How Many Oblivious Transfers are Needed for Secure Multiparty Computation?*

Danny Harnik**, Yuval Ishai***, and Eyal Kushilevitz†

Department of Computer Science, Technion, Haifa, Israel.
E-mail: {harnik,yuvali,eyalk}@cs.technion.ac.il.

Abstract. Oblivious transfer (OT) is an essential building block for secure multiparty computation when there is no honest majority. In this setting, current protocols for $n \geq 3$ parties require *each pair of parties* to engage in a single OT for *each gate* in the circuit being evaluated. Since implementing OT typically requires expensive public-key operations (alternatively, expensive setup or physical infrastructure), minimizing the number of OTs is a highly desirable goal.
In this work we initiate a study of this problem in both an information-theoretic and a computational setting and obtain the following results.

- If the adversary can corrupt up to $t = (1-\epsilon)n$ parties, where $\epsilon > 0$ is an arbitrarily small constant, then a total of $O(n)$ OT channels between pairs of parties are necessary and sufficient for general secure computation. Combined with previous protocols for "extending OTs", $O(nk)$ invocations of OT are sufficient for computing arbitrary functions with computational security, where k is a security parameter.
- The above result does not improve over the previous state of the art in the important case where $t = n - 1$, when the number of parties is small, or in the information-theoretic setting. For these cases, we show that an arbitrary function $f : \{0,1\}^n \to \{0,1\}^*$ can be securely computed by a protocol which makes use of a single OT (of strings) between each pair of parties. This result is tight in the sense that at least one OT between each pair of parties is necessary in these cases. A major disadvantage of this protocol is that its communication complexity grows exponentially with n. We present natural classes of functions f for which this exponential overhead can be avoided.

1 Introduction

Secure multiparty computation (MPC) [46, 23, 7, 11] provides a powerful and general tool for distributing computational tasks between mutually distrusting

* Research supported by grant 1310/06 from the Israel Science Foundation and the Technion VPR fund. Part of this research was done while visiting IPAM.
** Supported in part by a fellowship from the Lady Davis Foundation.
*** Supported by grant 2004361 from the U.S.-Israel Binational Science Foundation.
† Supported by grant 2002354 from the U.S.-Israel Binational Science Foundation.

parties without compromising the privacy of their inputs. We consider the problem of secure computation in the case where a majority of the parties can be corrupted. In this case, secure computation of nontrivial functions implies the existence of *oblivious transfer* (OT) [44, 40, 17] — a secure two-party protocol which allows a receiver to select one of two strings held by a sender and learn this string (but not the other) without revealing its selection. Moreover, OT can be used as a building block for general MPC protocols that tolerate an arbitrary number of corrupted parties [47, 24, 22, 33, 21]. These protocols involve a large number of OT invocations which typically constitute their efficiency bottleneck. Indeed, standard implementations of OT require expensive public-key operations, whereas alternative "information-theoretic" implementations of OT require either a trusted setup [3] or physical infrastructure [13] and may be viewed as being at least as expensive. Thus, minimizing the number of OTs in MPC protocols is a highly desirable goal.

How many OTs are needed to secure the world? In a world consisting of just two parties, this question was essentially answered by Beaver [4] (see also [31]). In a pure information-theoretic setting, ignoring computational efficiency issues, computing a two-argument function whose shorter input has length ℓ generally requires $\Theta(\ell)$ OTs. (Some specific functions require fewer OTs; see [15, 5] for a more refined study of the "OT complexity" of information-theoretic secure two-party computation.) Quite remarkably, it is possible to do much better if computational security is required. Assuming the existence of one-way functions, a "seed" of k OTs, where k is a security parameter, can be used for implementing an arbitrary polynomial number of OTs.[1] This implies that k OTs are sufficient for secure two-party computation of arbitrary functions, even ones whose input length is much bigger than k.

Given Beaver's result, it is natural to expect that k OTs would be sufficient for computationally secure MPC protocols involving an arbitrary number of parties. Unfortunately, known protocols are very far from achieving this goal. Beaver's OT extension technique crucially relies on the fact that the number of OTs required by Yao's two-party protocol [46] is equal to the length of the *shorter* input. No similar protocols are known for $n \geq 3$ parties. To make things worse, the number of OT invocations in current protocols (e.g., [22, 21]) does not only depend on the length of the inputs but also on the complexity of the function f being computed. Specifically, these protocols require *each pair of parties* to invoke an OT protocol for *each gate* of a circuit computing f.[2] In the computational setting it is possible to apply Beaver's OT extension protocol between each pair of parties, requiring only k OTs for each of the $\binom{n}{2}$ pairs. Thus, the state of the art prior to the current work can be summarized as follows:

[1] In contrast, it is not known how to implement even a single OT using a one-way function alone, and the possibility of a black-box construction of this type was ruled out by Impagliazzo and Rudich [30].

[2] Some MPC protocols do not rely on OT but rather on other "public-key" primitives such as threshold homomorphic encryption [19]; however, in these protocols too the number of public-key operations grows linearly with the circuit size of f.

– In the information-theoretic setting, the number of OTs needed by n parties to compute a circuit of size s is $O(n^2 s)$.
– In the computational setting (assuming one-way functions exist) the total number of OTs is $O(n^2 k)$.

The above state of affairs leaves much to be desired and gives rise to several natural questions: Can one reduce the quadratic dependence on the number of parties while maintaining security against a dishonest majority? Can the dependence on the circuit size in the information-theoretic case and the dependence on the security parameter in the computational case be eliminated?

1.1 Our Contribution

We answer the above questions affirmatively, obtaining several upper and lower bounds on the OT complexity of both information-theoretic and computationally secure MPC with no honest majority. Before describing our results, we outline (and justify) some essential details of our model.

Model. We consider a network of n parties that are connected via a synchronous network of secure point-to-point channels (secure channels are necessary in the information-theoretic setting, and can be cheaply implemented in a computational setting via the use of a hybrid encryption). The parties wish to compute a function f, which by default is a polynomial-time computable function taking one input bit from each party and returning an output of an arbitrary length (our results can be generalized to the case where each party has an ℓ-bit input – see below). Our goal is to design an OT-efficient protocol which securely computes f in the presence of a *semi-honest* (aka "honest-but-curious") adversary which may corrupt at most t parties, where the security threshold t satisfies $n/2 < t < n$. In the computational setting, restricting the attention to security in the semi-honest model is justified by the fact that it is possible to use one-way functions (and no additional OTs) for upgrading security to the malicious model [23, 21]. Finally, we allow each pair of parties to invoke an ideal OT oracle during the execution of the protocol and count the number of invocations of this oracle. (This model is also referred to as the "OT-hybrid" model.) Using a suitable composition theorem [10, 21], each call to the OT oracle can be substituted with an actual secure OT protocol. It is important to stress that our basic OT primitive is *string* OT; that is, the sender's strings are of arbitrary length (yet this length counts towards the communication complexity of our protocols). This is justified by the fact that OT of long strings can be easily reduced to a single invocation of OT of short keys of length k by using symmetric encryption and no public-key operations. Furthermore, most efficient implementations of OT (cf. [38]) directly realize OT of k-bit strings rather than bits.

In the above model, we obtain the following main results.

Number of OT channels. We start by examining the required number of "OT channels" between pairs of parties, that is, the number of distinct pairs that should jointly invoke the OT primitive. We show that if the adversary can corrupt up to $t = (1-\epsilon)n$ parties, where $\epsilon > 0$ is an arbitrarily small constant, then a total of $O(n)$ OT channels between pairs of parties are sufficient and necessary for general MPC. This is a quadratic improvement over previous protocols, which require OTs between each pair of parties. The $O(n)$ upper bound relies on a technique of Bracha [9] for distributing computations among several committees, a technique for combining oblivious transfers [27], and explicit constructions of dispersers [25, 42, 26]. Using OT extension protocols [4, 31], the $O(n)$ bound implies that $O(nk)$ invocations of OT are sufficient for computing arbitrary functions with computational security[3] when $t = (1-\epsilon)n$. The lower bound (in a more general form) relies on results from extremal graph theory. We note that the $\Omega(n)$ lower bound holds also if the OT channels are chosen *dynamically*, namely the identity of the pairs of parties which can invoke the OT oracle can be chosen during the execution of the protocol.

Coping with a bigger security threshold. The above results do not improve over the previous state of the art in the important case where $t = n-1$, when the number of parties is small, or in the information-theoretic setting. For these cases, we show that an arbitrary function $f : \{0,1\}^n \to \{0,1\}^*$ can be securely computed by a protocol which makes use of a single OT between each pair of parties. We also show that this result is tight, in the sense that when $t = n-1$ at least one OT between each pair of parties is necessary. At a high level, the protocol proceeds by $n-1$ iterations, where in the end of the i-th iteration the first $i+1$ parties hold additive shares of the truth table of $f(x_1, \ldots, x_i, \cdot, \ldots, \cdot)$, namely f restricted by the inputs of the first i parties. A major disadvantage of this protocol is that its communication complexity grows exponentially with n. We present natural classes of functions f for which this exponential overhead can be avoided. These include sparse polynomials, decision trees, deterministic and nondeterministic finite automata, and CNF and DNF formulas, which capture useful secure computation tasks (cf. [2]). Some of these efficient protocols rely on expander-based constructions of extractors for bit-fixing sources [32].

In the case where each party holds an ℓ-bit input (rather than a 1-bit input) the above upper and lower bounds on the number of OTs grow by a factor of ℓ, whereas the bounds on the number of OT channels remain unchanged.

2 Preliminaries

Throughout the paper we use the following notation: By $P_1, \ldots, P_n$ we denote the n parties, the security threshold t is the maximal number of parties that the adversary can corrupt, and k stands for a security parameter when considering computational or statistical security. When an n-party function f has a single output, we assume by default that the output is given to the first party P_1.

[3] This protocol inherits the security and assumptions of the underlying OT extension protocol. In particular, the protocol of [4] can be based on one-way functions but is only proved to be secure against *non-adaptive* adversaries, whereas the protocol of [31] in the random oracle model can be shown to be adaptively secure.

Our model for secure multiparty computation follows standard definitions from the literature [10, 21]. The availability of an OT primitive is captured by considering an *OT-hybrid* model, in which each pair of parties can invoke an ideal OT oracle. By a *t-secure protocol* for f we refer by default to a protocol which is *perfectly* secure in the *semi-honest* model against an adversary that may *adaptively* corrupt at most t parties. Perfect security will sometimes be relaxed to statistical or computational security.

3 Counting OT Channels: Upper and Lower Bounds

A closely related question to the number of required OT calls is the number of required OT channels in a network. That is, given a network of n parties, we look at a graph where each node stands for a party and an edge stands for the ability to run an OT between two parties. On each such edge, we assume the ability to execute arbitrarily many OT calls. In addition, there exist private communication channels between *every* pair of parties. The question is how many OT channels are needed in order to simulate a full network of OTs (a network in which every two parties can execute an OT functionality).

More precisely, define the n party OT function f_{OT} as a function that takes inputs from two parties (if more than two parties provide inputs then the function outputs an abort symbol). The first party inputs two string s_0, s_1 and the second inputs a bit c. The output s_c is received by the second party. The question at hand is how many OT channels are required for the network to be able to securely compute the function f_{OT}.

OT Channels: Static vs. Dynamic. When discussing OT channels, special care needs to be taken when modelling the network. The simpler case is when the network is *static*, i.e., the OT channels are set in advance and known to the adversary (this case is suitable for an implementation of OTs based on some physical infrastructure). A stronger model (for the honest parties) allows a *dynamic* network, in which the parties may set the OT channels as part of the protocol (while trying to hide this information from the adversary). Our upper bounds do not take advantage of the dynamic setting and work also in the static setting. We prove our lower bounds initially in the static case and then extend them to the dynamic case.

OT Channels and Counting OTs. Counting OT channels is an interesting question in its own right, and may directly capture the case where OTs are implemented via some physical infrastructure (e.g., noisy point-to-point channels). Moreover, its connection to the number of OT calls needed for secure computation is two fold:

- As means of achieving **upper bounds** on the number of OT calls in a *computational* setting. In the *two-party* computational setting, it is known how to achieve a polynomial number of OT calls at the price of just k calls [4, 31] (where k is the security parameter). Therefore, if we only need, say, $O(n)$ OT channels, then we can simulate the whole network at the price of

$O(nk)$ OT calls, which is better than the trivial upper bound of $\binom{n}{2}k$ OT calls.
- As a mechanism for proving **lower bounds** on the number of OT calls (see Theorem 4). Namely, the minimal number of channels needed to securely compute the functionality f_{OT} is in particular a lower bound on the number of OT calls necessary.

We note that the function f_{OT} is just a single example of a function for which the lower bounds hold; a similar lower bound holds for any function that is *complete* for n-party computation, in the sense that it can be used as an oracle for computing arbitrary functions t-securely. A sufficient condition for completeness is that for every pair of inputs, one of the two-party criteria from [34, 6, 28] is met for some restriction of the other inputs.

3.1 Upper bounds for $t = (1 - \delta)n$: The Committees Method

We turn to the case that $t = (1-\delta)n$ (we mostly think of δ as a constant fraction, but the discussion is not restricted to this case). Consider the following strategy: from the n parties choose m committees, each of size d, where a party can (and will) participate in several different committees. Assume that each committee has a full network of OT channels between them.

Using each committee we generate a candidate for an OT protocol between party A and party B as follows: The sender and receiver additively share their inputs (s_0, s_1 and c respectively) between all committee members. The committee members now run a secure computation protocol that computes a random additive sharing of s_c between the committee members (this is done using their OT channels and the "GMW protocol" [22]). Now each committee member sends his share of s_c to the receiver B who reconstructs the output. This constitutes a secure OT protocol as long as not all of the committee has been corrupted (if all of the committee is corrupted then there is no security at all). In total, for each of the m committees we have a candidate for an OT protocol, which is secure if not all of the underlying committee is corrupted. It is known how one can combine OT candidates protocols to 1-secure OT protocol as long as a majority of the candidates are secure. This method is called an $(\lceil \frac{m+1}{2} \rceil, m)$-*robust combiner* for OT and its existence was pointed out in [27] (and [37]) based on amplification techniques from [14, 45].

Our goal is therefore to solve the following combinatorial problem: find a collection of "small" committees such that every adversary, corrupting at most $t = (1 - \delta)n$ of the parties, covers less than half of the committees. A simple probabilistic argument shows that such collections exist and moreover a random collection whose size depends only on δ (and not on n) is a good solution with high probability.

An Explicit Construction. We next give an explicit choice of committees that satisfies the above requirements. Consider a bipartite graph with m vertices on the left (the committees) and n vertices on the right (the parties). Every committee has d edges connecting it to all the parties it consists of (that is, the graph is d-regular on its left side). The requirement for the committees protocol to be secure is that every set of $m/2$ vertices on the left are connected to more than $(1-\delta)n$ vertices on the right. This is exactly the setting of a *disperser*[4] with very high min-entropy (min-entropy of $\log(n/2)$ out of the possible $\log n$). There are several explicit constructions that we can use for this task including Goldreich and Wigderson [25], Reingold et al. [42] and Gradwohl et al. [26], all of which have near optimal degree (up to constants with respect to the lower bounds of [41]). Specifically we can work with $d = \lceil \frac{1}{\delta} \rceil$, and $m = n + o(n)$ (or even $m = n - o(n)$ if using the construction of [26]).

Corollary 1 *There exists an explicit construction of a network consisting of* $(n+o(n))\binom{\lceil 1/\delta \rceil}{2}$ *OT channels such that the network can t-securely compute f_{OT} in the presence of an adversary that corrupts up to $t = (1-\delta)n$ of the parties. Specifically:*

– *If δ is a constant then the network needs $O(n)$ OT channels.*
– *As long as $\delta \geq \frac{1}{\sqrt{n}}$, the construction requires strictly less than the $\binom{n}{2}$ OT channels of the full network.*

The above corollary can be combined with OT extension protocols [4, 31] to yield a $((1-\delta)n)$-secure protocol for an *arbitrary* function f that uses a total of $O(kn)$ OTs. This protocol inherits the security and assumptions of the underlying OT extension protocol (see Footnote 3).

Related works using committees. The idea of virtually performing tasks by committees has been used in distributed computing and cryptography. Originating in the work of Bracha [9] in the context of Byzantine agreement, committees have been used in the same context by [8, 12], for MPC [29] and for leader election [39, 48, 35]. Committees have recently been used by Fitzi et al. [18] to achieve Perfectly Secure Message Transmission (PSMT) in a partial network of secure channels. It should be noted that while the task of PSMT is reminiscent of our question regarding OT channels, there are inherent differences. For example, our committees protocol (above) can effectively achieve an OT call even between two parties that are isolated in the OT graph (not connected by an OT channel to any other party). In PSMT, on the other hand, there is no chance of achieving secure communication with a node that is not connected by any secure channel.[5]

On non-adaptive adversaries. In the case that the adversary is non-adaptive but the network is dynamic, one can do much better. In fact, only a single good committee is needed. Indeed, a randomly chosen committee of size k/δ has probability of $1 - 2^{-\Omega(k)}$ of being good. Note, however, that this simple protocol can be trivially broken by an adaptive adversary who first learns the identity of the committee members and then corrupts all of them.

[4] A definition and discussion on dispersers can be found in, e.g. [43].
[5] Recall that in our OT channels model we assume a *full* network of secure channels to be intact.

3.2 Lower Bounds for $t = n - 1$: Full OT Network is Necessary

In this section we look at the strictest security scenario, where the adversary can corrupt all but one of the parties. We show that given an almost full network of OT channels except for one missing channel, it is impossible to complete the network (i.e., securely compute the function f_{OT}). As a first step we consider a static network with just 3 parties, A,B and C.

Claim 2 *Let A, B and C form a network where C has OT channels with both A and B, but there is no OT channel between A and B. Then, any 2-secure protocol for f_{OT} over this partial network can be used (as a black box) to obtain a two-party OT protocol in the plain model.*

Proof: We transform the given 3-party protocol π_3 into a two-party OT protocol π_2 in two steps: first we eliminate all invocations of the OT oracle, and then we obtain a two-party protocol by letting one of the parties simulate A and the other simulate B and C. These steps are captured by the following two lemmas.

Lemma 1. *Any 3-party protocol π_3 as in Claim 2 can be used (as a black box) to obtain a protocol π'_3 over a network with no OT channels, such that π'_3 is secure against an adversary that corrupts either $\{A, C\}$ or $\{B, C\}$.*

The protocol π'_3 is obtained from π_3 by implementing each OT call via the trivial protocol in which C sends its input to the other OT participant (either A or B). Note that this trivial OT protocol is perfectly secure against an adversary that corrupts either $\{A, C\}$ or $\{B, C\}$, since the input of C is guaranteed to be known to the adversary.

We can now use π'_3 to implement OT between A and B in a 3-party network without OT channels. An important observation is that C has no inputs in this protocol.

Lemma 2. *Let Π be a protocol between A, B, C that computes a function for which C has no inputs, and suppose that Π is secure against $\{A, C\}$ and $\{B, C\}$. Then Π is also secure against $\{A\}$ and $\{B\}$.*

The lemma follows simply by observing that when C has no input the view of a corrupted A can be simulated using the simulation of $\{A, C\}$ and likewise for B.

We can now use π'_3 to get a two-party OT protocol π_2 by letting one party simulate A and the other party simulate B, C. Due to Lemma 2 we get that the protocol is secure against corruption of either party and hence constitutes an OT protocol in the plain model between two parties. ∎

We Generalize Claim 2 to hold for a *dynamic* network of n parties (rather than a static 3-party network). The proof appears in the full version.

Claim 3 *Any $(n-1)$-secure protocol for f_{OT} over an n-party partial network with at most $\binom{n}{2} - 1$ OT channels can be used (as a black box) to obtain a two-party OT protocol in the plain model.*

As corollaries of the previous lemma, we get the lower bounds that we were seeking for the number of OT invocations:

Theorem 4. *Any n-party protocol Π that $(n-1)$-securely computes f_{OT} using less than $\binom{n}{2}$ OT calls can be used (as a black box) to implement a two-party OT protocol in the plain model. In particular, there is no such Π with perfect or statistical security, and its existence with computational security cannot be based on one-way functions in black-box way.*

3.3 Lower Bounds for Corruption of $t = n - d$ Parties

We show impossibility results for this case that are based on extremal graph theory and give tight bounds (for different ranges of d). The bounds hold also in the dynamic network model.

Theorem 5 (Lower bound for general d). *Consider a network of n parties in the presence of an adversary that can corrupt $t = n - d$ parties.*

1. *Suppose the network (even a dynamic one) has $o(n^2/d)$ OT channels. Then any $(n-d)$-secure protocol for f_{OT} in this network be used (as a black box) to implement a two-party OT protocol in the plain model.*
2. *Suppose d is a constant and the network (even a dynamic one) has less than $(1-c))\binom{n}{2}$ OT channels (for some constant c). Then any $(n-d)$-secure protocol for f_{OT} in this network be used (as a black box) to implement a two-party OT protocol in the plain model.*

Proof: The two claims follow the same principle. The crux is that unless every two sets of parties of size d be connected in the OT graph, then one can build an OT in the plain model. Suppose that there exist two disconnected sets of size at least d then define each of the groups as A and B and the rest of the graph as C. We now reduce this setting to the case of the previous section where we have parties A, B and C such that A and B are not connected and the adversary may corrupt either $\{A, C\}$ or $\{B, C\}$ (since these are sets of size at most $n-d$). By Claim 2, such a setting would allow building a two-party OT protocol in the plain model.

Proving (1) for the static case. Due to the outline above, we examine graphs where every two sets of size d must contain at least one edge between them. This means, in particular, that in the graph of *non-edges* there exists no clique of size $2d$. By Turan's Theorem, such a graph can have at most $(1-1/(2d))n^2/2$ non-edges, and hence the OT graph must have at least $\frac{n^2}{4d}$ edges.

Proving (2). The above argument is limited since it only considers the fact that the non-edges graph must not contain a clique of size $2d$. But actually, the graph cannot even contain a bipartite $d \times d$ clique which is a stricter constraint. For such a graph there are rather tight results when d is a constant. Namely, for constant d the Erdös-Stone-Simonovits Theorem [16] states that the non-edges graph must have at most $o(n^2)$ missing edges, which amounts to the OT graph containing $(1 - o(1))\binom{n}{2}$ OT channels. As in Claim 3, even if the layout of OT channels is not known in advance it can simply be guessed. Since the number of missing edges is constant, the probability that a random guess is correct about the eventual network is inverse polynomial, which suffices to extend any static network result (with constant d) to a similar result for dynamic networks.

Sketch of proof of (1) in the dynamic case. We provide an argument that there must be at least $\frac{n^2}{2d}$ edges in the graph (when assuming that every two sets of size d must contain at least one edge between them). This argument provides a slightly better bound than Turan's Theorem (since Turan's theorem discusses anti-cliques rather than bipartite $d \times d$ anti-cliques) but, more crucially, gives us information that is useful for proving the bound in the dynamic case. Specifically, we get a guarantee that if there are less than $\frac{n^2}{2d}$ edges in the graph then in every partition of the graph to d sized sets, at least one of the sets has no neighbor with some other d-sized set (the other set is not necessarily in the partition). This information, gives rise to an efficient procedure that finds (with noticeable probability) sets A, B and C where A and B are of size d and have no connecting edges in the dynamically set network. This, in turn, allows to build a two-party OT in the plain model from a secure protocol for f_{OT} in the dynamic setting. The complete proof appears in the full version. ∎

4 Upper Bounds for the case of $t = n - 1$

4.1 The Tables Method

The tables method is a generic secure computation protocol that computes a function by an iterative process on the truth table of the function. The truth table of a function $f : \{0,1\}^n \to \{0,1\}^m$ is simply a $2^n m$ bit long string such that the i^{th} cell (or entry) contains the value $f(x)$ where x is the string representing the integer i. Denote the bits of the string x by $x_1, \ldots, x_n$. The idea is to use the fact that restricting the value of the variable x_1 to be either 0 or 1 amounts to looking either at the first or at the second half of the table. Denote by T^f the truth table of f and by $T^f|_b$ the new table when fixing the first input variable x_1 to b, for $b \in \{0,1\}$. Thus, $T^f|_0$ is simply the first half of the table T^f while the second half is $T^f|_1$.[6] Similarly, denote the table of f after fixing the j most significant bits of x as $T^f|_{x_1,\ldots,x_j}$.

The idea of the protocol is that at the j^{th} iteration the j parties $P_1, \ldots, P_j$ jointly distribute additive shares of the table $T^f|_{x_1,\ldots,x_j}$ between themselves. At the end of the protocol all parties hold a share of the table $T^f|_{x_1,\ldots,x_n}$ which consists simply of the single value $f(x)$. The full protocol TABLES is presented in Figure 1.

[6] The first and second halves of a table correspond to fixing of the "most significant" bit, x_1. For every other x_i, the fixing of the i^{th} bit x_i amounts to a different partition of the table into two halves.

TABLES $f(x_1, \ldots, x_n)$
Let $f : \{0,1\}^n \to \{0,1\}^m$ be a function to be computed by n parties $P_1, \ldots, P_n$, each holding a single input bit $x_1, \ldots, x_n$ respectively.

- **Initialization stage:** P_1 computes $T^f|_{x_1}$, i.e. the truth table of f when restricted to his input x_1. Let $S_1^1 = T^f|_{x_1}$.
- **Iteration stage:** The following steps are repeated sequentially for each $j \in [n-1]$. At the beginning of the j^{th} iteration, each of the parties $P_1, \ldots, P_j$ holds a share $S_1^j, \ldots, S_j^j$ (respectively) such that $\bigoplus_{i\in[j]} S_i^j = T^f|_{x_1,\ldots,x_j}$. At the end of the iteration, the table $T^f|_{x_1,\ldots,x_{j+1}}$ is shared among the parties $P_1, \ldots, P_{j+1}$.
  1. For each $i \in [j]$, party P_i chooses a random mask $R_i \in \{0,1\}^{2^{n-j}m}$ and calculates $T_i^0 = S_i^j|_0 \oplus R_i$ and $T_i^1 = S_i^j|_1 \oplus R_i$.
  2. P_{j+1} runs an OT protocol with every P_i such that $i \in [j]$. They run the protocol $OT(T_i^0, T_i^1; x_{j+1})$ with P_i as sender and P_{j+1} as receiver.
  3. For each $i \in [j]$, party P^i sets $S_i^{j+1} = R_i$ while party P_{j+1} sets $S_{j+1}^{j+1} = \bigoplus_{i\in[j]} T_i^{x_{j+1}}$.
- **Output stage:** Each party sends its share S_i^n to P_1 who outputs $\bigoplus_{i\in[n]} S_i^n$.

Fig. 1. The TABLES protocol.

Theorem 6. *Protocol TABLES is an $(n-1)$-secure protocol for the function f. The protocol involves a single OT call between each pair of players.*

A proof of Theorem 6 appears in the full version. Note that the protocol can be easily generalized to handle ℓ-bit inputs rather than single bits. In such a case, each party runs ℓ consecutive iterations, one for each input bit. The number of OTs between each pair of players grows to ℓ.

4.2 Applying the Tables Method

The advantage of the tables method is that it requires exactly one OT call between each pair of parties (overall, $\binom{n}{2}$ OT calls) and presents a plausibility result for $(n-1)$-secure computation of any function on n bits, matching the lower bound on the number of OTs for the case of $t = n-1$ (Theorem 4).

The main problem with the tables method, however, is that the strings sent in the (string) OT are of length $2^n m$. This makes the protocol inefficient except when the input domain of f is of feasible size. In the following we show that for certain classes of functions one can get efficient protocols that still require a minimal number of OTs. For example, we describe how to securely compute any function in NC^0, namely a function in which each bit of the output depends on a constant number of input bits. Note that, under standard cryptographic assumptions, there exist non-trivial cryptographic primitives such as one-way functions and pseudorandom generators in NC^0 [1].

Proposition 7 *For every function $f \in \mathrm{NC}^0$ there exists an* efficient *$(n-1)$-secure computation protocol using just $\binom{n}{2}$ OT calls.*

Proof sketch: For a function $f : \{0,1\}^n \to \{0,1\}^m$ in NC^0 it is guaranteed that each output bit is a function of $c = O(1)$ input bits. We call these the c input bits that *affect* the output bit.

The straightforward protocol runs m separate TABLES protocols, one for each output bit. Since each output bit is affected by only c parties, then each TABLES protocol can be executed only by the c parties that affect this output, using a table of size 2^c which is constant. However, if each protocol is run separately then the number of OT calls would grow to $\binom{c}{2}m$, which may be bigger than $\binom{n}{2}$ when m is large. Using a careful scheduling of the TABLES protocols, all OT calls between each pair of parties can be computed using a *single* OT invocation. (See full version for details.) Thus, the overall number of OTs remains $\binom{n}{2}$, matching the lower bound. ∎

Note that the above mentioned schedule works for every function f (not necessarily in NC^0). The efficiency though is only guaranteed for limited types of functions. More precisely, efficiency is guaranteed for every function where each output bit is affected only by a *logarithmic* number of input variables.

Extension to bounded degree polynomials. A straightforward extension of the above proposition follows from observing that if the output stage is not executed then the above protocol efficiently computes an additive secret-sharing for each output bit. At the same cost, the parties can get an additive secret-sharing of the *sum* of various outputs. This is done simply by each party taking a local sum of the various shares that it holds to create a new share for the sum. This forms an efficient low communication $(n-1)$-secure protocol for all *logarithmic degree polynomials* whose representation as the sum of monomials has only a *polynomially* many terms.

4.3 Oblivious Linear Branching Programs

This section puts forward a generalization of the tables method that extends the class of functions that we can securely compute by an efficient protocol. The class of functions that we deal with is a linear version of oblivious branching programs.

Definition 8 (Oblivious Linear Branching Programs) *A linear branching program LBP on an n-bit input is an ordered set of triples $\{(i_1; M_1^0, M_1^1), \ldots, (i_s; M_s^0, M_s^1)\}$ and an initial vector $v_0 \in \{0,1\}^{w_0}$. Each triple contains an index $i_j \in [n]$ and a pair of boolean $w_{j-1} \times w_j$ matrices M_j^0, M_j^1 (where $w_j \geq 1$). The* size *of the program is s and its* width *w is the maximal w_j over all $j \in [s]$. On input $x \in \{0,1\}^n$ the output of the program is $LBP(x) = v_0 M_1^{x_{i_1}} \cdots M_s^{x_{i_s}}$.*

Theorem 9. *There exists an $(n-1)$-secure computation protocol for computing the output of a linear branching program LBP. The protocol makes at most sn OT calls on ℓ-bit strings (where s and ℓ are the size and width of LBP, respectively).*

The protocol runs along the lines of the TABLES protocol (Figure 1), with the main difference being that at step (1), the i^{th} party computes $T_i^0 = S_i^j M_j^0 \oplus R_i$ and $T_i^1 = S_i^j M_j^1 \oplus R_i$ (rather than $T_i^0 = S_i^j|_0 \oplus R_i$ and $T_i^1 = S_i^j|_1 \oplus R_i$). The complete protocol appears in the full version.

4.4 Functions Captured by Linear Branching Programs

As mentioned before, the protocol for linear branching programs is a generalization of the tables method. As such, it captures the same applications as the previous method, but it also captures other functions that could not be efficiently computed in the previous method. We highlight some function classes that can be computed using this methodology:

Tables. The LBP model is indeed a generalization as exemplified by the following presentation: consider the initial vector v_0 that is the truth table of the function f. For each iteration, the two matrices M_j^0 and M_j^1 are simply two projection matrices (and hence also linear operations). M_j^0 leaves only the first half of the coordinates while M_j^1 leaves the second half of the coordinates.

Oblivious branching programs. Similar to linear branching programs, oblivious branching programs inquire a single variable at each layer and move to a new state according to its answer. This can be viewed as a layered graph where each node has two outgoing edges labeled 0 and 1 going to the next level. The width of the program is the maximal number of nodes in a layer and the number of layers is the length of the program. The simulation of such a branching program by an LBP looks at the intermediate states as indicator vectors of length w (all zeros except a single one indicating the current state). The matrix for input 0 has as its i^{th} row, the indicator vector that the i^{th} state should move to in case that the input bit is 0 and likewise for the second matrix. Other models of computation or functions that are captured by their view as a branching program include **decision trees**, **oblivious automata** and *membership in small (polynomial size) set*.

Oblivious counting branching programs. As in the oblivious branching program case, a non-deterministic branching program allows going from one state to a number of states. A *counting branching program* outputs the number of accepting paths that a non-deterministic branching program has. Such non-determinism can easily be incorporated into LBPs by allowing the state vector to vary from an indicator with a single one. The i^{th} row of the matrix will have a 1 for each possible move from the i^{th} state to the next level. If the operations are executed over a large enough field, then the intermediate vector holds in each location the number of paths that lead up to this state. Thus, over a large field this implements a counting branching program. If working over the field $GF(2)$ then this is simply a *parity branching program* that indicates the parity of the number of paths that lead to a state. Unfortunately, the most natural non-deterministic model is not captured by LBPs. This is a non-deterministic procedure that asks whether their exists an accepting path to the program at all (an operation that is no longer a linear one). In Section 5, we present protocols for secure computation for this model.

Sparse Polynomials. LBPs allow for a simple and efficient computation of a monomial over input bits.[7] In addition, an LBP can incorporate in it a number of parallel LBP computations and have the last operation sum their outputs (simply by incorporating this linear operation in the last pair of matrices). Thus LBPs can compute a polynomial as a sum of all of its monomials. For the program to be efficient, the only limitation is that the number of monomials is polynomial. Note that this captures a larger family of functions than in Section 4.2. A closely related question is can one compute a DNF formula using LBPs (DNFs are the OR of monomials rather than their sum). This is a special case of the non-deterministic question addressed in the next section.

5 Secure Computation for Non-Deterministic LBP

In this section we suggest a method of securely computing a nondeterministic (or existential) linear branching program. As opposed to counting branching programs that give the sum of the number of solutions (and are easy to compute by LBPs), asking whether or not the exists a solution is a non-linear operation and therefore is not captured by the general framework. A good example is the computation of DNF formulas. Like sparse polynomials these are a polynomial size collection of monomials over n input bits but the question is whether x satisfies at least one of the monomials (rather than their sum). The problem arises from the fact that the OR operation is not a linear one and hence it is not captured by the LBP model. A natural approach is to first compute the sum of the monomials over a large enough field (to avoid a wraparound), and then check whether this sum is zero or not. However, revealing the sum is not a good solution as it leaks more information on the inputs than the desired output (it differentiates, for instance, whether there was a single satisfied monomial or many of them).

We propose a generic method that securely computes the existential analogue of any counting LBP. The method is secure against adversaries that can corrupt up to $t \leq n - \Omega(k)$ where k is the security parameter, and adds a statistical error of at most 2^{-k}. For simplicity we state and prove the theorem formally for limited LBPs where each party has a single input bit and note that as in the previous sections, this protocol may be generalized to more complex LBPs, at the price of additional OT invocations.

Theorem 10. *Let L be a LBP of length n computing a function $f : \{0,1\}^n \rightarrow \mathbb{Z}_p$ where p is a $(k/2)$-bit prime and k is the security parameter. Then there exists an efficient n-party statistically t-secure protocol with $t = n - O(k)$ for the predicate g defined as:*

$$g(x) = \begin{cases} 0, & f(x) = 0; \\ 1, & \text{otherwise.} \end{cases}$$

[7] For example, an LBP for the AND function over n bits simply uses vectors and matrices of dimension 1 and takes $v_0 = (0)$ and the i^{th} triplet is $(i, 0, 1)$.,

The protocol requires $4\binom{n}{2}$ OT calls.

Proof: The basic idea is to add a randomization stage in each of the iterations of the secure computation protocol for L. This randomization should give an output with the following properties: The output should be uniformly distributed over the domain if $f(x) \neq 0$ but should always be 0 if $f(x) = 0$. Therefore, if the output is not 0 then we know for sure that $f(x) \neq 0$ but learn nothing else about $f(x)$. If the output is 0, then it is most likely that $f(x) = 0$. An error only happens if the uniformly distributed output happened to hit 0 which happens with probability that is inverse of the domain size (we will choose the domain to be of size 2^k).

Cayley expanders and a matrix representation. For the randomization steps we use a constant degree *Cayley expander graph* with a specific structure. A Cayley graph is described over a multiplicative group by a small set of generators $\{G_1, \ldots, G_d\}$. For each element (vertex) v in the group, its neighbors are $\{G_1 \cdot v, \ldots, G_d \cdot v\}$. We can use any expander graph with a constant degree such that its generators can be represented as *affine transformation* over $\mathbb{Z}_p^m$. In particular we can use the expander graph of Margulis [36] and Gaber and Galil [20]. This is an expander over $\mathbb{Z}_N \times \mathbb{Z}_N$ for some integer N, and we take N to be a prime p in the order of $2^{k/2}$. The expander has degree 8 and as we required can be presented by 8 affine transformations.

For simplicity we will describe the construction over $\mathbb{Z}_p^2$ (as in the Margulis graph) although this can be generalized. Suppose that each step is an affine transformation, e.g. a step moves from vertex $v \in \mathbb{Z}_p^2$ to the vector $Av + e$ where $A \in [\mathbb{Z}]_{2,2}$ is a 2×2 matrix and $e \in \mathbb{Z}_p^2$ is a vector. For each such generator we define the corresponding 3×3 matrix $G \in [\mathbb{Z}]_{3,3}$ as:

$$G = \begin{pmatrix} a_{11} & a_{12} & e_1 \\ a_{21} & a_{22} & e_2 \\ 0 & 0 & 1 \end{pmatrix}$$

Notice that multiplying the vector $v = (v_1, v_2, 1)$ by G simply amounts to a step in the expander from vertex (v_1, v_2) with the third coordinate remaining 1. If $A_1, \ldots, A_n$ denotes a series of steps where each $A_i \in \{G_1, \ldots, G_d\}$ and let $v = (0, 0, 1)$ then $A_n A_{n-1} \ldots A_1 v$ stands for a random walk starting at vertex $(0, 0)$ and the first 2 coordinates of the output hold the end vertex of the walk. On the other hand, $A_n A_{n-1} \ldots A_1 \overline{0}$ simply equals $\overline{0}$ (where $\overline{0}$ stands for the vector $(0, 0, 0)$).

The randomization technique. Basically, each party in its turn will contribute a random step A_i in the expander. Our goal is that at the end of the execution the output will be the multiplication $A_n \ldots A_1 v$ where v is the vector $(0, 0, f(x))$. Hence, if $f(x) = 0$ the output will simply be $\overline{0}$. On the other hand if $f(x) \neq 0$ then the output represents the end of a random walk starting at $(0, 0)$. We use the following result of Kamp and Zuckerman [32], which states that an adversary that does not know $\Omega(k)$ of the n expander steps has essentially no knowledge about the outcome of the random walk. The precise statement is that a random walk on a good expander (where each step is represented by a single symbol) is an extractor for a *symbol fixing source*.[8]

Theorem 11 (adapted from Kamp and Zuckerman [32], Theorem 3.1). *Let $a_1, \ldots, a_n$ be a series of steps on an expander of degree d, size d^m and second eigenvalue $\lambda \leq d^{-\alpha}$ and let t be such that $n - t \geq \frac{1}{2\alpha}\left(m + \frac{2}{\log d}\log\frac{1}{2\varepsilon}\right)$. Then conditioned on the view of an adversary that observes at most t elements in the series, the output of the walk is ε-close to uniform.*

In our application the graph has parameters $d = 8$, $\alpha \approx 0.06$ (due to [20]) and the graph is of size 2^k, thus $m = k/3$. When choosing $\varepsilon = 2^{-k}$ the requirement in the Theorem translates to $n - t \geq \Omega(k)$.

Corollary 12 *Let $A_1, \ldots, A_n$ be a sequence of randomly chosen generator matrices for a good expander graph (e.g. the Margulis graph) with vertex set $\mathbb{Z}_p^2$ (for prime p in the order of $2^k/2$) . Let $v = (0, 0, c)$ for $c \neq 0$ and denote $u = A_n \ldots A_1 v$. Then conditioned on the view of an adversary that observes at most $t = n - \Omega(k)$ of the sequence the pair (u_1, u_2) is 2^{-k}-close to the uniform distribution on $\mathbb{Z}_p^2$.*

Proof: (of Corollary 12) The corollary follows directly from Theorem 11 in the case that $c = 1$. It is left to show that it also holds for any $c \neq 0$. This can be seen by breaking the vector v into the sum of c vectors of the type $(0, 0, 1)$. For each of the c vectors the random walk gives an almost uniform distribution. When summing up we have a uniform distribution multiplied by c. Since we are working in $\mathbb{Z}_p$ then multiplication amounts to a permutation on the elements of $\mathbb{Z}_p$ and the output remains close to uniform. Note that this is the only place where we require that p is prime. ▮

The actual protocol. The protocol is the same protocol as the general one for computing LBPs only at each iteration, the acting party (that redistributes the shares) chooses a random step in the extractor. The matrix operations of the LBP will always be multiplied from the right, while the random steps of the expander will be multiplied from the left. Technically, the following changes are applied:

- Instead of starting with the vector v_0 of length w_0, the protocol starts with a matrix B_0 of 3 vectors ($B_0 \in [\mathbb{Z}_p]_{3,w_0}$). The first two rows of B_0 are all zero vectors and the third is the vector v_0. Accordingly, the protocol runs throughout with 3 row matrices rather than single row vectors.
- At step (1) of the iteration stage, rather than computing two values, party P_i computes $2d$ values, two for each generator of the expander. They are for each $\tau \in [d]$: $T_i^{0\tau} = G_\tau S_i^j M_j^0 \oplus R_i$ and $T_i^{1\tau} = G_\tau S_i^j M_j^1 \oplus R_i$.
- At step (2) of the iteration, the acting party P_i chooses a random $\tau \in [d]$ and runs a $\binom{2d}{1}$-OT protocol with each party according to his input x_j and τ. Such a protocol requires $\log d + 1$ OT calls.

[8] A symbol fixing source is a randomness source for which t of n symbols are fixed while the rest are uniformly distributed.

– At the output step, all parties send the *first two* rows of their shares (but not the third row!) to the designated party P_1. This party calculates the sum and outputs 0 if the sum was $(0,0)$ and 1 otherwise.

The correctness and security of the overall protocol follows since the modified LBP protocol forms a $n-1$-secure computation for a sharing of the following vector

$$u = A_n \ldots A_1 B_0 M_1^{x_{i_1}} \ldots M_n^{x_{i_n}}$$

In addition, $B_0 M_1^{x_{i_1}} \ldots M_n^{x_{i_n}}$ is simply the vector $(0,0,f(x))$. Therefore, combined with corollary 12, we get that if P_1 outputs the correct value (up to an error probability of at most $1/p^2 \leq 2^{-k}$). Moreover, if $f(x) \neq 0$ then an adversary that corrupts up to t parties (including P_1) sees a value that is statistically close to uniform, hence leaking no additional information on $f(x)$ or x. This concludes the proof of Theorem 10. ∎

Acknowledgements. We thank Ronen Shaltiel for pointers on constructions of dispersers.

References

1. B. Applebaum, Y. Ishai, and E. Kushilevitz. Cryptography in NC^0. In *45th FOCS*, pages 166–175, 2004.
2. O. Barkol and Y. Ishai. Secure computation of constant-depth circuits with applications to database search problems. In *CRYPTO*, pages 395–411, 2005.
3. D. Beaver. Precomputing oblivious transfer. In *CRYPTO*, pages 97–109, 1995.
4. D. Beaver. Correlated pseudorandomness and the complexity of private computations. In *28th STOC*, pages 479–488, 1996.
5. A. Beimel and T. Malkin. A quantitative approach to reductions in secure computation. In *TCC 04'*, pages 238–257, 2004.
6. A. Beimel, T. Malkin, and S. Micali. The all-or-nothing nature of two-party secure computation. In *CRYPTO '99, LNCS*, volume 1666, pages 80–97. Springer, 1999.
7. M. BenOr, S. Goldwasser, and A. Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *20th STOC*, pages 1–10, 1988.
8. P. Berman, J. Garay, and K. Perry. Bit optimal distributed consensus. In *Computer Science Research*, pages 313–32. Plenum Publishing Corporation, 1992.
9. G. Bracha. An $o(\log n)$ expected rounds randomized byzantine generals protocol. *Journal of the ACM*, 34(4):910–920, 1987.
10. R. Canetti. Security and composition of multiparty cryptographic protocols. *Journal of Cryptology*, 13(1):143–202, 2000.
11. D. Chaum, C. Crépeau, and I. Damgård. Multiparty unconditionally secure protocols. In *20th STOC*, pages 11–19, 1988.
12. B. Coan and J. Welch. Modular construction of a byzantine agreement protocol with optimal message bit complexity. *Information and Computation*, 97(1), 1992.
13. C. Crépeau and J. Kilian. Achieving oblivious transfer using weakened security assumptions. In *29th FOCS*, pages 42–52, 1988.

14. I. Damgård, J. Kilian, and L. Salvail. On the (im)possibility of basing oblivious transfer and bit commitment on weakened security assumptions. In *Eurocrypt '99, LNCS*, volume 1592, pages 56–73. Springer, 1999.
15. Y. Dodis and S. Micali. Lower bounds for oblivious transfer reductions. In *Eurocrypt '99, LNCS*, volume 1592, pages 42–55. Springer, 1999.
16. P. Erdos and M. Simonovits. A limit theorem in graph theory. *Stud. Sci. Math. Hung.*, 1:51–57, 1966.
17. S. Even, O. Goldreich, and A. Lempel. A randomized protocol for signing contracts. *Communications of the ACM*, 28(6):637–647, 1985.
18. M. Fitzi, M. Franklin, J. Garay, and H. Vardhan. Towards optimal and efficient perfectly secure message transmission. In *TCC 07'*, pages 311–322, 2007.
19. M. Franklin and S. Haber. Joint encryption and message-efficient secure computation. *J. Cryptology*, 9(4):217–232, 1996.
20. O. Gabber and Z. Galil. Explicit constructions of linear-sized superconcentrators. *JCSS*, 22(3):407–420, 1981.
21. O. Goldreich. **Foundations of Cryptography** - *Volume 2.* Cambridge University Press, 2004.
22. O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game - a completeness theorem for protocols with honest majority. In *19th STOC*, pages 218–229, 1987.
23. O. Goldreich, S. Micali, and A. Wigderson. Proofs that yield nothing but their validity, or all languages in NP have zero-knowledge proof systems. *Journal of the ACM*, 38(1):691–729, 1991.
24. O. Goldreich and R. Vainish. How to solve any protocol problem - an efficiency improvement. In *CRYPTO '87, LNCS*, volume 293, pages 73–86. Springer, 1987.
25. O. Goldreich and A. Wigderson. Tiny families of functions with random properties: A quality-size trade-off for hashing. *Rand. Structs. and Algs.*, 11(4):315–343, 1997.
26. R. Gradwohl, G. Kindler, O. Reingold, and A. Ta-Shma. On the error parameter of dispersers. In *APPROX-RANDOM*, pages 294–305, 2005.
27. D. Harnik, J. Kilian, M. Naor, O. Reingold, and A. Rosen. On tolerant combiners for oblivious transfer and other primitives. In *EUROCRYPT ' 2005, LNCS*, volume 3494, pages 96–113. Springer, 2005.
28. D. Harnik, M. Naor, O. Reingold, and A. Rosen. Completeness in two-party secure computation - a computational view. In *36th STOC*, pages 252–261, 2004.
29. M. Hirt and U. Maurer. Player simulation and general adversary structures in perfect multiparty computation. *Journal of Cryptology*, 13(1):31–60, 2000.
30. R. Impagliazzo and S. Rudich. Limits on the provable consequences of one-way permutations. In *21st STOC*, pages 44–61, 1989.
31. Y. Ishai, J. Kilian, K. Nissim, and E. Petrank. Extending oblivious transfers efficiently. In *CRYPTO '03, LNCS*, volume 2729, pages 145–161. Springer, 2003.
32. J. Kamp and D. Zuckerman. Deterministic extractors for bit-fixing sources and exposure-resilient cryptography. In *44th FOCS*, pages 92–101, 2003.
33. J. Kilian. Founding cryptography on oblivious transfer. In *20th STOC*, pages 20–31, 1988.
34. J. Kilian. A general completeness theorem for two-party games. In *23rd STOC*, pages 553–560, 1991.
35. V. King, J. Saia, V. Sanwalani, and E. Vee. Towards secure and scalable computation in peer-to-peer networks. In *47th FOCS*, pages 87–98, 2006.
36. G. Margulis. Explicit constructions of concentrators. *Problemy peredaci informacii*, 9(4):71–80, 1973.

37. R. Meier, B. Przydatek, and J. Wullschleger. Robuster combiners for oblivious transfer. In *TCC '07*, volume 4392 of *LNCS*, pages 404–418. Springer, 2007.
38. M. Naor and B. Pinkas. Efficient oblivious transfer protocols. In *SIAM Symposium on Discrete Algorithms (SODA 2001)*, pages 448–457, 2001.
39. R. Ostrovsky, S. Rajagopalan, and U. Vazirani. Simple and efficient leader election in the full information model. In *26th STOC*, pages 234–242, 1994.
40. M.O. Rabin. How to exchange secrets by oblivious transfer. TR-81, Harvard, 1981.
41. J. Radhakrishnan and A. Ta-Shma. Bounds for dispersers, extractors, and depth-two superconcentrators. *SIAM J. Discrete Math.*, 13(1):2–24, 2000.
42. O. Reingold, S. Vadhan, and A. Wigderson. Entropy waves, the zig-zag graph product, and new constant-degree expanders and extractors. *ECCC*, 8(18), 2001.
43. R. Shaltiel. Recent developments in explicit constructions of extractors. *Bulletin of the EATCS*, 77:67–95, 2002.
44. S. Wiesner. Conjugate coding. *SIGACT News*, 15(1):78–88, 1983.
45. J. Wullschleger. Oblivious transfer amplification. In *EUROCRYPT ' 2007, LNCS*, volume 4515, pages 555–572. Springer, 2004.
46. A. C. Yao. Protocols for secure computations. In *23rd FOCS*, pages 160–164, 1982.
47. A. C. Yao. How to generate and exchange secrets. In *27th FOCS*, pages 162–167, 1986.
48. D. Zuckerman. Randomness-optimal sampling, extractors, and constructive leader election. In *28th STOC*, pages 286–295, 1996.