Adaptively Secure Multi-Party Computation with Dishonest Majority

Sanjam Garg and Amit Sahai

University of California, Los Angeles, USA
{sanjamg,sahai}@cs.ucla.edu

Abstract. Adaptively secure multiparty computation is an essential and fundamental notion in cryptography. In this work we focus on the basic question of constructing a multiparty computation protocol secure against a *malicious*, *adaptive* adversary in the *stand-alone* setting without assuming an honest majority, in the plain model. It has been believed that this question can be resolved by composing known protocols from the literature. We show that in fact, this belief is fundamentally mistaken. In particular, we show:

- **Round inefficiency is unavoidable when using black-box simulation:** There does not exist any $o(\frac{n}{\log n})$ round protocol that adaptively securely realizes a (natural) n-party functionality with a black-box simulator. Note that most previously known protocols in the adaptive security setting relied on black-box simulators.
- **A constant round protocol using non-black-box simulation:** We construct a *constant round* adaptively secure multiparty computation protocol in a setting without *honest majority* that makes crucial use of non-black box techniques.

Taken together, these results give the first resolution to the question of adaptively secure multiparty computation protocols with a malicious dishonest majority in the plain model, open since the first formal treatment of adaptive security for multiparty computation in 1996.

1 Introduction

The notion of *secure computation* is central to cryptography. Introduced in the seminal works of [1, 2], secure multi-party computation (MPC) allows a group of (mutually) distrustful parties $P_1, \ldots, P_n$, with private inputs $x_1, \ldots, x_n$, to jointly compute any functionality f in such a manner that the honest parties obtain correct outputs and no group of malicious parties learns anything beyond their inputs and prescribed outputs. In this setting we can consider an adversary that can *adaptively* corrupt parties throughout the protocol execution depending on its view during the execution. Adaptively secure multiparty computation is an essential and fundamental notion in cryptography. We refer the reader to ([3], Section 1) for further discussion on the importance of considering adaptive adversaries.

Canetti, Feige, Goldreich and Naor [3] constructed the first adaptively secure MPC protocol in the standalone setting, assuming the presence of an honest

R. Safavi-Naini and R. Canetti (Eds.): CRYPTO 2012, LNCS 7417, pp. 105–123, 2012.

majority. Beaver constructed an adaptively secure zero-knowledge protocol [4] (see also [5]) and an adaptively secure OT protocol [6]. Similar results for general *two-party* computation were established in [7, 8]. Assuming a trusted common random string (CRS), Canetti, Lindell, Ostrovsky and Sahai [9] gave the first adaptively secure MPC protocol without honest majority in the two-party and the multi-party setting, in fact under an even strong notion of security called the UC security (which can be achieved only with a trusted setup). In this paper, we focus on the following *basic* question:

Is it possible to construct multiparty computation protocols in the standalone setting (without any trusted setup) secure against a malicious, adaptive adversary that may corrupt any number of parties?

Previous Work on This Question: Choi, Dachman-Soled, Malkin and Wee [10, 11] give a construction of an adaptively secure multi-party computation protocol when given access to an ideal commitment (more formally, in the commitment hybrid model). At the same time, many adaptively secure protocols for securely realizing the commitment functionality (e.g. [12]) are known. And we know that it is possible to compose protocols by the composition theorem of Canetti [13], which holds in the adaptive security setting.

Surprisingly, however, it turns out that a straightforward application of these results does not (in fact as we argue, it **cannot**) achieve adaptive security in the multiparty setting![1] We stress that all the results stated in the previous paragraph (with proper formalization) are correct, and yet *still* the conclusion does not follow.

Adaptively Secure Composition: More than Meets the Eye. Somewhat surprisingly, a 2-party adaptively secure protocol *fails* to guarantee security when executed in the setting of n-parties, even if only two of the parties are *ever* talking to each other. (Thus, the 2-party adaptively secure commitment protocol of [12] is not necessarily adaptively secure in the n-party setting.) Indeed Canetti [13] (Theorem 10, Page 38) requires that in order to obtain an n-party adaptively secure protocol via the composition theorem, all the protocols being composed must be secure in the n-party setting to begin with. Nevertheless, this might seem surprising, and we demonstrate this issue by considering an example. We know that the vast majority of the simulators for MPC protocols are *black-box*. Now, consider an adaptively secure protocol in the 2-party case with a black-box simulator. Suppose that this 2-party protocol is executed in the setting of n parties out of which only two of them communicate. The black-box simulator for the 2-party protocol must rely on "rewinding" for the proof of security. However,

[1] Indeed, this composition seemed so "obvious" that in [11], Corollaries 2 and 3 claimed a result for adaptively secure multi-party computation in the plain model, and were given without proof. After seeing our work, the authors of [11] have corrected their paper to only refer to the two-party case in their corollaries. We stress that the corollaries of [11] do apply to the two-party setting, and that nothing in this paper should be taken to imply that any of the proofs given in [11] are incorrect.

in the adaptive n-party setting an adversary can also corrupt parties that *do not communicate* during the execution of the protocol. What if this happens during a rewinding? This case is never handled in the simulation for the 2-party case, and thus the proof of composition security breaks down. Indeed this seemingly small issue turns out to be a fundamental barrier to constructing adaptively secure MPC protocols. Not only does the proof break down, but as we show below, there are explicit impossibility results possible in the black-box setting. Thus, we show that in the setting without honest majority, we need to completely rethink the techniques used to construct adaptively secure multi-party computation protocols.

1.1 Our Results

We consider an asynchronous multi-party network[2] where the communication is open (i.e. all the communication between the parties is seen by the adversary) and delivery of messages is not guaranteed. (For simplicity, we assume that delivered messages are authenticated. This can be achieved using standard methods.) The two main results of the paper are:

Round inefficiency with a black-box simulation: There does not exist any $o(\frac{n}{\log n})$ round protocol that adaptively securely realizes a natural n-party functionality (more specifically an extension of the commitment functionality to the setting of n parties) with a black-box simulator. This result holds in the standalone setting in the plain model. We stress that all protocols that deal with adaptive security in the standalone model that we are aware of employ a black-box simulator. This implies that the techniques previously used to build adaptively secure multiparty protocols must incur a substantial efficiency loss.

A round efficient protocol with non-black box simulation: On the other hand, we show that non-black-box techniques can be used to circumvent the above described impossibility result. Assuming collision resistant hash functions, trapdoor permutations, augmented non-committing encryption [3, 9] and dense cryptosystems [14] we construct a *constant round* adaptively secure n-party protocol where the adversary is allowed to corrupt up to $n-1$ parties in the *non-erasure* model. If security against corruption of all n parties is desired then our construction yields a protocol with round complexity that is proportional to the depth of the circuit being evaluated. Alternatively, in the setting where all n parties can be corrupted, we can get a constant-round protocol if data *erasures* are allowed. This result shows a new area where non-black-box techniques can allow us to overcome round efficiency barriers that would otherwise exist.

[2] The fact that the network is asynchronous means that the messages are not necessarily delivered in the order which they are sent.

Discussion: The negative result leaves open the question of constructing adaptively secure protocols with black-box simulation, *but with poor round complexity*. We find this direction not very interesting in light of our positive result constructing round-efficient protocols using non black-box techniques. Nevertheless, we provide a sketch of a round-inefficient black-box construction in the full version, which is nearly tight with respect to our impossibility result.

On Erasures: Our positive results include round efficient protocols both in the setting of *erasures* and *non-erasures*. On the other hand our negative result holds even when parties are allowed to erase everything except their input. (Note that for our positive result with erasures, honest parties are certainly not required to erase their inputs.) From the earliest works on adaptive security [15] with erasures, it has been a major design goal to reduce the amount of erasures necessary. We refer the reader to ([13], Section 5.2) for a more general discussion on how trusted erasures may be a problematic assumption. Nevertheless, in light of our negative result we may also want to consider protocols that allow honest parties to erase their inputs during the protocol. We argue that it is particularly unreasonable to consider such protocols: Consider, for example, a setting where many hospitals are collaborating on some research involving their patient records. In order to do this research, they execute an MPC protocol, where each hospital's input is its database of patient records. We do not expect any hospital to be willing to erase all of its patient records (even from backup facilities, as backup facilities could also be adaptively corrupted), even temporarily, just to enable an MPC computation for research purposes. Worse, any protocol in the dishonest majority setting that requires honest parties to erase its inputs would enable an adversary, simply by aborting the protocol, to cause all honest parties to lose all of their inputs *forever*. While the example of hospital patient records underscore how undesirable erasure of inputs can be, this issue would be quite problematic in most contexts. Thus, we do not consider protocols where inputs can be erased[3]. Recall, however, that we can achieve round-efficient adaptive security without requiring erasures at all using non-black-box techniques.

1.2 Our Techniques

The central idea for our impossibility result is to argue that a black-box simulator of an $o(\frac{n}{\log n})$ round protocol for an n-party functionality does not gain anything via "rewindings" in the adaptive setting. Now we elaborate on this. Consider an r round (such that r is $o(\frac{n}{\log n})$) protocol for an n-party functionality with a black-box simulator. Consider an adversary that behaves completely honestly in

[3] It is not hard to see that if we were to allow erasure of inputs, then the following solution would be possible: The parties first non-malleably secret share their inputs among all parties. Subsequently, all parties erase everything except their own share (and the shares they receive from other parties). Finally, they run the protocol on the shares as inputs instead. However, we again stress that we find this solution highly unsatisfactory in light of the discussion above.

the protocol itself, however, after each round of the protocol it corrupts roughly $\omega(\log n)$ parties. Furthermore, the parties to be corrupted are chosen randomly (in fact pseudo-randomly based on the protocol messages so far) among the uncorrupted parties so far. On corruption of an honest party, the simulator is obliged to provide to the adversary the input of the party just corrupted. In its "main thread" execution with the adversary, to help the simulator in simulation, the simulator is also provided with these inputs. However, every time the simulator "rewinds" the adversary, the adversary will (with overwhelming probability) choose to corrupt at least one party that is not among the ones corrupted in the main thread. The simulator therefore will be unable to proceed in any "rewinding." Note that the only additional power awarded to a *black-box* simulator is essentially the ability to "rewind" the adversary which is essentially useless in our context. We therefore conclude that no such simulator can exist.

As is clear from the impossibility result just described, the problem of round inefficiency will be inherent to any simulator that "rewinds." In order to get around this problem, we turn to the non black-box simulation technique of Barak [16]. However, Barak's protocols are far from being adaptively secure. To achieve adaptive security, we adapt and make use of a technique developed in the context of concurrently secure computation [17–19].

Technical Overview for the Construction of Our Constant Round Protocol: Now we give a detailed technical overview of our construction. We will start by giving a high level idea of the final protocol and then delving into the details of sub-protocol (along with specifics of constructions) that need to be built. Throughout the following description, we advise the reader to keep in mind that our goal is to construct a round efficient protocol and as is clear from the negative result stated above this cannot be done with a simulator that "rewinds." Therefore we will restrict ourselves to a "straight-line" or a "non-rewinding" simulator.

- **Reducing the problem of adaptively secure MPC to generation of common random strings.** The starting point of our construction is the observation that an adaptively secure MPC protocol (Theorem 3, [20])[4] for any functionality can be realized in OT-hybrid (oblivious transfer) model. Note that in this construction each OT call is made between two parties. Further note that for an OT call between two parties security is required only if at least one of the two parties is honest. Additionally, note that we can adaptively securely realize OT functionality in the CRS hybrid (common random string) model (e.g., using [9]). Therefore in order to construct an adaptively secure protocol it suffices for us to adaptively securely realize the CRS functionality between every pair of parties where the CRS generated by a pair of parties is required to be honestly generated only if at least one of the two parties is honest.
- **Generating a common random string between a pair of parties.** Now we are left with the goal of adaptively securely realizing CRS between every pair of parties. We start by giving intuition for a protocol that adaptively

[4] We refer the reader to Remark 2 of [20] for discussion on variants of this result.

securely realizes CRS between two parties and then sketch the extension to the **setting of multiple parties**. We do this by constructing a **coin flipping protocol secure in the adaptive setting** in which the simulator can simulate in a straight-line manner. In order to construct such a coin flipping protocol our simulator will need the ability to *equivocate* on its commitments. In other words, we will need that our simulator can open its commitments to any value of its choice even after it has made those commitments. Looking ahead the simulator will also need the ability to *extract* (the reasons for which we see later) from the commitments made by the adversary. More specifically, we will need that the simulator can extract the values committed to by the adversary. Next we will first describe a mechanism that allows a straight-line simulator to equivocate on its commitments in the adaptive setting. Subsequently, we will see how equivocation can be used in setting up coin flipping (and the need of extractability in the process).

- **Equivocal commitment scheme in the adaptive setting.** We consider the *public-coin* zero-knowledge protocol[5] of Barak [16]. Even though this protocol is secure against adaptive corruptions of the verifier, it is far from being adaptively secure if we were to consider adaptive corruption of the prover. We will modify Barak's protocol as follows. For every bit sent by the prover in the Barak's protocol, we will require that our prover instead sends a random string of appropriate (length of a pseudorandom bit commitment) length. Note that in this modified protocol no actual proof is given. Furthermore, all the messages sent by an honest prover and an honest verifier in this modified protocol are just random bits and thus adaptive corruption of parties participating in an execution of this modified protocol does not help the adversary in any way. However, a key idea is that we can define an NP-relation that accepts a transcript if only if there exist decommitments of the prover messages such the decommitted prover messages along with the verifier messages form an accepting transcript of an execution of the Barak's protocol. Roughly speaking our modified protocol has two properties, with respect to this NP-relation:
  - No adaptively corrupted cheating prover interacting with an honest verifier in our modified protocol can output a witness for the transcript generated.
  - Consider any execution in which the prover is honest. In this execution our simulator (simulating the prover) can internally use the simulator of Barak's protocol and always output a witness for the transcript generated.

  We can reduce this transcript to a graph (can be constructed using an NP-reduction) that is Hamiltonian if and only if there exits a witness corresponding to the above NP-relation. Furthermore, given the witness we can also deduce the Hamiltonian cycle in the obtained graph. This graph can now be used to generate commitments such that a party with access to a

[5] In a public-coin zero-knowledge protocol all messages of the verifier correspond to random bits ("coin flips").

cycle in the graph can open them in any way. We refer the reader to Section 3 for more details on this.

Note that an execution of the modified Barak's protocol guarantees equivocability of commitments sent on behalf of only one of the two parties. Therefore we will have to set up *two* equivocal commitments. This can be easily achieved by execution modified Barak's protocol twice between the two parties switching the roles the two parties play in the two executions of the modified Barak's protocol.

- **Coin flipping protocol secure in the adaptive setting.** Next using equivocal commitments, we construct a coin flipping protocol between two parties A and B. One standard approach for constructing such a coin flipping protocol is to have the two-parties commit to random strings (via equivocal commitments) which they subsequently open one by one. The output of the protocol corresponds to the exclusive or of the two strings. Lets consider the case in which A opens first. The key technical problem that arises in this case is that if B is corrupted then the straight-line simulator (simulating A) without knowledge of the value committed to by B will not be able to force the output of the protocol to a value of its choice.

  We solve this problem by doing two coin flips both of which roughly follow the same outline as above. The first coin flipping is done in-order to setup a public key of a public key encryption scheme (with pseudorandom public-keys and pseudorandom ciphertexts). In this protocol we require that B opens first and this allows the simulator to force the output of the protocol to a value of its choice (in a straight-line manner) as long as A is honest. Subsequently the parties execute a second coin flipping protocol in which we require that B (B opens later now), in addition to the commitment it sends, is required to send encryption of the randomness used in generating the commitment using the public key generated in the first coin flipping. This allows the simulator to extract the value committed by B (if B is corrupted) even before A needs to open its committed value and thereby allowing it to simulate in a straight line manner. However, in case B is honest then the simulator will have to explain the encryptions as if they were honestly generated. We achieve this in a way similar to [9].
- **Setting up multiple common random strings.** Additionally other well known issues relating to *non-malleability* arise in constructing of constant round protocols [21] because of the need to execute different protocol instances in *parallel*. We deal with issue using the *two-slot technique* of [17]. Concretely we consider Pass' [22] family of non-black-box zero knowledge protocols with strong *simulation soundness* properties, i.e., any one of these protocols continues to remain *sound* even when all the other protocols in the family are being simulated. We prove that modifying these protocols just like we modified the Barak's protocol above suffices for our purposes.

Roadmap. We start by providing our impossibility result for black-box simulation in Section 2. Next we recall some very basic notions and setup notation

in Section 3. Finally we provide the construction of our constant round protocol in Section 5 using sub-protocols constructed in Section 4.

2 Round Inefficiency with a Black-Box Simulation Is Unavoidable

In this section, we show the existence of a deterministic n-party functionality for which there does not exist any $o(\frac{n}{\log n})$ round adaptively secure protocol, with a black-box simulator.

Before proceeding to the formal proof, we first give some intuition behind our impossibility result. The central idea to our proof is to argue that a black-box simulator (say) $\mathcal{S}$ of an $o(\frac{n}{\log n})$ round protocol does not gain any thing via "rewindings" in the adaptive setting. Informally speaking, this means that the simulator fails to get any useful information from any look-ahead thread and even in this setting it must still be able to extract the adversary's input. However, a simulator must have some additional power over a real adversary, and the only additional power awarded to a *black-box* simulator is essentially the ability to rewind the adversary. We therefore conclude that black-box simulators cannot exist for any $o(\frac{n}{\log n})$ round protocol, as stated in Theorem 1 below.

Theorem 1. *There exist a deterministic n-party functionality for which there does not (assuming one way functions) exist any $o(\frac{n}{\log n})$ round adaptively secure protocol with respect to* black-box *simulators.*

Proof. We will organize our proof into two main parts.

1. First, we consider the commitment functionality F, where there are two special parties – the *committer* C and the *receiver* R, and $n-2$ *dummy* parties. Let Π be any $o(\frac{n}{\log n})$-round n-party protocol that adaptively securely realizes F with respect to a black-box simulator. Then, for large enough n, we first construct an adversary $\mathcal{A}$ for Π, that corrupts C, such that *every* black-box simulator $\mathcal{S}$ for Π gets full participation from the adversary in the "main thread," but fails to get any "useful" information from the rewindings. Our adversary $\mathcal{A}$, has the inputs of dummy parties *hard-coded* inside itself and it acts in the following way. It starts by corrupting the committer C. It follows the honest committer strategy on behalf of C, except that after each round of Π it corrupts roughly $\omega(\log n)$ parties. Furthermore, the parties to be corrupted are chosen randomly (in fact pseudo-randomly based on the protocol messages so far) among the uncorrupted parties so far. On corruption of an honest party, the simulator is obliged to provide to the adversary the input of the party just corrupted. In its "main thread" execution with the adversary, to help the simulator in simulation, the simulator is also provided with these inputs. However, every time the simulator "rewinds" the adversary, the adversary will (with overwhelming probability) choose to corrupt at least one party that is not among the ones corrupted in the main

thread. The simulator therefore will be unable to proceed in any "rewinding." However, by security of the protocol such a simulator must still be able to extract the input of C. Proving this is in fact the crux of our proof.

2. Next, we consider *another* real-life adversary $\mathcal{A}'$, that corrupts all parties except C, uses the black-box simulator $\mathcal{S}$ (constructed above) and actually succeeds in extracting the input of the honest *committer*. This contradicts the assumption that Π securely realizes F.

Combining the two parts, we conclude that for the n-party commitment functionality F (as described above), there does not exist any $o(\frac{n}{\log n})$-round protocol that adaptively securely realizes F with respect to black-box simulators. We note that this is sufficient to prove Theorem 1. We give more details on both parts of the proof in the full-version.

3 Building Blocks for Our Constant Round Protocol

In this section we recall and define some very basic notions and setup notation. Let k denote a security parameter. We say that a function is *negligible* in the security parameter k if it is asymptotically smaller than the inverse of any fixed polynomial. Otherwise, the function is said to be *non-negligible* in k. We say that an event happens with *overwhelming* probability if it happens with a probability $p(k) = 1 - \nu(k)$ where $\nu(k)$ is a negligible function of k. In this section, we recall the definitions of basic primitives studied in this paper. We now discuss the main cryptographic primitives that we use in our construction.

Underlying Standard Commitment. The basic underlying commitment scheme Com is the standard non-interactive commitment scheme based on a one-way permutation f and a hard-core predicate b of f. That is, in order to commit to a bit σ, one computes $\mathsf{Com}(\sigma) = \langle f(U_k), b(U_k) \oplus \sigma \rangle$, where U_k is the uniform distribution over $\{0,1\}^k$. Note that Com is computationally secret, and produces pseudorandom commitments: that is, the distributions $\mathsf{Com}(0)$, $\mathsf{Com}(1)$, and U_{k+1} are computationally indistinguishable. Let the length of the commitment, for one bit message, generated by the pseudorandom commitment scheme be $\ell_C(k)$ ($k+1$ in the above case). For simplicity of exposition, in the sequel, unless necessary, we will assume that random coins are an implicit input to the commitment function. Furthermore, we will sometimes abuse notation and use the same notation to generate commitments to strings, which can be thought of as a concatenation of commitments of individual bits.

The Modifier Graph Based Commitment Scheme IDCom_G. We use the notation of [19] and some of the text has been taken verbatim from there [19].

To commit to a 0, the sender picks a random permutation π of the nodes of G, and commits to the entries of the adjacency matrix of the permuted graph one by one, using Com. The sender also commits (using Com) to the permutation π. These values are sent to the receiver as $c = \mathsf{IDCom}_G(0)$. To decommit, the sender decommits to π and decommits to every entry of the adjacency matrix. The receiver verifies that the graph it received is $\pi(G)$.

To commit to a 1, the sender chooses a randomly labeled q-cycle, and for all the entries in the adjacency matrix corresponding to edges on the q-cycle, it uses Com to commit to 1 values. For all the other entries, including the commitment to the permutation π, it simply produces random values from U_{k+1} (for which it does not know the decommitment!). These values are sent to the received as $c = \mathsf{IDCom}_G(1)$. To decommit, the sender opens only the entries corresponding to the randomly chosen q-cycle in the adjacency matrix.

This commitment scheme has the property of being computationally secret, i.e. the distributions $\mathsf{IDCom}_G(0)$ and $\mathsf{IDCom}_G(1)$ are computationally indistinguishable for any graph G. Also, given the opening of any commitment to both a 0 and 1, one can extract a Hamiltonian cycle in G. Finally, as with the scheme of [23], given a Hamiltonian cycle in G, one can generate commitments to 0 and then open those commitments to both 0 and 1.

Furthermore, here if the simulator has knowledge of a Hamiltonion cycle in G, it can also produce a random tape for the sender explaining $c = \mathsf{IDCom}_G(0)$ as a commitment to both 0 and 1. If, upon corruption of the sender, the simulator has to demonstrate that c is a commitment to 0 then all randomness is revealed. To demonstrate that c was generated as a commitment to 1, the simulator opens the commitments to the edges in the q-cycle and claims that all the unopened commitments are merely uniformly chosen strings (rather than commitments to the rest of G). This can be done since commitments produced by the underlying commitment scheme Com are pseudorandom.

In this setting as well, we will sometimes abuse notation and use the same notation to generate commitments to strings. In particular, we will use the notation $c = \mathsf{IDCom}_G(m; r)$ to denote the function that generates a commitment to m using random coins r. Furthermore a commitment $c = \mathsf{IDCom}_G(0^\kappa; r')$ to the zero string of length κ can be explained to any value m using the function $r = \mathsf{IDOpen}_G(m, r', \mathsf{w})$, where w is a Hamiltonian cycle in the graph G.

Dense Cryptosystems. In our construction we will need an the encryption scheme that has *pseudo-random public keys*. More specifically, we require that the public key is indistinguishable from a random string. Such an encryption scheme can be constructed from dense cryptosystems [14]. Furthermore, we will require that the scheme has pseudorandom ciphertexts. More formally:

Definition 1 (Encryption with pseudorandom ciphertexts). *A public-key cryptosystem (G, E, D) has pseudorandom ciphertexts of length $\ell_E(k)$ if for all non-uniform polynomial time adversaries $\mathcal{A}$ we have*

$$\begin{aligned} &\Pr\left[(pk, sk) \leftarrow G(1^k) : \mathcal{A}^{E_{pk}(\cdot)}(pk) = 1\right] \\ \approx &\Pr\left[(pk, sk) \leftarrow G(1^k) : \mathcal{A}^{R_{pk}(\cdot)}(pk) = 1\right], \end{aligned} \tag{1}$$

where $R_{pk}(m)$ runs $c \leftarrow \{0,1\}^{\ell_E(k)}$ and every time returns a fresh c. We require that the cryptosystem has errorless decryption.

Barak's Non-black Box Technique. We use the non black-box simulation technique of Pass [22] (which in turn builds on the work of Barak [16]). Consider

a "special" **NTIME**$(T(k))$ relation R_{Sim} as follows.[6] Let $k \in \mathbb{N}$ and let $T : N \to N$ be a "nice" function that satisfies $T(k) = k^{\omega(1)}$. Let $\{\mathcal{H}_k\}_{h \in \{0,1\}^k}$ be hash function family where $h \in \mathcal{H}_k$ maps $\{0,1\}^*$ to $\{0,1\}^k$. Let the triple $\langle h, c, r\rangle$ be the input to R_{Sim}. Further, consider a witness that consists of a program Π, a string $y \in \{0,1\}^{(|r|-k)}$, and string s. Then $R_{Sim}(\langle h, c, r\rangle, \langle \Pi, s, y\rangle) = 1$ if and only if:

1. $c = \mathsf{Com}(h(\Pi); s)$.
2. $\Pi(c, y) = r$ within $T(k)$ steps.

Witness Indistinguishable Universal Argument. The function $T(k)$ corresponding to the above describe relation R_{Sim} is super-polynomial in k. This implies that the language corresponding to R_{Sim} does not lie in NP (but rather in **NTIME**$(T(k))$). Such languages are beyond the scope of the "standard" witness indistinguishable proof systems (designed to handle NP-languages only), and will thus require the usage of a Witness Indistinguishable Universal Argument (WI-UARG) [24]. We note that the WI-UARG protocol of Barak and Goldreich [24] is public coin and the running time of the verifier in the protocol is bounded by a fixed polynomial.

4 Sub-protocols Used in Our Constant Round Protocol

In the construction of our final adaptively secure MPC protocol will use a *concurrently secure trapdoor generator protocol* $\langle P, V\rangle$ and a *coin flipping protocol* $\langle A, B\rangle$. In this section we will give a constructions of these protocols. Furthermore, we will prove special properties about these protocols that are useful for us in our final construction.

4.1 Trapdoor Generator Protocol

In this section we describe a *family of trapdoor generator protocols* $\langle P, V\rangle_i$ where $i \in \{1 \dots m\}$. $\langle P, V\rangle_i$ is a two party protocol between P_i and V_i and at the end of the protocol both parties output a Graph (let's say G). Consider the setting in which one protocol instance of each of the protocols $\langle P, V\rangle_1, \langle P, V\rangle_2 \dots \langle P, V\rangle_m$ is being executed concurrently in between n parties – $Q_1, \dots Q_n$ with inputs $x_1 \dots x_n$.[7] **We stress that in these protocol executions each Q_i could**

[6] The relation presented is slightly oversimplified and will make Barak's protocol work only when the hash function family is collision resistant against "slightly" super-polynomial sized circuits [16]. However, this can be modified to work assuming collision resistance against polynomial sized circuits only. It does not affect the analysis in this paper and we refer the reader to [24] for details.

[7] As we will see later that we only need security in the setting of parallel composition. However, in this section we will stick with the notion of concurrent composition and argue security in this setting. From this it follows immediately that our protocol is also secure in the less demanding setting of parallel composition.

potentially be playing the role of multiple P_j's and V_k's where $j, k \in \{1 \dots m\}$. In this setting we consider an adversary $\mathcal{A}$ that adaptively corrupts parties (an honest party reveals its input and random coins to the adversary on corruption).

However, for simplicity of exposition, we will model this instead as a setting of $n + 2m$ parties – $Q_1, \dots Q_n$ with inputs $x_1 \dots x_n$ and $P_1, \dots P_m, V_1, \dots V_m$ with no inputs. Furthermore, parties P_i and V_i execute an instance of the protocol $\langle P, V \rangle_i$. In this setting, we will consider an adversary that adaptively corrupts any of these parties. *We stress that any adversary in the original setting where each Q_i could potentially be playing the role of multiple P_j's and V_k's can always be used to construct an adversary in this setting.* This follows from the fact that in the original setting when an adversary corrupts a party Q_i it additionally corrupts multiple P_j's and V_k's. Analogously in this setting our adversary can continue to corrupt all the parties playing the roles of Q_i and P_j's and V_k's. Throughout the rest of this sub-section we will stick to this setting.
Very informally, assuming collision resistant hash functions, we will require that our protocol satisfies the following security properties:

1. For every such adversary $\mathcal{A}$ that adaptively corrupts parties, there exists a non-black box simulator $\mathcal{S}_{\langle P,V \rangle}$ (that obtains the inputs of parties adaptively corrupted by $\mathcal{A}$) such that the view of the adversary $\mathcal{A}$ in its interaction with honest parties and the view of the adversary $\mathcal{A}$ in its interaction with $\mathcal{S}_{\langle P,V \rangle}$ are computationally indistinguishable.
2. For every $i \in [m]$ such that P_i is not corrupted, $\mathcal{S}_{\langle P,V \rangle}$ outputs a Hamiltonian cycle in the graph that the execution of $\langle P, V \rangle_i$ yields.
3. For every $i \in [m]$ such that V_i is not corrupted, $\mathcal{A}$ cannot output a Hamiltonian cycle in the graph that parties P_i and V_i executing $\langle P, V \rangle_i$ output. Furthermore, we require that the $\mathcal{A}$ cannot output a Hamiltonian cycle even when for every $i \in [m]$ such that P_i is not corrupted it is additionally provided with the Hamiltonian cycle in the graph that the execution of $\langle P, V \rangle_i$ yields.
4. Finally, since we are in the adaptive setting, on corruption, an honest party (or, the simulator on behalf of the honest party in the simulated setting) must provide its input and random coins to the adversary. We will require that all the above properties hold even when this additional communication happens with the adversary.

Next we build some notation that will be useful in the formal description of our protocol $\langle P, V \rangle_i$ (in Figure 1).

Shadow Version of WI-UARG. Recall that in the WI-UARG protocol V only sends random bits. Finally at the end of the protocol V outputs 1 or 0. We will modify the WI-UARG protocol into what we call the *shadow version of the WI-UARG* protocol. The prover in the shadow protocol, for every bit sent by the prover in the original protocol, sends a random string in $\{0,1\}^{\ell_C(k)}$ (recall that $\ell_C(k)$ is the length of a pseudorandom commitment). Furthermore, the behavior of the verifier V remains unmodified. We will denote this modified protocol by

Com is a pseudo-random commitment scheme with output from $\{0,1\}^{\ell_C(k)}$.[a] We also use the "shadow version" of the 5-round public-coin WI-UARG protocol which we denote by sWI-UARG. Further let $\mu(k) = (m(k) \cdot 4k^3 + \ell(k) + k)$.

Setup : V_i sends $h \stackrel{\$}{\leftarrow} \mathcal{H}_k$ to P_i.

Slot 1 :

1. P_i sends a random string in $c_1 \stackrel{\$}{\leftarrow} \{0,1\}^{k \cdot \ell_C(k)}$ to V_i.
2. V_i sends a challenge string $r_1 \stackrel{\$}{\leftarrow} \{0,1\}^{i\mu(k)}$.

Slot 2 :

1. P_i sends a random string in $c_2 \stackrel{\$}{\leftarrow} \{0,1\}^{k \cdot \ell_C(k)}$ to V_i.
2. V_i sends a challenge string $r_2 \stackrel{\$}{\leftarrow} \{0,1\}^{(m+1-i)\mu(k)}$.

Main Proof Stage : P_i and V_i engage in the shadow version of the WI-UARG protocol[b] in which P_i proves to V_i the following statement:

- there exists Π, s, y, b such that $R_{Sim}(\langle h, c_b, r_b\rangle, \langle \Pi, s, y\rangle) = 1$.

Output Stage : Let `transcript` be the transcript of the above execution. Let G be a graph (can be constructed using an NP-reduction) that is Hamiltonian if and only if $\exists w$ such that $R_{uarg}(\texttt{transcript}, w) = 1$. Both parties output G.

[a] We use commitments based on one-way permutation just for simplicity of exposition. At the cost of a small complication, the one-message scheme could have been replaced by the Naor's [25] 2-message commitment scheme, which can be based on "ordinary" one-way functions.

[b] As already pointed, we advise the reader to keep in mind that both the honest prover and the honest verifier of the shadow version of the WI-UARG protocol just send random bits and that no real proof in an honest execution is ever given.

Fig. 1. $\langle P, V\rangle_i$ (the i^{th} protocol in the family of $m(k)$ protocols)

sWI-UARG. Consider an instance of execution of the sWI-UARG protocol with transcript `transcript`. Note that this transcript contains messages sent by the prover and the messages sent by the verifier. Further note that every $\ell_C(k)$ bit string sent by the prover could be interpreted (if they really are) as a commitment to a bit using the commitment scheme Com. Let w be the de-commitment information (if it exists) corresponding to all the $\ell_C(k)$ bit strings in `transcript` that are sent by the prover. Also let $\texttt{transcript}' = \mathsf{unshadow}(\texttt{transcript}, w)$[8] denote the transcript obtained by replacing every $\ell_C(k)$ bit string in `transcript` that is sent by P with the corresponding committed bit (as per Com). Let $R_{uarg}(\texttt{transcript}, w) = 1$ if and only if $V(\mathsf{unshadow}(\texttt{transcript}, w)) = 1$.

We stress that in the shadow version of the WI-UARG protocol both the honest prover and the honest verifier just send random strings and that *no real proof is actually given*. However, we also consider a modification of the prover strategy of shadow version of the WI-UARG protocol called the *simulated shadow prover*. The simulated shadow prover additionally obtains a witness for

[8] Note that the function unshadow is inefficient and is used just to define the NP-Relation.

the statement being proven and corresponding to every bit sent by the prover in the original WI-UARG protocol instead sends a commitment to that bit using the Com commitment scheme. Note that transcript `transcript` generated when the prover follows the simulated shadow prover strategy is such that there exists w such that $R_{uarg}(\texttt{transcript}, w) = 1$. Finally, note that the messages generated by a simulated shadow prover are computationally indistinguishable from the messages generated by an honest prover of the shadow version of the WI-UARG protocol. We will use this shadow prover strategy in our proof.

Common Parameters. All parties receive two parameters $m(k)$ and $\ell(k)$ as input. $m(k)$ corresponds to the size of the family of $\langle P, V \rangle$ protocols. In the adaptive setting, on corruption a party reveals its input to the adversary. We need a bound of this additional information sent to the adversary. This bound $\ell(k)$ corresponds to the sum of the lengths of inputs of parties $Q_1, \ldots, Q_n$.

Discussion about the Protocol. Observe that the entire protocol as described in Figure 1 involves only random strings from honest P_i and honest V_i. Also note that the main proof stage involves an execution of the shadow version of the WI-UARG protocol. As already pointed out an honest execution of this protocol does not involve any actual proof being given. Therefore, the graph generated in an honest execution of $\langle P, V \rangle_i$ will essentially never be Hamiltonian. We provide full details on our simulator for the family of $\langle P, V \rangle$ protocols and the proofs of the security properties in the full version.

4.2 Coin-Flipping Protocol

Now we describe our coin flipping protocol. $\langle A, B \rangle$ is a protocol between two parties A and B. Both A and B in the $\langle A, B \rangle$ protocol get graphs G_1 and G_2 as common input and output a random string of length $\ell'(k)$. We assume that no PPT adversary can output a Hamiltonian cycle in G_1 if B is honest. Similarly, we assume that no PPT adversary can output a Hamiltonian cycle in G_2 if A is honest. Consider the setting in which an instance of the $\langle A, B \rangle$ protocol is being executed. In this setting we consider an adversary $\mathcal{A}$ that adaptively corrupts parties (an honest party reveals its input and random coins to the adversary on corruption) and (assuming dense cryptosystems [14]) require that:

1. For every adaptive adversary $\mathcal{A}$, there exists a simulator $\mathcal{S}_{\langle A,B \rangle}$ which gets as input a Hamiltonian cycle in G_1 if A is honest (before the start of the protocol), a Hamiltonian cycle in G_2 if B is honest (before the start of the protocol) and a string crs (sampled from the uniform distribution) of length $\ell'(k)$. Furthermore, $\mathcal{S}_{\langle A,B \rangle}$ obtains the input of every party that $\mathcal{A}$ corrupts. In this setting we require that the view of the adversary $\mathcal{A}$ in its interaction with the honest parties and the view of the adversray $\mathcal{A}$ in its interaction with $\mathcal{S}_{\langle A,B \rangle}$ are computationally indistinguishable.
2. The output of the protocol execution is crs as long as either A or B is not corrupted till the end of the protocol.

Our Protocol $\langle A, B\rangle$. $(\mathsf{IDCom}, \mathsf{IDOpen})$ is a graph based commitment scheme. And, (G, E, D) is an encryption scheme with pseudorandom ciphertexts and pseudo-random public keys (of length $\ell_1(k)$). Both parties get graphs G_1 and G_2 as common input.

1. A generates a commitment $c = \mathsf{IDCom}_{G_1}(\alpha; r_1)$, where α is a random string in $\{0,1\}^{\ell_1(k)}$ and r_1 are the random coins. It sends c to B.
2. B sends a random string $\beta \in \{0,1\}^{\ell_1(k)}$ to A.
3. A sends (α, r_1) to B.
4. B aborts the protocol if $c \neq \mathsf{IDCom}_{G_1}(\alpha; r_1)$.
5. Both parties set $pk := \alpha \oplus \beta$.
6. A generates a commitment $d = \mathsf{IDCom}_{G_1}(\gamma; r_2)$, where γ is a random string in $\{0,1\}^{\ell'(k)}$ and sends it to B.
7. B generates commitments $f_i = \mathsf{IDCom}_{G_2}(\delta_i; s_i)$, where δ is a random string in $\{0,1\}^{\ell'(k)}$ and δ_i is the i^{th} bit of δ. It also generates $e_{i,\delta_i} = E_{pk}(s_i; t_i)$ and $e_{i,1-\delta_i}$ as a random string in $\{0,1\}^{\ell_2(k)}$ (where $\ell_2(k)$ is the appropriate length). Finally it sends f_i, $e_{i,0}$ and $e_{i,1}$ for all $i \in [\ell'(k)]$ to A.
8. A sends (γ, r_2) to B.
9. B aborts if $d \neq \mathsf{IDCom}_{G_1}(\gamma; r_2)$. Next, B sends δ_i, s_i, t_i for every $i \in [\ell'(k)]$ to A.
10. A aborts if for some $i \in [\ell'(k)]$, $f_i \neq \mathsf{IDCom}_{G_2}(\delta_i; s_i)$ or $e_{i,\delta_i} \neq E_{pk}(s_i; t_i)$.
11. Both parties output $\gamma \oplus \delta$ as the output of the protocol.

Intuition behind the Proof. If B is honest before Step 9, then $\mathcal{S}_{\langle A,B\rangle}$ equivocates in the messages (sent on behalf of B) and thereby forcing the output of the protocol to a value of its choice. Now consider the case in which B is corrupted before Step 9. In this case we need to force the output of the protocol only if A is not corrupted. In this case the simulator $\mathcal{S}_{\langle A,B\rangle}$ will be able to force the value pk generated in Step 3 of the protocol to a value of its choice. Subsequently it can force the output of the protocol to a value of its choice by extracting the values committed by B in Step 7 and then later equivocating in Step 8. Further note that the simulation itself is straight line. However in proving indistinguishability of simulation from real interaction we do rewind the adversary.[9] We provide full details on our simulator for the $\langle A, B\rangle$ protocols and the proof of the security properties in the full version.

5 Our Constant Round Protocol

Let f be any adaptively well-formed[10] functionality. In this section we will give a constant round protocol Π that adaptively-securely realizes f. Let $Q_1, \ldots, Q_n$ be n parties that hold inputs $x_1, \ldots x_n$ respectively. Let f be the function that they wish to evaluate on their inputs. Furthermore, let $\ell(k) = |x_1| + |x_2| \ldots |x_n|$.

[9] This is not a problem as rewinding is used only in the proof in order to reach a contradiction.

[10] We require[9] that the functionality reveals its random input in case all parties are corrupted.

We start by describing a protocol that adaptively securely realizes the $\mathcal{F}_{n-crs}$ functionality (Figure 2). Note that whenever a party is corrupted then it reveals its input and random coins to the adversary. In our construction we use a *family of trapdoor generator protocols* $\langle P, V \rangle_i$ where $i \in \{1 \dots m\}$ (Section 4.1) and a *coin flipping protocol* $\langle A, B \rangle$ (Section 4.2). The protocol proceeds as follows.

1. **Trapdoor Creation Phase:** $Q_i \leftrightarrow Q_j$: For all $i, j \in [n]$, such that $i \neq j$, Q_i and Q_j engage in an execution of the protocol $\langle P, V \rangle_t$ (with common input n^2 and $\ell(k)$ where $t = i \cdot (n-1) + j$), where Q_i plays the role of P_t and Q_j places the role of the V_t. Let $G_{i,j}$ be the output of the protocol. All these executions are done in parallel.
2. **Coin Flipping Phase:** $P_i \leftrightarrow P_j$: For all $i, j \in [n]$, such that $i < j$, Q_i and Q_j engage in an execution of the protocol $\langle A, B \rangle$, denoted as $\langle A, B \rangle^{i,j}$, where Q_i plays the role of A and Q_j plays the role of B with common input $G_{i,j}$ and $G_{j,i}$. Q_i and Q_j output the output of $\langle A, B \rangle^{i,j}$ as $\mathsf{crs}_{i,j}$. All these executions are done in parallel.

Common input: Let $Q_1, \dots, Q_n$ be n parties that hold inputs $x_1, \dots x_n$ respectively. Furthermore, let $\ell(k) = |x_1| + |x_2| \dots |x_n|$. $\mathcal{F}_{n-crs}$ sets up a list $\mathcal{L}$ that is initially set to be empty. Let $\mathcal{S}$ be the ideal world adversary and let A at any point be the set of corrupted parties.

1. On receiving a messages (crs, i, j) from party Q (including $\mathcal{S}$), $\mathcal{F}_{n-crs}$:
   - *If* $\exists(\mathsf{crs}, i, j, crs_{i,j}) \in \mathcal{L}$*:* Sends $crs_{i,j}$ to Q.
   - *If* $(\mathsf{crs}, i, j, \cdot) \notin \mathcal{L}$ *and if at least one of* Q_i *or* Q_j *is not in* A*:* Samples a random string $crs_{i,j} \in \{0,1\}^{\ell'(k)}$, adds $(\mathsf{crs}, i, j, crs_{i,j})$ to $\mathcal{L}$ and sends $crs_{i,j}$ to Q.
   - *If both* $Q_i, Q_j \in A$*:* Obtain crs from $\mathcal{S}$ and send the obtained crs to Q.
2. On receiving a message $(\mathsf{corrupt}, Q_i)$ from $\mathcal{A}$, $\mathcal{F}_{n-crs}$ adds Q_i to A and sends x_i to $\mathcal{S}$.

Fig. 2. $\mathcal{F}_{n-crs}$

Theorem 2. *Assuming collision resistant hash functions and dense cryptosystems [14], the* constant round *protocol just described above adaptively securely evaluates* $\mathcal{F}_{n-crs}$ *(Figure 2).*

The formal proof of the above theorem appears in the full version.

Realizing All Functionalities. Now we elaborate on how we can construct an adaptive secure protocol for any functionality.

Theorem 3. *Assuming collision resistant hash functions, trapdoor permutations, augmented non-committing encryption and dense cryptosystems, for any* $n \geq 2$*, there exists an* n*-party* constant round *MPC protocol that is secure against any malicious adversary which may adaptively corrupt at most* $n - 1$ *parties*

Recall that we have already constructed a protocol that adaptively securely realizes $\mathcal{F}_{n-crs}$ ideal functionality. Therefore we are left with just constructing a protocol secure in the $\mathcal{F}_{n-crs}$-hybrid model. This is implied by the following proposition.

Proposition 1. *Assuming trapdoor permutations and augmented non-committing encryption, for any $n \geq 2$, there exists an n-party constant round MPC protocol in the $\mathcal{F}_{n-crs}$-hybrid model that is secure against any malicious adversary which may adaptively corrupt at most $n-1$ parties.*

Remark on the above Proposition. Note that if security against corruption of all n parties is desired then a proposition (similar to the one above) that yields a protocol with round complexity that depends on the depth of the circuit being evaluated still holds. Additionally in this setting we can get a constant-round protocol if data *erasures* are allowed. We refer the reader to Remark 2 in [20] for discussion on this.

Proof Sketch. The proof of the above proposition is implicit in a number of previous works. For concreteness, we will describe one way of constructing such a protocol. Observe that the $\mathcal{F}_{n-crs}$ ideal functionality can be split into $\frac{n(n-1)}{2}$ ideal functionalities each generating a common random string for each pair of parties (each of these ideal functionalities works correctly as long as at least one of the two parties it is serving is honest). Next note that, using [9], given access to a common random string, we can construct an adaptively secure OT protocol. Using this protocol and applying the UC composition theorem [26] (Composing Different Setups, Page 61), multiple times, we can construct a protocol that achieves adaptively secure OT[11] between every pair of parties (as long as at least one of the two parties it is serving is honest). Finally, using these OT channels [20] we can adaptively securely realize any functionality.

Acknowledgements. Research supported in part from a DARPA/ONR PROCEED award, NSF grants 1136174, 1118096, 1065276, 0916574 and 0830803, a Xerox Faculty Research Award, a Google Faculty Research Award, an equipment grant from Intel, and an Okawa Foundation Research Grant. This material is based upon work supported by the Defense Advanced Research Projects Agency through the U.S. Office of Naval Research under Contract N00014-11-1-0389. The views expressed are those of the author and do not reflect the official policy or position of theDepartment of Defense or the U.S. Government.

We gratefully thank Abhishek Jain, Yuval Ishai, Manoj Prabhakaran, and Akshay Wadia for valuable discussions about this work. We would also like to thank Divya Gupta and the anonymous reviewers of CRYPTO 2012 for their comments on the previous drafts of this paper.

References

1. Yao, A.C.: How to generate and exchange secrets. In: Proc. 27th FOCS, pp. 162–167 (1986)
2. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game or a completeness theorem for protocols with honest majority. In: STOC, pp. 218–229 (1987)

[11] [9] assume trapdoor permutations and augmented non-committing encryption in their construction.

3. Canetti, R., Feige, U., Goldreich, O., Naor, M.: Adaptively secure multi-party computation. In: STOC, pp. 639–648 (1996)
4. Beaver, D.: Adaptive zero knowledge and computational equivocation (extended abstract). In: STOC, pp. 629–638 (1996)
5. Lindell, Y., Zarosim, H.: Adaptive zero-knowledge proofs and adaptively secure oblivious transfer. J. Cryptology 24(4), 761–799 (2011)
6. Beaver, D.: Equivocable Oblivious Transfer. In: Maurer, U.M. (ed.) EUROCRYPT 1996. LNCS, vol. 1070, pp. 119–130. Springer, Heidelberg (1996)
7. Beaver, D.: Adaptively Secure Oblivious Transfer. In: Ohta, K., Pei, D. (eds.) ASIACRYPT 1998. LNCS, vol. 1514, pp. 300–314. Springer, Heidelberg (1998)
8. Katz, J., Ostrovsky, R.: Round-Optimal Secure Two-Party Computation. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 335–354. Springer, Heidelberg (2004)
9. Canetti, R., Lindell, Y., Ostrovsky, R., Sahai, A.: Universally composable two-party and multi-party secure computation. In: STOC, pp. 494–503 (2002)
10. Choi, S.G., Dachman-Soled, D., Malkin, T., Wee, H.: Improved Non-committing Encryption with Applications to Adaptively Secure Protocols. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 287–302. Springer, Heidelberg (2009)
11. Choi, S.G., Dachman-Soled, D., Malkin, T., Wee, H.: Simple, Black-Box Constructions of Adaptively Secure Protocols. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 387–402. Springer, Heidelberg (2009)
12. Pass, R., Wee, H.: Black-Box Constructions of Two-Party Protocols from One-Way Functions. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 403–418. Springer, Heidelberg (2009)
13. Canetti, R.: Security and composition of multi-party cryptographic protocols. Cryptology ePrint Archive, Report 1998/018 (1998), http://eprint.iacr.org/
14. De Santis, A., Persiano, G.: Zero-knowledge proofs of knowledge without interaction. In: Proceedings of the 33rd Annual Symposium on Foundations of Computer Science, SFCS 1992, pp. 427–436. IEEE Computer Society, Washington, DC (1992)
15. Beaver, D., Haber, S.: Cryptographic Protocols Provably Secure against Dynamic Adversaries. In: Rueppel, R.A. (ed.) EUROCRYPT 1992. LNCS, vol. 658, pp. 307–323. Springer, Heidelberg (1993)
16. Barak, B.: How to go beyond the black-box simulation barrier. In: Proc. 42nd FOCS, pp. 106–115 (2001)
17. Pass, R., Rosen, A.: Bounded-concurrent secure two-party computation in a constant number of rounds. In: FOCS, pp. 404–413 (2003)
18. Pass, R., Rosen, A.: New and improved constructions of nonmalleable cryptographic protocols. SIAM J. Comput. 38(2), 702–752 (2008)
19. Barak, B., Sahai, A.: How to play almost any mental game over the net - concurrent composition via super-polynomial simulation. In: FOCS, pp. 543–552 (2005)
20. Ishai, Y., Prabhakaran, M., Sahai, A.: Founding Cryptography on Oblivious Transfer – Efficiently. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 572–591. Springer, Heidelberg (2008)
21. Katz, J., Ostrovsky, R., Smith, A.: Round Efficiency of Multi-Party Computation with a Dishonest Majority. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 578–595. Springer, Heidelberg (2003)
22. Pass, R.: Bounded-concurrent secure multi-party computation with a dishonest majority. In: Proc. 36th STOC, pp. 232–241 (2004)

23. Feige, U., Shamir, A.: Zero Knowledge Proofs of Knowledge in Two Rounds. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 526–544. Springer, Heidelberg (1990)
24. Barak, B., Goldreich, O.: Universal arguments and their applications. SIAM J. Comput. 38(5), 1661–1694 (2008)
25. Naor, M.: Bit commitment using pseudorandomness. Journal of Cryptology 4(2), 151–158 (1991); Preliminary version In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 128–136. Springer, Heidelberg (1990)
26. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols (2000), `http://eprint.iacr.org/`