Efficient Multiparty Protocols via Log-Depth Threshold Formulae

(Extended Abstract)

Gil Cohen[1], Ivan Bjerre Damgård[2], Yuval Ishai[3], Jonas Kölker[2], Peter Bro Miltersen[2], Ran Raz[1], and Ron D. Rothblum[1]

[1] Weizmann Institute, Rehovot, Israel
{gil.cohen,ran.raz,ron.rothblum}@weizmann.ac.il
[2] Aarhus University, Aarhus, Denmark
{ivan,epona,bromille}@cs.au.dk
[3] Technion, Haifa, Israel
yuvali@cs.technion.ac.il

Abstract. We put forward a new approach for the design of efficient multiparty protocols:

1. Design a protocol π for a small number of parties (say, 3 or 4) which achieves security against a *single* corrupted party. Such protocols are typically easy to construct, as they may employ techniques that do not scale well with the number of corrupted parties.
2. Recursively compose π with itself to obtain an efficient n-party protocol which achieves security against a constant fraction of corrupted parties.

The second step of our approach combines the "player emulation" technique of Hirt and Maurer (J. Cryptology, 2000) with constructions of logarithmic-depth formulae which compute threshold functions using only constant fan-in threshold gates.

Using this approach, we simplify and improve on previous results in cryptography and distributed computing. In particular:

– We provide conceptually simple constructions of efficient protocols for Secure Multiparty Computation (MPC) in the presence of an honest majority, as well as broadcast protocols from point-to-point channels and a 2-cast primitive.
– We obtain new results on MPC over blackbox groups and other algebraic structures.

The above results rely on the following complexity-theoretic contributions, which may be of independent interest:

– We show that for every $j, k \in \mathbb{N}$ such that $m \triangleq \frac{k-1}{j-1}$ is an integer, there is an explicit (poly(n)-time) construction of a logarithmic-depth formula which computes a good approximation of an (n/m)-out-of-n threshold function using only j-out-of-k threshold gates and no constants.
– For the special case of n-bit majority from 3-bit majority gates, a non-explicit construction follows from the work of Valiant (J. Algorithms, 1984). For this special case, we provide an explicit construction with a better approximation than for the general threshold case, and also an *exact* explicit construction based on standard complexity-theoretic or cryptographic assumptions.

1 Introduction

Secure multiparty computation (MPC) enables a set of parties to jointly accomplish some distributed computational task, while maintaining the secrecy of the inputs and the correctness of the outputs in the presence of coalitions of dishonest parties. Originating from the seminal works of [41,25,4,10], secure MPC has been the subject of an enormous body of work.

Despite this body of work, MPC protocols remain quite complicated and their security is difficult to prove. In this work we propose a new general approach to the construction of efficient[1] multiparty protocols in the presence of an honest majority. This approach enables us to obtain conceptually simple derivations of known feasibility results (or slightly weaker variants of such results), and also to obtain new results.

Our approach is inspired by and builds on the "player emulation" technique of Hirt and Maurer [28], who obtain secure MPC protocols by reducing the construction of an n-party protocol to the task of constructing a protocol π for a constant (e.g., three or four) number of parties. The motivation of [28] was to obtain n-party protocols that are secure with respect to general (non-threshold) adversary structures. A disadvantage of their n-party protocols is that their complexity grows exponentially with n. This seems inevitable when considering arbitrary adversary structures.

Our motivation is very different: We would like to use the atomic protocol π for constructing *efficient* n-party protocols in the traditional MPC setting of *threshold* adversary structures. Since π only involves a small number of parties, its design may employ simpler techniques that do not scale well with the number of corrupted parties. Thus, our goal is to simplify the design of efficient n-party protocols by reducing it to the design of a simpler atomic protocol π.

To make the approach of [28] scale with the number of parties, we introduce a new complexity-theoretic primitive: a logarithmic-depth formula[2] which is composed only of constant-size threshold gates and computes an n-input threshold function. The problem of constructing such formulae is closely related to a classical problem in complexity theory. In this work we also make a contribution to this complexity-theoretic problem, which may be of independent interest.

In addition to providing conceptually simple protocols, our approach is very general and can be applied in a variety of settings and models. In contrast to most traditional MPC protocols, it is not tied to some underlying algebraic structure. We demonstrate this generality by obtaining new results on MPC over black-box groups and other algebraic structures, improving on previous results from the literature.

[1] Here and throughout this work, by "efficient" we mean polynomial-time in the number of parties and the input size.

[2] A formula is a circuit with fan-out 1. A logarithmic-depth formula (more precisely, infinite family of formulas) is one whose depth is $O(\log n)$, where n is the number of inputs. Throughout this paper we consider only *monotone* formulas without negations or constants.

Before proceeding to describe the details of our approach, we note that the goal of designing MPC protocols whose complexity grows (only) polynomially with the number of parties also has relevance to *two-party* cryptography. Indeed, there are general techniques for applying MPC protocols with security in the presence of an honest majority (where the number of parties grows with the security parameter) towards two-party tasks such as zero-knowledge proofs and secure two-party computation [30,31].

1.1 Our Approach

In the following, for simplicity, we consider the case of perfect security against a *passive* adversary. In this setting, parties are honest but curious. That is, they follow the protocol but may attempt to learn secret information based on what they see. We note that, in contrast to the norm, the extension of this approach to the case of an active adversary is relatively straightforward.

We first give an overview of the player emulation technique of Hirt and Maurer [28] and then proceed to describe how we overcome the exponential blow-up incurred by [28] in the case of threshold adversary structures.

Recall that security of MPC protocols is defined by comparing a real protocol to an ideal protocol, in which, in addition to the parties involved in the computation, there is a trusted party. A protocol is deemed secure if for every adversary in the real protocol controlling a subset of the parties, there is an equivalent adversary controlling the same subset in the ideal protocol.

The technique from [28] is to reduce the design of n-party protocols to the design of protocols that support only 3 parties (the minimal number of parties for perfect security in the passive security model).

We proceed to present an informal description of the reduction. Indeed, suppose that the 3-party case has been solved. That is, for every computational task involving three parties there exists a secure protocol that securely implements this task when at most one of the parties is passively corrupted.[3] We describe how to use this protocol to securely implement computational tasks using a larger number of parties.

Consider n parties that wish to securely accomplish some joint computational task. It is best to think of this task as being specified by an ideal protocol π_0 which involves, in addition to the n parties, a trusted party τ. The ideal protocol is secure (by definition) even if the adversary controls any subset of the parties that does not contain τ.

Consider a new protocol π_1 that involves the n original parties but where we replace the trusted party τ with three new virtual parties v_1, v_2, v_3. Since in π_0, the trusted party τ is just involved in a computational task, we can use the given 3-party protocol to simulate τ using v_1, v_2, v_3. When is the new protocol π_1 secure? Since π_0 was only insecure whenever the adversary controlled τ and

[3] Since we deal with *perfect* security, the size of the secure protocol depends only on the size of the original protocol. In particular, any constant size protocol can be implemented securely in constant size.

since the 3-party protocol is secure as long as the adversary controls at most one of the virtual parties, π_1 is secure as long as the adversary does not control two or more of the virtual parties.

We continue this process by designing a new protocol π_2 in which the virtual party v_1 is itself simulated by three new virtual parties w_1, w_2, w_3. Since π_1 is only insecure whenever the adversary controls more than one of v_1, v_2, v_3 and since the protocol for emulating v_1 is secure when at most one of w_1, w_2, w_3 is controlled by the adversary, π_2 is secure as long as the adversary does not control either v_2 and v_3 or one of v_2, v_3 and two or more of w_1, w_2, w_3.

We continue in this process simulating virtual parties by more virtual parties. The sets of corrupted parties against which the resulting protocol is secure can be described by looking at a formula composed of 3-input majority gates which we denote by Maj_3. Each wire represents a virtual party. The protocol π_1 can be represented by a simple formula F_1 consisting of a single Maj_3 gate where the three input wires correspond to the virtual parties v_1, v_2, v_3 and the output wire corresponds to τ. We assign to each input wire corresponding to an honest party a value of 0 and a value of 1 to those corresponding to dishonest parties. It can be easily verified that the protocol is secure whenever the formula F_1 evaluates to 0.

Similarly, the protocol π_2 can be represented by a formula F_2 which is constructed from F_1 by connecting the input wire corresponding to v_1 with an additional Maj_3 gate with three new input wires (corresponding to w_1, w_2, w_3). It is easy to verify that the new protocol is secure whenever the formula evaluates to 0.

Suppose that we continue on like this but instead of arbitrarily choosing which virtual party to simulate, we choose it according to some formula F, composed only of Maj_3 gates.[4] Once we reach the input layer of the formula, we associate each input variable to a real party and every remaining virtual party is simulated by the real party associated with the corresponding input wire.

As above, the protocol is secure against every set T of parties on which the formula F evaluates to 0. (Here and in the following we associate a set T with its characteristic vector χ_T.) Thus, to obtain a protocol that is secure for a particular adversary structure, it suffices to provide a formula that evaluates to 0 on all sets in the structure. Since, in contrast to [28], our goal is merely to obtain security in the presence of an honest majority, we need only to construct a formula that computes the majority function (using only Maj_3 gates and no constants).

Such a formula was implicitly constructed by Hirt and Maurer [28] for general Q_2 functions[5] and in particular for majority. Unfortunately, the formula of [28] has linear depth. This yields a protocol whose complexity grows exponentially with the number of parties, since when traversing the formula we increased the

[4] Actually, [28] do not present their construction in the terminology of Maj_3 formulae; we use this presentation since it is more intuitive and is better suited for our purposes.

[5] A monotone function $f : \{0,1\}^n \to \{0,1\}$ is said to be of type Q_d if $f(x_1) = f(x_2) = \ldots = f(x_d) = 0$ implies that $x_1 \vee x_2 \vee \ldots \vee x_d \neq 1^n$.

complexity of the protocol by a constant multiplicative factor (corresponding to the number of operations in the 3-party protocol) at every layer.

To overcome the exponential blowup, we replace the formula of [28] by a *logarithmic-depth* formula (which computes the majority function using only Maj_3 gates). Using the formula-based protocol described above, the logarithmic depth results in an efficient protocol, namely one whose complexity only grows polynomially with the number of parties. In Section 1.2 we describe the construction of a good "approximation" of such a formula as well as exact constructions under standard complexity-theoretic assumptions.

This approach is indeed very general and can be used in different models of secure MPC. For example, it can be used to obtain both passive security as outlined above and active security by using an underlying 4-party protocol that is secure against one active party and a log-depth threshold formula composed of two-out-of-four threshold gates (denoted by Th_2^4) which we also construct (see Section 1.2).

In fact, this reduction gives us a "cookbook" for designing secure multiparty protocols. The first step is to design a protocol for a constant number of parties that is secure against one dishonest party and the second step is to use a logarithmic-depth threshold from thresholds formula to obtain an efficient multiparty protocol that is secure against a constant fraction of corrupted parties.

We demonstrate the generality of this approach by deriving protocols in both passive and active settings and in different MPC models which differ in the type of underlying algebraic structure, including models for which no protocols were known. We also obtain conceptually simple protocols for classical problems in distributed computing such as broadcast protocols.

Simplified feasibility results. The classical results of Ben-Or et al. [4] and Chaum et al. [10] allow n parties to evaluate an arbitrary function, using secure point-to-point channels, with perfect security against $t < n/2$ passively corrupted parties or $t < n/3$ actively corrupted parties. We can derive conceptually simpler variants of these results by applying our approach with π being a 3-party or 4-party instance of the simple MPC protocol of Maurer [36]. On the one hand our results are slightly weaker because they either need the threshold t to be slightly suboptimal or alternatively require (standard) complexity theoretic assumptions to construct an appropriate formula for implementing the protocol. It is instructive to note that the complexity of Maurer's protocol grows exponentially with the number of parties. Our approach makes this a non-issue, as we only use the protocol from [36] with a constant number of parties.[6]

MPC over blackbox algebraic structures. There has been a considerable amount of work on implementing MPC protocols for computations over different algebraic

[6] While in the present work we apply our approach only to perfectly secure protocols, one could apply a similar technique to derive the result of Rabin and Ben-Or [38], namely a statistically secure protocol which tolerates $t < n/2$ actively corrupted parties.

structures such as fields, rings, and groups. Algebraic computations arise in many application scenarios. While it is possible in principle to emulate each algebraic operation by a sequence of boolean operations, this is inefficient both in theory and in practice. In particular, the communication complexity of the resulting protocols grows with the computational complexity of the algebraic operations rather than just with the bit-length of the inputs and outputs. This overhead can be avoided by designing protocols which make a *blackbox* (i.e., oracle) use of the underlying structure. The advantage of such protocols is that their communication complexity and the number of algebraic operations they employ are independent of the complexity of the structure.

MPC over rings and k-linear maps. The work of Cramer *et al.* [13] shows how to efficiently implement secure MPC over blackbox *rings*. We obtain a simpler derivation of such a protocol by noting that the simple protocol of Maurer [36] directly generalizes to work over a blackbox ring. As before, one could not apply this protocol directly because its complexity is exponential in the number of parties. We show how to use a similar approach for obtaining the first blackbox feasibility results for MPC over *k-linear maps*.

MPC over groups. The problem of MPC over blackbox *groups* was introduced by Desmedt *et al.* [17] and further studied in [39,16,15]. To apply our approach in the group model, we need to specify the atomic protocol π that we use. For the case of passive security, we directly construct a simple 3-party protocol that has security against one corrupted party. This protocol is loosely based on a protocol by Feige *et al.* [19] and considerably simplifies the 3-party instance of a general result from [16].

In the active security model, we rely on the recent work of [15] who obtain the first MPC protocols with active security in the group model. The complexity of the protocol of [15] grows exponentially with the number of parties. However, we only need to employ the [15] protocol for four parties and so we do not suffer the exponential blowup. Thus, we settle the main problem left open in [15] by applying our technique to an instance of their results.

We also obtain the first *two-party* MPC protocols over blackbox groups. In the passive corruption model, we combine a group product randomization technique due to Kilian [32] with a "subset sum" based statistical secret sharing of group elements. We then get security against active corruptions by combining this two-party protocol with our efficient n-party protocol for the active model via the IPS compiler [31].

Broadcast. Broadcast is one of the most basic problems in distributed computing. Recall that in a broadcast protocol a broadcaster wants to send a message to all other parties. A broadcast protocol should end with all parties holding the same value, even if some of the parties, possibly including the broadcaster, behave adversarially. Obtaining efficient broadcast protocols is a highly nontrivial task [37,18,22]. Our generic approach for MPC protocols can be used to directly construct simple broadcast protocols for $t < n/3$ corrupted parties. We also get a simplified proof of a result of Fitzi and Maurer [21], showing that an ideal primitive allowing broadcast for 3 parties (so-called 2-cast) implies broadcast with $t < n/2$ corrupted parties. Our proof technique also yields broadcast for the more general case of Q_2 adversaries which was previously an open problem.

1.2 Threshold Formulae from Threshold Gates

Motivated by the above applications to MPC, we consider the problem of constructing a logarithmic-depth threshold formula from threshold gates. Before discussing the general problem, we first discuss the special case of constructing a logarithmic depth formula composed of Maj_3 gates that computes the majority function. Note that this is exactly the type of formula required in the setting of *passive* MPC security.

Majority from Majorities. A closely related problem was considered by Valiant [40] who proved the existence of a logarithmic-depth monotone formula that computes the majority function where the formula uses And and Or gates, both of fan-in 2. As noted independently by several authors [6,26,42,24], a slight modification of Valiant's argument shows the existence of a logarithmic-depth formula composed of Maj_3 gates that computes the majority function.

Valiant's proof is based on the probabilistic method and is non-constructive. Namely, the proof only assures us of the existence of a formula with the above properties, but does not hint on how to find it efficiently. Motivated by the applications presented in Section 1.1, we ask whether Valiant's proof can be derandomized using only Maj_3 gates and no constants.[7] We raise the following conjecture:

Conjecture 1 (Majority from Majorities). There exists an algorithm A that given an odd integer n as input, runs in poly(n)-time and generates a formula F on n inputs, with the following properties:

– F consists only of Maj_3 gates and no constants.
– $\mathsf{depth}(F) = O(\log n)$.
– F computes the majority function on n inputs.

A derandomization for Valiant's proof for formulas over And and Or gates follows from the seminal paper of Ajtai, Komlós and Szemerédi [1], though the latter does not seem to imply a derandomization in the context of Maj_3 gates, where constants are not allowed.[8]

In this paper we make a significant progress towards proving Conjecture 1. In particular, we prove that relaxed variants of the conjecture hold. In addition, we show that the conjecture follows from standard complexity assumptions, namely,

[7] We cannot allow the use of the constant 0, as this would correspond to assuming parties to be incorruptible. The use of the constant 1 alone is not helpful in our context.

[8] Note that And and Or gates can be implemented using Maj_3 gates and constants.

$\mathsf{E} \triangleq \mathsf{DTIME}(2^{O(n)})$ does not have $2^{\varepsilon n}$-size circuits for some constant $\varepsilon > 0$. Note that the latter follows from the existence of exponentially hard one-way functions.[9] See details in Section 2.

Threshold Formulae from Threshold Gates. Motivated by applications to the active MPC setting, and being a natural complexity-theoretic problem on its own, we initiate the study of a generalization of the majority from majorities problem, which we call *the threshold from thresholds problem.*

For integers $2 \leq j \leq k$, define the threshold function $\mathsf{Th}_j^k : \{0,1\}^k \to \{0,1\}$ as follows. $\mathsf{Th}_j^k(x) = 1$ if and only if the Hamming weight of x is at least j. Note that $\mathsf{Maj}_3 = \mathsf{Th}_2^3$.

Unlike the majority from majorities problem, it is not a priori clear what threshold function, if any, can be computed by a log-depth formula composed only of Th_j^k gates, even if no explicit construction is required.

We make significant progress also on this question. Roughly speaking, we provide an explicit construction of a logarithmic depth formula composed solely of Th_j^k gates, that well approximates $\mathsf{Th}_{n/m}^n$, where $m = \frac{k-1}{j-1}$. For further details, see Section 2.3.

Organization. In Section 2 we state our results and in Section 3 we present the proof techniques of the complexity-theoretic part. For an overview of the applications to cryptography and distributed computing, as well as formal statements and full proofs of our results, see the full version.

2 Our Results

We first describe the applications of our approach in cryptography and distributed computing, and then proceed to the complexity-theoretic results.

2.1 Cryptographic Results

We start by stating known results that we rederive using our approach, and later state our new results.

In the passive Ring-MPC model, we get the following results.

– If the majority from majorities conjecture (Conjecture 1) holds then we obtain an explicit MPC protocol that has optimal security in the passive model. That is, it is secure as long as at most a $\frac{1}{2} - \Omega(\frac{1}{n})$ fraction of the n parties (more precisely, $t < n/2$) are passively corrupted.
  As noted above and stated formally in Theorem 3, Conjecture 1 follows from widely-believed conjectures in complexity theory and cryptography.

[9] We find it curious that *perfectly secure* MPC results are based on the existence of (sufficiently strong) one-way functions.

- An unconditional explicit and close to optimal protocol in the passive model in which the fraction of dishonest parties is at most $\frac{1}{2} - 2^{-O(\sqrt{\log n})}$ out of the n parties (in contrast to the optimal threshold of $\frac{1}{2} - \Omega(\frac{1}{n})$).
- A randomized construction of an optimal protocol in the passive model. By randomized construction we mean that the protocol is constructed by a randomized algorithm which may fail with negligible (undetectable) probability, but otherwise outputs the description of a perfect protocol.

We obtain the following result in the *active* Ring-MPC model.

- An explicit but non-optimal protocol that is secure against any *active* adversary that controls at most a $\frac{1}{3} - \Omega(\frac{1}{\sqrt{\log n}})$ fraction of the n parties (in contrast to the optimal bound of $\frac{1}{3} - \Omega(\frac{1}{n})$).

Next we state our new results in the blackbox group model, introduced by Desmedt *et al.* [17,16]. In this model the function computed by the protocol is specified by an arithmetic circuit over a (possibly non-Abelian) group, and the parties are restricted to making blackbox access to the group. (This includes oracle access to the group operation, taking inverses, and sampling random group elements.) In particular, the number of group operations performed by the protocol should not depend on the structure of the group or the complexity of implementing a group operation using, say, a Boolean circuit.

- **Group-MPC, passive:** The best explicit protocol of [16] offers perfect security against a $\frac{1}{n^\epsilon}$ fraction of passively corrupted parties, for any constant $\epsilon > 0$, where n is the total number of parties.
  We improve upon the latter by constructing an explicit protocol that has perfect security against an (almost optimal) $\frac{1}{2} - 2^{-O(\sqrt{\log n})}$ fraction of passively corrupted parties. Alternatively, we get an optimal bound of $\frac{1}{2} - \Omega(\frac{1}{n})$ assuming the majority from majorities conjecture, via a non-uniform construction, or under standard derandomization or cryptographic assumptions.
- **Group-MPC, active:** In a recent work, Desmedt *et al.* [15] constructed a secure MPC protocol in the group model with security against an *active* adversary. However, their result only gives a protocol whose complexity depends exponentially on the number of parties, regardless of the corruption threshold.
  We construct an *efficient* secure MPC protocol in the group model where an active adversary can control (an almost optimal) $\frac{1}{3} - \Omega(\frac{1}{\sqrt{\log n}})$ fraction of the n parties.
- **Secure two-party computation over groups:** We construct the first secure *two-party* protocols over blackbox groups. Our protocols offer statistical security against active corruptions (assuming an oblivious transfer oracle) and rely on the afforementioned n-party protocols over black-box groups.

Finally, our protocols for the Ring-MPC model described above can be generalized to yield the following new result for MPC over k-linear maps.

– **MPC over k-linear maps:** We show that, for any constant k and any basis B of k-linear maps over finite Abelian groups, there are efficient MPC protocols for computing circuits over B which only make blackbox access to functions in B and group operations. This generalizes previous results for MPC over blackbox rings [13], which follow from the case $k = 2$, and can potentially be useful in cryptographic applications that involve complex bilinear or k-linear maps. These protocols are perfectly secure against a $\frac{1}{k} - \Omega(\frac{1}{\sqrt{\log n}})$ fraction of passively corrupted parties or a $\frac{1}{k+1} - \Omega(\frac{1}{\sqrt{\log n}})$ fraction of actively corrupted parties.

2.2 Distributed Computing Results

Broadcast. It is well known that broadcast can be implemented over point-to-point channels if and only if less than a third of the parties are actively corrupted [37,18] or, more generally, if and only if no three of the subsets the adversary may corrupt cover the entire set of parties [28,20], a so called Q_3-adversary.

In this paper we show that a trivial broadcast protocol for 4 parties where one is actively corrupted easily implies the result of [20] using existing constructions of (super-logarithmic depth) formulae. Substituting instead our own logarithmic depth formula constructions implies a simple polynomial-time broadcast protocol for less than $n(\frac{1}{3} - \Omega(\frac{1}{\sqrt{\log n}}))$ corrupted parties.

Broadcast from 2-cast. In [21], Fitzi and Maurer identify a minimal primitive that allows to improve the $\frac{n}{3}$ corruption threshold: if we are given the ability to broadcast among any subset of 3 parties for free, a so-called *2-cast* primitive, then broadcast becomes possible when less than $\frac{n}{2}$ parties are corrupted. It is natural to ask whether 2-cast also implies broadcast secure against general Q_2-adversaries (where no two corruptible subsets cover the entire set of parties). This problem was previously open.

We apply our approach to construct broadcast protocols based on a 2-cast primitive. Together with existing constructions of (super-logarithmic depth) formulae composed of Maj_3-gates, this immediately implies a construction of broadcast from 2-cast for every Q_2-adversary, resolving the above problem. Substituting instead our logarithmic-depth formula constructions, we get a simplified derivation of polynomial-time protocols for the case of an honest majority considered in [21]. We do not know if the formula based approach also implies the results in [11], which consider generalizations of the 2-cast primitive.

2.3 Complexity-Theoretic Results

In this section we describe our results on constructing threshold formulae from threshold gates. For the special case of computing majority from Maj_3 gates we obtain stronger results which we state first.

Majority from Majorities Our first complexity-theoretic result shows that given a small promise on the bias of the input (defined as the difference between the normalized Hamming weight and 1/2), Conjecture 1 holds.

Theorem 2. *There exists an algorithm A that given an odd integer n as input, runs in* $\text{poly}(n)$*-time and computes a formula F on n inputs, with the following properties:*

- *F consists only of Maj_3 gates and no constants.*
- $\mathsf{depth}(F) = O(\log n)$.
- *$\forall x \in \{0,1\}^n$ such that $\mathsf{bias}(x) \geq 2^{-O(\sqrt{\log n})}$ it holds that $F(x) = \mathsf{Maj}(x)$.*

Our second result shows that under standard complexity hardness assumptions, Conjecture 1 holds.

Theorem 3. *If there exists an $\varepsilon > 0$ such that $\mathsf{E} \triangleq \mathsf{DTIME}(2^{O(n)})$ does not have $2^{\varepsilon n}$-size circuits then Conjecture 1 holds. In particular, if there exist exponentially hard one-way functions then Conjecture 1 holds.*[10]

In fact, the proof of Theorem 3 explicitly presents an algorithm for constructing a formula as in Conjecture 1 given the truth table of any function in E, on a suitable number of inputs, that cannot be computed by $2^{\varepsilon n}$-size circuits.

Thresholds Formulae from Threshold Gates

Lemma 4. *There exists an algorithm A that given $t, j, k \in \mathbb{N}$ as input, where j, k are constants in t such that $j \geq 2$ and $k \geq 2j - 1$,*[11] *runs in* $\exp(t)$*-time and generates a formula F with the following properties:*

- *F has $mt + 1$ inputs, where $m = \left\lfloor \frac{k-1}{j-1} \right\rfloor$.*
- *F consists only of Th_j^k gates and no constants.*
- $\mathsf{depth}(F) = O(t)$.
- *$\forall x \in \{0,1\}^{mt+1}$ it holds that $F(x) = \mathsf{Th}_{t+1}^{mt+1}(x)$.*

Lemma 4 generalizes results of [2,28,3], who proved it for particular values of j and k, and uses a similar technique. We note that the depth of the formula generated in Lemma 4 is linear, which is too large for our applications. Nevertheless, the following theorem, which uses Lemma 4 as a building block, shows that a formula with *logarithmic depth* can be generated efficiently assuming a sufficient "bias" on the input.

[10] A one-way function f is *exponentially hard* if there exists an $\varepsilon > 0$ such that every family of $2^{\varepsilon n}$-size circuits can invert f with only $2^{-\varepsilon n}$ probability. If there exists such a function f, then the language $\mathcal{L}_f$ is in E but does not have $2^{\varepsilon n}$-size circuits, where $\mathcal{L}_f = \{(y, x', 1^n) : y$ has a preimage of length n under f which starts with $x'\}$.

[11] Throughout the paper we assume, without loss of generality, that $k \geq 2j - 1$. The complementary case can be reduced to this one by using Th_{k-j+1}^k gates and interpreting 0 as 1 and vice versa.

Theorem 5. *There exists an algorithm A that given $n, j, k \in \mathbb{N}$ as input, where j, k are constants in n such that $j \geq 2$ and $k \geq 2j - 1$, runs in $\mathrm{poly}(n)$-time and generates a formula F on n inputs, with the following properties:*

- *F consists only of Th_j^k gates and no constants.*
- *$\mathsf{depth}(F) = O(\log n)$.*
- *$\forall x \in \{0,1\}^n$ with normalized Hamming weight at least $\frac{1}{m} + \Omega(\frac{1}{\sqrt{\log n}})$, it holds that $F(x) = 1$, where $m = \left\lfloor \frac{k-1}{j-1} \right\rfloor$.*
- *$\forall x \in \{0,1\}^n$ with normalized Hamming weight at most $\frac{1}{m} - \Omega(\frac{1}{\sqrt{\log n}})$, it holds that $F(x) = 0$.*

Note that Theorem 2 is not a special case of Theorem 5 (with $j = 2, k = 3$) as the required promise on the bias in Theorem 2 is exponentially smaller than that in Theorem 5.

We do not know whether an analog of Conjecture 1 is plausible for the threshold from thresholds problem, even without the time-efficiency requirement. Theorem 5 might serve as evidence for the affirmative. However, the probabilistic argument used in the majority from majorities problem (see, e.g., [24]) breaks for this more general case. We consider this to be an interesting open problem for future research.

3 Proof Overview of Complexity-Theoretic Results

In this section we give an overview of our complexity-theoretic constructions. For simplicity, we start by giving an overview of our construction of a logarithmic-depth formula composed of Maj_3 gates, and no constants, that computes the majority function for inputs with constant bias. That is, we informally describe an efficient algorithm that given n, ε as inputs, where $\varepsilon > 0$ is constant in n, outputs a logarithmic-depth formula with n inputs which computes the majority function correctly on inputs with bias at least ε. It is not hard to see that it is enough to construct a logarithmic-depth *circuit*, since such a circuit can be efficiently converted to an equivalent logarithmic-depth formula.

To this end, we design an algorithm called $\mathsf{ShrinkerGenerator}$ that given n, ε as inputs, generates a *constant-depth* circuit $\mathsf{Shrinker}$ with n inputs and $\frac{n}{2}$ outputs, composed of Maj_3 gates and no constants, such that

$$\forall x \in \{0,1\}^n \quad \mathsf{bias}(x) \geq \varepsilon \implies \mathsf{bias}(\mathsf{Shrinker}(x)) \geq \varepsilon.$$

Thus, $\mathsf{Shrinker}$ shrinks the number of variables to half while maintaining the bias, assuming the input has a sufficiently large bias. By repeatedly calling $\mathsf{ShrinkerGenerator}$ on inputs $n, \frac{n}{2}, \frac{n}{4}, \ldots, 2$ (with the same ε) and concatenating the resulting circuits, one gets a logarithmic-depth circuit that computes the majority function assuming the input has large enough bias.

A key object we use in the design of $\mathsf{ShrinkerGenerator}$ is a Boolean sampler. Roughly speaking, a Boolean sampler is a randomized algorithm which on input $x \in \{0,1\}^n$ approximates the Hamming weight of x by reading only a small number of the bits of x. More precisely, a (d, ε, δ)-Boolean sampler is a randomized algorithm that on input $x \in \{0,1\}^n$ with normalized Hamming weight ω, samples at most d bits of x, and outputs $\beta \in [0,1]$ such that $\mathsf{Pr}[|\omega - \beta| \geq \varepsilon] \leq \delta$.

We will use a special type of samplers which take their samples in a non-adaptive fashion, and their output is simply the average of the sampled bits. For any $\varepsilon, \delta > 0$ there exist efficient (d, ε, δ)-Boolean samplers, with $d = O(\varepsilon^{-2} \cdot \delta^{-1})$, that on inputs of length n use only $\log n$ random bits.

Because such a sampler is non-adaptive and simply outputs the average of the sampled bits, it can be represented as a bipartite graph $G = (L, R, E)$, with $|L| = |R| = n$. For an input $x \in \{0,1\}^n$, the i'th vertex in L is labeled with the i'th bit of x. Each vertex in R represents one of the possible $\log n$ bit random strings used by the sampler. Each right vertex r is connected to the d left-vertices that are sampled by the algorithm when r is used as the random string.

The algorithm $\mathsf{ShrinkerGenerator}$ on inputs n, ε starts by constructing a graph G that represents a $(d, \frac{\varepsilon}{2}, \frac{1}{8})$-Boolean sampler, with $d = \text{poly}(\frac{1}{\varepsilon}) = O(1)$. It then arbitrarily chooses half of the right vertices in G and discards the rest. This gives a bipartite graph $G' = (L', R', E')$ with $|L'| = n$, $|R'| = \frac{n}{2}$ and constant right-degree d. The circuit $\mathsf{Shrinker}$ that the algorithm $\mathsf{ShrinkerGenerator}$ outputs is given by placing a circuit that computes the majority function on d inputs for every right vertex. The inputs of this majority circuit are the neighbors of the respective right vertex. Note that as d is constant, a constant-depth circuit that computes the majority function on d inputs can be found in constant time.

As for the correctness of the construction, assume now that $x \in \{0,1\}^n$ has some constant bias ε and, without loss of generality, assume that the bias is towards 1 (i.e., $\mathsf{wt}(x) \geq (\frac{1}{2} + \varepsilon)n$). Then, by the guarantee of the sampler, for all but $\frac{1}{8}$ of the right vertices in the original graph G, the fraction of neighbors with label 1 of a right vertex is at least $\frac{1}{2} + \varepsilon - \frac{\varepsilon}{2} > \frac{1}{2}$. Thus, all but $\frac{1}{8}$ of the (constant-size) majority circuits located in R output 1. Hence, the fraction of majority circuits that output 0 in R' is at most $\frac{n/8}{n/2} = \frac{1}{4} \leq \frac{1}{2} - \varepsilon$, as desired.

3.1 Supporting Sub-Constant Bias

For sub-constant ε, the sampler technique described above is wasteful, as it requires us to use a sequence of $O(\log n)$ layers with fan-in $O(\varepsilon^{-2})$. For sub-constant ε, this results in a circuit with a super-logarithmic depth. However, we observe that one layer of fan-in $O(\varepsilon^{-2})$ circuits is enough to amplify the bias from ε to 0.4 (rather than just keep the bias at ε). This reduces us to the constant bias case, which can be solved as above with an additional $O(\log n)$-depth.

Thus, in order to obtain an $O(\log n)$-depth circuit on n inputs, that computes majority correctly for inputs with bias at least ε, it is enough to construct an $O(\log n)$-depth circuit with $O(\varepsilon^{-2})$ inputs that computes majority correctly on all inputs.

Using a naive brute-force algorithm, one can efficiently find an optimal-depth circuit on roughly $\log n$ inputs that computes majority. By plugging this circuit into the above scheme, one immediately gets an $O(\log n)$-depth circuit that computes majority on n inputs with bias roughly $\varepsilon = \Omega(\frac{1}{\sqrt{\log n}})$.

We improve on this by using an additional derandomization idea. Specifically, we construct an $O(\log n)$-depth circuit on $2^{O(\sqrt{\log n})}$ inputs, that computes majority (under no assumption on the bias). Thus, we obtain an explicit construction of a circuit that computes majority assuming the bias is at least $\varepsilon = 2^{-O(\sqrt{\log n})}$.

We first describe a randomized construction of an $O(\log m)$-depth circuit on m inputs for majority, where m is set, in hindsight, to $2^{O(\sqrt{\log n})}$. Our construction only uses $O(\log^2 m)$ random bits (compared to $\mathrm{poly}(m)$ random bits used in Valiant's construction). We then show how to derandomize this construction.

Our randomized construction works as follows. Consider an input $x \in \{0,1\}^m$ with bias ε. Suppose that we sample uniformly and independently at random 3 bits of x and compute their majority. It is shown in [24] that the majority's bias is at least 1.2ε (as long as ε is not too large).

Thus, by placing m majority gates of fan-in 3, and selecting their inputs from x uniformly and independently at random, the output of the m majority gates will have bias of at least 1.1ϵ with overwhelming probability. By composing $O(\log(1/\varepsilon))$ such layers, we can amplify the bias to a constant. Note that this construction uses $O(m \cdot \log m \cdot \log(1/\varepsilon))$ random bits.

To save on the number of random bits used (which is essential for the derandomization step), instead of sampling the inputs of each one of the m gates uniformly at random, we choose them in each layer using a 6-wise independent hash function. While 3-wise independence suffices for the expectation of the bias to be as before, the 6-wise independence guarantees that the outputs of the majority gates in each layer are pairwise independent. Using tail inequalities we show that, with probability $1 - o(1)$, the bias increases in each layer as before.

By composing $O(\log(1/\varepsilon))$ such layers, each of which requires $O(\log m)$ random bits, we obtain a circuit as desired. The total number of random bits used is $O(\log(m) \cdot \log(1/\varepsilon))$, which is bounded by $O(\log^2 m)$. We derandomize the construction by placing all $2^{O(\log^2 m)}$ majority circuits that can be output by the randomized construction and taking the majority vote of these circuits.

Since we have a guarantee that almost all (a $1 - o(1)$ fraction) of the circuits correctly compute majority, it is enough to compute the majority vote at the end using a circuit with $2^{O(\log^2 m)}$ inputs that works for, say, constant bias. Such a circuit, with depth $O(\log^2 m)$, can be constructed in time $2^{O(\log^2 m)}$ by the constant-bias scheme described earlier.

As we set $m = 2^{O(\sqrt{\log n})}$, we get a $\mathrm{poly}(n)$-time uniform construction of an $O(\log n)$-depth circuit on $2^{O(\sqrt{\log n})}$ inputs that computes majority correctly on all inputs. This circuit is then used in the scheme described above.

Threshold formulae from thresholds gates. The scheme described above works also in the more general setting of threshold from thresholds. Indeed, in the paper we present the scheme in the general setting. To apply the scheme in the thresholds setting, one needs to construct a small circuit that computes the required threshold formula, to be used by ShrinkerGenerator. We accomplish this by extending results of [2,28,3].

4 The Player Emulation Technique

The formulas obtained in the previous section can be used to construct efficient multiparty protocols via the "player emulation" technique from [28]. Variants of this technique, also referred to as player *virtualization* or *simulation*, were used for different purposes in several other works (e.g. [5,9,27,14,31,35,34]). While implementing player emulation in the passive security model is quite straightforward, in the active security model it requires more care. In the following we give more details on the implementation of this technique.

Recall that in a single player emulation step, the role of a party τ participating in a protocol Π is replaced by a secure protocol π which involves a small set of parties $v_1, \ldots, v_k$, along with the parties of Π. We will typically let $k = 3$ (resp., $k = 4$) in the case of security against a passive (resp., active) adversary, and let π be a protocol which remains secure as long as at most one of the emulating parties v_i is corrupted. Furthermore, the total computational complexity of all parties in π (which is typically cast in some algebraic computation model) is only bigger by a constant factor than that of the emulated party τ in Π. As explained in the Introduction, a logarithmic-depth threshold formula defines a sequence of such player emulation steps which result in transforming an atomic protocol π for a constant number of parties into an efficient n-party protocol which tolerates an optimal or near-optimal fraction of corrupted parties.

The application of the player emulation technique in [28] is formulated in a specialized framework for secure MPC and is restricted to the protocol compiler of BGW [4].[12] However, the technique is quite insensitive to many of these details and can be applied with other protocols and notions of security from the literature.

A conceptually simple way for implementing a player emulation step is by viewing the role of τ in Π as a *reactive* ideal functionality, which interacts with the parties in Π (receiving incoming messages as inputs and delivering outgoing messages as outputs), and maintains a state information during this interaction. The protocol π emulating τ then needs to realize the corresponding functionality using the emulating parties v_i instead of τ. Note that protocol π does not only involve the players emulating τ. It also specifies how players communicating with τ should translate their messages into whatever format π uses[13].

The protocol π can satisfy any composable notion of security that applies to reactive functionalities, namely one which ensures that π can be securely used as a substitute for τ in an arbitrary execution environment if at most a single v_i

[12] Since the atomic protocols π we employ in this work all have a similar high-level structure to the BGW protocol, they can be used within the framework of [28] in a similar way.

[13] Alternatively, if the communication channels are modeled as an ideal functionality, one can extend the definition of this functionality so it will do the translation, and then in a final step implement the translation. This leads in some cases to a slightly simpler protocol in the end.

is corrupted. The protocols π we use in this work all satisfy the standard notion of UC-security from [8], which suffices for this purpose.[14]

Alternatively, it is possible to implement a player emulation step by only relying on protocols for secure function evaluation which satisfy the standard definitions of standalone security [7,23]. The idea is to first ensure that only a single message is sent in each round of Π, and then implement a round in which τ interacts with party P by a protocol involving P and the emulating parties v_i. The functionality realized by such a protocol is determined by the choice of a concrete (robust) secret sharing scheme which is used to distribute the state of τ between the emulating parties.

Acknowledgements. Ishai was supported by the European Research Council as part of the ERC project CaC (grant 259426). Damgaård, Kölker and Miltersen acknowledge support from the Danish National Research Foundation and The National Science Foundation of China (under the grant 61061130540) for the Sino-Danish Center for the Theory of Interactive Computation, within which part of this work was performed; and also from the CFEM research center (supported by the Danish Strategic Research Council) within which part of this work was performed. Cohen, Raz and Rothblum were supported by ISF (Israel Science Foundation) grants and by the I-CORE Program of the Planning and Budgeting Committee.

References

1. M. Ajtai, J. Komlós, and E. Szemerédi. An o(n log n) sorting network. In *STOC*, pages 1–9, 1983.
2. S. Akers and T. Robbins. Logical design with three-input majority gates. *Computer Design*, 45(3):12–27, 1963.
3. O. Barkol, Y. Ishai, and E. Weinreb. On locally decodable codes, self-correctable codes, and t-private PIR. *Algorithmica*, 58(4):831–859, 2010.
4. M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *STOC*, pages 1–10, 1988.
5. G. Bracha. An O(log n) expected rounds randomized byzantine generals protocol. *J. ACM*, 34(4):910–920, 1987.
6. P. Bro Miltersen. Lecutre notes. Available from author, 1992.
7. R. Canetti. Security and composition of multiparty cryptographic protocols. *J. Cryptology*, 13(1):143–202, 2000.
8. R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *FOCS*, pages 136–145, 2001.

[14] In particular, all these protocols are perfectly secure with a straight-line black-box simulator, which was shown in [33] to imply UC-security in the case of secure function evaluation. We note that while standard UC-security is cast in an asynchronous network model and does not guarantee output delivery, it can be extended to capture synchronous protocols which guarantee output delivery (cf. [12, Chapter 4]).

9. D. Chaum. The spymasters double-agent problem: Multiparty computations secure unconditionally from minorities and cryptographically from majorities. In *CRYPTO*, pages 591–602, 1989.
10. D. Chaum, C. Crépeau, and I. Damgård. Multiparty unconditionally secure protocols (extended abstract). In *STOC*, pages 11–19, 1988.
11. J. Considine, M. Fitzi, M. K. Franklin, L. A. Levin, U. M. Maurer, and D. Metcalf. Byzantine agreement given partial broadcast. *J. Cryptology*, 18(3):191–217, 2005.
12. R. Cramer, I. Damgård, and J. B. Nielsen. *Secure Multiparty Computation and Secret Sharing - An Information Theoretic Appoach*. 2012. Book draft, available at `http://www.daimi.au.dk/∼ivan/MPCbook.pdf`.
13. R. Cramer, S. Fehr, Y. Ishai, and E. Kushilevitz. Efficient multi-party computation over rings. In *EUROCRYPT*, pages 596–613, 2003.
14. I. Damgård, Y. Ishai, M. Krøigaard, J. B. Nielsen, and A. Smith. Scalable multiparty computation with nearly optimal work and resilience. In *CRYPTO*, pages 241–261, 2008.
15. Y. Desmedt, J. Pieprzyk, and R. Steinfeld. Active security in multiparty computation over black-box groups. In *SCN*, pages 503–521, 2012.
16. Y. Desmedt, J. Pieprzyk, R. Steinfeld, X. Sun, C. Tartary, H. Wang, and A. C.-C. Yao. Graph coloring applied to secure computation in non-abelian groups. *J. Cryptology*, 25(4):557–600, 2012.
17. Y. Desmedt, J. Pieprzyk, R. Steinfeld, and H. Wang. On secure multi-party computation in black-box groups. In *CRYPTO*, pages 591–612, 2007.
18. D. Dolev. The byzantine generals strike again. *J. Algorithms*, 3(1):14–30, 1982.
19. U. Feige, J. Kilian, and M. Naor. A minimal model for secure computation (extended abstract). In *STOC*, pages 554–563, 1994.
20. M. Fitzi and U. M. Maurer. Efficient byzantine agreement secure against general adversaries. In *DISC*, pages 134–148, 1998.
21. M. Fitzi and U. M. Maurer. From partial consistency to global broadcast. In *STOC*, pages 494–503, 2000.
22. J. A. Garay and Y. Moses. Fully polynomial byzantine agreement for $n > 3t$ processors in $t + 1$ rounds. *SIAM J. Comput.*, 27(1):247–290, 1998.
23. O. Goldreich. *Foundations of Cryptography: Volume 2, Basic Applications*. Cambridge University Press, New York, NY, USA, 2004.
24. O. Goldreich. Valiant's polynomial-size monotone formula for majority. `http://www.wisdom.weizmann.ac.il/~oded/PDF/mono-maj.pdf`, 2011.
25. O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In *STOC*, pages 218–229. ACM, 1987.
26. A. Gupta and S. Mahajan. Using amplification to compute majority with small majority gates. *Computational Complexity*, 6(1):46–63, 1996.
27. D. Harnik, Y. Ishai, E. Kushilevitz, and J. B. Nielsen. OT-combiners via secure computation. In *TCC*, pages 393–411, 2008.
28. M. Hirt and U. M. Maurer. Player simulation and general adversary structures in perfect multiparty computation. *J. Cryptology*, 13(1):31–60, 2000.
29. S. Hoory, A. Magen, and T. Pitassi. Monotone circuits for the majority function. In *APPROX-RANDOM*, pages 410–425, 2006.
30. Y. Ishai, E. Kushilevitz, R. Ostrovsky, and A. Sahai. Zero-knowledge proofs from secure multiparty computation. *SIAM J. Comput.*, 39(3):1121–1152, 2009.
31. Y. Ishai, M. Prabhakaran, and A. Sahai. Founding cryptography on oblivious transfer - efficiently. In *CRYPTO*, pages 572–591, 2008. Preliminary full version in `http://www.cs.illinois.edu/∼mmp/research.html`.

32. J. Kilian. Founding cryptography on oblivious transfer. In *STOC*, pages 20–31, 1988.
33. E. Kushilevitz, Y. Lindell, and T. Rabin. Information-theoretically secure protocols and security under composition. *SIAM J. Comput.*, 39(5):2090–2112, 2010.
34. Y. Lindell, E. Oxman, and B. Pinkas. The IPS compiler: Optimizations, variants and concrete efficiency. In *CRYPTO*, pages 259–276, 2011.
35. C. Lucas, D. Raub, and U. M. Maurer. Hybrid-secure mpc: trading information-theoretic robustness for computational privacy. In *PODC*, pages 219–228, 2010.
36. U. M. Maurer. Secure multi-party computation made simple. *Discrete Applied Mathematics*, 154(2):370–381, 2006.
37. M. C. Pease, R. E. Shostak, and L. Lamport. Reaching agreement in the presence of faults. *J. ACM*, 27(2):228–234, 1980.
38. T. Rabin and M. Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority (extended abstract). In D. S. Johnson, editor, *STOC*, pages 73–85. ACM, 1989.
39. X. Sun, A. C.-C. Yao, and C. Tartary. Graph design for secure multiparty computation over non-abelian groups. In *ASIACRYPT*, pages 37–53, 2008.
40. L. G. Valiant. Short monotone formulae for the majority function. *J. Algorithms*, 5(3):363–366, 1984.
41. A. C.-C. Yao. Protocols for secure computations (extended abstract). In *FOCS*, pages 160–164, 1982.
42. U. Zwick. Lecture notes. `http://www.cs.tau.ac.il/~zwick/circ-comp-new/six.ps`, 1996.