Secure Multiparty Computation with Minimal Interaction

Yuval Ishai[1]*, Eyal Kushilevitz[2]**, and Anat Paskin[2]

[1] Computer Science Department, Technion and UCLA (yuvali@cs.technion.ac.il)
[2] Computer Science Department, Technion ({eyalk,anatp}@cs.technion.ac.il)

Abstract. We revisit the question of secure multiparty computation (MPC) with two rounds of interaction. It was previously shown by Gennaro et al. (Crypto 2002) that 3 or more communication rounds are necessary for general MPC protocols with guaranteed output delivery, assuming that there may be $t \geq 2$ corrupted parties. This negative result holds regardless of the total number of parties, even if *broadcast* is allowed in each round, and even if only *fairness* is required. We complement this negative result by presenting matching positive results.
Our first main result is that if only *one* party may be corrupted, then $n \geq 5$ parties can securely compute any function of their inputs using only *two* rounds of interaction over secure point-to-point channels (without broadcast or any additional setup). The protocol makes a black-box use of a pseudorandom generator, or alternatively can offer unconditional security for functionalities in NC^1.
We also prove a similar result in a client-server setting, where there are $m \geq 2$ clients who hold inputs and should receive outputs, and n additional servers with no inputs and outputs. For this setting, we obtain a general MPC protocol which requires a single message from each client to each server, followed by a single message from each server to each client. The protocol is secure against a single corrupted client and against coalitions of $t < n/3$ corrupted servers.
The above protocols guarantee output delivery and fairness. Our second main result shows that under a relaxed notion of security, allowing the adversary to selectively decide (after learning its own outputs) which honest parties will receive their (correct) output, there is a general 2-round MPC protocol which tolerates $t < n/3$ corrupted parties. This protocol relies on the existence of a pseudorandom generator in NC^1 (which is implied by standard cryptographic assumptions), or alternatively can offer unconditional security for functionalities in NC^1.

Key Words: Secure multiparty computation, round complexity.

1 Introduction

This work continues the study of the round complexity of secure multiparty computation (MPC) [53,29,9,13]. Consider the following motivating scenario. Two or more employees wish to take a vote on some sensitive issue and let their manager only learn

* Supported in part by ISF grant 1310/06, BSF grant 2008411 and NSF grants 0830803, 0716835, 0627781.

** Work done in part while visiting UCLA. Supported in part by ISF grant 1310/06 and BSF grant 2008411.

whether a majority of the employees voted "yes". Given an external trusted server, we have the following minimalist protocol: each employee sends her vote to the server, who computes the result and sends it to the manager.

When no single server can be completely trusted, one can employ an MPC protocol involving the employees, the manager, and (possibly) additional servers. A practical disadvantage of MPC protocols from the literature that offer security against malicious parties is that they involve a substantial amount of interaction. This interaction includes 3 or more communication rounds, of which at least one requires broadcast messages.

The question we consider is whether it is possible to obtain protocols with only two rounds of interaction, which resemble the minimal interaction pattern of the centralized trusted server solution described above. That is, we would like to employ several *untrusted* servers instead of a single trusted server, but still require each employee to only send a single message to each server and each server to only send a single message to the manager. (All messages are sent over secure point-to-point channels, without relying on a broadcast channel or any further setup assumptions.)

In a more standard MPC setting, where there are n parties who may contribute inputs and expect to receive outputs, the corresponding goal is to obtain MPC protocols which involve only two rounds of point-to-point communication between the parties.

The above goal may seem too ambitious. In particular:

- Broadcast is a special case of general MPC, and implementing broadcast over secure point-to-point channels generally requires more than two rounds [23].
- Even if a free use of broadcast messages is allowed in each round, it is known that three or more communication rounds are necessary for general MPC protocols which tolerate $t \geq 2$ corrupted parties and guarantee output delivery, regardless of the total number of parties [26].

However, neither of the above limitations rules out the possibility of realizing our goal in the case of a *single* corrupted party, even when the protocols should guarantee output delivery (and in particular fairness). This gives rise to the following question:

Question 1. Are there general MPC protocols (i.e., ones that apply to general functionalities with n inputs and n outputs) that resist a single malicious party, guarantee output delivery, and require only two rounds of communication over point-to-point channels?

The above question may be highly relevant to real world situations where the number of parties is small and the existence of two or more corrupted parties is unlikely.

Another possibility left open by the above negative results is to tolerate $t > 1$ malicious parties by settling for a weaker notion of security against malicious parties. A common relaxation is to allow the adversary who controls the malicious parties to *abort* the protocol. There are several flavors of "security with abort." The standard notion from the literature (cf. [28]) allows the adversary to first learn the output, and then decide whether to (1) have the correct outputs delivered to the uncorrupted parties, or (2) abort the protocol and have *all* uncorrupted parties output a special abort symbol "$\perp$".

Unfortunately, the latter notion of security is not liberal enough to get around the first negative result. But it turns out that a further relaxation of this notion, which we refer to as security with *selective abort*, is not ruled out by either of the above negative results. This notion, introduced in [30], differs from the standard notion of security with abort in that it allows the adversary (after learning its own outputs) to individually decide for each uncorrupted party whether this party will obtain its correct output or will output "$\perp$".[1] Indeed, it was shown in [30] that two rounds of communication over point-to-point channels are sufficient to realize *broadcast* under this notion, with an arbitrary number of corrupted parties. This gives rise to the following question:

Question 2. Are there general MPC protocols that require only two rounds of communication over point-to-point channels and provide security with *selective abort* against $t > 1$ malicious parties?

We note that both of the above questions are open even if broadcast messages are allowed in each of the two rounds.

1.1 Our Results

We answer both questions affirmatively, complementing the negative results in this area with matching positive results.

- Our first main result answers the first question by showing that if only *one* party can be corrupted, then $n \geq 5$ parties can securely compute any function of their inputs with guaranteed output delivery by using only *two* rounds of interaction over secure point-to-point channels (without broadcast or any additional setup). The protocol can provide computational security for general functionalities (assuming one-way functions exist) or statistical security for functionalities in NC^1.
- We also prove a similar result in the client-server setting (described in the initial motivating example), where there are $m \geq 2$ clients who hold inputs and/or should receive outputs, and n additional servers with no inputs and outputs. For this setting, we obtain a general MPC protocol which requires a single message from each client to each server, followed by a single message from each server to each client. The protocol is secure against a single corrupted client and against coalitions of $t < n/3$ corrupted servers,[2] and guarantees output delivery to the clients. We note that the proofs of the negative results from [26] apply to this setting as well, ruling out protocols that resist a coalition of a client and a server.

As is typically the case for protocols in the setting of an honest majority, the above protocols are in fact UC-secure [12,43]. Moreover, similarly to the constant-round protocols from [21,46] (and in contrast to the protocol from [7]), the general version of the above protocols can provide computational security while making only a *black-box* use

[1] Our notions of "security with abort" and "security with selective abort" correspond to the notions of "security with unanimous abort and no fairness" and "security with abort and no fairness" from [30]. We note that the negative result from [26] can be extended to rule out the possibility of achieving fairness in our setting with $t > 1$.

[2] Achieving the latter threshold requires the complexity of the protocol to grow exponentially in the number of servers n. When $t = O(n^{1/2} \log n)$, the complexity of the protocol can be made polynomial in n.

of a pseudorandom generator (PRG). This suggests that the protocols may be suitable for practical implementations.

Our second main result answers the second question, showing that by settling for security with selective abort, one can tolerate a constant fraction of corrupted parties:

– There is a general 2-round MPC protocol over secure point-to-point channels which is secure with *selective abort* against $t < n/3$ malicious parties. The protocol can provide computational security for general functionalities (assuming there is a PRG in NC^1, which is implied by most standard cryptographic assumptions [2]) or statistical security for functionalities in NC^1.

We note that the bound $t < n/3$ matches the security threshold of the best known 2-round protocols in the *semi-honest* model [9,7,33]. Thus, the above result provides security against malicious parties without any loss in round complexity or resilience. In the case of security against malicious parties, previous constant-round MPC protocols (e.g., the ones from [7,25,39]) require at least 3 rounds using broadcast, or at least 4 rounds over point-to-point channels using a 2-round implementation of broadcast with selective abort [30].

Our results are motivated not only by the quantitative goal of minimizing the amount of interaction, but also by several *qualitative* advantages of 2-round protocols over protocols with three or more rounds. In a client-server setting, a 2-round protocol does not require servers to communicate with each other or even to know which other servers are employed. The minimal interaction pattern also allows to break the secure computation process into two *non-interactive* stages of input contribution and output delivery. These stages can be performed independently of each other in an asynchronous manner, allowing clients to go online only when their inputs change, and continue to (passively) receive periodic outputs while inputs of other parties may change. Finally, their minimal interaction pattern allows for a simpler and more direct *security analysis* than that of comparable protocols from the literature with security against malicious parties.

1.2 Related Work

The round complexity of secure computation has been the subject of intense study. In the 2-party setting, 2-round protocols (in different security models and under various setup assumptions) were given in [53,52,10,31,15]. Constant-round 2-party protocols with security against malicious parties were given in [45,41,46,37,35]. In [41] it was shown that the optimal round complexity for secure 2-party computation without setup is 5 (where the negative result is restricted to protocols with black-box simulation).

More relevant to our work is previous work on the round complexity of MPC with an honest majority and guaranteed output delivery. In this setting, constant-round protocols were given in [4,7,6,5,33,25,17,34,19,21,39,40,16]. In particular, it was shown in [25] that 3 rounds are sufficient for general secure computation with $t = \Omega(n)$ malicious parties, where one of the rounds requires broadcast. Since broadcast in the presence of a single malicious party can be easily done in two rounds, this yields 4-round protocols in our setting. The question of minimizing the exact round complexity of MPC over point-to-point networks was explicitly considered in [39,40]. In contrast to the present work, the focus of these works is on obtaining nearly optimal resilience.

Two-round protocols with guaranteed output delivery were given in [26] for specific functionalities, and for general functionalities in [19,16]. However, the protocols from [19,16] rely on broadcast as well as *setup* in the form of correlated randomness.

The round complexity of verifiable secret sharing (VSS) was studied in [25,24,40,50]. Most relevant to the present work is the existence of a 1-round VSS protocol which tolerates a single corrupted party [25]. However, it is not clear how to use this VSS protocol for the construction of two-round MPC protocols. The recent work on the round complexity of *statistical VSS* [50] is also of relevance to our work. In the case where $n = 4$ and $t = 1$, this work gives a VSS protocol in which both the sharing phase and the reconstruction phase require two rounds. Assuming that two rounds of reconstruction are indeed necessary (which is left open by [50]), the number of parties in the statistical variant of our first main result is optimal. (Indeed, 4-party VSS with a single round of reconstruction reduces to 4-party MPC of a linear function, which is in NC^1.)

Finally, a non-interactive model for secure computation, referred to as the *private simultaneous messages* (PSM) model, was suggested in [22] and further studied in [32]. In this model, two or more parties hold inputs as well as a shared secret random string. The parties privately communicate to an external referee some predetermined function of their inputs by simultaneously sending messages to the referee. Protocols for the PSM model serve as central building blocks in our constructions. However, the model of [22] falls short of our goal in that it requires setup in the form of shared private randomness, it cannot deliver outputs to some of the parties, and does not guarantee output delivery in the presence of malicious parties.

Organization. Following some preliminaries (Section 2), Section 3 presents a 2-round protocol in the client-server model. Our first main result (a fully secure protocol for $t = 1$ and $n \geq 5$) is presented in Section 4 and our second main result (security with selective abort for $t < n/3$) in Section 5. For lack of space, some of the definitions and protocols, as well as most of the proofs, are deferred to the full version.

2 Preliminaries

2.1 Secure Computation

We consider n-party protocols that involve two rounds of synchronous communication over secure point-to-point channels. All of our protocols are secure against rushing, adaptive adversaries, who may corrupt at most t parties for some specified security threshold t. See [11,12,28] and the full version for more complete definitions.

In addition to the standard simulation-based notions of *full security* (with guaranteed output delivery) and *security with abort*, we consider several other relaxed notions of security. Security in the *semi-honest model* is defined similarly to the standard definition, except that the adversary cannot modify the behavior of corrupted parties (only observe their secrets). *Privacy* is also the same as in the standard definition, except that the environment can only obtain outputs from the adversary (or simulator) and not from the uncorrupted parties. Intuitively, this only ensures that the adversary does not learn anything about the inputs of uncorrupted parties beyond what it could have learned by submitting to the ideal functionality some (distribution over) valid inputs. Privacy, however, does not guarantee any form of correctness. *Privacy with knowledge of outputs* is similar to privacy except that the adversary is also required to "know" the (possibly incorrect) outputs of the honest parties. This notion is defined similarly to full security (in particular, the environment receives outputs from both the simulator and the honest parties), with the difference that the ideal functionality first delivers the corrupted parties' output to the simulator, and then receives from the simulator an output to deliver to *each* of the uncorrupted parties. Finally, *security with selective abort* is defined similarly to security with abort, except that the simulator can decide for each uncorrupted party whether this party will receive its output or $\perp$.

2.2 The PSM Model

A *private simultaneous messages* (PSM) protocol [22] is a non-interactive protocol involving m parties P_i, who share a common random string r, and an external referee who has no access to r. In such a protocol, each party sends a single message to the referee based on its input x_i and r. These m messages should allow the referee to compute some function of the inputs without revealing any additional information about the inputs. Formally, a PSM protocol for a function $f : \{0,1\}^{\ell \times m} \to \{0,1\}^*$ is defined by a randomness length parameter $R(\ell)$, m message algorithms $A_1, ..., A_m$ and a reconstruction algorithm $\mathtt{Rec}$, such that the following requirements hold.

- Correctness: for every input length ℓ, all $x_1, ..., x_m \in \{0,1\}^\ell$, and all $r \in \{0,1\}^{R(\ell)}$, we have $\mathtt{Rec}(A_1(x_1, r), ..., A_m(x_m, r)) = f(x_1, ..., x_m)$.
- Privacy: there is a simulator $\mathcal{S}$ such that, for all $x_1, ..., x_m$ of length ℓ, the distribution $\mathcal{S}(1^\ell, f(x_1, ..., x_m))$ is indistinguishable from $(A_1(x_1, r), ..., A_m(x_m, r))$.

We consider either perfect or computational privacy, depending on the notion of indistinguishability. (For simplicity, we use the input length ℓ also as security parameter, as in [28]; this is without loss of generality, by padding inputs to the required length.)

A *robust* PSM protocol should additionally guarantee that even if a subset of the m parties is malicious, the protocol still satisfies a notion of "security with abort." That is, the effect of the messages sent by corrupted parties on the output can be simulated by either inputting to f a valid set of inputs (independently of the honest parties' inputs) or by making the referee abort. This is formalized as follows.

- Statistical robustness: For any subset $T \subset [m]$, there is an efficient (black-box) simulator $\mathcal{S}$ which, given access to the common r and to the messages sent by (possibly malicious) parties $P_i^*, i \in T$, can generate a distribution x_T^* over $x_i, i \in T$, such that the output of $\mathtt{Rec}$ on inputs $A_T(x_T^*, r), A_{\bar{T}}(x_{\bar{T}}, r)$ is statistically close to the "real-world" output of $\mathtt{Rec}$ when receiving messages from the m parties on a randomly chosen r. The latter real-world output is defined by picking r at random, letting party P_i pick a message according to A_i, if $i \notin T$, and according to P_i^* for $i \in T$, and applying $\mathtt{Rec}$ to the m messages. In this definition, we allow $\mathcal{S}$ to produce a special symbol $\perp$ (indicating "abort") on behalf of some party P_i^*, in which case $\mathtt{Rec}$ outputs $\perp$ as well.

The following theorem summarizes some known facts about PSM protocols that are relevant to our work.

Theorem 1. *[22] (i) For any $f \in \mathrm{NC}^1$, there is a polynomial-time, perfectly private and statistically robust PSM protocol. (ii) For any polynomial-time computable f, there is a polynomial-time, computationally private and statistically robust PSM protocol which uses any pseudorandom generator as a black box.*

For self-containment, the full version contains a full description and proof of the robust variants, which are only sketched in [22, Appendix C].

2.3 Secret Sharing

An (n, t)-*threshold* secret sharing scheme, also referred to as a t-*private* secret sharing scheme, is an n-party secret sharing scheme in which every t parties learn nothing about the secret, and every $t + 1$ parties can jointly reconstruct it. In this work, we rely on variants of several standard secret sharing schemes, such as Shamir's scheme [49], a bivariate version of Shamir's scheme [9], and the CNF scheme [36].

Recall that in Shamir's scheme over a finite field $\mathbb{F}$ (where $|\mathbb{F}| > n$), a secret $s \in \mathbb{F}$ is shared by picking a random polynomial p of degree (at most) t over $\mathbb{F}$ such that $p(0) = s$, and distributing to each party P_i the value of p on a distinct field element associated with this party. In the bivariate version of Shamir's scheme, P_i receives the i'th row and column (from an $n \times n$ matrix of evaluations) of a random bivariate polynomial $p(x, y)$ of degree at most t in each variable such that $p(0, 0) = s$. In the CNF scheme over an Abelian group G, a secret $s \in G$ is shared by first additively breaking it into $\binom{n}{t}$ shares, a share per size-t subset of $[n]$, and then distributing to P_i all shares corresponding those subsets T such that $i \notin T$.

We will refer to a few abstract properties of secret sharing schemes which will be useful for our protocols. In a d-*multiplicative* secret sharing scheme over $\mathbb{F}$, each party should be able to apply a local computation (denoted MULT) on its shares of d secrets, such that the outcomes of the n local computations always add up to the product of the d secrets (where addition and multiplication are in $\mathbb{F}$). The standard notion of multiplicative secret sharing from [20] corresponds to the case $d = 2$. The three concrete schemes, mentioned above, are d-multiplicative when $n > dt$.

Another property we will rely on is *pairwise verifiability*. This property has been implicitly used in the context of verifiable secret sharing [9,20,25]. In a pairwise verifiable scheme, checking that the shares are globally consistent (with some sharing of some secret) reduces to pairwise equality tests between values that are locally computed by pairs of parties. More concretely, each pair i, j defines an equality test between a value computed from the share of P_i and a value computed from the share of P_j. These $\binom{n}{2}$ equality tests should have the property that for any subset T of two or more parties, if all $\binom{|T|}{2}$ tests involving parties in T pass then the shares given to T are consistent with some valid sharing of a secret. The CNF and bivariate Shamir schemes are pairwise verifiable. For instance, in the CNF scheme, each pair of parties compares the $\binom{n-2}{t}$ additive shares they should have in common. See the full version for more details, including a construction of an efficient secret sharing scheme over the binary field $\mathbb{F}_2$ which is both d-multiplicative and pairwise verifiable.

One last property we will need is *efficient extendability*. A secret sharing scheme is efficiently extendable, if for any subset $T \subseteq [n]$, it is possible to efficiently check whether the (purported) shares to T are consistent with a valid sharing of some secret s. Additionally, in case the shares are consistent, it is possible to efficiently sample a (full) sharing of some secret which is consistent with that partial sharing. This property is satisfied, in particular, by the schemes mentioned above, as well as any so-called "linear" secret sharing scheme.

3 A Protocol in the Client-Server Model

In this section, we present a two-round protocol which operates in a setting where the parties consist of m *clients* and n *servers*. The clients provide the inputs to the protocol (in its first round) and receive its output (in its second round) but the "computation" itself is performed by the servers alone. Our construction provides security against any adversary that corrupts either a single client or at most t servers. We refer to this kind of security as $(1, t)$-security[3]. The protocol in this setting illustrates some of the techniques we use throughout the paper, and it can be viewed as a warmup towards our main results; hence, we do not present here the strongest statement (e.g., in terms of resilience) and defer various improvements to the full version. Specifically, for any functionality $f \in$ POLY, we present a 2-round $(1, t)$-secure MPC protocols (with guaranteed output delivery) for $m \geq 2$ clients and $n = \Theta(t^3)$ servers. The protocol makes a black-box use of a PRG, or alternatively can provide unconditional security for $f \in \mathrm{NC}^1$.

Tools. Our protocol relies on the following building blocks:

1. An (n, t)-*secret sharing scheme* for which it is possible to check in NC^1 whether a set of more than t shares is consistent with some valid secret. For instance, Shamir's scheme satisfies this requirement. Unlike the typical use of secret sharing in the context of MPC, our constructions do not rely on linearity or multiplication property of the secret sharing scheme.
2. A set system $\mathcal{T} \subseteq 2^{[n]}$ of size ℓ such that (a) $\mathcal{T}$ is t-*resilient*, meaning that every $B \subseteq [n]$ of size t avoids at least $\ell/2 + 1$ sets; and (b) $\mathcal{T}$ is *$(t + 1)$-pairwise intersecting*, meaning that for all $T_1, T_2 \subseteq \mathcal{T}$ we have $|T_1 \cap T_2| \geq t + 1$. See the full version for a construction with $n = \Theta(t^3), \ell = poly(n)$.
3. A PSM protocol, with the best possible privacy (according to Theorem 1, either perfect or computational) for some functions f' depending on f (see below).

Perfect security with certified randomness. We start with a protocol for $m \geq 2$ clients and $n = \Theta(t^3)$ servers, denoted Π^R, that works in a scenario where each set of servers $T \in \mathcal{T}$, shares a common random string r_T (obtained in a trusted setup phase). We explain how to get rid of this assumption later.

– **Round 1:** Each Client i secret-shares its input x_i among the n servers using the t-private secret sharing scheme.

[3] Recall that the impossibility results of [26] imply that general 2-round protocols in this setting tolerating a coalition of a client and a server are impossible.

– **Round 2:** For each $T \in \mathcal{T}$ and $i \in [m]$, the set T runs a PSM protocol with the shares s received from the clients in Round 1 as inputs, r_T as the common randomness, and Client i as the referee (i.e., one message is sent from each server in T to Client i). This PSM protocol computes the following functionality f'_i:
  - If all shares are consistent with some input value x, then $f'_i(s) = f(x)$.
  - Else, if the shares of a single Client i are inconsistent, let $f'_i(s) = \perp$.
  - Otherwise, let j be the smallest such that the shares of Client j are inconsistent. Then, $f'_i(s)$ is an "accusation" of Client j; i.e., a pair $(j, f(x'))$, where x' is obtained from x by replacing x_j with 0.
– **Reconstruction:** Each Client i computes its output as follows: If all sets T blame some Client j, then output the (necessarily unanimous) "backup" output $f(x')$ given by the PSM protocols. Otherwise, output the majority of the outputs reported by non-blaming sets T.

Proof idea. If the adversary corrupts at most t servers (and no client), then privacy follows from the use of a secret sharing scheme (with threshold t). By the t-resilience of the set system, a majority of the sets $T \in \mathcal{T}$ contain no corrupted server and thus will not blame any client and will output the correct value $f(x)$.

If the adversary corrupts Client j, then all servers are honest. Every set $T \in \mathcal{T}$ either does not blame any client or blames Client j. Consider two possible cases: (a) Client j makes all sets T observe inconsistency: in such a case, Client j receives $\perp$ from all T and hence does not learn any information; moreover, all honest clients will output the same backup output $f(x')$. (b) Client j makes some subsets T observe consistent shares: since the intersection of every two subsets in $\mathcal{T}$ is of size at least $t+1$ then, using the $(t+1)$ reconstruction threshold of the secret sharing scheme, every two non-blaming sets must agree on the same input x. This means that Client j only learns $f(x)$. Moreover, all other (honest) clients will receive the actual output $f(x)$ from at least one non-blaming set T and, as discussed above, all outputs from non-blaming sets must agree.

Observe that the fact that a set T uses the same random string r_T in all m PSM instances it participates in does not compromise privacy. This is because in each of them the output goes to a different client and only a single client may be corrupted.[4]

Lemma 1. *Π^R is a 2-round, $(1,t)$-secure MPC protocol for $m > 1$ clients and $n = \Theta(t^3)$ servers, assuming that the servers in each set $T \in \mathcal{T}$ have access to a common random string r_T (unknown to the clients). The security can be made perfect for $f \in \mathrm{NC}^1$, and computational for $f \in$ POLY by making a black-box use of a PRG.*

Note that the claim about $f \in \mathrm{NC}^1$ holds since the functions f' evaluated by the PSM sub-protocols are in NC^1 whenever f is.

Removing the certified randomness assumption. If we have at least 4 clients, we can let each Client i generate its own candidate PSM randomness r^i_T and send it to all servers in T. Each PSM protocol, corresponding to some set T, is executed using each of these

[4] Alternatively, r_T can be made sufficiently long so that the set T can use a distinct portion of r_T in each invocation of a PSM sub-protocol.

strings, where in the i-th invocation (using randomness r_T^i) Client i receives no message (otherwise, the privacy of the protocol could be compromised). The other clients receive the original messages as prescribed by the PSM protocol. Upon reconstruction, Client i lets the PSM output for a set T be the majority over the $m-1$ PSM outputs it sees. We observe that this approach preserves perfect security. If statistical (or computational) security suffices, we can rely on 2 or 3 clients as well. Here, the high level idea is to use a transformation as described for the case $m \geq 4$, and let Client i authenticate the consistency of the randomness r_T^j used in the j'th PSM protocol executed by set T, using a random string it sent in Round 1. Upon reconstruction, each client only considers the PSM executions which passed the authentication. Combining the above discussion with Theorem 1, we obtain the following theorem.

Theorem 2. *There exists a statistically $(1, t)$-secure 2-round general MPC protocol in the client-server setting for $m \geq 2$ clients and $n = \Theta(t^3)$ servers. For $f \in \mathrm{NC}^1$, the protocol is perfectly secure if $m \geq 4$, and statistically secure otherwise. The protocol is computationally secure for $f \in$ POLY.*

4 Full Security for $t = 1$

In this section, we return to the standard model where all parties may contribute inputs and receive outputs. We present a 2-round protocol in this model for $n \geq 5$ parties and $t = 1$. This protocol uses some similar ideas to our basic client-server protocol above, but it is different in the types of secret sharing scheme and set system that it employs. Specifically, we use the following ingredients:

1. A 1-private *pairwise verifiable* secret sharing scheme (see Section 2.3). For simplicity, we use here the CNF scheme, though one could use the bivariate version of Shamir's scheme for better efficiency. Recall that in the 1-private CNF scheme the secret s is shared by first randomly breaking it into n additive parts $s = s_1 + \ldots + s_n$, and then distributing each s_i to all parties *except* for party i. Here we can view a secret as an element of $\mathbb{F}_2^m$.
2. A robust $(n-2)$-party PSM protocol (see Section 2.2). In particular, such a PSM protocol ensures that the effect of any single malicious party on the output can be simulated in the ideal model (allowing the simulator to send "abort" to the functionality).
3. A simple set system, consisting of the $\binom{n}{2}$ sets $T_{i,j} = [n] \setminus \{i, j\}$. (Note that, for $n \geq 5$, we have $|T_{i,j}| \geq 3$.)

Again, we assume for simplicity that members of each set $T_{i,j}$ share common randomness $r_{i,j}$. Similarly to the client-server setting, this assumption can be eliminated by letting 3 of the parties in $T_{i,j}$ pick their candidate for $r_{i,j}$ and distributing it to the parties in the set (in Round 1 of our protocol), and then letting $T_{i,j}$ execute the PSM sub-protocol (in Round 2) using each of the 3 candidates and sending the outputs to P_i, P_j (which are not in the set); the final PSM output will be the majority of these three outputs. Finally, for a graph G, let $\mathrm{VC}(G)$ denote the size of the *minimal vertex cover* in G.

Our protocol proceeds as follows:

- **Round 1:** Each party P_k shares its input x_k among all *other* parties using a 1-private, $(n-1)$-party CNF scheme (i.e., each party gets $n-2$ out of the $n-1$ additive shares of x_k). In addition, to set up the consistency checks, each pair P_i, P_j $(i < j)$ generates a shared random pad $s_{i,j}$ by having P_i pick such a pad and send it to P_j.
- **Round 2:** For each "dealer" P_k, each pair P_i, P_j send the $n-3$ additive shares from P_k they should have in common, masked with the pad $s_{i,j}$, to all parties.[5] Following this stage, each party P_i has an inconsistency graph $G_{i,k}$ corresponding to each dealer P_k $(k \neq i)$, with node set $[n] \setminus \{k\}$ and edge (j,l) if P_j, P_l report inconsistent shares from P_k.
In addition, each set $T_{i,j}$ invokes a robust PSM protocol whose inputs are all the shares received (in Round 1) by the $n-2$ parties in this set, and whose outputs to P_i, P_j (which are not in $T_{i,j}$) are as follows:
- If all input shares are consistent with some input x, then both P_i, P_j receive $v = f(x)$.
- Else, if shares originating from exactly one P_k are inconsistent, then P_k gets $\perp$ (in case $k \in \{i,j\}$) and the other party(s) get an "accusation" of P_k; namely, a pair (k, x^*) where $x^* = (x_1, \ldots, x_{k-1}, x'_k, x_{k+1}, \ldots, x_n)$. Here, each x_j (for $j \neq k$) is the protocol input recovered from the (consistent) shares and $x'_k = x_k$ if the shares of any $n-3$ out of the $n-2$ parties in $T_{i,j}$ are consistent with each other and $x'_k = 0$ (a default value) otherwise.
- Else, if shares originating from more than one party are inconsistent, output $\perp$.
- **Reconstruction:** Each party P_i uses the $n-1$ inconsistency graphs $G_{i,k}$ $(k \neq i)$, and the PSM outputs that it received, to compute its final output:
(a) If some inconsistency graph $G_{i,k}$ has $\mathrm{VC}(G_{i,k}) \geq 2$ then the PSM output of $T_{i,k}$ is of the form (k, x^*); substitute x^*_k by 0, to obtain x', and output $f(x')$.
Else, (b) if some inconsistency graph $G_{i,k}$ has a vertex cover $\{j\}$ and at least 2 edges, consider the PSM outputs of $T_{i,j}, T_{i,k}$ (assume that $i \neq j$; if $i = j$ it is enough to consider the output of $T_{i,k}$). If any of them outputs v of the form $f(x)$ then output v; otherwise, if the output is of the form (k, x^*), output $f(x^*)$.
Else, (c) if some inconsistency graph $G_{i,k}$ contains exactly one edge (j, j'), consider the outputs of $T_{i,j}, T_{i,j'}$ (again, assume $i \notin \{j, j'\}$), and use any of them which is non-$\perp$ to extract the output (either directly, if the output is of the form $f(x)$, or $f(x^*)$ from an output (k, x^*)).
Finally, (d) if all $G_{i,k}$'s are empty, find some $T_{i,j}$ that outputs $f(x)$ (with no accusation), and output this value.

Intuitively, a dishonest party P_d may deviate from the protocol in very limited ways: it may distribute inconsistent shares (in Round 1) which will be checked (in Round 2) and will either be caught (if the inconsistency graph has VC larger than 1) or will be "corrected" (either to a default value or to its original input, if the VC is of size at

[5] This is similar to Round 2 of the 2-round VSS protocol of [25], except that we use point-to-point communication instead of broadcast; note that, in our case, if the dealer is dishonest, then all other parties are honest.

most 1). P_d may report false masked shares, for the input of some parties, but this will result in very simple inconsistency graphs (with vertex cover of size 1) that can be detected and fixed. And, finally, P_d may misbehave in the robust PSM sub-protocols (in which it participates) but this has very limited influence on their output (recall that, for sets in which P_d participates, it does not receive the output). A detailed analysis appears in the full version. This proves:

Theorem 3. *There exists a general, 2-round MPC protocol for $n \geq 5$ parties which is fully secure (with guaranteed output delivery) against a single malicious party. The protocol provides statistical security for functionalities in NC^1 and computational security for general functionalities by making a black-box use of a pseudorandom generator.*

5 Security with Selective Abort

This section describes our second main result; namely, a 2-round protocol which achieves security with *selective abort* against $t < n/3$ corruptions. This means that the adversary, after learning its own outputs, can selectively decide which honest parties will receive their (correct) output and which will output "$\perp$". More precisely, we prove the following theorem:

Theorem 4. *There exists a general 2-round MPC protocol for $n > 3t$ parties which is t-secure, with* selective abort. *The protocol provides statistical security for functionalities in NC^1 and computational security for functionalities in POLY, assuming the existence of a pseudorandom generator in NC^1.*

Our high-level approach is to apply a sequence of reductions, where the end protocol we need to construct only satisfies the relaxed notion of "privacy with knowledge of outputs", described in Section 2, and only applies to vectors of degree-3 polynomials. In particular,

1. We concentrate, without loss of generality, on functionalities which are deterministic with a public output.
2. We reduce (using unconditional one-time MACs) the secure evaluation of a function $f \in$ POLY to a private evaluation, with knowledge of outputs, of a related functionality $f' \in$ POLY. The reduction is statistical, and if $f \in \mathrm{NC}^1$ then so is f'.
3. We reduce the private evaluation with knowledge of outputs of a function $f' \in$ POLY to a private evaluation with knowledge of outputs of a related functionality f'', where f'' is a vector of degree-3 polynomials. The reduction (using [34]) is perfect for functions in NC^1, and only computationally secure (using [2]) for general functionalities in POLY.
4. We present a 2-round protocol that allows $dt + 1$ parties to evaluate a vector of degree-d polynomials, for all $d \geq 1$, and provides privacy with knowledge of outputs. In particular, for $d = 3$ the protocol requires $n = 3t + 1$ parties.

In the following subsections, we describe steps 2–4 in detail.

5.1 A private protocol with knowledge of outputs

In this section, we present a 2-round protocol for degree-d polynomials which is private with knowledge of outputs. Let $p(x_1, \ldots, x_m)$ be a multivariate polynomial over a finite field $\mathbb{F}$, of total degree d. Assume, without loss of generality, that the degree of each monomial in p is exactly d.[6] Hence, p can be written as $p = \sum_{g_1 \le \ldots \le g_d} \alpha_g \prod_{l=1}^{d} x_{g_l}$. We start by describing a protocol for evaluating p with security in the semi-honest model. (This protocol is similar to previous protocols from [9,33].) The protocol can rely on any d-multiplicative secret sharing scheme over $\mathbb{F}$. Recall that, in such a scheme, each party should be able to apply a local computation MULT to the shares it holds of some d secrets, to obtain an additive share of their product.

- **Round 1:** Each party P_i, $i \in [n]$, shares every input x_h it holds by computing shares $(s_1^h, \ldots, s_n^h)$, using the d-multiplicative scheme, and distributes them among the parties. P_i also distributes random *additive* shares of 0; i.e., it sends to each P_j a field element z_i^j such that $z_1^j, \ldots, z_n^j$ are random subject to the restriction that they sum up to 0.
- **Round 2:** Each party P_i, $i \in [n]$, computes $y_i = p_i(s_i^1, \ldots, s_i^m) + \sum_{j=1}^{n} z_i^j$, where $p_i(s_i^1, \ldots, s_i^m) \triangleq \sum_{g_1 \le \ldots \le g_d} \alpha_g \text{MULT}(i, s_i^{g_1}, \ldots, s_i^{g_d})$. It sends y_i to all parties.
- **Outputs:** Each party computes and outputs $\sum_{i=1}^{n} y_i$ which is equal to $p(s^1, \ldots, s^m)$, as required.

We will refer to the above protocol as the "basic protocol". The proof of correctness and privacy in the semi-honest case are standard, and are omitted. Interestingly, this basic protocol happens to be *private* with knowledge of outputs (but not secure) against malicious parties for $d \le 2$, when using Shamir's scheme as its underlying secret sharing scheme. However, the following simple example demonstrates that the basic protocol is not private against malicious parties already for $d = 3$. [7]

Example 1. Consider 4 parties where only P_1 is corrupted and the parties want to compute the degree-3 polynomial $x_1x_2x_3$ (party P_4 has no input). We argue that, when $x_3 = 0$, party P_1 can compute x_2, contradicting the privacy requirement. Let $q_2(z) = r_2z + x_2$ and $q_3(z) = r_3z$ be the polynomials used by P_2, P_3 (respectively) to share their inputs. Their product is $q(z) = r_2r_3z^2 + x_2r_3z$. Note that the messages sent by P_1 to the other 3 parties in Round 1 can make P_1 learn (in Round 2) an arbitrary linear combination of the values of $q(z)$ at 3 distinct points. Since the degree of p is at most 2, this means that P_1 can also learn an arbitrary linear combination of the coefficients of q. In particular, it can learn x_2r_3. This alone suffices to violate the privacy of x_2, because it can be used to distinguish with high probability between, say, the case where $x_2 = 0$ and the case $x_2 = 1$.

[6] Otherwise, replace each monomial $m(x)$ of degree $d' < d$ by $m(x) \cdot x_0^{d-d'}$, where x_0 is a dummy variable whose value is set to 1 (by using some fixed valid n-tuple of shares).

[7] Note that degree-3 polynomials are "complete", in the sense that they can be used to represent every function, whereas degree-2 polynomials are not [33].

To prevent badly-formed shares from compromising privacy, we use the following variant of *conditional disclosure of secrets (CDS)* [27] as a building block. This primitive will allow an honest player to reveal a secret s subject to the condition that two secret values a, b held by other two honest players are equal.

Definition 1. *An MCDS (multiparty CDS) protocol is a protocol for n parties, which include three distinct special parties S, A, B. The sender S holds a secret s, and parties A, B hold inputs a, b (respectively). The protocol should satisfy the following properties (as usual, the adversary is rushing).*

1. *If $a = b$, and A, B, S are honest, then all honest parties output s.*
2. *If $a = b$, and A, B are honest, then the adversary's view is independent of a, even conditioned on s.*
3. *If $a \neq b$, and A, B, S are honest, then the adversary's view is independent of s, even conditioned on a, b.*

Note that there is no requirement when $a \neq b$ and some of the special parties are corrupted (e.g., a corrupted A may still learn s). To be useful for our purposes, an MCDS protocol needs to have only two rounds, and also needs to satisfy the technical requirement that the message sent by A and B in the first round do not depend on the values a and b.

A simple MCDS protocol with the above properties may proceed as follows (see the full version for a proof): In Round 1, party A picks random independent values $r, z \in \mathbb{F}$ and sends them to B, and party S sends s to A. In Round 2, A sends to each of the parties $m_A = a \cdot r - z + s$ and B sends $m_B = z - b \cdot r$. Each party outputs $m_A + m_B$.

An MCDS protocol as above will be used to compile the basic protocol for $n = dt + 1$ semi-honest parties into a protocol Π_{priv} which is private against malicious parties. For this, we instantiate the basic protocol with a d-multiplicative secret sharing scheme which is also pairwise-verifiable and efficiently extendable (see Section 2.3). More precisely, the parties run the basic protocol, and each party P_i masks its Round 2 message with a sum of random independent masks $s_{i,j,k,h}$, corresponding to a shared input x_h and a pair of parties P_j, P_k (not holding x_h). In parallel, the MCDS protocol is executed for revealing each pad $s_{i,j,k,h}$ under the condition that the shares of x_h given to P_j and P_k are consistent, as required by the pairwise verifiable scheme (where a, b in the MCDS are values locally computed by P_j, P_k that should be equal by the corresponding local check). Intuitively, this addresses the problem in Example 1 by ensuring that, if a party sends inconsistent shares of one of its inputs to the honest parties, some consistency check would fail (by pairwise-verifiability), and thus at least one random mask is not "disclosed" to the adversary, and so the adversary learns nothing.

The resulting protocol Π_{priv} proceeds as follows:

- **Round 1:**
  - Each party P_i, $i \in [n]$ shares every input x_h it holds by computing shares $(s^h_1, \dots, s^h_n)$ and distributing them among the parties. Each P_i also sends to each P_j a share z^i_j where $z^i_1, \dots, z^i_n$ form a random additive sharing of 0.
  - Each triple of distinct parties P_i, P_j, P_k such that $j < k$ runs, for each $h \in [m]$ such that x_h is not held by $\{P_i, P_j, P_k\}$, Round 1 of the MCDS protocol (playing the roles of S, A, B respectively, where all n parties receive the MCDS output), with secret $s = s_{i,j,k,h}$, selected independently at random by P_i.

– **Round 2:**
  - Each party P_i, $i \in [n]$, computes $y_i = p_i(s_i^1, \ldots, s_i^m) + \sum_{j=1}^n z_i^j$, where $p_i(s_i^1, \ldots, s_i^m) \triangleq \sum_{g_1 \leq \ldots \leq g_d \in [m]} \alpha_g \textsc{Mult}(i, s_i^{g_1}, \ldots, s_i^{g_d})$. It sends $y_i' \triangleq y_i + \sum_{j,k,h} s_{i,j,k,h}$ to all parties.
  - Each triple of parties P_i, P_j, P_k runs Round 2 of their MCDS protocols for each (relevant) x_h, where a, b are the outputs of the relevant local computations applied to shares of x_h held by P_j, P_k which should be equal. Denote by $s_{i,j,k,h}^u$ the output of P_u in this MCDS protocol.

– **Outputs:** Each party P_u computes $\sum_{i=1}^n y_i' - \sum_{i,j,k,h} s_{i,j,k,h}^u$.

See the full version, for a proof of the following lemma.

Lemma 2. *Suppose $n = dt + 1$. Then the protocol Π_{priv}, described above, computes the degree-d polynomial p and satisfies statistical t-privacy with knowledge of outputs.*

Remark 1. The above protocol can be easily generalized to support a larger number of parties $n > dt + 1$. This can be done by letting all parties share their inputs among only the first $dt + 1$ parties in the first round, and letting only these $dt + 1$ parties reply to all parties in the second round. A similar generalization applies to the other protocols in this section.

Our protocols were described as if we need to evaluate a single polynomial. To evaluate a vector of polynomials (which is actually required for our application), we make the following observation. Both the basic semi-honest protocol and Π_{priv} can be directly generalized to this case by running one copy of Round 1, except for the additive shares od 0 that are distributed for each output, and then executing Round 2 separately for each polynomial (using the corresponding additive shares). The analysis of the extended protocols is essentially the same. Combining Π_{priv}, instantiated with bivariate Shamir, with the above discussion, we get the following lemma:

Lemma 3. *For any $d \geq 1$ and $t \geq 1$, there exists a 2-round protocol for $n = dt + 1$ parties which evaluates a vector of polynomials of total degree d over a finite field $\mathbb{F}$ of size $|\mathbb{F}| \geq n$, such that the protocol is statistically t-private with knowledge of outputs.*

The transition from degree-3 polynomials to general functions $f \in$ POLY is essentially done by adapting known representations of general functions by degree-3 polynomials [34,2]. That is, securely evaluating $f(x_1, \ldots, x_m) : \{0,1\}^m \to \{0,1\}^*$ is reduced to securely evaluating a vector of randomized polynomials $p(x_1, \ldots, x_m, r_1, \ldots, r_l)$ of degree $d = 3$, over (any) finite field $\mathbb{F}_p$. However, the reduction is not guaranteed to work if the adversary shares a value of x_i's which is not in $\{0,1\}$. If the secret domain of the underlying secret sharing is $\mathbb{F}_2$, then the adversary is unable to share non-binary values, and there is no problem. This is the case with the CNF scheme over $\mathbb{F}_2$, but using $(3t + 1, t)$-CNF would result in exponential (in n) complexity for the protocol. An alternative approach is to rely on (say) bivariate Shamir, but using a variant of the above reduction from [14], applied to a function f' over $\mathbb{F}^m$ (rather than $\{0,1\}^m$) related to f, which is always consistent with $f(x)$, for some $x \in \{0,1\}^m$. In particular, $f' \in \mathrm{NC}^1$ if $f \in \mathrm{NC}^1$ and $f' \in$ POLY if $f \in$ POLY . Another solution is to devise an efficient 3-multiplicative, pairwise-verifiable $(3t+1)$-party scheme over $\mathbb{F}_2$. See the full version, for more details on both solutions. We obtain the following:

Lemma 4. *Suppose there exists a PRG in* NC^1*. Then, for any* n*-party functionality* f*, there exists a 2-round MPC protocol which is (computationally)* t*-private with knowledge of outputs, assuming that* $n > 3t$*. Alternatively, the protocol can provide statistical (and unconditional) privacy with knowledge of outputs for* $f \in \mathrm{NC}^1$*.*

5.2 From privacy with knowledge of outputs to security with selective abort

The final step in our construction is a reduction from secure evaluation of functions with selective abort to private evaluation with knowledge of outputs. For this, we make use of unconditional MACs. Our transformation starts with a protocol Π' for evaluating a single output function f, which is private with knowledge of outputs. We then use Π' to evaluate an augmented (single output) functionality f', which computes f along with n MACs on the output of f, where the i-th MAC uses a private key chosen by party P_i at random. That is, f' takes an input x, and $k_i \in \mathcal{K}$ from each party P_i, and returns $y = f(x)$ along with $\mathtt{MAC}(y, k_1), \ldots, \mathtt{MAC}(y, k_n)$. The protocol Π is obtained by running Π' on f' and having each party P_i locally verify that the output y it gets is consistent with the corresponding MAC. If so, then P_i outputs y; otherwise, it outputs $\perp$. Intuitively, this is enough for getting security with selective abort since to make an uncorrupted party output an inconsistent value, the adversary would have to find y' with $\mathtt{MAC}(y', k) = \mathtt{MAC}(y, k)$ for a random unknown k and a known y, which can only be done with negligible probability. A formal construction and a proof of Theorem 4 appear in the full version.

Acknowledgements. We thank the anonymous CRYPTO 2010 referees for helpful comments and suggestions. We also would like the thank the third author's husband, Beni, for a lot of help on technical issues and for proofreading the paper.

References

1. N. Alon, M. Merritt, O. Reingold, G. Taubenfeld and R.N. Wright. Tight bounds for shared memory systems accessed by Byzantine processes. In *Journal of Distributed Computing*,18(2): 99–109,2005.
2. B. Applebaum, Y. Ishai, and E. Kushilevitz. Computationally private randomizing polynomials and their applications. *Computational Complexity*, 15(2):115–162, 2006.
3. D.A. Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in NC1. In *Proc. 18th STOC*, pp. 150–164. 1986.
4. J. Bar-Ilan and D. Beaver. Non-cryptographic fault-tolerant computing in a constant number of rounds. In *Proc. 8th ACM PODC*, pp. 201–209. 1989.
5. D. Beaver. Minimal-Latency Secure Function Evaluation. In *Eurocrypt '00*, pp. 335–350, 2000. LNCS No. 1807.
6. D. Beaver, J. Feigenbaum, J. Kilian, and P. Rogaway. Security with low communication overhead (extended abstract). In *Proc. of CRYPTO '90*.
7. D. Beaver, S. Micali, and P. Rogaway. The round complexity of secure protocols (extended abstract). In *Proc. 22nd STOC*, pp. 503–513. 1990.
8. A. Beimel Secure Schemes for Secret Sharing and Key Distribution. Phd. thesis. Dept. of Computer Science, 1996.

9. M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness Theorems for Noncryptographic Fault-Tolerant Distributed Computations. *Proc.* 20*th STOC88*, pp. 1–10.
10. C. Cachin, J. Camenisch, J. Kilian, and J. Muller. One-round secure computation and secure autonomous mobile agents. In *Proceedings of ICALP'00*, 2000.
11. R. Canetti. Security and composition of multiparty cryptographic protocols. *Journal of Cryptology*, 13(1): 143–202, 2000.
12. R. Canetti. Universally composable security: A new paradigm for cryptographic protocols.cfik03 In *FOCS*, pp. 136–145, 2001.
13. D. Chaum, C. Crepeau, and I. Damgard. Multiparty Unconditionally Secure Protocols. In *Proc.* 20*th STOC88*, pp. 11–19.
14. R. Cramer, S. Fehr, Y. Ishai, E. Kushilevitz: Efficient Multi-party Computation over Rings. In *Proc. EUROCRYPT 2003*, pp. 596–613
15. S. G. Choi, A. Elbaz, A. Juels, T. Malkin, and M. Yung. Two-Party Computing with Encrypted Data. In *Proc. ASIACRYPT 2007*, pp. 298–314.
16. S. G. Choi, A. Elbaz, T. Malkin, and M. Yung. Secure Multi-party Computation Minimizing Online Rounds. In *Proc. ASIACRYPT 2009*, to appear.
17. R. Cramer and I. Damgård. Secure distributed linear algebra in a constant number of rounds. In *Proc. Crypto 2001*.
18. R. Cramer, I. Damgård, S. Dziembowski, M. Hirt, and T. Rabin. Efficient multiparty computations with dishonest minority. In *Eurocrypt '99*, pp. 311–326, 1999. LNCS No. 1592.
19. R. Cramer, I. Damgård, and Y. Ishai. Share conversion, pseudorandom secret-sharing and applications to secure computation. In *Proc. of second TCC*, 2005.
20. R. Cramer, I. Damgård, U. M. Maurer. General Secure Multi-party Computation from any Linear Secret-Sharing Scheme. EUROCRYPT 2000: 316–334
21. I. Damgård and Y. Ishai. Secure multiparty computation using a black-box pseudorandom generator. In Proc. *CRYPTO 2005*.
22. U. Feige, J. Kilian, and M. Naor. A minimal model for secure computation. In *Proc.* 26*th STOC*, pp. 554–563. 1994.
23. M. J. Fischer and N. A. Lynch. A lower bound for the time to assure interactive consistency. *Information Processing Letters*, 14(4): 183–186, 1982.
24. M. Fitzi, J. A. Garay, S. Gollakota, C. P. Rangan, K. Srinathan. Round-Optimal and Efficient Verifiable Secret Sharing. *In Proc. TCC 2006*, pp. 329–342.
25. R. Gennaro, Y. Ishai, E. Kushilevitz, and T. Rabin. The Round Complexity of Verifiable Secret Sharing and Secure Multicast. In *Proc.* 33*th STOC*. 2001.
26. R. Gennaro, Y. Ishai, E. Kushilevitz, and T. Rabin. On 2-Round Secure Multiparty Computation. In *Proc. Crypto 2002*, pp. 178–193.
27. Y. Gertner, Y. Ishai, E. Kushilevitz, T. Malkin. Protecting Data Privacy in Private Information Retrieval Schemes. STOC 1998: 151–160
28. O. Goldreich. *Foundations of Cryptography: Basic Applications*. Cambridge University Press, 2004.
29. O. Goldreich, S. Micali, and A. Wigderson. How to Play Any Mental Game. In *Proc.* 19*th STOC*, pp. 218–229. 1987.
30. S. Goldwasser and Y. Lindell. Secure Multi-Party Computation without Agreement. *J. Cryptology* 18(3), pp. 247–287, 2005.
31. O. Horvitz and J. Katz. Universally-Composable Two-Party Computation in Two Rounds. In *Proc. CRYPTO 2007*, pp. 111–129.
32. Y. Ishai and E. Kushilevitz. Private simultaneous messages protocols with applications. In *ISTCS97*, pp. 174–184, 1997.
33. Y. Ishai and E. Kushilevitz. Randomizing polynomials: A new representation with applications to round-efficient secure computation. In *Proc.* 41*st FOCS*, 2000.

34. Y. Ishai and E. Kushilevitz. Perfect Constant-Round Secure Computation via Perfect Randomizing Polynomials. In *Proc. ICALP '02*.
35. Y. Ishai, M. Prabhakaran, and A. Sahai. Founding Cryptography on Oblivious Transfer - Efficiently. *In Proc. CRYPTO 2008*, pp. 572–591.
36. M. Ito, A. Saito, T. Nishizeki Secret sharing scheme realizing general access structure. Electronics and Communications in Japan, Part III: Fundamental Electronic Science, Volume 72 Issue 9, pp. 56–64.
37. S. Jarecki and V. Shmatikov. Efficient Two-Party Secure. Computation on Committed Inputs. *EUROCRYPT 2007*, pp. 97–114.
38. M. Karchmer, and A. Wigderson. On Span Programs. Proceedings of the 8th Structures in Complexity conference, pp. 102–111, 1993.
39. J. Katz and C.-Y. Koo. Round-Efficient Secure Computation in Point-to-Point Networks. *Proc. EUROCRYPT 2007*, pp. 311–328.
40. J. Katz, C.-Y. Koo, R. Kumaresan. Improving the Round Complexity of VSS in Point-to-Point Networks. *Proc. ICALP 2008*, pp. 499–510.
41. J. Katz and R. Ostrovsky. Round-Optimal Secure Two-Party Computation. *Proc. CRYPTO 2004*, pp. 335–354.
42. J. Katz, R. Ostrovsky, and A. Smith. Round Efficiency of Multi-party Computation with a Dishonest Majority. In *EUROCRYPT 2003*, pp. 578–595.
43. E. Kushilevitz, Y. Lindell, and T. Rabin. Information-theoretically secure protocols and security under composition. In *STOC 2006*, pp. 109–118. Full version: Cryptology ePrint Archive, Report 2009/630.
44. L. Lamport, R.E. Shostack, and M. Pease. The Byzantine generals problem. *ACM Trans. Prog. Lang. and Systems*, 4(3):382–401, 1982.
45. Y. Lindell. Parallel Coin-Tossing and Constant-Round Secure Two-Party Computation. In *Crypto '01*, pp. 171–189, 2001. LNCS No. 2139.
46. Y. Lindell and B. Pinkas. An efficient protocol for secure two-party computation in the presence of malicious adversaries. In *Proc. EUROCRYPT 2007*, pp. 52–78.
47. N. Lynch. *Distributed Algorithms*. Morgan Kaufman, 1996.
48. R. Pass. Bounded-concurrent secure multi-party computation with a dishonest majority. In *Proc. STOC 2004*, pp. 232–241.
49. A. Shamir How to share a secret. In Communications of the ACM 22, pp. 612–613.
50. A. Patra, A. Choudhary, T. Rabin, and C. P. Rangan. The Round Complexity of Verifiable Secret Sharing Revisited. In *Proc. CRYPTO 2009*, pp. 487–504.
51. T. Rabin and M. Ben-Or. Verifiable Secret Sharing and Multiparty Protocols with Honest Majority. In *Proc.* 21*st STOC*, pp. 73–85. 1989.
52. T. Sander, A. Young, and M. Yung. Non-Interactive CryptoComputing For NC1. In *Proc.* 40*th FOCS*, pp. 554–567. IEEE, 1999.
53. A. C-C. Yao. How to Generate and Exchange Secrets. In *Proc.* 27*th FOCS*, pp. 162–167. IEEE, 1986.