nontrivial even in the latter case.

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

Example 6. 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 7. *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 $\mathcal{H}^m = \{h : \mathcal{X} \to \{0,1\}^m \,|\, h = h_1 \circ \ldots \circ h_m,\ h_i \in \mathcal{H}\}$ 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, $1 \leq i \leq n$, 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, $1 \leq i \leq n$, 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 (though with exponential communication complexity). It will be useful to first present such a protocol for *indicator* functions. For reasons to be clarified later on, it will be convenient to include the zero-function.

Definition 8. *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 9. *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 $s = \sum_{i=1}^{n} d_i$. We describe a non-interactive protocol, in the output-server model. Fix a function $h \in \mathcal{H}$ that we want to compute. The protocol $\mathrm{P}(\Pi_{\mathrm{ind}})(h)$ is as follows.

Preprocessing: 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 that 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 $m_{n,a_n} = -\sum_{i=1}^{n-1} m_{i,a_i}$). For $i \in [n]$, we send the vectors $\{m_{i,b}\}_{b\in\mathcal{X}_i}$ to P_i as the correlated randomness.

Sending messages: For $i \in [n]$, player P_i (on input x_i) sends to P_0 the message $M_i \triangleq m_{i,x_i}$.

Computing $h(x_1, \ldots, x_n)$: P_0 outputs 1 if $\sum_{i=1}^{n} M_i = \mathbf{0}$ and 0 otherwise.

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 messages $M_{\overline{T}}$ 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 constrains described above, that is, all vectors are independent with the exception that $\sum_{i=1}^{n} m_{i,x_i} = 0$.

As for communication complexity, each party P_i receives $d_i \leq d$ binary vectors of length $s \leq dn$ in the preprocessing stage and sends one of them as a message. Hence, at most $d^2 n$ bits. □

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

Theorem 10. *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 $\mathcal{X} \triangleq \mathcal{X}_1 \times \cdots \times \mathcal{X}_n$. Let $\mathcal{H}$ be the set of all functions $h : \mathcal{X} \to \{0,1\}^m$. There exists an* NIMPC *protocol Π for $\mathcal{H}$ with communication complexity $|\mathcal{X}| \cdot m \cdot d^2 \cdot n$.*

Proof. Let $\Pi_{\text{ind}} = (\text{Gen}, \text{Enc}, \text{Dec})$ be the NIMPC for $\mathcal{H}_{\text{ind}}$, described in Lemma 9. Fix $h \in \mathcal{H}$. Assume for simplicity that $m = 1$ (see Claim 7). Protocol $\text{P}(\Pi)(h)$ is as follows.

Preprocessing:

- Let $I \subseteq \mathcal{X}$ be the set of ones of h (i.e., $I = h^{-1}(1)$). For each $a \in I$, let $r^a = (r_1^a, \ldots, r_n^a) \leftarrow \text{Gen}(h_a)$. For $a \in \mathcal{X} \setminus I$, let $r^a \leftarrow \text{Gen}(h_0)$.
- Choose a random permutation π of $\mathcal{X}$ and define a matrix R, where $R_{i,b} \triangleq r_i^{\pi(b)}$ for $i \in [n]$ and $b \in \mathcal{X}$. Send to P_i the random strings $(R_{i,b})_{b \in \mathcal{X}}$ (that is, the ith row of R).

Sending Messages: Define a matrix M, where $M_{i,b} \triangleq \text{Enc}_i(x_i, R_{i,b})$ for every $i \in [n]$ and $b \in \mathcal{X}$. Each P_i sends to P_0 the message $M_i \triangleq (M_{i,b})_{b \in \mathcal{X}}$.

Computing h: Server P_0 outputs 1 if for some $b \in \mathcal{X}$, $\text{Dec}(M_{1,b}, \ldots, M_{n,b}) = 1$. Otherwise, it outputs 0.

Correctness: Fix $x = (x_1, \ldots, x_n) \in \mathcal{X}$. The server returns 1 if and only if $\text{Dec}(M_{1,b}, \ldots, M_{n,b}) = 1$ for some $b \in \mathcal{X}$, namely, if and only if $\text{Dec}(\text{Enc}_1(x_1, R_{1,b}), \ldots, \text{Enc}_n(x_n, R_{n,b})) = 1$. This happens if and only if $\text{Dec}(\text{Enc}_1(x_1, R_1^a), \ldots, \text{Enc}_n(x_n, R_n^a)) = 1$ for $a = \pi(b)$. By the correctness of Π_{ind} and the protocol description, the above happens if and only if $h_a(x_1, \ldots, x_n) = 1$ for some $a \in I$, that is, if and only if $h(x_1, \ldots, x_n) = 1$. Communication Complexity is obtained by applying Π_{ind} for $|\mathcal{X}|$ times.

Robustness: Fix $T \subseteq [n]$ and $x_{\overline{T}} \in \mathcal{X}_{\overline{T}}$. We wish to simulate the distribution $(M_{\overline{T}}, R_T)$ given $h|_{\overline{T}, x_{\overline{T}}}$. We can think of this distribution as being composed of rows, where each row b is of the form $(M_{\overline{T}}^a, r_T^a)$ for $a = \pi(b)$ for some $b \in \mathcal{X}$, where the permutation π is random.

Observation 11. *If $a_{\overline{T}} = x_{\overline{T}}$ and $h(a) = 1$ then this row was generated for the function h_a, and if $a_{\overline{T}} = x_{\overline{T}}$ and $h(a) = 0$ then this row was generated for h_0. Finally, if $a_{\overline{T}} \neq x_{\overline{T}}$, then this row is distributed as if it was generated for h_0.*

We next construct a simulator Sim_T for the protocol $\text{P}(\Pi)$ on function h. Simulator Sim_T uses the simulator $\text{Sim}_T^{\Pi_{\text{ind}}}$ – the simulator for set T from protocol $\text{P}(\Pi_{\text{ind}})$ of Lemma 9. 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 $S_{x_T} = \text{Sim}_T^{\Pi_{\text{ind}}}(h_{x_T})$ (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 $\text{Sim}_T^{\Pi_{\text{ind}}}(h_0)$ 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 $\text{Sim}_T^{\Pi_{\text{ind}}}$. 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 $\text{Sim}_T^{\Pi_{\text{ind}}}$ and Observation 11. □

Remark 12. 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

$\mathcal{H}' \subset \mathcal{H}_{\text{ind}}$ of indicator functions. For this set of functions, we would get an NIMPC with communication complexity $|\mathcal{H}'| \cdot m \cdot \text{poly}(d, n)$ (rather than the $|\mathcal{X}| \cdot m \cdot \text{poly}(d, n)$ 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 protocol for symmetric functions. In fact, this result is a corollary of a more general result on NIMPC for *abelian group programs*. We next define abelian programs and symmetric functions and formally state our results. The proofs of these results appear in the full version of this paper.

Definition 13. *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 $f : G \to \{0,1\}$.*

Definition 14. *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 poly(n)-size and t is constant.

Theorem 15. *Let t be a positive integer and G an abelian group of size m. Let $S_1, \ldots, S_n$ be subsets of G. Let $d \triangleq \max_{i \in [n]} |S_i|$. Then, there is a t-robust* NIMPC *protocol for $\mathcal{H}^G_{S_1,\ldots,S_n}$ with communication complexity $O(d^{t+2} \cdot n^{t+2} \cdot m^3)$.*

Corollary 16. *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 $O(d^{t+2} \cdot n^{t+3d-1})$. In particular, for the case of a boolean $h : \{0,1\}^n \to \{0,1\}$, the communication complexity is $O(2^t \cdot n^{t+5})$.*

In the rest of this section, we give a high level overview of the construction, focusing for simplicity on the case $t = 1$.

4.1 Group Extension

Recall that a boolean function $h : \{0,1\}^n \to \{0,1\}$ is symmetric if and only if there exists a function $f : \{0, \ldots, n\} \to \{0,1\}$ such that $h(x_1, \ldots, x_n) = f(\sum_{i=1}^n x_i)$. We start by considering a relaxation of the problem where the players are allowed to choose their inputs from a larger domain, which is a group: namely, instead of having an input $x_i \in \{0,1\}$, we allow each player P_i to have an input $x_i \in \{0, \ldots, n\}$, which can be thought of as an element of the group $G \triangleq \mathbb{Z}_{n+1}$. Given a boolean symmetric function $h : \{0,1\}^n \to \{0,1\}$,

where $h(x_1, \ldots, x_n) = f(\sum_{i=1}^{n} x_i)$, we extend h in the natural way to a function $h : G^n \to \{0,1\}$, that is, $h(x_1, \ldots, x_n) = f(\sum_{i=1}^{n} x_i)$, where the sum is of elements of G. The first step of our construction is a fully robust NIMPC protocol for the set $\mathcal{H}$ 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.

To obtain the NIMPC protocol for $\mathcal{H}$, we would like to use the PSM protocol from [21] 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 fails miserably 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 $\mathbf{1}$), 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 [21]. 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 [21] 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 Limiting the Inputs of One Player

The previous subsection gives an NIMPC 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 $\Pi_0(h)$.[2]

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 $\{0,1\}^{|T|}$. 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

[2] An important point is that only the preprocessing stage of protocol Π_0 actually depends on h, but we ignore these subtleties here.

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 α_0 and α_1 (where $\alpha_b : G \to \{0,1\}$). Let us call these masked versions g_0 and g_1. Specifically, $g_j(X_2, \ldots, X_n) \triangleq h_j(X_2, \ldots, X_n) \oplus \alpha_j(\sum_{i=2}^{n} X_i)$, for $j \in \{0,1\}$. In the preprocessing stage, we give the masking functions α_0 and α_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 α_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 $g_{x_1}(x_2, \ldots, x_n) = h(x_1, \ldots, x_n) \oplus \alpha_{x_1}(\sum_{i=2}^{n} x_i)$. It also knows α_{x_1}, which was sent by P_1. So now, to "unmask" $h(x_1, \ldots, x_n)$ using α_{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, and combine the solutions to the two problems above.

4.3 A Secret Sharing Composition

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 $f : G \to \{0,1\}$. It is easy to see that Π_1 can work for any function $h'(X_1, \ldots, X_n)$ of this form. 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 $h(x_1, \ldots, x_{i-1}, a, x_{i+1}, \ldots, x_n)$.

References

1. Applebaum, B., Ishai, Y., Kushilevitz, E.: Cryptography in NC^0. In: Proc. FOCS 2004, pp. 166–175 (2004)
2. Barak, B., Garg, S., Kalai, Y.T., Paneth, O., Sahai, A.: Protecting Obfuscation against Algebraic Attacks. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 221–238. Springer, Heidelberg (2014)
3. Barak, B., Goldreich, O., Impagliazzo, R., Rudich, S., Sahai, A., Vadhan, S., Yang, K.: On the (Im)possibility of Obfuscating Programs. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 1–18. Springer, Heidelberg (2001)
4. Barrington, D.M.: Bounded-Width Polynomial-Size Branching Programs Recognize Exactly Those Languages in NC^1. In: Proc. STOC 1986, pp. 1–5 (1986)
5. Bellare, M., Hoang, V.T., Rogaway, P.: Foundations of garbled circuits. In: Proc. ACM CCS 2012, pp. 784–796 (2012)
6. Boneh, D., Sahai, A., Waters, B.: Functional Encryption: Definitions and Challenges. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 253–273. Springer, Heidelberg (2011)
7. Brakerski, Z., Rothblum, G.N.: Virtual Black-Box Obfuscation for All Circuits via Generic Graded Encoding. In: Lindell, Y. (ed.) TCC 2014. LNCS, vol. 8349, pp. 1–25. Springer, Heidelberg (2014)
8. Canetti, R.: Security and composition of multiparty cryptographic protocols. Journal of Cryptology 13(1), 143–202 (2000)
9. Dodis, Y., Katz, J., Xu, S., Yung, M.: Key-Insulated Public Key Cryptosystems. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 65–82. Springer, Heidelberg (2002)
10. Feige, U., Kilian, J., Naor, M.: A Minimal Model for Secure Computation. In: Proc. STOC 1994, pp. 554–563 (1994)
11. Garg, S., Gentry, C., Halevi, S., Raykova, M., Sahai, A., Waters, B.: Candidate Indistinguishability Obfuscation and Functional Encryption for All Circuits. In: Proc. FOCS 2013, pp. 40–49 (2013)
12. Goldreich, O.: Foundations of Cryptography, vol. 2. Cambridge University Press (2004)
13. Goldwasser, S., et al.: Multi-input Functional Encryption. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 578–602. Springer, Heidelberg (2014)
14. Goldwasser, S., Lewko, A.B., Wilson, D.A.: Bounded-Collusion IBE from Key Homomorphism. In: Cramer, R. (ed.) TCC 2012. LNCS, vol. 7194, pp. 564–581. Springer, Heidelberg (2012)
15. Goldwasser, S., Rothblum, G.N.: On Best-Possible Obfuscation. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 194–213. Springer, Heidelberg (2007)
16. Gorbunov, S., Vaikuntanathan, V., Wee, H.: Functional Encryption with Bounded Collusions via Multi-party Computation. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 162–179. Springer, Heidelberg (2012)

17. Gordon, S.D., Malkin, T., Rosulek, M., Wee, H.: Multi-party Computation of Polynomials and Branching Programs without Simultaneous Interaction. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 575–591. Springer, Heidelberg (2013)
18. Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-based encryption for fine-grained access control of encrypted data. In: Proc. ACM CCS 2006, pp. 89–98 (2006)
19. Goyal, V., Ishai, Y., Sahai, A., Venkatesan, R., Wadia, A.: Founding Cryptography on Tamper-Proof Hardware Tokens. In: Micciancio, D. (ed.) TCC 2010. LNCS, vol. 5978, pp. 308–326. Springer, Heidelberg (2010)
20. Halevi, S., Lindell, Y., Pinkas, B.: Secure Computation on the Web: Computing without Simultaneous Interaction. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 132–150. Springer, Heidelberg (2011)
21. Ishai, Y., Kushilevitz, E.: Private simultaneous Messages Protocols with Applications. In: ISTCS 1997, pp. 174–184 (1997)
22. Ishai, Y., Kushilevitz, E.: Randomizing Polynomials: A New Representation with Applications to Round-Efficient Secure Computation. In: FOCS 2000, pp. 294–304 (2000)
23. Ishai, Y., Kushilevitz, E.: Perfect Constant-Round Secure Computation via Perfect Randomizing Polynomials. In: Widmayer, P., Triguero, F., Morales, R., Hennessy, M., Eidenbenz, S., Conejo, R. (eds.) ICALP 2002. LNCS, vol. 2380, pp. 244–256. Springer, Heidelberg (2002)
24. Ishai, Y., Kushilevitz, E., Meldgaard, S., Orlandi, C., Paskin-Cherniavsky, A.: On the Power of Correlated Randomness in Secure Computation. In: Sahai, A. (ed.) TCC 2013. LNCS, vol. 7785, pp. 600–620. Springer, Heidelberg (2013)
25. Ishai, Y., Kushilevitz, E., Ostrovsky, R., Sahai, A.: Cryptography with constant computational overhead. In: Proc. STOC 2008, pp. 433–442 (2008)
26. Kilian, J.: Founding Cryptography on Oblivious Transfer. In: Proc. STOC 1988, pp. 20–31 (1988)
27. Naor, M., Pinkas, B., Sumner, R.: Privacy Preserving Auctions and Mechanism Design. In: Proc. ACM Conference on Electronic Commerce 1999, pp. 129–139 (1999)
28. O'Neill, A.: Definitional Issues in Functional Encryption. IACR Cryptology ePrint Archive 2010: 556
29. Sahai, A., Waters, B.: Fuzzy Identity-Based Encryption. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 457–473. Springer, Heidelberg (2005)
30. Yao, A.C.C.: How to Generate and Exchange Secrets. In: Proc. 27th FOCS 1986, pp. 162–167 (1986)

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. [8,12]), 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). 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 that determine the outputs of honest parties.

Given a t-robust NIMPC protocol (according to Definition 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.