Multi-Party Indirect Indexing and Applications

Matthew Franklin, Mark Gondree, and Payman Mohassel

Department of Computer Science
University of California, Davis
{franklin, gondree, mohassel}@cs.ucdavis.edu

Abstract. We develop a new multi-party generalization of Naor-Nissim indirect indexing, making it possible for many participants to simulate a RAM machine with only poly-logarithmic blow-up. Our most efficient instantiation (built from length-flexible additively homomorphic public key encryption) improves the communication complexity of secure multi-party computation for a number of problems in the literature. Underlying our approach is a new multi-party variant of oblivious transfer which may be of independent interest.

Keywords: communication complexity, oblivious RAM machine, privacy-preserving protocols, secure multiparty computation.

1 Introduction

Naor-Nissim indirect indexing [24] allows two parties to privately access an array at a shared index. We develop a multiparty generalization of Naor-Nissim indirect indexing, and show that our methods have many cryptographic applications. For example, we can transform any non-private multiparty protocol into a private one, in a manner that preserves its communication efficiency. Further, we can construct a multiparty generalization of Naor-Nissim circuits with look-up tables [24], enabling any number of parties to privately and obliviously simulate a RAM machine with only polylogarithmic overhead. The tools we build also yield automatic generalizations and efficiency improvements for several other protocols, including those for secure distributed constraint satisfaction [34, 35, 39, 29] and private stable matching [18, 11].

Underlying our techniques is a useful multiparty generalization of oblivious transfer (mOT), which may be of independent interest. In mOT, the role of the chooser is divided among many participants, each of whom holds a share of an input and receives a share of the output. We define this primitive and its related security notions, and provide two main constructions. Our first construction is generic, and can be built from black-box access to any ordinary two-party oblivious transfer. Our second construction is highly efficient and uses length-flexible additively homomorphic public key encryption [8, 9].

The paper is organized as follows. In Section 2, we define our multiparty generalization of Naor-Nissim indirect indexing. In Section 3, we show how this tool yields multiparty generalizations of existing protocols and efficiency improvements in existent multiparty protocols. In Section 4, we reduce the construction of multiparty indirect indexing to that of a simpler protocol, which can be seen as a multiparty variant of the well-known oblivious transfer primitive. In Section 5, we provide an efficient construction for this new protocol.

1.1 Background and Related Work

General secure multiparty computation (*e.g.*, see [14, 15]) can be used to privately implement the functions of interest in our paper, though rather inefficiently. Particularly, the communication complexity of such a construction for our mOT function would be linear in the size of the database. We are most interested in protocols with sublinear communication complexity.

Ostrovsky and Shoup [31] design communication-efficient protocols for the case where the database is shared between k servers and the index to be accessed is held by a single chooser. Only the chooser will learn the element in this position. Our setting is more general, as the index and final output cannot be learned by any one party, and are instead shared. As a result, our protocols automatically give new constructions for the problem considered by Ostrovsky and Shoup. Their goal, however, is information-theoretic security, while we work in the computational setting.

Naor and Pinkas [26] introduce *distributed oblivious transfer* which distributes the task of the database among multiple servers to compute the standard oblivious transfer functionality. Unconditional security is guaranteed as long a limited number of these participants do not collude. Unlike our mLUT protocol, the database is not shared explicitly between the servers. Instead, the database sends these servers a "transfer function," which allows each to compute a value related to the original database. From these values, the chooser can compute the original desired value in the database.

Barkol and Ishai [2] design a communication-efficient secure multiparty protocol in which m parties share an input x, and all hold the same constant-depth circuit C. Parties then privately compute $C(x)$. Let $x = \sigma$ be an index shared between the parties and let circuit C hard-code elements of a database Δ and return the x-th element as its output. Our construction is different in the sense that the database and the final output are not known to any single party and are shared instead. These are crucial properties that we need in order to securely implement multiparty circuits with look-up tables.

Since its proposal by Rabin [33], oblivious transfer has been a widely studied primitive and many variants, reductions, and applications have been considered. Even, Goldreich and Lempel [10] formalized 1-out-of-2 OT as a generalization of Rabin's OT. This was further generalized by Brassard, Crépeau and Robert [4] into 1-out-of-n OT, under the name "all-or-nothing disclosure of secrets." We believe that the mOT primitive may be of independent interest. Goldreich and Vainish [17] and Killian [20] show that OT is a complete primitive in the sense that two parties can compute any circuit securely using only blackbox access to OT. Goldreich [15] provides a nice presentation of the completeness of OT using a linear (in the circuit size) number of invocations of 1-out-of-4 two-party OT. Our mOT primitive directly translates this result to the case of general multiparty computation in a straight-forward fashion, yielding a new proof of this result. It also leads to new proofs for other results in general secure multiparty computation such as, for example, given a secure two-party OT protocol, n parties can compute any function n-privately (*e.g.*, see [14]), given secure channels, n parties can compute any function t-privately (information theoretically) for $t < n/2$ (*e.g.*, see [3]), and similar results.

In concurrent and independent work, Ishai *et al.* [19] design an mOT protocol under the name "distributed OT." Both our protocol and theirs involve the use of efficient PIR protocols, though in different ways. Thus, our work gives new constructions for the results in their paper. Comparing our two tools, our database performs $O(n)$ work where theirs performs $O(n^2)$, where n is the size of the database. While both tools are comparable in terms of communication efficiency, theirs is only efficient in this sense under some limitations on the number of parties m, since the size of the messages passed in their scheme is linear in m. The length of the messages passed in our protocol is independent of the number of parties, and thus we impose no limit on the number of parties involved in our protocols. Additionally, our protocol has a logarithmic (in n) round complexity, while theirs has a linear (in m) round complexity (the database's response is a $\log n$-iterated encryption in the former, and an $m-1$-iterated encryption in the later).

1.2 Definitions and Notation

We use the following definitions and notations.

Notation 1 *We denote the negation of bit* b *by* $\neg b$.

Definition 2 **(t-privacy).** *A protocol is* t-private *if any set of at most* t *participants cannot compute after the protocol more then they could jointly compute solely from their set of private inputs and outputs.*

Notation 3 (Asymptotic notation) *We use the following asymptotic notation:* $o(f)$ *denotes that the asymptotic upper bound* f *is not tight;* $\Omega(f)$ *denotes that* *the asymptotic lower bound f is tight; and $\widetilde{O}(f)$ denotes the asymptotic upper bound $O(f)$, ignoring* $\mathrm{polylog}(f)$ *factors.*

Notation 4 (Share notation) *We let $([\delta]_1, [\delta]_2, \ldots, [\delta]_m)$ be the collection of the shares of δ split among m parties via some secret-sharing scheme, so that player i holds the share $[\delta]_i$. When the subscript can be determined from context, we abuse notation and omit the subcript for ease of exposition; thus, we may denote the share of player i as, simply, $[\delta]$.*

2 Secure Multiparty Computation with Look-Up Tables

Naor and Nissim [24] define and give a secure two-party protocol for circuits with look-up tables. In the computational model of *circuits with look-up tables*, gates of a circuit are represented by look-up tables (LUT). The LUT input wires define the table entries and an index, and the LUT output wires are set according to the value stored in the indexed position. The protocol for private LUT serves as a building block in a protocol for privately evaluating circuits with LUT (a variant of the garbled circuit transformation). Here, we extend the definition of the look-up table primitive to the multiparty case.

Definition 5 (Multiparty LUT). *In a multiparty LUT (mLUT) protocol, all the parties are both a chooser and a database holder. Each party i holds a share of the database Δ, and a share of the index σ. At the end of the protocol, each party learns a share of δ_σ, the element at position σ in database Δ. Let $\Delta = (\delta_0, \ldots, \delta_{n-1})$. Let party i's share of δ be denoted by $[\delta]_i$. Then, the mLUT protocol can be summarized by the following protocol Π.*

$$\Pi([\Delta]_1, [\sigma]_1; [\Delta]_2, [\sigma]_2; \ldots; [\Delta]_m, [\sigma]_m) \rightarrow ([\delta_\sigma]_1; [\delta_\sigma]_2; \ldots; [\delta_\sigma]_m)$$

Definition 6 (Private mLUT). *We call a mLUT protocol t-private if no coalition of up to t parties can learn any information about σ or any of the elements in Δ.*

Circuits with LUT amount to performing computations with tables as follows. (1) **Read operations:** The table values as well as the index specifying the location of the read item are either preset or the result of an intermediate computation. In particular, it is possible to perform any kind of indirect read. (2) **Write operations:** The value written to the table may be the result of an intermediate operation but the location should be predetermined. In other words, no indirect writes are allowed.

It follows that any computation on a RAM machine where write operations are oblivious, in the sense that the time and location of the write operations should not depend on the input and randomness, may be emulated by circuits with LUT.

Results of Pippenger and Ficher [32] imply that when considering circuits vs. Turing Machines there is no significant advantage to the latter since there exists a series of circuits of size comparable to the running time of the Turing Machine. Currently it is not known whether a similar result applies to circuits vs. RAM machines. Particularly, there is a potential gap between the two, *i.e.* a computation on a RAM machine may be much more efficient than any circuit family. But for circuits with LUT this gap is closed. Particularly, note that for any write-oblivious RAM machine M running in time $T(n)$, there exists a family of circuits with LUT of size $T(n)$ computing f_M. Now, all one needs to show is an efficient simulation of any RAM machine using a write-oblivious RAM machine. Such a simulation exists, with polylogarithmic blow-up [16, 24]. Specifically, for any RAM machine M running in time $T(n)$ using space $S(n)$, there exist a series of circuits with LUT of size $T(n)\mathrm{polylog}(S(n))$ computing f_M.

3 Applications

Although we have not yet provided a private protocol for multiparty LUT (mLUT), we show how such a protocol leads to immediate efficiency improvements for several privacy-preserving protocols in the literature and efficient multiparty generalizations of existing two-party protocols.

We note that by replacing the two-party private LUT of Naor and Nissim [24] with a private construction of mLUT, we generalize all the constructions given in that paper to the multiparty case. In Appendix A of the full version of this paper [12], we present a multiparty generalization of the communication complexity model and a transformation which makes any efficient, non-private protocol in this model into an efficient, private protocol with the same functionality. Also, a private mLUT protocol automatically yields the ability to simulate, as a multiparty computation, a private oblivious RAM machine with only a polylog (in size of the RAM) blowup in communication between the parties.

Furthermore, we believe our mLUT protocol to be useful in a variety of existing applications, such as private multiparty sampling protocols [19], distributing the function of an "auction issuer" in Naor-Pinkas-Sumner style auctions [27], private approximation protocols, and any setting where a global decision is privately computed using access to some of the inputs of several parties. In the remainder of this section, we discuss applying our tools to two such domains: protocols for distributed constraint satisfaction problems, and protocols for the stable matching problem.

3.1 Private DisCSPs

Distributed constraint satisfaction problems (DisCSPs) are composed of agents holding local variables, and a constraint network that restricts the legal assignments to agents' variables. A solution to a DisCSP is an assignment to variables that is in agreement with all the constraints ([38, 36]). To achieve this goal, agents run a protocol where they check assignments to their and other agents' variables for consistency. Distributed CSPs are an elegant model for many every day combinatorial problems that are distributed by nature, such as meeting scheduling [13, 23] in which agents attempt to schedule meetings according to their constrained personal schedule.

Nissim and Zivan [29] design new secure protocols for DisCSPs based on advanced search heuristics. The first protocol they design is a *centralized* protocol, where two of the agents collect "encrypted" data from all other parties, and obliviously perform a search algorithm. Their centralized algorithm avoids information leakage to all agents. their second protocol makes the first step toward a feasible *distributed* secured protocol for solving DisCSPs. They construct a network, whose nodes are small groups (*e.g.* pairs) of agents, from the original DisCSPs. Each node group obliviously performs the roles of all its members in the search algorithm. This protocol has the following disadvantages (1) it is *not fully distributed* and a small collusion of agents could learn information about the other participants' private inputs. (2) As mentioned in the paper, the protocol is not perfectly secure, *i.e.* the communication pattern in the protocol leaks information about the agents' private inputs.

Using our private construction for multiparty computation of circuits with LUT, we can securely extend the centralized protocol given in section 5 of [29] to a fully distributed one without adding any overhead in the communication or computation of their protocol. More specifically, the agents will collectively share the private data and obliviously perform the search algorithm. This leads to the first *fully distributed* and *completely secure* protocol for DisCSPs. For completeness, we include a brief description of our construction in Appendix B of the full version of this paper [12].

3.2 Private Stable Matching

Golle [18] initiated the study of privacy-preserving protocols for stable matching, arguing persuasively that such protocols could have great practical benefit. In Golle's framework, m "matching authorities" receive the encrypted preference lists from the participants and then perform a secure multiparty computation to return the stable matching to the participants. Franklin *et al.* [11] revisit

Golle's work and design substantially more efficient protocols for private stable matching in this framework.

Naor, Pinkas, and Sumner [27] observe, in considering this problem as a possible domain for their paper's techniques, that the algorithm for solving the stable matching problem requires the power of indirect addressing of a RAM and, thus, its translation into a circuit is rather inefficient. Indeed, the stable matching algorithm of Franklin *et al.* [11] can be efficiently implemented as a circuit of size $O(n^2)$ with access to a RAM. More specifically, one can implement their algorithm [11, Section 5] in the multiparty setting[1] by implementing their array/matrix accesses using our mLUT protocol. In this way, we extend this (very efficient) construction of theirs from two-party to multiparty, yielding a protocol in the same framework as Golle and Franklin *et al.*, but a factor of n more efficient than previous private stable matching protocols. The following table compares our results with those of the previous work.

Protocol	Total Work	Total Communication	Round Complexity
Golle [18]	$O(n^5)$	$O(mn^5)$	$\widetilde{O}(n^3)$
Franklin *et al.* [11]	$O(n^4\sqrt{\log n})$	$O(mn^3)$	$\widetilde{O}(n^2)$
Ours	$O(n^4)$	$O(mn^2)$	$\widetilde{O}(n^2)$

4 Protocols for private mLUT

In this section, we reduce the problem of constructing a protocol for private mLUT to a subproblem we call "generalized multiparty oblivious transfer." First we define this subproblem, and then we show our construction for mLUT. Later, we define a related protocol we call "multiparty oblivious transfer" and draw connections between this new primitive and general multiparty computation. Finally, in Section 5, we give a construction for an efficient, private g-mOT protocol, completing our private mLUT construction.

4.1 A construction for private mLUT

Our construction for the private mLUT protocol invokes a protocol called *generalized multiparty oblivious transfer* (g-mOT) for each share of the database. Parties get their shares of the output for each run of the g-mOT protocol and

[1] Franklin *et al.* generalize this two-party protocol to the multiparty case, but the resulting protocol is only secure in a new security model where one considers collections of pairs of matching authorities, where each pair is honest-majority. Our generalization is secure in the standard passive adversary security model where up to a certain threshold of players may be corrupted.

combine their shares in the appropriate way to compute shares of the indexed position in the original database Δ. We define generalized mOT below, and then describe this protocol in more detail.

Definition 7 (Generalized multiparty oblivious transfer). *Generalized multiparty oblivious transfer (g-mOT) is a protocol involving m parties where: at the beginning of the protocol, each party holds a share of a secret index σ and one distinguished party holds a table of n bits, the database $\Delta = (\delta_0, \ldots, \delta_{n-1})$; at the end of the protocol, each party holds a share of the database element δ_σ. In the terminology of oblivious transfer, every party is a chooser and one party is also the database. The protocol Π for $\binom{n}{1}$–$\mathsf{g\text{-}mOT}(m, t)$ can be summarized as:*

$$\Pi(\Delta, [\sigma]; [\sigma]; \ldots; [\sigma]) \rightarrow ([\delta_\sigma]; [\delta_\sigma]; \ldots; [\delta_\sigma])$$

We give a full security description of g-mOT later but, for our mLUT construction, we only require that this protocol be t-private.

For simplicity, we assume that the outputs and database are shared using XOR sharing in the construction below. Any other sharing scheme would work fine, however, as the overhead for switching between different sharing methods does not effect the overall complexity of our protocols. Again, let m be the number of parties participating in the protocol. Let chooser i hold $\Delta^i = [\Delta]_i$, where $\oplus \Delta^i = \Delta$. The protocol is outlined below.

Inputs: Each party holds a share of the database $\Delta = (\delta_0, \ldots, \delta_{n-1})$ and a share of the index σ.
Output: Each party holds a share of δ_σ.

- For $i = 1$ to m:
  - Parties run
    g-mOT$(\Delta^i, [\sigma]; [\sigma]; [\sigma]; \ldots; [\sigma]) \rightarrow ([\delta^i_\sigma]; [\delta^i_\sigma]; \ldots; [\delta^i_\sigma])$.
- Participant i locally computes a share of δ_σ as $[\delta_\sigma] = \oplus[\delta^j_\sigma]$.

Claim. The complete protocol is a t-private multiparty LUT. The protocol has $O(k\ell \log^2 n \mathrm{poly}(m))$ communication complexity and $O(\log n)$ round complexity, where k is a security parameter, m is the total number of parties, and the database is composed of n strings of bit-length ℓ.

*Proof (*Proof *(sketch)).* Our mLUT protocol uses m invocations of a generalized mOT protocol. Thus, the communication complexity of our mLUT construction is simply m times that of the g-mOT protocol from Section 5.2. Since we can run the generalized mOT protocols in parallel, the round complexity of the mLUT protocol remains the same as that of the g-mOT protocol. The t-privacy of the mLUT protocol follows from general composition theorems [5, 15] and the t-privacy of our g-mOT protocol.

4.2 Multiparty oblivious transfer

Before we give a construction for an efficient t-private generalized multiparty oblivious transfer protocol, we explore a related protocol we call multiparty oblivious transfer. We also give a detailed security definition for these protocols, as there may be interesting applications that require something stronger than t-privacy.

Multiparty oblivious transfer (mOT) is a protocol involving $m' + 1$ parties: m' choosers and a database. Each chooser holds a share of a secret index $\sigma \in [0, n-1]$. The database holds a table[2] of n bits, $\Delta' = (\delta_0, \ldots, \delta_{n-1})$. At the end of the protocol, each chooser holds a share of the database element δ_σ. The protocol Π for $\binom{n}{1}$–$\mathsf{mOT}(m', t)$ can be summarized as follows:

$$\Pi(\Delta'; [\sigma]_1; \ldots; [\sigma]_{m'}) \to (\emptyset; [\delta_\sigma]_1; \ldots; [\delta_\sigma]_{m'})$$

We consider mOT for its simplicity and because, in many scenarios, g-mOT reduces to mOT. For example, by letting $m = m' + 1$ it is clear that, when the inputs and outputs are XOR shares, there is a simple reduction of g-mOT to mOT. More specifically, the database in the g-mOT protocol can compute the database Δ' by permuting Δ according to x_0 (his share of the secret index) and blinding each entry by a random y_0 (his share of the output). Considering XOR shares, then, generalized mOT reduces to an invocation of the following mOT protocol Π.

$$\Pi(\Delta'; x_1, \ldots; x_{m'}) \to (\emptyset; y_1; \ldots; y_{m'}) \text{ where } \bigoplus_{i=0}^{m'} x_i = \sigma \text{ and } \bigoplus_{i=0}^{m'} y_i = \delta_\sigma$$

Definition 8 (Secure mOT). *Following Naor and Pinkas [26], we give a detailed, four-parameter security definition for this new variant of oblivious transfer. We relate this definition to the more common and intuitive security notion of t-privacy. We say the mOT protocol is (t_1, t_2, t_3, t_4)-secure if, when all the participants follow their steps properly (i.e., considering a passive adversary), the following properties are met:*

> input t_1-privacy*: no coalition of up to t_1 choosers should be able to learn any information about σ.*

[2] In Section 5.2, we consider a generalization of this definition, where the database is a table of n strings, each of length ℓ.

output t_2-privacy*: no coalition of up to t_2 choosers should be able to learn any information about δ_σ.*

chooser t_3-privacy*: the database should not be able to learn any information about σ, even when colluding with up to t_3 other participants.*

database t_4-privacy*: no coalition of up to t_4 non-database players should be able to learn any information about δ_j for $j \neq \sigma$.*

We could easily create information theoretic and computational variants of this definition by specifying the power of the adversary accordingly.

Remark 1. The following are automatic consequences.

- (t_1, t_2, t_3, t_4)-security implies $\min(t_1, t_2, t_3 + 1, t_4)$-privacy.
- It is necessary that $t_3 \leq \min(t_1, t_2)$. For g-mOT this becomes strict, $t_3 < \min(t_1, t_2)$.
- For g-mOT, since the database is a chooser, there is always a collusion of $t_3 + 1$ choosers who can learn σ, so $t_1 = t_3 + 1$. Furthermore, $t_1 = t_2$ because, for the database, learning σ implies learning δ_σ (and vice versa). Thus, for g-mOT, t-privacy implies $(t, t, t-1, t_4)$-security, for some $t_4 \geq t$.
- If the players are computationally unbounded, it must be the case that $(m' + 1)/2 > \min(t_1, t_2, t_3 + 1, t_4)$, or else we contradict known results for the privacy of unconditionally secure multiparty computation.

5 Protocols for private mOT and g-mOT

In this section, we give two constructions for multiparty oblivious transfer. The first mOT construction uses blackbox access to two-party oblivious transfer, showing that mOT can be constructed under a variety of complexity assumptions. The second is a construction of g-mOT which we rely on for our earlier applications, as it is efficient in terms of communication complexity. We leave open the problem of finding a fully black-box transformation of two-party oblivious transfer into multiparty oblivious transfer with sublinear (in size of the database) blowup in communication complexity.

5.1 A generic construction for 1-out-of-2 mOT

Here, we describe a generic construction for a 1-out-of-2 mOT protocol, using blackbox access to a two-party oblivious transfer protocol. For this construction, we consider the case where the secret σ is shared among the m' choosers using XOR sharing. Let chooser i hold share b_i and $\oplus b_i = \sigma$.

1. The database chooses $2m'$ bits, $\{(r_0^1, r_1^1), (r_0^2, r_1^2), \ldots, (r_0^{m'}, r_1^{m'})\}$ uniformly at random, such that the bits satisfy the following condition:
$$\bigoplus_{i=1}^{m'} r_{b_i}^i = \delta_{\oplus b_i}$$
2. For all $1 \leq i \leq m'$
   Chooser i and the database run a two-party oblivious transfer protocol, where the chooser's private input is b_i and the database's private input is the two element "database" (r_0^i, r_1^i).
3. The output for chooser i is $r_{b_i}^i$ which, according to the previous condition, is an XOR share of $\delta_{\oplus b_i} = \delta_\sigma$.

It is clear that the values of the $2m'$ variables which satisfy the above condition are precisely the solutions to the following set of $m' + 1$ linear equations:

$$\left\{r_1^i = \delta_0 \oplus \delta_1 \oplus r_0^i \mid i < m'\right\},\ r_0^{m'} = \delta_0 \oplus \bigoplus_{i=1}^{m'-1} r_0^i \text{ and } r_1^{m'} = \delta_1 \oplus \bigoplus_{i=1}^{m'-1} r_1^i$$

In this form, it is easier to see that the database can find a random solution to the above system by simply choosing the values for variables $\{r_0^i \mid i < m'\}$ uniformly at random. The remaining values are uniquely defined.

When the two-party oblivious transfer protocol is private, the above mOT protocol is $(m' - 1)$-private. This construction is essentially the same as that of Crépeau and Kilian [6], though in a different context, and our proof of security follows directly from theirs.

This 1-out-of-2 mOT construction protocol can be turned into a 1-out-of-n mOT protocol using a variant of the Brassard-Crépeau-Robert transform [4] which constructs 1-out-of-n oblivious transfer from (a linear number of invocations of) the 1-out-of-2 variant. While these constructions are not particularly efficient, they do demonstrate that mOT protocols can be constructed under a variety of standard cryptographic assumptions and in the information-theoretic case. For example, given secure channels, each two-party OT protocol can be replaced with the distributed OT (dOT) protocol of Naor and Pinkas [26]. Briefly, in a (r, m, ℓ, t)-dOT protocol, the database sends messages to m servers[3] and the chooser contacts r of the servers to reconstruct δ_σ, where no coalition of less than t servers learns σ and no coalition of the chooser with less than ℓ servers can compute more than can be jointly computed from these participant's

[3] We note the database itself might play the role of a server, sending itself a message, causing dOT to be a protocol among $m + 1$ parties.

inputs and outputs. A straight-forward argument of Nikov *et al.* [28] shows that a necessary and sufficient condition for dOT is $r \geq t + \ell$. Thus, our mOT protocol based on dOT will be τ-private for $\tau < \min(\ell + 1, t)$. Since $r \leq m$, this condition implies our mOT protocol is τ-private for $\tau < (m+1)/2$.

Using this construction for mOT instead of OT in a proof of the completeness of OT such as Goldreich's [15, §7.1.3.3] yields new proof that (given secure channels) n parties can compute any function τ-privately (information theoretically) for $\tau < n/2$. The original presentation of this result, due to Ben-Or, Goldwasser, and Wigderson [3], uses polynomial shares and requires a special, private polynomial degree-reduction technique to handle the degree growth during the interactive multiplication steps. This new proof avoids such complicated machinery. In fact, using a basic proof of the completeness of mOT while building mOT out of different tools (*e.g.*, secure channels, secure channels and one-way functions, two-party OT, etc) yields new proofs for a variety of interesting results in secure multiparty computation.

5.2 A construction for 1-out-of-n g-mOT

In this section, we describe a generic construction of a 1-out-of-n generalized multiparty oblivious transfer protocol. At a high level, the construction can be viewed as a non-black-box transformation from a two-party private information retrieval (PIR) protocol (see [30] for a recent survey). First, the two-party PIR protocol is converted into a two-party OT protocol. The owners of the secret sharing scheme engage in a multiparty computation, t-privately transforming their shares of σ into the messages $\bar{m}_0$ that would be sent to the database during the two-party OT protocol. A single chooser and the database then engage in the message passing of the original PIR protocol. The received messages $\bar{m}_1$ are then used as inputs to another multiparty computation, t-privately converting these messages into shares of δ_σ. In this construction, the sharing used for the inputs and outputs is some t-out-of-m linear secret sharing scheme with security parameter k, owned by an appropriate subset of the choosers.

One particularly efficient instantiation of our construction can be built using a two-round PIR protocol, the length-flexible additively homomorphic public key encryption [8, 9] and design ideas of Aiello-Ishai-Reingold [1]. In the remainder of this section, we discuss this highly efficient instantiation. The steps of this protocol are assembled in order and summarized below.

1. The choosers collaborate to create a (t-out-of-m) threshold, length-flexible, additively homomorphic) encryption system.
2. The choosers collaborate to compute the PIR scheme's first message $\bar{m}_0$, using their shares of σ (see Section 5.2).
3. The choosers send the public parameters, $E(\sigma)$, and $\bar{m}_0$ to the database.
4. The database uses $E(\sigma)$ to blind the database, according to the Aiello-Ishai-Reingold transform (see Section 5.2).
5. The database runs the PIR protocol as usual, using $\bar{m}_0$ and the blinded database (see Section 5.2).
6. The database sends its response $\bar{m}_1$ to the choosers.
7. The choosers collaborate to decrypt $\bar{m}_1$. In our case, they decrypt the response α times and then split the remaining ciphertext into shares (see Section 5.2).

Highly Efficient Two-Party PIR and OT A highly efficient two-party PIR scheme can be built from length-flexible additively homomorphic public key encryption [8, 9] using design ideas of Kushilevitz-Ostrovsky [21] (*e.g.*, following the presentation of Lipmaa [22]).

The database is composed of n ℓ-bit strings. The chooser takes her secret σ and constructs $\bar{q} = (\mathbf{q}_1, \ldots, \mathbf{q}_\alpha)$, the α-dimensional vector which indicates the position of σ in a $\lambda_1 \times \cdots \times \lambda_\alpha$ coordinate system. In this system, index $(i_1, \ldots, i_\alpha)$ is resolved in the following manner:

$$\Delta[(i_1, \ldots, i_\alpha)] = \Delta[i_1 \cdot \prod_{j=2}^{\alpha} \lambda_j + i_2 \cdot \prod_{j=3}^{\alpha} \lambda_j + \cdots + i_{\alpha-1} \cdot \lambda_\alpha + i_\alpha]$$

The first query sent to the database is the encryption of $\mathbf{q}_1$ with the corresponding public key. The database uses this to construct $\Delta[\mathbf{q}_1, i_2, \ldots, i_\alpha]$, a new database with $\alpha - 1$ dimensions. The next query is the encryption of $\mathbf{q}_2$, the first coordinate of the same element in this new database. We iterate in this fashion α times. This is a standard trick, due to Kushilevitz and Ostrovsky [21] and is used in the PIR scheme of Stern [37]. In the final round, the database's response is the α times encryption of δ_i. In fact this process happens in one round, since the encryption of $\bar{q} = (\mathbf{q}_1, \ldots, \mathbf{q}_\alpha)$ can be sent in a single message. When encryption is achieved using a *length-flexible* additively homomorphic public-key cryptosystem, this PIR protocol has $\Theta(k \log^2 n + \ell \log n)$ communication complexity, as shown by Lipmaa [22].

A modification of this PIR scheme, using the Aiello-Ishai-Reingold transform, yields a highly efficient OT scheme. The chooser encrypts σ using a homomorphic encryption scheme and sends this to the database with the corresponding public-key. The database takes advantage of the homomorphic property of the ciphertext to compute a new database where each entry δ_j is represented by $E(r_j(\sigma - j) + \delta_j)$, for some random r_j. Thus, for all $j \neq \sigma$, the j-th element of the database is the encryption of a random element. The original Aiello-Ishai-Reingold transform suggests that the homomorphic encryption scheme generated for this step be verifiable, such as the El-Gamal scheme, so the database can verify the correctness of the public-key sent by the chooser. As we consider only honest-but-curious adversaries, we can re-use the homomorphic encryption scheme used in the original PIR protocol and ignore the need for verifiable keys. The rest of the OT protocol proceeds just as in the original PIR protocol, but the database's response must now be decrypted $\alpha + 1$ times to recover δ_σ. This transformation increases the communication complexity by a term of $\ell + k(\log n + 1)$ bits, which does not effect the overall asymptotic complexity.

Input share conversion In our g-mOT scheme, the choosers hold shares of σ using some linear secret sharing scheme. We describe below how the choosers can engage in an efficient t-private multiparty protocol to convert their shares of σ into an encryption of $\bar{q} = (\mathbf{q}_1, \ldots, \mathbf{q}_\alpha)$. For simplicity, we represent the database as the $\alpha = \log n$-dimensional $2 \times \cdots \times 2$ system[4].

The choosers interact to define a t-out-of-m threshold version of the length-flexible homomorphic encryption scheme. In reality, $\mathbf{q}_i$ is a λ_i-length bit string of Hamming weight 1. Locally, the database uses $E(\mathbf{q}_i)$, the bit-wise encryption of this value, to process the representation of the database at step i. In our simplified scenario (for all i, $\lambda_i = 2$) this bit string is simply $\mathbf{q}_i = (\neg b_i, b_i)$, where b_i is the i-th bit in the binary representation of σ. In other words, if we let Δ^j denote the $\alpha - j$-dimensional database constructed in round j of the PIR protocol, then

$$\Delta^{j+1}[i] = \neg \mathbf{q_j} \cdot \Delta^j[i] + \mathbf{q_j} \cdot \Delta^j[2^i + 1]$$

Since the encryption of the negation of a bit can be computed by the database, trivially, via the homomorphic property, it suffices to let $\mathbf{q}_i = b_i$. Damgård *et al.* [7] provide efficient, private constant-round multiparty protocols for computing shares of the binary representation of a secret, from shares of the secret.

[4] For efficiency in communication complexity when using this representation, we require the use of length-flexible additively homomorphic encryption. It is possible to use a generic additively homomorphic encryption system and achieve sublinear communication complexity by using a different representation, at the cost of increasing the round complexity (by a factor of $\log n$) during this pre-processing phase. Such a choice would not effect the efficiency of the complete protocol.

Using the homomorphic property, the choosers' shares are encrypted and combined, and $E(\mathbf{q})$, $E(\sigma)$, and the public key are sent to the database by a chooser. From this, the database can run its portion of the OT protocol, and send its response.

Output conversion The response from the database is jointly decrypted α times by the choosers to recover $E(\delta_\sigma)$, the desired element encrypted using the same t-threshold (length-flexible) additively homomorphic encryption scheme. This is already, in a sense, a share of δ_σ. Using the homomorphic property, this ciphertext can be split into additive shares for the choosers, or a different type of sharing if desired.

5.3 Analysis

Claim. The complete protocol of Section 5.2 has $O(k\ell \log^2 n \mathrm{poly}(m))$ communication complexity and $O(\log n)$ round complexity, where k is a security parameter, m is the total number of players, and the database is composed of n strings of bit-length ℓ.

*Proof (*Proof *(sketch)).* The primitives used by the input share conversion protocol have $O(\mathrm{poly}(m, \log q))$ communication complexity, where q is the size of the field in which σ lives. Since σ is a pointer into a table of size n, the communication complexity becomes, in our case, $O(\mathrm{poly}(m, \log\log n)) = o(\mathrm{poly}(m) \log n)$. Also, the messages passed between the database and the other parties are the same as those passed during the oblivious transfer protocol from Section 5.2, whose communication complexity is $\Theta(k \log^2 n + \ell \log n)$. Thus, our complete protocol has $O(m(k \log^2 n + \ell \log n) + \mathrm{poly}(m) \log n) = O(k\ell \log^2 n \mathrm{poly}(m))$ communication complexity and $O(\log n)$ round complexity.

Claim. The complete protocol of Section 5.2 is t-private, assuming the threshold length-flexible additively homomorphic public-key encryption scheme is IND-CPA secure.

*Proof (*Proof *(sketch)).* The above security claim follows from the security of the share conversion protocols, from general composition theorems [5, 15], and from the same security arguments of [22] since (although we make use of the protocol in a non-blackbox manner) the transcript of the messages passed between the chooser and database in our protocol is identical.

More specifically, the g-mOT protocol is $(t, t, t-1, m)$-secure, because the Aiello-Ishai-Reingold transform makes the OT scheme information-theoretically database-private. When the PIR protocol is converted into an OT protocol using a transformation that provides computational sender privacy, like the Naor-Pinkas transform [25], the resulting mOT protocol is $(t, t, t-1, t)$-secure. The threshold, length-flexible homomorphic encryption scheme of Damgård and Jurik [9] is IND-CPA secure in the standard model, under the Paillier and composite DDH assumptions.

References

1. Bill Aiello, Yuval Ishai, and Omer Reingold. Priced oblivious transfer: How to sell digital goods. In *Proc. of Eurocrypt 2001*, pages 119–135, 2001.
2. Omer Barkol and Yuval Ishai. Secure computation of constant-depth circuits with application to database search problems. In *Proc. of CRYPTO'05*, 2005.
3. Michal Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant disributed computation. In *Proc. of STOC '88*, pages 1–10, 1988.
4. Gilles Brassard, Claude Crépeau, and Jean-Marc Robert. Information theoretic reductions among disclosure problems. In *Proc. of FOCS*, pages 168–173, 1986.
5. Ran Canetti. Security and composition of multiparty cryptographic protocols. In *Journal of Cryptology*, volume 13, pages 143–202, 2000.
6. Claude Crépeau and Joe Kilian. Achieving oblivious transfer using weakened security assumptions. In *Proc. of FOCS*, pages 42–52, 1988.
7. Ivan Damgård, Matthias Fitzi, Eike Kiltz, Jesper Buus Nielsen, and Tomas Toft. Unconditionally secure constant-rounds multi-party computation for equality, comparison, bits and exponentiation. In *Proc. of the Theory of Cryptography Conference (TCC)*, pages 285–304, 2006.
8. Ivan Damgård and Mads Jurik. A generalisation, a simplification and some applications of Paillier's probabilistic public-key system. In *Public Key Cryptography (PKC)*, pages 119–136, 2001.
9. Ivan Damgård and Mads Jurik. A length-flexible threshold cryptosystem with applications. In *Information Security and Privacy*, pages 350–364, 2003.
10. Shimon Even, Oded Goldreich, and Abraham Lempel. A randomized protocol for signing contracts. *Comm. of the ACM*, 28(6):637–647, 1985.
11. Matthew Franklin, Mark Gondree, and Payman Mohassel. Improved efficiency for private stable matching. In *Proc. of CT-RSA*, 2007.
12. Matthew Franklin, Mark Gondree, and Payman Mohassel. Multi-party indirect indexing and applications. Cryptology ePrint Archive, Report 2007/341, 2007.
13. Eugene C. Freuder and Richard J. Wallace. Constraint-based multi-agent meeting scheduling: effects of agent heterogeneity on performance and privacy loss. In *Proc. of the 3rd Workshop on Distributed Constraint Reasoning (DCR-02)*, pages 176–182, 2002.
14. O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority. In *Proc. of STOC '87*, pages 218–229, 1987.
15. Oded Goldreich. *Foundations of Cryptography*. Cambridge University Press, 2001.
16. Oded Goldreich and Rafail Ostrovsky. Software protection and simulation on oblivious rams. *Journal of the ACM*, 43(3):431–473, 1996.
17. Oded Goldreich and Ronen Vainish. How to solve any protocol problem - an efficiency improvement. In *Proc. of CRYPTO'87*, pages 73–86, 1987.
18. Philippe Golle. A private stable matching algorithm. In *Financial Crypto (FC '06)*, 2006.

19. Yuval Ishai, Tal Malkin, Martin J. Strauss, and Rebecca N. Wright. Private multiparty sampling and approximation of vector combinations. In *Proc. of International Colloquium on Automata, Languages and Programming (ICALP)*, 2007.
20. Joe Kilian. A general completeness theorem for 2-party games. In *Proc. of STOC '91*, pages 553–560, 1991.
21. Eyal Kushilevitz and Rafail Ostrovsky. Replication is not needed: Single database, computationally-private information retrieval. In *Proc. of FOCS*, pages 364–373, 1997.
22. Helger Lipmaa. Verifiable homomorphic oblivious transfer and private equality test. In *ASIACRYPT 2003*, pages 416–433, 2003.
23. A. Meisels and O. Lavee. Using additional information in DisCSP search. In *Proc. of the 5th Workshop on Distributed Constraint Reasoning (DCR-04)*, 2004.
24. Moni Naor and Kobbi Nissim. Communication preserving protocols for secure function evaluation. In *Proc. of STOC '01*, pages 590–599, 2001.
25. Moni Naor and Benny Pinkas. Oblivious transfer and polynomial evaluation. In *Proc. of STOC '99*, pages 245–254, 1999.
26. Moni Naor and Benny Pinkas. Distributed oblivious transfer. In *ASIACRYPT'00*, pages 205–219, 2000.
27. Moni Naor, Benny Pinkas, and Reuban Sumner. Privacy preserving auctions and mechanism design. In *EC '99: Proc. of the 1st ACM conference on Electronic Commerce*, pages 129–139. ACM Press, 1999.
28. Ventzislav Nikov, Svetla Nikova, Bart Preneel, and Joos Vandewalle. On unconditionally secure distributed oblivious transfer. In *INDOCRYPT '02: Proc. of the 3rd International Conference on Cryptology*, pages 395–408, 2002.
29. Kobbi Nissim and Roie Zivan. Secure DisCSP protocols - from centralized towards distributed solutions. In *Proc. of the 6th Workshop on Distributed Constraint Reasoning (DCR-05)*, 2005.
30. Rafail Ostrovsky and William E. Skeith III. A survey of single database PIR: Techniques and applications. Cryptology ePrint Archive, Report 2007/059, 2007.
31. Rafail Ostrovsky and Victor Shoup. Private information storage (extended abstract). In *Proc of STOC '97*, pages 294–303, 1997.
32. Nicholas Pippenger and Michael J. Fischer. Relations among complexity measures. *Journal of the ACM*, 26(2):361–381, 1979.
33. Michael O. Rabin. How to exchange secrets with oblivious transfer. Technical Report TR-81, Harvard University, 1981. Available as the Cryptology ePrint Archive Report 2005/187, at `http://eprint.iacr.org/`.
34. Marius-Calin Silaghi. Solving a distributed CSP with cryptographic multi-party computations, without revealing constraints and without involving trusted servers. In *Proc. of the 4th Workshop on Distributed Constraint Reasoning (DCR-03)*, 2003.
35. Marius-Calin Silaghi and Debasis Mitra. Distributed constraint satisfaction and optimization with privacy enforcement. In *Proc. of the 3rd International Conference on Intelligence Agent Technology*, pages 531–535, 2004.
36. G. Solotorevsky, E. Gudes, and A. Meisels. Modeling and solving distributed constraint satisfaction problems (DCSPs). In *Constraint Processing-96*, pages 561–562, 1996.
37. Julien P. Stern. A new and efficient all-or-nothing disclosure of secrets protocol. In *ASIACRYPT'98*, pages 357–371, 1998.
38. Makoto Yokoo. Algorithms for distributed satisfaction problems: A review. In *Autonomous Agents and Multi-Agent Sys.*, pages 198–212, 2000.
39. Makoto Yokoo, Koutarou Suzuki, and Katsutoshi Hirayama. Secure distributed constraint satisfaction: Reaching agreement without revealing private information. In *Artificial Intelligence*, pages 229–246, 2005.