Ad Hoc PSM Protocols: Secure Computation Without Coordination

Amos Beimel[1(✉)], Yuval Ishai[2,3], and Eyal Kushilevitz[2]

[1] Department of Computer Science, Ben Gurion University, Beer Sheva, Israel
amos.beimel@gmail.com
[2] Department of Computer Science, Technion, Haifa, Israel
{yuvali,eyalk}@cs.technion.ac.il
[3] Department of Computer Science, UCLA, Los Angeles, USA

Abstract. We study the notion of *ad hoc secure computation*, recently introduced by Beimel et al. (ITCS 2016), in the context of the *Private Simultaneous Messages* (PSM) model of Feige et al. (STOC 2004). In ad hoc secure computation we have n parties that may potentially participate in a protocol but, at the actual time of execution, only k of them, whose identity is *not* known in advance, actually participate. This situation is particularly challenging in the PSM setting, where protocols are non-interactive (a single message from each participating party to a special output party) and where the parties rely on pre-distributed, correlated randomness (that in the ad-hoc setting will have to take into account all possible sets of participants).

We present several different constructions of ad hoc PSM protocols from standard PSM protocols. These constructions imply, in particular, that efficient information-theoretic ad hoc PSM protocols exist for NC^1 and different classes of log-space computation, and efficient computationally-secure ad hoc PSM protocols for polynomial-time computable functions can be based on a one-way function. As an application, we obtain an information-theoretic implementation of *order-revealing encryption* whose security holds for two messages.

We also consider the case where the actual number of participating parties t may be larger than the minimal k for which the protocol is designed to work. In this case, it is unavoidable that the output party learns the output corresponding to each subset of k out of the t participants. Therefore, a "best possible security" notion, requiring that this will be the *only* information that the output party learns, is needed. We present connections between this notion and the previously studied notion of *t-robust PSM* (also known as "non-interactive MPC"). We show that constructions in this setting for even simple functions (like AND or threshold) can be translated into non-trivial instances of program obfuscation (such as *point function obfuscation* and *fuzzy point function obfuscation*, respectively). We view these results as a negative indication that protocols with "best possible security" are impossible to realize efficiently in the information-theoretic setting or require strong assumptions in the computational setting.

J.-S. Coron and J.B. Nielsen (Eds.): EUROCRYPT 2017, Part III, LNCS 10212, pp. 580–608, 2017.
DOI: 10.1007/978-3-319-56617-7_20

1 Introduction

The notion of *ad hoc secure computation* was recently put forward in [4]. In the ad-hoc secure computation problem, there are n parties that may potentially take part in a secure computation protocol. At the time that the protocol is executed, some k of these n parties actually participate in the execution. The goal is to design (efficient) protocols that can work for *every* set of k parties S, without knowing the set of participants in advance. As a concrete example, think of a voting application, where n parties are registered to the elections but only k of them (the identity of which becomes known only in real time) end up participating in the vote.

In most standard secure computation models, the ad-hoc nature of the protocol does not pose a significant challenge: the participating parties can interact with each other and use a standard general-purpose secure protocol to perform the computation. The problem is most challenging in situations where preprocessing or setup are required, or where interaction is limited. In the extreme, where non-interactive secure protocols are needed, the single message sent by each party P_i cannot depend on the messages of other parties, whose identities are not even known to P_i.

A simple model for non-interactive secure computation is the *Private Simultaneous Messages* (PSM) model of [14,17]. In this model, there are n parties $P_1, \ldots, P_n$ and a special party called the *referee*. Before the input is known, the parties are given correlated randomness[1] $(r_1, \ldots, r_n)$. In the online phase, each party P_i gets an input x_i and sends a single message m_i, depending on x_i and r_i, to the referee. Based on the n received messages, the referee should be able to compute the value of a pre-determined function f on the input $x = (x_1, \ldots, x_n)$, namely $f(x)$. The security requires that the referee learns no additional information about x. It is known that PSM protocols exist for every finite function f [14] and *efficient* PSM protocols exist for every function in NC^1 and for classes of functions defined by different types of (polynomial-size) branching programs [14,17]. In a computational setting, efficient PSM protocols for all polynomial-time computable functions can be based on one-way functions by using Yao's garbled circuit construction [14,20]. The simplicity of the PSM model makes it an attractive candidate for a complexity theoretic study (see, e.g., [2]) and its limited interaction pattern makes it useful in applications, such as minimizing the round-complexity of secure protocols in the standard point-to-point model (see, e.g., [18]).

In this paper, we study the ad hoc version of the PSM model, where the referee receives messages from a subset of size k out of the n parties. We assume that the parameter k is known in advance, but the parties are not aware of the identity of other participating parties. Before describing our results in detail (in Sect. 1.1), we discuss some possible variants of the question. First, the original

[1] Both in the original PSM model and in its ad-hoc variant, it suffices for the parties to share a source of *common* randomness that is unknown to the referee. The use of more general correlated randomness can help reduce the randomness complexity.

PSM model was mainly studied in the information-theoretic security setting. In this work, we consider both the information-theoretic variant and the computational variant. In fact, the computational version of ad hoc PSM was first considered in [4], where it was shown that such protocols can be constructed based on the existence of a weak form of *Multi-Input Functional Encryption* (MIFE) [15], a primitive whose general realization is essentially equivalent to the existence of general indistinguishability obfuscation.

Second, the problem of ad hoc PSM is significantly different in the case where we are guaranteed that exactly k parties will send messages vs. the case where possibly more than k parties may participate. Most of the time, we will assume that only k parties send messages and that this guarantee is assured by some other mechanism, such as a public bulletin board reporting the current participant count, or an anonymous communication medium that hides all information except the fact that a message has been sent. On the other hand, in a setting where a set S of more than k parties may send messages in the protocol, the referee unavoidably may compute the function f on any subset $S' \subset S$ of size k and learn the value $f(x_{S'})$. Therefore, in this case, our security notion is a "best possible security" definition, requiring that this will be the *only* information that the referee learns in the protocol. This can be formalized either using a strong simulation-based definition or a weaker indistinguishability-based definition.

Finally, it will be convenient and, in fact, very natural in the ad-hoc setting to think of f as a symmetric function. Most of our results do not rely on this and can be extended to even allow the computed function to depend on the set of participants S, i.e. to output $f_S(x_S)$.

1.1 Our Results

Let us start by demonstrating our results using a concrete task of computing the SUM function. In this case, each party P_i is given an input $x_i \in \mathbb{Z}_m$ and the goal is to compute their sum $\sum_{i\in[n]} x_i$ (all additions in this example are mod m). A standard PSM protocol for SUM works by giving the parties randomness $r_1, \ldots, r_n \in_R \mathbb{Z}_m$ subject to the constraint that $\sum_{i\in[n]} r_i = 0$. Then, each party P_i, sends a message $m_i = x_i + r_i$ to the referee who outputs $\sum_{i\in[n]} m_i = \sum_{i\in[n]} x_i$, as needed. Moreover, due to the choice of the r_i's, no additional information about the inputs is revealed to the referee.

In the ad-hoc version of the problem, we wish to compute the SUM of any set S of k parties that may send messages in the protocol. One option is to prepare, for each potential set S of size k, independent randomness $r_1^S, \ldots, r_k^S$ that is random subject to their sum being 0 and proceed by P_i sending a message (using the corresponding randomness r_j^S), for each set S to which it belongs. While this solution works, its randomness complexity and communication complexity are proportional to $\binom{n}{k}$, which is much more than what we are shooting for. Instead, we describe an *efficient* solution for this problem.

In our ad hoc PSM protocol, the randomness consists of values $r_1, \ldots, r_n \in_R \mathbb{Z}_m$ subject to the constraint that $\sum_{i\in[n]} r_i = 0$, as in the original PSM protocol.

In addition, we produce shares $\{r_{j,i}\}_{i\in[n]}$, for each r_j, using a k-out-of-n secret sharing scheme (e.g., Shamir). The randomness given to each P_i consists of its r_i and its shares of all other random values; that is $\{r_{j,i}\}_{j\in[n]\setminus\{i\}}$. Then, each party P_i that participates in the protocol (i.e., $i \in S$) sends as its message the value $m_i = x_i + r_i$, as well as all its shares. The referee sums up all the m_i's that it got from the k participants, as well as all the values r_j, for $j \notin S$, that it can reconstruct from the k shares that it received for each such r_j, to get $\sum_{i\in S}(x_i + r_i) + \sum_{i\notin S} r_i = \sum_{i\in S} x_i$, as needed. In terms of security, each r_i, for $i \in S$, remains hidden as the referee receives exactly $k-1$ shares for these random elements. In fact, the view of the referee can be simulated from its view in the original PSM, where parties P_j, for $j \notin S$, have input $x_j = 0$. Also note that if, say, $k + 1$ parties send messages then the referee learns all inputs. However, for the SUM function, the best possible security definition (that allows the referee to learn the output on all subsets of size k) allows to recover all $k + 1$ inputs in most cases (at least when $\gcd(k, m) = 1$).

Next, we describe in some detail our main results. The first question that we ask (in Sect. 3) is whether the existence of a standard k-party PSM computing a function f guarantees the existence of a k-out-of-n ad hoc PSM protocol for f. We first prove the existence of an inefficient transformation of this kind but that has an overhead of $\binom{n}{k}$. While this transformation may be useful for the case where the number of parties is small (and also proves the existence of an ad hoc PSM protocol for every function f), our aim is to get an efficient transformation (i.e., with $\mathrm{poly}(n)$ overhead). We next present such a transformation that works whenever f is symmetric, and is efficient whenever k is small (essentially, $2^{O(k)} \log n$). When $k = O(1)$, the overhead is as small as $O(\log n)$ (this construction relies on perfect hash families, and its complexity depends on the size of such families of functions from $[n]$ to $[k]$). The fact that the complexity of each party grows only logarithmically with the number of parties will be useful for the application discussed in Sect. 6.

Then, in Sect. 4, we ask whether an ad hoc PSM protocol for f can be constructed more efficiently based on a standard PSM protocol for a *related* (n-argument) function g. We prove that this is indeed possible, while incurring only $O(n)$ overhead over the complexity of the protocol for g. Moreover, the computational complexity of g is closely related to that of f in computational models for which efficient PSM protocols are known (e.g., if f is in NC^1 then so is g, and if f has a polynomial-size branching program then so does g). This implies efficient ad hoc PSM protocols for branching programs in the information-theoretic setting and for circuits in the computational setting, where the latter relies on the existence of a one-way function. In addition, in Sect. 5, we present an explicit ad hoc PSM for the equality function.

In Sect. 6, we show an interesting application of ad hoc PSM protocols. Specifically, we show how to construct an order revealing encryption (ORE) from an ad hoc PSM protocol for the "greater-than" function. An order revealing encryption, presented in [10] as a generalization of order preserving encryption [1], is a private-key encryption that enables computing the order between two messages

(that is, checking if $m_1 < m_2$, $m_1 = m_2$, or $m_1 > m_2$), given their encryptions (without knowing the private key), but does not disclose any additional information. We construct information-theoretically secure order revealing encryption that is secure as long as only two messages are encrypted. In our construction, we use an ad hoc PSM protocol constructed in Sect. 3 with $n = 2^\lambda$ parties (where λ is the security parameter), relying on the fact that the complexity of each party in the protocol from Sect. 3 only grows logarithmically with the number of parties. We also give a solution for a bigger number of messages, but with a weaker security guarantee.

The above results refer to the case where exactly k parties send messages in the protocol. We next examine (in Sect. 7) the case where more than k (but up to some threshold t) parties may send messages. In this case, as discussed above, one needs to settle for a "best possible security" definition. We extend the above transformation from standard PSM to ad hoc PSM to this case, showing that it is possible to construct a PSM protocol for f with best possible security from a so-called "t-robust PSM" protocol [5] for a related function g', incurring only $O(n)$ overhead. A t-robust PSM is a protocol where up to t parties may collaborate with the referee in trying to learn information about the inputs of other parties. In this case, it is always possible for the adversary to get the output of f on many inputs, by replacing the messages of the collaborating parties with messages that correspond to other inputs. Therefore, for such protocols also one may only hope for a "best possible security". Our results connect these two best possible security settings (in both directions). It should be noted, however, that efficient t-robust PSM protocols in the information-theoretic setting are currently known only for limited families of functions, and limited values of t [5].

In Sect. 8, we examine the possibility of constructing efficient PSM protocols with best possible security, in the *computational* setting. (The naive transformation of Sect. 3 shows that it is possible to get best possible security even in the information theoretic case but without efficiency.) The two-way connection with t-robust PSM already implies a two-way connection between this problem and general-purpose obfuscation. However, it is not clear a-priori that the connection has relevance in the case of *simple* functions. We give evidence that efficient ad-hoc PSM protocols with best possible security are difficult to design even for very simple functions. For instance, a protocol for a threshold function implies a construction of *fuzzy point function obfuscation* [7], a primitive whose only known constructions rely on multilinear maps. In fact, even a protocol for the AND function, gives a construction of *point function obfuscation*.

2 The Setting

We consider a network of n parties, denoted $P_1, \ldots, P_n$, and a referee; Each party P_i holds an input x_i, and the parties hold correlated random strings $r_1, \ldots, r_n$. We want to execute a protocol, where only a subset of the parties $S \subseteq \{P_1, \ldots, P_n\}$ participates in the protocol, each one of them sends a single message to the referee. If exactly k parties participate and send messages then,

based on these k messages, the referee should be able to compute the value $f(x_S)$ but learn no other information about x_S, where $x_S = (x_i)_{i \in S}$. The subset S of participating parties is selected in an ad hoc manner and, in particular, the participating parties are not aware of each other. This is the main source of difficulty in this model. The referee itself necessarily learns the set of participants S (as it receives messages directly from the participants; avoiding this would require the use of anonymous communication). We often assume that f is symmetric; while this is a natural assumption in such a setting, most of our constructions can handle a much more general requirement, where the computed function itself may also depend on the set of participants S (i.e., the output is $f_S(x_S)$). We call the above model ad hoc PSM. We formalize this notion below starting with information-theoretic secure protocols.

Definition 2.1 (Ad hoc PSM: Syntax and correctness). *Let $\mathcal{X}, \mathcal{R}_1, \ldots, \mathcal{R}_n$, $\mathcal{M}$ and Ω be finite domains. A* k-out-of-n ad hoc PSM *for a function $f : \mathcal{X}^k \to \Omega$ is a triplet $\Pi = (\textsc{Gen}, \textsc{Enc}, \textsc{Dec})$ where*

- $\textsc{Gen}()$ *is a randomized function with output in $\mathcal{R}_1 \times \cdots \times \mathcal{R}_n$,*
- $\textsc{Enc}$ *is an n-tuple of deterministic functions $(\textsc{Enc}_1, \ldots, \textsc{Enc}_n)$, where $\textsc{Enc}_i : \mathcal{X} \times \mathcal{R}_i \to \mathcal{M}$,*
- $\textsc{Dec} : \binom{[n]}{k} \times \mathcal{M}^k \to \Omega$ *is a deterministic function satisfying the following correctness requirement: for any $S = \{i_1, \ldots, i_k\} \subseteq [n]$ and $x_S = (x_{i_1}, \ldots, x_{i_k}) \in \mathcal{X}^k$,*

$$\Pr\left[\begin{array}{l} r = (r_1, \ldots, r_n) \leftarrow \textsc{Gen}() : \\ \quad \textsc{Dec}(S, \textsc{Enc}_{i_1}(x_{i_1}, r_{i_1}), \ldots, \textsc{Enc}_{i_k}(x_{i_k}, r_{i_k})) = f(x_S) \end{array}\right] = 1.$$

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

Definition 2.2 (Ad hoc PSM: Perfect and statistical security). *We say that an* ad hoc PSM *protocol Π for f is* k-secure *if:*

- *For every set $S \in \binom{[n]}{k}$, given the messages of S, the referee does not get any additional information beyond the value of $f(x_S)$. Formally, there exists a randomized function* $\textsc{Sim}$ *(a "simulator") such that, for every $S \in \binom{[n]}{k}$ and for every $x_S \in \mathcal{X}^k$, we have $\textsc{Sim}(S, f(x_S)) \equiv M_S$, where M_S are the messages defined by $R \leftarrow \textsc{Gen}()$ and $M_S = (\textsc{Enc}_i(x_i, R_i))_{i \in S}$.*
- *For every $k' < k$ and every set $S' \in \binom{[n]}{k'}$, given the messages of S', the referee does not get any information on the input $x_{S'}$. Formally, there exists a randomized function* $\textsc{Sim}$ *such that, for every $k' < k$, every $S' \in \binom{[n]}{k'}$ and every $x_{S'} \in \mathcal{X}^{k'}$, we have $\textsc{Sim}(S') \equiv M_{S'}$, where $M_{S'}$ is defined as above.*

We say that an ad hoc PSM *protocol Π for f is* (k, ϵ)-(statistically) secure *if there exists a randomized function* $\textsc{Sim}$ *such that for every $S \in \binom{[n]}{k}$ and for every $x_S \in \mathcal{X}^k$,*

$$\mathrm{dist}(\textsc{Sim}(S, f(x_S)), M_S) \leq \epsilon.$$

Similarly, for every $S' \in \binom{[n]}{k'}$, we have $\mathrm{dist}(\textsc{Sim}(S'), M_{S'}) \leq \epsilon$.

Example 2.3. We next describe a very simple ad hoc PSM protocol for computing the difference of inputs for $k = 2$ parties; that is, for $i < j$ we want to compute $f(x_i, x_j) = x_i - x_j$ (over some Abelian group G). The randomness generation chooses a random element $r \in_R G$ and gives it to each party. The message of each P_i on input x_i is $m_i = x_i + r$. The output of the referee on messages m_i, m_j from parties P_i, P_j (where $i < j$) is $m_i - m_j = x_i - x_j$. The simulator SIM proving the security of the protocol gets as an input a set $S = \{i, j\}$ and $\Delta = x_i - x_j$. It chooses a random value r and outputs $r, r - \Delta$. Notice that both the messages in the protocol and the output of SIM is a pair (a, b), where a is uniformly distributed in G and b is $a - f(x_i, x_j)$, thus the simulation is as required.

While in most parts of this paper we will assume that at most k parties send messages, we next consider the scenario that parties execute an ad hoc PSM protocol and a set T of more than k parties sends messages. Clearly, for every $S \subset T$ of size k, the referee can compute the output of f on the inputs of S. Thus, the *best possible security requirement* is that the referee does not learn any additional information.

Definition 2.4. *An* ad hoc PSM *protocol Π for f is* (k, t, ϵ)-secure *if there exists a randomized function* SIM *such that, for every $t' \leq t$, every $T \in \binom{[n]}{t'}$ and every $x_T \in \mathcal{X}^t$,*

$$\mathrm{dist}(\mathrm{SIM}\left(T, (f(x_S))_{S \subseteq T, |S|=k}\right), M_T) \leq \epsilon.$$

An ad hoc PSM *protocol Π for f is* (k, t)-secure *if it is* $(k, t, 0)$*-secure.*

Remark 2.5. In Sect. 3.2, for every function f, we construct an *inefficient* (k, n) ad hoc PSM protocol. It follows from [16] (together with our result that (k, n)-secure ad hoc PSM protocols imply obfuscation) that efficient (k, n)-secure ad hoc PSM protocols for every function in NC^1 do not exist unless the polynomial-time hierarchy collapses. This impossibility result does not rule out, for example, efficient $(2, n)$-secure ad hoc PSM protocols for every function in NC^1 (and beyond) or efficient $(k, k + 1)$-secure ad hoc PSM protocols. We do not know if such efficient ad hoc PSM protocols exist.

For some functions the (k, t)-security requirement is not interesting as the best possible security already reveals a lot of information. For other functions this notion is interesting.

Example 2.6. Let f be the 2-party addition function over a field whose characteristic is not 2. Suppose that a referee got messages from parties P_1, P_2, P_3 in an ad hoc PSM for f, thus, it can compute the sum of every two inputs of these parties, namely, $x_1 + x_2 = s_{1,2}, x_1 + x_3 = s_{1,3}$, and $x_2 + x_3 = s_{2,3}$. From these sums it can compute the inputs, e.g., $x_1 = 2^{-1}(s_{1,2} + s_{1,3} - s_{2,3})$.

Example 2.7. Consider the $n/2$-party AND function and an input where the value of exactly $n/2$ of the input variables is 1. Assume that the referee gets messages from the n parties for this input. If the referee does not know the set of variables whose value is 1, then it will not be able to efficiently determine it.

We next consider *computationally-secure* ad hoc PSM protocols. In such protocols we want all algorithms to be efficient. We start by defining their syntax.

Definition 2.8 (Computational ad hoc PSM: Syntax). *Let $n(\lambda)$, $k(\lambda)$, and $\ell(\lambda)$ be polynomials, and $F = \{f_\lambda : (\{0,1\}^{\ell(\lambda)})^{k(\lambda)} \to \{0,1\}^*\}_{\lambda\in\mathbb{N}}$ be a collection of functions. A protocol $\Pi = (\textsc{Gen}, \textsc{Enc}, \textsc{Dec})$ is a $(k(\lambda), n(\lambda))$-computational* ad hoc PSM *protocol for F if*

- *Algorithm $\textsc{Gen}(1^\lambda)$ is a polynomial time algorithm that generates $n(\lambda)$ random strings (for $n(\lambda)$ parties).*
- *Algorithms $\textsc{Enc}$ and $\textsc{Dec}$ run in polynomial time.*
- *There exists a negligible function* negl() *such that for any $\lambda \in \mathbb{N}$, any $S \subseteq [n(\lambda)]$, and any $x_S = (x_i)_{i\in S} \in \left(\{0,1\}^{\ell(\lambda)}\right)^{k(\lambda)}$,*

$$\Pr\left[r \leftarrow \textsc{Gen}(1^\lambda) : \textsc{Dec}\left(S, (\textsc{Enc}_i(x_i, r_i))_{i\in S}\right) = f_\lambda(x_S)\right] \geq 1 - \mathrm{negl}(\lambda).$$

We next present three definitions of security for computational ad hoc PSM protocols. The first definition is simulation-based and it applies to k-security (i.e., to the scenario where exactly k parties send their messages).

Definition 2.9 (Computational ad hoc PSM: Simulation-based security). *Let $n(\lambda)$, $k(\lambda)$, and $\ell(\lambda)$ be polynomials, and $F = \{f_\lambda : (\{0,1\}^{\ell(\lambda)})^{k(\lambda)} \to \{0,1\}^*\}_{\lambda\in\mathbb{N}}$ be a collection of functions. We say that an* ad hoc PSM *protocol $(\textsc{Gen}, \textsc{Enc}, \textsc{Dec})$ is $k(\lambda)$-simulation-based secure if there exists a probabilistic non-uniform polynomial algorithm $\textsc{Sim}$ whose inputs are 1^λ and the value of f such that the two ensemble of distributions*

$$\left((m_i)_{i\in S} : \begin{array}{l} r \leftarrow \textsc{Gen}(1^\lambda), \\ \forall_{i\in S}\, m_i \leftarrow \textsc{Enc}(x_i, r_i) \end{array}\right) \begin{array}{l} \lambda \in \mathbb{N}, S \in \binom{[n(\lambda)]}{k(\lambda)}, \\ (x_i)_{i\in S} \in \left(\{0,1\}^{\ell(\lambda)}\right)^{k(\lambda)} \end{array}$$

and

$$\left(\textsc{Sim}(1^\lambda, f_\lambda((x_i)_{i\in S}))\right) \begin{array}{l} \lambda \in \mathbb{N}, S \in \binom{[n(\lambda)]}{k(\lambda)}, \\ (x_i)_{i\in S} \in \left(\{0,1\}^{\ell(\lambda)}\right)^{k(\lambda)} \end{array}$$

are indistinguishable in polynomial time.

Simulation-based security is a strong requirement that cannot be achieved for computational ad hoc PSM protocols with best possible security (see discussion in [3]). Thus, for such protocols, we define weaker security – virtual black-box (VBB) security, where the adversary can output only one bit and that uses indistinguishability-based security. To simplify the notation, we only consider the case where $t = n(\lambda)$.

Definition 2.10 (Computational ad hoc PSM: Virtual black-box Security). *Let $n(\lambda)$, $k(\lambda)$, and $\ell(\lambda)$ be polynomials, and $F = \{f_\lambda : (\{0,1\}^{\ell(\lambda)})^{k(\lambda)} \to$*

$\{0,1\}^*\}_{\lambda\in\mathbb{N}}$ *be a collection of functions. We say that an* ad hoc PSM *protocol* $(\textsc{Gen}, \textsc{Enc}, \textsc{Dec})$ *is* $(k(\lambda), n(\lambda))$*-VBB-secure if, for every non-uniform polynomial time adversary* $\mathcal{A}$ *that outputs one bit, there exists a non-uniform probabilistic polynomial time algorithm* Sim *and a negligible function* $\mathrm{negl}(\lambda)$ *such that for every* $\lambda \in \mathbb{N}$*, every* $S \in \binom{[n(\lambda)]}{k(\lambda)}$*, and every* $x_1, \ldots, x_{n(\lambda)} \in (\{0,1\}^{\ell(\lambda)})^{n(\lambda)}$

$$\left|\Pr\left[\mathcal{A}(1^\lambda, m_1 \ldots, m_{n(\lambda)}) = 1\right] - \Pr\left[\textsc{Sim}^{f_\lambda}(1^\lambda) = 1\right]\right| \leq \mathrm{negl}(\lambda),$$

where

- *The first probability is over the messages generated in the following way: first compute* $r \leftarrow \textsc{Gen}(1^\lambda)$ *and then* $m_i \leftarrow \textsc{Enc}(x_i, r_i)$*, for every* $i \in [n(\lambda)]$*.*
- *The second probability is over the randomness of the simulator, which has access to an oracle* f_λ *that on query* $S \in \binom{[n(\lambda]}{k(\lambda)}$ *returns* $f_\lambda(x_S)$*.*

Definition 2.11 (Computational ad hoc PSM: Indistinguishability-based security). *Let* $n(\lambda)$*,* $k(\lambda)$*, and* $\ell(\lambda)$ *be polynomials, and* $F = \{f_\lambda : (\{0,1\}^{\ell(\lambda)})^{k(\lambda)} \to \{0,1\}^*\}_{\lambda\in\mathbb{N}}$ *be a collection of functions. Consider the following game between an adversary* $\mathcal{A}$ *and a challenger:*

1. *The adversary on input* 1^λ *chooses a set* $T \subseteq [n(\lambda)]$ *and two inputs* $(x_i^0)_{i\in T}$ *and* $(x_i^1)_{i\in T}$ *and sends* T *and the two inputs to the challenger.*
2. *The challenger chooses a uniformly random bit* $b \in \{0,1\}$ *and computes* $(r_1, \ldots, r_n) \leftarrow \textsc{Gen}(1^\lambda)$ *and* $m_i \leftarrow \textsc{Enc}(x_i^b, r_i)$*, for every* $i \in T$*. It then sends* $(m_i)_{i\in T}$ *to the adversary.*
3. *The adversary outputs a bit* b'*.*

The adversary wins the game if $b' = b$ *and* $f_\lambda(x_S^0) = f_\lambda(x_S^1)$*, for every* $S \subseteq T$ *such that* $|S| = k(\lambda)$*.*

We say that a computational ad hoc PSM *protocol* $(\textsc{Gen}, \textsc{Enc}, \textsc{Dec})$ *is a* $(k(\lambda), n(\lambda))$*-indistinguishably-secure* ad hoc PSM *protocol for* F *if, for every non-uniform polynomial-time adversary* $\mathcal{A}$*, the probability that* $\mathcal{A}$ *wins is at most* $1/2 + \mathrm{negl}(\lambda)$ *for some negligible function* negl*.*

Our default model in the rest of the paper, unless explicitly mentioned, is (k, k)-secure ad hoc PSM protocol with perfect security.

3 Ad hoc PSM Protocols for a Function f from a PSM for f

In this section we present a k-out-of-n ad hoc PSM protocol for any function f by applying transformations to k-party PSM protocols for the same f.

3.1 From a PSM for f to an n-out-of-n ad hoc PSM for f

In an n-party PSM protocol for f, if all n parties send messages, then the referee learns $f(x_1, \ldots, x_n)$ and does not learn any additional information. In an n-out-of-n ad hoc PSM protocol, there is an additional requirement: if less than n parties send messages, then the referee should learn no information. The definition of PSM does not imply the latter requirement.

Example 3.1. Consider a function which returns x_1. In a PSM for this function, P_1 can send its input, while in an ad hoc PSM protocol, this should not be done.

In many PSM protocols this additional requirement does hold. Furthermore, for many functions, the requirement for smaller sets of active participants follows from the security requirements of the PSM.

Example 3.2. Consider a PSM protocol for the AND function. If the input of P_n is 0, then the output of AND is 0 for every input for $P_1, \ldots, P_{n-1}$. Thus, the messages $m_1, \ldots, m_n$ of $P_1, \ldots, P_n$ are equally distributed when $x_n = 0$, for every input for $P_1, \ldots, P_{n-1}$. Since the messages of $P_1, \ldots, P_{n-1}$ are independent of x_n, the messages of parties $P_1, \ldots, P_{n-1}$ are equally distributed for every input for these parties. I.e., in any PSM protocol for AND the referee does not learn any information from the messages of $P_1, \ldots, P_{n-1}$ or, similarly, any other set of less than n active participants.

Lemma 3.3. *If there is an n-party PSM protocol Π for f with randomness complexity* $\mathrm{Rand}(\Pi)$ *and communication complexity* $\mathrm{Comm}(\Pi)$*, then there is an n-out-of-n* ad hoc PSM *protocol for f with randomness complexity* $\mathrm{Rand}(\Pi) + n \cdot \mathrm{Comm}(\Pi)$ *and communication complexity* $n \cdot \mathrm{Comm}(\Pi)$.

Proof. We construct an ad hoc PSM protocol Π_{ah} for f from the PSM protocol Π for f, as follows.

Randomness generation:

- Generate randomness for the PSM protocol Π; denote $r_1, \ldots, r_n$ the generated randomness of $P_1, \ldots, P_n$, respectively.
- Choose n uniformly random strings $u_1, \ldots, u_n$, each of length $\mathrm{Comm}(\Pi)$.
- Share each u_j, for $j \in [n]$, using an n-out-of-n secret sharing scheme; let $u_{j,i}$ be the i-th share of u_j.
- The randomness of P_i in Π_{ah} is $r_i, (u_{j,i})_{j \in [n]}$.

Message generation:

- Let m_i be the messages of P_i in Π on input x_i and randomness r_i.
- The message of P_i in Π_{ah} is $m_i \oplus u_i$ and $(u_{j,i})_{j \in [n]}$.

If the n parties send their messages, then the referee can reconstruct u_i for every $i \in [n]$, compute m_i, and reconstruct $f(x_1, \ldots, x_n)$ using the decoding algorithm of Π.

The security of Π_{ah} when n parties send messages follows from the security of Π (as the strings $u_1, \ldots, u_n$ are random). When less than n parties send their messages, the referee gets less than n shares of each u_i, thus, these strings act as random pads and the referee learns no information. □

3.2 A Naive ad hoc PSM Protocol for Any f

In this section we show how to construct, given a (standard) k-party PSM protocol for f, a k-out-of-n ad hoc PSM protocol for f that is n-secure (that is, if a set T of at least k parties send their messages, then the referee can compute the output of f on any subset of inputs of size k and learns no additional information).

Theorem 3.4. *If there is a k-party PSM protocol Π for f with randomness complexity* $\mathrm{Rand}(\Pi)$ *and communication complexity* $\mathrm{Comm}(\Pi)$*, then there is a (k,n)-secure* ad hoc PSM *protocol for f with randomness complexity* $\binom{n}{k} \cdot (\mathrm{Rand}(\Pi) + n \cdot \mathrm{Comm}(\Pi))$ *and communication complexity* $\binom{n}{k} \cdot n \cdot \mathrm{Comm}(\Pi)$.

Proof. Let Π_{ah} be the k-out-of-k ad hoc PSM protocol constructed from Π in Lemma 3.3. We construct an ad hoc PSM protocol Π' for f as follows:

Randomness generation:

- For each set $S \in \binom{[n]}{k}$, independently generate randomness for Π_{ah} and give this randomness to the parties in S.

Message generation:

- Each party P_i sends its message in protocol Π_{ah}, associated with the set S', for every S' of size k such that $i \in S'$.

Function reconstruction by the referee: For a set S of k participating parties, the referee (only) uses the messages of the parties in S of the PSM Π_{ah} for S to reconstruct $f(x_S)$.

We next prove the security when a set T of size at least k sends messages. We claim that the referee only learns $f(x_S)$ for every $S \subseteq T$ of size k. Since the randomness of each execution of the PSM protocol Π_{ah} is chosen independently, the referee can only learn information from the messages of Π_{ah} for each set S of size k. In an execution for a set $S \subseteq T$ it can only learn $f(x_S)$. In any execution of the PSM protocol for S such that $S \not\subseteq T$, the referee misses a message of at least one party thus, by Lemma 3.3, learns no information from this execution.

The randomness and communication of the ad hoc PSM protocol Π' are $\binom{n}{k}$ times larger than the randomness and communication, respectively, of the PSM protocol Π_{ah}. □

As every function f has a PSM realizing it [14,17], the previous theorem implies that every function has an ad hoc PSM protocol.

Corollary 3.5. *For every k-argument function f, there is an n-secure k-out-of-n* ad hoc PSM *protocol realizing f.*

3.3 A 2-out-of-n ad hoc PSM Protocol from a PSM Protocol for f

Suppose that we have a 2-party PSM protocol Π for a *symmetric* function f. We denote the parties in this PSM by Q_0 and Q_1. We want to construct an ad hoc PSM protocol Π^* for f using Π. The idea is to instruct the first party P_i to simulate Q_0 and instruct the second party P_j to simulate Q_1. The problem is that in an ad hoc PSM protocol a party does not know who the other party is; informally, it does not know if it is the "first party" or the "second party". Instead, we execute a few copies of the PSM protocol Π where, in some copies of the PSM, party P_i plays the role of Q_0, and in other copies it plays the role of Q_1. Specifically, we view each $i \in [n]$ in its $\log n$-bit binary representation $i = (i_1, \ldots, i_{\log n})$, and execute $\log n$ copies of Π, where in the ℓth copy P_i plays the role of Q_{i_ℓ}. Since for any $i \neq j$, there exists an index ℓ such that $i_\ell \neq j_\ell$, in the ℓth copy P_i, P_j simulate both Q_0 and Q_1 and the referee can compute f from this copy.

However, information can now leak when P_i and P_j simulate, in some copy, the same Q_b; that is, if $i_\ell = j_\ell$, for some ℓ. In particular, in such copy, P_i and P_j send the same message if $x_i = x_j$. To overcome this problem, in the ℓth copy, where party P_i plays the role of Q_{i_ℓ}, party P_i "encrypts" its message m using a key k_{i_ℓ} and each party playing the role of $Q_{\bar{i_\ell}}$ sends the key k_{i_ℓ} as part of its message. Thus, if both P_i, P_j play the role of the same party Q_b, then the referee does not obtain the key, and cannot learn any information from this copy of the PSM. The formal description of the ad hoc PSM protocol Π^* follows.

Randomness generation:

- Let p be a prime such that $\log p \geq \max\{\mathrm{Comm}(\Pi), \log n\}$, where $\mathrm{Comm}(\Pi)$ is the length of the messages in the PSM protocol Π. All arithmetic in the protocol is in $\mathbb{F}_p$.
- For $\ell = 1$ to $\log n$:
  - Independently generate randomness for the PSM protocol Π; denote by $r_{\ell,0}, r_{\ell,1}$ the generated randomness of Q_0, Q_1, respectively.
  - Choose four random values $a_{\ell,0}, b_{\ell,0}, a_{\ell,1}, b_{\ell,1} \in_R \mathbb{F}_p$.
- The randomness of P_i, where $i = (i_1, \ldots, i_{\log n})$, is

$$(r_{\ell,i_\ell}, a_{\ell,0}, b_{\ell,0}, a_{\ell,1}, b_{\ell,1})_{1 \leq \ell \leq \log n}.$$

Message generation for every $P_i \in S$:

- For every $\ell \in \{1, \ldots, \log n\}$, party P_i computes $m_{i,\ell}$ – the message that Q_{i_ℓ} sends in Π on input x_i with randomness r_{ℓ,i_ℓ}.
  P_i sends $(m_{i,\ell} + a_{\ell,i_\ell} \cdot i + b_{\ell,i_\ell})_{1 \leq \ell \leq \log n}$ and $(a_{\ell,\bar{i_\ell}}, b_{\ell,\bar{i_\ell}})_{1 \leq \ell \leq \log n}$.

Assume that P_i and P_j send messages and the referee wants to compute $f(x_i, x_j)$. It finds an index ℓ such that $i_\ell \neq j_\ell$. Without loss of generality, $i_\ell = 0$ and $j_\ell = 1$, that is, in the ℓth copy of Π, party P_i plays the role of Q_0 and P_j plays the role of Q_1. As P_i sends $m_{i,\ell} + a_{\ell,0} \cdot i + b_{\ell,0}$ and P_j sends $a_{\ell,0}, b_{\ell,0}$, the

referee can recover $m_{i,\ell}$ – the message of Q_0. Similarly, the referee can recover $m_{j,\ell}$ – the message of Q_1 – and, using the reconstruction procedure of Π, it can compute $f(x_i, x_j)$.

By the privacy property of protocol Π, the referee does not learn any additional information from an ℓth copy of the PSM, where $i_\ell \neq j_\ell$. Furthermore, this is true also for the concatenation of the messages in all the copies where $i_\ell \neq j_\ell$; note that since f is symmetric the output of the protocol in each such copy is the same. On the other hand, in any copy where $i_\ell = j_\ell$, the referee gets two "encrypted" messages $m_{i,\ell} + a_{\ell,i_\ell} \cdot i + b_{\ell,i_\ell}$ and $m_{j,\ell} + a_{\ell,i_\ell} \cdot j + b_{\ell,i_\ell}$. Since $i \neq j$ (and $a_{\ell,i_\ell}, b_{\ell,i_\ell}$ are random), then all pairs of "encrypted" messages are possible and the referee learns no information from this copy of Π. The security of Π^* follows.

Let $\mathrm{Rand}(\Pi)$ and $\mathrm{Comm}(\Pi)$ be the randomness complexity and communication complexity of Π, respectively. The randomness complexity of the new Π^* is

$$O\left(\log n \cdot \max\{\mathrm{Rand}(\Pi), \mathrm{Comm}(\Pi), \log n\}\right),$$

and the communication complexity of Π^* is $O(\log n \cdot \max\{\mathrm{Rand}(\Pi), \log n\})$.

3.4 A k-out-of-n ad hoc PSM Protocol from a PSM Protocol for f

We want to generalize the above ad hoc PSM protocol Π^* to larger values of k. Again, we will execute many copies of the original k-party PSM protocol Π. The properties we require are: (1) for every set $S \subseteq [n]$ of size k, there exists a copy in which the parties in S play roles of distinct parties in Π, and (2) in copies where the parties in S do not play roles of distinct parties in Π, no information is leaked. To achieve the first requirement, we use a perfect hash family.

Definition 3.6. *A perfect hash family $H = \{h : [n] \to [k]\}$ is a set of functions such that for any set $S \subseteq [n]$ of size k, there exists at least one $h \in H$ that is 1-1 over S.*

Example 3.7. For $k = 2$, the family of bit-functions $H = \{h_1, \ldots, h_{\log n}\}$, where $h_\ell(i) = i_\ell + 1$ (and i_ℓ is the ℓth bit in the binary representation of i) is a perfect hash family.

A perfect hash family with $\binom{n}{k}$ functions can be easily constructed, but much more efficient constructions, probabilistic or explicit, are possible. E.g., picking the h's at random, it is enough to have $|H| \approx e^k \cdot k \log n$ (for a specific size-k set S, a random function is 1-1 w/prob $k!/k^k > e^{-k}$, by Sterling formula, and we need to take care of about n^k such sets).

We next describe the ad hoc PSM protocol, assuming a k-party PSM protocol Π for a symmetric function f and a perfect hash family H.

Theorem 3.8. *Assume that there is a k-party PSM protocol Π for a* symmetric *function f with randomness complexity* $\mathrm{Rand}(\Pi)$ *and communication complexity* $\mathrm{Comm}(\Pi)$. *Then, there is a k-out-of-n* ad hoc PSM *protocol for f with*

randomness complexity $O(e^k \cdot k \log n \cdot \big(\mathrm{Rand}(\Pi) + k^2 \cdot \max\{\mathrm{Comm}(\Pi), \log n\}\big))$ *and communication complexity* $O(e^k \cdot k^3 \log n \max\{\mathrm{Comm}(\Pi), \log n\})$.

Proof. Denote the parties of Π by $Q_1, \ldots, Q_k$. We construct a k-out-of-n ad hoc PSM protocol Π^* as follows.

Randomness generation:

- Let p be a prime such that $\log p \geq \max\{\mathrm{Comm}(\Pi), \log n\}$, where $\mathrm{Comm}(\Pi)$ is the length of the messages in Π.
- For every $h \in H$ do:
  - Independently generate randomness for the hth copy of Π; let $r_{h,1}, \ldots, r_{h,k}$ be the generated randomness for $Q_1, \ldots, Q_k$ respectively.
  - Choose k random polynomials $A_{h,1}(Y), \ldots, A_{h,k}(Y)$ of degree $k-1$ over $\mathbb{F}_p$.
  - Consider each polynomial $A_{h,j}(Y)$ as an element in $\mathbb{F}_p^k$ and share it in a k-out-of-k additive sharing scheme; denotes its shares as $A_{h,j,1}, \ldots, A_{h,j,k}$.
- The randomness of P_i in the ad hoc PSM protocol Π^* is

$$(r_{h,h(i)}, A_{h,h(i)}, A_{h,1,h(i)}, \ldots, A_{h,k,h(i)})_{h \in H}.$$

Message generation for every $P_i \in S$:

- For every $h \in H$, party P_i computes $m_{i,h}$ – the messages that $Q_{h(i)}$ sends in the PSM protocol Π on input x_i with randomness $r_{h,h(i)}$. Party P_i sends $(A_{h,h(i)}(i) + m_{i,h})_{h \in H}$ and, in addition, the shares $(A_{h,j,h(i)})_{h \in H, j \in [k]}$.

Assume that a set S of size k sends messages and the referee wants to compute $f(x_S)$. The referee finds a function $h \in H$ that is 1-1 on S. Let $i \in S$. Party P_i plays the role of $Q_{h(i)}$ in the hth copy of Π, and sends the message $A_{h,h(i)}(i) + m_{i,h}$. Furthermore, all k parties in S send their shares in a k-out-of-k secret-sharing scheme with the secret $A_{h,h(i)}$. Thus, the referee can reconstruct $A_{h,h(i)}$, compute $A_{h,h(i)}(i)$, and recover $m_{i,h}$. Similarly, the referee can recover all k messages in the hth copy of Π and can decode $f(x_S)$.

By the privacy property of protocol Π, the referee does not learn any additional information from an hth copy of Π, for every h such that h is 1-1 on S. Furthermore, this is true also for the concatenation of the messages in all such copies and, since f is symmetric, the output of the protocol in each such copy is the same. On the other hand, in any copy where h is not 1-1 on S, the referee does not get any information on $A_{h,h(i)}$, since it gets at least two identical shares of this secret. The referee gets at most k messages "encrypted" by the same secret key $A_{h,h(i)}$. The values $\{A_{h,h(i)}(i)\}_{i \in S}$ are k points on a random polynomial of degree $k-1$, thus, they are uniformly distributed and serve as random pads, i.e., the referee gets no information from such hth copy of the PSM Π.

The randomness complexity of Π^* is

$$|H| \cdot \left(\mathrm{Rand}(\Pi) + k^2 \cdot \max\{\mathrm{Comm}(\Pi), \log n\}\right) \approx e^k \cdot k \log n \cdot \left(\mathrm{Rand}(\Pi) + k^2 \cdot \max\{\mathrm{Comm}(\Pi), \log n\}\right),$$

To analyze the communication complexity of Π^*, note that for each h, each party P_i sends its encrypted message and also a share for $A_{h,j}$, for all $j \in [k]$. All together, the communication complexity of each party is

$$|H| \cdot k^2 \cdot \max\{\mathrm{Comm}(\Pi), \log n\} \approx e^k k^3 \log n \cdot \max\{\mathrm{Comm}(\Pi), \log n\}.$$

□

Remark 3.9. There may be several functions in H, say h, h', that are 1-1 on S (and, moreover, h_S is different than h'_S). Since we assume here that the function f is symmetric, the output is the same in both copies of Π and, since the randomness is independent, there is no additional information. If f was not symmetric the referee may learn multiple outputs (under different orders) and hence additional information on the input.

4 An ad hoc PSM Protocol Based on a PSM Protocol for a Related Function

In this section we construct an ad hoc PSM protocol for f from a PSM protocol for a related function g. The construction is similar to the construction of the ad hoc PSM protocol for SUM described in Sect. 1.1. To construct the ad hoc PSM protocol for the k argument function $f : X^k \to Y$, we define a (partial) n-argument function $g : (X \cup \{\perp\})^n \to Y \cup \{\perp\}$, where if there are more than $n-k$ inputs that are $\perp$, the function outputs $\perp$, if there are exactly $n-k$ inputs that are $\perp$, the function outputs the output of f on the k non-$\perp$ inputs, and if there are less than $n-k$ inputs that are $\perp$, then the function is undefined (in the latter case, we do not care what g outputs).

Lemma 4.1. *If there exists a PSM protocol Π_g for g with randomness complexity* $\mathrm{Rand}(\Pi_g)$ *and communication complexity* $\mathrm{Comm}(\Pi_g)$*, then there exists an* ad hoc PSM *protocol for f with randomness complexity* $\mathrm{Rand}(\Pi_g) + n \cdot \max\{\mathrm{Comm}(\Pi_g), \log n\}$ *and communication complexity* $n \cdot \max\{\mathrm{Comm}(\Pi_g), \log n\}$.

Proof. We construct an ad hoc PSM protocol Π_f for f from the PSM protocol Π_g as follows.

Randomness generation:

- Generate randomness for the PSM protocol Π_g; let $r_1, \ldots, r_n$ be the generated randomness of $P_1, \ldots, P_n$, respectively.

– Let $m_{\perp,j}$ be the message that P_j sends in Π_g with randomness r_j and input $\perp$. Share $m_{\perp,j}$ using a k-out-of-n secret sharing scheme; let $m_{\perp,j,i}$ be the i-th share.
– The randomness of P_i in the ad hoc PSM protocol is $r_i, (m_{\perp,j,i})_{j\neq i}$.

Message generation:

– The message of P_i on input x_i is its message on input x_i and randomness r_i in the PSM protocol Π_g and, in addition, $(m_{\perp,j,i})_{j\neq i}$.

Assume that parties in a set S of size exactly k send messages. Then, the referee has the k messages in Π_g of the parties in S with inputs $x_i \neq \perp$ and, for each $j \notin S$, it has k shares of the message $m_{\perp,j}$. Thus, the referee can reconstruct $g(y_1, \ldots, y_n) = f((x_i)_{i\in S})$, where $y_i = x_i$ if $i \in S$ and $y_i = \perp$ otherwise. On the other hand, since each party $p_i \in S$ does not send its share of $m_{\perp,i}$, the referee gets $k-1$ shares of $m_{\perp,i}$; hence, the referee has no information on $m_{\perp,i}$. Thus, when k parties send messages, the referee in Π_f has the same information that the referee has in Π_g and the privacy requirement for Π_f protocol follows from the privacy requirement of the PSM Π_g.

Assume that parties in a set S of size less than k parties send messages. In this case, we claim that the referee in Π_f gets no information even if we give it more information, namely, $m_{\perp,j}$ for every $P_j \notin S$. In this case, the referee gets messages of inputs whose output is $\perp$. By the privacy of the PSM protocol, these messages are distributed as the messages when all the inputs are $\perp$, that is, the referee does not learn any information on the inputs.

The randomness in the above ad hoc PSM Π_f is $\mathrm{Rand}(\Pi_g) + n \cdot \mathrm{Comm}(\Pi_g)$. The communication in Π_f is $O(n \cdot \mathrm{Comm}(\Pi_g))$ (assuming $\mathrm{Comm}(\Pi_g)$ is at least $\log n$). □

Example 4.2. Assume that $f : \{0,1\}^k \to \{0, \ldots, k\}$ is a symmetric function (that is, the output of f only depends on the number of 1's in the input). The function f has a small branching program (i.e., the size of the branching program is $O(k^2)$), thus f itself has an efficient PSM protocol [17]. Furthermore, the function g has a branching program of size $O(nk^2)$, thus, it has an efficient PSM protocol, i.e., a PSM with communication complexity $O(n^2k^4)$. This implies an ad hoc PSM protocol for f with communication $O(n^3k^4)$.

If a function f has a small non-deterministic branching program, then the corresponding function g has a small non-deterministic branching program, thus, by [17], g has an efficient PSM protocol. By Lemma 4.1, we get for all $k \leq n$ efficient k-secure ad hoc PSM protocols for every function that has a small non-deterministic branching programs.

Similarly, if f has a small circuit, then g has a small circuit, thus, by using Yao's garbled circuit construction [14,20] we get a simulation-based-secure PSM for g assuming the existence of a one-way function. By Lemma 4.1, we get for all $k \leq n$ efficient computational k-secure ad hoc PSM protocols (with simulation-based-security) for every function that has a small circuit assuming the existence of a one-way function.

5 A Protocol for Equality

Define the equality function EQ : $(\{0,1\}^\ell)^k \to \{0,1\}$ as the function, whose input is k strings of length ℓ and whose output is 1 if and only if all strings are equal. We next present an ad hoc PSM protocol for EQ.

Lemma 5.1. *There is a statistically-secure* ad hoc PSM *protocol for* EQ *whose randomness complexity and communication complexity are* $O(n+\ell)$.

Proof. We next describe the ad hoc PSM protocol.

Randomness generation:

- Let p be a prime number such that $\log p > \max\{n, \ell\}$.
- Choose at random an element $a \in \mathbb{F}_p$ such that $a \neq 0$.
- Choose $k-1$ random elements $r_0, \ldots, r_{k-2}$ in $\mathbb{F}_p$ and define the polynomial $Q(Y) = \sum_{i=0}^{k-2} r_i Y^i$ (over $\mathbb{F}_p$).
- Choose n random elements $j_1, \ldots, j_n$ in $\mathbb{F}_p$
- The randomness of P_i in the ad hoc PSM protocol is $(j_i, Q(j_i), a)$.

Message generation for every $P_i \in S$:

- P_i sends $j_i, Q(j_i) + ax_i$.

Function reconstruction by the referee:

- Assume the referee gets k pairs $(\ell_1, z_1), \ldots, (\ell_k, z_k)$. If all point lie on a polynomial of degree $k-2$ answer "equal", otherwise answer "not equal".

First assume that all k inputs are equal, say to α. In this case the k pairs lie on the polynomial $Q(Y) + a\alpha$ and the referee answers "equal". Furthermore, since the free coefficient of $Q(Y) + a\alpha$ is $r_0 + a\alpha$, the values $(\ell_1, z_1), \ldots, (\ell_k, z_k)$ are independent of α.

We next consider the case that not all of the k inputs are equal. Since $j_1, \ldots, j_n$ are uniformly distributed, we can assume, without loss of generality, that $S = \{P_1, \ldots, P_k\}$. Fix any inputs $x_1, \ldots, x_k$ such that $x_k \neq x_\ell$ for some $1 \leq \ell < k$ (again, this is w.l.o.g.). We prove that with probability at least $1 - k/p$ over the choice of $j_1, \ldots, j_k$, the values $z_1, \ldots, z_k$ are uniformly distributed in $\mathbb{F}_p^k$. In particular, this implies that with probability at least $1 - k/p$, the referee answers "not equal". Furthermore, it implies the privacy for this case.

Fix any $j_1, \ldots, j_{k-1}$ and $z_1, \ldots, z_{k-1}$. Let $H(Y)$ and $M(Y)$ be the polynomials of degree $k-2$ such that $H(j_i) = x_i$ and $M(j_i) = z_i$ for every $1 \leq i \leq k-1$. Such polynomials exist and they are unique. Notice that for every $a \neq 0$ there exists a unique polynomial $Q(Y)$ of degree $k-2$ that can be chosen in the randomness generation of the protocol, where $Q(Y) = M(Y) - a \cdot H(Y)$ (since both the r.h.s. and the l.h.s. are polynomials of degree $k-2$ that agree on the $k-1$ points $j_1, \ldots, j_{k-1}$). Thus, the message of P_k is

$$z_k = Q(j_k) + ax_k = M(j_k) - a \cdot H(j_k) + ax_k.$$

The protocol fails (i.e., outputs "equal" although not all inputs are equal) if and only if $z_k = M(j_k)$; the last equality is true if and only if $H(j_k) = x_k$. Notice that since $H(Y) \neq x_k$ since $H(j_\ell) = x_\ell \neq x_k$. Since $H(Y) \neq x_k$ is a polynomial of degree $k-2$, there are at most $k-2$ values of j_k such that $H(j_k) = x_k$. Thus, with probability at least $1-(k-2)/p \geq 1-(k-2)/2^n$, the referee in this case outputs "not equal". Assuming that such j_k is not chosen, $z_k = M(j_k)+a(-H(j_k)+x_k)$; as a is chosen at random, the value z_k is random (provided that the kth pair does not lie on the polynomial). □

6 Order Revealing Encryption from an Ad Hoc PSM Protocol

An order revealing encryption is a private-key encryption that enables computing the order between two messages (that is, checking if $m_1 < m_2$, $m_1 = m_2$, or $m_1 > m_2$), given their encryptions (without knowing the private key), but does not disclose any additional information. In this section, we show how to use ad hoc PSM protocols to construct information-theoretically secure order revealing encryption that is 2-bounded (namely, the encryption is secure as long as only two messages are encrypted).

Definition 6.1. *The greater than function,* $\mathrm{GTE}_\ell : \{0,1\}^\ell \times \{0,1\}^\ell \to \{-1, 0, 1\}$*, is defined as follows:*

$$\mathrm{GTE}_\ell(x,y) = \begin{cases} -1 & If x < y \\ 0 & If x = y \\ 1 & If x > y, \end{cases}$$

where we identify the strings in $\{0,1\}^\ell$ *with the integers in* $\{0,\ldots,2^\ell - 1\}$.

Definition 6.2 (Order Revealing Encryption (ORE): Syntax and correctness). *Let* $\epsilon : \mathbb{N} \to [0, 0.5)$*. An* $\epsilon(\lambda)$*-ORE for messages in* $\{0,1\}^\ell$ *is composed of 4 efficient algorithms:*

- $\textsc{Gen}_{ORE}$ *is a randomized key generation algorithm, that on input* 1^λ *(where* λ *is a security parameter), outputs a key* k*;*
- $\textsc{Enc}_{ORE}$ *is an encryption algorithm, that on input message* m *and a key* k*, outputs an encryption* c*;*
- $\textsc{Dec}_{ORE}$ *is a decryption algorithm, that on input an encryption* c *and a key* k*, outputs a message* m *satisfying the following correctness requirement for any* $m \in \{0,1\}^\ell$*:*

$$\Pr\left[k \leftarrow \textsc{Gen}_{ORE}(1^\lambda) : \textsc{Dec}_{ORE}\left(\textsc{Enc}_{ORE}(m,k),k\right) = m\right] \geq 1 - \epsilon(\lambda).$$

- $\textsc{Comp}_{ORE}$ *is a comparison algorithm, that given any two encryptions* c_1, c_2*, outputs a value in* $\{-1,0,1\}$ *such that for any* $m_1, m_2 \in \{0,1\}^\ell$*:*

$$\Pr\left[\begin{array}{l} k \leftarrow \textsc{Gen}_{ORE}(1^\lambda), c_1 \leftarrow \textsc{Enc}_{ORE}(m_1,k), \\ c_2 \leftarrow \textsc{Enc}_{ORE}(m_2,k) : \textsc{Comp}_{ORE}\left(c_1,c_2\right) = \mathrm{GTE}_\ell(m_1,m_2) \end{array}\right] \geq 1 - \epsilon(\lambda).$$

If the comparison algorithm is the comparison over the integers (e.g., it returns -1 *whenever* $c_1 < c_2$*), then the encryption is called Order* Preserving *Encryption (OPE).*

Remark 6.3. Given the private key k and an encryption c, one can use a binary search using COMP_{ORE} to decrypt c. That is, we do not need to specify the decryption algorithm. For efficiency, one can avoid this binary search by encrypting the message using a standard (semantically secure) encryption scheme in addition to the ORE encryption.

We next define the security requirement of ORE. Our definition is the information theoretic analogue of the IND-OCPA security requirement from [8]. The definition of IND-OCPA is similar to the traditional IND-CPA definition of private key encryption, however, as the adversary can learn the order between two messages from their encryptions, the IND-OCPA definition prevents the adversary from using this information by limiting the encryption queries that it can make (see (1) in Definition 6.4 below).

Definition 6.4 (ORE: Security). *Consider the following game between an all-powerful adversary and a challenger:*

- *The input of both parties is a security parameter* 1^λ *and a bound on the number of queries* 1^t*.*
- *The challenger chooses a random bit* b *with uniform distribution and generates a key* $k \leftarrow \text{GEN}_{ORE}(1^\lambda)$*.*
- *For* $i = 1$ *to* t *do:*
  - *The adversary chooses two message* $m_0^i, m_1^i \in \{0,1\}^\ell$ *and sends them to the challenger.*
  - *The challenger computes* $c_i \leftarrow \text{ENC}_{ORE}(m_b^i, k)$ *and sends* c_i *to the adversary.*
- *The adversary returns a bit* b'*.*

We say that the adversary wins if $b = b'$ *and for every* $1 \le i < j \le t$

$$\text{GTE}_\ell(m_0^i, m_0^j) = \text{GTE}_\ell(m_1^i, m_1^j). \tag{1}$$

Let $\epsilon : \mathbb{N} \to [0, 0.5)$*. We say that an ORE is* $\epsilon(\lambda)$*-secure if for every polynomial* $t(\lambda)$ *and every adversary* $\mathcal{A}$ *the probability that* $\mathcal{A}$ *with parameters* $1^\lambda, 1^{t(\lambda)}$ *wins is at most* $1/2 + \epsilon(\lambda)$*. We say that an ORE is* t*-bounded* $\epsilon(\lambda)$*-secure if for every adversary* $\mathcal{A}$ *the probability that* $\mathcal{A}$ *with parameters* $1^\lambda, 1^t$ *wins is at most* $1/2 + \epsilon(\lambda)$*.*

We next describe some relevant results for OPE and ORE. In this discussion all encryption schemes are computationally secure. Order preserving encryption was introduced by Agrawal et al. [1]; their motivation was encrypting a database while allowing to answer range queries given the encrypted data (without the secret key). A cryptographic treatment of OPE was given by Boldyreva et al. [8, 9]; they gave a formal definition of OPE (called IND-OCPA) and showed that,

in any OPE satisfying this definition, the length of the encryption is $2^{\omega(\ell)}$, where ℓ is the length of the messages (this is true even if the attacker can only ask to encrypt 3 messages). In a follow up work, Boldyreva et al. [10,11] defined ORE. As ORE is a special case of multi-input functional encryption (MIFE) [15], it is implied by indistinguishability obfuscation (iO). Boneh et al. [12] constructed ORE directly from multi-linear maps (with bounded multi-linearity). t-bounded ORE can be constructed based on the LWE assumption or from pseudorandom generators computable by small-depth circuits [13].

We next show how to construct ORE from an ad hoc PSM protocol for the greater than function GTE_ℓ.

Theorem 6.5. *There exists a 2-bounded $1/2^\lambda$-secure ORE with messages in $\{0,1\}^\ell$ and encryptions of length $O(\ell^2\lambda + \lambda^2)$.*

Proof. We start with a 2-out-of-n ad hoc PSM protocol Π_{GTE} for GTE: The function GTE_ℓ has a deterministic branching program of size $O(\ell)$ thus, by [17], it has a PSM protocol with randomness and communication complexity $O(\ell^2)$. By Theorem 3.8, GTE_ℓ has an ad hoc PSM protocol with complexity $O(\ell^2 \log n + \log^2 n)$. Note that Theorem 3.8 requires that the function for which we construct an ad hoc PSM protocol is symmetric. As $\mathrm{GTE}_\ell(m_2, m_1) = -\mathrm{GTE}_\ell(m_1, m_2)$, the transformation described in Theorem 3.8 from a PSM protocol to an ad hoc PSM protocol is valid for GTE_ℓ.

We next describe a construction of ORE, that is, we desribe algorithms $(\textsc{Gen}_{\mathrm{ORE}}, \textsc{Enc}_{\mathrm{ORE}}, \textsc{Comp}_{\mathrm{ORE}})$ (by Theorem 6.3 we do not need to describe $\textsc{Dec}_{\mathrm{ORE}}$). We use the ad hoc PSM Π_{GTE} with $n = 2^\lambda$ parties (where λ is the security parameter).

- Algorithm $\textsc{Gen}_{\mathrm{ORE}}$ generates a key k by choosing a random string for $\textsc{Gen}_{\mathrm{GTE}}$, this key has length $O(\ell^2 \log n + \log^2 n)$. We emphasize that during the key generation we do not apply $\textsc{Gen}_{\mathrm{GTE}}$ as its output is too long (it contains n stings).
- Algorithm $\textsc{Enc}_{\mathrm{ORE}}$ encrypts a message x by choosing a random party P_i (where $1 \le i \le n$) and using $\textsc{Gen}_{\mathrm{GTE}}(k)$ to generate the random string r_i of P_i in Π_{GTE}.[2] The encryption of x is i and $c \leftarrow \textsc{Enc}_{\mathrm{GTE},i}(x, r_i)$ – the message of P_i on input x and randomness r_i.
- Algorithm $\textsc{Comp}_{\mathrm{ORE}}((i_1, c_1), (i_2, c_2))$ returns $\textsc{Dec}_{\mathrm{GTE}}(\{i_1, i_2\}, c_1, c_2)$ if $i_1 \neq i_2$ and "FAIL" otherwise.

If two messages are encrypted using different parties (i.e., $i_1 \neq i_2$), then the correctness of the comparison and the security of Π_{GTE} guarantees that, given the two encryptions, exactly their order is revealed (i.e., the first message is smaller, equal, or greater than the second message). If the two messages are encrypted using the same party (i.e., $i_1 = i_2$), then correctness and security are not guaranteed. However, the probability of this event is $1/n = 1/2^\lambda$, which is negligible. □

[2] The time required to generate r_i is $O(\ell^2 \log n + \log^2 n)$.

Remark 6.6. In the proof of Theorem 6.5, we can replace the ad hoc PSM protocol for GTE_ℓ obtained via the PSM protocol from [17] by any ad hoc PSM protocol for GTE_ℓ as long as its complexity is $\eta(n,\lambda)\log^c n$ for some function η and constant c. In particular, if we use a $(2,t)$-secure ad hoc PSM protocol for GTE_ℓ, then the resulting ORE would be t-bounded secure.

The ORE of Theorem 6.5 is secure only when 2 messages are encrypted. If 3 messages are encrypted, then the adversary gets 3 messages of the ad hoc PSM protocol for GTE_ℓ and the security of the ad hoc PSM protocol is broken. We can construct a t-bounded $1/\lambda$-secure ORE as sketched below:

- The key generation algorithms generates keys for $\alpha = \mathrm{poly}(\lambda, t)$ copies of the ORE of Theorem 6.5.
- The encryption algorithms encrypts m using a random subset of the keys of size $\lambda\sqrt{\alpha}$.
- Given encryptions of two messages, if there is a key that was used to encrypt both messages, then use the comparison algorithm of that copy to compare the two messages. The probability that no such key exists is $2^{-O(\lambda)}$.

The security of the above ORE is guaranteed as long as no 3 messages are encrypted with the same key. The probability that there are 3 messages that are encrypted under the same key can be reduced to $1/\lambda$ if α is big enough.

7 NIMPC Vs. (k,t)-Secure Ad Hoc PSM

In this section we consider two notions of PSM protocols, (k,t)-secure ad hoc PSM protocols and Non-Interactive secure MPC (NIMPC) protocols. Recall that an ad hoc PSM is (k,t)-secure if the referee getting at most t messages does not learn any information beyond the value of f on any subset of size k of the inputs. A t-robust NIMPC for a function f is a PSM protocol, where a referee colluding with t parties can only compute the values of the function when the inputs of the non-colluding parties is fixed (see [4] for a formal definition of NIMPC protocols). We show that the existence of NIMPC protocols is equivalent to the existence of (k,t)-secure ad hoc PSM protocols.

In the information-theoretic setting, these results should be interpreted as negative results, maybe implying that efficient protocols do not exists in both models. In the computaional setting, this results imply an efficient construction of computational ad hoc PSM protocols.

7.1 Ad hoc PSM ⇒ NIMPC

Given an n-out-of-$2n$ ad hoc PSM protocol for a boolean function f, we construct an n-party robust NIMPC protocol for f with the same complexity.

Lemma 7.1. *If there exists an (n,t)-secure n-out-of-$2n$* ad hoc PSM *protocol for a boolean function $f:\{0,1\}^n \to \{0,1\}$, then there exists an n-party $(t-n)$-robust* NIMPC *protocol for f with the same communication complexity.*

Proof. Let Π^* be the guaranteed ad hoc PSM protocol. Consider the following NIMPC protocol Π.

Randomness generation:

- Let $r_1, \ldots, r_{2n} \leftarrow \text{GEN}_{\Pi^*}()$.
- Choose at random n random bits $b_1, \ldots, b_n$.
- For $i \in [n]$ let
  - $M_{i,0} \leftarrow (2i - b_i, \text{ENC}_{\Pi^*,2i-b_i}(r_{2i-b_i}, 0))$.
  - $M_{i,1} \leftarrow (2i - 1 + b_i, \text{ENC}_{\Pi^*,2i-1+b_i}(r_{2i-1+b_i}, 1))$.
- The randomness of P_i is $M_{i,0}, M_{i,1}$.

Message generation for every $P_i \in S$:

- P_i on input $x_i \in \{0,1\}$ sends M_{i,x_i}.

Function reconstruction by the referee:

- The referee gets n messages, where for each i it gets from P_i either the messages of P_{2i} or P_{2i-1}. It uses the decryption of Π^* to compute f.

We next argue that Π is robust. Let A be a set of parties in Π of size $\tau \leq t-n$. The randomness of A and the messages of all other parties in Π are messages of distinct $n+\tau \leq t$ parties in Π^*. By the (n,t)-security of Π^*, from these messages the referee in Π^* can only compute the output of f on any subset of size n of these parties in Π^*, i.e., the inputs of the parties in Π that are not in A are fixed. Thus, in Π, the referee and the set A can only compute the residential function. Thus, the (n,t)-security of Π^* implies the $(t-n)$-robustness of Π. Notice that the referee knows the identity of the party in Π^* for which the messages was generated; however, by choosing random b_i's, it does not know if this message is for an input 0 or 1. □

7.2 NIMPC ⇒ Ad Hoc PSM

Our goal is to construct a (k,t)-secure ad hoc PSM protocol for a boolean function f from an NIMPC protocol Π computing a related function. We would like to use ideas similar to the construction in Sect. 4. Recall that, given a k-argument function $f : X^k \to Y$, we defined an n-argument function $g : (X \cup \{\perp\})^n \to Y \cup \{\perp\}$, where if there are exactly $n-k$ inputs that are $\perp$ then the output of g is the output of f on the k non-$\perp$ inputs, and it is $\perp$ otherwise.[3] We constructed a k-secure ad hoc PSM protocol for f by first generating the randomness using the PSM for g, and sharing the messages of each party with input $\perp$. We would like to start from an NIMPC protocol for g and get a (k,t)-secure ad hoc PSM protocol for f. There is a problem with this solution – in the resulting ad hoc PSM protocol the referee will get for each

[3] In Sect. 4, if there were less than $n-k$ inputs that are $\perp$, then the function was undefined; here we need to define the output as $\perp$.

active party messages for some input x_i and for the input $\perp$. The definition of the robustness of NIMPC protocols guarantees that if it gets one message from a party, then the referee can only evaluate the function on points where the input of this party is fixed to some (unknown) value. The definition does not guarantee that if a referee gets two messages from one party then it can only evaluate the output on points where the input of this party is fixed to one of these two (unknown) values.

To overcome this problem we define a new function $g'' : \{0,1\}^{3n} \to Y$ with $3n$ variables $x_{1,0}, x_{1,1}, x_{2,0}, x_{2,1}, \ldots, x_{n,0}, x_{n,1}, y_1, \ldots, y_n$, where given an assignment $a_{1,0}, a_{1,1}, a_{2,0}, a_{2,1}, \ldots, a_{n,0}, a_{n,1}, c_1, \ldots, c_n$ of g'', we define an assignment $a_1, a_2, \ldots, a_n$ of g as follows:

$$a_i = \begin{cases} \perp & \text{if } a_{i,0} = a_{i,1}, \\ c_i & \text{if } a_{i,0} = 1, a_{i,1} = 0, \\ 1 - c_i & \text{if } a_{i,0} = 0, a_{i,1} = 1 \end{cases}$$

and $g''(a_{1,0}, a_{1,1}, a_{2,0}, a_{2,1}, \ldots, a_{n,0}, a_{n,1}, c_1, \ldots, c_n) = g(a_1, a_2, \ldots, a_n)$.

Theorem 7.2. *If there is a $3n$-party $2n$-robust NIMPC protocol $\Pi_{g''}$ for g'' with randomness complexity* $\mathrm{Rand}(\Pi_{g''})$ *and communication complexity* $\mathrm{Comm}(\Pi_{g''})$ *then there exists a (k,n)-secure* ad hoc PSM *protocol for f with randomness complexity* $O(\mathrm{Rand}(\Pi_{g''}) + n \cdot \mathrm{Comm}(\Pi_{g''}))$ *and communication complexity* $O(n \cdot \mathrm{Comm}(\Pi_{g''}))$.

Proof. Denote the parties of the NIMPC $\Pi_{g''}$ by $P_{1,0}, P_{1,1}, \ldots, P_{n,0}, P_{n,1}, Q_1, \ldots, Q_n$. We next describe an ad hoc PSM protocol Π_f for f.

Randomness generation:

- Generate the randomness of $\Pi_{g''}$ for g''; let $r_{1,0}, r_{1,1}, \ldots, r_{n,0}, r_{n,1}, q_1, \ldots, q_n$ be the generated randomness of $P_{1,0}, P_{1,1}, \ldots, P_{n,0}, P_{n,1}, Q_1, \ldots, Q_n$ respectively.
- For every $1 \le j \le n$:
  - Choose $c_i \in \{0,1\}$ at random and let m_i be the message that Q_i sends in $\Pi_{g''}$ with randomness q_i and input c_i.
  - For every $b \in \{0,1\}$, let $m_{j,b}$ be the message that $P_{j,b}$ sends in $\Pi_{g''}$ with randomness $r_{j,b}$ and input 0.
- The randomness of P_i in the ad hoc PSM protocol is

$$r_{i,0}, r_{i,1}, c_i, (m_i)_{1 \le i \le n}, (m_{j,b})_{1 \le i \le n, b \in \{0,1\}}.$$

Message generation for every $P_i \in S$:

- Let u_i be the message that $P_{i,c_i \oplus x_i}$ sends in $\Pi_{g''}$ with input 1 and randomness $r_{i,c_i \oplus x_i}$.
- P_i sends $(c_i \oplus x_i), u_i$ and, in addition, $(m_i)_{1 \le i \le n}, (m_{j,b})_{j \ne i, b \in \{0,1\}}$.

Assume that a set $S \in \binom{[n]}{k}$ sends messages. To compute the value of f on the inputs of S, the referee applies the decoding procedure of $\Pi_{g''}$, where for every $i \in [n]$,

– If $i \in S$, then the message of $P_{i,c_i \oplus x_i}$ is u_i (i.e., an encoding of 1); otherwise it is $m_{i,c_i \oplus x_i}$ (i.e., an encoding of 0),
– the message of $P_{i,1-(c_i \oplus x_i)}$ is $m_{i,1-(c_i \oplus x_i)}$ (i.e., an encoding of 0), and
– the message of Q_i is m_i (i.e., an encoding of c_i).

The correctness follows as these messages correspond to the input

$$(z_{i,b})_{i \in [n], b \in \{0,1\}}, (c_i)_{i \in [n]}$$

where:

– If $i \notin S$, then $z_{i,0} = z_{i,1} = 0$, that is, it correspond to the input $a_i = \perp$ of g.
– If $i \in S$, then $z_{i,c_i \oplus x_i} = 1$ and $z_{1-(i,c_i \oplus x_i)} = 0$,
  • If $x_i = c_i$, then $z_{i,0} = 1$ and $z_{i,0} = 0$, that is, it correspond to the input $a_i = c_i = x_i$ of g,
  • If $x_i \neq c_i$, then $z_{i,0} = 0$ and $z_{i,0} = 1$, that is, it correspond to the input $a_i = 1 - c_i = x_i$ of g.

  That is, if $i \in S$, then it correspond to the input $a_i = x_i$ of g.

To conclude, the referee reconstructs

$$g''((z_{i,b})_{i \in [n], b \in \{0,1\}}, (c_i)_{i \in [n]}) = g((x_i)_{i \in S}, (\perp)_{i \notin S}) = f((x_i)_{i \in S}).$$

For the (k,t)-security, note that if a set T of size t' sends messages in the ad hoc PSM protocol for f, then the referee gets two messages for $P_{i,c_i \oplus x_i}$ for every $i \in S$ and one message for every other party. Thus, by the robustness of the NIMPC protocol $\Pi_{g''}$, the referee can only compute outputs of g, where the input of every $i \notin S$ is fixed to $\perp$ and the input of every $i \in S$ is either x_i or $\perp$. Since g is defined to be $\perp$ if the number of non-bottom inputs is not k, the referee can only compute the values of f on subsets of size k of T. □

The transformation of Theorem 7.2 also applies if the NIMPC protocol is computationally-secure. Specifically, in [4] it is shown that if iO and one-way functions exist, then there is a computational indistinguishably-secure NIMPC protocol for every function. This implies that if iO and one-way functions exist then there is a computational (k,n)-indistinguishably-secure ad hoc PSM protocol for every function f.

8 Ad Hoc Protocols for and Threshold Imply Nontrivial Obfuscation

Computational ad hoc PSM protocols for general functions imply obfuscation. This follows from Lemma 7.1, showing that ad hoc PSM protocols imply NIMPC protocols, and by results of [4], showing that NIMPC protocols imply obfuscation. To prove this result, ad hoc PSM protocols for fairly complex functions, i.e., universal functions, are used. In this section, we show that ad hoc PSM protocols for *simple* functions already imply obfuscation for interesting functions.

Specifically, computational ad hoc PSM protocols for AND with VBB security imply point function obfuscation and ad hoc PSM protocols for threshold functions with VBB security imply fuzzy point function obfuscation [7]. There are several definitions of point function obfuscation in the literature (see [6]). In this paper, we consider the strong virtual black-box notion of obfuscation of Barak et al. [3] for point function and fuzzy point function obfuscation. This notion was considered for point function obfuscation in, e.g., [19]. As the only known constructions for fuzzy point function obfuscation are based on strong assumptions (e.g., iO), these results imply that even ad hoc PSM protocols with VBB security for the threshold function may require strong assumptions.

Notation 8.1. *For every $x \in \{0,1\}^n$, define the point function $I_x : \{0,1\}^n \to \{0,1\}$ where $I_x(y) = 1$ if $x = y$ and $I_x(y) = 0$ otherwise. For every $x \in \{0,1\}^n$ and $0 < \delta < 1$, define the fuzzy point function $F_x^\delta : \{0,1\}^n \to \{0,1\}$ where $F_x^\delta(y) = 1$ if $\mathrm{dist}(x,y) \leq \delta n$ and $F_x^\delta(y) = 0$ otherwise, where $\mathrm{dist}(x,y)$ is the Hamming distance. We will also denote by I_x and F_x the canonical circuits that compute these functions.*

Lemma 8.2. *If there exists an $(n, 2n)$-VBB-secure* ad hoc PSM *protocol for* AND*, then there is a point function obfuscation, i.e., an obfuscation for $\{I_x\}_{x\in\{0,1\}^n}$.*

Proof. The obfuscation algorithm of a point function I_x uses the computational ad hoc PSM protocol $\Pi_{\mathrm{AND}} = (\textsc{Gen}_{\mathrm{AND}}, \textsc{Enc}_{\mathrm{AND}}, \textsc{Dec}_{\mathrm{AND}})$ for AND. We denote the $2n$ parties in Π_{AND} by $\{P_{i,b}\}_{i\in[n],b\in\{0,1\}}$. Algorithm $\textsc{Obf}(1^n, x)$ is as follows:

- Let $(r_{i,b})_{i\in[n],b\in\{0,1\}} \leftarrow \textsc{Gen}_{\mathrm{AND}}(1^n)$.
- For every $i \in [n]$ let $z_{i,x_i} \leftarrow 1$ and $z_{i,\overline{x_i}} \leftarrow 0$.
- For every $i \in [n]$ and $b \in \{0,1\}$ let $m_{i,b} \leftarrow \textsc{Enc}_{\mathrm{AND}}(z_{i,b}, r_{i,b})$.
- Return a circuit C that on input $y \in \{0,1\}^n$ computes

$$\textsc{Dec}_{\mathrm{AND}}(\{(i,y_i)\}_{i\in[n]}, (m_{i,y_i})_{i\in[n]}).$$

Correctness: The circuit C returns the output of the decoding algorithm $\textsc{Dec}$ on the messages $(m_{i,y_i})_{i\in[n]}$, which encode the inputs $(z_{i,y_i})_{i\in[n]}$. Hence, C returns $\mathrm{AND}((z_{i,y_i})_{i\in[n]})$. If $y = x$, then for every $i \in [n]$, $y_i = x_i$ and $z_{i,y_i} = 1$, thus C returns 1. If $y \neq x$, then $y_i = \overline{x_i}$ for at least one $i \in [n]$, thus $z_{i,y_i} = 0$ and C returns 0.

Security: Let $\mathcal{A}$ be an adversary attacking the obfuscation in the real world, that is, the adversary gets the above circuit C. We construct a simulator $\textsc{Sim}$, with an oracle access to I_x, such that there exists a negligible function negl() for which, for every $x \in \{0,1\}^n$,

$$|\Pr[\mathcal{A}(1^n, C) = 1] - \Pr[\textsc{Sim}^{I_x}(1^n) = 1]| \leq \mathrm{negl}(n), \tag{2}$$

where the first probability is taken over the randomness of $\mathcal{A}$ and the randomness of $\textsc{Obf}(1^n, x)$ and the second probability is taken over the randomness of $\textsc{Sim}$.

We first define an attacker $\mathcal{A}_{\text{AND}}$ against the ad hoc PSM protocol Π_{AND}: $\mathcal{A}_{\text{AND}}$ gets as an input $2n$ messages and generates a circuit C from these messages as Obf does, and executes $\mathcal{A}$ on C. By the VBB-security of Π_{AND}, there exists a simulator Sim_{AND} for the adversary $\mathcal{A}_{\text{AND}}$; this simulator Sim_{AND} should have an oracle access to the function AND on any n of the $2n$ inputs $(z_{i,b})_{i\in[n],b\in\{0,1\}}$.

The simulator Sim for the obfuscation, with oracle access to I_x, emulates Sim_{AND}, where the queries to AND are answered as follows: if a query contains two variables $z_{i,0}$ and $z_{i,1}$, for some $i \in [n]$, then the answer is 0 (as the value of one of them is zero). Otherwise, for every i there is exactly one y_i such that z_{i,y_i} is in the query; in this case $z_{i,y_i} = 1$ if and only if $y_i = x_i$, i.e., $\text{AND}((z_{i,y_i})_{i\in[n],b\in\{0,1\}}) = 1$ iff $x = (y_1, \ldots, y_n)$ iff $I_x((y_1, \ldots, y_n)) = 1$. In this case, Sim answers the query by invoking its oracle I_x. The VBB-security of Π_{AND} implies that (2) holds. □

For $\delta < 0.5$, let $\text{Th}_\delta : \{0,1\}^n \to \{0,1\}$ be the following function:

$$\text{Th}_\delta(x_1, \ldots, x_n) = 1 \text{ iff } \sum_{i=1}^{n} x_i \geq (1-\delta)n.$$

We next construct fuzzy point function obfuscation from an ad hoc PSM protocol for Th_δ with VBB security. The construction and its proof of correctness are similar to those in Lemma 8.2; however, the proof of security is more involved. For this proof, we need the following claim.

Claim 8.3. Let $\delta < 0.5$. There is an efficient algorithm that, given a point w such that $F_x^\delta(w) = 1$ and an oracle access to F_x^δ, can find x.

Proof. Let $w = (w_1, \ldots, w_n)$ and $\overline{w} = (\overline{w_1}, \ldots, \overline{w_n})$. Since $\text{dist}(x, w) \leq \delta n < 0.5n$, it must be that $\text{dist}(x, \overline{w}) > 0.5n > \delta n$, i.e., $F_x^\delta(\overline{w}) = 0$. There must be a j such that $F_x^\delta(w_1, \ldots, w_j, \overline{w_{j+1}}, \ldots, \overline{w_n}) = 1$ and $F_x^\delta(w_1, \ldots, w_{j-1}, \overline{w_j}, \ldots, \overline{w_n}) = 0$. Furthermore, such j can be found by $n-1$ queries to the oracle F_x^δ. Let $v = (w_1, \ldots, w_j, \overline{w_{j+1}}, \ldots, \overline{w_n})$; it must be that $\text{dist}(x, v) = \lfloor \delta n \rfloor$. If $v_i = x_i$, for some i, and we flip the ith bit in v (i.e., consider $v \oplus e_i$), then the distance between the resulting sting and x will be larger than δn. On the other hand, if $v_i \neq x_i$, then $\text{dist}(x, v \oplus e_i) < \text{dist}(x, v) \leq \delta n$. Thus, the following procedure recovers x:

– For $i = 1$ to n: if $F_x^\delta(x, v \oplus e_i) = 0$ then $x_i = v_i$, otherwise $x_i = \overline{v_i}$.

□

Lemma 8.4. *Let $\delta < 0.5$. If there is an $(n, 2n)$-VBB-secure* ad hoc PSM *protocol for Th_δ, then there is a fuzzy point function obfuscation, i.e., an obfuscation for $\{F_x^\delta\}_{x\in\{0,1\}^\ell}$.*

Proof. The obfuscation algorithm $\text{Obf}_{\text{fuzzy}}$ of a fuzzy point function F_x^δ uses the computational n-out-of-$2n$ ad hoc PSM protocol $\Pi_{\text{Th}} = (\text{Gen}_{\text{Th}}, \text{Enc}_{\text{Th}}, \text{Dec}_{\text{Th}})$ for Th_δ. We denote the parties in Π_{Th} by $\{P_{i,b}\}_{i\in[n],b\in\{0,1\}}$. Algorithm $\text{Obf}(1^n, x)$ is as follows:

– Let $(r_{i,b})_{i\in[n],b\in\{0,1\}} \leftarrow \text{GEN}_{\text{Th}}(1^n)$.
– For every $i \in [n]$ let $z_{i,x_i} \leftarrow 1$ and $z_{i,\overline{x_i}} \leftarrow 0$.
– For every $i \in [n]$ and $b \in \{0,1\}$ let $m_{i,b} \leftarrow \text{ENC}_{\text{Th}}(z_{i,b}, r_{i,b})$.
– Return a circuit C that on input $y \in \{0,1\}^n$ computes

$$\text{DEC}_{\text{Th}}(\{(i,y_i)\}_{i\in[n]}, (m_{i,y_i})_{i\in[n]}).$$

Correctness: The circuit C returns the output of the decoding algorithm DEC on the messages $((m_{i,y_i})_{i\in[n]})$, which encode the inputs $(z_{i,y_i})_{i\in[n]}$. Hence, C returns $\text{Th}_\delta((z_{i,y_i})_{i\in[n]})$. If $\text{dist}(x,y) \le \delta n$, then $y_i = x_i$ for at least $(1-\delta)n$ values of i, and $z_{i,y_i} = 1$ for at least $(1-\delta)n$ values of i, thus, C returns 1. If $\text{dist}(x,y) > \delta n$, then $y_i = \overline{x_i}$ for more than $(1-\delta)n$ values of i, thus, $z_{i,y_i} = 1$ for less than $(1-\delta)n$ values of i and C returns 0.

Security: Let $\mathcal{A}$ be an adversary attacking the obfuscation in the real world. We construct a simulator $\text{SIM}_{\text{fuzzy}}$, with an oracle access to F_x^δ, such that there exists a negligible function negl() for which for every $x \in \{0,1\}^n$

$$|\Pr[\mathcal{A}(1^n, C) = 1] - \Pr[\text{SIM}_{\text{fuzzy}}^{F_x^\delta}(1^n) = 1]| \le \text{negl}(n), \tag{3}$$

where the first probability is taken over the randomness of $\mathcal{A}$ and the randomness of $\text{OBF}_{\text{fuzzy}}(1^n, x)$ and the second probability is taken over the randomness of $\text{SIM}_{\text{fuzzy}}$.

We first define an attacker $\mathcal{A}_{\text{Th}}$ against the ad hoc PSM protocol Π_{Th}: $\mathcal{A}_{\text{Th}}$ gets as an input $2n$ messages and generates a circuit C from these messages as $\text{OBF}_{\text{fuzzy}}$ does, and executes $\mathcal{A}$ on C. By the VBB-security of Π_{Th}, there exists a simulator SIM_{Th} for the adversary $\mathcal{A}_{\text{Th}}$; this simulator SIM_{Th} should have an oracle access to the function Th_δ of any n of the inputs $(z_{i,b})_{i\in[n],b\in\{0,1\}}$.

The simulator $\text{SIM}_{\text{fuzzy}}$ for the obfuscation, with oracle access to F_x^δ, emulates SIM_{Th}, where the queries to Th_δ are answered as follows: If for every i there is exactly one y_i such that z_{i,y_i} is in the query, then $z_{i,y_i} = 1$ if and only if $y_i = x_i$, i.e., $\text{Th}_\delta((z_{i,y_i})_{i\in[n],b\in\{0,1\}}) = 1$ iff $\text{dist}(x,(y_1,\ldots,y_n)) \le \delta n$ iff $F_x^\delta((y_1,\ldots,y_n)) = 1$. Thus, in this case, $\text{SIM}_{\text{fuzzy}}$ answers the query by invoking its oracle F_x^δ.

The challenging case is when a query contains two variables $z_{i,0}$ and $z_{i,1}$ for some $i \in [n]$; we call such queries "illegal". In this case, we do not know how to answer the query directly (e.g., as we did in Lemma 8.2). The idea of answering the query is that if Th_k returns 1 on the query, then the simulator can find a point w such that $F_x^\delta(w) = 1$ (as explained below), from such point it finds x (using Claim 8.3), computes $(z_{i,b})_{i\in[n],b\in\{0,1\}}$ as $\text{OBF}_{\text{fuzzy}}$ does, and answers the current and future queries using these values. If the simulator does not find such point w, then it returns 0.

Consider a query to Th_δ that contains exactly α pairs $z_{i,0}$ and $z_{i,1}$ for some $\alpha > 0$ and assume that the answer to the query is 1. Without loss of generality, the query is

$$(z_{i,y_i})_{1\le i\le n-2\alpha},\ z_{n-2\alpha+1,0},\ z_{n-2\alpha+1,1},\ldots,\ z_{n-\alpha,0},\ z_{n-\alpha,1}$$

for some $y_1, \ldots, y_{n-2\alpha}$. The value of exactly α of the variables

$$z_{n-2\alpha+1,0}, z_{n-2\alpha+1,1}, \ldots, z_{n-\alpha,0}, z_{n-\alpha,1}$$

is 1, thus, $\sum_{i=1}^{n-2\alpha} z_{i,y_i} + \alpha \geq (1-\delta)n$. Furthermore,

$$\sum_{i=n-2\alpha+1}^{n} z_{i,0} + \sum_{i=n-2\alpha+1}^{n} z_{i,1} = 2\alpha,$$

i.e., at least one of the sums is at least α. This implies that if the answer to the query is 1, then $\mathrm{Th}_x^\delta(y_1, \ldots, y_{n-2\alpha}, 0, \ldots, 0) = 1$ or $\mathrm{Th}_x^\delta(y_1, \ldots, y_{n-2\alpha}, 1, \ldots, 1) = 1$. Therefore, for each "illegal" query, the simulator asks two queries to the oracle Th_x^δ; if the answers to both of them are zero, the simulator answers 0 to the query. Otherwise, the simulator uses Claim 8.3 to find x, computes $(z_{i,b})_{i\in[n],b\in\{0,1\}}$ as $\mathrm{OBF}_{\mathrm{fuzzy}}$ does, and answers all further queries of $\mathrm{SIM}_{\mathrm{Th}}$ using these values. The VBB security of Π_{Th} implies thet (3) holds. □

Acknowledgments. We thank David Cash and David Wu for helpful discussions about Order Revealing Encryption.

The first author was supported by ISF grant 544/13 and by a grant from the BGU Cyber Security Research Center. The second and third authors were partially supported by ISF grant 1709/14, BSF grant 2012378, and NSF-BSF grant 2015782. Research of the second author was additionally supported from a DARPA/ARL SAFEWARE award, NSF Frontier Award 1413955, NSF grants 1619348, 1228984, 1136174, and 1065276, a Xerox Faculty Research Award, a Google Faculty Research Award, an equipment grant from Intel, and an Okawa Foundation Research Grant. This material is based upon work supported by the Defense Advanced Research Projects Agency through the ARL under Contract W911NF-15-C-0205. The views expressed are those of the authors and do not reflect the official policy or position of the Department of Defense, the National Science Foundation, or the U.S. Government.

References

1. Agrawal, R., Kiernan, J., Srikant, R., Xu, Y.: Order preserving encryption for numeric data. In: Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data, pp. 563–574 (2004)
2. Applebaum, B., Raykov, P.: From private simultaneous messages to zero-information Arthur-Merlin protocols and back. In: Kushilevitz, E., Malkin, T. (eds.) TCC 2016. LNCS, vol. 9563, pp. 65–82. Springer, Heidelberg (2016). doi:10.1007/978-3-662-49099-0_3
3. Barak, B., Goldreich, O., Impagliazzo, R., Rudich, S., Sahai, A., Vadhan, S.P., Yang, K.: On the (im)possibility of obfuscating programs. J. ACM **59**(2), 6 (2012)
4. Beimel, A., Gabizon, A., Ishai, Y., Kushilevitz, E.: Distribution design. In: Sudan, M. (ed.) Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science, pp. 81–92. ACM, New York (2016)
5. Beimel, A., Gabizon, A., Ishai, Y., Kushilevitz, E., Meldgaard, S., Paskin-Cherniavsky, A.: Non-interactive secure multiparty computation. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014. LNCS, vol. 8617, pp. 387–404. Springer, Heidelberg (2014). doi:10.1007/978-3-662-44381-1_22

6. Bellare, M., Stepanovs, I.: Point-function obfuscation: a framework and generic constructions. In: Kushilevitz, E., Malkin, T. (eds.) TCC 2016. LNCS, vol. 9563, pp. 565–594. Springer, Heidelberg (2016). doi:10.1007/978-3-662-49099-0_21
7. Bitansky, N., Canetti, R., Kalai, Y.T., Paneth, O.: On virtual grey box obfuscation for general circuits. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014. LNCS, vol. 8617, pp. 108–125. Springer, Heidelberg (2014). doi:10.1007/978-3-662-44381-1_7
8. Boldyreva, A., Chenette, N., Lee, Y., O'Neill, A.: Order-preserving symmetric encryption. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 224–241. Springer, Heidelberg (2009). doi:10.1007/978-3-642-01001-9_13
9. Boldyreva, A., Chenette, N., Lee, Y., O'neill, A.: Order-preserving symmetric encryption. Technical report 2012/624, IACR Cryptology ePrint Archive (2012). http://eprint.iacr.org/2012/624
10. Boldyreva, A., Chenette, N., O'Neill, A.: Order-preserving encryption revisited: improved security analysis and alternative solutions. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 578–595. Springer, Heidelberg (2011). doi:10.1007/978-3-642-22792-9_33
11. Boldyreva, A., Chenette, N., O'Neill, A.: Order-preserving encryption revisited: improved security analysis and alternative solutions. Technical report 2012/625, IACR Cryptology ePrint Archive (2012). http://eprint.iacr.org/2012/625
12. Boneh, D., Lewi, K., Raykova, M., Sahai, A., Zhandry, M., Zimmerman, J.: Semantically secure order-revealing encryption: multi-input functional encryption without obfuscation. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9057, pp. 563–594. Springer, Heidelberg (2015). doi:10.1007/978-3-662-46803-6_19
13. Brakerski, Z., Komargodski, I., Segev, G.: Multi-input functional encryption in the private-key setting: stronger security from weaker assumptions. In: Fischlin, M., Coron, J.-S. (eds.) EUROCRYPT 2016. LNCS, vol. 9666, pp. 852–880. Springer, Heidelberg (2016). doi:10.1007/978-3-662-49896-5_30
14. Feige, U., Kilian, J., Naor, M.: A minimal model for secure computation. In: Proceedings of the 26th ACM Symposium on the Theory of Computing, pp. 554–563 (1994)
15. Goldwasser, S., Gordon, S.D., Goyal, V., Jain, A., Katz, J., Liu, F.-H., Sahai, A., Shi, E., Zhou, H.-S.: Multi-input functional encryption. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 578–602. Springer, Heidelberg (2014). doi:10.1007/978-3-642-55220-5_32
16. 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). doi:10.1007/978-3-540-70936-7_11
17. Ishai, Y., Kushilevitz, E.: Private simultaneous messages protocols with applications. In: 5th Israel Symposium on Theory of Computing and Systems, pp. 174–183 (1997)
18. Ishai, Y., Kushilevitz, E., Paskin, A.: Secure multiparty computation with minimal interaction. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 577–594. Springer, Heidelberg (2010). doi:10.1007/978-3-642-14623-7_31
19. Lynn, B., Prabhakaran, M., Sahai, A.: Positive results and techniques for obfuscation. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 20–39. Springer, Heidelberg (2004). doi:10.1007/978-3-540-24676-3_2
20. Yao, A.C.: How to generate and exchange secrets. In: Proceedings of the 27th IEEE Symposium on Foundations of Computer Science, pp. 162–167 (1986)