Nearly Simultaneously Resettable Black-Box Zero Knowledge

Joshua Baron[1], Rafail Ostrovsky[2], and Ivan Visconti[3]

[1] UCLA, Los Angeles, CA, USA 90095
jwbaron@math.ucla.edu, rafail@cs.ucla.edu
[2] Università di Salerno, 84084 Fisciano (SA) - Italy
visconti@dia.unisa.it

Abstract. An important open question in Cryptography concerns the possibility of achieving secure protocols even in the presence of physical attacks. Here we focus on the case of proof systems where an adversary forces the honest player to re-use its randomness in different executions. In 2009, Deng, Goyal and Sahai [1] constructed a *simultaneously resettable* non-black-box zero-knowledge argument system that is secure against resetting provers *and* verifiers.

In this work we study the case of the black-box use of the code of the adversary and show a nearly simultaneously resettable black-box zero-knowledge proof systems under standard assumptions. Compared to [1], our protocol is a *proof* (rather then just argument) system, but requires that the resetting prover can reset the verifier up to a bounded number of times (which is unavoidable for black-box simulation), while the verifier can reset the prover an arbitrary polynomial number of times. The main contribution of our construction is that the round complexity is independent of the above bound. To achieve our result, we construct a constant-round nearly simultaneously resettable coin-flipping protocol that we believe is of independent interest.

Keywords: Reset attacks, Black-box simulation.

1 Introduction

In this work, we study the feasibility of achieving efficient zero-knowledge proof systems in the presence of physical attacks. Specifically, we examine the role of the black-box use of the code of the adversary with respect to simultaneously resettable proof systems. Such proof systems are of interest as examples of proof systems that are secure under very relaxed constraints on the re-use of the same randomness in multiple executions. In the case of *resettable zero knowledge (rZK)*, a malicious verifier may cheat against an honest prover who must use the same random tape polynomially many times. Further, *resettably sound* zero knowledge constrains the randomness used by the verifier: a malicious prover may try to cheat against an honest verifier who must use the same random tape polynomially many times. The former property was introduced and instantiated by Canetti, Goldreich, Goldwasser and Micali [2]; the later property was introduced by Micali and Reyzin [3] and later instantiated by Barak, Goldreich, Goldwasser and Lindell [4]. Because rZK is a generalization of concurrent zero knowledge (cZK) [5,6,7,8,9], every rZK proof system is a cZK proof system. A question opened by [4] and resolved by [1] is *"Does there exist a resettably sound rZK proof system for all $\mathcal{NP}$?"*. [1] answered this question in the affirmative, but they required a construction with a security proof that required a non-black-box simulator strategy, which utilize the specific strategy of a cheating verifier in its specification. Currently, there is no known practical protocol that relies on a non-black-box[1] simulation strategy, while for instance there *do* exist efficient constructions for cZK and concurrent non-malleable zero knowledge that rely on black-box simulation strategies [10,11], which work against any verifier strategy.

It is proved in [4] that resettably sound black-box zero-knowledge arguments can be constructed for languages in $\mathcal{BPP}$. Instead, we study whether there exist *t-bounded resettably sound rZK proof systems with black-box simulation*, and more in general, with only a black-box use of the code of the adversary (i.e., both the simulation and the proof of soundness do not rely on non-black-box uses of the code of the adverary). Such proof systems are rZK but also allow a malicious prover to conduct at most $t(n)$ resets against an honest verifier, where t is any fixed polynomial and n is the security parameter. Such a security setting has practical applications (indeed, in [12] it has been considered for the case of e-passports) because real hardware that may be reset to break security, such as smart cards or stateless devices, have certain wear costs; after enough resets, the hardware may simply break, a simple counter may run out, or built-in battery may become depleted. Our black-box construction is also of theoretical interest, and moreover may lead to more efficient near-simultaneously rZK protocols. Indeed while all known non-black-box constructions based on standard assumptions are inefficient, there are in several cases alternative efficient black-box constructions [10,11]. Further, unlike [4], we obtain unconditional soundness, a property that is hard to achieve when the simulator is non-black-box.

We remark that constructing t-resettably sound rZK proof systems for any language $L \in \mathcal{NP}$ with black-box simulation is quickly accomplished if round complexity is allowed to be t-dependent. For any t, take any rZK proof system with black-box simulation and repeat it sequentially with independent randomness $t+1$ times; a verifier then accepts only if he accepts for each of the $t+1$ protocol runs. What we desire is to construct a t-resettably sound rZK proof system where the round complexity is *t-independent.*

1.1 Overview of Our Contribution

For all $\mathcal{NP}$ and for any polynomial t, we construct a t-resettably sound rZK proof system with black-box use only of the code of the adversary and round complexity $O(n^{\epsilon})$ for security parameter n and for any constant $\epsilon > 0$. We require

[1] We ignore controversial non-black-box extraction assumptions.

standard assumptions as the existence of enhanced trapdoor permutations and collision-resistant hash functions.

We re-examinine the rZK protocol of [2]. Their protocol involves an adaptation of the cZK construction of [6]. We first give a high-level description of the protocol of [2]. V commits to a set of n^ϵ strings of length n. Then, in the next $2n^\epsilon$ rounds, P first commits to an n-bit string and V subsequently decommits to the next string, eventually decommitting to the entire set. The protocol concludes by P giving a rWI proof that either $x \in L$ or that at least one of the strings committed by P is identical to the subsequent decommitment of V. Clearly, such a protocol is not t-resettably sound (for any $t \geq 1$), as a resetting prover can simply obtain one of V's committed strings, rewind a round and then commit to that string. However, we use this protocol as a basis to construct our protocol.

We think of the initial commitment by V to n^ϵ strings of length n as a *database*. The idea for our protocol is that V should commit to a database of $poly(n, t)$ strings of length n; then, in each of the next $2n^\epsilon$ rounds, P asks for n entries of V's database, which V then reveals. Finally, P provides a t-resettably sound, rWI proof that either $x \in L$ or that P can commit to a large constant fraction of V's database. The idea is that even if P was able to successfully ask for $tn^{\epsilon+1}$ indices of the database, P would still not know a large constant fraction of the database; in this way, the protocol will be t-resettably sound.

We overcome several challenges to accomplish such a protocol. First, we require the simulator to discover significantly more indices of V's database than a t-resetting P^* possibly could. We note that we can modify the (black-box) simulator strategy given in [2]; there, at each prover commitment phase, the simulator executes an independent look-ahead subprotocol call to t discover the string that V would decommit to. In fact, these look-ahead subprotocol calls are independent from one round to the next. We take advantage of this independence by having our simulator execute *polynomially* many look-ahead subprotocol calls for each round and proving that such a strategy produces more than half of V's database. On the other hand, we will show that for suitable parameters, a t-resetting P^* will only be able to discover at most $1/16$ of V's database except with negligible probability. Therefore, our protocol starts by V committing to a large database followed by $2n^\epsilon$ rounds where V decommits to the n (distinct) random indices in each round that P asks for. Finally, P commits to a guess of the *entire* database and proves that either $x \in L$ or that a large constant fraction of the guess correctly corresponds to V's database.

However to have a meaningful statement for the proof given by P, it seems that V should reveal the entire database, but this exposes again V to reset attacks. Therefore, a second challenge is that V will reveal a small fraction of the database and P will prove that it committed to a large portion of this fraction. The challenge of such a strategy is that a cheating V^* might skew the distribution of what it reveals to be used for the rWI proof at the conclusion of our protocol such that the simulator might not discover enough entries of the database. Therefore, we require a special coin-flipping protocol by which V reveals n uniformly random elements of its database, at which point P proves, using a t-resettably sound rWI proof, that either $x \in L$ or that P's database guess, committed to before the final proof, contains a correct guess for at least $1/4$ of the n indices that V last decommitted to. Since the n revealed indices are uniformly distributed over V's database, only if P's initial database guess matches at least $1/4$ of V's database will P be able to give a correct WI proof without using the witness for $x \in L$ (except with negligible probability).

A first attempt for such a coin-flipping protocol between left player P_L and right player P_R might be for P_L to a apply pseudorandom function family, f_*^{PRF}, with a previously committed seed, s, on input a random string sent by P_R. The string sent by P_R would be constructed by applying a $(t+1)$-wise independent hash function on the transcript thus far (P_R cannot use a pseudorandom function because P_L^* is unbounded and could distinguish the output). However, a cheating P_L^* might commit to a seed in such a way that even on random input, the output of the pseudorandom function has skewed distribution. Once can also try to modify the protocol by having P_R hash and output the pair (R, R') and send it to P_L, who then computes $f_s^{PRF}(R) \oplus R'$. However, cheating P_R^* could then simply rewind, keep R fixed and send whatever R' he wished. Instead, we solve our problem as follows: P_R hashes to obtain the triple (R, R', r'), computes a (statistically hiding) commitment to R', denoted c, using randomness r', and sends (R, c) to P_L. Then, using previous committed seed s, P_L applies f_s^{PRF} to the concatenation of R and c, which also binds the output of the pseudorandom function to R' before R' has been revealed. Finally, P_R opens the decommitment of R' to P_L, and both set the random string τ to be the sum of the output of the pseudorandom function and R'. We remark that an adversary (resetting or not) may always guess $O(\log n)$ bits of the coin-flipping output; however, since the output length is n, an adversary will have only a negligible chance of correctly guessing a constant fraction of the coin-flipping output.

A final note is that while P_R^* may construct R and R' as he wishes, it is very important that a cheating P_L^* formats his pseudorandom function outputs correctly; otherwise, upon discovering P_R's R and R' , P_L^* could simply rewind and lie about the output of f_s^{PRF}. Therefore, P_L must send a rWI proof that either $x \in L$ or the function output was formatted correctly. In this way, only in the case that P_L^* is cheating with $x \notin L$ the correct formatting will need to be assured; we can bootstrap the witness for $x \in L$ to make the rWI proof of consistency witness hiding.

A third challenge is that our coin-flipping protocol makes the larger protocol not admissible. In particular, the simulator in [2] was given in the so-called hybrid model, where a cheating V^* is somewhat constrained. Therefore to prove rZK for our protocol, we must demonstrate that our protocol is not meaningfully different enough from the protocol of [2] even though their simulator no longer applies to our setting. To accomplish this task, we prove a theorem based on the observation that the only place where the simulators might differ are where they play identically to the honest prover against the verifier but using a different witness. We therefore construct a variant of our own protocol that more explicitly models the protocol of [2] but is only rZK (and *not* bounded resettably sound). We then construct the simulator for the intermediate protocol. Finally, we prove that the existence of a simulator for this intermediate protocol implies the existence of a simulator for the protocol that we desire. We believe that such a proof strategy is of independent interest.

We note that our protocol has communication complexity that is dependent of t; finding a protocol using standard assumptions with communication complexity independent of t is an interesting open question[2], as is improving the round complexity of our construction.

1.2 Other Related Work

The notion of rZK was first introduced by [2]; later constructions of black-box rZK protocols have better round complexity. In [7] it is shown a rZK proof system with black-box simulation and $\omega(\log^2 n)$ rounds. The protocol improved the round complexity of [6], by examining a static simulator rewind schedule and showing that such a schedule produced a successful single extraction, except with negligible probability. It is not clear how to adapt such a scheduling strategy to the polynomial many successful extractions that we require. The results of [9] can also be used to construct a black-box rZK proof system. Their protocol requires $\tilde{O}(\log n)$ rounds and also build upon the protocol of [6]. The simulator strategy of [9] relies on a careful analysis of the random tapes used by the simulator throughout its run together with the oblivious simulator strategy of [7] to obtain a single successful extraction, while our approach relies on segmenting the simulator of [2] and running various of its subprotocols in parallel to obtain polynomially many successful look-aheads. Finding compatibility between the two approaches is an interesting open question.

The first resettably sound (non-black-box) zero-knowledge argument was constructed by [4]. Deng and Lin [13] constructed a zero-knowledge argument secure in a weakened notion of simultaneous resettability: both cheating prover and verifier can reset the other polynomially many times, but can only reset a particular party with a fixed random tape (e.g., an incarnation) a bounded number of times. Their protocol requires only a constant number of rounds and also required non-black-box simulation in the proof.

The construction of a simultaneously rZK argument was first provided by [1] and requires polynomial round complexity. Their protocol relies on the prover initially committing to his challenges for the extraction stage and using the resettably sound zero-knowledge argument of [4] to prove that either $x \in L$ or that the decommitted challenges are correct. Because the protocol heavily relies on the non-black-box zero-knowledge argument of [4], the simulator used for the security proof is non-black-box. Recently, it has been shown in [19] how to obtain a constant-round resettably sound resettable witness indistinguishable argument of knowledge.

We note that all the protocols listed here are in the standard model. In particular, [2,4,13,14,15,16] also provide constructions in the bare public-key model.

[2] Using less standard assumptions like complexity leveraging, constructing a protocol t-independent communication complexity seems to be more easily to accomplish.

In addition, [17] give a resettable black-box *statistical* zero-knowledge proof for several non-trivial languages, but they do not have a construction for all $\mathcal{NP}$. As such research is incomparable to our work, we do not consider it here.

2 Preliminaries, Definitions and Tools

We denote by n the security parameter throughout our discussion., by $[m]$ the set $\{1, ..., m\}$, by $x|y$ the concatenation of x and y, by U_n the uniform distribution over $\{0,1\}^n$ and by f_*^{PRF} a pseudorandom function family, where $f_s^{PRF}(x)$ is the evaluation of the function specified by seed s at x.

We will denote by C the PPT committer and by R the PPT receiver. We use the standard notions of statistically (respectively, computationally) binding and computationally (respectively, statistically) hiding. When statistical hiding or binding is discussed for a commitment scheme, the not mentioned property is assumed to hold with computational security.

We denote by $(P_1(x), P_2(y))$ the interactive protocol between party P_1 with input x and party P_2 with input y; moreover, we denote the sequential composition of protocols $\pi_i = (P_1^i, P_2^i)$ and $\pi_j = (P_1^j, P_2^j)$ by $(\pi_i, \pi_j) = ((P_1^i, P_1^j), (P_2^i, P_2^j))$.

We refer the reader to [2] for the definition of rZK (and witness indistinguishable) proof systems, and the definition of admissible proof systems as well as the hybrid model. Our definition of t-bounded resettable soundness follows from the definition in [4] for resettable soundness, except that a malicious prover P^* has a bound of $t(n)$ many resets he can execute against a verifier V. We omit the formal definition here due to lack of space.

We will utilize the following construction[3] of Dwork and Naor [18].

Theorem 1 (zaps). *If enhanced trapdoor permutations exist, then for every language L there exists a two-round simultaneously resettable WI proof system.*

3 Black-Box rZK with t-Resettable Soundness

Before giving the exact protocol specification for our candidate construction $\Pi = (P, V)$, we first outline its crucial steps. We consider our protocol as the composition of three subprotocols, π_0, π_1, and π_2, for two reasons. The first reason is that the purpose of each of the subprotocols is distinct and so discussing them separately is natural. The second reason is that in order to prove rZK of Π, we will construct another protocol that will rely on the first two subprotocols but will require a different third subprotocol, π_2'. In what follows, fix a language $L \in \mathcal{NP}$, let n be the security parameter, let $\epsilon > 0$ be any constant, and let t be the polynomial resetting bound of the prover.

[3] In fact, zaps are not inherently resettable WI, though they are resettably sound, as noted by [4]. However, when the prover's zap message is computed using a random tape that is a pseudorandom function applied to V's initial message and P's random tape, as is done here, zaps are rWI. We will therefore refer to zaps here and implicitly assume that their instantiation in our protocol constructions utilize the appropriate PRF-random tape construction.

Table 1. Outline of protocol $\Pi = (\pi_0, \pi_1, \pi_2)$. CHCom is statistically binding, while SHCom is statistically hiding.

$\pi_{\mathbf{0}}$:	*Setup Phase* 1) P sets up SHCom 2) V constructs DB, $\lvert DB \rvert = 16tn(n^\epsilon + 1)$ 3) V sets up zap, V sends $\mathsf{SHCom}(DB)$
$\pi_{\mathbf{1}}$:	*Extraction Phase* 1) n^ϵ Iterations: i) P asks V for n indices of DB ii) V decommits to the n indices
$\pi_{\mathbf{2}}$:	*rWI Proof Phase* 1) P guesses DB as γ; P sends $\mathsf{PBCom}(\gamma)$ 2) P and V jointly generate n random indices, τ, of DB (coin flipping) 3) V decommits to the indices τ of DB 4) P sends zap for "$x \in L$ or γ agrees with at least 1/4 of the n values of DB in positions τ"

Table 2. Outline of nearly resettable coin-flipping subprotocol with output τ. f_*^{PRF} is a pseudorandom function family. (P executes P_L and V executes P_R.)

Coin Flipping Subprotocol
1) P_L sends $\mathsf{CHCom}(s)$ to P_R
2) a) P_R applies $(t+1)$-wise independent hash function h to transcript to obtain (R, R', r') b) P_R computes $c \leftarrow \mathsf{SHCom}(R')$ using randomness r' c) P_R sends R, c to P_L
3) P_L computes $r \leftarrow f_s^{PRF}(R \vert c)$, and sends r to P_R
4) P_L sends zap for "$x \in L$ or r formatted correctly"
5) P_R decommits R'
6) P_L and P_L output $\tau = r \oplus R'$

For subprotocol π_0, P and V instantiate the proof system. P sends the setup message for a 2-round statistically hiding, computationally binding commitment scheme. V then constructs an ordered database, DB, consisting of $16tn(n^\epsilon + 1)$ random distinct strings of length n, and sends a statistically hiding commitment of DB to P. V also sends the setup message used by P to execute zap proofs. At this point, P applies a pseudorandom function ($f_*^{P,1}$with seed chosen using P's initial random tape) to the current transcript and uses the output as his random tape for the rest of the protocol.

For subprotocol π_1, for each of the n^ϵ sequential iterations, P asks for a random sequence of n entries of DB, which V then decommits to. Note that for this subprotocol, a resetting P can discover at most $tn^{1+\epsilon}$ entries of DB. We note that this protocol is very similar to the protocol in [2]; where their protocol requires $O(n)$-length (random) P commitments, our protocol requires $O(n \log n)$-length random index requests, where both protocols require corresponding V decommitments.

For subprotocol π_2, P guesses V's database and commits to the guess (which we call γ) using a non-interactive perfectly binding commitment scheme. P and V then attempt to jointly compute an $(n \log \lvert DB \rvert)$-length random string as follows: P commits to a seed s using a non-interactive perfectly binding commitment scheme. V uses a $(t+1)$-wise independent hash function h with input the transcript (of π_1 and π_2 thus far) to output a random triple (R, R', r'). V then computes c, a statistically hiding commitment to R' using randomness r', and sends R and c to P. P sends back, using the PRF family $f_*^{P,2}$, $r = f_s^{P,2}(R \vert c)$. P proves using a zap that either $x \in L$ or that r is properly formed from R,

c and the commitment of s. Note that P's commitment to s earlier in π_2 can now be viewed as a partial commitment to a random string. V then decommits to R' and sets $\tau = R' \oplus r$ as the set of n indices of DB. V decommits to the n indices of DB corresponding to τ. Therefore, a resetting P^* can discover at most $tn(n^\epsilon + 1)$ entries, or 1/16 of DB, including protocol π_1. P provides a zap that either $x \in L$ or that 1/4 of the entries of P's guess γ corresponds to the final n decommitments from DB (that correspond to τ). We denote this language by Λ_2. We note that only in the case of $x \notin L$ we have that the property that τ is distributed randomly is important; we use this fact to "bootstrap" well-formedness of τ in the unbounded, cheating P^* case.

Theorem 2. *Assuming the existence of enhanced trapdoor permutations and collision-resistant hash functions, protocol Π is a t-bounded resettably sound rZK proof system for L.*

The proof that Π is rZK will follow from proofs that Π is t-resettably sound and complete (Lemma 3), that (π_0, π_2) is rWI (Theorem 4) and that there exists a specific, simpler protocol Π' that is also rZK (Theorem 6). We will then prove that since Π' is both rZK and sufficiently similar to Π, in a manner we define as *near compatible*, Π is also rZK (Lemma 5).

Lemma 3. *Π is t-resettably sound and t-resettably complete for L.*

Completeness follows from the completeness of the zap protocols. Resettable soundness follows from the rWI of zap proofs and the security of the coin-tossing protocol; since P^* cannot discover 1/4 of the database DB except with negligible probability (due to the statistical hiding property of V's commitment scheme), P^* cannot find a correct witness for language Λ_2 except with negligible probability due to the distribution of the coin-tossing protocol output.

In what follows, we will need that the sequential composition (π_0, π_2) is rWI for languages Λ_2 and L.

Theorem 4. *Assuming the existence of enhanced trapdoor permutations and collision-resistant hash functions, protocol (π_0, π_2) is rWI for Λ_2 and for L.*

For lack of space, we omit the proof of Theorem 4. The intuition for the proof is that rWI holds due to the rWI of zap proofs, the security of the respective commitment schemes, and the security of the coin-flipping protocol.

3.1 From a rZK Proof System to a New rZK Proof System

We now outline how we prove rZK of protocol Π by constructing another protocol where rZK is easier to prove. We note that this definition may likely be generalized, but we only detail properties that will apply in our case for simplicity. It is important to note that the "simpler" rZK protocol does not need to be t-resettably sound; since the purpose here is to prove rZK, resettable soundness is not required. Due to lack of space, we omit here the precise definition of *near-compatible* protocols.

The idea of our transformation stems from the idea of constructing a rZK proof system for L from a rWI proof system; see the constructions of [2,7,9]. What generally occurs is that first the setup of the rWI proof is executed; then a so-called extraction protocol is executed, where a cheating prover learns nothing, but a simulator learns some secret s. Finally, a rWI proof is completed for the language "$x \in L$ or the secret s has been learned"; in the specific case of [2], the "secret" was that the prover had committed to a string before the verifier had decommitted to that same string, while in our case, the prover commits to a largely correct guess of the database previously committed to by the verifier.

At a high level, we say that a protocol $\Pi = (\pi_0, \pi_1, \pi_2)$, which is the protocol that we wish to prove rZK, is near-compatible to a protocol $\Pi' = (\pi_0, \pi_1, \pi_2')$ if the following holds. Fix a language $L \in \mathcal{NP}$. Then (π_0, π_1, π_2') is *rZK* for L. (π_0, π_1, π_2) is an interactive proof for L, and (π_0, π_2) must be rWI so that it does not reveal to the verifier whether the transcript is generated by using the genuine witness of a real prover or by fake witness belonging to a simulator[4]. Finally, we wish that the extraction stage, π_1, is essential for the simulator to complete both (π_0, π_2') and (π_0, π_2) but extraneous for the honest prover.

Lemma 5. *(Informal) Fix a language L. Let (π_0, π_1, π_2) be near-compatible to (π_0, π_1, π_2'). Let (π_0, π_1, π_2') be rZK with a simulator that plays honestly for π_0 and π_2' and such that any witness extracted by the simulator is, except with negligible probability, a valid witness for (π_0, π_1, π_2) (with the same messages sent for π_0 and π_1). Then (π_0, π_1, π_2) is rZK.*

For lack of space, the formal version of Lemma 5 in omitted. The intuition for the proof of Lemma 5 is that by definition of near-compatible protocols and by the lemma statement, if simulator Sim' for (π_0, π_1, π_2') is able to extract a witness to complete the protocol, then so is simulator Sim for (π_0, π_1, π_2) that acts identically to Sim' for π_1 and honestly for π_0 and π_2. This is because both simulators act identically for the rounds where extraction occurs. Further, (π_0, π_2) being rWI implies that V^* cannot distinguish whether the transcript is generated by a real prover using a witness for $x \in L$ or by a simulator using an extracted witness.

4 An Admissible, Near-Compatible rZK Proof System

Here we outline an admissible rZK proof system that has the same initialization phase and extraction phase as protocol Π but with a simplified end stage in order to make the proof of rZK easier. In particular, (π_0, π_1, π_2') is not constructed to be t-resettably sound, and therefore the verifier can eventually reveal the entire DB.

[4] Some additional technical properties specified in the precise definition: it is enough for our purposes that the setup phase, π_0, consists of one round of messages sent by P followed by a round of messages sent by V. In order to prove the lemma, we will require security reductions that will need limited access to the prover's random tape; therefore, P's message for π_0 must be public coin.

For lack of space, we will only sketch π_2'. π_2' has the same inputs as π_2 above. π_2' also begins like π_2: first, prover commits his guess γ using a statistically binding commitment scheme and sends it to V. Then, however, V decommits the *entire* DB. P then executes a zap that either $x \in L$ or that γ corresponds to 1/4 of DB; we denote this new language by Λ_3.

Note that by construction, (π_0, π_1, π_2') satisfies the respective properties of near-compatibility. Further, (π_0, π_1, π_2') is admissible since the verifier, after its initial message, only sends decommitments.

Since (π_0, π_1, π_2') and (π_0, π_1, π_2) are near-compatible, the only remaining subtlety is to note that the witness extraction property of Lemma 5 holds. But this is indeed the case because the simulator, will extract a set of entries from V that correspond to at least 1/2 of V's DB except with negligible probability. Since, for π_2, the entries chosen for Λ_2 are selected uniformly at random from DB, if the simulator knows 1/2 of DB, then the simulator will know 1/4 of the entries selected for Λ_2 except with negligible probability.

Theorem 6. *Assuming the existence of enhanced trapdoor permutations and collision-resistant hash functions, protocol* (π_0, π_1, π_2') *is zero knowledge in the hybrid model (i.e, hZK) for L.*

Since the protocol (π_0, π_1, π_2') is hZK and already in the form needed to transform zero-knowledge proofs secure in the hybrid model to zero-knowledge proofs secure in the resettably model, Theorem 6 implies the following[5].

Corollary 7. (π_0, π_1, π_2') *is rZK for L.*

We would like to contrast the protocol (π_0, π_1, π_2') with that given in [2]. As noted in the high-level outline of Π in Section 3, the extraction stage of [2] and the subprotocol π_1 are very similar. Indeed, π_2' is a natural extension of the protocol in [2] because in both their protocol and ours, DB is revealed and the rWI proof incorporates the whole DB. In order to prove Theorem 6, we will need the fact that (π_0, π_2') is rWI for Λ_3 and for L.

Lemma 8. *Assuming the existence of enhanced trapdoor permutations and collision-resistant hash functions, then protocol* (π_0, π_2') *is rWI for* Λ_3 *and for L.*

4.1 High-Level Simulator Strategy in the Proof of Theorem 6

In [2], the high level strategy of the simulator was that it would try to "look ahead" to try to figure out the verifier's commitment ahead of time, but otherwise

[5] We note that the simulator does not change from Theorem 6 to Corollary 7. The reason is that the proof in [2] that takes a hZK protocol and proves that it is rZK does not change the simulator; rather, it proves that for every hybrid adversary there exists a corresponding adversary that however is still simulatable. In particular, if the simulator given here in the hybrid model only rewinds during π_1 and otherwise plays honestly, so does the simulator in the full rZK model.

play honestly for all other (non-extraction stage) rounds. This is also true for the simulator here: the simulator would like to discover as many DB decommitments as possible and otherwise plays honestly. The most important difference between the rZK simulator here and the simulator in [2] is that their protocol only requires 1 successful look-ahead to proceed, while our protocol requires polynomially many successful look-aheads to proceed.

To construct our simulator, we will use a nearly identical strategy as the simulator from [2] except that we will execute the individual (main-thread level) look-aheads $|DB|$ many times in parallel. Namely, the simulator Sim' in the extraction stage, π_1, attempts to discover half of DB; if this has not occurred at the end of the extraction stage, then the simulator simply aborts and fails to complete. One of the main inefficiencies of the [2] simulator is that it computes a distinct look-ahead subprotocol run (embedded in the subroutine $\mathsf{NextProverMsg}$, which then unfolds recursively, see details in [2]) at each of the n^ϵ round iterations of the extraction stage. The idea of their simulator is that if the simulator makes a distinct look-ahead subprotocol run at each round, which in turn consists of polynomially many look-ahead attempts, then except with negligible probability, the simulator will be able to extract one "secret". Since the look-ahead subprotocol success probability is independent from one round to the next, the strategy of our simulator is that instead of making one independent look-ahead subprotocol run at each round, we make $poly(n, t) = |DB|$ independent calls at each (main-thread) iteration of the extraction stage[6]. By a union bound, our simulator will also fail to extract only with negligible probability. A subtlety is that $|DB|$ successful look-aheads might not reveal as much of $|DB|$ as desired. However, because the prover messages in π_1 consist of n randomly chosen indices, V^* is unable to both complete the protocol with P/Sim' and sufficiently control the distribution of the prover messages that V^* chooses to proceed with.

We omit the full simulator specification and proof here due to lack of space.

Acknowledgments. The work of the second author is 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, 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. The work of the third author has been done while visiting UCLA and is supported in part by the European Commission through the FP7 programme under contract 216676 ECRYPT II.

[6] We make these independent calls only at the main-thread level of the recursion; in this manner, simulator run-time does not expand exponentially.

References

1. Deng, Y., Goyal, V., Sahai, A.: Resolving the simultaneous resettability conjecture and a new non-black-box simulation strategy. In: FOCS 2009, pp. 251–260 (2009)
2. Canetti, R., Goldreich, O., Goldwasser, S., Micali, S.: Resettable zero-knowledge (extended abstract). In: STOC 2000, pp. 235–244. ACM (2000)
3. 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)
4. Barak, B., Goldreich, O., Goldwasser, S., Lindell, Y.: Resettably-sound zero-knowledge and its applications. In: FOCS 2001, pp. 116–125 (2001)
5. Dwork, C., Naor, M.,, S.: Concurrent zero-knowledge. In: STOC 1998, pp. 409–418. ACM (1998)
6. 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)
7. Kilian, J., Petrank, E.: Concurrent and resettable zero-knowledge in poly-logarithmic rounds. In: STOC 2001, pp. 560–569. ACM (2001)
8. Canetti, R., Kilian, J., Petrank, E., Rosen, A.: Black-box concurrent zero-knowledge requires $\tilde{\Omega}(\log n)$ rounds. In: STOC 2001, pp. 570–579. ACM, USA (2001)
9. Prabhakaran, M., Rosen, A., Sahai, A.: Concurrent zero knowledge with logarithmic round-complexity. In: FOCS 2002, pp. 366–375. IEEE Computer Society (2002)
10. Micciancio, D., Petrank, E.: Simulatable Commitments and Efficient Concurrent Zero-Knowledge. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 644–645. Springer, Heidelberg (2003)
11. Ostrovsky, R., Pandey, O., Visconti, I.: Efficiency Preserving Transformations for Concurrent Non-Malleable Zero Knowledge. In: Micciancio, D. (ed.) TCC 2010. LNCS, vol. 5978, pp. 535–552. Springer, Heidelberg (2010)
12. Blundo, C., Persiano, G., Sadeghi, A.R., Visconti, I.: Improved Security Notions and Protocols for Non-Transferable Identification. In: Jajodia, S., Lopez, J. (eds.) ESORICS 2008. LNCS, vol. 5283, pp. 364–378. Springer, Heidelberg (2008)
13. Deng, Y., Lin, D.: Instance-Dependent Verifiable Random Functions and Their Application to Simultaneous Resettability. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 148–168. Springer, Heidelberg (2007)
14. 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. Crescenzo, G., Persiano, G., Visconti, I.: Improved Setup Assumptions for 3-Round Resettable Zero Knowledge. In: Lee, P.J. (ed.) ASIACRYPT 2004. LNCS, vol. 3329, pp. 530–544. Springer, Heidelberg (2004)
16. 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)
17. Garg, S., Ostrovsky, R., Visconti, I., Wadia, A.: Resettable Statistical Zero Knowledge. In: Cramer, R. (ed.) TCC 2012. LNCS, vol. 7194, pp. 494–511. Springer, Heidelberg (2012)
18. Dwork, C., Naor, M.: Zaps and their applications. In: FOCS 2000, pp. 283–293. IEEE Computer Society (2000)
19. Cho, C., Ostrovsky, R., Scafuro, A., Visconti, I.: Simultaneously Resettable Arguments of Knowledge. In: Cramer, R. (ed.) TCC 2012. LNCS, vol. 7194, pp. 530–547. Springer, Heidelberg (2012)