Concurrent Zero Knowledge in the Bounded Player Model

VIPUL GOYAL
Microsoft Research
India
vipul@microsoft.com

ABHISHEK JAIN
UCLA
USA
abhishek@cs.ucla.edu

RAFAIL OSTROVSKY
UCLA
USA
rafail@cs.ucla.edu

SILAS RICHELSON
UCLA
USA
sirichel@math.ucla.edu

IVAN VISCONTI
University of Salerno
ITALY
visconti@dia.unisa.it

Abstract

In this paper we put forward the *Bounded Player Model* for secure computation. In this new model, the number of players that will ever be involved in secure computations is bounded, but the number of computations has no a priori bound. Indeed, while the number of devices and people on this planet can be realistically estimated and bounded, the number of computations these devices will run can not be realistically bounded. We stress that in the *Bounded Player model*, in addition to no a priori bound on the number of sessions, there is no synchronization barrier, no trusted party, and simulation must be performed in polynomial time.

In this setting, we achieve concurrent Zero Knowledge (cZK) with *sub-logarithmic* round complexity. Our security proof is (necessarily) non-black-box, our simulator is "straight-line" and works as long as the number of rounds is $\omega(1)$.

We further show that unlike previously studied relaxations of the standard model (e.g., bounded number of sessions, timing assumptions, super-polynomial simulation), concurrent-secure computation is still impossible to achieve in the Bounded Player model. This gives evidence that our model is "closer" to the standard model than previously studied models, and study of this model might shed light on constructing round efficient concurrent zero-knowledge in the standard model as well.

1 Introduction

Zero-knowledge proofs, introduced in the seminal work of Goldwasser, Micali and Rackoff [GMR85], are a fundamental building block in cryptography. Loosely speaking, a zero-knowledge proof is an interactive proof between two parties — a prover and a verifier — with the seemingly magical property that the verifier does not learn anything beyond the validity of the statement being proved. Subsequent to their introduction, zero-knowledge proofs have been the subject of a great deal of research (see, for example, [BSMP91, DDN91, Ost91, OW93, DNS98, CGGM00, Bar01, IKOS09]), and have found numerous applications in cryptography (e.g., [GMW87, FFS88]).

Concurrent zero knowledge. The original definition of zero knowledge is only relevant to the "stand-alone" setting where security holds only if the protocol runs in isolation. As such, unfortunately, it does not suffice if one wishes to run a zero-knowledge proof over a modern network environment, such as the Internet. Towards that end, Dwork, Naor and Sahai [DNS98] initiated the study of *concurrent zero-knowledge* (cZK) proofs that remain secure even if several instances of the protocol are executed concurrently under the control of an adversarial verifier. Subsequent to their work, cZK has been the subject of extensive research, with a large body of work devoted to studying its round-complexity. In the standard model, the round-complexity of cZK was improved from polynomial to slightly super-logarithmic in a sequence of works [RK99, KP01, PRS02]. In particular, the $\tilde{O}(\log k)$-round construction of [PRS02] nearly matches the lower bound of $\tilde{\Omega}(\log k)$ w.r.t. black-box simulation [CKPR01] (see also [KPR98, Ros00]).

Despite a decade of research, the $\tilde{O}(\log k)$-round construction of [PRS02] is still the most round-efficient cZK protocol known. Indeed, the lower bound of [CKPR01] suggests that a breakthrough in non-black-box simulation techniques is required to achieve cZK with sub-logarithmic round complexity.[1]

Round-efficient cZK in relaxed models. While the round-complexity of cZK in the standard model still remains an intriguing open question, a long line of work has been dedicated towards constructing round-efficient cZK in various relaxations of the standard model. Notable mentions include the bounded-concurrency model [Bar01], the bare public key model [CGGM00], the super-polynomial time simulation (SPS) model [Pas03], timing model [DNS98], and, various setup models (such as common reference string [BSMP91], etc.). Below, we briefly discuss the state of the art on some of these models.

BOUNDED CONCURRENCY MODEL. An interesting relaxation of the standard model (and related to our setting) that has been previously studied is the bounded-concurrency model [Bar01], where an apriori bound is assumed over the number of sessions that will ever take place (in particular, this bound is known to the protocol designer). It is known how to realize constant-round bounded cZK [Bar01], and also constant-round bounded-concurrent secure two-party and multi-party computation [Lin03a, PR03, Pas04].

Even though our model can be seen as related to (and a generalization of) the bounded concurrency model, the techniques used in designing round efficient bounded concurrent zero-knowledge do not seem to carry over to our setting. In particular, if there is even a single player that runs an unbounded number of sessions, the simulation strategies in [Bar01, Lin03a, PR03, Pas04] breakdown completely. This seems inherent because of the crucial difference this model has from our setting (which can understood by observing that general concurrent secure computation is possible in the bounded concurrent setting but impossible in our setting).

BARE PUBLIC KEY AND OTHER PREPROCESSING MODELS. The zero-knowledge pre-processing model was proposed in [KMO89] in the stand-alone setting and in [CO99] in the context of cZK. In [CO99], interaction is needed between all the involved players in a preprocessing phase. Then, after a synchronization-barrier is passed, the preprocessing is over and actual proofs start. Interactions in each phase can take place concurrently, but the two

[1] In this paper we only consider results based on standard complexity-theoretic and number-theoretic assumptions; in particular, we not consider "non-falsifiable" assumptions such as the knowledge of exponent assumption.

phases can not overlap in time. An improved model was later proposed in [CGGM00] where the preprocessing is required to be non-interactive, and the model is called "Bare Public-Key" (BPK) model, since the non-interactive messages played in the preprocessing can be considered as public announcements of public keys. In this model it is known how to obtain constant-round concurrent zero knowledge under standard assumptions [SV12].

The crucial restriction of the BPK model is that all players who wish to ever participate in protocol executions must be fixed during the preprocessing phase, and new players cannot be added "on-the-fly" during the proof phase. We do *not* make such a restriction in our work and as such, the techniques useful in constructing secure protocols in the BPK model have limited relevance in our setting. In particular, constant round concurrent ZK is known to exist in the BPK model using only black-box simulation, while in our setting, non-black-box techniques are *necessary* to construct a sublogarithmic round concurrent ZK protocol.

OTHER MODELS. Round efficient concurrent zero-knowledge is known in a number of other models as well (which do not seem to be directly relevant to our setting). In the SPS model [Pas03], the zero-knowledge simulator is allowed to run in super-polynomial time, as opposed to running in polynomial time (as per the standard definition of [GMR85]). Indeed, this relaxation has yielded not only constant-round cZK [Pas03], but also concurrent-secure computation [LPV09, CLP10, GGJS12]. This stands in contrast to the standard model, where concurrent-secure computation is known to be impossible to achieve [Lin04, Lin03b, CKL03, CF01]. Similarly, in the timing model [DNS98], where an upper-bound is assumed on the delivery time needed of a message (and therefore the adversary is assumed to have only limited control of the communication network), constant-round cZK is known [DNS98, Gol02, PTV10], as well as is multi-party computation secure w.r.t. general concurrent composition [KLP05]. Finally, note that similar results hold in popular models such as the common reference string [BSMP91, SCO$^+$01, CLOS02], key registration [BCNP04], etc.

Our Question. While the above relaxations of the standard model discussed above have their individual appeal, each of these models suffers from various drawbacks, either w.r.t. the security guarantees provided (e.g., as in the case of the SPS model), or w.r.t. the actual degree of concurrency tolerated (e.g., as in the case of the timing model). Indeed, despite the extensive amount of research over the last decade, the round-complexity of cZK still remains open. In this work, we ask the question whether it is possible to construct cZK protocols with sub-logarithmic round-complexity in a natural model that does not suffer from the drawbacks of the previously studied models; namely, it does not require any preprocessing, assumes no trusted party or timing assumptions or an a priori bound on the number of protocol sessions, and requires standard polynomial-time simulation and standard complexity assumptions.

1.1 Our Results

In our work, we construct a concurrent (perfect) zero-knowledge argument system with sub-logarithmic round-complexity in a mild relaxation of the standard model; we refer to this as the *Bounded Player model*. In this model we only assume that there is an a priori (polynomial) upper-bound on the total number of players that may ever participate in protocol executions. We do not assume any synchronization barrier, or trusted party, and the simulation must be performed in polynomial time. In particular, we do *not* assume any a priori bound on the number of sessions, and achieve security under unbounded concurrency. As such, our model can be viewed as a strengthening of the bounded-concurrency model.[2] Below, we give an informal statement of our main result.

Theorem 1 *Assuming dense crypto systems and claw-free permutations, there exists an $\omega(1)$-round concurrent perfect zero-knowledge argument system with concurrent soundness in the Bounded Player model.*[3]

[2]Note that an upper-bound on the total number of concurrent executions implies an upper-bound on the total number of players as well.

[3]We note that if one only requires *statistical* (as opposed to perfect) zero knowledge, then the assumption on claw-free permutations can be replaced by collision-resistant hash functions. We further note that our assumption on dense cryptosystems can be further relaxed to

Our security proof is (necessarily) non-black-box (see below), and the simulator of our protocol works in a "straight-line" manner. Our result is actually stronger since we only require a bound on the number of possible verifiers, while there is no restriction on the number of provers. We prove concurrent soundness since sequential and concurrent soundness are distinct notions in the Bounded Player model for the same reasons as shown by [MR01] in the context of the BPK model.

We stress that while our model bears some resemblance to the BPK model, known techniques from the BPK model are not applicable to our setting. Indeed, these techniques crucially rely upon the presence of the synchronization barrier between the pre-processing phase and the protocol phase, while such a barrier is not present in our model. As such, achieving full concurrency in our model is much harder and involves significantly different challenges.

We further show that the impossibility results of Lindell for concurrent-secure computation [Lin04] also hold in the Bounded Player model. This gives evidence that the Bounded Player model is much closer to the standard model than the previously studied models, and the study of this model might shed light towards the goal of constructing round efficient concurrent zero-knowledge in the standard model as well.

1.2 Our Techniques

Recall that in the Bounded Player model, the only assumption is that the total number of players that will ever be present in the system is a priori bounded. Then, an initial observation towards our goal of constructing sub-logarithmic round cZK protocols is that the black-box lower-bound of Canetti et al. [CKPR01] is applicable to our setting as well. Indeed, the impossibility result of [CKPR01] relies on an adversarial verifier that opens a polynomial number $\ell(k)$ of sessions and plays adaptively at any point of time, depending upon the transcript generated "so far". The same analysis works in the Bounded Player model, by assuming that the adversarial verifier registers a new key each time a new session is played. In particular, consider an adversarial verifier that schedules a session s_i to be contained inside another session s_j. In this case, a black-box simulator does not gain any advantage in the Bounded Player model over the standard model. The reason is that since the adversarial verifier of [CKPR01] behaves adaptively on the transcript at any point, after a rewind the same session will be played with a fresh new key, thus rendering essentially useless the fact that the session was already solved before. Note that this is the same problem that occurs in the standard model, and stands in contrast to what happens in the BPK model (where identities are fixed in the preprocessing and therefore do not change over rewinds).

From the above observation, it is clear that we must resort to non-black-box techniques. Now, a natural approach to leverage the bound on the number of players is to associate with each verifier V_i a public key pk_i and then design an FLS-style protocol [FLS90] that allows the ZK simulator to extract, in a non-black-box manner, the secret key sk_i of the verifier and then use it as a "trapdoor" for "easy" simulation. The key intuition is that once the simulator extracts the secret key sk_i of a verifier V_i, it can perform easy simulation of *all* the sessions associated with V_i. Then, since the total number of verifiers is bounded, the simulator will need to perform non-black-box extraction only an a priori bounded number of times (once for each verifier), which can be handled in a manner similar to the setting of bounded-concurrency [Bar01].

Unfortunately, the above intuition is misleading. In order to understand the problem with the above approach, let us first consider a candidate protocol more concretely. In fact, it suffices to focus on a preamble phase that enables non-black-box extraction (by the simulator) of a verifier's secret key since the remainder of the protocol can be constructed in a straightforward manner following the FLS approach. Now, consider the following candidate preamble phase (using the non-black-box extraction technique of [BL02]): first, the prover and verifier engage in a coin-tossing protocol where the prover proves "honest behavior" using a Barak-style non-black-box ZK protocol [Bar01]. Then, the verifier sends an encryption of its secret key under the public key that is determined from the output of the coin-tossing protocol.

trapdoor permutations by modifying our protocol to use the coin-tossing protocol of Barak and Lindell [BL02]. We leave this for the full version of the paper.

In order to analyze this protocol, we will restrict our discussion to the simplified case where only one verifier is present in the system (but the total number of concurrent sessions are unbounded). At this point, one may immediately object that in the case of a single verifier identity, the problem is not interesting since the Bounded Player model is identical to the bare-public key model, where one can construct four-round cZK protocols using rewinding based techniques. However, simulation techniques involving rewinding do not "scale" well to the case of polynomially many identities (unless we use a large number of rounds) and fail[4]. Moreover the use of Barak's [Bar01] straight-line simulation technique is also insufficient since it works only when the number of concurrent sessions is bounded (even when there is a single identity), but instead our goal is to obtain unbounded concurrent zero knowledge. In contrast, our simulation approach is "straight-line" for an unbounded number of sessions and scales well to a large bounded number of identities. Therefore, in the forthcoming discussion, we will restrict our analysis to straight-line simulation. In this case, we find it instructive to focus on the case of a single identity to explain our key ideas.

We now turn to analyze the candidate protocol. Now, following the intuition described earlier, one may think that the simulator can simply cheat in the coin-tossing protocol in the "inner-most" session in order to extract the secret key, following which all the sessions can be simulated in a straight-line manner, without performing any additional non-black-box simulation. Consider, however, the following adversarial verifier strategy: the verifier schedules an unbounded number of sessions in such a manner that the coin-tossing protocols in all of these sessions are executed in a "nested" manner. Furthermore, the verifier sends the ciphertext (containing its secret key) in each session only *after* all the coin-tossing protocols across all sessions are completed. Note that in such a scenario, the simulator would be forced to perform non-black-box simulation in an unbounded number of sessions. Unfortunately, this is a non-trivial problem that we do not know how to solve. More concretely, note that we cannot rely on techniques from the bounded-concurrency model since we cannot bound the total number of sessions (and thus, the total number of messages across all sessions). Further, all other natural approaches lead to a "blow-up" in the running time of the simulator. Indeed, if we were to solve this problem, then we would essentially construct a cZK protocol in the standard model, which remains an important open problem that we do not solve here.

In an effort to bypass the above problem, our first idea is to use multiple ($\omega(1)$, to be precise) preamble phases (instead of only one), such that the simulator is required to "cheat" in only one of these preambles. This, however, immediately raises a question: in which of the $\omega(1)$ preambles should the simulator cheat? This is a delicate question since if, for example, we let the simulator pick one of preambles uniformly at random, then with non-negligible probability, the simulator will end up choosing the first preamble phase. In this case, the adversary can simply perform the same attack as it did earlier playing only the first preamble phase, but for many different sessions so that the simulator will still have to cheat in many of them. Indeed, it would seem that any randomized oblivious simulation strategy can be attacked in a similar manner by simply identifying the first preamble phase where the simulator would cheat with a non-negligible probability.

Towards that end, our key idea is to use a specific probability distribution such that the simulator cheats in the first preamble phase with only negligible probability, while the probability of cheating in the later preambles increases gradually such that the "overall" probability of cheating is 1 (as required). Further, the distribution is such that the probability of cheating in the i^{th} preamble is less than a fixed polynomial factor of the total probability of cheating in one of the previous $i-1$ blocks. Very roughly speaking, this allows us to prevent the adversary from attacking the *first* preamble where the simulator cheats with non-negligible probability. More specifically, for any session, let us call the preamble where the simulator cheats the "special" preamble. Further, let us say that the adversary "wins" a session if he "stops" that session in the special preamble *before* sending the ciphertext containing the verifier's secret key. Otherwise, the adversary "loses" that session. Then, by using the properties of our probability distribution, we are able to show that the adversary's probability of losing a session is less than $1/n$ times the probability of winning. As a consequence, by careful choice of parameters, we are able to show

[4]Indeed when the simulator rewinds the adversarial verifier, there is a different view and therefore the adversary will ask to play with new identities, making useless the work done with the old ones, as it happens in the standard model.

that the probability of the adversary winning more than a given polynomially bounded number of sessions *without losing any sessions* w.r.t. any given verifier is negligible. Once we obtain this fixed bound, we are then able to rely on techniques from the bounded-concurrency model [Bar01] to handle the bounded number of non-black-box simulations. For the sake of brevity, the above discussion is somewhat oversimplified. We refer the reader to the later sections for more details.

Extension to concurrent-secure computation - an impossibility. Once we have a cZK protocol (as discussed above) in the Bounded Player model, it may seem that it should be possible to obtain concurrent-secure computation as well by using techniques from [Pas04]. Unfortunately, this turns out not to be the case, as we discuss below.

The key technical problem that arises in the setting of secure computation w.r.t. unbounded concurrency is the following. We cannot a priori bound the total number of "output delivery messages" (across all sessions) to the adversary; further, the session outputs cannot be "predicted" by the simulator before knowing the adversary's input. As such, known non-black-box simulation techniques cannot handle these unbounded number of messages and they inherently fail.[5] We remark that the same technical issue, in fact, arises in the standard model as well.

While the above argument only explains why known techniques fail, we can also obtain a formal impossibility result. Indeed, it is not difficult to see that the impossibility result of Lindell [Lin04] also holds for the Bounded Player model. (See Appendix C for details.)

2 Preliminaries and Definitions

2.1 Bounded Player Model

In this paper, we consider a new model of concurrent security, namely, the *bounded player model*, where we assume that there is an a-priori (polynomial) upper bound on the total number of player that will ever be present in the system. Specifically, let n denote the security parameter. Then, we will consider an upper bound $N = \text{poly}(n)$ on the total number of players that can engage in concurrent executions of a protocol at any time. We assume that each player P_i ($i \in N$) has an associated unique identity id_i, and that there is an established mechanism to enforce that party P_i uses the same identity id_i in each protocol execution that it participates in. We stress that such identities, do not have to be established in advance. New players can join the system with their own (new) identities, as long as the number of players does not exceed N.

We note that this requirement is somewhat similar in spirit to the *bounded-concurrency model* [Bar01, Lin03a, PR03, Pas04], where it is assumed that the adversary cannot start more than an a-priori fixed number of concurrent executions of a protocol. We stress, however, that in our model, there is *no* a-priori bound on the total number of protocol sessions that may be executed concurrently. In this respect, one can view the Bounded Player model as a strengthening of the bounded-concurrency model.[6] Indeed, one can argue that while the number of devices and people on this planet can be realistically estimated and bounded, the number of concurrent protocol executions on these devices can not.

Implementing the Bounded Player model. We formalize the Bounded Player model by means of a functionality F_{bp}^N that registers the identities of the player in the system. Specifically, a player P_i that wishes to participate in protocol executions can, at any time, register an identity id_i with the functionality F_{bp}^N. The registration functionality does not perform any checks on the identities that are registered, except that each party P_i can register at most one identity id_i, and that the total number of identity registrations are bounded by N. In other words, F_{bp}^N refuses

[5] We note that this problem does not occur in the case of zero knowledge because the adversary does not have any input, and the session outputs are fixed to be 1.

[6] Note that an upper bound on the total number of concurrent executions implies an upper bound on the total number of player as well.

to register any new identities once N number of identities have already been registered. The functionality F_{bp}^N is formally defined in Figure 1.

Functionality F_{bp}^N

F_{bp}^N initializes a variable *count* to 0 and proceeds as follows.

- **Register commands:** Upon receiving a message $(\textbf{register}, \mathit{sid}, \mathsf{id}_i)$ from some party P_i, the functionality checks that no pair (P_i, id_i') is already recorded and that $\mathit{count} < N$. If this is the case, it records the pair (P_i, id_i) and sets $\mathit{count} = \mathit{count} + 1$. Other wise, it ignores the received message.
- **Retrieve commands:** Upon receiving a message $(\mathsf{retrieve}, \mathit{sid}, P_i)$ from some party P_j or the adversary A, the functionality checks if some pair (P_i, id_i) is recorded. If this the case, it sends $(\mathit{sid}, P_i, \mathsf{id}_i)$ to P_j (or A). Otherwise, it returns $(\mathit{sid}, P_i, \perp)$.

Figure 1: The Bounded Player Functionality F_{bp}^N.

In our constructions we will only require that the identities correspond to values in the range of a one-way function. We note that in this particular case, the functionality F_{bp}^N bears much resemblance to the *bulletin-board certificate authority* functionality [KL11], which suffices for obtaining authenticated channels [Can04]. We finally remark that our model is also closely related to the *Bare Public-Key model*, introduced by Canetti et al. [CGGM00]. However, we stress that unlike the Bare Public-Key model, we do not assume any synchronization barrier between the registration phase and the protocol computation phase. In particular, we allow parties to register their identities even after the computation begins.

2.2 Concurrent Zero Knowledge in Bounded Player Model

In this section, we formally define concurrent zero knowledge in the Bounded Player model. Our definition, given below, is an adaptation of the one of [PRS02] to the Bounded Player mode, by also considering non-black-box simulation. Some of the text below is taken verbatim from [PRS02].

Let PPT denote probabilistic-polynomial time. Let $\langle P, V\rangle$ be an interactive argument for a language L. Consider a concurrent adversarial verifier V^* that, given input $x \in L$, interacts with an unbounded number of independent copies of P (all on the same common input x and moreover equipped with a proper witness w), without any restriction over the scheduling of the messages in the different interactions with P. In particular, V^* has control over the scheduling of the messages in these interactions. Further, we say that V^* is an N-bounded concurrent adversary if it assumes at most N verifier identities during its (unbounded) interactions with P.[7]

The transcript of a concurrent interaction consists of the common input x, followed by the sequence of prover and verifier messages exchanged during the interaction. We denote by $\mathsf{view}_{V^*}^P(x, z, N)$ the random variable describing the content of the random tape of the N-bounded concurrent adversary V^* with auxiliary input z and the transcript of the concurrent interaction between P and V^* on common input x.

Definition 1 (cZK in Bounded Player model) *Let $\langle P, V\rangle$ be an interactive argument system for a language L. We say that $\langle P, V\rangle$ is* concurrent zero-knowledge *in the Bounded Player model if for every N-bounded concurrent non-uniform* PPT *adversary V^*, there exists a* PPT *algorithm $\mathcal{S}$, such that the following ensembles are computationally indistinguishable, $\{\mathsf{view}_{V^*}^P(x, z, N)\}_{x\in L, z\in\{0,1\}^*, N\in \mathit{poly}(n)}$ and $\{\mathcal{S}(x, z, N)\}_{x\in L, z\in\{0,1\}^*, N\in \mathit{poly}(n)}$.*

[7] Thus, V^* can open multiple sessions with P for every unique verifier identity.

2.3 Building Blocks

In this section, we discuss the main building blocks that we will use in our cZK construction.

Perfectly Hiding Commitment Scheme. In our constructions, we will make use of a perfectly hiding string commitment scheme, denoted $\mathbf{Com}$. For simplicity of exposition, we will make the simplifying assumption that $\mathbf{Com}$ is a non-interactive perfectly hiding commitment scheme (even though such a scheme cannot exist). In reality, $\mathbf{Com}$ would be taken to be a 2-round commitment scheme, which can be based on collections of claw-free permutations [GK96]. Unless stated otherwise, we will simply use the notation $\mathbf{Com}(x)$ to denote a commitment to a string x, and assume that the randomness (used to create the commitment) is implicit.

Perfect Witness Indistinguishable Argument of Knowledge. We will also make use of a perfect witness-indistinguishable argument of knowledge system for all of $\mathcal{NP}$ in our construction. Such a scheme can be constructed, for example, by parallel repetition of the 3-round Blum's protocol for Graph Hamiltonicity [Blu87] instantiated with a perfectly hiding commitment scheme. We will denote such an argument system by $\langle P_{\mathsf{pWI}}, V_{\mathsf{pWI}} \rangle$.

Perfect Witness Indistinguishable Universal Argument. In our construction, we will use a perfect witness-indistinguishable universal argument system, denoted $\langle P_{\mathsf{pUA}}, V_{\mathsf{pUA}} \rangle$. Such an argument system can be constructed generically from a (computational) witness-indistinguishable universal argument pUA by using techniques of [PR05b, PR05a]. Specifically, in protocol $\langle P_{\mathsf{pUA}}, V_{\mathsf{pUA}} \rangle$, the prover P and verifier V first engage in an execution of pUA, where instead of sending its messages in the clear, P commits to each message using a perfectly hiding commitment scheme. Finally, P and V engage in an execution of a perfect zero knowledge argument of knowledge where P proves that the "decommitted" transcript of pUA is "accepting". The resulting protocol is still a "weak" argument of knowledge.

Perfect (Bounded-Concurrent) Zero-Knowledge. Our cZK argument crucially uses as a building block, a variant of the bounded cZK argument of Barak [Bar01]. Similarly to [PR05a], we modify the protocol appropriately such that it is *perfect* bounded cZK. Specifically, instead of a statistically binding commitment scheme, we will use a perfectly hiding commitment scheme. Instead of a computationally witness-indistinguishable universal argument (UARG), we will use a perfect witness indistinguishable UARG, denoted $\langle P_{\mathsf{pUA}}, V_{\mathsf{pUA}} \rangle$. Further, the length parameter $\ell(N)$ used in the modified protocol is a function of N, where N is the bound on the number of verifiers in the system. Protocol $\langle P_{\mathsf{pB}}, V_{\mathsf{pB}} \rangle_N$ is described in Figure 3 and can be based on claw-free permutations.

Resettable Witness Indistinguishable Proof System. We will further use a *resettable* witness-indistinguishable proof system [CGGM00] for all of $\mathcal{NP}$. Informally speaking, a proof system is resettable witness indistinguishable if it remains witness indistinguishable even against an adversarial verifier who can *reset* the prover and receive multiple proofs such that the prover uses the *same* random tape in each of the interactions. While the focus of this work is not on achieving security against reset attacks, such a proof system turns out to be useful when arguing concurrent soundness of our protocol (where our proof relies on a rewinding based argument). We will denote such a proof system by $\langle P_{\mathsf{rWI}}, V_{\mathsf{rWI}} \rangle$. It follows from [CGGM00] that such a proof system can be based on perfectly hiding commitments.

Dense Cryptosystems [SP92]. We will use a semantically secure public-key encryption scheme, denoted as $(\mathbf{Gen}, \mathbf{Enc}, \mathbf{Dec})$ that supports *oblivious* key generation (i.e., it should be possible to sample a public key without knowing the corresponding secret key). More precisely, there exists a deterministic algorithm $\mathbf{OGen}$ that takes as input the security parameter 1^n and a sufficiently long random string σ and outputs a public key

$pk \leftarrow \mathbf{OGen}(1^n, \sigma)$, where pk is perfectly indistinguishable from a public key chosen by the normal key generation algorithm $\mathbf{Gen}$. For simplicity of exposition, we will assume that the $\mathbf{OGen}$ algorithm simply outputs the input randomness σ as the public key. Such schemes can be based on a variety of number-theoretic assumptions such as DDH [SP92].

3 Concurrent Zero Knowledge in Bounded Player Model

In this section, we describe our concurrent zero-knowledge protocol in the bounded player model.

Relation R_{sim}. We first recall a slight variant of Barak's [Bar01] $\mathbf{NTIME}(T(n))$ relation R_{sim}, as used previously in [PR05a]. Let $T : \mathbb{N} \to \mathbb{N}$ be a "nice" function that satisfies $T(n) = n^{\omega(1)}$. Let $\{\mathcal{H}_n\}_n$ be a family of collision-resistant hash functions where a function $h \in \mathcal{H}_n$ maps $\{0,1\}^*$ to $\{0,1\}^n$, and let $\mathbf{Com}$ be a perfectly hiding commitment scheme for strings of length n, where for any $\alpha \in \{0,1\}^n$, the length of $\mathbf{Com}(\alpha)$ is upper bounded by $2n$. The relation R_{sim} is described in Figure 2.

Instance: A triplet $\langle h, c, r\rangle \in \mathcal{H}_n \times \{0,1\}^n \times \{0,1\}^{\mathrm{poly}(n)}$.
Witness: A program $\Pi \in \{0,1\}^*$, a string $y \in \{0,1\}^*$ and a string $s \in \{0,1\}^{\mathrm{poly}(n)}$.
Relation: $R_{\mathsf{sim}}(\langle h, c, r\rangle, \langle \Pi, y, s\rangle) = 1$ if and only if:

1. $|y| \leq |r| - n$.
2. $c = \mathbf{Com}(h(\Pi); s)$.
3. $\Pi(y) = r$ within $T(n)$ steps.

Figure 2: R_{sim} - A variant of Barak's relation [PR05a]

Remark 1 *The relation presented in Figure 2 is slightly oversimplified and will make Barak's protocol work only when $\{\mathcal{H}_n\}_n$ is collision-resistant against "slightly" super-polynomial sized circuits. For simplicity of exposition, in this manuscript, we will work with this assumption. We stress, however, that as discussed in prior works* [BG02, Pas04, PR05b, PR05a]*, this assumption can be relaxed by using a "good" error-correcting code* ECC *(with constant distance and polynomial-time encoding and decoding procedures), and replacing the condition* $c = \mathbf{Com}(h(\Pi); s)$ *with* $c = \mathbf{Com}(\mathsf{ECC}(h(\Pi)); s)$.

3.1 Our Protocol

We are now ready to present our concurrent zero knowledge protocol, denoted $\langle P, V\rangle$. Let P and V denote the prover and verifier respectively. Let N denote the bound on the number of verifiers present in the system. Let f_{owf} denote a one-way function, and $(\mathbf{Gen}, \mathbf{Enc}, \mathbf{Dec})$ denote a dense public key encryption scheme. Let $\langle P_{\mathsf{pB}}, V_{\mathsf{pB}}\rangle_N$ denote the perfect zero-knowledge argument system as described above. Further, let $\langle P_{\mathsf{pWI}}, V_{\mathsf{pWI}}\rangle$ denote a perfect witness indistinguishable argument of knowledge, and let $\langle P_{\mathsf{rWI}}, V_{\mathsf{rWI}}\rangle$ denote a resettable witness indistinguishable proof system.

The protocol $\langle P, V\rangle$ is described in Figure 4. For our purposes, we set the length parameter $\ell(N) = n^3 \cdot N \cdot P(n)$, where $P(n)$ is a polynomial upper bound on the total length of the prover messages in the protocol plus the length of the secret key of the verifier.

The completeness property of $\langle P, V\rangle$ follows immediately from the construction. Due to lack of space, we defer the proof of soundness to Appendix A. We remark that, in fact, we prove *concurrent soundness* of $\langle P, V\rangle$,

Parameters: Security parameter n, length parameter $\ell(N)$.
Common Input: $x \in \{0,1\}^{\text{poly}(n)}$.
Private Input to P: A witness w such that $R_L(x,w) = 1$.

Stage 1 (Preamble Phase):

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

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

$V \to P$: Send $r \stackrel{R}{\leftarrow} \{0,1\}^{\ell(N)}$.

Stage 2 (Proof Phase):

$P \leftrightarrow V$: A perfect WI UARG $\langle P_{\mathsf{pUA}}, V_{\mathsf{pUA}} \rangle$ proving the OR of the following statements:

1. $\exists w \in \{0,1\}^{\text{poly}(|x|)}$ s.t. $R_L(x,w) = 1$.
2. $\exists \langle \Pi, y, s \rangle$ s.t. $R_{\mathsf{sim}}(\langle h, c, r \rangle, \langle \Pi, y, s \rangle) = 1$.

Figure 3: Protocol $\langle P_{\mathsf{pB}}, V_{\mathsf{pB}} \rangle_N$

i.e., we show that a computationally-bounded adversarial prover who engages in multiple concurrent executions of $\langle P, V \rangle$ (where the scheduling across the sessions is controlled by the adversary) cannot prove a false statement in any of the executions, except with negligible probability. We note that similarly to the Bare Public-Key model [CGGM00], "stand-alone" soundness does not imply concurrent soundness in our model. Informally speaking, this is because the standard approach of reducing concurrent soundness to stand-alone soundness by "internally" emulating all but one verifier does not work since the verifier's secret keys are private. Indeed, Micali and Reyzin [MR01] gave concrete counter-examples to show that stand-alone soundness does not imply concurrent soundness in the BPK model. We note that their results immediately extend to our model.

We now turn to prove that protocol $\langle P, V \rangle$ is concurrent zero-knowledge in the Bounded Player model.

3.2 Proof of Concurrent Zero Knowledge

In this section, we prove that the protocol $\langle P, V \rangle$ described in Section 3 is concurrent zero-knowledge in the bounded player model. Towards this end, we will construct a non-black-box (polynomial-time) simulator and then prove that the concurrent adversary's view output by the simulator is indistinguishable from the real view. We start by giving an overview of the proof and then proceed to give details.

Overview. Barak's argument system [Bar01] is zero-knowledge in the bounded-concurrency model where the concurrent adversary is allowed to open at most $m = m(n)$ concurrent sessions for a fixed polynomial m. Loosely speaking, Barak's simulator takes advantage of the fact that the total number of prover messages across all sessions is bounded; thus it can commit to a machine that takes only a bounded-length input y that is smaller than the challenge string r, and outputs the next message of the verifier, in any session. In our model, there is no bound on the total number of sessions, thus we cannot directly employ the same strategy. Towards this, an important observation in our setting is that once we are able to "solve" a verifier identity (i.e., learn secret key of a verifier), then the simulator does not need to do Barak-style simulation anymore for that identity. But what of the number of Barak-style simulations that the simulator needs to perform *before* it can learn any secret key? Indeed, if this number were unbounded, then we would run into the same problems that one encounters when trying to construct non-black-box cZK in the standard model. Fortunately, we are able to show that the simulator only needs to perform a bounded number of Barak-style simulations before it can learn a secret key. Thus, we obtain the

Parameters: Security parameter n, $N = N(n)$, $t = \omega(1)$.
Common Input: $x \in \{0,1\}^{\text{poly}(n)}$.
Private Input to P: A witness w s.t. $R_L(x, w) = 1$.
Private Input to V: A public key $pk = (y_0, y_1)$ and secret key $sk = (b, x_b)$ s.t. $b \xleftarrow{R} \{0,1\}$, $y_b = f_{\mathsf{owf}}(x_b)$.

Stage 1 (Preamble Phase): Repeat the following steps t times.

$V \to P$: Send $pk = (y_0, y_1)$.

$P \to V$: Choose $\sigma_p \xleftarrow{R} \{0,1\}^n$ and send $c_p = \mathbf{Com}(\sigma_p)$.

$V \to P$: Send $\sigma_v \xleftarrow{R} \{0,1\}^n$.

$P \to V$: Send σ_p. Let $\sigma = \sigma_p \oplus \sigma_v$.

$P \leftrightarrow V$: An execution of $\langle P_{\mathsf{pB}}, V_{\mathsf{pB}} \rangle_N$ to prove the following statement: $\exists s$ s.t. $c = \mathbf{Com}(\sigma_p; s)$.

$V \to P$: Send $e_1 = \mathbf{Enc}_\sigma(x_b)$, $e_2 = \mathbf{Enc}_\sigma(x_b)$.

$V \leftrightarrow P$: An execution of resettable WI $\langle P_{\mathsf{rWI}}, V_{\mathsf{rWI}} \rangle$ to prove the following statement: $\exists \langle i, b, x_b, s \rangle$ s.t. $e_i = \mathbf{Enc}_\sigma(x_b; s)$ and $y_b = f_{\mathsf{owf}}(x_b)$.

Stage 2 (Proof Phase):

$P \leftrightarrow V$: An execution of perfect WIAOK $\langle P_{\mathsf{pWI}}, V_{\mathsf{pWI}} \rangle$ to prove the OR of the following statements:

1. $\exists w \in \{0,1\}^{\text{poly}(|x|)}$ s.t. $R_L(x, w) = 1$.
2. $\exists \langle b, x_b \rangle$ s.t. $y_b = f_{\mathsf{owf}}(x_b)$.

Figure 4: Protocol $\langle P, V \rangle$

following strategy: the simulator commits to an "augmented machine" that is able to simulate almost all of the simulator messages by itself; the remaining simulator messages are given as input to this machine. As discussed above, we are able to bound the total number of these messages, and thus by setting the challenge string r to be more than this bound, we ensure that the simulation is correct. More in details, the input passed by the simulator to the machine consists of transcripts of concurrent sessions where again the simulator had to use Barak-style simulation[8] and the (discovered) secret keys of the verifiers to be used by the machine to carry on the simulation by itself (without performing Barak-style simulation).

The Simulator. We now proceed to describe our simulator. The simulator SIM consists of two main parts, namely, SIM_{easy} and SIM_{extract}. Loosely speaking, SIM_{extract} is only used to cheat in a "special" preamble block of a session in order to learn the secret key of a verifier, while SIM_{easy} is used for the remainder of the simulation, which includes following honest prover strategy in preamble blocks and simulating the proof phase of each session using the verifier's secret key as the trapdoor witness. Specifically, SIM_{extract} cheats in the $\langle P_{\mathsf{pB}}, V_{\mathsf{pB}} \rangle_N$ protocol by committing to an augmented verifier machine Π that contains the code of SIM_{easy}, allowing it to simulate all of the simulator messages except those generated by SIM_{extract} (in different sessions). As we show below, these messages can be bounded to a fixed value. We now describe the simulator in more detail.

SETUP AND INPUTS. Our simulator SIM interacts with an adversary $V^* = (V_1^*, \ldots, V_N^*)$ who controls verifiers $V_1, \ldots, V_N$. V^* interacts with SIM in m sessions, and controls the scheduling of the messages. We give SIM non-black-box access to V^*. Throughout the interaction, SIM keeps track of a tuple $\vec{\beta} = (\beta_1, \ldots, \beta_N)$ represent-

[8]The reason we pass this transcript as input is that in this way we can avoid the blow up of the running time of the simulator when nested Barak-style simulations are performed.

ing the secret keys SIM has learned so far. At any point during the interaction either $\beta_i = \text{sk}_i$ (more precisely, β_i is one of the coordinates of sk_i) or β_i is the symbol $\perp$. Initially, SIM sets each β_i to $\perp$, but it updates $\vec{\beta}$ throughout the interaction as it extracts secret keys. Additionally, SIM keeps a counter vector $\vec{a} = (a_1, \ldots, a_N)$, incrementing a_i each time it executes a preamble block using SIM_{extract} against V^*. We have SIM halt and output FAIL if any a_i ever surpasses n^3. Our technical lemma shows that this happens with negligible probability. Finally, we have SIM keep track of a set of tuples

$$\Psi = \{((i,j,k)_\gamma; \phi_\gamma) : \gamma = 1, \ldots, n^3 N\}$$

where each $(i,j,k)_\gamma \in [N] \times [m] \times [t]$ and ϕ_γ is a string. The tuples $(i,j,k)_\gamma$ represent the preamble blocks played by SIM_{extract}; specifically, (i,j,k) corresponds to the $k-$th block of the $j-$th session against V^*. The string ϕ_γ is the collection of simulator messages sent in block $(i,j,k)_\gamma$. This set of tuples Ψ (along with β) will be the extra input given to the augmented machine. As we show below, the total size of Ψ will be a priori bounded by a polynomial in n.

Consider the interaction of SIM with some V^* impersonating V_i. Each time V^* opens a session on behalf of V_i, SIM chooses a random $k \in \{1, \ldots, t\}$ according to a distribution D_t which we define later. This will be the only preamble block of the session played by SIM_{extract} provided that $\beta_i = \perp$ when the block begins. If SIM has already learned the secret key sk_i, it does not need to call SIM_{extract}. We now describe the parts of SIM beginning with SIM_{easy}.

THE SUB-SIMULATOR SIM_{EASY}. Recall that SIM_{easy} is run on input β and Ψ. When SIM_{easy} is called to execute the next message of a preamble block, it checks if the message is already in Ψ. If this is the case, SIM_{easy} just plays the message. Otherwise, SIM_{easy} plays fairly, choosing a random σ_p and sending $c_p = \text{Com}(\sigma_p; s)$ for some s. Upon receiving σ_v, it returns σ_p and completes $\langle P_{\text{pB}}, V_{\text{pB}} \rangle$ using s as its witness. Its receipt of encryptions (e_1, e_2) and acceptance of $\langle P_{\text{rWI}}, V_{\text{rWI}} \rangle$ ends the preamble block. If SIM_{easy} does not accept V^*'s execution of $\langle P_{\text{rWI}}, V_{\text{rWI}} \rangle$ it aborts the interaction, as would an honest prover.

When SIM_{easy} is called to execute $\langle P_{\text{pWI}}, V_{\text{pWI}} \rangle$ then it checks if the secret key of the verifier is in β. If yes, SIM_{easy} completes $\langle P_{\text{pWI}}, V_{\text{pWI}} \rangle$ using sk_i as its witness. Otherwise, $\beta_i = \perp$ and SIM_{easy} halts outputting FAIL. Our technical lemma shows that the latter does not happen, except with negligible probability.

THE SUB-SIMULATOR SIM_{EXTRACT}. When SIM_{extract} is called to execute preamble block k of session j with verifier V_i^*, it receives Ψ, β and a as input. We assume $\beta_i = \perp$ since otherwise, SIM would not have called SIM_{extract}. Immediately upon being called, SIM_{extract} increments a_i and adds the tuple $((i,j,k); \phi)$ to Ψ. Initially, ϕ is the empty string, but each time SIM_{extract} sends a message, it appends the message to ϕ. By the end of the block, ϕ is a complete transcript of the simulator messages in preamble block (i,j,k).

The preamble block begins normally, with SIM_{extract} choosing a random string and sending c_p, a commitment to it. Upon receiving σ_v, however, SIM_{extract} runs **Gen** obtaining key pair (σ, τ) for the encryption scheme and returns $\sigma_p = \sigma \oplus \sigma_v$. Next, SIM_{extract} enters $\langle P_{\text{pB}}, V_{\text{pB}} \rangle$ which it completes using the already extracted secret key. Formally, when V^* sends h, beginning $\langle P_{\text{pB}}, V_{\text{pB}} \rangle$, SIM_{extract} chooses a random s and sends $\text{Com}(h(\Pi); s)$, where Π is the next message function of V^*, augmented with the ability to compute all the intermediate messages sent by SIM_{easy}. The machine Π takes input $y = (\Psi, \beta)$ and outputs the next verifier message in an interaction between V^* and a machine M who plays exactly like SIM_{easy} with the following exception. For each tuple $((i,j,k); \phi) \in \Psi$, M reads its messages of block (i,j,k) from the string y. In order to simulate SIM_{easy} in the subprotocols $\langle P_{\text{pWI}}, V_{\text{pWI}} \rangle$, M also uses the tuple $\vec{\beta} = (\beta_1, \ldots, \beta_N)$ received as input, where each β_i is the secret key of the i'-th verifier (if available), and $\perp$ otherwise.

After committing to Π, and receiving r, SIM_{extract} completes $\langle P_{\text{pUA}}, V_{\text{pUA}} \rangle$ using witness $(\Pi, \Psi \| \beta, s)$ where Ψ and β might have been updated by other executions of SIM_{extract} occurring between the time SIM_{extract} sent $\text{Com}(h(\Pi); s)$ and received r. Our counter ensures that $|\Psi|$ is a priori bounded, while $|\beta|$ is bounded by definition. By construction, Π correctly predicts V^*'s message r, and so $(\Pi, \Psi \| \beta, s)$ is a valid witness for $\langle P_{\text{sUA}}, V_{\text{sUA}} \rangle$. Finally, SIM_{extract} receives encryptions e_1, e_2 and the proof of correctness in $\langle P_{\mathsf{rWI}}, V_{\mathsf{rWI}} \rangle$. It now decrypts the

ciphertexts using τ thereby learning secret key sk_i of V_i^*. If the decrypted value is a valid secret key sk_i, then it updates β by setting $\beta_i = \text{sk}_i$. Otherwise, it outputs the abort symbol $\perp$ and stops. (It is easy to see that since the proof system $\langle P_{\mathsf{rWI}}, V_{\mathsf{rWI}} \rangle$ is sound, the probability of simulator outputting $\perp$ at this step is negligible.)

Analysis. There are two situations in which SIM outputs fail: if some counter a_i exceeds n^3, or if SIM_{easy} enters an execution $\langle P_{\text{pWI}}, V_{\text{pWI}} \rangle$ without knowledge of sk. Note that the latter will not happen, as to enter an execution of $\langle P_{\text{pWI}}, V_{\text{pWI}} \rangle$, all preamble blocks, in particular the one played by SIM_{extract}, must be complete, ensuring that SIM_{extract} will have learned sk. In our main technical lemma, we show that no counter will surpass n^3 by proving that after SIM has run SIM_{extract} n^3 times against each V_i controlled by V^* it has, with overwhelming probability, learned sk. Before stating the lemma, we introduce some terminology.

Now, focusing on a given verifier, we say that V^* has *stopped* session j in block k if the $k-$th preamble block of session j has begun, but the $(k+1)-$th has not. We say that V^* is playing *strategy* $\vec{k'} = (k'_1, \ldots, k'_m)$ if session j is stopped in block k'_j for all $j = 1, \ldots, m$. As the interaction takes polynomial time, V^* only gets to play polynomially many strategies over the course of the interaction. Let $k_j \in \{1, \ldots, t\}$ be the random number chosen by SIM at the beginning of session j as per distribution D_t. This gives us a tuple $\vec{k} = (k_1, \ldots, k_m)$ where the k_j are chosen independently according to the distribution D_t (defined below). At any time during the interaction, we say that V^* has *won* (resp. *lost*, *tied*) session j if $k'_j = k_j$ (resp. $k'_j > k_j$, $k'_j < k_j$). A win for V^* corresponds to SIM having run SIM_{extract}, but not yet having learned sk. As SIM only gets to call SIM_{extract} n^3 times, a win for V^* means that SIM has used up one of its budget of n^3 without any payoff. A loss for V^* corresponds to SIM running SIM_{extract} and learning sk, thereby allowing SIM to call SIM_{easy} in all remaining sessions. A tie means that SIM has not yet called SIM_{extract} in the session, and therefore has not used any of its budget, but has not learned sk.

Notice that these wins and ties are "temporary" events. Indeed, by the end of each session, V^* will have lost, as he will have completed the preamble block run by SIM_{extract}. However, we choose to use this terminology to better convey the key intuition of our analysis: for SIM to output FAIL, it must be that at some point during the interaction, for some identity, V^* has won at least n^3 sessions and has not lost any. We will therefore focus precisely on proving that the probability that a PPT adversary V^* runs in the experiment m sessions so that the counter for one identity reaches the value n^3 is negligible.

For a verifier strategy $\vec{k'}$ and a polynomial m, let $P_{(\vec{k'},m)}(W, L)$ be the probability that in an $m-$session interaction between V^* and SIM that V^* wins for some identity exactly W sessions and loses exactly L, given that V^* plays strategy $\vec{k'}$. The probability is over SIM's choice of $\vec{k}$ with $k_j \in \{1, \ldots, t\}$ chosen independently according to D_t (defined below) for all $j = 1, \ldots, m$.

THE DISTRIBUTION D_t AND THE MAIN TECHNICAL LEMMA. Define D_t to be the distribution on $\{1, \ldots, t\}$ such that

$$p_{k'} = \text{Prob}_{k \in D_t}(k = k') = \varepsilon n^{k'},$$

where ε is such that $\sum p_{k'} = 1$. Note that ε is negligible in n.

Lemma 1 (Main Technical Lemma) *Let $\vec{k'}$ be a verifier strategy and $m = m(n)$ a polynomial. Then we have*

$$P_{(\vec{k'},m)}(n^3, 0)$$

is negligible in n.

The above proves that any verifier strategy has a negligible chance of having n^3 wins and no losses. As V^* plays polynomially many (i.e., N) strategies throughout the course of the interaction, the union bound proves that V^* has a negligible chance of ever achieving n^3 wins and 0 losses. From this it follows that, with overwhelming probability, V^* will never have at least n^3 wins and no losses, which implies that SIM outputs FAIL with negligible

probability as desired. The main idea of the proof is similar to the random tape switching technique of [PRS02] and [MP07].

Proof. We fix a verifier strategy $\vec{k'}$ and a polynomial m and write $P(W,L)$ instead of $P_{(\vec{k'},m)}(W,L)$. Let $p_{k'}$ (resp. $q_{k'}$) be the probability that V^* wins (resp. loses) a session given that he stops the session in block k'. We chose the distribution D_t carefully to have the following two properties. First, since $p_1 = \varepsilon n$ is negligible, we may assume that V^* never stops in the first block of a session. And secondly, for $k' \geq 2$ we have,

$$q_{k'} = \sum_{i=1}^{k'-1} p_{k'} = \varepsilon \frac{n^{k'}-1}{n-1} \geq \frac{\varepsilon n^{k'}}{2n} = \frac{p_{k'}}{2n}.$$

It follows that no matter which what block V^* stops a session in, it will hold that the probability he wins in that session is less then $2n$ times the probability that he looses that session. We will use this upper bound on the probability of V^* winning a single session to show that $P(n^3,0)$ is negligible.

Let A be the event, $(W,L) = (n^3,0)$, B be the event $W+L = n^3$ and $\neg B$ the event $W+L \neq n^3$. Since, $A \subset B$, and since $P(A|\neg B) = 0$, we have that

$$P(n^3,0) = P(A) = P(A|B)P(B) + P(A|\neg B)P(\neg B) = P(A|B)P(B) \leq P(A|B),$$

and so it suffices to prove that $P(A|B)$ is negligible. We continue the proof for the case $W+L = n^3$ (and thus $m \geq n^3$).

If $W+L = n^3$ then V^* ties all but n^3 of the sessions. Let $\mathcal{C} = \{C \subset [m] : |C| = n^3\}$. Then $\mathcal{C}$ is the set of possible positions for the sessions which are not ties. We are looking to bound $P((W,L) = (n^3,0)|W+L = n^3)$ and so we condition on the $C \in \mathcal{C}$. Once a fixed C is chosen, the position of each session which is not a tie is determined. Each such session must either be a win or a loss for V^*. Let p be the probability that some such session is a win. Since we proved already that the probability that V^* wins in a given session is less then $2n$ times the probability that V^* looses in that session, we have that $p \leq 2n(1-p)$. Solving gives $p \leq (1 - \frac{1}{2n+1})$. It follows that for any $C \in \mathcal{C}$, the probability that all sessions in C are wins is

$$\left(1 - \frac{1}{2n+1}\right)^{n^3} \leq \left[\left(1 - \frac{1}{2n+1}\right)^{2n+1}\right]^n \leq e^{-n}.$$

From the viewpoint of random tape switching, we have shown that for every random tape causing every session of C to be a win, there are exponentially many which cause a different outcome.

We therefore have

$$\begin{aligned} P(n^3,0) &\leq P((W,L) = (n^3,0)|W+L = n^3) \\ &= \sum_{C \in \mathcal{C}} P((W,L) = (n^3,0)|C)P(C) \\ &\leq e^{-n} \sum_{C \in \mathcal{C}} P(C) = e^{-n}, \end{aligned}$$

as desired.

Bounding the length parameter $\ell(N)$. From the above lemma, it follows easily that the total length of the auxiliary input y to the machine Π committed by SIM_{extract} (at any time) is bounded by $n^3 \cdot N \cdot P(n)$, where $P(n)$ is a polynomial upper bound on the total length of prover messages in one protocol session plus the length of a secret. Thus, when $\ell(N) \geq n^3 \cdot N \cdot P(n)$, we have that $|y| \leq |r| - n$, as required.

We now show through a series of hybrid experiments that the simulator's output is perfectly indistinguishable from the output of the adversary when interacting with honest provers.

Our hybrid experiments will be H_i for $i = 0, \ldots, 6$. We write $H_i \approx H_j$ if V^* cannot distinguish between its interaction with H_i and H_j.

H_0. The experiment H_0 is the fair prover. In each preamble block it sends $c_p = \text{Com}(\sigma_p; s)$ for random σ_p, receives σ_v and returns σ_p. It completes $\langle P_{\text{pB}}, V_{\text{pB}} \rangle$ using s as its witness. It receives the encryptions and V^*'s proof of $\langle P_{\text{rWI}}, V_{\text{rWI}} \rangle$ completing the preamble block. We provide H_0 with a witness that $x \in L$ which it uses to complete $\langle P_{\text{pWI}}, V_{\text{pWI}} \rangle$ at the end of each session.

H_1. The experiment H_1 plays similarly to H_0. However, the execution of H_1 takes exponential time. It begins by computing the verifier secret keys by inverting the one-way functions in exponential time. It will use knowledge of secret key in the protocols $\langle P_{\text{pWI}}, V_{\text{pWI}} \rangle$. The perfect witness indistinguishability of $\langle P_{\text{pWI}}, V_{\text{pWI}} \rangle$ ensures that $H_1 \approx H_0$.

H_2. The experiment H_3 plays similarly to H_1. The only difference is that in all the preamble blocks where the simulator would have tried to extract, in its first message of $\langle P_{\text{pB}}, V_{\text{pB}} \rangle$, we commit to the augmented machine Π. As before, the augmented machine Π predicts the next message of V^* and is able to simulate all fair messages of H_2. It therefore must take as input only the prover messages of the preamble blocks where H_2 does not play fairly. We have H_2 keep track of a set of tuples $\Psi = \{((i,j,k)_\gamma; y_\gamma) : \gamma = 1, \ldots, n^3 N\}$, where the tuple $((i,j,k); y)$ means that in the $k-$th preamble block of the $j-$th session against V_i^*, H_2 sent messages y. Π also receives a tuple $\vec{\beta} = (\beta_1, \ldots, \beta_N)$ where β_i could correspond to a secret key of the i-th verifier and to $\perp$ otherwise. The simulated prover will use them in $\langle P_{\text{pWI}}, V_{\text{pWI}} \rangle$ different than $\perp$. As the only difference between the output of H_2 and H_1 is that sometimes H_2 commits to a different value than H_1 does, the perfect hiding of Com ensures that $H_2 \approx H_1$.

H_3. The experiment H_3 plays similarly to H_2. The only difference is that in $\langle P_{\text{pB}}, V_{\text{pB}} \rangle$, instead of proving the statement honestly, the simulator of $\langle P_{\text{pB}}, V_{\text{pB}} \rangle$ is run by using in the underlying $\langle P_{\text{pUA}}, V_{\text{pUA}} \rangle$ values (Π, Ψ, β, s) where Π is the augmented machine committed in the chosen execution of $\langle P_{\text{pB}}, V_{\text{pB}} \rangle$, Ψ is the record of all messages sent in the chosen preamble blocks where it deviates from fair play, β is the vector of known keys, and s is the witness to be used in the non-chosen execution of $\langle P_{\text{pB}}, V_{\text{pB}} \rangle$. The reason that $|\Psi|$ can be a priori bounded by a polynomial in n is that by the main technical lemma, we have that Π needs messages for at most $n^3 N$ chosen preamble blocks where H_3 deviates from fair play. The perfect witness indistinguishability of $\langle P_{\text{pUA}}, V_{\text{pUA}} \rangle$ ensures that $H_3 \approx H_2$.

H_4. The experiment H_4 plays similarly to H_3. However, there is an update in all the preamble blocks where the simulator would have tried to extract by playing a fake σ_p. The update consists in running **Gen** therefore obtaining key pair (σ, τ) for the encryption scheme, and then in sending $\sigma_p = \sigma \oplus \sigma_v$. By the fact that the public key of dense secure cryptosystem has the uniform distribution, we have that $H_4 \approx H_3$.

H_5. The experiment H_5 plays similarly to H_4. The only difference is that in all the preamble blocks where the simulator would have tried to extract, and where the adversary plays the resettable witness indistinguishable proof, if the extracted strings (obtained by decrypting the two encryptions) do not give a secret key then the experiment aborts. If the experiment does not abort, then it continues by using the extracted secret key instead of the one obtained by running in exponential time.

The unconditional soundness of the resettable witness indistinguishable proof guarantees that the above abort can happen only with negligible probability, therefore correct secret keys are extracted during this experiment and can be used after the extraction. Therefore $H_5 \approx H_4$.

H_6. This is our simulator. We no longer give it a witness that $x \in L$, and we no longer allow it to run in exponential time, so it obtains sk_i only through decryptions. Again, by the main technical lemma, the probability that H_6 successfully learns each sk_i before is needed is overwhelming. Our technical lemma shows that H_6 will not output FAIL except with negligible probability. Therefore we have that $H_6 \approx H_5$.

References

[Bar01] Boaz Barak. How to go beyond the black-box simulation barrier. In *FOCS*, pages 106–115, 2001.

[BCNP04] Boaz Barak, Ran Canetti, Jesper Buus Nielsen, and Rafael Pass. Universally composable protocols with relaxed set-up assumptions. In *FOCS*, pages 186–195, 2004.

[BG02] Boaz Barak and Oded Goldreich. Universal arguments and their applications. In *IEEE Conference on Computational Complexity*, pages 194–203, 2002.

[BL02] Boaz Barak and Yehuda Lindell. Strict polynomial-time in simulation and extraction. In *STOC*, pages 484–493, 2002.

[Blu87] Manuel Blum. How to prove a theorem so no one else can claim it. In *Proceedings of the International Congress of Mathematicians*, pages 1444–1451, 1987.

[BSMP91] Manuel Blum, Alfredo De Santis, Silvio Micali, and Giuseppe Persiano. Noninteractive zero-knowledge. *SIAM J. Comput.*, 20(6):1084–1118, 1991.

[Can01] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *FOCS*, pages 136–145, 2001.

[Can04] Ran Canetti. Universally composable signature, certification, and authentication. In *CSFW*, 2004.

[CF01] Ran Canetti and Marc Fischlin. Universally composable commitments. In *CRYPTO*, Lecture Notes in Computer Science, pages 19–40. Springer, 2001.

[CGGM00] Ran Canetti, Oded Goldreich, Shafi Goldwasser, and Silvio Micali. Resettable zero-knowledge (extended abstract). In *STOC*, pages 235–244, 2000.

[CKL03] Ran Canetti, Eyal Kushilevitz, and Yehuda Lindell. On the limitations of universally composable two-party computation without set-up assumptions. In *EUROCRYPT*, pages 68–86, 2003.

[CKL06] R. Canetti, E. Kushilevitz, and Y. Lindell. On the limitations of universally composable two-party computation without set-up assumptions. *J. Cryptology*, 19(2):135–167, 2006.

[CKPR01] Ran Canetti, Joe Kilian, Erez Petrank, and Alon Rosen. Black-box concurrent zero-knowledge requires $\widetilde{\Omega}(\log n)$ rounds. In *STOC*, pages 570–579, 2001.

[CLOS02] R. Canetti, Y. Lindell, R. Ostrovsky, and A. Sahai. Universally composable two-party and multi-party secure computation. In *STOC*, pages 494–503, 2002.

[CLP10] Ran Canetti, Huijia Lin, and Rafael Pass. Adaptive hardness and composable security in the plain model from standard assumptions. In *FOCS*, pages 541–550, 2010.

[CO99] Giovanni Di Crescenzo and Rafail Ostrovsky. On concurrent zero-knowledge with pre-processing. In *CRYPTO*, pages 485–502, 1999.

[DDN91] Danny Dolev, Cynthia Dwork, and Moni Naor. Non-malleable cryptography (extended abstract). In *STOC*, pages 542–552, 1991.

[DGS09] Yi Deng, Vipul Goyal, and Amit Sahai. Resolving the simultaneous resettability conjecture and a new non-black-box simulation strategy. In *FOCS*, pages 251–260, 2009.

[DNS98] Cynthia Dwork, Moni Naor, and Amit Sahai. Concurrent zero-knowledge. In *STOC*, pages 409–418, 1998.

[FFS88] Uriel Feige, Amos Fiat, and Adi Shamir. Zero-knowledge proofs of identity. *Journal of Cryptology*, 1:77–94, 1988.

[FLS90] Uriel Feige, Dror Lapidot, and Adi Shamir. Multiple non-interactive zero knowledge proofs based on a single random string (extended abstract). In *FOCS*, pages 308–317, 1990.

[GGJS12] Sanjam Garg, Vipul Goyal, Abhishek Jain, and Amit Sahai. Concurrently secure computation in constant rounds. In *EUROCRYPT*, 2012.

[GK96] Oded Goldreich and Ariel Kahan. How to construct constant-round zero-knowledge proof systems for np. *J. Cryptology*, 9(3):167–190, 1996.

[GMR85] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof-systems (extended abstract). In *STOC*, pages 291–304, 1985.

[GMW87] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In *STOC '87: Proceedings of the 19th annual ACM conference on Theory of computing*, pages 218–229, New York, NY, USA, 1987. ACM Press.

[Gol02] Oded Goldreich. Concurrent zero-knowledge with timing, revisited. In *STOC*, pages 332–340, 2002.

[IKOS09] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Zero-knowledge proofs from secure multiparty computation. *SIAM J. Comput.*, 39(3):1121–1152, 2009.

[KL11] Dafna Kidron and Yehuda Lindell. Impossibility results for universal composability in public-key models and with fixed inputs. *J. Cryptology*, 24(3):517–544, 2011.

[KLP05] Yael Tauman Kalai, Yehuda Lindell, and Manoj Prabhakaran. Concurrent general composition of secure protocols in the timing model. In *STOC*, pages 644–653, 2005.

[KMO89] Joe Kilian, Silvio Micali, and Rafail Ostrovsky. Minimum resource zero-knowledge proofs (extended abstract). In *FOCS*, pages 474–479, 1989.

[KP01] Joe Kilian and Erez Petrank. Concurrent and resettable zero-knowledge in poly-loalgorithm rounds. In *STOC*, pages 560–569, 2001.

[KPR98] Joe Kilian, Erez Petrank, and Charles Rackoff. Lower bounds for zero knowledge on the internet. In *FOCS*, pages 484–492, 1998.

[Lin03a] Yehuda Lindell. Bounded-concurrent secure two-party computation without setup assumptions. In *STOC*, pages 683–692, 2003.

[Lin03b] Yehuda Lindell. General composition and universal composability in secure multi-party computation. In *FOCS*, pages 394–403, 2003.

[Lin04] Yehuda Lindell. Lower bounds for concurrent self composition. In Moni Naor, editor, *TCC*, volume 2951 of *Lecture Notes in Computer Science*, pages 203–222. Springer, 2004.

[LPV09] Huijia Lin, Rafael Pass, and Muthuramakrishnan Venkitasubramaniam. A unified framework for concurrent security: universal composability from stand-alone non-malleability. In *STOC*, pages 179–188, 2009.

[MP07] Silvio Micali and Rafael Pass. Precise zero knowledge, 2007.

[MR01] Silvio Micali and Leonid Reyzin. Soundness in the public-key model. In *CRYPTO*, pages 542–565, 2001.

[Ost91] Rafail Ostrovsky. One-way functions, hard on average problems, and statistical zero-knowledge proofs. In *Structure in Complexity Theory Conference*, pages 133–138, 1991.

[OW93] Rafail Ostrovsky and Avi Wigderson. One-way fuctions are essential for non-trivial zero-knowledge. In *ISTCS*, pages 3–17, 1993.

[Pas03] Rafael Pass. Simulation in quasi-polynomial time, and its application to protocol composition. In *EUROCRYPT*, pages 160–176, 2003.

[Pas04] Rafael Pass. Bounded-concurrent secure multi-party computation with a dishonest majority. In *STOC*, pages 232–241, 2004.

[PR03] Rafael Pass and Alon Rosen. Bounded-concurrent secure two-party computation in a constant number of rounds. In *FOCS*, pages 404–413, 2003.

[PR05a] Rafael Pass and Alon Rosen. Concurrent non-malleable commitments. In *FOCS*, pages 563–572, 2005.

[PR05b] Rafael Pass and Alon Rosen. New and improved constructions of non-malleable cryptographic protocols. In *STOC*, pages 533–542, 2005.

[PRS02] Manoj Prabhakaran, Alon Rosen, and Amit Sahai. Concurrent zero knowledge with logarithmic round-complexity. In *FOCS*, pages 366–375, 2002.

[PTV10] Rafael Pass, Wei-Lung Dustin Tseng, and Muthuramakrishnan Venkitasubramaniam. Eye for an eye: Efficient concurrent zero-knowledge in the timing model. In *TCC*, pages 518–534, 2010.

[RK99] Ransom Richardson and Joe Kilian. On the concurrent composition of zero-knowledge proofs. In *EUROCRYPT*, pages 415–431, 1999.

[Ros00] Alon Rosen. A note on the round-complexity of concurrent zero-knowledge. In *CRYPTO*, pages 451–468, 2000.

[SCO+01] Alfredo De Santis, Giovanni Di Crescenzo, Rafail Ostrovsky, Giuseppe Persiano, and Amit Sahai. Robust non-interactive zero knowledge. In *CRYPTO*, pages 566–598, 2001.

[SP92] Alfredo De Santis and Giuseppe Persiano. Zero-knowledge proofs of knowledge without interaction. In *FOCS*, pages 427–436. IEEE Computer Society, 1992.

[SV12] Alessandra Scafuro and Ivan Visconti. On round-optimal zero knowledge in the bare public-key model. In *EUROCRYPT 2012 - 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques*, volume 7237 of *Lecture Notes in Computer Science*, pages 153–171. Springer, 2012.

Appendix

A Proof of Soundness

In this section, we prove the soundness of our cZK protocol described in Section 3. In fact, we will prove *concurrent soundness* of $\langle P, V\rangle$, i.e., we will show that a computationally-bounded adversarial prover who engages in multiple concurrent executions of $\langle P, V\rangle$ (where the scheduling across the sessions is controlled by the adversary) cannot prove a false statement in any of the executions, except with negligible probability. We note that similar to the bare-public key model [CGGM00], "stand-alone" soundness does not imply concurrent soundness in our model. Informally speaking, this is because the standard approach of reducing concurrent soundness to stand-alone soundness by "internally" emulating all but one verifier does not work since the verifier's secret keys are private. Indeed, Micali and Reyzin [MR01] gave concrete counter-examples to show that stand-alone soundness does not imply concurrent soundness in the bare public key model. We note that their results immediately extend to our model.

We now proceed to formally prove the concurrent soundness of our protocol $\langle P, V\rangle$. We claim the following theorem.

Theorem 2 *The protocol $\langle P, V\rangle$ presented in Figure 4 is concurrently sound.*

Proof of Theorem 2. We first introduce some notation. Recall that in our protocol, in the execution of $\langle P_{\mathsf{pWI}}, V_{\mathsf{pWI}}\rangle$, the prover proves the OR of two statements. We will call a witness corresponding to the first (resp., second) part of the statement as *true* (resp., *trapdoor*) witness.

We first state a basic lemma related to the soundness of each instance of $\langle P_{\mathsf{pB}}, V_{\mathsf{pB}}\rangle_N$ across all executions of $\langle P, V\rangle$. Its proof is essentially identical to [Bar01], hence below we only discuss a proof sketch, using the terminology of [DGS09].

Lemma 2 *Let $\hat{P}$ be any non-uniform probabilistic polynomial time adversarial prover that engages in any polynomial $m = m(n)$ number of concurrent executions of $\langle P, V\rangle$ with N honest verifiers. Then, every instance of $\langle P_{\mathsf{pB}}, V_{\mathsf{pB}}\rangle_N$ across all executions of $\langle P, V\rangle$ is sound.*

Proof (Sketch). Let us assume the contrapositive, i.e., with non-negligible probability ϵ, there exists at least one pair (i, k) such that $\hat{P}$ successfully convinces the verifier of a false statement in the k^{th} instance (out of $t = \omega(1)$ instances) of $\langle P_{\mathsf{pB}}, V_{\mathsf{pB}}\rangle_N$ in session i. Let S denote the set of all such pairs (i, k) and let $v = |S|$.

Now consider any pair $(i^*, k^*) \in S$. Let $\hat{x}$ denote the statement proved by $\hat{P}$ in $\langle P_{\mathsf{pB}}, V_{\mathsf{pB}}\rangle_N^{i^*,k^*}$. We have that with probability at least ϵ/v, $\hat{x}$ is *false*. In this case, we will construct a super-polynomial time machine M that finds collisions for the hash function.[9] Without loss of generality, assume that $\hat{P}$ is deterministic. Consider the transcript of messages (across all sessions) that occur before $\hat{P}$ sends the second protocol message in $\langle P_{\mathsf{pB}}, V_{\mathsf{pB}}\rangle_N^{i^*,k^*}$. Note that this transcript, in particular, includes the hash function h that the verifier sends to $\hat{P}$ as the first message of $\langle P_{\mathsf{pB}}, V_{\mathsf{pB}}\rangle_N^{i^*,k^*}$. We will call this transcript as the prefix for the rest of the protocol. Let $\epsilon' = \epsilon/v$. Now it must be the case that for at least $\epsilon'/2$ fraction of the prefixes, the probability (over the rest of the verifier coins) that the adversarial prover $\hat{P}$ will succeed in $\langle P_{\mathsf{pB}}, V_{\mathsf{pB}}\rangle_N^{i^*,k^*}$ is at least $\epsilon'/2$. We will call this set of prefixes to be good. The machine M works as follows. It first runs $\hat{P}$ and invokes the weak knowledge extractor E for the universal argument system $\langle P_{\mathsf{pUA}}, V_{\mathsf{pUA}}\rangle$. The probability (over all verifier random coins) of the prefix being good and E succeeding (given that prefix is good) is at least $\frac{\epsilon'}{2} \cdot p(\frac{\epsilon'}{2})$, where p is a polynomial[10]. Now,

[9] As mentioned earlier, for simplicity of exposition, we are assuming that the hash function family is collision-resistant against super-polynomial time adversaries. This assumption can be relaxed by use of good error correcting codes [BG02, Pas04, PR05b, PR05a].

[10] Recall that the success probability of the weak knowledge extractor is polynomially related to the success probability of the prover [Bar01].

if E succeeds and extracts a program, say Π, M rewinds $\hat{P}$ up to the point where it sent the second message in $\langle P_{\mathsf{pB}}, V_{\mathsf{pB}} \rangle_N^{i^*,k^*}$ and continues with fresh random coins; in particular, it chooses a fresh random string $r \xleftarrow{R} \{0,1\}^{\ell(N)}$ in $\langle P_{\mathsf{pB}}, V_{\mathsf{pB}} \rangle_N^{i^*,k^*}$). It then runs the extractor E again and if it succeeds, M obtains another program Π'. By a simple counting argument, it follows immediately that if S_Π is the set of all possible outputs of Π, then $r \in S_\Pi$ with only negligible probability. Thus, we have that $\Pi \neq \Pi'$. However, since $h(\Pi) = h(\Pi')$ (this follows from the computational binding property of $\mathbf{Com}$), we have found collisions Π, Π' for h. The probability of finding collision can be computed as:

$$\begin{aligned} \Pr[\text{Coll}] &= \Pr[\mathsf{pre} \text{ is good prefix}] \cdot \Pr[E \text{ succeeds in two independent executions with } \mathsf{pre}] - \Pr[\Pi = \Pi'] \\ &= \frac{\epsilon'}{2} \cdot (p(\frac{\epsilon'}{2}))^2 - \text{negl}(n). \end{aligned}$$

It follows that the probability of this event is noticeable in n, which is a contradiction. This completes the proof of Lemma 2.

Completing the Proof of Theorem 2. Let us assume the contrapositive, i.e., assume that $\langle P, V \rangle$ is not concurrently sound. Then, with non-negligible probability ϵ, there exists an i such that $\hat{P}$ succeeds in proving a false statement to the verifier in session i. Let S denote the set of all such i and let $v = |S|$.

Now, consider any $i \in S$. Note that it immediately follows from the (stand-alone) soundness of $\langle P_{\mathsf{pWI}}, V_{\mathsf{pWI}} \rangle$ that with probability at least $\frac{\epsilon}{v} - \text{negl}(n)$, $\hat{P}$ use a *trapdoor witness* in $\langle P_{\mathsf{pWI}}, V_{\mathsf{pWI}} \rangle$ in session i. Let $\tilde{V}$ denote the verifier in session i and let $pk = (y_0, y_1)$ denote the public key of $\tilde{V}$. Now, we run $\hat{P}$ such that in all protocol executions involving verifier $\tilde{V}$, we only use the secret key x_b corresponding to y_b, where $b \xleftarrow{R} \{0,1\}$. We now invoke the knowledge extractor E for $\langle P_{\mathsf{pWI}}, V_{\mathsf{pWI}} \rangle$ on $\hat{P}$ in session i. It follows from a standard argument (based on using "good" prefixes) that E successfully extracts a *trapdoor witness* with probability $p = p(\epsilon)$ where p is some polynomial. We now consider two cases:

1. With probability α, E outputs a witness $\hat{x}_{1-b}$ such that $y_{1-b} = f_{\mathsf{owf}}(\hat{x}_{1-b})$.
2. With probability $p - \alpha$, E outputs a witness $\hat{x}_b$ such that $y_b = f_{\mathsf{owf}}(\hat{x}_b)$.

If α is non-negligible (in n), then it is immediate to see that we can build a polynomial-time inverter for one-way function f_{owf}. Specifically, the inverter I for f_{owf} works as follows. It runs the entire experiment with $\hat{P}$ in the same manner as above, except that y_{1-b} is taken from an external challenger for f_{owf}. When E outputs a value $\hat{x}_{1-b}$, I outputs it as the pre-image of y_{1-b} w.r.t. f_{owf}. Note that I succeeds with non-negligible probability α, which is a contradiction.

On the other hand, if α is negligible (in n), then we now focus on the second case. Let $\tilde{m}$ denote the total number of protocol sessions of $\langle P, V \rangle$ involving verifier $\tilde{V}$. Then, we have that with probability $p - \text{negl}(n)$, when $\tilde{V}$ (only) uses the secret key x_0 in all $\tilde{m}$ protocol sessions, the extractor E outputs a value $\hat{x}_0$, and similarly, when $\tilde{V}$ (only) uses x_1, E outputs a value $\hat{x}_1$, where $\hat{x}_b$ is such that $f_{\mathsf{owf}}(\hat{x}_b) = y_b$. Then, by a standard hybrid argument, there exists a session j (out of the $\tilde{m}$ sessions involving $\tilde{V}$) such that when $\tilde{V}$ (only) uses the secret key x_0 (resp., x_1) in session j, the extractor E outputs a value $\hat{x}_0$ (resp., $\hat{x}_1$), with probability at least $p' = \frac{p - \text{negl}(n)}{\tilde{m}}$.[11] Let H_0 (resp., H_1) denote the hybrid experiment where $\tilde{V}$ uses x_0 (resp., x_1) in session j. Let $\hat{x}_b$ be the random variable that denotes the value that E extracts from $\hat{P}$ in experiment H_b.

We will now argue that $\hat{x}_0 \stackrel{c}{\equiv} \hat{x}_1$, except with negligible probability, which is a contradiction to the above hypothesis, and thus concludes our proof. Let $\{e_1^q, e_2^q\}_{q=1}^t$ denote the $t = \omega(1)$ pairs of ciphertexts that $\tilde{V}$ sends to $\hat{P}$ in session j. Further, let $\{\langle P_{\mathsf{rWI}}, V_{\mathsf{rWI}} \rangle^q\}_{q=1}^t$ denote the t instances of $\langle P_{\mathsf{rWI}}, V_{\mathsf{rWI}} \rangle$ in session j. We consider three intermediate hybrid experiments H_{enc_1}, H_{wi} and H_{enc_2} described as follows.

[11] Here, the hybrids are such that $\tilde{V}$ uses x_0 in all session $j' < j$, and x_1 in all sessions $j' > j$.

HYBRID H_{enc_1}: This is the same as H_0, except that $\tilde{V}$ prepares each ciphertext $\{e_1^q\}_{q=1}^t$ to be an encryption of the secret key x_1. We now invoke the knowledge extractor E (for for $\langle P_{\mathsf{pWI}}, V_{\mathsf{pWI}}\rangle$) on $\hat{P}$ in $\langle P_{\mathsf{pWI}}, V_{\mathsf{pWI}}\rangle$ in session i. Let $\hat{x}_{\mathsf{enc}_1}$ be the random variable that denotes the value that E outputs.

We now claim that $\hat{x}_0 \stackrel{c}{\equiv} \hat{x}_{\mathsf{enc}_1}$. Suppose that this is not the case. Then, by a standard hybrid argument, there exists $q \in [t]$ such that $\hat{x}_{0:q}$ is distinguishable from $\hat{x}_{0:q+1}$, where $\hat{x}_{0:q}$ is the random variable that denotes the value extracted by E in the intermediate hybrid experiment $H_{0:q}$ that is essentially the same as H_0, except that $e_1^1, \ldots, e_1^q$ are prepared as encryptions of x_1. (Thus, we have that $H_{0:t}$ is the same as H_{enc_1}.) In this case, we first note that if the execution of $\langle P_{\mathsf{pWI}}, V_{\mathsf{pWI}}\rangle$ in session i concludes *before* $\hat{P}$ receives (e_1^{q+1}, e_2^{q+1}), then the witness used in $\langle P_{\mathsf{pWI}}, V_{\mathsf{pWI}}\rangle$ must be information-theoretically independent of the value encrypted in e_1^{q+1}, which gives us a contradiction. Therefore, we now only consider the case where the execution of $\langle P_{\mathsf{pWI}}, V_{\mathsf{pWI}}\rangle$ in session i concludes *after* $\hat{P}$ receives (e_1^{q+1}, e_2^{q+1}). In this case, we will construct a polynomial-time machine M that breaks the semantic security of the encryption scheme $(\mathbf{Gen}, \mathbf{Enc}, \mathbf{Dec})$.

M works in the same manner as hybrid $H_{0:q}$, except that it also interacts with an external challenger C (for the encryption scheme $(\mathbf{Gen}, \mathbf{Enc}, \mathbf{Dec})$) in the following manner. M receives a public key σ from C and then "forces" it to be the outcome of the $(q+1)^{th}$ coin-tossing subprotocol in session j. Specifically, after receiving the value σ_p from $\hat{P}$ in the $(q+1)^{th}$ coin-tossing subprotocol, M rewinds $\hat{P}$ and sends a value $\sigma_v = \sigma \oplus \sigma_p$. It now sends x_0, x_1 to C and receives a challenge ciphertext e^*. M continues in the same manner as $H_{0:q}$, except that it prepares $e_1^{q+1} = e^*$. Now, note that if e^* is an encryption of x_0, then this machine is identical to $H_{0:q}$, otherwise it is identical to $H_{0:q+1}$. M now invokes the knowledge extractor E on $\hat{P}$ in $\langle P_{\mathsf{pWI}}, V_{\mathsf{pWI}}\rangle$ in session i. Note that the sessions i and j may be interleaved in such a manner that when E rewinds $\hat{P}$ to send a new "challenge" in $\langle P_{\mathsf{pWI}}, V_{\mathsf{pWI}}\rangle$, either of the following two events happen:

1. $\hat{P}$ sends a new commitment string $c' = \mathbf{Com}(\sigma'_p)$ in the $(q+1)^{th}$ coin-tossing subprotocol in session j. In this case, M simply continues session j honestly until it receives σ'_p. At this point, it rewinds $\hat{P}$ again to send a value $\sigma_v = \sigma \oplus \sigma'_p$ and then continues honestly.

2. Alternatively, $\hat{P}$ may simply send a new value σ'_p and then proceed to prove its correctness in the execution of $\langle P_{\mathsf{pB}}, V_{\mathsf{pB}}\rangle_N^{j,q+1}$. If this is the case, then M simply aborts.

Now, conditioned on the event that M does not abort, we have that at some point, E stops and outputs a value, say, $\hat{x}$. Then, M finds b such that $f_{\mathsf{owf}}(\hat{x}) = y_b$ and outputs b to C. It follows easily that M succeeds with noticeable (in n) advantage, which is a contradiction. Thus it only remains to argue that M aborts only with negligible probability. To see this, we first note that it follows from the Soundness Lemma 2 that $\hat{P}$ only proves a true statement in each instance of $\langle P_{\mathsf{pB}}, V_{\mathsf{pB}}\rangle_N$, except with negligible probability. Then, by the computational binding property of the commitment scheme, we have that $\hat{P}$ cannot send decommitment c to two different values σ_p and σ'_p, except with negligible probability. Thus, we have the $\sigma'_p = \sigma_p$, except with negligible probability.

HYBRID H_{wi}: This is the same as H_{enc_1}, except that for every $q \in [t]$, $\tilde{V}$ uses the witness corresponding to e_1^q in the resettable-WI $\langle P_{\mathsf{rWI}}, V_{\mathsf{rWI}}\rangle^q$. We now invoke the knowledge extractor E on $\hat{P}$ in $\langle P_{\mathsf{pWI}}, V_{\mathsf{pWI}}\rangle$ in session i in experiment H_{wi}. Let $\hat{x}_{\mathsf{wi}}$ be the random variable that denotes the value that E outputs.

We now claim that $\hat{x}_{\mathsf{enc}_1} \stackrel{c}{\equiv} \hat{x}_{\mathsf{wi}}$. Suppose that this is not the case. Then by a standard hybrid argument, there exists $q \in [t]$ such that $\hat{x}_{\mathsf{enc}_1:q}$ is distinguishable from $\hat{x}_{\mathsf{enc}_1:q+1}$ with noticeable probability, where $\hat{x}_{\mathsf{enc}_1:q}$ is the random variable that denotes the value extracted by E in the intermediate hybrid experiment $H_{\mathsf{enc}_1:q}$ that is essentially the same as H_{wi}, except that $\tilde{V}$ uses the witness corresponding to e_1^ℓ in $\langle P_{\mathsf{rWI}}, V_{\mathsf{rWI}}\rangle^\ell$ for every $\ell \in [1, q]$. (Thus, we have that $H_{\mathsf{enc}_1:t}$ is the same as H_{enc_1}.) In this case, we will construct a polynomial-time machine M that breaks the resettable witness indistinguishability property of $\langle P_{\mathsf{rWI}}, V_{\mathsf{rWI}}\rangle$. M works in the same manner as hybrid $H_{\mathsf{enc}_1:q}$, except that it forwards the $(q+1)^{th}$ instance of $\langle P_{\mathsf{rWI}}, V_{\mathsf{rWI}}\rangle$ in session j, i.e., $\langle P_{\mathsf{rWI}}, V_{\mathsf{rWI}}\rangle^{q+1}$, to an external prover P (for the resettable-WI protocol $\langle P_{\mathsf{rWI}}, V_{\mathsf{rWI}}\rangle$) in the following manner. M first gives w_1,

w_2 to P, where w_1 is the witness corresponding to e_1^{q+1}, and similarly, w_2 is the witness corresponding to e_2^{q+1}. Now, during the execution of $\langle P_{\mathsf{rWI}}, V_{\mathsf{rWI}}\rangle^{q+1}$, M simply forwards each message msg_P from P to $\hat{P}$ and similarly forwards each response $\mathsf{msg}_{\hat{P}}$ from $\hat{P}$ to P. It then runs the knowledge extractor E on $\hat{P}$ in $\langle P_{\mathsf{pWI}}, V_{\mathsf{pWI}}\rangle$ in session i to extract a value, say $\hat{x}$. Note that if sessions i and j are scheduled such that when E rewinds $\hat{P}$ in $\langle P_{\mathsf{pWI}}, V_{\mathsf{pWI}}\rangle$ in session i, $\hat{P}$ sends a new l^{th} round-message $\mathsf{msg}'_{\hat{P}}$ in $\langle P_{\mathsf{rWI}}, V_{\mathsf{rWI}}\rangle^{q+1}$, then M *resets* P to the point where its supposed to receive the l^{th} round message and sends $\mathsf{msg}'_{\hat{P}}$. It then continues the execution in the same manner as described above. When E finally outputs $\hat{x}$, then M finds b such that $f_{\mathsf{owf}}(\hat{x}) = y_b$ and outputs b to P. It follows easily that M succeeds with noticeable (in n) advantage, which is a contradiction.

HYBRID H_{enc_2}: This is the same as H_{enc_2}, except that $\tilde{V}$ prepares each ciphertext e_2^q to be an encryption of x_1. We now run the extractor E on $\hat{P}$ in experiment H_{enc_2}. Let $\hat{x}_{\mathsf{enc}_2}$ be the random variable that denotes the value that E outputs. For the same reasons as argued above (for Hybrid H_{enc_1}), it follows that $\hat{x}_{\mathsf{wi}} \stackrel{c}{\equiv} \hat{x}_{\mathsf{enc}_2}$.

This concludes the proof of Theorem 2.

B Concurrent Self-Composition in the Bounded Player Model

In this section, we present the definition for concurrent (self-composition) secure multi-party computation in the bounded player model. The definition we give below is an adaptation of the definition of concurrent secure computation with adaptive inputs [Lin04, Pas04], to the setting of bounded player model. Parts of the definition below have been taken almost verbatim from [Lin04, Pas04].

We first setup notation. We denote computational indistinguishability by $\stackrel{c}{\equiv}$, and the security parameter by n. For notational simplicity, we let the lengths of the parties' inputs be n. An n-ary functionality is denoted as $f : (\{0,1\}^*)^n \rightarrow (\{0,1\}^*)^n$, where $f = f_1, \ldots, f_n$. Let $P_1, \ldots, P_n$ denote the set of n-player that wish to jointly compute f. The output of P_i with input x_i is defined to be $f_i(\vec{x})$, where $\vec{x} = x_1, \ldots, x_n$. In the context of concurrent composition, each party uses many inputs (one per execution) and these may be chosen *adaptively* based on previous outputs. The fact that bounded player model is considered relates to the fact that the total number of parties that may engage in concurrent protocol executions is a-priori bounded.

In this work, we consider a malicious, static adversary. The scheduling of the messages across the concurrent executions is controlled by the adversary. We do not focus on fairness, hence we do not guarantee output delivery. 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, where a trusted party computes the function output on the inputs of the parties. Unlike in the case of stand-alone computation, in the setting of concurrent executions, the trusted party computes the functionality many times, each time upon different inputs. We now proceed to describe the ideal and real models of computation.

IDEAL MODEL. In the ideal model, there is a trusted party that computes the functionality f based on the inputs handed to it by the player. Let there be N parties $P_1, \ldots, P_N$ where arbitrary (possibly intersecting) subsets of n parties may engage in an arbitrary (polynomial) number of concurrent sessions. Let $\mathcal{I} \subset N$ denote the subset of corrupted parties controlled by the adversary. An execution in the ideal model with an adversary with auxiliary input z corrupting parties $\mathcal{I}$ proceeds as follows:

Inputs: The inputs of the parties $P_1, \ldots, P_N$ are respectively determined by probabilistic polynomial time Turing machines $M_1, \ldots, M_N$ and the initial inputs $x_1, \ldots, x_N$ 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 n that depends on the number of sessions initiated by the adversary.

Session initiation: The adversary initiates a new session by sending a $(\mathsf{start\text{-}session}, P_i)$ to the trusted party. If $P_i \notin \mathcal{I}$, then the trusted party sends $(\mathsf{start\text{-}session}, s)$ to P_i, where s is the index of the session.

Honest parties send inputs to trusted party: Upon receiving $(\mathsf{start\text{-}session}, s)$ from the trusted party, honest party P_i applies its input-selecting machine M_i to its initial input x_i, the session number s and its previous outputs, and obtains a new input $x_{i,j}$.[12] P_i then sends $(s, x_{i,j})$ to the trusted party.

Corrupted parties send inputs to trusted party: Whenever the adversary wishes, it may send a message $(s, x'_{i,j})$ to the trusted party for any $x'_{i,s} \in \{0,1\}^n$ of its choice, on behalf of a corrupted party P_i. It can send the pairs $(s, x'_{i,s})$ in any order it wishes and can also send them adaptively. The only limitation is that for any s, at most one pair indexed by s can be sent to the trusted party on behalf of P_i.

Trusted party answers corrupted parties: When the trusted party has received messages $(s, x'_{i,j})$ from a set of n parties $P_{\ell_1}, \ldots, P_{\ell_n}$ (where $\ell_1, \ldots, \ell_n \in [N]$), it sets $\vec{x}'_s = (x'_{\ell_1,s}, \ldots, x'_{\ell_n,s})$. It then computes $f(\vec{x}'_s)$ and sends $(s, f_{\ell_i}(\vec{x}'_s))$ to party P_{ℓ_i} for every $\ell_i \in \mathcal{I}_s$, where $\mathcal{I}_s \subseteq \mathcal{I}$ denotes the set of corrupted parties in session s. Note that $\mathcal{I}_s$ must be such that $|\mathcal{I}_s| < n$.

Adversary instructs the trusted party to answer honest parties: When the adversary sends a message of the type $(\mathsf{send\text{-}output}, s, \ell_i)$ to the trusted party, the trusted party sends $(s, f_{\ell_i}(\vec{x}'_s))$ to party P_{ℓ_i}.

Outputs: Each honest party P_i always outputs the values $f_i(\vec{x}'_s)$ that it obtained from the trusted party. The adversary may output an arbitrary (probabilistic polynomial-time computable) function of its initial-input and the messages obtained from the trusted party.

Let $\mathcal{S}$ be a non-uniform probabilistic polynomial-time machine (representing the ideal-model adversary). Then, the $\mathsf{ideal\ execution}$ of f with security parameter n, input selecting machines $M = M_1, \ldots, M_N$, initial inputs $\vec{x} = (x_1, \ldots, x_N)$ and auxiliary input z to $\mathcal{S}$, denoted $\text{IDEAL}^N_{f,\mathcal{I},\mathcal{S},M}(n, \vec{x}, z)$, is defined as the output vector of the honest parties and $\mathcal{S}$ from the above ideal execution.

REAL MODEL. We next consider the real model in which a real two-party protocol is executed (and there exists no trusted third party). Let f, $\mathcal{I}$, N be as above and let Π be a multi-party protocol for computing f. Let A denote the adversary. Then, the real concurrent execution of Π with security parameter n, input selecting machines $M = M_1, \ldots, M_N$, initial inputs $\vec{x} = (x_1, \ldots, x_N)$ and auxiliary input z to A, denoted $\text{REAL}^N_{\Pi,\mathcal{I},A,M}(n, \vec{x}, z)$, is defined as the output vector of the honest parties and A, resulting from the following real-world process. The real world execution proceeds as follows. Each honest party P_i first chooses an identity id_i and registers it with F^N_{bp}. A corrupted party may choose to register its identity at any time it wishes, even after the computation begins. An honest party initiates a new session whenever it receives a $\mathsf{start\text{-}session}$ message from A. It then applies its input selecting machine to its initial input, the session number and its previously received outputs, and obtains the input for this session. Note that arbitrary (possibly intersecting) sets of n (out of N) player may be participating in concurrent executions of Π. 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 $(s, \mathsf{msg}, P_i, P_j)$ to an honest party P_i on behalf of a corrupted party P_j. If that honest party is participating in session s, and this is the first message it has received from P_j, then it first retrieves the identity id_j of P_j from F^N_{bp}. It then adds (msg, P_i, P_j) to its view of session s and replies according to the instructions of Π and this view.

SECURITY DEFINITION. Having defined the ideal and real models of computation, we are now ready to give our formal security definition.

[12] Specifically, in the first session, $x_{i,1} = M_i(x_i, 1)$. In the later sessions s, $x_{i,s} = M_i(x_i, s, y_{i,1}, \ldots, y_{i,w})$, where w sessions have concluded and the outputs of P_i were $y_{i,1}, \ldots, y_{i,w}$.

Definition 2 (Concurrent Self-Composition in Bounded Player Model).) *Let $N = N(n)$ be a polynomial and let f and Π be as above. Protocol Π is said to* securely compute *f under concurrent composition in the N-bounded player model if for every real model non-uniform probabilistic polynomial-time adversary A, there exists an ideal-model non-uniform probabilistic expected polynomial-time adversary $\mathcal{S}$, such that for all input-selecting machines $M = M_1, \ldots, M_N$, every $z \in \{0,1\}^*$, every $\vec{x} = (x_1, \ldots, x_N)$, and every $\mathcal{I} \subset [N]$,*[13]

$$\left\{\text{IDEAL}^{N}_{f,\mathcal{I},\mathcal{S},M}(n,\vec{x},z)\right\}_{n\in\mathbb{N}} \stackrel{c}{\equiv} \left\{\text{REAL}^{F^N_{bp}}_{\Pi,\mathcal{I},A,M}(n,\vec{x},z)\right\}_{n\in\mathbb{N}}$$

C Impossibility Results in Bounded Player Model

In [Lin04], Lindell gave broad impossibility results for unbounded concurrent self-composition in the standard model. We observe that the impossibility result of [Lin04] carries over in a straightforward manner to bounded player model considered in the present work. Below, in what is largely an informal discussion, we elaborate on this observation. [Lin04, Lin03b, CKL06, KL11]

Lindell's impossibility result [Lin04] for unbounded concurrent self-composition in the standard model is obtained by combining three different results. Below, we will recall all of these results and discuss how each of them carry over to the bounded player model. First, we recall some basic definitions from [Lin04]. A large part of text below is taken verbatim from [Lin04].

Security under concurrent general composition. Informally speaking, concurrent general composition considers the case that a protocol ρ for securely computing some functionality f, is run concurrently (many times) with arbitrary other protocols π. In other words, the secure protocol ρ is run many times in a network in which arbitrary activity takes place. (Note that in contrast, in concurrent self-composition, we only consider security for concurrent executions of the same protocol ρ.) The formalize security in this setting, we model the arbitrary network activity π as a "calling protocol" with respect to the functionality ff. That is, π is a protocol that contains, among other things, "ideal calls" to a trusted party that computes a functionality f. This means that in addition to standard messages sent between the parties, protocol π's specification contains instructions of the type "send the value x to the trusted party and receive back output y". Then, the real-world scenario is obtained by replacing the ideal calls to f in protocol π with real executions of protocol ρ. The composed protocol is denoted π^ρ and it takes place without any trusted help. Security is defined by requiring that for every protocol π that contains ideal calls to f, an adversary interacting with the composed protocol π^ρ (where there is no trusted help) can do no more harm than in an execution of π where a trusted party computes all the calls to f. This therefore means that ρ behaves just like an ideal call to f, even when it is run concurrently with any arbitrary protocol π. We refer the reader to [Lin04] for a formal security definition.

Concurrent general composition in the bounded player model. We note that security under concurrent general composition can be naturally defined in the bounded player model by considering an a-priori bound on the total number of player in the system, in the same manner as in Definition 2. More specifically, we will consider an a-priori bound N on the total number of player in the system. Then, arbitrary (possibly intersecting) subsets of parties may be involved in unbounded concurrent executions of ρ, in the presence of arbitrary other protocols π. (Note that π can be at-most an N-party protocol.) Security is defined in the same manner as above.

Functionalities that enable bit transmission. Informally speaking, a functionality enables bit transmission if it can be used by the parties to send bits to each other. We now recall the formal definition from [Lin04].

[13] Here it should be implicit that $\mathcal{I}$ is such that the adversary corrupts at most $n-1$ parties in each protocol execution.

Definition 3 (Bit-transmitting functionality) *A deterministic functionality $f = (f_1, f_2)$ enables bit transmission from P_1 to P_2 if there exists an input y for P_2 and a pair of inputs x, x' for P_1 such that $f_2(x; y) \neq f_2(x'; y)$. Likewise, f enables bit transmission from P_2 to P_1 if there exists an input x for P_1 and a pair of inputs y, y' for P_2 such that $f_1(x; y) \neq f_1(x; y')$. A functionality enables bit transmission if it enables bit transmission from P_1 to P_2 and from P_2 to P_1.*

The above definition can be easily generalized to probabilistic functionalities, as well as to multi-party functionalities in a straightforward way. We refer the reader to [Lin04] for more details.

Extending Lindell's impossibility result to bounded player model. We now consider the three steps involved in the impossibility result in [Lin04], and briefly discuss why they carry over to the bounded player model.

Step 1: First, it is shown in [Lin04] that for every functionality f that enables bit transmission, security under unbounded concurrent self-composition is equivalent to security under concurrent *general* composition. That is, if f enables bit transmission, then f can be securely computed unbounded concurrent self-composition if and only if it can be securely computed under concurrent general composition.

We note that [Lin04] proves this (unconditional) result for two-party setting where only *one set* of parties run all of the protocol executions. As such, the result already works in the bounded player model.

Step 2: Next, we use the result of [Lin03b], where it is shown that security under concurrent general composition implies security in the universal composability framework [Can01]. This result is also unconditional, and in fact, also works in a setup model (such as a common reference string, etc).

Once again, we note that [Lin04] obtains this result even for the restrictive case where only *one set* of parties engage in two-party protocol executions (the adversary is assumed to be static). As such, this result is also applicable to the bounded player model.

Step 3: Finally, one can use the result of Canetti et al. [CKL06] that shows a large class of functionalities for which UC security cannot be achieved. With respect to the bounded player model, we note that very recently, Kidron and Lindell [KL11] show that the results of [CKL06] can be extend to the bulletin-board certificate authority model, which is formalized in essentially the same manner as our bounded player model, in that the parties register their unique identities to a functionality. We note that the result in [KL11] already works when the number of parties are a-priori bounded, as such it is applicable to our setting.

Combining these three steps, we can obtain broad impossibility results for concurrent self-composition in the bounded player model. In order to obtain the formal statement, let us first recall the class of functionalities Ψ for which concurrent general composition is shown to be impossible [Lin03b]. The following is taken verbatim from [Lin04, Lin03b].

1. Let $f : \{0,1\}^* \to \{0,1\}^*$ be a deterministic polynomial-time function that is (weakly) one-way. Then, the functionality $(x, \lambda) \to (\lambda, f(x))$ cannot be securely computed under concurrent general composition by any non-trivial protocol.

2. Let $f : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$ be a deterministic polynomial-time functionality. If f depends on both parties' inputs, then the functionality $(x, y) \to (f(x, y), f(x, y))$ cannot be securely computed under concurrent general composition 2. Let $f : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^* \times \{0,1\}^*$ be a deterministic polynomial-time functionality and let $f = (f_1, f_2)$. If f is not completely revealing[14] then the functionality $(x, y) \to (f_1(x, y), f_2(x, y))$ cannot be securely computed under concurrent general composition by any non-trivial protocol.

[14] Informally, a functionality is completely revealing if one party can choose an input so that the output of the functionality will reveal the other party's input. See [Lin03b, Lin04] for details.

Further, let Φ be the set of all two-party functionalities that enable bit transmission. Then, we obtain the following result:

Corollary 1 *Let f be a functionality in $\Phi \cap \Psi$. Then f cannot be securely computed under unbounded concurrent self composition by any non-trivial protocol.*

Remark. We note that the above discussion is relevant to the "fixed-roles" setting where the parties play the same roles in each session in the concurrent self-composition setting. If we allow interchangeable roles, then as shown in [Lin04], essentially all functionalities are impossible to realize. We refer the reader to [Lin04] for more details.