

Adaptively Secure Multi-Party Computation with Dishonest Majority

Sanjam Garg* and Amit Sahai*

University of California, Los Angeles, USA

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.

* 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 the Department of Defense or the U.S. Government.

1 Introduction

The notion of *secure computation* is central to cryptography. Introduced in the seminal works of [Yao86,GMW87], secure multi-party computation (MPC) allows a group of (mutually) distrustful parties P_1, \dots, P_n , with private inputs x_1, \dots, 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 ([CFGN96], Section 1) for further discussion on the importance of considering adaptive adversaries.

Canetti, Feige, Goldreich and Naor [CFGN96] constructed the first adaptively secure MPC protocol in the standalone setting, assuming the presence of an honest majority. Beaver constructed an adaptively secure zero-knowledge protocol [Bea96a] (see also [LZ11]) and an adaptively secure OT protocol [Bea96b] (see also [GWZ09]). Similar results for general *two-party* computation were established in [Bea98,KO04]. Assuming a trusted common random string (CRS), Canetti, Lindell, Ostrovsky and Sahai [CLOS02] 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 [CDMW09a,CDMW09b] 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. [PW09]) are known. And we know that it is possible to compose protocols by the composition theorem of Canetti [Can98], 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 [PW09] is not necessarily adaptively secure in the n -party setting.) Indeed Canetti [Can98] (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, 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

¹ Indeed, this composition seemed so “obvious” that in [CDMW09b], 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 [CDMW09b] have corrected their paper to only refer to the two-party case in their corollaries. We stress that the corollaries of [CDMW09b] 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 [CDMW09b] are incorrect.

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 network² 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 [CFG96,CLOS02] and dense cryptosystems [DSP92] 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.

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 Appendix A, 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 [BH92] with erasures, it has been a major design goal to reduce the amount of erasures necessary. We refer the reader to ([Can98], 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 erased³. Recall, however, that we can achieve round-efficient adaptive security without requiring erasures at all using non-black-box techniques.

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

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

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 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 [Bar01]. 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 [PR03,PR08,BS05].

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, [IPS08])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 [CLOS02]). 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 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

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.

⁴ We refer the reader to Remark 2 of [IPS08] for discussion on variants of this result.

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 protocol⁵ of Barak [Bar01]. 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 exists 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 cycle in the graph can open them in any way. We refer the reader to Section 4 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 [CLOS02].

- **Setting up multiple common random strings.** Additionally other well known issues relating to *non-malleability* arise in constructing of constant round protocols [KOS03] because of the need to execute different protocol instances in *parallel*. We deal with issue using the *two-slot technique* of [PR03]. Concretely we consider Pass’ [Pas04] 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

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

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 recalling the formal definition of adaptive secure MPC in Section 2. We then provide our impossibility result for protocols with black-box simulation in Section 3. Next we recall some very basic notions and setup notation in Section 4. Finally we provide the construction of our constant round protocol in Section 6 using sub-protocols constructed in Section 5.

2 Adaptively Secure Multi-Party Computation

In this section we recall the formal definition of adaptively secure MPC. We adapt our definition of adaptive MPC from [CFG96, Can98]. Parts of the text have been taken verbatim from [CFG96]. In order to define adaptive MPC we first describe the real world model process; next we describe the ideal process; and finally present the definition.

Real World. An n -party protocol π is a collection of n interactive, probabilistic algorithms, where the i^{th} algorithm is run by the i^{th} party, P_i . Each P_i has input $x_i \in \{0, 1\}^*$, random input $r_i \in \{0, 1\}^*$, and the security parameter k . We assume that all communication is done via a *broadcast channel*. That is, any message sent by some party at some communication round is received by all other parties (including the adversary) before the beginning of the next round.

At the onset of the computation \mathcal{A} receives some auxiliary information z . Next, the computation proceeds according to the given model of computation. For concreteness, we specify the following (synchronous, with rushing) model of computation. The computation proceeds in ROUNDS; each round proceeds in MINI-ROUNDS, as follows. Each mini-round starts by allowing \mathcal{A} to *corrupt* parties one by one in an adaptive way. (The behavior of the system upon corruption of a party is described below.) Next \mathcal{A} chooses an uncorrupted party, P_i , that was not yet activated in this round and **ACTIVATES** it. Upon activation, P_i receives the messages sent to it in the previous round, generates its messages for this round, and the next mini-round begins. \mathcal{A} also learns the messages sent by P_i . Once all the uncorrupted parties were activated, \mathcal{A} generates the messages to be sent by the corrupted parties that were not yet activated in this round, and the next round begins.

Once a party is corrupted the party’s input and random input (note that the adversary already knows the entire history of the messages sent and received by the party) become known to \mathcal{A} .

At the end of the computation (say, at some pre-determined round) all parties locally generate their outputs. The uncorrupted parties output whatever is specified in the protocol. The corrupted parties output any arbitrary PPT function of the view of \mathcal{A} .

The overall output of the real-world experiment consists of all the values output by all parties at the end of the protocol, and is denoted by $\text{REAL}_{\mathcal{A}}^{\pi}(1^k, \mathbf{x}, \mathbf{r}, z)$. Let $\text{REAL}_{\mathcal{A}}^{\pi}(1^k, \mathbf{x}, z)$ denote the random variable $\text{REAL}_{\mathcal{A}}^{\pi}(1^k, \mathbf{x}, \mathbf{r}, z)$ where \mathbf{r} is uniformly chosen. Let $\text{REAL}_{\mathcal{A}}^{\pi}$ denote the ensemble $\{\text{REAL}_{\mathcal{A}}^{\pi}(1^k, \mathbf{x}, z)\}_{k \in \mathbb{N}, \mathbf{x} \in \{0, 1\}^*, z \in \{0, 1\}^*}$.

Ideal World. We first define the ideal world experiment, where n parties P_1, \dots, P_n interact with a *trusted party* for computing a function $f : \mathbb{N} \times (\{0, 1\}^*)^n \times \{0, 1\}^* \rightarrow (\{0, 1\}^*)^n$. Each party P_i obtain an input $x_i \in \{0, 1\}^*$. Note that the parties wish to evaluate $f(k, \mathbf{x}, r_f)_1, \dots, f(k, \mathbf{x}, r_f)_n$ where $r_f \leftarrow \{0, 1\}^s$ and s is a value determined by the security parameter and P_i learns $f(k, \mathbf{x}, r_f)_i$. The ideal world computation in presence of the **ADAPTIVE IDEAL WORLD ADVERSARY** \mathcal{S} (with random input r) and an (incorruptible) **TRUSTED PARTY**, \mathcal{F} proceeds as follows.

First corruption stage: First, as in the real-life model, \mathcal{S} gets an auxiliary information z . Next, \mathcal{S} proceeds in iterations, where in each iteration \mathcal{S} may decide to corrupt some party based on \mathcal{S} ’s random input and the information gathered so far. Once a party is corrupted its input becomes known to \mathcal{S} . Let B denote the set of parties corrupted at the end of this stage.

Computation stage: Each honest party P_i sends its input x_i to the trusted party \mathcal{F} . For each corrupted party, the adversary may select any value y_i (based on the information gathered so far) and send it to the trusted party. Let y_1, \dots, y_n be the inputs that were sent to the trusted party \mathcal{F} . Next, \mathcal{F} chooses r_f randomly from $\{0, 1\}^s$ and hands each party P_i the value $f(k, \mathbf{y}, r_f)_i$.

Second corruption stage: Upon learning the corrupted parties' outputs of the computation, \mathcal{S} proceeds in another sequence of iterations, where in each iteration \mathcal{S} may decide to corrupt some additional party, based on the information gathered so far. Upon corruption, \mathcal{S} sees the corrupted party's input and output.

Output stage: Each uncorrupted party P_i outputs $f(k, \mathbf{y}, r_f)_i$, and the corrupted parties output some arbitrary PPT function of the information gathered by the adversary during the computation in the ideal process.

Post-execution corruption: Once the outputs are generated, just like in the second corruption phase, \mathcal{S} at any point in the protocol may decide to adaptively proceed in another sequence of iterations, where in each iteration \mathcal{S} may decide to corrupt some additional party, based on the information gathered so far.

The overall output of the ideal process consists of all the values output by all parties at the end of the protocol, and is denoted by $\text{IDEAL}_{\mathcal{S}}^{\mathcal{F}}(1^k, \mathbf{x}, z, \mathbf{r})$, where $\mathbf{r} = (r, r_f)$. Let $\text{IDEAL}_{\mathcal{S}}^{\mathcal{F}}(1^k, \mathbf{x}, z)$ denote the distribution of $\text{IDEAL}_{\mathcal{S}}^{\mathcal{F}}(1^k, \mathbf{x}, z, \mathbf{r})$ when \mathbf{r} is uniformly distributed. Let $\text{IDEAL}_{\mathcal{S}}^{\mathcal{F}}$ denote the distribution ensemble $\{\text{IDEAL}_{\mathcal{S}}^{\mathcal{F}}(1^k, \mathbf{x}, z)\}_{k \in \mathbb{N}, \mathbf{x} \in \{0,1\}^n, z \in \{0,1\}^*}$

Equivalence of Computations. Informally, we require that executing a protocol π in the real world roughly emulates the ideal process for evaluating f .

Definition 1. (*Adaptive Security*) Let f be any adaptively well-formed⁶ n -ary function, π be a protocol for n parties. We say that π adaptively securely evaluates f if for every real world adversary \mathcal{A} there exists an ideal world adversary \mathcal{S} , such that $\text{REAL}_{\mathcal{A}}^{\pi} \stackrel{c}{\approx} \text{IDEAL}_{\mathcal{S}}^{\mathcal{F}}$.

3 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 (as per Definition 1) 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

⁶ We require[CLOS02] that the functionality reveals its random input r_f in case all parties are corrupted.

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 next.

PART 1. Simulator failing in all rewindings

Let F be an n -party commitment functionality with two special parties – the *committer* C with input x_C , the *receiver* R , and $n - 2$ *dummy* parties with inputs $x_1, x_2 \dots x_{n-2}$ respectively, each of which is a k -bit long *uniform random string*. In this setting we start by making the notion of a *round* precise. In the n -party setting a round involves a messages being sent by every party. We will think of a round as a two step process. In the first step all the honest parties (the ones that are supposed to according to the protocol) send their messages and in the second step the adversary sends the messages on behalf of all the corrupted parties (again for the ones that are supposed to according to the protocol).

Description of \mathcal{A} . We will first construct an adversary \mathcal{A} , that has the inputs $x_1, x_2 \dots x_{n-2}$ *hard-coded* inside itself. In other words the adversary is aware of the values $x_1, x_2 \dots x_{n-2}$ and it will use them in its execution. \mathcal{A} acts in the following way. It starts by corrupting the committer C . Furthermore, it follows the honest committer strategy on behalf of C , except that for each round of Π (which has $r = o(\frac{n}{\log n})$ rounds) after the messages sent on behalf of all honest parties have been received, \mathcal{A} corrupts $\mu(k) = \frac{n}{2r}$ parties chosen randomly (in fact, pseudorandom in the transcript so far) among the uncorrupted dummy parties. At this point the simulator provides the inputs (and the randomness) of the corrupted dummy parties to adversary. Our adversary ignores the randomness, matches the provided input values against the values hard-coded inside it. It aborts with a special abort message **Match-Abort** if any of the values does not match. Finally, it responds with appropriate messages on behalf of all honest parties.

Next we argue that no black-box simulator \mathcal{S} , that completes the “main thread,” can succeed in preventing the adversary from aborting in the look-ahead threads (except with negligible probability) and therefore fails to get any “useful” information from the rewindings. We first introduce some notations and conventions for the remainder of the proof. We will borrow some of our notations and conventions from [BL04]. Without loss of generality, we assume that R is the protocol initiator in Π . Recall that the number of rounds in Π is r , where one round consists of messages sent by the honest parties and then the adversary.

Now, recall that a black-box simulator \mathcal{S} for any protocol has ‘oracle access’ to the real world adversary \mathcal{A} . Formally, we consider \mathcal{A} as a non-interactive algorithm that gets as input the history of the messages sent to \mathcal{A} , and outputs the next message that \mathcal{A} would send in an execution of Π corresponding to the specific execution. More formally, \mathcal{S} can query \mathcal{A} with any sequence of messages of the form $(\alpha_1, \beta_1, \alpha_2, \beta_2 \dots, \alpha_i), i \leq r$ or $(\alpha_1, \beta_1, \alpha_2, \beta_2 \dots, \alpha_i, \beta_i), i \leq r$ (i.e., the query contains the history of the messages sent to the adversary), and it will receive back the next message that \mathcal{A} would send in any execution of Π in which it received this sequence of messages. Note that this history consists of two kinds of messages. The α_i s correspond to the messages of the protocol Π . On the other hand β_i s corresponds to the inputs (along with the random coins) of the honest parties that the simulator sends in response to the adversary’s corruption requests. For the sake of simplicity, we will assume without loss of generality, that the simulator \mathcal{S} always follows the following two conventions:

1. It never asks the same query twice.
2. If \mathcal{S} queries \mathcal{A} with γ , then it must have queried \mathcal{A} with all the proper prefixes of γ prior to this query.
3. \mathcal{S} outputs a view that corresponds to the messages sent between the adversary \mathcal{A} and the simulator \mathcal{S} in the “main thread” of the interaction.

We note that it is easy to modify any black-box simulator such that it follows the above conventions, without affecting its output distribution.

Note that our adversary never aborts with **Match-Abort** in the real world. This is because in the real world an honest party on corruption always responds with its true input which will match with the inputs hard-coded inside the adversary. Therefore, by indistinguishability of the outputs generated by the adversary and the simulator, the adversary must not abort (in the “main thread”) in its interaction with the simulator. Therefore \mathcal{S} needs to make at least r queries of the form $(\alpha_1, \beta_1, \alpha_2, \beta_2 \dots, \alpha_i, \beta_i)$ for each $i \in [r]$ in order to be able to just complete the “main thread.” Therefore, the simulator \mathcal{S} makes exactly one query of the form $(\alpha_1, \beta_1, \alpha_2, \beta_2 \dots, \alpha_i, \beta_i)$ for each $i \in [r]$ and in order to avoid **Match-Abort** in any of these executions (and maintain indistinguishability of the outputs generated by the adversary and the simulator) it must corrupt the same parties as \mathcal{A} . Note that in the “main thread” of the execution the adversary corrupts a total of up to $r\mu(k)$ (i.e., $\frac{n}{2}$) parties and our simulator is allowed to corrupt only the same parties. This in particular implies that our simulator cannot corrupt more than $\frac{n}{2}$ parties. Let E be the event that a simulator successfully makes any additional query of the form $(\alpha_1, \beta_1, \alpha_2, \beta_2 \dots, \alpha'_i, \beta'_i)$ such that $\alpha'_i \neq \alpha_i$ for some $i \in [r]$ to the adversary such that the adversary does not output **Match-Abort**. We will next argue that E happens with negligible probability allowing us to conclude that \mathcal{S} fails every time it rewinds \mathcal{A} .

For the sake of contradiction we start by assuming there exists a simulator such that E happens with a non-negligible probability. Now observe that within this query β_i corresponds to the inputs of $\frac{n}{r}$ (i.e., $\omega(\log n)$) parties which are chosen pseudo-randomly in $\alpha_1, \beta_1, \alpha_2, \beta_2 \dots, \alpha_i$. We start by changing the experiment slightly. Instead of choosing the parties to be corrupted pseudo-randomly we choose these parties randomly. It follows from that security of the pseudorandom function that E still happens with a non-negligible probability. The probability, in this experiment, that all the freshly chosen $\omega(\log n)$ parties come from the $\frac{n}{2}$ parties that the simulator corrupts (for simulation of the “main-thread”) is less than $\left(\frac{n}{2 \cdot (n-2)}\right)^{\omega(\log n)}$, which is negligible. Additionally, recall that inputs of dummy parties were chosen randomly and the simulator can correctly guess these values only with a negligible probability. This is a contradiction.

PART 2. Adversary \mathcal{A}' against which F is insecure

We already have a simulator \mathcal{S} that does not rewind the adversary \mathcal{A} (described above) and still successfully extracts the input x_C of the committer. Next we will construct another adversary \mathcal{A}' that corrupts R and all the dummy parties, executes \mathcal{S} internally and successfully extracts the input x_C of the committer. \mathcal{A}' internally executes \mathcal{S} and follows the same strategy as \mathcal{A} , except that all messages it sends on behalf of C (i.e., the α_i s) are actually obtained from an external honest committer C . \mathcal{S} must still continue to extract the input of the committer C . Since the simulator never rewinds \mathcal{A} , we will never need to rewind the external party C . This contradicts the security of the protocol Π and concludes the proof.

4 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 $\text{Com}(\sigma) = \langle f(U_k), b(U_k) \oplus \sigma \rangle$, 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 $\text{Com}(0)$, $\text{Com}(1)$, 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 Feige-Shamir Commitment Scheme. We briefly define the Feige-Shamir trapdoor commitment scheme [FS89], which is based on the the zero-knowledge proof of Hamiltonicity of Blum [Blu87]. Some of the text has been taken verbatim from there [BS05]. First, fix a graph G (with q nodes) with a Hamiltonian cycle (We specify the graph to be used later). Then, in order to commit to 0, the committer commits to a random permutation of G using the underlying commitment scheme Com (and decommits by revealing the entire graph and the permutation). In order to commit to 1, the committer commits to a graph containing a randomly labeled q -cycle only (and decommits by opening the cycle only). Note that the ability to decommit to 0 and 1 implies that the committer knows a Hamiltonian cycle in G . On the other hand, given a Hamiltonian cycle in G , it is possible to generate commitments that are indistinguishable from legal ones, and yet have the property that one can decommit to both a 0 and a 1. Note that if the graph G is not hamiltonian, then this commitment scheme is a perfectly-binding computationally-hiding scheme.

The modifier graph based commitment scheme IDCom_G . We use the notation of [BS05] and some of the text has been taken verbatim from there [BS05]. Our graph-based scheme, introduced in [CLOS02], which we denote IDCom_G , differs from the [FS89] scheme above in the following way:

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 $c = \text{IDCom}_G(0)$. 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 $c = \text{IDCom}_G(1)$. 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 $\text{IDCom}_G(0)$ and $\text{IDCom}_G(1)$ 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 [FS89], 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 $c = \text{IDCom}_G(0)$ 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 $c = \text{IDCom}_G(m; r)$ to denote the function that generates a commitment to m using random coins r . Furthermore a commitment $c = \text{IDCom}_G(0^\kappa; r')$ to the zero string of length κ can be explained to any value m using the function $r = \text{IDOpen}_G(m, r', w)$, 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 [DSP92]. Furthermore, we will require that the scheme has pseudorandom ciphertexts. More formally:

Definition 2 (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 [Pas04] (which in turn builds on the work of Barak [Bar01]). Consider a “special” $\mathbf{NTIME}(T(k))$ relation R_{Sim} as follows.⁷ Let $k \in \mathbb{N}$ and let $T : \mathbb{N} \rightarrow \mathbb{N}$ 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. $c = \text{Com}(h(\Pi); s)$.
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 $\mathbf{NTIME}(T(k))$). 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) [BG08]. We note that the WI-UARG protocol of Barak and Goldreich [BG08] is public coin and the running time of the verifier in the protocol is bounded by a fixed polynomial.

5 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.

5.1 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 $x_1 \dots x_n$.⁸ **We stress that in these protocol executions each Q_i could 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.

⁷ 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 [Bar01]. 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 [BG08] for details.

⁸ 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.

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).

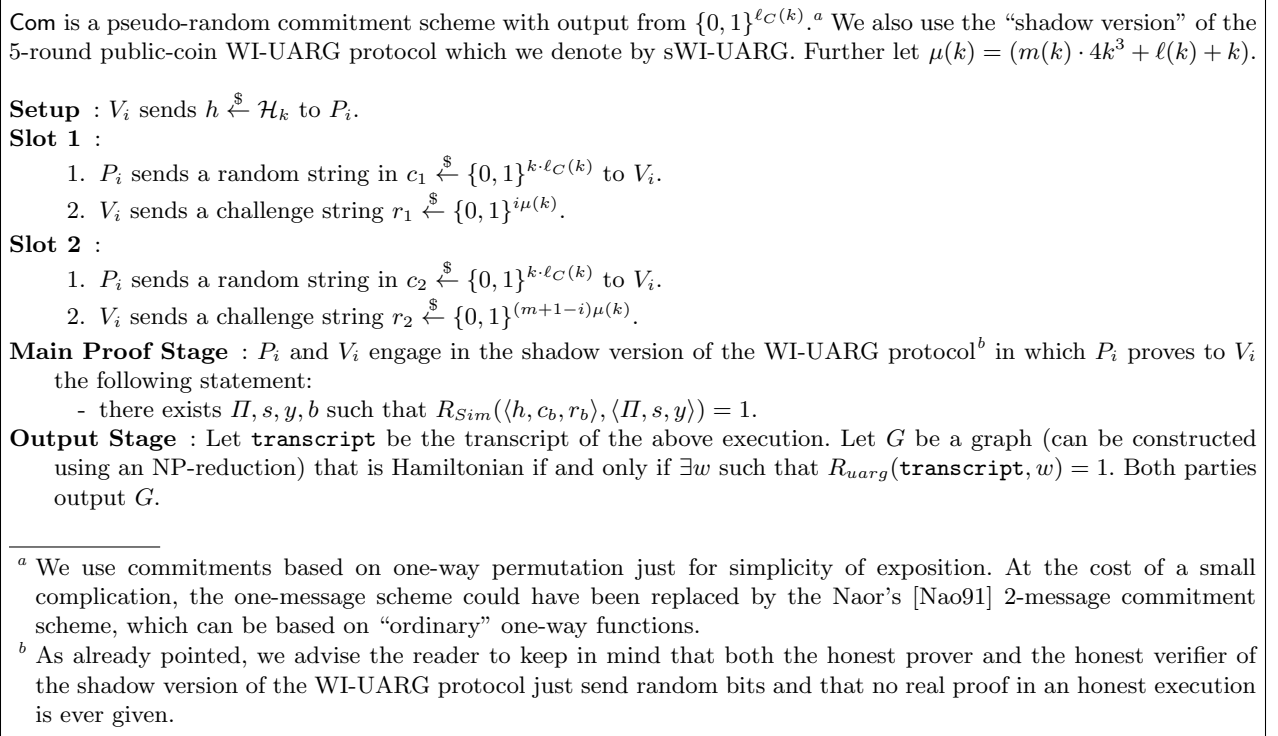


Fig. 1: $\langle P, V \rangle_i$ (the i^{th} protocol in the family of $m(k)$ protocols).

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 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 **transcript'** = $\text{unshadow}(\mathbf{transcript}, w)$ ⁹ denote the transcript obtained by replacing

⁹ Note that the function unshadow is inefficient and is used just to define the NP-Relation.

every $\ell_C(k)$ bit string in `transcript` that is sent by P with the corresponding committed bit (as per `Com`). Let $R_{uarg}(\text{transcript}, w) = 1$ if and only if $V(\text{unshadow}(\text{transcript}, w)) = 1$.

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 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` generated when the prover follows the simulated shadow prover strategy is such that there exists w such that $R_{uarg}(\text{transcript}, w) = 1$. 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, \dots, 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 next.

5.2 Our simulator $\mathcal{S}_{\langle P, V \rangle}$.

Recall that we are in the 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, P_i and V_i execute the protocol $\langle P, V \rangle_i$. In this setting we consider an adversary \mathcal{A} that adaptively corrupts parties of its choice. Note that whenever an honest party is corrupted it is required to reveal its input and random coins to the adversary. In our setting we have two kind of parties–

1. Q_1, \dots, Q_n : These parties have inputs but do not send any other messages. The sum of the lengths of the inputs of these parties is bounded by $\ell(k)$.
2. $P_1, \dots, P_m, V_1, \dots, V_m$: These parties do not have any inputs. Furthermore, all the messages sent by these parties are just random coins, and therefore, the protocol transcript, at any point, reveals all the random coins of all these parties used thus far. Therefore without loss of generality, we assume that the corruption of these parties does not reveal anything to the adversary.¹⁰

We will now describe our simulator $\mathcal{S}_{\langle P, V \rangle}$. We will describe the simulation strategy for each P_i and V_i separately. Our simulator maintains a list, **challenge seeds**, of seeds of a pseudorandom generator. It also maintains a special output tape, **special outputs**, on which it outputs Hamiltonian cycles in the graphs that different executions of $\langle P, V \rangle$ generate.

Simulating corruption requests. Whenever the adversary \mathcal{A} sends a request to corrupt a party Q_i for some $i \in [n]$, then our simulator $\mathcal{S}_{\langle P, V \rangle}$ obtains Q_i 's input x_i and sends it to the adversary. Note that Q_i does not send any messages and there is nothing else to handle.

On the other hand we next consider the case in which the adversary \mathcal{A} sends a request to corrupt a party P_i or a party V_i for some $i \in [m]$. As already pointed out, note that the protocol transcript reveals the random coins of P_i and V_i used so far. Furthermore these parties do not have any inputs. Therefore without loss of generality we can assume that on corruption these parties need not reveal anything to the adversary. However the adversary, from now on, gets full control of the corrupted party. More specifically, all future messages on behalf of the corrupted party are sent by \mathcal{A} .

¹⁰ For simplicity, we assume that on corruption honest parties do not need to reveal their future random coins to the adversary. The setting in which on corruption parties are forced into revealing all their future coins as well can be handled in the simulated setting by having the simulator send a pseudorandom string of appropriate length which can be succinctly represented by a k bit string.

Simulating V_i . Internally run the honest verifier strategy, for the protocol $\langle P, V \rangle_i$, but instead of sending a truly random strings as challenges (in Slot 1 and 2 of the protocol), pick a k bit long random string, write it to the special tape **challenge seeds**, expand it to the appropriate length of the challenge (r_1 or r_2) through a pseudorandom generator g and send the expanded string as the challenge.

Simulating P_i . We consider \mathcal{A} as a non-interactive algorithm, denoted by $\Pi_{\mathcal{A}}$, that gets as input $h_1, h_2 \dots h_i$, and outputs h_{i+1} , where h_{i+1} corresponds to the message that the \mathcal{A} generates given the history of messages $h_1, h_2 \dots h_i$. Now we modify this function slightly. Observe that the history of the messages $h_1, h_2 \dots h_i$ sent to \mathcal{A} , among other messages, consists of challenge strings (r_1 and r_2) that $\mathcal{S}_{\langle P, V \rangle}$ sends to the adversary \mathcal{A} on behalf of honest V_i 's. Let $\Pi'_{\mathcal{A}}$ be a function that is a slight modification of $\Pi_{\mathcal{A}}$. $\Pi_{\mathcal{A}}$ expects in the input sequence a sequence of messages $h_1, h_2 \dots h_i$. Some these messages correspond to challenge strings (which are long). Let the set of indices of these messages be \mathcal{I} . Our function $\Pi'_{\mathcal{A}}$ expects the same input except that instead of the challenge strings it expects a succinct description for these messages (i.e., a k bit string that can be expanded to the challenge string using a pseudorandom generator). $\Pi'_{\mathcal{A}}$ on input $h_1, h_2 \dots h_i$ outputs $\Pi_{\mathcal{A}}(f_1(h_1), \dots, f_i(h_i))$, where $f_t(h_t)$ for $t \in [i]$ outputs $g(h_t)$ (recall that g is the pseudorandom generator used previously) if $t \in \mathcal{I}$ and h_t otherwise. Furthermore let $\Pi(\rho, \tau)$ be a function that outputs $g(\tau)$ if $\rho = 0$ and outputs $\Pi'_{\mathcal{A}}(\tau)$ if $\rho = 1$. Next we describe the strategy of the simulator in simulating P_i . Note that in this interaction simulator (while sending messages on behalf of P_i) is interacting with V_i which is either controlled by the adversary \mathcal{A} or simulated by the simulator itself.

1. Receive the hash function h from V_i .
2. Generate a commitment $c_1 = \text{Com}(h(\Pi); s_1)$ and send it to the V_i in Stage 1 and receive back r_1 .
3. Similarly, generate a commitment $c_2 = \text{Com}(h(\Pi); s_2)$ and send it to V_i in Stage 2 and receive back r_2 .
4. In main proof stage, our simulator executes the simulated shadow prover strategy in the WI-UARG protocol using the witness $\langle \Pi, s_1, (\rho, \tau), 1 \rangle$, where (ρ, τ) depends on whether V_i was controlled by \mathcal{A} or not when r_1 , on behalf of V_i , was sent.
 - If V_i was not controlled by \mathcal{A} : Then the challenge r_1 corresponds to the output of Π on input $(0, \tau)$ where τ is a k bit long random string that $\mathcal{S}_{\langle P, V \rangle}$ (while simulating V_i) must have written on the tape **challenge seeds**.
 - If V_i was controlled by \mathcal{A} : Then the challenge r_1 corresponds to the output of Π on input $(1, \tau)$ where τ consists of all the messages $\mathcal{S}_{\langle P, V \rangle}$ sends to \mathcal{A} before the adversary \mathcal{A} (controlling party V_i) sends r_1 except the challenge strings that the simulator might have sent on behalf of any other honest V_j where $j \in [m]$ and $j \neq i$. Corresponding to each of these challenge strings τ consists of a k bit string instead, that can be appropriately expanded to the challenge string (of appropriate length using the pseudorandom generator g).

Excluding the challenges r_1 and r_2 , we note that the length of the P_i messages and the V_i messages in any $\langle P, V \rangle_i$ can be bounded by $2k^3$ if we use the specific WI-UARG of Barak and Goldreich [BG08]. Recall that we execute at most one instance of the protocol $\langle P, V \rangle_i$ for each $i \in [m]$. Also note that adaptive corruptions of Q_1, \dots, Q_n might require that an additional $\ell(k)$ bits be sent to the adversary. Then, the total length of messages sent by $\mathcal{S}_{\langle P, V \rangle}$, not including the challenges r_1 and r_2 , is bounded by $(m(k) \cdot 2k^3 + \ell(k))$. The challenges r_1, r_2 are long, but since they are constructed using a pseudorandom generator (with seed written on **challenge seeds**), they have a description that is bounded by k . Since the number of executions of $\langle P, V \rangle$ is bounded by $m(k)$ and since each execution contains only 2 challenges, it follows that the total length of the description in all cases is bounded by $\mu(k) - k$.

5. Let transcript_i be the transcript generated in the above execution of $\langle P, V \rangle_i$. If P_i is honest till the end of this execution then $\mathcal{S}_{\langle P, V \rangle}$ knows the opening of all commitments that are ever sent on behalf of P_i . Let w_i be the de-commitment information corresponding to all the $\ell_C(k)$ bit strings in transcript_i that are sent by the prover. Also let $\text{transcript}'_i = \text{unshadow}(\text{transcript}_i, w_i)$ denote the transcript obtained by replacing every $\ell_C(k)$ bit string in transcript_i that is sent by P with the corresponding committed bit (as per **Com**). Furthermore, from the completeness of the WI-UARG protocol it follows that the simulated shadow prover always succeeds in making the verifier accept. Therefore $\text{transcript}'_i$ is such that $V(\text{transcript}'_i, w_i) = 1$. This allows us to conclude that $R_{\text{uarg}}(\text{transcript}_i, w_i) = 1$. Additionally, simulator can use w_i to obtain a Hamiltonian cycle in the

graph (using the NP-reduction used to generate the graph itself) generated in this execution and outputs it on the `special outputs` tape.

5.3 Security of the $\langle P, V \rangle$ protocol

In this section we will argue some security properties about the simulator just constructed.

Lemma 1. *Consider 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 $\langle P, V \rangle_i$. Assuming collision resistant hash functions, for every adaptive adversary \mathcal{A} , the non-black box simulator $\mathcal{S}_{\langle P, V \rangle}$ (described above), obtains inputs of only those parties that \mathcal{A} corrupts, and:*

Indistinguishability: *The view of \mathcal{A} in its interaction with honest parties and view of \mathcal{A} in its interaction with $\mathcal{S}_{\langle P, V \rangle}$ are computationally indistinguishable.*

Extractability: *Except with negligible probability, $\mathcal{S}_{\langle P, V \rangle}$ additionally outputs (on the `special outputs` tape) a Hamiltonian cycle in the graph that P_i and V_i output in the execution $\langle P, V \rangle_i$ for every $i \in [m]$ for which P_i has not been corrupted.*

Proof. We show the first claim on indistinguishability of the outputs by a sequence of hybrid experiments.

- H_0 : This hybrid corresponds to the setting of honest parties interacting with the adversary \mathcal{A} .
- H_1 : Same as the previous hybrid, except that we start simulating all honest V_i s as specified in the description of the simulator. More specifically, for an honest V_i in an execution of the protocol $\langle P, V \rangle_i$, we internally follow the honest V_i strategy, but instead of sending truly random strings as challenges (in Slot 1 and 2 of the protocol), we pick a k bit long random string, write it to the special tape `challenge seeds`, expand it to the appropriate length of the challenge (r_1 or r_2) through a pseudorandom generator g and send the expanded string as the challenge.

The only difference between hybrids H_1 and H_0 is in the use pseudorandom strings instead of truly random strings. Therefore the outputs of the two hybrids are computationally indistinguishable.

- H_2 : Same as the previous hybrid, except that we start simulating honest P_i s as specified in the description of the simulator. Recall that an honest P_i just sends random strings in $\{0, 1\}^{\ell_C(k)}$. However, in the simulated setting instead of sending random strings, the simulator instead sends commitments using a pseudorandom commitment scheme.

Again the only difference between hybrid H_2 and H_1 is in the use of pseudorandom commitment strings instead of random strings. Therefore the outputs of the two hybrids are computationally indistinguishable.

Observe that the hybrid H_2 corresponds to the actual strategy of the simulator. This proves that the outputs of \mathcal{A} in its interaction with honest parties and view of \mathcal{A} in its interaction with $\mathcal{S}_{\langle P, V \rangle}$ are computationally indistinguishable.

Observe that for all executions $\langle P, V \rangle_i$ for $i \in [m]$ for which P_i has not been corrupted the simulator obtains a valid witness to execute the simulated shadow version of the WI-UARG. Therefore unless P_i is corrupted our simulation can always make V_i accept in the execution of $\langle P, V \rangle_i$. This follows from the completeness of the WI-UARG protocol. Finally, the decommitment information corresponding to the commitments sent by $\mathcal{S}_{\langle P, V \rangle}$ (on behalf of P_i , if it remains honest till the end of the execution) can be used to deduce a Hamiltonian cycle (as explained in the description of the simulator) in the graph the execution of $\langle P, V \rangle_i$ outputs.

Lemma 2. *Consider 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 $\langle P, V \rangle_i$. Consider the following game between an adaptive adversary \mathcal{A} and the non-black box simulator $\mathcal{S}_{\langle P, V \rangle}$ (described above) for some $i \in [m]$.*

1. *The adversary \mathcal{A} interacts with the simulator $\mathcal{S}_{\langle P, V \rangle}$ corrupting parties adaptively. However it is not allowed to corrupt V_i .*
2. *After the end of the interaction, for every $j \in [m]$ such that $j \neq i$ for which P_j has not been corrupted, $\mathcal{S}_{\langle P, V \rangle}$ provides \mathcal{A} with a Hamiltonian cycle in the graph that P_j and V_j output in the execution $\langle P, V \rangle_j$.*

Let E be the event that \mathcal{A} outputs a Hamiltonian cycle in the graph that P_i and V_i output in the execution $\langle P, V \rangle_i$. Adversary \mathcal{A} is said to violate special soundness if E happens with non-negligible probability. We claim that no PPT adversary (assuming collision resistant hash functions) can violate special soundness.

Proof. We start by observing that the adversary's ability to corrupt parties adaptively does not help it in violating special soundness in any way. More specifically, we argue that if there exists an adaptive adversary \mathcal{A} that violates special soundness then there exists a *special* adversary \mathcal{A}' that also violates special soundness. The special adversary, however, does not corrupt parties adaptively. Furthermore, a special adversary does not corrupt any P_j for $j \in [m]$ such that $j \neq i$. Additionally before the start of the interaction it corrupts V_j for all $j \in [m]$ such that $j \neq i$ and corrupts Q_j for all $j \in [n]$.

Next, we provide this transformation. Our adversary \mathcal{A}' starts by corrupting P_i, V_j for all $j \in [m]$ such that $j \neq i$ and Q_j for all $j \in [n]$. \mathcal{A}' also obtains the inputs x_1, \dots, x_n after it corrupts parties Q_j for all $j \in [n]$. Let \mathcal{T}' be the set of these corrupted parties. Our adversary \mathcal{A}' internally simulates \mathcal{A} . Also let, \mathcal{T} at any point be the set of parties corrupted by \mathcal{A} . We first describe how our adversary \mathcal{A}' handles corruption requests of \mathcal{A} . If \mathcal{A} requests to corrupt Q_j then provide x_j to \mathcal{A} . On the other hand if \mathcal{A} corrupts any other party (i.e., V_j for some $j \in [m]$ such that $j \neq i$ or P_j for some $j \in [n]$) then that party is added to \mathcal{T} . Every message sent on behalf of T by \mathcal{A} such that $T \in \mathcal{T}'$ is forwarded by \mathcal{A}' to the simulator $\mathcal{S}_{\langle P, V \rangle}$. On the other hand if $\mathcal{S}_{\langle P, V \rangle}$ expects a message from a party $T \in \mathcal{T}'$ such that $T \notin \mathcal{T}$, then T corresponds to a party V_j for $j \neq i$ that \mathcal{A} has not corrupted and \mathcal{A}' needs to send a message on V_j 's behalf. In this case \mathcal{A}' generates the next message for V_j using the honest V_j strategy and sends it both to honest P_j and the adversary \mathcal{A} . On the other hand, any message sent on behalf of T by the simulator $\mathcal{S}_{\langle P, V \rangle}$ such that $T \notin \mathcal{T}$ is forwarded by \mathcal{A}' to \mathcal{A} . Any other message sent on behalf of T by the simulator $\mathcal{S}_{\langle P, V \rangle}$ will be such that T corresponds to some party P_j where $j \neq i$ that \mathcal{A} has internally chosen to corrupt. Our adversary \mathcal{A}' just ignores this message. Finally we deal with the case in which \mathcal{A} expects a message from a party $T \in \mathcal{T}'$ such that $T \notin \mathcal{T}$. In this case T can correspond to P_i ¹¹ and \mathcal{A}' needs to send a message on behalf of P_i . \mathcal{A}' chooses this message according to the honest P_i strategy and sends it on behalf of P_i . After the end of the interaction, for every $j \in [m]$ such that $j \neq i$, $\mathcal{S}_{\langle P, V \rangle}$ provides \mathcal{A}' with a Hamiltonian cycle in the graph that P_j and V_j output in the execution $\langle P, V \rangle_j$. For every P_j such that $P_j \notin \mathcal{T}$, \mathcal{A}' forwards the Hamiltonian cycle in the graph that P_j and V_j output in the execution $\langle P, V \rangle_j$ to the adversary \mathcal{A} . Finally, \mathcal{A}' outputs the output of \mathcal{A} as its output.

Now that we have sketched our transformation, we need to argue that if \mathcal{A} succeeds in violating special soundness then so does \mathcal{A}' . Observe that the view of the adversary \mathcal{A} in direct interaction with $\mathcal{S}_{\langle P, V \rangle}$ and the view of the adversary \mathcal{A} in interaction with \mathcal{A}' (that subsequently interacts with $\mathcal{S}_{\langle P, V \rangle}$) is the same except with respect to the messages sent on behalf of P_i , before it is corrupted and with respect to the messages sent on behalf of uncorrupted V_j 's. In the direct interaction between adversary \mathcal{A} and $\mathcal{S}_{\langle P, V \rangle}$ all these aforementioned messages correspond to pseudorandom strings. On the other hand, \mathcal{A}' sends random strings instead. Therefore, the two distributions are computationally indistinguishable.

Now, we are left to show that no special adversary \mathcal{A}' can violate special soundness. We will argue this in two steps.

1. We say that an adversary P^* violates *stand-alone soundness* of $\langle P, V \rangle_i$ if P^* can interact with an honest V_i and also output (with non-negligible probability) a Hamiltonian cycle in the graph that the execution of $\langle P, V \rangle_i$ yields. We first start by pointing out that no PPT adversary can violate stand-alone soundness of $\langle P, V \rangle_i$. We show that if any adversary P^* can violate stand-alone soundness of $\langle P, V \rangle_i$ then we can contradict the stand-alone soundness of the zero knowledge protocol of Pass.
2. Next we show that any special adversary \mathcal{A}' that violates special soundness can be reduced to an adversary that violates stand-alone soundness of $\langle P, V \rangle_i$, thereby reaching a contradiction.

We first start by pointing out that the protocol $\langle P, V \rangle_i$ is stand-alone sound. Pass constructs a family of zero-knowledge protocols $\{cZK_1, \dots, cZK_m\}$ such that each of them is stand-alone sound. Our protocol $\langle P, V \rangle_i$ is the same as that of Pass, except one difference. The main proof stage of our protocol $\langle P, V \rangle_i$ consists of the shadow version of the WI-UARG protocol, while cZK_i just consists of the (normal) WI-UARG protocol. In other words, the main proof stage of our protocol $\langle P, V \rangle_i$ involves sending commitments

¹¹ It could also correspond to some V_j that has not been corrupted by V_j but this case has already been handled above.

of the messages (during the main proof stage) that would be sent in the $c\mathcal{ZK}_i$. Additionally, there is no common input that parties get in $\langle P, V \rangle_i$ while in $c\mathcal{ZK}_i$ the prover and the verifier expect to receive a theorem statement that the prover eventually proves to the verifier. Note that we consider executions of $c\mathcal{ZK}_i$ where the prover is proving a theorem statement that is vacuously false. In other words the theorem statement does not have any witnesses. In further discussion we restrict ourselves to only such protocols $c\mathcal{ZK}_i$ and of course soundness holds against any cheating prover of this protocol.

Lets consider an adversary P^* that violates stand-alone soundness of $\langle P, V \rangle_i$. More specifically, consider an adversary P^* that interacts with an honest V_i and outputs a Hamiltonian cycle in the graph that this execution of $\langle P, V \rangle_i$ yields with probability p_0 . We will use this P^* to construct an adversarial prover P^{**} , that internally simulates P^* and violates the stand-alone soundness of the zero knowledge protocol $c\mathcal{ZK}_i$. More specifically, P^{**} succeeds in making an honest verifier \mathcal{V} of $c\mathcal{ZK}_i$ accept a false theorem statement. Before the main proof stage, P^{**} just relays the messages between the internally simulated P^* and \mathcal{V} . However, for the messages of the main proof stage, P^* only provides commitments of the messages that \mathcal{V} expects. Therefore we will have to extract the messages from the commitments that P^* sends.

We start by building some notation. Let c_i denote the commitment of message m_i that P^* sends in round i of the main proof stage. In response to this message P^* expects to receive a value r_i . Note that i varies from 1 to t , where t is the number of rounds in the main proof stage. Note that t is constant. Furthermore, all the responses r_1, \dots, r_t consist of just random strings as the main proof stage only consists of a public coin WI-UARG protocol. Now we will describe how P^{**} proceeds in the main proof stage. For every $i \in [t]$, P^{**} proceeds as follows:

1. After P^{**} obtains c_i from P^* , it halts the “main thread” and starts a look-ahead thread with P^* sending freshly chosen random values r_i^1, \dots, r_i^i to it. (Recall that the protocol is public coin and we can proceed in this manner.) If P^* completes this execution and outputs a Hamiltonian cycle in the graph generated, then P^{**} can use the cycle to extract m_i from c_i (the cycle in the graph corresponds to the decommitment information for the commitments sent in the execution). If P^* aborts at any point in the look ahead thread then P^{**} also aborts.
2. P^{**} sends the extracted value m_i to \mathcal{V} to which \mathcal{V} responds with r_i . P^{**} forwards the received value r_i to P^* in the “main thread.”

We now need to argue that P^{**} violates the stand-alone soundness of the zero knowledge protocol $c\mathcal{ZK}_i$ with a non-negligible probability. Let p_i be the conditional probability (over the random choices of r_i, \dots, r_t) that P^* outputs a Hamiltonian cycle in the graph generated in the execution given the transcript of the execution (in the “main thread”) so far. By a simple counting argument we claim that for every i the probability p_i for $i \in \{1, \dots, t\}$ (over the choices of r_i, \dots, r_t) is greater than $p_{i-1}/2$ with probability at least $p_{i-1}/2$ (over the choices of the prefix of the transcript before message r_i). Therefore the probability $p_i = \frac{p_0}{2^i}$. Finally, the probability that P^{**} succeeds is at least $p_1^2 \cdot p_2^2 \cdot \dots \cdot p_{t-1}^2 \cdot p_t^3$, which is non-negligible given that p_0 is non-negligible.

Now we are left to argue that if a special adversary \mathcal{A}' violates special soundness, then we can use it to construct an adversary \mathcal{A}'' that violates stand-alone soundness of $\langle P, V \rangle_i$. This follows in a way very similar to Pass [Pas04] and we just sketch an outline of the proof here. We will argue this in two steps. First we will construct an adversary \mathcal{A}'' that violates stand-alone soundness of $\langle P, V \rangle_i$ in interaction with a honest V_i that obtains its randomness for generating the challenges r_1 and r_2 using a pseudorandom generator. We will denote this modified V_i strategy by mV_i . Next we can argue that \mathcal{A}'' violates the stand-alone soundness of $\langle P, V \rangle_i$ in interaction with an honest V_i as well. This follows from the security of the pseudo-random generator.

Now, we elaborate on the first part of the argument. Observe that, \mathcal{A}' in its interaction with $\mathcal{S}_{\langle P, V \rangle}$ (where $\mathcal{S}_{\langle P, V \rangle}$ generates the messages on behalf of V_i) outputs a Hamiltonian cycle in the graph generated in the execution of $\langle P, V \rangle_i$. We now describe our adversary \mathcal{A}'' whose goal is to violate stand-alone soundness of $\langle P, V \rangle_i$ in interaction with an external honest mV_i . \mathcal{A}'' internally executes $\mathcal{S}_{\langle P, V \rangle}$ and \mathcal{A}' . However, $\mathcal{S}_{\langle P, V \rangle}$ instead of generating messages on behalf of V_i on its own will have to externally obtain messages from an external party mV_i (playing as V_i). Unfortunately, in this case \mathcal{A}'' will not be able to run $\mathcal{S}_{\langle P, V \rangle}$ because in order to simulate P_j for $j \neq i$, $\mathcal{S}_{\langle P, V \rangle}$ needs a short description of the messages sent on behalf of honest parties, including the ones sent by the external party mV_i . In the above scenario, however, since the execution $\langle P, V \rangle_i$ is externally forwarded to mV_i , the simulator does not have a short description of the honest verifier’s

challenges r_1, r_2 . To solve this problem we construct a modified simulator \mathcal{E} (modification of $\mathcal{S}_{\langle P, V \rangle}$). We now describe the simulator \mathcal{E} that has to simulate messages on behalf of all P_j where $j \neq i$. Observe that except for the “long” challenges r_1, r_2 sent by the external party mV_i the simulator has a description of all messages sent to the adversary that is shorter than $\ell(k) - k$. However, in order to succeed in the main proof stage of $\langle P, V \rangle_j$ for $j \neq i$, even in the presence of “long” messages (to which the simulator does not have a short description), we use the fact that it is sufficient to have a short description of the messages sent in just *one* of the slots of $\langle P, V \rangle_j$. We consider two cases:

- If both the “long” challengers r_1, r_2 sent by the external party mV_i are contained in the same slot (or if they are not contained in any of the two slots), then the simulator can always use the empty slot in order to perform the simulation.
- Assume, on the other hand, that r_1 is contained in Slot 1, and r_2 is contained in Slot 2. By construction of the protocols it follows that the length of either the first or the second challenge in PV_j is at least $\mu(k)$ longer than the corresponding challenge in $\langle P, V \rangle_i$ (sent by the external party mV_i). Thus there exists a slot in $\langle P, V \rangle_j$ such that even if we include the external party mV_i ’s challenge, from the protocol $\langle P, V \rangle_i$, in whole, in the description, we still have $\mu(k) - k$ to describe the other messages, which means that the simulation can be performed.

Note that the only difference between $\mathcal{S}_{\langle P, V \rangle}$ and \mathcal{E} is the choice of witness used in the main proof stage. While $\mathcal{S}_{\langle P, V \rangle}$ always picks the first slot, \mathcal{E} picks the “appropriate” slot (which can be any of the two). Thus if \mathcal{A}' , when interacting with $\mathcal{S}_{\langle P, V \rangle}$ succeeds in outputting a Hamiltonian cycle in the graph generated in $\langle P, V \rangle_i$ then \mathcal{A}' when interacting with \mathcal{E} , will also succeed in outputting a Hamiltonian cycle in the graph generated in $\langle P, V \rangle_i$, or else the witness indistinguishability of the WI-UARG used in $\langle P, V \rangle_i$ would be broken. From this we conclude that \mathcal{A}' breaks the stand-alone soundness of $\langle P, V \rangle_i$. This concludes the proof.

5.4 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 [DSP92]) 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 adversary \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 $\langle A, B \rangle$. (IDCom, 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 $c = \text{IDCom}_{G_1}(\alpha; r_1)$, 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 $c \neq \text{IDCom}_{G_1}(\alpha; r_1)$.
5. Both parties set $pk := \alpha \oplus \beta$.
6. A generates a commitment $d = \text{IDCom}_{G_1}(\gamma; r_2)$, where γ is a random string in $\{0, 1\}^{\ell_1(k)}$ and sends it to B .

7. B generates commitments $f_i = \text{IDCom}_{G_2}(\delta_i; s_i)$, 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 $d \neq \text{IDCom}_{G_1}(\gamma; r_2)$. Next, B sends δ_i, s_i, t_i for every $i \in [\ell'(k)]$ to A .
10. A aborts if for some $i \in [\ell'(k)]$, $f_i \neq \text{IDCom}_{G_2}(\delta_i; s_i)$ 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.¹² We provide full details on our simulator for the $\langle A, B \rangle$ protocols and the proof of the security properties next.

5.5 Our Simulator for the coin-flipping protocol

We will now describe our simulator $\mathcal{S}_{\langle A, B \rangle}$. Note that if A is not corrupted (at the start of the protocol) then the simulator obtains a Hamiltonian cycle w_1 in G_1 and hence it can equivocate on the commitments it sends on A 's behalf. Similarly, if B is not corrupted (at the start of the protocol) then the simulator obtains a Hamiltonian cycle w_2 in G_2 and hence it can equivocate on the commitments it sends on B 's behalf.

Note that at different points during the execution of the protocol the adversary can corrupt A and/or B . Before every step our simulator checks to see if the party that is supposed to send the next message is already corrupted. If this is indeed the case, then the simulator skips this message. Furthermore, after every message sent by the simulator the adversary can corrupt A and/or B . We describe the actions of the simulator, in case of such a corruption, after every step of the simulator. Further note that without loss of generality we can assume that parties on corruption reveal only those random coins that are not revealed by the transcript itself. We say that there is nothing to reveal when the random coins of the party, being corrupted, are revealed by the transcript itself. Finally, besides random coins, on corruption a party must also provide its input to the adversary. Similarly, during simulation our simulator also directly forwards the input of the corrupted party (that is given to the simulator on corruption of the party) to the adversary. The simulation of the inputs of parties can always be done as explained above and therefore we will skip mentioning it.

Our simulator starts by sampling a public-key secret-key pair $(pk', sk') \leftarrow G(1^k)$.

1. (Skip if A has been already corrupted): $\mathcal{S}_{\langle A, B \rangle}$ on behalf of A generates a commitment $c = \text{IDCom}_{G_1}(0^{\ell_1(k)}; r'_1)$. It sends c to B .
 - If A is corrupted: Set $\alpha \xleftarrow{\$} \{0, 1\}^{\ell_1(k)}$ and $r_1 := \text{IDOpen}_{G_1}(\alpha, r'_1, w_1)$. Reveal $\alpha \circ r_1$ as the random coins of A .
 - If B is corrupted: Nothing to reveal.
2. (Skip if B has been already corrupted): $\mathcal{S}_{\langle A, B \rangle}$ on behalf of B sends a random string $\beta \in \{0, 1\}^{\ell_1(k)}$ to A .
 - If A is corrupted: Set $\alpha \xleftarrow{\$} \{0, 1\}^{\ell_1(k)}$ and $r_1 := \text{IDOpen}_{G_1}(\alpha, r'_1, w_1)$. Reveal $\alpha \circ r_1$ as the random coins of A .
 - If B is corrupted: Nothing to reveal.
3. (Skip if A has been already corrupted): $\mathcal{S}_{\langle A, B \rangle}$ on behalf of A sets $\alpha := pk' \oplus \beta$ and $r_1 := \text{IDOpen}_{G_1}(\alpha, r'_1, w_1)$. A sends (α, r_1) to B .
 - A or B is corrupted: Nothing to reveal.
4. (Skip if B is corrupted) Abort the protocol if $c \neq \text{IDCom}_{G_1}(\alpha; r_1)$.
 - A or B is corrupted: Nothing to reveal.

¹² This is not a problem as rewinding is used only in the proof in order to reach a contradiction.

5. Set $pk := \alpha \oplus \beta$.
 - A or B is corrupted: Nothing to reveal.
6. (Skip if A is corrupted) $\mathcal{S}_{\langle A, B \rangle}$ on behalf of A generates a commitment $d = \text{IDCom}_{G_1}(0^{\ell'(k)}; r'_2)$ and sends it to B .
 - If A is corrupted: Set $\gamma \xleftarrow{\$} \{0, 1\}^{\ell'(k)}$ and $r_2 := \text{IDOpen}_{G_1}(\gamma, r'_2, w_1)$. Reveal $\gamma \circ r_2$ as the random coins.
 - If B is corrupted: Nothing to reveal.
7. (Skip if B is corrupted) $\mathcal{S}_{\langle A, B \rangle}$ on behalf of B generates commitments $f_i = \text{IDCom}_{G_2}(0, s'_i)$ for $i \in \{1 \dots \ell'(k)\}$. It also generates $e_{i,0} = E_{pk}(\text{IDOpen}_{G_2}(0, s'_i, w_2); u_i)$ and $e_{i,1} = E_{pk}(\text{IDOpen}_{G_2}(1, s'_i, w_2); u'_i)$. Finally it sends $f_i, e_{i,0}$ and $e_{i,1}$ for all $i \in [\ell'(k)]$ to A . We stress that this step is executed only if B is honest. This will affect our simulation later as we see next.
8. Now we describe the strategy of simulator depending on what parties are honest at this point.

Both A and B are honest: Set $\gamma \xleftarrow{\$} \{0, 1\}^{\ell'(k)}$ and $r_2 := \text{IDOpen}_{G_1}(\gamma, r'_2, w_1)$. Also set $\delta := \gamma \oplus \text{crs}$ and $s_i := \text{IDOpen}_{G_2}(\delta_i, s'_i, w_2)$, $t_i = (1 - \delta_i) \cdot u_i + \delta_i \cdot u'_i$. From now on our simulator $\mathcal{S}_{\langle A, B \rangle}$ generates messages on behalf of A using the honest A strategy and the values γ and r_2 derived above. Similarly, $\mathcal{S}_{\langle A, B \rangle}$ generates messages on behalf of B using the honest B strategy and the values δ_i, s_i and t_i derived above.

If A is honest but B is corrupted: For every $i \in [\ell'(k)]$, set $s_i = D_{sk'}(e_{i,0})$ and $s'_i = D_{sk'}(e_{i,1})$. Abort with a special abort **E-Abort** if $f_i = \text{IDCom}_{G_2}(0; s_i) = \text{IDCom}_{G_2}(1; s'_i)$. Otherwise, set $\delta_i = 0$ if $f_i = \text{IDCom}_{G_2}(0; s_i)$ or set $\delta_i = 1$ if $f_i = \text{IDCom}_{G_2}(1; s'_i)$. Set $\gamma := \delta \oplus \text{crs}$ and $r_2 := \text{IDOpen}_{G_1}(\gamma, r'_2, w_1)$. From now on our simulator $\mathcal{S}_{\langle A, B \rangle}$ generates messages on behalf of A using the honest A strategy and the values γ and r_2 derived above.

If A is corrupted but B is honest: If B is corrupted at this point then set $\delta \xleftarrow{\$} \{0, 1\}^{\ell'(k)}$ and $s_i := \text{IDOpen}_{G_2}(\delta_i, s'_i, w_2)$, $t_i = (1 - \delta_i) \cdot u_i + \delta_i \cdot u'_i$. Reveal $\delta_i \circ s_i \circ t_i$ as the random coins of B . Otherwise, receive (γ, r_2) from A (controlled by \mathcal{A}), abort if $d \neq \text{IDCom}_{G_1}(\gamma, r_2)$, and set $\delta := \gamma \oplus \text{crs}$ and $s_i := \text{IDOpen}_{G_2}(\delta_i, s'_i, w_2)$, $t_i = (1 - \delta_i) \cdot u_i + \delta_i \cdot u'_i$. From now on our simulator $\mathcal{S}_{\langle A, B \rangle}$ generates messages on behalf of B using the honest B strategy and the values δ_i, s_i and t_i derived above.

Both A and B are corrupted: Skip all future messages.

5.6 Security of the coin-flipping protocol

Lemma 3. *Consider an execution of the protocol $\langle A, B \rangle$ between two parties A and B on common input G_1 and G_2 and the (straight-line) simulator $\mathcal{S}_{\langle A, B \rangle}$ described above. $\mathcal{S}_{\langle A, B \rangle}$ receives Hamiltonian cycle in G_1 (resp., G) if A (resp., B) is honest and a random string crs of length $\ell'(k)$ as input. Assuming dense cryptosystems and that no PPT adversary can output a Hamiltonian cycle in G_1 (resp., G_2) if B (resp., A) is not corrupted before the start of the protocol. Then we claim that, for every adversary \mathcal{A} :*

Indistinguishability: *The view of the adversary \mathcal{A} in interaction with honest parties and the view of the adversary \mathcal{A} in interaction with $\mathcal{S}_{\langle A, B \rangle}$ are computationally indistinguishable. Furthermore, $\mathcal{S}_{\langle A, B \rangle}$ obtains inputs of only those parties that \mathcal{A} corrupts.*

Output: *If at least either A or B is not corrupted (till the end of the protocol) then the output of the protocol (except with negligible probability) is crs .*

Proof. Note that (as explained in Section 4) the commitments (as per IDCom_{G_1} or IDCom_{G_2}) generated by the adversary \mathcal{A} (on behalf of A or B) are going to be computationally binding as long as the party to which the commitments are sent is not corrupted. This follows directly from the assumption that no PPT adversary can output a Hamiltonian cycle in G_1 (resp., G_2) if B (resp., A) is not corrupted before the start of the protocol. We will use this property throughout our proof. We stress that in our proof we will rewind the adversary at different points but we stress that this is done only to reach a contradiction. Our simulator itself does not need to rely on rewinding.

Now, we consider a sequence of hybrids and establish our claim.

- H_0 : This hybrid corresponds to the setting in which the adversary interacts with the simulator $\mathcal{S}_{\langle A, B \rangle}$ that simulates all parties following honest party strategies. This hybrid is the same as the setting of honest parties interacting with the adversary directly.

- H_1 : (This change is only made if B is honest when Step 7 of the protocol is executed) Same as the previous hybrid except that our simulator generates commitments f_i (Step 7) differently. Our simulator samples δ randomly in $\{0, 1\}^{\ell'(k)}$ and sets $f_i = \text{IDCom}_{G_2}(0, s'_i)$ for every $i \in \{1 \dots \ell'(k)\}$. It also generates $s_i = \text{IDOpen}_{G_2}(\delta_i, s'_i, \mathbf{w}_2)$, $e_{i,\delta_i} = E_{pk}(s_i; t_i)$ and $e_{i,1-\delta_i}$ as a random string in $\{0, 1\}^{\ell_2(k)}$. Recall that this can be done because \mathcal{S} will know a Hamiltonian cycle \mathbf{w}_2 in the graph G_2 if it ever sends a commitment on behalf of B because that only happens if B is honest. From now on our simulator \mathcal{S} generates messages on behalf of B using the honest B strategy and the values δ_i , s_i and t_i derived above.

The computational indistinguishability of hybrids H_0 and H_1 follows from the pseudorandomness property (as explained in Section 4) of the underlying commitment scheme Com .

- H_2 : (This change is only made if B is honest when Step 7 of the protocol is executed) Same as the previous hybrid except that our simulator generates ciphertexts $e_{i,0}$ and $e_{i,1}$ differently. More specifically, our simulator samples δ randomly in $\{0, 1\}^{\ell'(k)}$ and sets $f_i = \text{IDCom}_{G_2}(0, s'_i)$ for every $i \in \{1 \dots \ell'(k)\}$. It also generates $e_{i,0} = E_{pk}(\text{IDOpen}_{G_2}(0, s'_i, \mathbf{w}_2); u_i)$ and $e_{i,1} = E_{pk}(\text{IDOpen}_{G_2}(1, s'_i, \mathbf{w}_2); u_i)$. Recall that this can be done because \mathcal{S} will know a Hamiltonian cycle \mathbf{w}_2 in the graph G_2 if it ever sends a commitment on behalf of B because that only happens if B is honest. Also set $s_i := \text{IDOpen}_{G_2}(\delta_i, s'_i, \mathbf{w}_2)$, $t_i = (1 - \delta_i) \cdot u_i + \delta_i \cdot u'_i$. From now on our simulator \mathcal{S} generates messages on behalf of B using the honest B strategy and the values δ_i , s_i and t_i derived above.

In Step 7 of the protocol, for each $i \in [\ell'(k)]$, we generate two ciphertexts $e_{i,0}$ and $e_{i,1}$. Recall that in hybrid H_1 , e_{i,δ_i} is a valid ciphertext while $e_{i,1-\delta_i}$ is a random string. On the other hand, in hybrid H_2 , both e_{i,δ_i} and $e_{i,1-\delta_i}$ are valid ciphertexts. We can argue indistinguishability by considering a sequence of $\ell'(k) + 1$ hybrids, $H_{1,0}, \dots, H_{1,i}, \dots, H_{1,\ell'(k)}$ from H_1 to H_2 . In hybrid $H_{1,i}$ for every $0 < j \leq i$ both e_{j,δ_j} and $e_{j,1-\delta_j}$ are valid ciphertexts and on the other hand for every $i < j \leq n$, e_{j,δ_j} is a valid ciphertext while $e_{j,1-\delta_j}$ is a random string. Hybrid $H_{1,0}$ is same as hybrid H_1 and hybrid $H_{1,\ell'(k)}$ is same as hybrid H_2 . Now, we will sketch the proof for indistinguishability between hybrids $H_{1,i-1}$ and $H_{1,i}$. The argument for the rest of the hybrids follows in a similar manner. For the sake of contradiction let's assume that there does exist an adversary \mathcal{A} and a distinguisher D such that D can distinguish between the two case. Then, we will construct an adversary C that can break the pseudorandomness property of the encryption scheme. Our adversary externally gets the public key pk' as input and needs to force the value pk set in the Step 5 of the protocol to be this value pk' . The strategy of B for this will depend of whether A was honest at the time when the message corresponding to Step 1 was sent. If it was honest then C just sets $\beta = \alpha \oplus pk'$ (Note that \mathcal{S} knows the value α). On the other hand, if A is malicious then, our simulator before sending β , starts a look ahead thread (can be repeated multiple times in case of aborts) with \mathcal{A} by sending β' and thereby obtaining α . It now obtains $\beta = \alpha \oplus pk'$ and sends β to A on behalf of B in Step 2. Since the commitment scheme IDCom_{G_1} is computationally binding, the adversary \mathcal{A} can decommit c to the same α only, thus forcing the value pk generated in the protocol to be pk' . Finally, C obtains a challenge ciphertext externally which is either a valid encryption of $\text{IDOpen}_{G_2}(1 - \delta_i, s'_i, \mathbf{w}_2)$ or a random string and uses the challenge string as $e_{i,1-\delta_i}$. This distribution corresponds to hybrid $H_{1,i-1}$ if the challenge string is a random string and to hybrid $H_{1,i}$ in the other case. Hence the ability of the distinguisher to distinguish between $H_{1,i-1}$ and $H_{1,i}$ directly translates to C 's ability in distinguishing valid ciphertexts from random strings.

- H_3 : Same as the previous hybrid except that our simulator generates commitments c and d differently. We will describe the change for c only. Our simulator changes generation of d in a similar manner. Our simulator samples α randomly in $\{0, 1\}^{\ell_1(k)}$ and sets $c = \text{IDCom}_{G_1}(0^{\ell_1(k)}; r'_1)$. It also generates $r_1 = \text{IDOpen}_{G_1}(\alpha, r'_1, \mathbf{w}_1)$. Recall that this can be done because \mathcal{S} will know a Hamiltonian cycle \mathbf{w}_1 in the graph G_1 if it ever sends a commitment on behalf of A because that only happens if A is honest. From now on our simulator \mathcal{S} generates messages on behalf of A using the honest A strategy and the values α and r_1 derived above.

The computational indistinguishability of hybrids H_2 and H_3 follows from the pseudorandomness property of the underlying commitment scheme Com .

- H_4 : If A is not corrupted before Step 3 starts, then \mathcal{S} needs to send (α, r) on behalf of A . In this case \mathcal{S} sets $\alpha := pk' \oplus \beta$ and $r_1 := \text{IDOpen}_{G_1}(\alpha, r'_1, \mathbf{w}_1)$ and sends (α, r_1) to B . Thereby forcing the pk generated in the protocol execution to be pk' .

The computational indistinguishability of hybrids H_3 and H_4 follows from the pseudorandomness property of the public keys of the public key encryption scheme.

- H_5 : If B is corrupted and A is honest when the message on behalf of B is sent in Step 7, then for every $i \in [\ell'(k)]$, set $s_i = D_{sk'}(e_{i,0})$ and $s'_i = D_{sk'}(e_{i,1})$. Abort with a special abort message **E-Abort** if $f_i = \text{IDCom}_{G_2}(0; s_i) = \text{IDCom}_{G_2}(1; s'_i)$. Otherwise, set $\delta_i = 0$ if $f_i = \text{IDCom}_{G_2}(0; s_i)$ or set $\delta_i = 1$ if $f_i = \text{IDCom}_{G_2}(1; s_i)$.

The computational indistinguishability of hybrids H_4 and H_5 follows from the fact that **E-Abort** happens with a negligible probability which in turn follows the fact that the commitment IDCom_{G_2} is conditionally binding for the adversary.

- H_6 : If B is corrupted and A is honest when the messages in Step 7 is sent then set $\gamma := \delta \oplus \text{crs}$ and $r_2 := \text{IDOpen}_{G_1}(\gamma, r'_2, \mathbf{w}_1)$. Furthermore, from now on our simulator \mathcal{S} generates messages on behalf of A using the honest A strategy and the values γ and r_2 derived above.

The computational indistinguishability of hybrids H_5 and H_6 follows from the fact that crs is chosen randomly.

- H_7 : If B is honest when the message in Step 9 is sent then set $\delta := \gamma \oplus \text{crs}$ and $s_i := \text{IDOpen}_{G_1}(\delta_i, s'_i, \mathbf{w}_1)$, $t_i = (1 - \delta_i) \cdot u_i + \delta_i \cdot u'_i$. From now on our simulator \mathcal{S} generates messages on behalf of B using the honest B strategy and the values δ_i, s_i and t_i derived above.

The computational indistinguishability of hybrids H_6 and H_7 follows from fact that the crs is chosen randomly.

Note that the simulation strategy in hybrid H_7 correspond to the strategy of our simulator itself. This completes the proof of indistinguishability. Also observe that the output of the protocol is always (except with negligible probability) crs if at least either A or B is not corrupted.

6 Our Constant Round Protocol

Let f be any adaptively well-formed¹³ functionality. In this section we will give a constant round protocol Π that adaptively-securely realizes f . Let Q_1, \dots, Q_n be n parties that hold inputs x_1, \dots, x_n respectively. Let f be the function that they wish to evaluate on their inputs. Furthermore, let $\ell(k) = |x_1| + |x_2| \dots |x_n|$.

We start by describing a protocol that adaptively securely realizes the $\mathcal{F}_{n-\text{crs}}$ 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 5.1) and a *coin flipping protocol* $\langle A, B \rangle$ (Section 5.4). 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 $\text{crs}_{i,j}$. All these executions are done in parallel.

Theorem 2. *Assuming collision resistant hash functions and dense cryptosystems [DSP92], the constant round protocol just described above adaptively securely evaluates $\mathcal{F}_{n-\text{crs}}$ (Figure 2).*

Proof. To prove the above theorem we begin by describing our simulator \mathcal{S} . The simulator maintains a list A of corrupted parties.

1. For the trapdoor creation phase, \mathcal{S} initiates the simulator $\mathcal{S}_{\langle P, V \rangle}$ for the parallel executions of the $\langle P, V \rangle$ protocols. During this adversary can adaptively corrupt any number of parties. Furthermore by Lemma 1, if Q_i is not corrupted (by the end of the trapdoor creation phase) then $\mathcal{S}_{\langle P, V \rangle}$ outputs $\mathbf{w}_{i,j}$ (on the **special outputs** tape) where $\mathbf{w}_{i,j}$ is a Hamiltonian cycle in the graph $G_{i,j}$.

¹³ We require[CLOS02] that the functionality reveals its random input in case all parties are corrupted.

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 $\exists (crs, i, j, crs_{i,j}) \in \mathcal{L}$: Sends $crs_{i,j}$ to Q .
 - If $(crs, i, j, \cdot) \notin \mathcal{L}$ 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 $(crs, i, j, crs_{i,j})$ 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 $(corrupt, Q_i)$ 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}

2. In the coin flipping phase, for all $i, j \in [n]$ such that $i < j$, \mathcal{S} initiates the simulator $\mathcal{S}_{\langle A, B \rangle}$ to simulate $\langle A, B \rangle^{i,j}$ instead of following the honest party strategies. We denote this instance of the simulator by $\mathcal{S}_{\langle A, B \rangle}^{i,j}$. Furthermore, \mathcal{S} can provide (from the **special outputs** tape) $\mathcal{S}_{\langle A, B \rangle}^{i,j}$ with a Hamiltonian cycle in the graph $G_{i,j}$ if Q_i is honest. Similarly, \mathcal{S} can also provide $\mathcal{S}_{\langle A, B \rangle}^{i,j}$ with a Hamiltonian cycle in the graph $G_{j,i}$ if Q_j is honest. It also provides $\mathcal{S}_{\langle A, B \rangle}^{i,j}$ with $crs_{i,j}$ obtained from \mathcal{F}_{n-crs} .

Now we will prove indistinguishability by a hybrid argument.

- H_0 : This hybrid corresponds to the setting of honest parties interacting with the adversary \mathcal{A} .
- H_1 : Same as the previous hybrid, except that we use simulator $\mathcal{S}_{\langle P, V \rangle}$ to interact with the adversary during the trapdoor creation phase.

The only difference in hybrids H_0 and H_1 is that \mathcal{S} use $\mathcal{S}_{\langle P, V \rangle}$ instead of following the honest party strategies for the trapdoor creation phase. The indistinguishability of the two hybrids follows from Lemma 1. Before we proceed to the next hybrid we point out the following fact.

- By Lemma 2, if Q_j is not corrupted (by the end of the trapdoor creation phase) then \mathcal{A} can not output a Hamiltonian cycle in the graph $G_{i,j}$ for any $i \in [n] \setminus \{j\}$, even when it is provided with a Hamiltonian cycle in all other graphs $G_{i',j'}$ (i.e., $i' \neq i, j' \neq j$) for every $i', j' \in [n]$ such that $i' \neq j'$, and $Q_{i'}$ is not corrupted.
- H_2 : Same as the previous hybrid, except that in the coin flipping phase, for all $i, j \in [n]$ such that $i < j$, \mathcal{S} instead of following the honest party strategies, executes an instance of $\mathcal{S}_{\langle A, B \rangle}$ to simulate $\langle A, B \rangle^{i,j}$ instead of following the honest party strategies. We denote this instance of the simulator by $\mathcal{S}_{\langle A, B \rangle}^{i,j}$. Furthermore, \mathcal{S} can provide (from the **special outputs** tape) $\mathcal{S}_{\langle A, B \rangle}^{i,j}$ with a Hamiltonian cycle in the graph $G_{i,j}$ if Q_i is honest. Similarly, \mathcal{S} can also provide $\mathcal{S}_{\langle A, B \rangle}^{i,j}$ with a Hamiltonian cycle in the graph $G_{j,i}$ if Q_j is honest. It also provides $\mathcal{S}_{\langle A, B \rangle}^{i,j}$ with $crs_{i,j}$ obtained from \mathcal{F}_{n-crs} .

We will argue indistinguishability of hybrids H_1 and H_2 by considering a sequence $H_{1,0}, \dots, H_{1,t}$ of $t = n(n-1)/2$ hybrids. In hybrid $H_{1,p}$ the simulator \mathcal{S} uses an instance of $\mathcal{S}_{\langle A, B \rangle}$ to simulate $\langle A, B \rangle^{i,j}$ for every $i, j \in [n]$ such that $i < j$ and $i(n-1) + j < p$ while it follows honest party strategies in simulation other instances of $\langle A, B \rangle$ protocol. Observe that the hybrid $H_{1,0}$ is the same as hybrid H_1 and hybrid $H_{1,t}$ is the same as H_2 . We are now now argue that hybrids $H_{1,p-1}$ and $H_{1,p}$ (where $1 < p \leq n(n-1)/2$) are indistinguishable. The only difference between $H_{1,p-1}$ and $H_{1,p}$ is that one addition instance of $\langle A, B \rangle$ is simulated in $H_{1,p}$ using $\mathcal{S}_{\langle A, B \rangle}$. Observe that all our \mathcal{S} can generate all messages the messages sent in all other execution on its own while obtaining the messages for this extra instance for an external party. Next we can argue that the ability of any distinguisher to distinguish between $H_{1,p-1}$ can $H_{1,p}$ can be used directly to contradict Lemma 3. Furthermore from Lemma 3 it also follows that the crs generated in this execution would match the one provided by \mathcal{S} .

This concludes the proof.

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 [IPS08] 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 [CLOS02], given access to a common random string, we can construct an adaptively secure OT protocol. Using this protocol and applying the UC composition theorem [Can00] (Composing Different Setups, Page 61), multiple times, we can construct a protocol that achieves adaptively secure OT¹⁴ 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 [IPS08] we can adaptively securely realize any functionality.

Acknowledgements

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

- [Bar01] B. Barak. How to go beyond the black-box simulation barrier. In *Proc. 42nd FOCS*, pages 106–115, 2001.
- [Bea96a] Donald Beaver. Adaptive zero knowledge and computational equivocation (extended abstract). In *STOC*, pages 629–638, 1996.
- [Bea96b] Donald Beaver. Equivocable oblivious transfer. In *EUROCRYPT*, pages 119–130, 1996.
- [Bea98] Donald Beaver. Adaptively secure oblivious transfer. In *ASIACRYPT*, pages 300–314, 1998.
- [BG08] Boaz Barak and Oded Goldreich. Universal arguments and their applications. *SIAM J. Comput.*, 38(5):1661–1694, 2008.
- [BH92] Donald Beaver and Stuart Haber. Cryptographic protocols provably secure against dynamic adversaries. In *EUROCRYPT*, pages 307–323, 1992.
- [BL04] Boaz Barak and Yehuda Lindell. Strict polynomial-time in simulation and extraction. *SIAM J. Comput.*, 2004.
- [Blu87] Manuel Blum. How to prove a theorem so no one else can claim it. In *Proceedings of the International Congress of Mathematicians*, pages 1444–1451, 1987.
- [BS05] Boaz Barak and Amit Sahai. How to play almost any mental game over the net - concurrent composition via super-polynomial simulation. In *FOCS*, pages 543–552, 2005.

¹⁴ [CLOS02] assume trapdoor permutations and augmented non-committing encryption [CFG96] in their construction. The augmentation of non-committing encryption consists of two additional properties. First, the encryption scheme should have an alternative key generation algorithm that generates only public encryption keys without the corresponding decryption key. Second, the standard and additional key generation algorithms should be invertible in the sense that given the output key or keys, it is possible to find the random coin tosses used in generating these keys.

- [Can98] Ran Canetti. Security and composition of multi-party cryptographic protocols. Cryptology ePrint Archive, Report 1998/018, 1998. <http://eprint.iacr.org/>.
- [Can00] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. 2000. <http://eprint.iacr.org/>.
- [CDMW09a] Seung Geol Choi, Dana Dachman-Soled, Tal Malkin, and Hoeteck Wee. Improved non-committing encryption with applications to adaptively secure protocols. In *ASIACRYPT*, pages 287–302, 2009.
- [CDMW09b] Seung Geol Choi, Dana Dachman-Soled, Tal Malkin, and Hoeteck Wee. Simple, black-box constructions of adaptively secure protocols. In *TCC*, pages 387–402, 2009.
- [CFGN96] Ran Canetti, Uriel Feige, Oded Goldreich, and Moni Naor. Adaptively secure multi-party computation. In *STOC*, pages 639–648, 1996.
- [CLOS02] Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai. Universally composable two-party and multi-party secure computation. In *STOC*, pages 494–503, 2002.
- [DDN91] Danny Dolev, Cynthia Dwork, and Moni Naor. Non-malleable cryptography (extended abstract). In *STOC*, pages 542–552, 1991.
- [DSP92] A. De Santis and G. Persiano. Zero-knowledge proofs of knowledge without interaction. In *Proceedings of the 33rd Annual Symposium on Foundations of Computer Science, SFCS '92*, pages 427–436, Washington, DC, USA, 1992. IEEE Computer Society.
- [FS89] Uriel Feige and Adi Shamir. Zero knowledge proofs of knowledge in two rounds. In *CRYPTO*, pages 526–544, 1989.
- [GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority. In *STOC*, pages 218–229, 1987.
- [Goy11] Vipul Goyal. Constant round non-malleable protocols using one way functions. In *STOC*, pages 695–704, 2011.
- [GWZ09] Juan A. Garay, Daniel Wichs, and Hong-Sheng Zhou. Somewhat non-committing encryption and efficient adaptively secure oblivious transfer. In *CRYPTO*, pages 505–523, 2009.
- [IPS08] Yuval Ishai, Manoj Prabhakaran, and Amit Sahai. Founding cryptography on oblivious transfer - efficiently. In *CRYPTO*, pages 572–591, 2008.
- [KO04] Jonathan Katz and Rafail Ostrovsky. Round-optimal secure two-party computation. In *CRYPTO*, pages 335–354, 2004.
- [KOS03] Jonathan Katz, Rafail Ostrovsky, and Adam Smith. Round efficiency of multi-party computation with a dishonest majority. In *EUROCRYPT*, pages 578–595, 2003.
- [LP11] Huijia Lin and Rafael Pass. Constant-round non-malleable commitments from any one-way function. In *STOC*, pages 705–714, 2011.
- [LZ11] Yehuda Lindell and Hila Zarosim. Adaptive zero-knowledge proofs and adaptively secure oblivious transfer. *J. Cryptology*, 24(4):761–799, 2011.
- [Nao91] Moni Naor. Bit commitment using pseudorandomness. *Journal of Cryptology*, 4(2):151–158, 1991. Preliminary version in CRYPTO' 89.
- [Pas04] Rafael Pass. Bounded-concurrent secure multi-party computation with a dishonest majority. In *Proc. 36th STOC*, pages 232–241, 2004.
- [PR03] Rafael Pass and Alon Rosen. Bounded-concurrent secure two-party computation in a constant number of rounds. In *FOCS*, pages 404–413, 2003.
- [PR08] Rafael Pass and Alon Rosen. New and improved constructions of nonmalleable cryptographic protocols. *SIAM J. Comput.*, 38(2):702–752, 2008.
- [PRS02] Manoj Prabhakaran, Alon Rosen, and Amit Sahai. Concurrent zero knowledge with logarithmic round-complexity. In *FOCS*, 2002.
- [PW09] Rafael Pass and Hoeteck Wee. Black-box constructions of two-party protocols from one-way functions. In *TCC*, pages 403–418, 2009.
- [Ros04] Alon Rosen. A note on constant-round zero-knowledge proofs for NP. In *TCC*, pages 191–202, 2004.
- [Yao86] Andrew Chi-Chih Yao. How to generate and exchange secrets. In *Proc. 27th FOCS*, pages 162–167, 1986.

A Black Box Positive Result

In this section we will outline the construction of an adaptively secure MPC protocol with a black-box simulator. However, as observed in Section 3 such a protocol cannot be round efficient protocol as we have restricted ourselves to black-box simulation only.

As already observed in Section 3 a black-box simulator must rely on rewinding only. And the key problem with rewinding in the context of adaptive security is that whenever a simulator rewinds, the adversary might

corrupt parties different from the ones that were corrupted in the main thread. In this case the simulator is not able to continue with the look-ahead thread and therefore obtains no additional information via rewindings. However, this can actually be handled by *sufficiently* increasing the round complexity of the protocol. Consider a protocol with round complexity that is sufficiently larger than the number of parties. For such a protocol, we can argue that there will exist a number of rounds during which the adversary does not corrupt any party. As we next see this solution indeed works. We stress that the main result of this paper is a constant round protocol using non-black box simulation. So we do not give full details of this construction but instead just sketch the construction.

We start by describing a black-box construction for the $\langle P, V \rangle$ protocol (see Section 5.1) secure in the setting of n -parties with an adaptive adversary. We will use a simple challenge-response based extractable statistically-binding string commitment scheme that has been used in several prior works, most notably [PRS02, Ros04]. We note that in contrast to [PRS02] we will need a multi-slot protocol where the number of slots actually depend on the number of parties. Let $\text{Com}(\cdot)$ denote the commitment function of a non-interactive perfectly binding string commitment scheme (as described in Section 4). Our protocol proceeds as follows.

1. V samples a random string $\alpha \in \{0, 1\}^{3k}$, and $2kn$ independently chosen random pairs $\{\beta_i^0, \beta_i^1\}_{i=1}^{2kn}$ of α such that $\forall i \in [2kn], \beta_i^0 \oplus \beta_i^1 = \alpha$; and commits to all of using Com . Let $A \leftarrow \text{Com}(\alpha)$, and $B_i^0 \leftarrow \text{Com}(\beta_i^0)$, $B_i^1 \leftarrow \text{Com}(\beta_i^1)$ for every $i \in [2kn]$. V then sends all the commitments to P .
2. For $j \in [2n]$:
 - (a) P sends a random string $s^j \in \{0, 1\}^k$ to V . Let s_i^j be the i^{th} bit of s^j .
 - (b) For every $i \in [k]$, V opens $B_{(j-1) \cdot k + i}^0$ if s_i^j is 0 and it opens $B_{(j-1) \cdot k + i}^1$ otherwise. P checks to see if the openings are correct and aborts otherwise.
3. P sends random string $\gamma \in \{0, 1\}^{3k}$ to V .
4. V opens the commitments A , B_i^0 and B_i^1 for every $i \in [2kn]$. P checks to see if the openings are correct and aborts otherwise.
5. **OUTPUT:** Let G be a graph (can be constructed using an NP-reduction) that is Hamiltonian if and only if $\exists w$ such that $\alpha \oplus \gamma = f(w)$, where f is a pseudorandom generator. Both parties output G .

Now we roughly argue that the protocol above is secure in the adaptive setting. Firstly, we require that the adversary cannot output a Hamiltonian cycle generated in an execution of the above protocol as long as V is not corrupted by the end of the protocol execution. This follows directly from the hiding property of Com . Secondly, we need to show a simulator that can simulate the view of the adaptive adversary and output the Hamiltonian cycle in the graph generated if P is not corrupted till the end of the protocol. Again our simulator for this is very simple. Our simulator follows the honest party protocol on behalf of the uncorrupted parties and forces the output to be a pre-chosen pseudorandom string allowing it to output the cycle in the graph generated. However note that the simulator needs to know α in order to force the output of the protocol to a value of its choice. Simulator can obtain α based on whether V was corrupted when the first message of the protocol was sent. If V was honest then our simulator is already aware of α . On the other hand if V was corrupted then our simulator needs to rely on rewinding to extract α . Observe that we have $2n$ slots and the adversary can corrupt up to n parties. Therefore there must exist at least n slots where no party is corrupted and our simulator can rewind in these slots with the hope of extracting α . Since the simulator has a number of rewinding opportunities we can argue that the simulator will always (except with negligible probability) be able to extract α before it sends β on behalf of P in the protocol.

We can construct an adaptively secure n -party protocol by executing the above protocol between every pair of parties and following the construction described in Section 6. However, we will need to execute the instances of the $\langle P, V \rangle$ protocol described above sequentially. Therefore the round complexity of the protocol will be $O(n^3)$.

We believe that the round complexity of the protocol can be improved to $O(n)$ by using non-malleable commitments [DDN91, Goy11, LP11] instead of Com , allowing for parallel executions of the $\langle P, V \rangle$ protocol. Note that this round complexity would be almost tight with respect to our impossibility result (Section 3). However, as pointed out earlier, since we have already constructed a constant round protocol we do not delve into optimizing the round complexity of this protocol.