Bounded-Concurrent Secure Multi-Party Computation with a Dishonest Majority*

Rafael Pass
Massachusetts Institute of Technology
`pass@csail.mit.edu`

June 4, 2004

Abstract

We show how to securely realize any multi-party functionality in a way that preserves security under an a-priori bounded number of concurrent executions, regardless of the number of corrupted parties. Previous protocols for the above task either rely on set-up assumptions such as a Common Reference String, or require an honest majority. Our constructions are in the plain model and rely on standard intractability assumptions (enhanced trapdoor permutations and collision resistant hash functions).

Even though our main focus is on feasibility of concurrent multi-party computation we actually obtain a protocol using only a constant number of communication rounds. As a consequence our protocol yields the first construction of constant-round *stand-alone* secure multi-party computation with a dishonest majority, proven secure under standard (polynomial-time) hardness assumptions; previous solutions to this task either require logarithmic round-complexity, or subexponential hardness assumptions.

The core of our protocol is a novel construction of (concurrently) simulation-sound zero-knowledge protocols, which might be of independent interest.

Finally, we extend the framework constructed to give a protocol for secure multi-party (and thus two-party) computation for any number of corrupted parties, which remains secure even when *arbitrary subsets of parties* concurrently execute the protocol, possibly with *interchangeable* roles. As far as we know, for the case of two-party or multi-party protocols with a dishonest majority, this is the first positive result for *any* non-trivial functionality which achieves this property in the plain model.

Keywords: secure multi-party computation with honest minority, concurrent composition, non black-box simulation, simulation-sound zero-knowledge, constant-round protocols.

*An extended abstract of this paper appeared in *36'th STOC*, 2004.

1 Introduction

The task of *secure multi-party computation* involves n parties $P_1, ..., P_n$, that wish to jointly and securely compute a functionality $f(\bar{x}) = (f_1(\bar{x}), ..., f_n(\bar{x}))$ of their corresponding private inputs $\bar{x} = x_1, ..., x_n$, such that party P_i receives the value $f_i(\bar{x})$. This functionality may be probabilistic in which case $f_i(\bar{x})$ is a random variable. Loosely speaking, the security requirements are that the parties learn nothing more from the protocol than their prescribed output, and that the output of each party is distributed according to the prescribed functionality. This should hold even in case that a subset of the parties maliciously deviates from the protocol. Shortly after its conceptualization, very strong results were established for secure multi-party computation. Specifically, it was shown that *any* probabilistic polynomial-time computable multi-party functionality can be securely computed, regardless of the number of corrupted parties, assuming the existence of enhanced trapdoor permutations [41, 25].

1.1 Secure Multi-Party Computation in the Concurrent Setting

The original setting in which secure multi-party protocols were investigated allowed the execution of a single instance of the protocol at a time. This is the so called *stand-alone* setting. A more realistic setting, however, is one which allows the *concurrent* execution of protocols. In the concurrent setting (originally introduced in the context of zero-knowledge [20, 18]), many protocols are executed at the same time. This setting presents the new risk of a coordinated attack in which an adversary interleaves many different executions of a protocol and chooses its messages in each instance based on other partial executions of the protocol.

It has been noticed that security of a specific protocol in the stand-alone setting does not necessarily imply its security in the, more demanding, concurrent setting. It is thus of great relevance to examine whether the original feasibility results for multi-party computation in the stand-alone setting still hold when many copies of the protocol are executed concurrently.

Multi-party computation with an honest majority For the case of an honest majority, Canetti [9], based on the works of Ben-Or, Goldwasser and Wigderson [6], and Rabin and Ben-Or [37], show that a protocol for secure multi-party computation that composes concurrently, can be achieved. (In fact, the protocol of [9] achieves the even stronger property of Universal Composability. See [9] for more details.) The proof of [9] relies on the fact that in the multi-party setting with an honest majority a specific simulation technique called *one-pass black-box simulation* (that is black-box simulation without the use of *rewinding*) can be performed. We note that this technique is inherently bound to fail in the case of a dishonest majority [12].

Multi-party computation with a dishonest majority For the case of a dishonest majority, on the other hand, the situation is less satisfactory. The only previous positive results in the plain model (without set-up assumptions) are in the special-case of two-party computation and can be summarized as follows:

- Lindell demonstrated the feasibility of m-bounded concurrent two-party computation using $O(m)$ communication rounds. [30]

- Recently, Pass and Rosen showed that the notion of m-bounded concurrent two-party computation could be achieved using only $O(1)$ communications rounds. [35].

The notion of m-bounded concurrent composition, first considered by Barak [1] in the context of zero-knowledge, relates to a restricted form of concurrent composition, where an a-priori bound m on the number of concurrent executions is specified before the protocol is constructed. Certainly the notion of unbounded concurrency, i.e, security is guaranteed even under an unbounded number of concurrent executions, is more appealing than that of bounded concurrency. However, a recent, surprising, result by Lindell [32] shows that unbounded concurrent two-party computation can not be achieved in general.[1] On the other hand, in the special case of zero-knowledge proofs, security under unbounded concurrent composition can be achieved [38].

We note that the lower bound by Lindell also extends to the multi-party setting with a dishonest majority. The upper bounds of [30] and [35], however, do not seem to directly extends to the multi-party setting, leaving the following important question unanswered:

Does there exist a protocol for secure multi-party computation with a dishonest majority whose security is preserved under an a-priori bounded number of concurrent executions?

This is the question that we address in the current work.

Concurrent composition using set-up assumptions Before we state our results, let us mention a few words about the use of set-up assumption in order to deal with concurrency. The literature discusses a wide array of set-up assumptions, most notably the timing assumption [18, 19], the Public-Key models [11, 16] and the Common Reference String (CRS) model [9, 14]. Whereas the former assumptions have been primarily used to obtain concurrent composition of zero-knowledge proofs, the latter assumption has been also used to obtain strong composition theorems for secure multi-party computation. Specifically, in the model assuming the existence of a Common Reference String, it has been shown how to achieve the very strong notion of Universal Composability for both two-party and multi-party secure computation [9, 14]. In particular, universal composability implies security under unbounded concurrent composition.

Nevertheless, although very strong result have been achieved in the CRS model it is still of both theoretical and practical interest to investigate the possibility of constructing protocols in the plain model, without any set-up assumption, that compose concurrently. Firstly, in some settings it might be difficult to obtain the trusted set-up phase needed in order to set up a common reference string. Secondly, recent work seems to indicate that the CRS model might not reflect the security concerns than one would like to address in reality [34]. Loosely speaking, standard simulation-based definitions of security in the CRS model do not cover certain "natural" security properties that are satisfied in the plain model (e.g., *deniability* of protocols). Moreover, known impossibility results for composition in the plain model (e.g., [13, 30]) will apply to any protocol that satisfies these "natural" security properties in the CRS model.

1.2 Our Results

In this work we address the above mentioned question and present a result of a positive nature. Specifically, we show that for any n-party functionality (two-party functionalities being a special case) there exist a protocol that is secure under bounded-concurrent composition, regardless of the number of corrupted parties.

[1] Note that the lower bound of [32] relies on the fact that the honest party chooses its input adaptively. It might still be possible to construct a protocol for unbounded concurrent two-party computation for the case when the honest party chooses its inputs ahead of the protocol executions.

The security of our protocol relies on the existence of enhanced trapdoor permutations [21], as well as on the existence of collision resistant hash functions. For an integer m, the notion of m-bounded concurrent composition refers to a setting in which at most m copies of a protocol are executed concurrently (typically, $m = m(k)$ is a fixed polynomial in the security parameter k). Our results are stated below.

Main Theorem *Assume the existence of enhanced trapdoor permutations and the existence of collision-resistant hash functions. Then, for any n-party functionality f and for any integer m, there exists a protocol Π that securely computes f under m-bounded concurrent composition, regardless of the number of corrupted parties. Moreover, the number of communication rounds is constant.*

Although our main focus is on feasibility of multi-party computation with a dishonest majority that composes concurrently, we actually achieve the best one could hope for in terms of round complexity. By doing so we get the following corollary which is interesting in its own:

Corollary 1 (Secure Multi-Party Computation with a Dishonest Majority) *Assume the existence of enhanced trapdoor permutations and the existence of collision-resistant hash functions. Then, for any n-party functionality f there exists a protocol Π that securely computes f, regardless of the number of corrupted parties. Moreover, the number of communication rounds is constant.*

Corollary 1 gives the first constant-round construction for (stand-alone) secure multi-party computation under standard polynomial-time hardness assumptions.[2] The only previous constant-round solution is a protocol by Katz, Ostrovsky and Smith [29], which relies on the assumption of dense cryptosystems and collision resistant hash functions, both secure against subexponential sized circuits, as well as enhanced trapdoor permutations. The protocol of Katz et al, in fact, relies on the non-malleable zero-knowledge protocol of Barak [2] and inherits both the subexponential hardness assumptions and the complexity of this protocol. Katz et al. also give a protocol based on polynomial-time hardness assumptions, this time relying on the technique of Chor and Rabin [15], although this protocol requires a logarithmic round-complexity.

We further note that an extra feature of our approach is the relative simplicity of the solution in comparison to the works of [30, 35, 29, 2].

On the model of computation: In this work we consider a *malicious static* computationally bounded (i.e., probabilistic polynomial-time) adversary that is allowed to corrupt an arbitrary number of parties. That is, before the beginning of the interaction the adversary corrupts a subset of the players that may deviate arbitrarily from the protocol.

The parties are assumed to be connected through a point-to-point *asynchronous* public network, without guaranteed delivery of messages. Following the standard of the area [21], we assume that the communication channels are *authenticated*.

The focus of this work is not on fairness, and we therefore present a definition where the adversary always receives the output of the functionality first and then decides which of the honest

[2] In the case of stand-alone security our security definition reduces down to *secure computation with abort and no fairness* in the taxonomy of [27]. As was noted in [27], in a setting with a broadcast channel, any protocol for secure computation with abort can be transformed (in a round-preserving way) into a protocol secure according to the more standard definition of secure computation with *unanimous abort*. This is obtained by letting the parties broadcast whether they received output or not, at the end of the protocol with abort.

parties that will receive its output (c.f. *Secure computation with abort and no fairness* in the taxonomy of [27]).

The concurrent setting considered is a generalization of Lindell's treatment [30, 32] for concurrent secure two-party computation, which, in turn, is adapted from previous works on concurrent zero-knowledge [18]. We consider a *single set* of n parties that are running many concurrent executions of the same multi-party protocol, each time playing the same roles. The honest parties may adaptively choose their inputs to the protocol based on outputs received from previous executions. However, in each instance of the protocol the honest parties are required to act independently. The adversary, on the other hand, may arbitrarily coordinate its actions between the different executions.

The restriction on m-bounded concurrency is formalized by requiring that during the execution of a single instance of the protocol, messages from no more than $m-1$ other executions are exchanged.

New techniques: Already in the case of (bounded) concurrent two-party computation, it was noticed that the main technical difficulty consists of constructing two zero-knowledge protocols that are *simulation-sound* with respect to each other [30, 35]. Simulation-soundness (introduced by Sahai in a different context [40]) means that the soundness of each of the protocols is preserved even when the other protocol is simulated at the same time with the roles of the prover and verifier reversed. The problem is related to the notion of *non-malleability* [17]. Unfortunately, since soundness, here, should hold also when *concurrently* simulating many zero-knowledge protocols, known techniques for achieving non-malleability, such as [17, 2], do not seem to apply. The works of [30, 35] constructed protocols satisfying this notion. We note that their techniques inherently make use of the fact that in the two-party setting only *two* protocols need to be constructed. In fact, both [30] and [35] construct two asymmetric protocols and rely on different methods for proving simulation-soundness in each direction.

In the multi-party setting, on the other hand, the difficulty consists of constructing a family of *polynomially* many zero-knowledge protocols that are simulation-sound with respect to each other under concurrent executions. The main technical contribution of this work is the construction of a family of constant-round protocols satisfying this property. Although our techniques partly rely on techniques from [30] and [35], the methods for achieving simulation-soundness are, in fact, quite different.

Interestingly, the problem of constructing a family of polynomially many zero-knowledge arguments, such that are the protocols are simulation-sound with respect to each other under *parallel* executions, has occurred before in the context of stand-alone secure multi-party computation [29]. Previously, the only known constant-round protocols satisfying this (potentially weaker) property were the non-malleable zero-knowledge protocols of Barak [2] (with extensions in [29]). Our construction improves upon these protocols both in terms of assumptions (we rely on standard polynomial-time hardness assumptions, while the protocols of [2] rely on subexponential hardness assumptions), and in terms of simplicity.

We believe that our techniques for achieving simulation-soundness, and the zero-knowledge protocols constructed, are of independent interest and might find applications elsewhere.

1.3 Extensions to Arbitrary Sets of Parties

Until now we have focused on concurrent composition of multi-party protocols for a *single set of parties.* In this setting two (or more) parties are executing multiple concurrent executions of the same protocol, each time playing the same roles (i.e., using *fixed roles*). The setting originated in works on concurrent zero-knowledge [20, 18], where in particular, one party acts as prover in all concurrent executions and another party acts as verifier in all executions. Alternatively, this setting can be seen as modeling many different provers and verifiers with the restriction that the adversary may only corrupt either parties acting as provers or parties acting as verifiers. In other words, the adversary may not corrupt both a party acting as prover and a party acting as verifier.

A more general setting considers concurrent composability of a protocol with *arbitrary subsets of parties* [31]. In this setting, arbitrary, and possibly intersecting, sets of parties concurrently execute the same protocol. In particular, there is no restriction on what role a specific party plays in the protocol, and the roles might be *interchanged*. This means that a party may play the role of P_1 in one execution and simultaneously play the role of P_2 in a different execution. In the case of zero-knowledge proofs, this would mean that a party may act as prover in some of the concurrent executions and act as verifier in the others.

In Section 3.3 we show that the framework developed in this paper can be extended to give a protocol for secure multi-party (and thus also two-party) computation for any number of corrupted parties, which remains secure under bounded-concurrent composition with arbitrary subsets of parties. As far as we know, for the case of two-party or multi-party protocols with a dishonest majority, this is the first positive result for *any* non-trivial functionality (zero-knowledge proofs being an important special-case) which achieves this property. We mention that our solution relies on the assumption that the participating parties have unique identities.

1.4 Subsequent Work and Open Problems

Unbounded concurrency The elegant lower bound of Lindell [32] shows that unbounded concurrent composition can not be achieved in the model that we are considering (even for a single set of parties). Nevertheless, it might still be possible to achieve unbounded concurrency in a weaker model, where for example the inputs of the honest parties are fixed ahead of the protocol execution (i.e., inputs are chosen in a non-adaptive way).

Non-malleable cryptographic primitives In subsequent work [36], our zero-knowledge protocols are used in order to construct simple *constant-round* protocols for non-malleable cryptographic primitives, such as commitments and zero-knowledge, under standard cryptographic assumptions. Previous constant-round constructions relied on sub-exponential hardness assumptions [2].

Cryptographic assumptions We have shown that the notion of (bounded-concurrent) secure multi-party computation with a dishonest majority can be achieved in a constant number of rounds based on standard polynomial-time hardness assumptions, such as the existence of enhanced trapdoor permutations and collision resistant hash-functions. In contrast, stand-alone secure multi-party computation with polynomial round-complexity can be based on the existence of only enhanced trapdoor permutations [24]. This gives rise to two interesting open problems:

- Is it possible to construct a constant-round protocol for stand-alone secure multi-party computation with a dishonest majority, based on only enhanced trapdoor permutations?
- Is it possible to construct a protocol (which might have a non-constant round-complexity) for bounded-concurrent secure two-party/multi-party computation with a dishonest majority based on only enhanced trapdoor permutations?

1.5 Organization of the Paper

Formal definitions of bounded-concurrent secure multi-party computation can be found in Section 2. Section 3 contains a high level description of our proof, as well as a description of our new zero-knowledge protocols. Formal proofs are given in Section B, with preliminaries in Section A.

2 Definitions: m-Bounded Concurrent Secure Computation

In this section we present the definition for m-bounded concurrent secure multi-party computation. The definition below is a generalization of the definition of secure multi-party computation with abort and no fairness of Goldwasser and Lindell [27] and the definition of concurrent secure two-party computation with adaptive inputs of Lindell [32]. Parts of the definition have been taken almost verbatim from [27] and [32].

The basic description and definition of secure computation follows [26, 33, 5, 8]. We denote computational indistinguishability by $\stackrel{c}{\equiv}$, and the security parameter by k. For notational simplicity, we let the lengths of the parties' inputs be k.

Multi-party computation. A multi-party protocol problem for n parties $P_1, ..., P_n$ is cast by specifying a random process that maps vectors of inputs to vectors of outputs (one input and one output for each party). We refer to such a process as a n-ary functionality and denote it $f : (\{0,1\}^*)^n \to (\{0,1\}^*)^n$, where $f = (f_1, ..., f_n)$. That is, for every vector of inputs $\bar{x} = (x_1, ..., x_n)$, the output-vector is a random variable $(f_1(\bar{x}), ..., f_n(\bar{x}))$ ranging over vectors of strings. The output of the i'th party (with input x_i) is defined to be $f_i(\bar{x})$. In the context of concurrent composition, each party actually uses many inputs (one for each execution) and these may be chosen adaptively based on previous outputs. The fact that m-bounded concurrency is considered relates to the allowed scheduling of messages by the adversary in the protocol executions; see the description of the real model below.

Adversarial behavior. In this work we consider a malicious, static adversary. That is, at the beginning of the execution the adversary is given a set I of corrupted parties which it controls and through which it interacts with the honest parties while arbitrarily deviating from the specified protocol. The focus of this work is not on fairness. We therefore present a definition where the adversary always receives its own output and can then decide when (if at all) the honest parties will receive their output. The scheduling of message delivery is decided by the adversary.

Security of protocols (informal). The security of a protocol is analyzed by comparing what an adversary can do in the protocol to what it can do in an ideal scenario. The ideal scenario is formalized by considering an incorruptible *trusted third party* to whom the parties send their inputs. The trusted party computes the functionality on the inputs and returns to each party its respective output.

Unlike in the case of stand-alone computation, here the trusted party computes the functionality many times, each time upon different inputs. Loosely speaking, a protocol is secure if any adversary interacting in the real protocol (where no trusted third party exists) can do no more harm than if it was involved in the above-described ideal computation.

Concurrent execution in the ideal model. Let $I \subset [n]$ denote the subset of corrupted parties. An ideal execution with an adversary who controls the parties I proceeds as follows:

Inputs: The inputs of the parties $P_1, ..., P_n$ are respectively determined by probabilistic polynomial-time Turing machines $M_1, ..., M_n$ and the initial inputs $x_1, ..., x_n$ to these machines. As will be described below, these Turing machine determine the input values to be used by the different parties in the protocol executions. These input values are computed from the initial input, the current session number and outputs that were obtained from executions that have already concluded. Note that the number of previous outputs ranges from zero (when no previous outputs have been obtained) to some polynomial in k that depends on the number of sessions initiated by the adversary.

Session initiation: The adversary initiates a new session by sending a (start-session, i) to the trusted party. If $i \in [n] - I$ (that is P_i is an honest party) the trusted party sends (start-session, j) to P_i, where j is the index of the session (i.e., this is the j'th session to be started by P_i).

Honest parties send inputs to trusted party: Upon receiving (start-session, j) from the trusted party, each honest party P_i applies its input-selecting machine M_i to its initial input x_i, the session number j and its previous outputs, and obtains a new input $x_{i,j}$. In the first session $x_{i,1} = M_i(x, 1)$. In later sessions j, $x_{i,j} = M_i(x, j, \alpha_{i,1}...\alpha_{i,\omega})$ where ω sessions have concluded and the outputs of P_i were $\alpha_{i,1}, ..., \alpha_{i,\omega}$. Each honest party P_i then sends $(j, x_{i,j})$ to the trusted party.

Corrupted parties send inputs to trusted party: When-ever the adversary wishes it may ask a corrupted party P_i to send a message $(j, x'_{i,j})$ to the trusted third party, for any $x'_{i,j} \in \{0,1\}^n$ of its choice. A corrupted party P_i can send the pairs $(j, x'_{i,j})$ in any order it wishes and can also send them *adaptively* (i.e., choosing inputs based on previous outputs). The only limitation is that for any j, at most one pair indexed by j can be sent to the trusted party.

Trusted party answers corrupted parties: When the trusted third party has received messages $(j, x'_{i,j})$ from all parties (both honest and corrupted) it sets $\bar{x}'_j = (x'_{1,j}, ..., x'_{n,j})$. It then computes $f(\bar{x}'_j)$ and sends $(j, f_i(\bar{x}'_j))$ to P_i for every $i \in I$.

Adversary instructs the trusted party to answer honest parties: When the adversary sends a message of the type (send-output, j, i) to the trusted party, the trusted party directly sends $(j, f_i(x'_j))$ to party P_i. If all inputs for session i have not yet been received by the trusted party the message is ignored. If the output has already been delivered to the honest party, or i is the index of a corrupted party, the messages is ignored as well.

Outputs: Each honest party P_i always outputs the vector $(f_i(\bar{x}'_1), f_i(\bar{x}'_2), ...)$ of outputs that it received from the trusted party. The corrupted parties may output an arbitrary (probabilistic polynomial-time computable) function of its initial input and the messages obtained from the trusted party.

Let $f : (\{0,1\}^*)^n \to (\{0,1\}^*)^n$ be a n-ary functionality, where $f = (f_1, ..., f_n)$. Let S be a non-uniform probabilistic polynomial-time machine (representing the ideal-model adversary) and let $I \subset [n]$ (the set of corrupted parties) be such that for every $i \in I$, the adversary S controls P_i. Then the ideal execution of f with security parameter k, input-selecting machines $M = M_1, ..., M_n$, initial inputs $\bar{x} = (x_1, ..., x_n)$ and auxiliary input z to S, denoted $\text{IDEAL}_{f,I,S,M}(k, \bar{x}, z)$, is defined as the output vector of the parties and the adversary S resulting from the ideal process described above.

Note that the definition of the ideal model does not include any reference to the bound m on the concurrency. This is because this bound is relevant only to the scheduling allowed to the adversary in the real model; see below. However, the fact that a concurrent setting is considered can be seen from the above-described interaction of the adversary with the trusted party. Specifically, the adversary is allowed to obtain outputs in any order that it wishes, and can choose its inputs adaptively based on previous outputs. This is inevitable in a concurrent setting where the adversary can schedule the order in which all protocol executions take place. In particular, the adversary can schedule the executions sequentially, thereby learning previous outputs before defining the next input.

Execution in the real model. We next consider the real model in which a real multi-party protocol is executed (and there exists no trusted third party).

We consider a scenario where the parties communicate through an *asynchronous* fully connected point-to-point network, without guaranteed delivery of messages. The communication lines between the parties are assumed to be authenticated and thus the adversary cannot modify messages sent by honest parties.

Let f, I be as above and let Π be a multi-party protocol for computing f. Furthermore, let A be a non-uniform probabilistic polynomial-time machine such that for every $i \in I$, the adversary A controls P_i. Then, the real m-bounded concurrent execution of Π with security parameter k, input-selecting machines $M = M_1, ..., M_n$, initial inputs $\bar{x} = (x_1, ..., x_n)$ and auxiliary input z to A, denoted $\text{REAL}^m_{\Pi,I,A,M}(k, \bar{x}, z)$, is defined as the output vector of the honest parties and the adversary A resulting from the following process. The parties run concurrent executions of the protocol, where every party initiates a new session whenever it receives a start-session from the adversary. The honest parties then apply their input-selecting machines to their initial input, the session number and their previously received outputs, and obtain the input for this new session. The scheduling of all messages throughout the executions is controlled by the adversary. That is, the execution proceeds as follows. The adversary sends a message of the form (j, x) to an honest party. The honest party then adds x to the view of its j'th execution of Π and replies according to the instructions of Π and this view. Note that the honest parties runs each execution of Π obliviously to the other executions. (Thus, this is stateless composition.) The fact that the network is asynchronous means that the adversary has complete control over the timing and delivery of messages. In particular, the adversary may decide never to deliver certain messages. This is modeled as follows: The parties send messages by placing them in a "out-box" and the adversary delivers (or does not deliver) them at will.

The fact that we are considering m-bounded concurrency means that the scheduling by the adversary must fulfill the following condition: for every execution i, from the time that the i'th execution begins until the time that it ends, messages from at most m different executions can be sent.

Note that this definition of concurrency covers the case that m executions are run simultaneously. However, it also includes a more general case where many more than m executions take place, and they possibly all overlap.

Security as emulation of a real execution in the ideal model. Having defined the ideal and real models, we can now define security of protocols. Loosely speaking, the definition asserts that a secure multi-party protocol (in the real model) emulates the ideal model (in which a trusted party exists). This is formulated by saying that for every real-model adversary there exists an ideal model adversary that can simulate an execution of the secure real-model protocol.

Definition 1 (security in the malicious model): *Let $m = m(k)$ be a polynomial and let f and Π be as above. Protocol Π is said to* t-securely compute f under m-bounded concurrent composition *if for* *every real-model non-uniform probabilistic polynomial-time adversary A, there exists an ideal-model non-uniform probabilistic expected polynomial-time adversary S, such that for all input-selecting machines $M = M_1, ..., M_n$, every $z \in \{0,1\}^*$, every $\bar{x} = (x_1, ..., x_n)$, where $x_1, ..., x_2 \in \{0,1\}^*$ and every $I \subset [n]$ with $|I| < t$,*

$$\left\{\text{IDEAL}_{f,I,S,M}(k, \bar{x}, z)\right\}_{k \in N} \stackrel{c}{\equiv} \left\{\text{REAL}^{m}_{\Pi,I,A,M}(k, \bar{x}, z)\right\}_{k \in N}$$

That is, concurrent executions of Π with A cannot be distinguished from concurrent invocations of f with S in the ideal model.

3 Proof Outline

In this section we give a high-level outline of our proof, as well as a description of our new zero-knowledge protocols. More details can be found in the Appendix.

The starting point of our proof is the constant-round secure multi-party computation protocol for semi-honest adversaries of Beaver, Micali and Rogaway [7, 39]. Our goal is to compile this protocol into a protocol that is secure against a malicious adversary, in a way that preserves security under an a-priori bounded number of concurrent executions. Goldreich, Micali and Wigderson [24] presented a general compilation technique for the stand-alone setting, showing the usefulness of zero-knowledge proofs in order achieve such a compilation. Unfortunately, it is not known if the technique of [24] is secure in the concurrent setting.

Inspired by the central role of zero-knowledge proofs in the compilation of [24], we show a compilation for the concurrent setting in the following two step:

1. Identify a suitable notion of "zero-knowledge" proofs, called special-purpose zero-knowledge, and show that the compilation task reduces down to the construction of such protocols.

2. Implement special-purpose zero-knowledge.

The high-level structure of our proof follows that of Pass and Rosen [35] which in turn builds on the work of Lindell [30].

3.1 Reducing the Problem to Special-Purpose Zero-Knowledge

3.1.1 Idealized Functionalities

Our starting point is the protocol of Canetti et al. [14]. They show that in a setting where all parties have access to a specific *idealized functionality*, protocols secure against a semi-honest adversary can be compiled into protocols secure against a malicious adversary.[3] The notion of ideal functionalities is a central tool in the framework of universal composability [9], and can be thought of as the introduction of a trusted third party that is designed to perform a specific task. In the case of Canetti et al. [14], the protocol requires usage of the *ideal one-to many zero-knowledge proof of knowledge* functionality.

[3]The result by Canetti et al. [14] is in fact stronger, as they construct a protocol that is universally composable [9]. Nevertheless, in this work we only require composition in the (less demanding) concurrent setting.

The $\mathsf{IdealZK}^{1:\mathrm{M}}$ functionality: Informally, the ideal one-to-many zero-knowledge proof of knowledge functionality (called $\mathsf{IdealZK}^{1:\mathrm{M}}$) for a language L is specified as follows: The prover sends an instance-witness pair (x, w) to the ideal functionality which, in turn, sends $(x, 1)$ to all the other parties if w is a valid witness for $x \in L$ and $(x, 0)$ otherwise. (That is one prover proves a statement to many verifiers). The setting in which the parties have access to (multiple) ideal one-to-many zero-knowledge functionalities is called the $\mathsf{ZK}^{1:\mathrm{M}}$-$\mathsf{Hybrid}$ model. Our motivation is to remove the usage of the ideal functionality and implement a protocol that performs the same task but in the real model. In order to achieve this, we show a general transformation that transforms *any* protocol $\tilde{\Pi}$ in the $\mathsf{ZK}^{1:\mathrm{M}}$-$\mathsf{Hybrid}$ model into a protocol Π in the real model (without any ideal functionalities or set-up assumptions). This is what we call *realizing* the ZK-$\mathsf{Hybrid}^{1:\mathrm{M}}$ model (see Theorem 3). Jumping ahead, we mention that this part also includes the construction of the special-purpose zero-knowledge protocols (as will be described in Section 3.2). We further show that the transformation satisfies the following properties:

Concurrency: The protocol's security is preserved under (bounded) concurrent composition.

Round-Efficiency: The protocol's round-complexity is preserved (up to a constant factor).

In particular the transformation can be applied to the protocol of [14] (when in turn applied to the protocol of [7]) in order to obtain a constant-round bounded concurrent secure multi-party computation protocol (See Theorem 4).

3.1.2 How to Realize any Protocol in the $\mathbf{ZK}^{1:\mathrm{M}}$-Hybrid Model

The above transformation is performed in three steps:

1. Reduce the usage of the $\mathsf{IdealZK}^{1:\mathrm{M}}$ functionality to the usage of the more standard two-party ideal zero-knowledge proof of knowledge functionality, $\mathsf{IdealZK}$. Whereas the $\mathsf{IdealZK}^{1:\mathrm{M}}$ functionality models interaction between a single prover and many verifiers, the $\mathsf{IdealZK}$ functionality models interaction between a single prover and single verifier. The idea underlying the reduction is straight-forward and similar to the idea used by Goldwasser and Lindell [27] to implement a broadcast channel. Roughly, whenever a prover wishes to prove a statement x using the $\mathsf{IdealZK}^{1:\mathrm{M}}$ functionality, he instead gives an individual proof of x to each other party using the $\mathsf{IdealZK}$ functionality. Whenever a party receives a proof of a statement x it sends an acknowledgment to all other parties. Finally, when a party has received both a proof (from the $\mathsf{IdealZK}$ functionality) and an acknowledgment from all other parties it accepts the proof of the instance x. See Section B.1 for more details.

2. Reduce the usage of the $\mathsf{IdealZK}$ functionality to the usage of a weaker ideal functionality, $\mathsf{MemberZK}$. Whereas the $\mathsf{IdealZK}$ functionality models a proof of knowledge, i.e., the prover not only proves the validity of a statement but also that it possesses a witness for it, the $\mathsf{MemberZK}$ functionality models simply a proof. Such a reduction was shown in [35], making use of a general transformation of zero-knowledge proofs into zero-knowledge proofs of knowledge due to [4]. We remark that the same reduction goes through also in the multi-party setting. See Section B.2.

3. Show how to transform any protocol using the $\mathsf{MemberZK}$ functionality (we call this setting the $\mathsf{MemberZK}$-Hybrid model) into a protocol in the real model by "plugging in" the special-purpose zero-knowledge protocols. See Section B.3. We note that such a transformation was shown in [30] and in [35] for the special case of two-party protocols. Unfortunately, these transformations do not seem to directly extend to the multi-party setting.

The proof of the last step essentially boils down to the construction of a family of n (one for each party) constant-round zero-knowledge protocols, $c\mathcal{ZK}_1, ..., c\mathcal{ZK}_n$. These protocols are used to instantiate calls to the MemberZK functionality, in the following way: a call to the MemberZK functionality initialized by a party P_i (acting as prover) is instantiated with an execution of the protocol $c\mathcal{ZK}_i$. In order for the instantiation to preserve security, the family of protocols needs to satisfy the following requirements:

- The protocols are bounded-concurrent zero-knowledge.
- The protocols compose concurrently with respect to arbitrary protocols with an a-priori bounded communication complexity.
- The protocols compose concurrently (an a-priori bounded number of times) with respect to themselves, with the roles of the prover and the verifier reversed. That is, simulation of the protocols can be performed even while simultaneously verifying them.
- Simulation of the protocols does not make use of rewinding.
- The protocols are *simulation-sound* with respect to each other. That is, the soundness of each one of the protocols is preserved even when the all the other protocols are simulated at the same time with the role of the prover and verifier reversed.

3.2 The Special-Purpose Zero-Knowledge Protocols

As in [35], our special purpose zero-knowledge protocols are based on the bounded-concurrent zero-knowledge protocol of Barak [1]. Before describing our protocols we therefore give a brief description of this protocol with an emphasis on the simulation technique, which will be useful in our setting. (Parts of the following paragraph are taken almost verbatim from [35]).

Barak's protocol relies on the existence of collision resistant hash functions. Other tools used in the protocol are perfectly binding bit-commitments and a witness-indistinguishable universal argument (WI UARG) [3]. More details on Barak's protocol can be found in [1]. The underlying idea behind Barak's protocol is the usage of an $NTIME(T(k))$ relation denoted R_{sim}. This relation is described in Figure 1.

Barak's $NTIME(T(k))$ Relation R_{Sim}

Input: A triplet $\langle h, c, r \rangle$.

Witness: A program Π, a string $y \in \{0,1\}^{(|r|-k)}$, and string s.

Relation: $R_{\mathsf{Sim}}(\langle h, c, r \rangle, \langle \Pi, s, y \rangle) = 1$ if and only if:

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

Figure 1: Barak's $NTIME(T(k))$ relation R_{sim}.

Let $T : \mathcal{N} \to \mathcal{N}$ be a "nice" function that satisfies $T(k) = k^{\omega(1)}$. Let $T' : \mathcal{N} \to \mathcal{N}$ be a function such that $T'(k) = T(k)^{\omega(1)}$. Suppose there exist a $T'(k)$-collision resistant hash functions ensemble $\{\mathcal{H}_k\}_{h \in \{0,1\}^k}$ where h maps $\{0,1\}^*$ to $\{0,1\}^k$. Barak's protocol is described in Figure 2.
Before we continue discussing the security of the protocol, let us remark that the assumption of hash functions collision resistant for slightly super-polynomial time can be relaxed to the standard

Barak's Non Black-Box ZK Protocol for $\mathcal{NP}$

Common Input: an instance x of a language L, security parameter 1^k.

Length parameter: $\ell(k)$.

Stage 1:

$V \to P$: Send $h \stackrel{\mathrm{R}}{\leftarrow} \mathcal{H}_k$.

$P \to V$: Send $c = \mathrm{Com}(0^k)$.

$V \to P$: Send $r \in \{0,1\}^{\ell(k)}$.

Stage 2: (Proof Body)

$P \leftrightarrow V$: A *WI UARG* proving the OR of the following two statements:

1. There exists $w \in \{0,1\}^{\mathrm{poly}(|x|)}$ so that $R_L(x, w) = 1$.
2. There exists a pair $\langle \Pi, s \rangle$ so that $R_{\mathsf{Sim}}(\langle h, c, r \rangle, \langle \Pi, s \rangle) = 1$.

Figure 2: Barak's bounded-concurrent zero-knowledge argument for $\mathcal{NP}$.

assumption of hash functions collision resistant for polynomial time [3]. Looking ahead, we note that the same relaxation can be done also to our protocols.

As shown in [1], Barak's protocol is computationally sound. Moreover, given access to the verifier's code (or, equivalently, to the verifier's next message function), the protocol can be simulated without making use of rewinding. To simulate a specific session, the simulator commits to the verifier's next-message function (instead of committing to zeros). The verifier's next message function is then a program whose output depends on all the prover messages sent between messages c and r in the relevant session. Now, if the length of all these messages is bounded by $\ell(k) - k$ the simulator will have a valid witness for stage 2 of the protocol. The zero-knowledge property then follows (with some work) from the witness indistinguishable property of stage 2.

Barak [1] has shown that the length of the prover's messages in a single execution can be bounded by $2k^2$. Thus, for m concurrent executions, taking $\ell(k) = 2m \cdot k^2 + k$ would do. We note that the length of the verifier's messages, *not counting r's*, can also be bounded by $2k^2$.

An important extra feature of the protocol is that it is sufficient that the simulator has a "short" description of the prover's messages. The protocol is thus simulatable even if the total length of the prover's messages is greater than $\ell(k) - k$, as long as the simulator has a description of a program for generating the prover's messages that is shorter than $\ell(k) - k$.

Our protocols: Suppose there exist a collision resistant hash functions ensemble $\{\mathcal{H}_k\}_{h \in \{0,1\}^k}$ as required by Barak's protocol. We are ready to describe our special purpose protocols, $c\mathcal{ZK}_1, ..., c\mathcal{ZK}_n$ (depicted in Figure 3).

The difference of the protocols $c\mathcal{ZK}_i$ from the protocol of Barak is that the prover (simulator) is given two opportunities to guess the verifier's next message (called the challenge). We call each such opportunity a slot. Messages that are exchanged between the message c_j and the message r_j are said to be contained in slot j. Note that it is sufficient to have a "short" description of the messages contained in one of the two slots to succeed in the simulation of $c\mathcal{ZK}_i$.

Note that the sequence of protocols satisfies the following two properties:

- the length of the first challenge, r_1, is strictly *increasing*,

Protocol i - $c\mathcal{ZK}_i$

Common Input: an instance x of a language L, security parameter 1^k.

Length parameter: $\ell(k)$.

Stage 0 (Set-up):

$V \rightarrow P$: Send $h \stackrel{\text{R}}{\leftarrow} \mathcal{H}_k$.

Stage 1 (Slot 1):

$P \rightarrow V$: Send $c_1 = \text{Com}(0^k)$.

$V \rightarrow P$: Send $r_1 \stackrel{\text{R}}{\leftarrow} \{0,1\}^{i\ell(k)}$ (first challenge).

Stage 2 (Slot 2):

$P \rightarrow V$: Send $c_2 = \text{Com}(0^k)$.

$V \rightarrow P$: Send $r_2 \stackrel{\text{R}}{\leftarrow} \{0,1\}^{(n+1-i)\ell(k)}$ (second challenge).

Stage 3: (Proof Body)

$P \leftrightarrow V$: A *WI UARG* proving the OR of the following two statements:

1. There exists $w \in \{0,1\}^{\text{poly}(|x|)}$ so that $R_L(x,w) = 1$.
2. There exists a tripple $\langle \Pi, s, j \rangle$ so that $R_{\mathsf{Sim}}(\langle h, c_j, r_j \rangle, \langle \Pi, s \rangle) = 1$.

Figure 3: The i'th special-purpose zero-knowledge protocol – $c\mathcal{ZK}_i$.

- the length of the second challenge, r_2, is strictly *decreasing.*

Thus, for any two protocols $c\mathcal{ZK}_i, c\mathcal{ZK}_j$ it holds that either the length of the first or the second challenge in $c\mathcal{ZK}_i$ is strictly larger than the corresponding challenge in $c\mathcal{ZK}_j$. This property will be essential to us when proving that the protocols are simulation sound with respect to each other.

Simulating the protocols: Let $\ell(k) = m(\gamma \cdot 4k^2 + \texttt{length}(\Pi')) + k$, where Π' denotes the protocol in the MemberZK-Hybrid model (that we wish to realize) and γ is the total number of ideal zero-knowledge calls in one execution of Π'. We note that the total length of messages sent by a party, not including the challenges r_1, r_2, is bounded by $m(\gamma \cdot 2k^2 + \texttt{length}(\Pi'))$, as both the length of the prover messages and verifier messages in the protocols $c\mathcal{ZK}_1, ..., c\mathcal{ZK}_n$ are bounded by $2k^2$. It follows that if the simulator can give a description of the various challenges r_1, r_2 (each being of length at least $\ell(k)$) that is shorter than k^2 it will always succeed in the simulation of $c\mathcal{ZK}_1, .., c\mathcal{ZK}_n$, even if simultaneously verifying these protocols. As in [35], this is done by letting the simulator use a pseudorandom generator[4] in order to generate verifier messages (and in particular the verifier's challenges r_1, r_2) when playing the role of the verifier in other protocols (see Section C.1 for further details).

Note that the simulator has two opportunities to succeed in the simulation, that is by either using slot 1 or 2. As a convention, we let the simulator always use the first slot to commit the verifier's code and acts as the honest prover (that is committing to zeros) in the second slot.

Soundness of the stand-alone protocols: It follows using the same argument as in [1] that the protocols are computationally sound.

[4]Note that the existence of pseudorandom generators follows from our assumptions [28]

Simulation soundness. The last step is to guarantee that the soundness of each one of the protocols $c\mathcal{ZK}_1, ..., c\mathcal{ZK}_n$ is not violated when simulating any (or all) of the other protocol with the roles of the prover and verifier reversed. To do this we show how to transform a cheating prover in this simulation/cheating scenario into a cheating prover for the stand alone case.

Suppose there exists a cheating prover P^* that manages to violate the soundness of one instance of protocol $c\mathcal{ZK}_i$ while it is verifying the simulation of multiple concurrent instances of the other protocols $c\mathcal{ZK}_j$, where $j \neq i$.

Using a standard argument, we show how to construct a cheating prover P^{**} for a single instance of $c\mathcal{ZK}_i$ by forwarding the messages of this "cheating" instance of $c\mathcal{ZK}_i$ to an external honest verifier V (the instance in which P^* is actually cheating can be "guessed" by picking one of the instances of $c\mathcal{ZK}_i$ at random). V's replies are then forwarded by P^{**} back to P^* as if they were generated by the simulator. Since P^* is assumed to be cheating in this specific instance of $c\mathcal{ZK}_i$, and since the verifier messages used by P^{**} are indistinguishable from the messages used by V, the (stand-alone) soundness of $c\mathcal{ZK}_i$ is presumably violated.

One problem that arises is that the code of the external verifier V is not available to us. This means that the straightforward simulation of the protocols $c\mathcal{ZK}_j$, where $j \neq i$, cannot be completed as it is, since it explicitly requires possession of a "short" description of the corresponding verifier messages. To overcome this problem we resort to an alternative simulation technique.

The alternative simulator For all protocol $c\mathcal{ZK}_j$ where $j \neq i$ we resort to the following alternative simulation technique.

We start by noting that except for the "long" challenges r_1, r_2 sent by the verifier of $c\mathcal{ZK}_i$ we do have a description of all messages sent to the adversary that is shorter than $\ell(k) - k$. In order to show that we can still perform a simulation, even in the presence of these messages (to which we do not have a short description), we use the fact that it is sufficient to have a short description of the messages sent in *one* of the slots of $c\mathcal{ZK}_i$.

We separate between two different schedulings:

Both the first and the second challenge r_1, r_2 in $c\mathcal{ZK}_i$ are contained in the same slot in $c\mathcal{ZK}_j$. It follows that there is always a "free" slot in $c\mathcal{ZK}_j$ which can be used to perform the simulation. The WI property of the $UARG$ guarantees that the the presumed cheating prover will still succeed in convincing the honest verifier of protocol $c\mathcal{ZK}_i$ with roughly the same probability.

The messages r_1, r_2 in $c\mathcal{ZK}_i$ occur in slot 1, 2 respectively in $c\mathcal{ZK}_j$. By the construction of the protocols it follows that the length of either the first or the second challenge in $c\mathcal{ZK}_j$ is at least $\ell(k)$ bits longer than the corresponding challenge in $c\mathcal{ZK}_i$. Thus there exist a slot in $c\mathcal{ZK}_j$ such that even if we include the verifier's challenge, from the protocol $c\mathcal{ZK}_i$, in whole, in the description, we still have $\ell(k) - k$ to describe the other messages, which means that simulation can be performed. As in the previous case, the WI property of the $UARG$ guarantees that the presumed cheating prover will still succeed in convincing the honest verifier of protocol $c\mathcal{ZK}_i$ with roughly the same probability.

3.3 Concurrent Composition with Arbitrary Sets of Parties

Until this point we have focused on concurrent composition of secure multi-party computation for a *single set* of parties, i.e., n parties that concurrently are running many executions of the same protocol, each time playing the same roles in the protocol. A more general setting considers *arbitrary sets of parties* [31]. In this setting, arbitrary, and possibly intersecting, sets of parties concurrently execute the same n-party protocol. In particular, there is no restriction on what role a specific party plays in the protocol. This means that a party may play the role of P_1 in one execution and simultaneously play the role of P_2 in a different execution.

It is quite easy to see that unless the parties have identities it is impossible to construct a protocol for secure two-party (and thus also multi-party) computation in this general setting. This follows since without identities a man-in-the-middle attack can not be prevented. Let us therefore focus on a setting where all parties have unique identities.

First, assume that the participating parties have identities of length $O(\log k)$ (note that this is enough to model $poly(k)$ parties). In this case, essentially the same construction as previously described can be used. The only difference is that when instantiating calls to the $\mathsf{MemberZK}$ functionality with a special-purpose zero-knowledge protocol, this is done by using the protocol $c\mathcal{ZK}_{\mathsf{id}}$, where id is the identity of the prover initiating the zero-knowledge proof. Note that, whereas in the case of a single set of parties we instantiated the zero-knowledge calls according to the *role* of the prover in the the main protocol, the above-described instantiation is independent of the proving party's role and, instead, only depends on its *identity.* Using this construction the same proof as in the case of a single set of parties goes through.[5]

A more standard setting considers parties with identities of length k (i.e., 2^k possible identities). In this setting, the above-mentioned construction does not directly work. In fact, in order to make the same proof work, we need to construct a family of 2^k (instead of only $O(k)$) protocols that are simulation-sound with respect to each other. Below we outline how the special-purpose zero-knowledge protocols given in Figure 3 can be modified in order to achieve this property. Unfortunately, this modification increases the round-complexity.

For simplicity, let the identity id be given in base k. Let id_i denote the i'th digit of id in base k. Note that the length of id, denoted $|\mathsf{id}|$, is $k/\log k$. We show how the special-purpose protocols can be modified, using $2|\mathsf{id}|$ slots, instead of 2 slots, in order to be simulation-sound for 2^k identities. The protocol for a specific identity id is specified as follows. The first half of the slots will decode the identity id, i.e., the length of the challenge r_i in slot i is $\mathsf{id}_i \cdot l(k)$. The second half will be "mirror-slots" (in the same way that slot 2 was a mirror-slot to slot 1 for the special-purpose protocols in Figure 3), i.e, the challenge $r_{|\mathsf{id}|+i}$ in slot $|\mathsf{id}|+i$ is of length $(n+1)l(k) - r_i$. It follows, using the same argument as was used for the protocols in Figure 3, that the above construction yields a family of 2^k zero-knowledge protocols that are simulation-sound with respect to each other.

4 Acknowledgments

I am very grateful to Johan Håstad for many helpful conversations and great advice. In particular, Johan pointed out that the method described in the paper could be used to obtain constant-round protocols. I am also very grateful to Alon Rosen. This paper could not have been written without our endless conversation on the subject of protocol composition and zero-knowledge. Thanks also to Yehuda Lindell for helpful comments.

[5]It is shown in [14] that the compilation in the $\mathsf{ZK}^{1:\mathrm{M}}$-$\mathsf{Hybrid}$ model preserves security also in this setting. (The definition of the $\mathsf{ZK}^{1:\mathrm{M}}$ functionality needs to be slightly changed, as in [14], in order to specify what parties that receive the proof). The proofs of the reductions from the $\mathsf{ZK}^{1:\mathrm{M}}$-$\mathsf{Hybrid}$ model to the $\mathsf{MemberZK}$-Hybrid model are also unchanged. Thus the only point that needs to be addressed is the instantiation of the $\mathsf{MemberZK}$ calls.

References

[1] B. Barak. How to go Beyond the Black-Box Simulation Barrier. In *42nd FOCS*, pages 106–115, 2001.

[2] B. Barak. Constant-Round Coin-Tossing or Realizing the Shared-Random String Model. In *43rd FOCS*, pages 345-355, 2002.

[3] B. Barak and O. Goldreich. Universal Arguments and their Applications. *17th CCC*, pages 194–203, 2002.

[4] B. Barak and Y. Lindell. Strict Polynomial-Time in Simulation and Extraction. In *34th STOC*, pages 484–493, 2002.

[5] D. Beaver. Foundations of Secure Interactive Computing. In *Crypto91*, Springer-Verlag (LNCS 576), pages 377–391, 1991.

[6] D. Ben-Or, S. Goldwasser and A. Wigderson. Completeness Theorems for Non-Cryptographic Fault-Tolerant Distributed Computation. In *20'th STOC*, pages 1–10, 1988.

[7] D. Beaver, S. Micali and P. Rogaway The Round Complexity of Secure Protocols. In *22'th STOC*, pages 503–513, 1990.

[8] R. Canetti. Security and Composition of Multiparty Cryptographic Protocols. *Journal of Cryptology*, 13(1):143–202, 2000.

[9] R. Canetti. Universally Composable Security: A New Paradigm for Cryptographic Protocols. In *34th STOC*, pages 494–503, 2002.

[10] R. Canetti and M. Fischlin. Universally Composable Commitments. In *Crypto2001*, Springer LNCS 2139, pages 19–40, 2001.

[11] R. Canetti, O. Goldreich, S. Goldwasser and S. Micali. Resettable Zero-Knowledge. In *32nd STOC*, pages 235–244, 2000.

[12] R. Canetti, E. Kushilevitz and Y. Lindell. On the Limitations of Universally Composable Two-Party Computation Without Set-Up Assumptions. In *Eurocrypt2003*, Springer LNCS 2656, pages 68–86, 2003.

[13] R. Canetti, J. Kilian, E. Petrank and A. Rosen. Black-Box Concurrent Zero-Knowledge Requires (almost) Logarithmically Many Rounds. *SIAM Jour. on Computing*, Vol. 32(1), pages 1–47, 2002.

[14] R. Canetti, Y. Lindell, R. Ostrovsky and A. Sahai. Universally Composable Two-Party and Multy-Party Computation. In *34th STOC*, pages 494–503,2002.

[15] B. Chor, M. Rabin. Achieving Independence in Logarithmic Number of Rounds. In *6th PODC*, pages 260-268, 1987.

[16] I. Damgard. Efficient Concurrent Zero-Knowledge in the Auxiliary String Model. In *EuroCrypt2000*, LNCS 1807, pages 418–430, 2000.

[17] D. Dolev, C. Dwork and M. Naor. Non-Malleable Cryptography. *SIAM Jour. on Computing*, Vol. 30(2), pages 391–437, 2000.

[18] C. Dwork, M. Naor and A. Sahai. Concurrent Zero-Knowledge. In *30th STOC*, pages 409–418, 1998.

[19] C. Dwork and A. Sahai. Concurrent Zero-Knowledge: Reducing the Need for Timing Constraints. In *Crypto98*, Springer LNCS 1462 , pages 442–457, 1998.

[20] U. Feige and A. Shamir. Witness Indistinguishability and Witness Hiding Protocols. In *22nd STOC*, pages 416–426, 1990.

[21] O. Goldreich. Draft of a Chapter on general Protocols. Available at: `http://www.wisdom.weizmann.ac.il/~oded/PSBookFrag/prot.ps`

[22] O. Goldreich. *Foundation of Cryptography – Basic Tools.* Cambridge University Press, 2001.

[23] O. Goldreich and H. Krawczyk. On the Composition of Zero-Knowledge Proof Systems. *SIAM Jour. on Computing*, Vol. 25(1), pages 169–192, 1996.

[24] O. Goldreich, S. Micali and A. Wigderson. Proofs that Yield Nothing But Their Validity or All Languages in NP Have Zero-Knowledge Proof Systems. *JACM*, Vol. 38(1), pp. 691–729, 1991.

[25] O. Goldreich, S. Micali and A. Wigderson. How to Play any Mental Game – A Completeness Theorem for Protocols with Honest Majority. In *19th STOC*, pages 218–229, 1987.

[26] S. Goldwasser and L. Levin. Fair Computation of General Functions in Presence of Immoral Majority. In *CRYPTO'90,* Springer-Verlag (LNCS 537), pages 77–93, 1990.

[27] S. Golwasser and Y. Lindell Secure Computation without Agreement. In *16th DISC*, Springer-Verlag (LNCS 2508), pages 17-32, 2002.

[28] J. Håstad, R. Impagliazzo, L.A. Levin and M. Luby. Construction of Pseudorandom Generator from any One-Way Function. *SIAM Jour. on Computing*, Vol. 28 (4), pages 1364–1396, 1999.

[29] J. Katz, R. Ostrovsky and A. Smith. Round Efficiency of Multi-Party Computation with a Dishonest Majority, In *EuroCrypt2003.* Springer LNCS 2656 pages 578–595, 2003.

[30] Y. Lindell. Bounded-Concurrent Secure Two-Party Computation Without Setup Assumptions. To appear in *35th STOC*, pages 683-692, 2003.

[31] Y. Lindell. General Composition and Universal Composability in Secure Multi-Party Computation. In *44th FOCS*, pages 394–403, 2003

[32] Y. Lindell. Lower Bounds for Concurrent Self Composition. To appear in *1st TCC*, 2004.

[33] S. Micali and P. Rogaway. Secure computation. Unpublished manuscript, 1992. Preliminary version in *CRYPTO'91*, Springer-Verlag (LNCS 576), pages 392–404, 1991.

[34] R. Pass. On Deniability in the Common Reference String and Random Oracle Model. In *Crypto2003.* Springer LNCS 2729, pages 316–337, 2003.

[35] R. Pass, A. Rosen. Bounded-Concurrent Secure Two-Party Computation in a Constant Number of Rounds. In *44th FOCS*, pages 404–413, 2003.

[36] R. Pass, A. Rosen. New and Improved Constructions of Non-malleable Cryptographic Primitives. Manuscript in preparation.

[37] T. Rabin and M. Ben-Or. Verifiable Secret Sharing and Multi-party Protocols with Honest Majority. In *21st STOC*, pages 73–85, 1989.

[38] R. Richardson and J. Kilian. On the Concurrent Composition of Zero-Knowledge Proofs. In *EuroCrypt99*, Springer LNCS 1592, pages 415–431, 1999.

[39] P. Rogaway. The Round Complexity of Secure Protocols. PhD thesis, MIT, 1991.

[40] A. Sahai. Non-Malleable Non-Interactive Zero Knowledge and Adaptive Chosen-Ciphertext Security. In em 40'th FOCS, pages 543-553, 1999.

[41] A. Yao. How to Generate and Exchange Secrets. In *27th FOCS*, pages 162–167, 1986.

Appendix

A Definitions and Preliminaries

A.1 Definition: The ZK-Hybrid Model

The notion of ideal functionalities is a central tool in the framework of universal composability [9], and can be thought of as the introduction of a trusted third party that is designed to perform a specific task. The communication with the trusted third party is specified as follows. All parties have a secret, authenticated and unblockable channel to the trusted third party through which they send their input. The trusted party computes the output, hands the output of the corrupted parties to the adversary, and asks the adversary to deliver the output to the honest parties. The adversary does not see the output destined to the honest party but may deliver it (or not), at will.

One of the most basic and useful ideal functionalities in the design of cryptographic protocols is the *ideal zero-knowledge proof of knowledge* functionality.

The IdealZK functionality: The ideal zero-knowledge functionality is parameterized by an $\mathcal{NP}$-relation R_L and defined as follows:

- Upon receiving a message (ZK-prover,j, x, w) from a party P_i (called the prover), the functionality sends (ZK-proof,$i, x, R_L(x, w)$) to party P_j (called the verifier). Unless the party P_j is corrupted, the functionality also sends the message (ZK-proof,$i, j, x, R_L(x, w)$) to the adversary (controlling the network).

Remark: Since we consider an asynchronous execution of protocols, formally, both the input and the output of the functionality should also contain a session identifier (as was done in [9]) not to mix up messages from different sessions. In order to simplify the notation (for this and other ideal functionalities that we use), we leave out the session identifier and instead assume that this information is added to all messages (sent to and from ideal functionalities) in some canonical way.

Thus, slightly over-simplified, the prover sends an instance-witness pair (x, w) and an index j to the ideal functionality, which in turn, sends $(x, 1)$ to the verifier (party P_j) if w is a valid witness for $x \in L$ and $(x, 0)$ otherwise. In order to model provers that fail in proving true instance (clearly, one can not force a prover to prove something that is does not want to), we define a special symbol $\perp$ such that for every R_L and every x, it holds that $(x, \perp) \notin R_L$.

The setting in which the parties have access to (multiple) ideal zero-knowledge functionalities is called the ZK-Hybrid model. The power of this model has been demonstrated in for example [14] where a universally composable two-party computation protocol was constructed.

A.2 Definition: The MemberZK-Hybrid Model

Note that the ideal zero-knowledge functionality models a *proof of knowledge*, i.e., a proof whereby the verifier gets convinced not only of the validity of the statement proved, but also that the prover has an (NP) witness to the statement. The following weaker functionality was introduced in [30] as a technical tools, to model zero-knowledge proofs, as opposed to proofs of knowledge:

The ideal functionality MemberZK is parameterized with a language $L \in \mathcal{NP}$ and defined as follows (using the syntax similar to [35], here adapted to suit the multi-party setting):

- Upon receiving a message (MemberZK-prover,j, x, b) from a party P_i (called the prover), the functionality sends (MemberZK-proof,$i, x, b \cdot \chi_L(x)$) to party P_j (called the verifier), where $\chi_L(x) = 1$ if $x \in L$ and 0 otherwise. Unless the party P_j is corrupted, the functionality also sends the message (MemberZK-proof,$i, j, x, b \cdot \chi_L(x)$) to the adversary (controlling the network).

Roughly, (over-simplified) the prover sends (x, b) to the ideal functionality, that send (x, b) to the verifier if $x \in L$ and $(x, 0)$ otherwise. The extra bit b is used to model provers that fail in proving true statements (as in the definition of IdealZK, one can never force a prover to prove a statement if he does not want to do so).

The technical differences between the IdealZK and the MemberZK functionalities are as follows. While in the IdealZK functionality the prover needs to supply both a witness and a statement, in the MemberZK functionality it is sufficient for the prover to just supply a statement. In other words, the IdealZK functionality provides a *proof of knowledge*, while the ideal MemberZK functionality only provides a proof of validity. We call a model where parties have access to the MemberZK functionality, the MemberZK-Hybrid model.

Non-abusing adversaries: Note that the MemberZK functionality in general is not efficiently computable since it might be hard to decide whether a given instance belongs to an $\mathcal{NP}$ language without possessing a witness. In order to bypass this problem, the notion of *non-abusing* adversaries was introduced in [35]. Non-abusing adversaries exist only in the MemberZK-Hybrid model.

Definition 2 ([35]) (non-abusing adversaries) *We say that an adversary A' in the* MemberZK-Hybrid *model is* non-abusing *if it with overwhelming probability only invokes the* MemberZK *functionality with inputs of the type (x, b) such that $b = 0$ if $\chi_L(x) = 0$.*

Non-abusing adversaries (almost) always make "proper" usage of the MemberZK functionality. That is, with overwhelming probability, they will only send values (x, b) that makes the MemberZK functionality send (x, b) to the verifier. In other words, non-abusing adversaries (almost) always explicitly, and truthfully, tell the MemberZK functionality if the statement is false.

We note that if restricting the attention to only non-abusing adversaries in the MemberZK-hybrid model the MemberZK functionality is *efficiently* computable.

A.3 Definition: The $\text{ZK}^{1:\text{M}}$-Hybrid Model

In the multi-party setting, another useful abstraction is a so called *one-to-many* extension of the IdealZK functionality (introduced in [14]). That is a single party proves a statement to many verifiers.

The $\text{IdealZK}^{1:\text{M}}$ functionality: The one-to-many ideal zero-knowledge functionality is parameterized by an $\mathcal{NP}$-relation R_L and defined as follows:

- Upon receiving a message (ZK-prover,x, w) from a party P_i, the functionality sends (ZK-proof,$i, x, R_L(x, w)$) to all other parties P_j, $j \in [n]$, and also to the adversary (controlling the network).

Thus, slightly over-simplified, the prover sends an instance-witness pair (x, w) to the ideal functionality, which in turn, sends $(x, 1)$ to all other parties if w is a valid witness for $x \in L$ and $(x, 0)$ otherwise.

We call the setting in which the parties have access to ideal one-to-many zero-knowledge functionalities the $\mathsf{ZK}^{1:\mathrm{M}}$-$\mathsf{Hybrid}$ model.

For the case of multi-party computation, [14] show a protocol for universally composable multi-party computation protocol in the $\mathsf{ZK}^{1:\mathrm{M}}$-$\mathsf{Hybrid}$ model.

A.3.1 Secure Multi-Party Computation in the ZK$^{1:\mathrm{M}}$-Hybrid model

Using notation from [9, 30] we proceed to a formal definition of bounded-concurrent secure multi-party computation in the $\mathsf{ZK}^{1:\mathrm{M}}$-$\mathsf{Hybrid}$ model.

Definition 3 (security in the $\mathsf{ZK}^{1:\mathrm{M}}$-$\mathsf{Hybrid}$ model): *Let $m = m(k)$ be a polynomial, let f be a n-ary functionality and $\tilde{\Pi}$ be a n-party protocol in the* $\mathsf{ZK}^{1:\mathrm{M}}$-$\mathsf{Hybrid}$ *model. Protocol $\tilde{\Pi}$ is said to* t-securely compute f under m-bounded concurrent composition *if for every* $\mathsf{ZK}^{1:\mathrm{M}}$-$\mathsf{Hybrid}$ *model non-uniform probabilistic polynomial-time adversary $\tilde{A}$, there exists an ideal-model non-uniform probabilistic expected polynomial-time adversary S, such that for all families of polynomial-size input-deciding circuits $X^1 = \{X^1_k\}_{\{k\in N\}}, ..., X^n = \{X^n_k\}_{\{k\in N\}}$, every $z \in \{0,1\}^*$ and every $I \subset [n]$ with $|I| < t$,*

$$\left\{\mathrm{IDEAL}_{f,I,S}(X^1_k, ..., X^n_k, z)\right\}_{k\in N} \stackrel{\mathrm{c}}{\equiv} \left\{\mathrm{ZK}^{1:\mathrm{M}}\text{-}\mathrm{HYBRID}^m_{\tilde{\Pi},I,\tilde{A}}(X^1_k, ..., X^n_k, z)\right\}_{k\in N}$$

That is, concurrent executions of $\tilde{\Pi}$ with $\tilde{A}$ cannot be distinguished from concurrent invocations of f with S in the ideal model.

A special-case of the powerful result of Canetti et al. [14] yields the following theorem,[6]

Theorem 1 ([14]) *Assume the existence of enhanced trapdoor permutations. Then, for any n-party functionality f and for any polynomial m there exists a protocol $\tilde{\Pi}$ in the* $\mathsf{ZK}^{1:\mathrm{M}}$-$\mathsf{Hybrid}$ *model that $(n-1)$-securely computes f under m-bounded concurrent composition.*

Unfortunately the protocol of [14] is not constant-round. However, as was noted by [29], in the case of static adversaries, the compilation technique of Canetti et al [14] can be used on the constant-round secure multi-party computation protocol of Beaver, Micali and Rogaway [7, 39] for semi-honest adversaries, yielding a constant-round protocol.

Theorem 2 ([14, 7, 39]) *Assume the existence of enhanced trapdoor permutations. Then, for any n-party functionality f and for any polynomial m there exists a* constant-round *protocol $\tilde{\Pi}$ in the* $\mathsf{ZK}^{1:\mathrm{M}}$-$\mathsf{Hybrid}$ *model that $(n-1)$-securely computes f under m-bounded concurrent composition.*

A.3.2 Attempts to Realize the ZK$^{1:\mathrm{M}}$-Hybrid Model

Canetti et al [14] have shown that the $\mathsf{ZK}^{1:\mathrm{M}}$-$\mathsf{Hybrid}$ model can be realized in the Common Reference String (CRS) model. However, as recently pointed out in [34], the traditional usage of simulation techniques in the CRS model is problematic. In particular, there are certain "natural" security properties (e.g. the notion of *deniability*) that are not covered by the standard simulation definition in the CRS model. We point out that the protocol of [14] indeed relies on such simulation techniques.

[6]The protocol of [14] in fact originally relies on the use of a broadcast channel. The result stated is obtained by using the implementation of a broadcast channel by Goldwasser and Lindell [27].

B Realizing the $\mathsf{ZK}^{1:\mathrm{M}}$-Hybrid model

Motivated by the powerful compilation technique of [14] (see Theorem 2), we wish to remove the usage of the $\mathsf{IdealZK}^{1:\mathrm{M}}$ functionality from their protocol. If done in an appropriate way, this would give rise to a concurrently composable multi-party protocol, for any number of corrupted parties, in the real model without the use of set-up assumption.

In order to achieve this task we give a general transformation, showing that *any* protocol in the $\mathsf{ZK}^{1:\mathrm{M}}$-Hybrid model can be transformed into a real life protocol with the following two extra properties:

(Simulatability) For every bounded-concurrent real life adversary (corrupting an arbitrary number of parties), there exist a bounded-concurrent $\mathsf{ZK}^{1:\mathrm{M}}$-Hybrid model adversary such that the output of both the $\mathsf{ZK}^{1:\mathrm{M}}$-Hybrid adversary and the honest parties is indistinguishable from the real life adversary and the honest parties.

(Round-efficiency) The real-life protocol has the same round-complexity (up to a constant factor) as the $\mathsf{ZK}^{1:\mathrm{M}}$-Hybrid protocol.

In particular, such a transformation can be applied to the protocol obtained by applying the compilation of [14] on the protocol of [7] (see Theorem 2), resulting in a constant-round bounded-concurrent multi-party computation protocol for any number of corrupted parties.

More formally,

Theorem 3 *Let $\tilde{\Pi}$ be a n-party protocol in the $\mathsf{ZK}^{1:\mathrm{M}}$-Hybrid model, and let $m = poly(k)$. Then, assuming the existence of enhanced trapdoor permutations and collision resistant hash functions, there exists a protocol Π for the real model with the following properties:*

1. **(Simulatability)** *For every real-model non-uniform probabilistic polynomial-time adversary A participating in protocol Π, there exists an non-uniform probabilistic expected polynomial-time $\tilde{A}$ for the protocol $\tilde{\Pi}$, such that for all families of polynomial-size input-deciding circuits $X^1 = \{X_k^1\}_{\{k \in N\}}, ..., X^n = \{X_k^n\}_{\{k \in N\}}$, every $z \in \{0,1\}^*$ and every $I \subset [n]$,*

$$\left\{\mathrm{ZK}^{1:\mathrm{M}}\text{-}\mathrm{HYBRID}^{m}_{\tilde{\Pi},I,\tilde{A}}(X_k^1, ..., X_k^n, z)\right\}_{k \in N} \stackrel{\mathrm{c}}{\equiv} \left\{\mathrm{REAL}^{m}_{\Pi,I,A}(X_k^1, ..., X_k^n, z)\right\}_{k \in N}$$

   That is, concurrent executions of $\tilde{\Pi}$ with $\tilde{A}$ in the $\mathsf{ZK}^{1:\mathrm{M}}$-Hybrid model cannot be distinguished from concurrent executions of Π with A in the real model.

2. **(Round-efficiency)** *The protocol Π uses the same round complexity as $\tilde{\Pi}$ up to a constant factor.*

Theorem 3 is proved in three steps:

1. Reduce the use of the $\mathsf{IdealZK}^{1:\mathrm{M}}$ functionality to a the use of the $\mathsf{IdealZK}$ functionality. (This reduction is, in fact, of independent interest. In particular, it provides a simpler proof of the multi-party case of [14]. See Section B.1).

2. Reduce the use of the $\mathsf{IdealZK}$ functionality to a the use of the $\mathsf{MemberZK}$ functionality. (Essentially this reduction was already done in [35]. We here simply note that the reduction of [35] also works in the multi-party setting).

3. Show how to transform any multi-party protocol using the MemberZK functionality into a protocol in the real model by instantiating the MemberZK functionality with protocol executions (not requiring calls to ideal functionalities). See Section B.3. We note that such a transformation was shown in [30] and in [35] for the special case of two-party protocols. Unfortunately, these transformations do seem to directly extend to the multi-party setting.

B.1 Step 1: Reducing IdealZK$^{1:\mathrm{M}}$to IdealZK

In this section we show how the IdealZK$^{1:\mathrm{M}}$ functionality can be implemented in the ZK-Hybrid model. The underlying idea is straight-forward and similar to the idea used by Goldwasser and Lindell [27] to implement a broadcast channel. Roughly, the protocol proceeds as follows: the prover gives ZK proofs to all parties using the IdealZK functionality. When a party receives a proof of a statement x it sends an acknowledgment to all other parties. When a party has received both a proof and an acknowledgment from all other parties it accepts a proof of the instance x. More formally,

Lemma 1 *Let $\tilde{\Pi}$ be a n-party protocol in the* ZK$^{1:\mathrm{M}}$-Hybrid *model, and let $m = poly(k)$. Then, there exists a protocol Π^* in the* ZK-Hybrid *model with the following properties:*

1. **(Simulatability)** *For every* ZK-Hybrid *model non-uniform probabilistic polynomial-time adversary A^* participating in protocol Π^*, there exists an non-uniform probabilistic expected polynomial-time $\tilde{A}$ for the protocol $\tilde{\Pi}$, such that for all families of polynomial-size input-deciding circuits $X^1 = \{X^1_k\}_{\{k \in N\}}, ..., X^n = \{X^n_k\}_{\{k \in N\}}$, every $z \in \{0,1\}^*$ and every $I \subset [n]$,*

$$\left\{\mathrm{ZK}^{1:\mathrm{M}}\text{-HYBRID}^{m}_{\tilde{\Pi},I,\tilde{A}}(X^1_k, ..., X^n_k, z)\right\}_{k \in N} \stackrel{c}{\equiv} \left\{\text{ZK-HYBRID}^{m}_{\Pi^*,I,A^*}(X^1_k, ..., X^n_k, z)\right\}_{k \in N}$$

   That is, concurrent executions of $\tilde{\Pi}$ with $\tilde{A}$ in the ZK$^{1:\mathrm{M}}$-Hybrid *model cannot be distinguished from concurrent executions of Π^* with A^* in the* ZK-Hybrid *model.*

2. **(Round-efficiency)** *The protocol Π^* uses the same round complexity as $\tilde{\Pi}$ up to a constant factor.*

Proof (of lemma 1): Our proof follows that of [27]. The protocol Π^* is obtained by replacing all IdealZK$^{1:\mathrm{M}}$ calls in $\tilde{\Pi}$ by the protocol in Figure 4.

Clearly the round-complexity of Π^* is at most a constant factor higher than the round-complexity of $\tilde{\Pi}$. It remains to show that security is preserved under the transformation.

Let A^* be a ZK-Hybrid model adversary attacking protocol Π^*. We construct a ZK$^{1:\mathrm{M}}$-Hybrid model adversary $\tilde{A}$ that internally incorporates A^* and externally forwards all messages in Π^*, that do not belong to the one-to-many ZK proof protocol, between A^* and the honest parties. Messages that belong to the one-to-many ZK proof protocol, however, are dealt with internally, as specified below. We treat each execution of the protocol independently, and separate the cases when the prover is corrupted and when it it honest:

- P_i (prover) is corrupted.
  1. In the first round of the one-to-many ZK protocol, the adversary A^*, controlling P_i, sends messages to the IdealZK functionality. $\tilde{A}$ emulates the ideal functionality and the responses of the honest parties for A^*. (While emulating the IdealZK functionality, $\tilde{A}$ records all values sent to the functionality. In particular, $\tilde{A}$, records the supposed witnesses w for statements x.)

A protocol implementing the $\mathsf{IdealZK}^{1:\mathrm{M}}$ functionality in the ZK-Hybrid model

Input: party P_i has input (x, w).

The protocol:

- For every $j \in [n]$, the prover P_i sends the message ($\mathsf{ZK\text{-}prover}$,$j, x, w$) to the IdealZK functionality.
- A party P_j receiving a message ($\mathsf{ZK\text{-}proof}$,i, x, b) from the IdealZK functionality sends an acknowledgment ($\mathsf{received\ ZK}^{1:\mathrm{M}}\mathsf{\text{-}proof}$,$i, x, b$) to all other parties (except the prover P_i and itself).
- Party P_j waits to receive the message ($\mathsf{received\ ZK}^{1:\mathrm{M}}\mathsf{\text{-}proof}$,$i, x, v$) from every other party (except the prover P_i). Then, P_j writes ($\mathsf{ZK}^{1:\mathrm{M}}\mathsf{\text{-}proof}$,$i, x, v$) on its transcript.

Figure 4: A one-to-many ZK proof.

2. Once all first-round messages have been delivered, $\tilde{A}$ proceeds as follows. If all (emulated) honest parties have received the same message ($\mathsf{ZK\text{-}proof}$, i, x, b) from the emulated IdealZK functionality, $\tilde{A}$ (controlling P_i) sends the message ($\mathsf{ZK}^{1:\mathrm{M}}\mathsf{\text{-}prover}$, i, x, w) to the $\mathsf{IdealZK}^{1:\mathrm{M}}$ functionality, where w is a witness recorded for the statement x.
3. Thereafter, $\tilde{A}$ continues the internal emulation of the honest parties for A^* until the protocol terminates. Whenever a given (emulated) honest party receives all of its second-round messages (the acknowledgments) and all these messages are equal, $\tilde{A}$ delivers output from the $\mathsf{IdealZK}^{1:\mathrm{M}}$ functionality to that party. (If A^* did not send the same first-round messages to all of the honest parties, $\tilde{A}$ still continues with the internal simulation, but does not send anything to the $\mathsf{IdealZK}^{1:\mathrm{M}}$ functionality.)

- P_i (prover) is honest.
  1. When $\tilde{A}$ receives the message ($\mathsf{ZK}^{1:\mathrm{M}}\mathsf{\text{-}prover}$, i, x, w) from the $\mathsf{IdealZK}^{1:\mathrm{M}}$ functionality, $\tilde{A}$ internally emulates a real execution of the one-to-many ZK protocol, with P_i as prover, for A^*.
  2. In the first-round of the protocol, $\tilde{A}$ directly sends messages ($\mathsf{ZK\text{-}prover}$, i, x, w)) from the (emulated) IdealZK functionality to all (corrupted) parties.
  3. $\tilde{A}$ thereafter proceeds with the emulation of the honest parties. As in the previous case, $\tilde{A}$ delivers the message ($\mathsf{ZK}^{1:\mathrm{M}}\mathsf{\text{-}prover}$, i, x, w) to a given honest party when that party receives all of its second-round messages (the acknowledgments) and all these messages are equal.

It remains to show the correctness of the simulation. First we note that from the construction of the internal emulation it directly follows that the the view of A^* when executing the protocol Π^* is identical to the view of A^* in the emulation by $\tilde{A}$ in protocol $\tilde{\Pi}$. Secondly we show that the output of the honest parties also is identical in the two executions. That is, the honest parties in Π^* add the message ($\mathsf{ZK}^{1:\mathrm{M}}\mathsf{\text{-}proof}$,$i, x, v$) to their transcript if and only if they receive the same message from the $\mathsf{IdealZK}^{1:\mathrm{M}}$ functionality in $\tilde{\Pi}$.

Let us start with the case that P_i is honest. This means that all honest parties will receive the same message from the IdealZK functionality in the emulated execution. Now $\tilde{A}$ only delivers the output from the $\mathsf{IdealZK}^{1:\mathrm{M}}$ functionality if the honest party receives all acknowledgments, which is the same condition as in the protocol. Thus, the output of the honest parties is identical in Π^* and $\tilde{\Pi}$.

Let us proceed to the case that P_i is corrupted. If P_i, controlled by A^*, sends the same message (through the IdealZK functionality) to all parties, we are back in the same case as when P_i is honest. On the other hand, if P_i does not do so, then $\tilde{A}$ will not send a message to the $\mathsf{IdealZK}^{1:\mathrm{M}}$ functionality. Thus, it only remains to check that none of the honest parties will accept the proof in the one-to-many ZK protocol, which follows directly from the construction. ■

Remark: We note that proof of the theorem also holds for the more demanding security definition of universal composability (even with respect to adaptive adversaries) (see [9] for a formal security definition). The one-to-many zero-knowledge protocol given in Figure 4 could therefore be used in [14] in order to simply their proof.

B.2 Step 2: Reducing IdealZK to MemberZK

It was shown in [35] that any *two-party* protocol Π^* in the ZK-Hybrid model can be transformed into a protocol Π' in the MemberZK-Hybrid model, such that any *non-abusing* adversary in the MemberZK-Hybrid model can be simulated in the ZK-Hybrid model. The proof of the theorem makes use of a generic transformation from zero-knowledge *proofs* into zero-knowledge *proofs of knowledge*. We remark that same proof goes through also in the multi-party setting. More formally,

Lemma 2 ([35]) *Let Π^* be a protocol in the* ZK-Hybrid *model, and let $m = poly(k)$. Then, assuming the existence of enhanced trapdoor permutations there exists a protocol Π' in the* MemberZK-Hybrid *model with the following properties:*

1. **(Simulatability of non-abusing adversaries)** *For every* non-abusing MemberZK-Hybrid *model non-uniform probabilistic polynomial-time adversary A' participating in protocol Π', there exists an non-uniform probabilistic expected polynomial-time A^* for the protocol Π^*, such that for all families of polynomial-size input-deciding circuits $X^1 = \{X_k^1\}_{\{k \in N\}}, ..., X^n = \{X_k^n\}_{\{k \in N\}}$, every $z \in \{0,1\}^*$ and every $I \subset [n]$,*

$$\left\{\text{ZK-HYBRID}^m_{\Pi^*,I,A^*}(X_k^1, ..., X_k^n, z)\right\}_{k \in N} \stackrel{c}{\equiv} \left\{\text{MEMBERZK-HYBRID}^m_{\Pi',I,A'}(X_k^1, ..., X_k^n, z)\right\}_{k \in N}$$

   That is, concurrent executions of Π^* with A^* in the ZK-Hybrid *model cannot be distinguished from concurrent executions of Π' with A' in the* MemberZK-Hybrid *model.*

2. **(Round-efficiency)** *The protocol Π' uses the same round complexity as Π^* up to a constant factor.*

B.3 Step 3: Realizing the MemberZK-Hybrid Model

The final step in the realization of the $\mathsf{ZK}^{1:\mathrm{M}}$-Hybrid model, is to show how to transform any protocol Π' in the MemberZK-Hybrid model into a protocol Π in the real model. This should be done in a way that enables simulation of any adversary in the real model by a *non-abusing* adversary in the MemberZK-Hybrid model. More formally,

Lemma 3 *Let Π' be a protocol in the* MemberZK-Hybrid *model, and let $m = poly(k)$. Then, assuming the existence of collision resistant hash functions, there exists a protocol Π with the following properties:*

1. **(Simulatability with non-abusing simulator)** *For every real-model non-uniform probabilistic polynomial-time adversary A participating in protocol Π, there exists an* non-abusing *non-uniform probabilistic expected polynomial-time A' for the protocol Π', such that for all families of polynomial-size input-deciding circuits $X^1 = \{X_k^1\}_{\{k \in N\}}, ..., X^n = \{X_k^n\}_{\{k \in N\}}$, every $z \in \{0,1\}^*$ and every $I \subset [n]$,*

$$\left\{\text{MEMBERZK-HYBRID}^m_{\Pi',I,A'}(X_k^1, ..., X_k^n, z)\right\}_{k \in N} \stackrel{c}{\equiv} \left\{\text{REAL}^m_{\Pi,I,A}(X_k^1, ..., X_k^n, z)\right\}_{k \in N}$$

That is, concurrent executions of Π' with A' in the MemberZK-Hybrid *model cannot be distinguished from m concurrent executions of Π with A in the real model.*

2. **(Round-efficiency)** *The protocol Π uses the same round complexity as Π' up to a constant factor.*

The proof of the above lemma is given in section C.
The composition theorem (Theorem 3) now follows from the lemma 1, 2, 3.

B.4 Reaching the goal of bounded-concurrent multi-party computation

In order to obtain a protocol for bounded-concurrent multi-party computation it only remains to apply the composition theorem (Theorem 3) on the protocol from Theorem 2. Thus, combining Theorem 3 with Theorem 2 yields,

Theorem 4 *Assume the existence of enhanced trapdoor permutations and collision resistant hash functions. Then, for any multi-party functionality f and for any polynomial m there exists a constant-round protocol Π that $(n-1)$-securely computes f under m-bounded concurrent composition.*

We mention the fact that the protocol has a simulator running in *strict* polynomial time, as opposed to the relaxed notion of *expected* polynomial time. See [4, 22] for a discussion on the subject.

C Proof of Lemma 3

On a high level, the proof is a generalization of the proofs of [30] and [35]. The protocol Π is constructed from protocol Π' by instantiating the MemberZK functionality using our special-purpose zero-knowledge protocols $c\mathcal{ZK}_1, ..., c\mathcal{ZK}_n$. Specifically, we let all invocations of the MemberZK functionality that are initiated by party P_i (acting as prover) be replaced with an execution of the protocol $c\mathcal{ZK}_i$.

C.1 The Simulator

We proceed to the description of the simulator A' in Π' for the real-life adversary A in Π. Let $I \subset [1, ..n]$ be the set of corrupted parties (that is $\forall i \in I$ party P_i is controlled by A).

We start by describing the adversary A' for Π' in the MemberZK model and thereafter prove the validity of the simulation.

A' internally incorporates the real model adversary A for Π and externally forwards all messages that are not part of any of the protocols $c\mathcal{ZK}_1, ..., c\mathcal{ZK}_n$ between the parties and A. Messages from the protocols $c\mathcal{ZK}_1, ..., c\mathcal{ZK}_n$ are dealt with internally.

Whenever A (controlling party P_i) wishes to prove a statement x, using protocol $c\mathcal{ZK}_i$ to the honest party P_j, A' internally invokes the simulation procedure $\mathtt{verify_i(j,x)}$ defined as follows:

$\mathtt{verify_i(j,x)}$:

- Internally run the honest verifier strategy, for the protocol $c\mathcal{ZK}_i$, against A (controlling P_i), but instead of sending a truly random strings as challenges (in Stage 1 and 2 of the protocol), picks a n bit long random string, expands it to the apprioriate length of the challenge (r_1 or r_2) through a pseudorandom generator (the existence of such a generator is implied by our assumptions [28]) and sends the expanded string as a challenge.
- If the honest verifier strategy accepts the proof, A', controlling party P_i, sends (MemberZK-prover, $j, x, 1$) to the MemberZK functionality, and otherwise it sends (MemberZK-prover, $j, x, 0$).

(We note that the technique of using a pseudorandom generator in order to generate the verifier's messages was used already in the two-party setting [35].)

Whenever A' (controlling some party P_i) receives a message of the type (MemberZK-proof,$j, x, 1$) from the ideal functionality (this means that A is waiting for a proof of the statement x from the honest party P_j, using $c\mathcal{ZK}_j$), A' invokes the simulation procedure $\mathtt{simulate_j(x)}$ defined as follows:

$\mathtt{simulate_j(x)}$:
Internally run the "non rewinding" simulator of the protocol $c\mathcal{ZK}_j$ for A:

- Commit to the verifier's (A's) next-message function in Stage 1.
- Commit to the verifier's (A's) next-message function in Stage 2.
- In Stage 3 of the protocol, act as the honest prover using the code of the verifier (from Stage 1) as a "fake" witness.

Note: In order to succeed in Stage 3, it is necessary that the simulator has a description of length bounded by $\ell(k) - k = m(\gamma \cdot 4k^2 + \mathtt{length}(\Pi'))$ of all messages send by the honest parties before the adversary A (controlling some party P_i) answers in Stage 1 of the protocol. We here note that since there is a bound on m concurrent executions, the total length of all messages sent between two consecutive messages in the protocol, not counting the "challenges" r_1, r_2, is bounded by $m(\gamma \cdot 2k^2 + \mathtt{length}(\Pi'))$ (See Section 3.2). The challenges r_1, r_2 are long though (at least of length $\ell(k)$ actually), but since they are constructed using a pseudorandom generator, they have a description that is bounded by, say, k^2. Since each of the protocols $c\mathcal{ZK}_1, ..., c\mathcal{ZK}_n$ contain only 2 challenges, it follows that the total length of the description needed is bounded by $\ell(k) - k$.

The only remaining case is when the adversary A' (controlling the network) receives the message (MemberZK-proof,$i', j', x, 1$) from the ideal functionality (this means that an honest party P_j' in waiting for a proof from the honest party P'_i using protocol $c\mathcal{ZK}'_i$). In this case A' emulates an interaction of the protocol $c\mathcal{ZK}_i$ between the honest parties P'_i and P'_j in Π, for A. This can be done by invoking the procedure $\mathtt{simulate}_{\mathtt{i'}}(\mathtt{x})$ on the code of the honest verifier.

We continue by showing the correctness of the simulation. It follows from the two following claims.

Claim 1 *The adversary A' in Π' is non-abusing. That is, A' only accepts a proof of x from A when $x \notin L$ with negligible probability.*

Claim 2 *The output of A and the honest parties in Π is indistinguishable from the output of A' and the honest parties in Π', i.e.*

$$\left\{\text{MEMBERZK-HYBRID}^m_{\Pi',I,A'}(X^1_k, ..., X^n_k, z)\right\}_{k\in N} \stackrel{c}{\equiv} \left\{\text{REAL}^m_{\Pi,I,A}(X^1_k, ..., X^n_k, z)\right\}_{k\in N}$$

We start by proving that A' in Π' is non-abusing.

Proof (of claim 1): The claim follows from the "soundness" of the protocols $c\mathcal{ZK}_1, ..., c\mathcal{ZK}_n$ in the simulated setting. Essentially, what we want to prove is that for any protocol $c\mathcal{ZK}_i$, the soundness of $c\mathcal{ZK}_i$ is left intact even when simulating all of the other protocols $c\mathcal{ZK}_j$, $j \neq i$. (In other words, the protocols $c\mathcal{ZK}_1, ..., c\mathcal{ZK}_n$ are all simulation-sound with respect to each other.)

Assume, for contradiction that there exist an adversary A and families of input deciding circuits $X^1, ..., X^n$ such that for infinitely many z, A convinces A' of a false statement (using one of the protocols $c\mathcal{ZK}_i, i \in I$) with non-negligible probability. (Recall that A tries to "cheat" in a protocol $c\mathcal{ZK}_i$, where $i \in I$, while receiving "simulated" proof of possibly false statements using proofs cZK_j, where $j \notin I$.)

We use A to construct a stand-alone adversary P^* that will breach the soundness of a single execution of a protocol $c\mathcal{ZK}_i, i \in I$. Let $p(k)$ denote the total number of sessions initiated by by A. (Note that $p(k)$ can be greater than the bound on concurrency $m(k)$ since all the sessions do not necessarily have to overlap.) P^* internally incorporates A, $X^1, ..., X^n$, uniformly chooses $i \in I$, and one of the maximum $\gamma p(n)$ executions of the the protocol $c\mathcal{ZK}_i$ in the concurrent executions of Π. P^* then internally emulates the simulation of A' for A (let us call this emulation E) for all executions of the protocols $c\mathcal{ZK}_1, ..., c\mathcal{ZK}_n$ except for the selected execution. When the selected execution is reached, P^* externally forwards all messages between A and an honest verifier V.

Let us start by assuming that P^* is able to perform such an emulation E. In this case, the probability that P^* convinces V is at most negligibly smaller than $1/(n\gamma p(k))$ times the probability that A succeeds in convincing A' of a false statement. (The negligible difference is due to the fact that A' uses pseudorandomness when verifying, while V uses real randomness. Formally, we show the above claim in two steps. First we note that P^* is able to convince a verifier, that obtains its randomness through a pseudorandom generator and then follows the honest verifier strategy, of a false statement with non-negligible probability. Next, we claim that if the soundness for an interactive argument can be broken for such a verifier, then it can also be broken for the honest verifier, or else we would have a distinguisher for the pseudorandom generator. This distinguisher is obtained by fixing the random coins of a cheating prover.) This, thus, contradicts the stand-alone soundness of $c\mathcal{ZK}_i$.

The above argument, however, assumes that P^* is able to emulate the simulation of A' for all but one uniformly selected execution of $c\mathcal{ZK}_i$. Unfortunately, P^* will not be able to run the simulator A', since A', in order to run the procedure $\mathtt{simulate}_{\mathtt{j}}$ (to simulate protocols $c\mathcal{ZK}_j$, where $j \notin I$) needs to have a short description of the messages sent by the honest parties, including the messages sent when verifying $c\mathcal{ZK}_i$. In the above scenario, however, since one of the executions of $c\mathcal{ZK}_i$ is externally forwarded to an honest verifier, the simulator does not have a short description of the honest verifier's challenges r_1, r_2. To solve this problem, we construct an emulator E for A', which acts as A' but instead of invoking the procedure $\texttt{simulate}_\texttt{j}$ runs the following procedure $\texttt{simulate}'_\texttt{j}(\texttt{x})$.

$\texttt{simulate}'_\texttt{j}(\texttt{x})$:
Internally run the "non rewinding" simulator of the protocol $c\mathcal{ZK}_i$ for A:

- Commit to the verifier's (A's) next-message function in Stage 1.
- Commit to the verifier's (A's) next-message function in Stage 2.
- In Stage 3 of the protocol act as the honest prover using the code of the verifier and a short description of all messages sent by the honest parties as a "fake" witness.

  Note: Except for the the "long" challenges r_1, r_2 sent by the external verifier of $c\mathcal{ZK}_i$ the simulator has a description of all messages sent to the adversary that is shorter than $\ell(k) - k$ (using the pseudorandom seeds). In order to succeed in Stage 3 of the protocol, even in the presence of these 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 *one* of the slots of $c\mathcal{ZK}_i$. We separate two cases:

  If both the "long" challenges r_1, r_2, sent by the external verifier, 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 the construction of the protocols it follows that the length of either the first or the second challenge in $c\mathcal{ZK}_j$ is at least $\ell(k)$ longer than the corresponding challenge in $c\mathcal{ZK}_i$ (sent by the external verifier). Thus there exist a slot in $c\mathcal{ZK}_j$ such that even if we include the verifier's challenge, from the protocol $c\mathcal{ZK}_i$, in whole, in the description, we still have $\ell(k) - k$ to describe the other messages, which means that simulation can be performed.

Note that the only difference between $\texttt{simulate}_\texttt{j}$ and $\texttt{simulate}'_\texttt{j}$ is the choice of witness used in stage 3 of the proof. While $\texttt{simulate}_\texttt{j}$ always picks the first slot, $\texttt{simulate}'_\texttt{j}$ picks the "free" slot (which can be any of the two slots). Thus if A, when interacting with A' succeeds in convincing A' of a false statement in one of the executions, then A, when interacting with E, will also succeed in doing so, or else the witness indistinguishability of the protocols $c\mathcal{ZK}_1, ..., c\mathcal{ZK}_n$ would be broken. It follows that P^* incorporating E succeeds in breaking the soundness of a stand-alone execution of $c\mathcal{ZK}_i$. ■

The fact that the adversary A' is non-abusing will be helpful in proving the indistinguishability of the simulation. Actually, this implies that the honest parties in Π', with overwhelming probability, receive the same verdict from the ideal functionality that A' decides, i.e., that the ideal functionality does not change the decision of A'. In other words, when A' performs the verification of a statement x, it runs the procedure $\texttt{verify}_\texttt{i}$, which makes a decision b to accept or reject. A' thereafter send (x, b) to the ideal functionality. The fact that A' is non-abusing means that with overwhelming probability the honest party (being the recipient of the proof) will receive (x, b) from the ideal functionality. We proceed to show the indistinguishability of the simulation.

Proof (of claim 2): We show the claim by a several hybrid experiments.

First, let $\hat{\Pi}$ be the same protocol as Π except that whenever the honest parties engage as verifiers of the protocols $c\mathcal{ZK}_i$, where $i \in I$, they uses step 1 of the procedure $\texttt{verify}_\texttt{i}$ instead of the honest verifier strategy, i.e., they verify the protocols $c\mathcal{ZK}_i$, $i \in I$, using pseudorandom strings as a challenges instead of a truly random strings. Clearly, since the only difference between the role of the honest parties in Π and in $\hat{\Pi}$ is in using pseudorandom strings instead of truly random strings, the output of the honest parties and A in Π is indistinguishable from the output of the honest parties and A in $\hat{\Pi}$. That is,

$$\left\{\text{REAL}^m_{\hat{\Pi},I,A}(X^1_k, ..., X^n_k, z)\right\}_{k\in\text{N}} \stackrel{\text{c}}{\equiv} \left\{\text{REAL}^m_{\Pi,I,A}(X^1_k, ..., X^n_k, z)\right\}_{k\in\text{N}}$$

Now, let Π'' be the same protocol as Π' except that all invocations of the $\mathsf{MemberZK}$ functionality performed by corrupted parties P_i, $i \in I$ are instantiated with an execution of $c\mathcal{ZK}_i$. (That is Π'' is a protocol where all proofs by corrupted parties P_i, $i \in I$ are performed using $c\mathcal{ZK}_i$, while the proofs by honest parties P_j, $j \notin I$ are performed using the ideal functionality.)

Let A'' be defined in analogy with A', i.e. A'' incorporates A and externally forwards all messages, except messages in cZK_i, $i \in I$, between the honest parties P_j, $j \notin I$ and A, while internally simulating executions of $c\mathcal{ZK}_i$, $i \in I$, using $\texttt{verify}_\texttt{i}$.

We start by noting that the only difference between the real-life execution $\hat{\Pi}$ with A and the $\mathsf{MemberZK}$-Hybrid model execution Π'' with A'' is that in $\hat{\Pi}$, honest parties P_j, $j \notin I$ directly retain the decisions made by the procedure $\texttt{verify}_\texttt{i}$, while in Π'' this decision thereafter has to go through the $\mathsf{MemberZK}$ functionality. This means that if, with overwhelming probability, the decision made by A'' is the same as the verdict of the ideal functionality (i.e. that A'' is non-abusing), the output of the honest parties and A'' in Π'' is indistinguishable from the output of honest parties and A in $\hat{\Pi}$.

We show that A'' in Π'' is non-abusing using the same technique as was used to show that A' is non-abusing. The proof of this fact is, however, much simplified as we can directly simulate A''.

(Assume, for contradiction that there exist an adversary A, and input-deciding circuits $X^1, ..., X^n$, such that for infinitely many z, A convinces A'' of a false statement with non-negligible probability. Construct the stand-alone adversary P^* that internally incorporates A, $X^1, ..., X^n$, uniformly chooses $i \in I$, and one of the maximum $\gamma p(n)$ executions of $c\mathcal{ZK}_i$ in the concurrent executions of $\hat{\Pi}$. P^* then internally runs the simulation $\texttt{verify}_\texttt{i}$ for all executions of $c\mathcal{ZK}_i$ except for the selected execution. When the selected execution is reached, P^* externally forwards all messages between A and an honest verifier V. The probability that P^* convinces V is at most negligibly smaller than $1/(n\gamma p(k))$ times the probability that A succeeds in convincing A'' of a false statement, which contradicts the stand-alone soundness of $c\mathcal{ZK}_i$.)

Thus,

$$\left\{\text{MEMBERZK-HYBRID}^m_{\Pi'',I,A''}(X^1_k, ..., X^n_k, z)\right\}_{k\in\text{N}} \stackrel{\text{c}}{\equiv} \left\{\text{REAL}^m_{\hat{\Pi},I,A}(X^1_k, ..., X^n_k, z)\right\}_{k\in\text{N}}$$

It remains to show that

$$\left\{\text{MEMBERZK-HYBRID}^{m}_{\Pi',I,A'}(X^1_k,...,X^n_k,z)\right\}_{k\in\mathbb{N}} \stackrel{c}{\equiv} \left\{\text{MEMBERZK-HYBRID}^{m}_{\Pi'',I,A''}(X^1_k,...,X^n_k,z)\right\}_{k\in\mathbb{N}}$$

Here again, we note that since (by claim 1) the adversary A' is non-abusing, the honest parties P_j, $j \notin I$ in Π' receives, with overwhelming probability, the same verdict from the ideal functionality that A' decides. We have previously shown that A'' also is non-abusing, which implies that the honest parties P_j in Π'', also, with overwhelming probabilty, receive the same verdict as A'' decides. Thus, with overwhelming probability, the only difference between an execution of Π' with A' and Π'' with A'' is that in Π', whenever executing the protocol $c\mathcal{ZK}_j$, A interacts with the procedure $\texttt{simulate}_\texttt{j}$, while in Π'' A interacts with the honest prover. But, by the indistinguishability of the simulation $\texttt{simulate}_\texttt{j}$ in this setting, it follows that the output of both the honest parties and A' in Π' is indistinguishable from the output of honest parties and A'' in Π''.

Now, summing up,

$$\left\{\text{MEMBERZK-HYBRID}^{m}_{\Pi',I,A'}(X^1_k,...,X^n_k,z)\right\}_{k\in\mathbb{N}} \stackrel{c}{\equiv} \left\{\text{REAL}^{m}_{\Pi,I,A}(X^1_k,...,X^n_k,z)\right\}_{k\in\mathbb{N}}$$

which concludes the proof. ■