Concurrent Zero Knowledge in the Bounded Player Model

Vipul Goyal[1], Abhishek Jain[2], Rafail Ostrovsky[3], Silas Richelson[4], and Ivan Visconti[5]

[1] Microsoft Research, India
vipul@microsoft.com
[2] MIT and Boston University, USA
abhishek@csail.mit.edu
[3] UCLA, USA
rafail@cs.ucla.edu
[4] UCLA, USA
sirichel@math.ucla.edu
[5] 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 is *not* a priori bounded. 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. Further, we note 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 [21], 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, and have found numerous applications in cryptography.

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 [16] initiated the study of 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 [34,25,33]. In particular, the $\tilde{O}(\log k)$-round construction of [33] nearly matches the lower bound of $\tilde{\Omega}(\log k)$ w.r.t. black-box simulation [11].

Despite a decade of research, the $\tilde{O}(\log k)$-round construction of [33] is still the most round-efficient cZK protocol known. Indeed, the lower bound of [11] 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: Bounded Concurrency. 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.

An interesting relaxation of the standard model (and related to our setting) that has been previously studied is the bounded-concurrency model [2], where an *a priori 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 [2], and also constant-round bounded-concurrent secure two-party and multi-party computation [31].

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 [2,31] 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 preprocessing model was proposed in [24] in the stand-alone setting and in [13] in the context of cZK. In [13], 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 phases can not overlap in time. An improved model was later proposed in [10] where the preprocessing is required to

[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.

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 with concurrent soundness under standard assumptions [14,15,37,36].

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 cZK is known to exist in the BPK model using only black-box simulation, while in our setting, non-black-box techniques are *necessary* to achieve sublogarithmic-round cZK.

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 [30], 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 [21]). Indeed, this relaxation has yielded not only constant-round cZK [30], but also concurrent-secure computation [26,12,18]. This stands in contrast to the standard model, where concurrent-secure computation is known to be impossible to achieve [27] even with static input [5,1,19]. Other models where constant round cZK (as well as concurrently secure computation) is known include the timing model [16], the common reference string [7,8] model, etc.

Our Question. While the above relaxations of the standard model 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 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]

Our security proof is (necessarily) non-black-box, 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 [29] 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. An important problem left open by our work is the existence of a constant round concurrent zero-knowledge protocol in the bounded player model. Our techniques (necessarily) require a super-constant number of rounds to keep the simulation time polynomial.

We further show that the impossibility results of Lindell for concurrent-secure computation [27] 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. [11] is applicable to our setting as well. Indeed, the impossibility result of [11] relies on

[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 trapdoor permutations by modifying our protocol to use the coin-tossing protocol of Barak and Lindell [4].

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 [11] 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 [17] 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 [2].

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 [4]): 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 [2]. 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.

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 [2] 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

[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.

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 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 [2] 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.

Impossibility of Concurrent-secure Computation. 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 [31]. 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 [27] also holds for the Bounded Player model. (See the full version [22] 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)

[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.

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* [2,31], 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. 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 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 $(\textbf{register}, sid, \mathsf{id}_i)$ from some party P_i, the functionality checks that no pair (P_i, id'_i) is already recorded and that $count < N$. If this is the case, it records the pair (P_i, id_i) and sets $count = count + 1$. Other wise, it ignores the received message.
- **Retrieve commands:** Upon receiving a message $(\mathsf{retrieve}, sid, P_i)$ 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 $(sid, P_i, \mathsf{id}_i)$ to P_j (or A). Otherwise, it returns $(sid, P_i, \perp)$.

Fig. 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 [23], which suffices for obtaining authenticated channels [9]. We finally remark that our model is also closely related to the *Bare Public-Key model*, introduced by Canetti et al. [10]. 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 [33] to the Bounded Player model, by also considering non-black-box simulation. Some of the text below is taken verbatim from [33].

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.[6]

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,*
$\{\mathsf{view}_{V^*}^P(x, z, N)\}_{x \in L, z \in \{0,1\}^*, N \in poly(n)}$ *and* $\{\mathcal{S}(x, z, N)\}_{x \in L, z \in \{0,1\}^*, N \in poly(n)}$.

2.3 Building Blocks

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

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

Perfectly Hiding Commitment Scheme. In our constructions, we will make use of a perfectly hiding string commitment scheme, denoted **Com**. For simplicity of exposition, we will make the simplifying assumption that **Com** is a non-interactive perfectly hiding commitment scheme (even though such a scheme cannot exist). In reality, **Com** would be taken to be a 2-round commitment scheme, which can be based on collections of claw-free permutations [20]. 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 [6] 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 [32]. 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 [2]. Similarly to [32], 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 [10] 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 [10] that such a proof system can be based on perfectly hiding commitments.

Dense Cryptosystems [35]. 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 **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 **Gen**. For simplicity of exposition, we will assume that the **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 [35].

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 [2] $\mathbf{NTIME}(T(n))$ relation R_{sim}, as used previously in [32]. Let $T : \mathbb{N} \rightarrow \mathbb{N}$ 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 **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: $|y| \leq |r| - n$, $c = \mathbf{Com}(h(\Pi); s)$ and $\Pi(y) = r$ within $T(n)$ steps.

Fig. 2. R_{sim} - A variant of Barak's relation [32]

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 *[3,31]*, 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 $c = \mathbf{Com}(\mathsf{ECC}(h(\Pi)); s)$.

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

Stage 1 (Preamble Phase):

$V \to P$: Send $h \overset{\mathrm{R}}{\leftarrow} \mathcal{H}_n$.
$P \to V$: Send $c = \mathbf{Com}(0^n)$.
$V \to P$: Send $r \overset{\mathrm{R}}{\leftarrow} \{0,1\}^{\ell(N)}$.

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

Fig. 3. Protocol $\langle P_{\mathsf{pB}}, V_{\mathsf{pB}} \rangle_N$

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 the full version [22]. We remark that, in fact, we prove *concurrent soundness* of $\langle P, V \rangle$, 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

Parameters: Security parameter n, $N = N(n)$, $t = \omega(1)$.
Common Input: $x \in \{0,1\}^{\mathrm{poly}(n)}$.
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. $b \overset{\mathrm{R}}{\leftarrow} \{0,1\}$, $y_b = f_{\mathsf{owf}}(x_b)$.

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

- $V \to P$: Send $pk = (y_0, y_1)$.
- $P \to V$: Choose $\sigma_p \overset{\mathrm{R}}{\leftarrow} \{0,1\}^n$ and send $c_p = \mathbf{Com}(\sigma_p)$.
- $V \to P$: Send $\sigma_v \overset{\mathrm{R}}{\leftarrow} \{0,1\}^n$.
- $P \to V$: 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. $\exists w \in \{0,1\}^{\mathrm{poly}(|x|)}$ s.t. $R_L(x, w) = 1$.
  2. $\exists \langle b, x_b \rangle$ s.t. $y_b = f_{\mathsf{owf}}(x_b)$.

Fig. 4. Protocol $\langle P, V \rangle$

Bare Public-Key model [10], "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 [29] 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 [2] 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 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 specifically, 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[7] 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

[7] 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.

controls the scheduling of the messages. We give SIM non-black-box access to V^*. Throughout the interaction, SIM keeps track of a tuple $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_N)$ representing the secret keys SIM has learned so far. At any point during the interaction either $\beta_i = \mathrm{sk}_i$ (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 $\boldsymbol{\beta}$ throughout the interaction as it extracts secret keys. Additionally, SIM keeps a counter vector $\boldsymbol{a} = (a_1, \ldots, a_N)$, incrementing a_i each time it executes a preamble block using SIM_{extract} against V_i^*. 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

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

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_i^*. 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 $c_p = \mathrm{Com}(\sigma_p; s)$ for some s. Upon receiving σ_v, it returns σ_p and completes $\langle P_{\mathrm{pB}}, V_{\mathrm{pB}} \rangle$ using s as its witness. Its receipt of encryptions (e_1, e_2) and acceptance of $\langle P_{\mathrm{rWI}}, V_{\mathrm{rWI}} \rangle$ ends the preamble block. If SIM_{easy} does not accept V^*'s execution of $\langle P_{\mathrm{rWI}}, V_{\mathrm{rWI}} \rangle$ it aborts the interaction, as would an honest prover.

When SIM_{easy} is called to execute $\langle P_{\mathrm{pWI}}, V_{\mathrm{pWI}} \rangle$ then it checks if the secret key of the verifier is in β. If yes, SIM_{easy} completes $\langle P_{\mathrm{pWI}}, V_{\mathrm{pWI}} \rangle$ 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 $\big((i,j,k);\phi\big)$ 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 $\text{Com}\big(h(\Pi); s\big)$, 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 $\big((i, j, k); \phi\big) \in \Psi$, 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 $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_N)$ 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 $\text{Com}\big(h(\Pi); s\big)$ 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* $\boldsymbol{k'} = (k'_1, \ldots, k'_m)$ 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 $\boldsymbol{k} = (k_1, \ldots, k_m)$ 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 $\boldsymbol{k'}$ and a polynomial m, let $P_{(\boldsymbol{k'},m)}(W, L)$ 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 $\boldsymbol{k'}$. The probability is over SIM's choice of $\boldsymbol{k}$ with $k_j \in \{1, \dots, t\}$ chosen independently according to D_t (defined below) for all $j = 1, \dots, m$.

The Distribution D_t and the Main Technical Lemma. Define D_t to be the distribution on $\{1, \dots, t\}$ such that $p_{k'} = \text{Prob}_{k \in D_t}\big(k = k'\big) = \varepsilon n^{k'}$, where ε is such that $\sum p_{k'} = 1$. Note that ε is negligible in n.

Lemma 1 (Main Technical Lemma). *Let $\boldsymbol{k'}$ be a verifier strategy and $m = m(n)$ a polynomial. Then we have $P_{(\boldsymbol{k'},m)}(n^3, 0)$ 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 [33] and [28].

Proof. We fix a verifier strategy $\boldsymbol{k'}$ and a polynomial m and write $P(W, L)$ instead of $P_{(\boldsymbol{k'},m)}(W, L)$. 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 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 $P\big((W, L) = (n^3, 0)\big|W + L = n^3\big)$ 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 $p \leq \big(1 - \frac{1}{2n+1}\big)$. 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: $P(n^3, 0) \leq P\big((W, L) = (n^3, 0)\big|W + L = n^3\big) = \sum_{C\in\mathcal{C}} P\big((W, L) = (n^3, 0)\big|C\big)P(C) \leq e^{-n} \sum_{C\in\mathcal{C}} P(C) = e^{-n}$ as desired.

Bounding the length parameter $\ell(N)$. From the above lemma, it follows 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.

In the full version [22] we show through a series of hybrid experiments that the simulation is perfectly indistinguishable from the real game.

Acknowledgments. Work supported in part by NSF grants 0830803, 09165174, 1065276, 1118126 and 1136174, US-Israel BSF grant 2008411, OKAWA Foundation Research Award, IBM Faculty Research Award, Xerox Faculty Research Award, B. John Garrick Foundation Award, Teradata Research Award, European Commission through the FP7 programme under contract 216676 ECRYPT II, MIUR Project PRIN "GenData 2020" and Lockheed-Martin Corporation Research Award. This material is based upon work supported by the Defense Advanced Research Projects Agency through the U.S. Office of Naval Research under Contract N00014 − 11 − 1 − 0392. The views expressed are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

References

1. Agrawal, S., Goyal, V., Jain, A., Prabhakaran, M., Sahai, A.: New Impossibility Results for Concurrent Composition and a Non-interactive Completeness Theorem for Secure Computation. In: Safavi-Naini, R. (ed.) CRYPTO 2012. LNCS, vol. 7417, pp. 443–460. Springer, Heidelberg (2012)
2. Barak, B.: How to go beyond the black-box simulation barrier. In: FOCS, pp. 106–115 (2001)
3. Barak, B., Goldreich, O.: Universal arguments and their applications. In: IEEE Conference on Computational Complexity, pp. 194–203 (2002)
4. Barak, B., Lindell, Y.: Strict polynomial-time in simulation and extraction. In: STOC, pp. 484–493 (2002)
5. Barak, B., Prabhakaran, M., Sahai, A.: Concurrent non-malleable zero knowledge. In: FOCS, pp. 345–354 (2006)
6. Blum, M.: How to prove a theorem so no one else can claim it. In: Proceedings of the International Congress of Mathematicians, pp. 1444–1451 (1987)
7. Blum, M., Santis, A.D., Micali, S., Persiano, G.: Noninteractive zero-knowledge. SIAM J. Comput. 20(6), 1084–1118 (1991)
8. Canetti, R., Lindell, Y., Ostrovsky, R., Sahai, A.: Universally composable two-party and multi-party secure computation. In: STOC, pp. 494–503 (2002)
9. Canetti, R.: Universally composable signature, certification, and authentication. In: CSFW (2004)
10. Canetti, R., Goldreich, O., Goldwasser, S., Micali, S.: Resettable zero-knowledge (extended abstract). In: STOC, pp. 235–244 (2000)
11. Canetti, R., Kilian, J., Petrank, E., Rosen, A.: Black-box concurrent zero-knowledge requires $\tilde{\Omega}(\log n)$ rounds. In: STOC, pp. 570–579 (2001)
12. Canetti, R., Lin, H., Pass, R.: Adaptive hardness and composable security in the plain model from standard assumptions. In: FOCS, pp. 541–550 (2010)
13. Di Crescenzo, G., Ostrovsky, R.: On Concurrent Zero-Knowledge with Pre-processing (Extended Abstract). In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 485–502. Springer, Heidelberg (1999)
14. Di Crescenzo, G., Persiano, G., Visconti, I.: Constant-Round Resettable Zero Knowledge with Concurrent Soundness in the Bare Public-Key Model. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 237–253. Springer, Heidelberg (2004)
15. Di Crescenzo, G., Visconti, I.: Concurrent Zero Knowledge in the Public-Key Model. In: Caires, L., Italiano, G.F., Monteiro, L., Palamidessi, C., Yung, M. (eds.) ICALP 2005. LNCS, vol. 3580, pp. 816–827. Springer, Heidelberg (2005)
16. Dwork, C., Naor, M., Sahai, A.: Concurrent zero-knowledge. In: STOC, pp. 409–418 (1998)
17. Feige, U., Lapidot, D., Shamir, A.: Multiple non-interactive zero knowledge proofs based on a single random string (extended abstract). In: FOCS, pp. 308–317 (1990)
18. Garg, S., Goyal, V., Jain, A., Sahai, A.: Concurrently Secure Computation in Constant Rounds. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 99–116. Springer, Heidelberg (2012)

19. Garg, S., Kumarasubramanian, A., Ostrovsky, R., Visconti, I.: Impossibility Results for Static Input Secure Computation. In: Safavi-Naini, R. (ed.) CRYPTO 2012. LNCS, vol. 7417, pp. 424–442. Springer, Heidelberg (2012)
20. Goldreich, O., Kahan, A.: How to construct constant-round zero-knowledge proof systems for np. J. Cryptology 9(3), 167–190 (1996)
21. Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof-systems (extended abstract). In: STOC, pp. 291–304 (1985)
22. Goyal, V., Jain, A., Ostrovsky, R., Richelson, S., Visconti, I.: Concurrent zero knowledge in the bounded player model. IACR Cryptology ePrint Archive 2012, 279 (2012)
23. Kidron, D., Lindell, Y.: Impossibility results for universal composability in public-key models and with fixed inputs. J. Cryptology 24(3), 517–544 (2011)
24. Kilian, J., Micali, S., Ostrovsky, R.: Minimum resource zero-knowledge proofs (extended abstract). In: FOCS, pp. 474–479 (1989)
25. Kilian, J., Petrank, E.: Concurrent and resettable zero-knowledge in poly-loalgorithm rounds. In: STOC, pp. 560–569 (2001)
26. Lin, H., Pass, R., Venkitasubramaniam, M.: A unified framework for concurrent security: universal composability from stand-alone non-malleability. In: STOC, pp. 179–188 (2009)
27. Lindell, Y.: Lower Bounds for Concurrent Self Composition. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 203–222. Springer, Heidelberg (2004)
28. Micali, S., Pass, R.: Precise zero knowledge (2007)
29. Micali, S., Reyzin, L.: Soundness in the Public-Key Model. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 542–565. Springer, Heidelberg (2001)
30. Pass, R.: Simulation in Quasi-Polynomial Time, and its Application to Protocol Composition. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 160–176. Springer, Heidelberg (2003)
31. Pass, R.: Bounded-concurrent secure multi-party computation with a dishonest majority. In: STOC, pp. 232–241 (2004)
32. Pass, R., Rosen, A.: New and improved constructions of non-malleable cryptographic protocols. In: STOC, pp. 533–542 (2005)
33. Prabhakaran, M., Rosen, A., Sahai, A.: Concurrent zero knowledge with logarithmic round-complexity. In: FOCS, pp. 366–375 (2002)
34. Richardson, R., Kilian, J.: On the Concurrent Composition of Zero-Knowledge Proofs. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 415–431. Springer, Heidelberg (1999)
35. Santis, A.D., Persiano, G.: Zero-knowledge proofs of knowledge without interaction. In: FOCS, pp. 427–436. IEEE Computer Society (1992)
36. Scafuro, A., Visconti, I.: On Round-Optimal Zero Knowledge in the Bare Public-Key Model. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 153–171. Springer, Heidelberg (2012)
37. Visconti, I.: Efficient Zero Knowledge on the Internet. In: Bugliesi, M., Preneel, B., Sassone, V., Wegener, I. (eds.) ICALP 2006, Part II. LNCS, vol. 4052, pp. 22–33. Springer, Heidelberg (2006)