Resettable Cryptography in Constant Rounds – the Case of Zero Knowledge

Yi Deng[†] Dengguo Feng[♯] Vipul Goyal[‡] Dongdai Lin[♯] Amit Sahai[§]
Moti Yung[♮]

[†] NTU Singapore and SKLOIS, Institute of Software, CAS, China

[‡] MSR India

[♯] SKLOIS, Institute of Software, CAS

[§] UCLA

[♮] Google Inc., USA

Abstract

A fundamental question in cryptography deals with understanding the role that randomness plays in cryptographic protocols and to what extent it is necessary. One particular line of works was initiated by Canetti, Goldreich, Goldwasser, and Micali (STOC 2000) who introduced the notion of resettable zero-knowledge, where the protocol must be zero-knowledge even if a cheating verifier can reset the prover and have several interactions in which the prover uses the same random tape. Soon afterwards, Barak, Goldreich, Goldwasser, and Lindell (FOCS 2001) studied the setting where the *verifier* uses a fixed random tape in multiple interactions. Subsequent to these works, a number of papers studied the notion of resettable protocols in the setting where *only one* of the participating parties uses a fixed random tape multiple times. The notion of resettable security has been studied in two main models: the plain model and the bare public key model (also introduced in the above paper by Canetti et. al.).

In a recent work, Deng, Goyal and Sahai (FOCS 2009) gave the first construction of a *simultaneous* resettable zero-knowledge protocol where both participants of the protocol can reuse a fixed random tape in any (polynomial) number of executions. Their construction however required $O(n^{\epsilon})$ rounds of interaction between the prover and the verifier. Both in the plain as well as the BPK model, this construction remain the only known simultaneous resettable zero-knowledge protocols.

In this work, we study the question of round complexity of simultaneous resettable zero-knowledge in the BPK model. We present a *constant round* protocol in such a setting based on standard cryptographic assumptions. Our techniques are significantly different from the ones used by Deng, Goyal and Sahai.

1 Introduction

A fundamental question in cryptography deals with understanding the role that randomness plays in cryptographic protocols and to what extent it is necessary. Progress on this question was made relatively early with the result of Goldreich and Oren [GO94] showing that zero knowledge protocols cannot exist in the setting where the parties do not have access to any randomness resource at all. While this work showed that randomness cannot be completely eliminated, it simultaneously motivated several natural questions studying the "extent" to which randomness

is necessary. A rich line of work deals with studying the usage of imperfect randomness in various settings (see [KLRZ08, DOPS04] and the references therein). Another line of work (and the one dealt with in this paper) studies whether all the random choices can be made "offline" and be fixed once and for all. In other words, is it possible to design cryptographic protocols where a party can reuse the same random tape in multiple (or even all) executions?

The question of reusing randomness in cryptographic protocols was first considered in the context of zero knowledge by Canetti, Goldreich, Goldwasser, and Micali [CGGM00] who proposed the notion of *resettable zero knowledge*. In resettable zero knowledge, the zero knowledge property is required to hold even if a malicious verifier can "reset" the prover to the initial state and start a new interaction *where the prover uses the same random tape*. Canetti et al. [CGGM00] proposed constructions of resettable zero knowledge protocols based on standard cryptographic assumptions. Barak, Goldreich, Goldwasser, and Lindell [BGGL01] showed how to construct zero knowledge protocols for opposite setting (where soundness is required to hold even if the verifier uses the same random tape in multiple executions), which following Micali and Reyzin [MR01b][1] they call resettably sound (rS) zero-knowledge. Barak et. al. also showed that any resettable sound zero-knowledge protocol must make use of non-black-box simulation techniques (introduced in a breakthrough work of Barak [Bar01]).

Subsequent to these two works, a number of papers have studied the notion of resettable security primarily in the setting where *only one* of the participating parties uses a fixed random tape multiple times. Protocols have been proposed in the so called *plain model* (cf. [CGGM00, BGGL01, BLV03, DL07a, GS09]. A larger body of literature studies resettable security in the so called *bare public key* (BPK) model. In the BPK model, a (possibly adversarial chosen) public key is selected and published by the verifier(s) before any protocol interaction starts [2]. Protocol for resettable security in the BPK model were studied in [CGGM00, MR01b, ZDLZ03, CPV04, DL07b, YZ07]. A more complete account of the related works is given in a later subsection.

In a recent work, Deng, Goyal and Sahai (FOCS 2009) gave the first construction of a *simultaneous* resettable zero-knowledge protocol where both participants of the protocol can reuse a fixed random tape in any (polynomial) number of executions. Their construction was in the plain model. The construction however required n^ϵ rounds of interaction between the prover and the verifier. Even in the BPK model, the DGS construction remains the best known simultaneous resettable zero-knowledge protocol. This motivates the following question:

"Does there exist a polylogarithmic (or even constant) round simultaneous resettable zero-knowledge protocol in the BPK model?"

Our Results. In this paper, we resolve the above question by constructing a constant round protocol for simultaneous resettable zero-knowledge in the BPK model. Our main theorem is as follows.

Theorem 1.1 If there exist trapdoor permutations and collision resistant hash function families, then there exist constant-round resettably-sound resettable ZK arguments for **NP** in the BPK model.

[1] Micali and Reyzin defined resettable soundness (and other soundness notions) in what is called the bare public key model.

[2] Such a model is quite different from having a "setup assumption" where one would assume, e.g., that a trusted party ensured that the public key was chosen correctly.

We leave open the question of round complexity of simultaneous resettable zero-knowledge in the plain model. Note that every resettable zero-knowledge protocol is also concurrent zero-knowledge [CGGM00]. Hence, a breakthrough will be required to construct a protocol in the plain model which matches the round complexity of the one in the BPK model given in our paper.

Our Techniques. The techniques used in our paper are quite different from the ones used in the DGS construction [DGS09]. Here we outline the main technical problem which is required to be resolved to obtain a constant round construction of simultaneous resettable zero-knowledge in the BPK model.

The source of large round complexity in the DGS construction is the usage of recursive rewinding strategies (cf. [RK99, KP01, PRS02]) which are coupled with a novel non-black-box simulation strategy. In the BPK model however, it is indeed possible to avoid recursive rewinding because of the existence of a "long term" trapdoor associated with the public key of the verifier (which the simulator can try to extract). At a high level, our protocol in the BPK would follow the following structure. The verifier would first prove knowledge of a long term trapdoor associated with the public key using a zero-knowledge protocol. The prover would then give a witness indistinguishable argument of knowledge (WIAOK) proving either $x \in L$ or that it "knows" such a trapdoor. Very roughly, now once the simulator extracts a long term trapdoor for a public key, it never needs to rewinds a session with that public key (and the simulation can be done straight line). This would lead to a much simpler rewinding strategy avoiding large round complexity.

The key problem that arises while implementing the above approach in the simultaneous resettable setting is that obtaining a WIAOK protocol from the prover to the verifier is non-trivial and quite complex (since an adversarial verifier may rewind the prover to extract the witness). Instead, we would like to resort to using ZAPs [DN00] which are two round WI protocol (and hence already "secure" in the simultaneous resettable setting). Using a ZAP leads to the following problem. To arrive a contradiction in the proof of (resettable) soundness, the prover should be forced to prove a false statement about the trapdoor of the verifier (since we are not using an argument of *knowledge* protocol). This is turn means that the theorems the verifier proves about its long term trapdoor must also be false (this is important for the proof of resettable zero-knowledge to go through). However note that statements about the same public key (and the long term trapdoor) are being proven by the verifier in multiple sessions. To simulate its proof in all of those sessions, it seems that the verifier will need to use a (constant round) concurrent zero-knowledge protocol!

To overcome this problem, the verifier needs to be able to prove different statements in different sessions *with the same public key* such that some of them could be false while the others are true. This might suggest that the witness (containing the trapdoor) used by the verifier in each session is different. Yet we need that once we extract a trapdoor for any of these sessions, it should be a long term trapdoor which should enable the simulator to simulate every session with this public key (including even future sessions). Our protocol uses a careful technique to resolve this tension between "using sufficiently different witnesses in each session" and yet having "a common long term trapdoor binding them all". Our full protocol is described in Section 3.

Related Work. Subsequent to the works of Canetti et al. [CGGM00] and Barak et al. [BGGL01] described above, a number of works have investigated the problem of security against resetting attacks for zero-knowledge protocols in the plain model. Barak, Lindell, and Vadhan [BLV03] constructed the first constant-round public-coin argument that is *bounded* resettable zero-knowledge. Deng and Lin [DL07a] showed a zero-knowledge argument system that is bounded resettable zero-knowledge and satisfies a weak form of resettable soundness.

A larger body of work has investigated the same problems in a relaxed setting, called the "bare public key" (BPK) model, introduced by [CGGM00], which assumes that parties must register (arbitrarily chosen) public keys prior to any attack taking place. [CGGM00] presented a constant-round resettable zero-knowledge argument in the BPK model, the round complexity of which was improved by Micali and Reyzin [MR01b]. Micali and Reyzin [MR01b] also first investigated different notions of soundness in the BPK model, including the notion of resettable soundness. Di Crescenzo, Persiano, and Visconti [CPV04] described a resettable zero-knowledge protocol with concurrent soundness, and Deng and Lin [DL07b] improved the computational assumptions needed to obtain this result. Yung and Zhao [YZ07] also construct resettable zero-knowledge and concurrently sound arguments in the BPK model, using a general and efficient transformation. Micali and Reyzin [MR01a] also proposed a stronger variant of the BPK model for constructing bounded-secure protocols, and provided constant-round bounded resettable zero-knowledge arguments in this model; this result was strengthened by Zhao et al. [ZDLZ03] also in a bounded setting for resettable zero knowledge.

Goyal and Sahai [GS09] study the notion of general resettable two-party and multi-party computation and presented general feasibility results when only one of the parties may be reset. In this work, we restrict ourselves to the study of the zero-knowledge functionality.

Rest of this paper. We provide some basic definitions in section 2. In section 3, we construct a constant-round resettably-sound *concurrent* ZK arguments for NP in the BPK model. At last, we apply the transformation of Deng, Goyal and Sahai [DGS09] to the protocol constructed in section 3 to obtain our main result.

2 Definitions

Notation. We abbreviate probabilistic polynomial time as PPT. A function $f(n)$ is said to be negligible if for every polynomial $q(n)$ there exists an N such that for all $n \geq N$, $f(n) \leq 1/q(n)$. If L is a language in NP, we define the *associated relation* as the relation $R_L = \{(x, w) \,|\, x \in L; w \text{ is a witness for } `x \in L'\}$.

Interactive Arguments in the BPK Model. The bare public-key model (BPK model) assumes that:

- A public file F that is a collection of records, each containing a verifier's public key, is available to the prover.
- An (honest) prover P is an interactive polynomial-time algorithm that is given as inputs a secret parameter 1^n, a n-bit string $x \in L$, a witness w for $x \in L$, a public file F and a random tape r.
- An (honest) verifier V is an interactive polynomial-time algorithm that works in two stages. In stage one (key registration stage), on input a security parameter 1^n and a random tape

r, V generates a key pair (pk, sk) and stores pk in the file F. In stage two (proof stage), on input sk, an n-bit string x and a random string ρ, V performs the interactive protocol with a prover, and outputs "accept x" or "reject x".

Definition 2.1 (Complete Interactive Arguments in the BPK Model) *We say that the protocol $< P, V >$ is complete for a language L in $\mathcal{NP}$, if for all n-bit string $x \in L$ and any witness w such that $(x, w) \in R_L$, the probability that V interacting with P on input w, outputs "reject x" is negligible in n.*

Malicious Resetting Provers in the BPK model. Let s be a positive polynomial and P^* be a PPT algorithm on input 1^n.

A resetting attack by a s-resetting malicious prover P^* in the BPK model is defined as the following process:

- Run the key generation stage of V on input 1^n and a random string r to obtain pk and sk. P^* obtains pk and V stores the corresponding sk.
- Choose $s(n)$ random string ρ_i, $1 \leq i \leq s(n)$, for V.
- P^* is allowed to initiate any (polynomial) number of sessions with each verifier and interact with it in the second stage (proof stage) of the protocol. The i-th verifier uses input sk, ρ_i.

Definition 2.2 (Resettably sound arguments in the BPK model) $< P, V >$ *satisfies* resettable soundness *for an NP language L in the BPK model if for all positive polynomial s, for all s-resetting malicious prover P^*, the probability that in an execution of resetting attack, P^* ever receives "accept x" for $x \notin L$ from any of these oracles is negligible in n*

Malicious Resetting/Concurrent Verifiers in the BPK model. A resetting attack by a (s, t)-resetting malicious PPT verifier V^*, for any two positive polynomials s and t, can be defined as the following process:

- In the key generation stage, on input 1^n, V^* receives s instances $x_1, ..., x_{s(n)} \in L$ of length n each, and, outputs an arbitrary public file F
- Choose $r_1, ..., r_{s(n)}$ for P uniformly at random.
- In proof stage, V^* starts in the final configuration of the key generation stage, is given oracle access to $s^3(n)$ provers, $P(x_i, w_i, pk_j, r_k, F)$, $1 \leq i, j, k \leq s(n)$.
- V^* finally outputs its entire view of the interaction (i.e., its random tape and the messages received from the provers). The total number of steps of V^* in both stages is at most $t(n)$.

The concurrent attack by V^* is defined in the same way except that we choose s^2 random tapes $r_{i,j}, 1 \leq i, j \leq s$, and V^* is allowed to interact with s^2 provers $P(x_i, w_i, pk_j, r_{i,j}, F)$ $(1 \leq i, j \leq s)$ concurrently. Note that here each random tape is used only once.

Definition 2.3 (Resettable zero-knowledge in the BPK model) $< P, V >$ *is (non-black-box) resettable zero knowledge for an NP language L in the BPK model if for every pair of positive polynomials (s, t), for all (s, t)-resetting malicious verifier V^*, there exists a simulator S, given as input the description of V^*, such that for every $x_1, ..., x_{s(n)} \in L$, the following two distributions are computational distinguishable:*

1. *The output of V^* at the end of a resetting attack described above,*
2. *The output of $S(V^*, x_1, ..., x_{s(n)})$.*

Definition 2.4 (Concurrent zero-knowledge in the BPK model) *$< P, V >$ is (non-black-box) concurrent zero-knowledge for an NP language L in the BPK model if for every pair of positive polynomials (s, t), for all (s, t)-concurrent malicious verifier V^*, there exists a simulator S, given as input the description of V^*, such that for every $x_1, ..., x_{s(n)} \in L$, the following two distributions are computational distinguishable:*

1. *The output of V^* at the end of a concurrent attack described above,*
2. *The output of $S(V^*, x_1, ..., x_{s(n)})$.*

3 Constructing Resettably-Sound *Concurrent* Zero Knowledge Arguments for NP in the BPK Model

As a first step towards obtaining a simultaneous resettable zero-knowledge protocol, we present a resettably-sound *concurrent* zero knowledge argument for an NP language in the BPK Model in this section. We will later show how to use a compiler described in [DGS09] to obtain our main theorem.

Let (G, E, D) be a semantically secure public-key encryption scheme, where G, E, and D denote key-generation algorithm, encryption algorithm, and decryption algorithm respectively. The commitment scheme Com is a statistically binding and computationally hiding commitment scheme. $Com(s, r)$ denotes the commitment to a string s using the random tape r. The protocol proceeds as follows.

The resettably-Sound Concurrent ZK Argument (P, V) in the BPK model

The key registration stage: V runs the key generation algorithm G of a semantically secure public key encryption scheme (G, E, D) twice independently, $(pk_0, sk_0) = G(1^n, r_0^k)$, $pk_1, sk_1) = G(1^n, r_1^k)$, publishes (pk_0, pk_1) and stores r_b^k and sk_b for a random $b \in \{0, 1\}$.

The proof stage (main protocol):

Common input: x (supposedly in L) and verifier's public key (pk_0, pk_1).

P's private input: the witness w such that $(x, w) \in R_L$.

V's private input: the randomness r_b^k used in key generation for one the public keys

P's randomness: r_p.

V's randomness: r_v.

1. P sends a commitment $c = Com(e, r)$ to a random challenge e.
2. V Compute two ciphertexts of 0 under pk_0 and pk_1 independently, $c_0 = E(pk_0, 0, r_0)$, $c_1 = E(pk_1, 0, r_1)$; Send c_0, c_1 and the first message a of the 3-round WI proof of Hamiltonian Cycle for the following statement:
   (a) there exists r_b^k such that $(pk_b, sk_b) = G(1^n, r_b^k)$ (equivalently, "I know one of secret keys"); and,

(b) there exist r_0 and r_1 such that $c_0 = E(pk_0, 0, r_0)$ and $c_1 = E(pk_1, 0, r_1)$ (i.e., both cipertexts are encryption of 0).

The randomness used by V in this step as well as the rest of the protocol is generated by applying a pseudorandom function f_{r_v} to the first message c of the prover.

3. P sends e and execute the BGGL protocol in which P proves that either: 1) there exists r such that $c = Com(e, r)$, or, 2) $x \in L$.

4. V now responds to the challenge e by sending the final message z of the 3-round WI protocol of Hamiltonian Cycle.

5. P executes a ZAP in which P proves that either $x \in L$ or there exists r_d^k, $d \in \{0, 1\}$, such that $(pk_d, sk_d) = G(1^n, r_d^k)$ and $0 = D(sk_d, c_d)$ (i.e., one of the decryptions result to the message 0).

Remark 1. For simplicity of presentation, we view *com* and ZAPs as non-interactive protocol requiring only one message in each direction. However our construction can indeed use two round protocols for each in a straight-forward way.

Remark 2. Note that there is fine difference between the verifier and the prover in proving a ciphertext is an encryption of 0: the verifier uses the knowledge of randomness in encryption to prove the ciphertext is an encryption of 0, while the prover uses the knowledge of the secret key (more precisely, randomness that used to generate the public/secret key pair) to prove that one plaintext is actually 0. We stress that this difference is crucial for security proof. In the course of simulation, once our simulator extracts the randomness used for generating one of pk_0 and pk_1 (note that it does not need the randomness used in these encryptions by the verifier to execute a session), it can handle all sessions under the same public key (pk_0, pk_1). On the other hand, in the proof of soundness, the reduction algorithm, playing the role of verifier, needs only one of secret keys to execute a session, and this will enable it to use the power of cheating prover to either break the semantic security of the other public key scheme or break the WI property of the underlying 3-round WI protocol if such a cheating prover exists.

We now state the following theorem.

Theorem 3.1 The above protocol (P, V) is a resettably-sound concurrent zero knowledge argument.

The completeness is obvious. We will prove concurrent zero knowledge and resettable-soundness in next two subsections.

Hardness assumption. Note that the 2-round statistically-binding commitment scheme and semantically secure public key encryption scheme can be based on trapdoor permutations, which also imply the existence of ZAPs. In addition, we need to assume collision-resistant hash functions required for the resettably sound BGGL protocol (which makes use of non-black-box simulation techniques). Thus we can base the above resettably-sound concurrent ZK argument on the assumption of existence of trapdoor permutations and collision-resistant hash function families.

3.1 Proof of Concurrent Zero-Knowledge

Let V^* be an concurrent malicious verifier. Assume w.l.g. in real world, on input a fixed YES instance sequence $x_1, ..., x_{s(n)} \in L$ of length n each, V^* generates s public keys $F = ((pk_0^1, pk_1^1), ..., (pk_0^s, pk_1^s))$, and interacts with $s^2(n)$ incarnations of prover, $P(x_i, w_i, (pk_0^j, pk_1^j), r_{i,j}, F)$, $1 \leq i, j \leq s(n)$. We now construct a simulator S as required by definition 2.3.

S operates as follows. First, given a fixed YES instance sequence $x_1, ..., x_s \in L$ of length n each as input, S runs the key-generation phase of V^* to obtain the public file F.

In proof stage, the first task of S is to extract one r_b^k ($b \in \{0, 1\}$) for each public key pair (pk_0^j, pk_1^j) such that r_b^k is the randomness used for generating one public key pk_b^j. Note that once these r_b^k's are obtained, S is able to carry out all sessions successfully in a straight-line manner by decrypting one of two ciphetexts (and relying on the soundness of the WI protocol). We say a session under public key (pk_0^j, pk_1^j) is *solved* if S already extracted the corresponding randomness r_b^k; otherwise, we say it is *unsolved*.

The extraction is done in a sequential way. Once receiving an accepting execution of the 3-round WI protocol in an *unsolved* session under public key (pk_0^j, pk_1^j), S rewinds to the beginning of step 3, sends a random challenge e' and runs the simulator for BGGL protocol to prove that c is a commitment to e'. When another accepting execution of this subprotocol is obtained, S solved all sessions under this public key.

We would like to make the following remarks on the above extraction:

- The non-black-box simulator for the *standalone* BGGL protocol handles only a single session, but it runs in a concurrent setting. This means, during the execution of this subprotocol, many other sessions may appear. To deal with this issue, we have the following strategy. First observe that all the other sessions are being executed honestly by the simulator (and the current rewinding thread will be aborted if an unsolved session reaches its final prover message). Thus, we consider these sessions (and the part of the simulator handling these sessions) as part of the adversarial machine itself. Then our modified non-black-box simulator Sim will now simply act on this new machine (by using its code) instead of the original one.

- For the analysis of running time to go through, we use the Goldreich-Kahan technique to bound the running time of S.

The detailed description of S follows.

The Simulator S:

Input: the code of V^*, s YES instances $x_1, ..., x_s$.

1. select a random tape for V^*, and run the key-generation phase of V^* to obtain the public file $F = ((pk_0^1, pk_1^1), ..., (pk_0^s, pk_1^s))$.

2. Set $h \leftarrow (x_1, ..., x_s)$ and $\mathcal{S} \leftarrow \emptyset$.

3. Do the following:

   (a) Adopt the honest prover strategy until the final ZAP in every session, and extend h to include the transcript generated in this step. If V^* terminates during this step, return h; Otherwise, go to next step.

(b) If a *solved* session reaches the final ZAP, use the relevant randomness and secret key to produce a prover message of the final ZAP, and extend h to include this message. If V^* terminates during this step, return h; Otherwise, go to next step.

(c) If an *unsolved* session reaches the end of of the underlying 3-round WI protocol, and the resulting transcript (a, e, z) so far is accepting, do the following:

- (**Estimation**) Suppose that the first two messages sent in the current session are $c, (c_0, c_1, a)$, and the corresponding public key is (pk_0^j, pk_1^j). Rewind P^* to the point (we call it **rewinding point**) where the verifier's message (c_0, c_1, a) was just sent, and repeat the following until it receives the accepting transcript (a, e, z) of the underlying 3 round WI argument n^2 times: send the honest challenge e and choose *independent randomness* to execute the underlying BGGL protocol honestly; when another unsolved session reaches the final ZAP, S aborts the current thread[3].
  We denote by X the total number of iterations (or threads) of this step.
- (**Extraction**) Rewind V^* to the above **rewinding point** again, and repeat the following until it obtains another accepting transcript (a, e', z') with $e \neq e'$ until the $X + 1^{st}$ iteration is reached. If all iterations fails, output "$\perp$".
  - For the current session, S send a new random challenge $e' \neq e$, and then runs the non-black-box simulator Sim to prove that c is a commitment to e', where Sim proceeds exactly the same as the simulator for the BGGL protocol (except for acting on the new adversarial machine as described earlier).
  - For any other solved session, S executes the strategy described in step b; if an unsolved session reaches the final ZAP, S aborts the current iteration.

(d) From the two accepting transcripts of the 3-round WI protocol (a, e, z) and (a, e', z'), compute the randomness r_b^k such that $(pk_b^j, sk_b^j) = G(1^n, r_b^k)$,[4] and update $\mathcal{S}$ to include r_b^k, and go to step 1. (Note that the above step 3(c) does not update history).

The concurrent zero knowledge property of our protocol follows from the following claims.

Claim 1 S runs in expected polynomial time.

Claim 2 The output h by S is indistinguishable from real interaction.

Proof of Claim 1. We first count the number of queries which the simulator makes to the adversary. Observe that the number of queries which S makes in a single solved session is a constant C. Suppose that for a specific session i, S enters step 3(c) with probability p_i, then we have for this session, the expected number of iterations in step 3(c) is at most $p_i \cdot (2n^2/p_i) < 2n^2$. Since V^* is only allowed to initiate s^2 sessions, the entire simulation of S will makes an expected $s^2 \cdot C \cdot (2n^2 + 1)$ number of queries (which is polynomial). Since each query additionally requires only polynomial time, the overall running time of the simulator is expected polynomial. □

Proof of Claim 2. We first prove the probability that S outputs $\perp$ is negligible. Observe that S outputs $\perp$ only if it fails to extract a relevant secret key.

[3] in this case, S cannot proceed further without knowledge of the relevant secret key.

[4] Note that we can also compute the randomness that were used in the two encryptions to 0, but we don't need it to carry out the final ZAP.

Assume that for session i, S enters step 3(c) with probability p_i (taken over the random coins used in step 3 of the protocol; here prover proves that e is the correct challenge). We claim that in a single run of the **Extraction** in step 3(c), the probability that S obtains an accepting transcript of the 3-round WI protocol is at least $p_i - neg(n)$ for some negligible function neg (except for a negligible fraction of protocol prefixes, i.e., transcripts of steps 1 and 2), otherwise, we can use V^* to break either the computational-hiding property of the scheme Com or the zero knowledge property of the BGGL protocol.

Note that the Goldreich-Kahan technique [GK96] guarantees that, the estimation n^2/X of p_i is within a constant factor of p_i except with exponentially small probability, thus, we conclude that $X > n^2/(c \cdot p_{r_0})$ holds for some constant c except with exponentially small probability.

Thus, the probability that S enters step 3(c) but doesn't extract out the randomness used in generation of some public key is

$$\begin{aligned} & p_i(1 - p_i + neg)^X \\ \leq & p_i(1 - p_i + neg)^{n^2/(c \cdot p_i)} \end{aligned}$$

which is negligible.

Observe that the only difference between S and the honest prover is that they use different witness to carry out the final ZAP in each session. Now by the WI property of the ZAP, we conclude that h is indistinguishable from the real interaction between honest provers and V^*. $\square$

3.2 Proof of Resettable-Soundness

Assume that there is a PPT resetting P^* that can cheat an honest verifier V (and complete a protocol execution) on a NO instance x with noticeable probability p. We shall now consider the following 5 hybrid verifier strategies. We shall prove that in each hybrid, the probability of the verifier being able to cheat (in some session) is still noticeable. In the final hybrid, we note that the above cheating probability must be negligible by the soundness of the ZAP system (and thus arrive at a contradiction). We shall first describe the hybrid strategies and then argue that the probability of cheating remains negligible in each.

V_1: Follow the honest verifier strategy V, except that whenever V is instructed to applying the pseudorandom function specified by its random tape to generate randomness, V_1 uses truly random coins (while still making sure that for a given prover first message c, it always uses the same random coins).

V_2: Follow the strategy below.

1. In the key registration stage, V_2 acts exactly as V_1.
2. In the proof stage, V_2 first picks a session i at random.
   Suppose that the first prover message in session i is c, and that the public key is (pk_0, pk_1) and the secret key stored by V_2 is sk_b for some $b \in \{0, 1\}$.
3. For all sessions having a first prover message different than c, V_2 executes honest verifier's strategy throughout the entire interaction between P^* and V_2.

4. For all sessions having the first prover message c, V_2 executes honest verifier's strategy *until* when a session among them first completes an accepting proof via BGGL protocol for the correctness of challenge e, and then rewinds to the point where it received c for the first time, computes two encryptions of 0 under *both* public key pk_b and pk_{1-b} honestly again, produces a fake first massage a that can answer e successfully according to the 3-round WI protocol[5], and continue (without using the actual witness).

V_3**:** Follow the strategy of V_2 except that, in item 4 of V_2, computes an encryption of 0 under public key pk_b and an encryption of 1 under public key pk_{1-b} after extracting the challenge e and then rewinding (but produces the first message a in the same way as V_2),

V_4**:** Follow the strategy of V_3 except that, in *all sessions*, whenever V_3 needs to use r_b^k as partial witness to carry out the 3-round WI protocol, V_4 uses r_{1-b}^k.

V_5**:** Follow the strategy of V_4 except that, after rewinding, V_5 computes two encryptions of 1 under pk_0 and pk_1 respectively in those sessions having the first prover message c.

First, we have that P^* can cheat V_1 with probability negligibly close to p, due to the pseudorandomness of the pseudorandom function specified by the random tape of V.

We now prove that P^* can cheat V_2 in a session having the first prover message c with probability negligibly close to $p/poly$, where $poly$ is the total number of distinct first prover messages appeared in the whole interaction between P^* and V_2. Observe that for a randomly chosen first prover message c, P^* will cheat V_1 in a session having this first prover message with probability exactly $p/poly$, and that the only difference between the second run of V_2 and V_1 is the way in which the transcript (a, e, z) is produced. Since in the 3-round protocol for Hamiltonian Cycle, the simulated transcript (a, e, z) is computationally indistinguishable to a real one, we conclude that V_2 will accept with probability negligibly close to $p/poly$ in a session having the first prover message c.

We further claim that P^* can also cheat V_3 in a session having the first prover message c with probability negligibly close to $p/poly$. Notice that the only difference between V_2 and V_3 is, in their second run (after rewinding), V_2 encrypts to 0 under public key pk_{1-b}, while V_3 encrypts to 1 under public key pk_{1-b}. Notice also that in both their second runs, the message a is produced independently of these encryptions. Thus, if the aforementioned claim is false, we can construct an algorithm V_h to break the semantic security of the public key encryption scheme: V_h acts as V_2 except that, after rewinding, it obtains the ciphertext (that is supposed to be 0 or 1) under the public key pk_{1-b} from an external challenger, instead of computing this ciphertext itself; When P^* convinces V_h to accept in a session having the first prover message c, V_h outputs 0, otherwise, outputs 1. Observe that if the ciphertext obtained from encryption oracle is an encryption of 0, then V_h is identical to V_2; if this ciphertext is an encryption of 1, V_h is identical to V_3. Hence, in a session having the first message c, if there is a non-negligible gap between the probability that V_2 accepts and the probability that V_3 accepts, V_h breaks the semantic security of the underlying public key encryption scheme.

For strategies V_3 and V_4, we observe that the only difference between them is that they use different witnesses to carry out the 3-round WI protocol. Consider the following algorithm V_{wi}.

[5]In the 3-round WI protocol for Hamiltonian Cycle, given a challenge e, there exists a simple simulator that can produce an accepting transcript (a, e, z) efficiently.

V_{wi}**:**

1. In the key registration stage, V_{wi} generates two public keys honestly, i.e., it computes $(pk_0, sk_0) = G(1^n, r_0^k)$, $(pk_1, sk_1) = G(1^n, r_1^k)$, publishes (pk_0, pk_1), chooses a random bit b and stores *both* r_0^k and r_1^k.
2. Like V_2, V_{wi} first picks a session i at random. Again, suppose that the first prover message in session i is c.
3. For all sessions having a first prover message different than c, when a session with a distinct first prover message c' was initiated for the first time, V_{wi} executes honest verifier's strategy to compute two encryptions of 0, $c_0 = E(pk_0, 0, r_0)$ and $c_1 = E(pk_1, 0, r_1)$, send (r_0^k, r_1^k, r_0, r_1) to an independent honest prover P_{wi} of the 3-round WI protocol, and forward the P_{wi}'s first message a' along with c_0, c_1 to P^*; Once a session with the first prover message c' first completes the correctness proof via BGGL protocol for the challenge e', V_{wi} sends e' to P_{wi} and forward P_{wi}'s answer z' to P^*; in all sessions with c' as the first prover message, V_{wi} sends the same (a', c_0, c_1) to P^*, and if P^* reveals the same e' again and completes the correctness BGGL proof, V_{wi} answers with the same z'; Otherwise, V_{wi} outputs "failure".
4. When P^* sends c for the first time, V_{wi} acts the same as the above strategy: computes two encryptions of 0, sends all random tapes to an independent P_{wi} and forward P_{wi}'s first message a (and the two encryptions) to P^*. Once P^* repeats c, V_{wi} responds with the same a. When a session with the first prover message c first completes an accepting proof via BGGL protocol for the correctness of challenge e, it rewinds to the point where it received c for the first time, computes an encryptions of 0 under public key pk_b and an encryption of 1 under public key pk_{1-b}, produces a fake first massage a that can answer e successfully according to the 3-round WI protocol, and continue.

We first note that V_{wi} outputs "failure" only if P^* opens some commitment c' to two different values and gives two accepting proofs for both. Due to the statistically-binding property of the commitment scheme and resettable-soundness of the BGGL protocol, the probability that V_{wi} outputs "failure" is negligible. Note also that, each independent P_{wi} is run once (i.e., the 3-round WI protocol is executed in *concurrent* setting), and that if all these P_{wi}'s uses r_b^k (resp., r_{1-b}^k) as partial witness, then V_{wi} is identical to V_3 (resp., V_4). Note that the 3-round WI protocol is *concurrent* witness indistinguishable. Thus, we conclude that the probability that P^* cheats V_4 in a session with the first prover message c is negligibly close to $p/poly$.

Finally, notice that both V_4 and V_5 do not use knowledge of the randomness r_b^k (used in generation the public/secret key pair (pk_b, sk_b)) to carry out any session in their entire interaction, and the only difference between them is that they encrypt different messages under pk_b in sessions having the first prover message c after rewinding. Similar to the analysis of V_2 and V_3, due to the semantic security of the public key encryption scheme (pk_b, sk_b), the probability that P^* cheats V_5 in a session with the first prover message c is negligibly close to $p/poly$. However, since both ciphertexts in these sessions are encryptions of 1, by the soundness of the ZAP system, P^* can cheat V_5 in any one of these sessions only with negligible probability. Thus we have p is negligible.

4 Simultaneous Resettable Zero-Knowledge Arguments for NP in the BPK model

In this section, we apply the transformation of [DGS09] to the resettably-sound concurrent ZK arguments presented in last section, and obtain simultaneously resettable arguments for NP in the BPK model. This establishes theorem 1.1.

Given a resettably-sound concurrent ZK argument (P_{RC}, V_{RC}) for NP language L in the BPK model and a common input $x \in L$, the simultaneously resettable argument (P, V) for L proceeds as follows.

The key registration stage: V acts exactly the same as V_{RC} in the key registration stage.

The proof stage:

Common input: x (supposedly in L) and verifier's public key ver_k

P's randomness: (γ_p^1, γ_p^2)

V's randomness: (γ_v^1, γ_v^2)

1. P uses randomness γ_p^1 to generate a random string r_p (of appropriate length) and a first verifier message ρ_p of a ZAP system. P sends $C_p = Com(r_p)$ and ρ_p (where Com is a perfect binding commitment scheme).

2. V sets $(\tau_v^1, \tau_v^2) = f_{\gamma_v^1}(x, ver\_k, C_p)$. Using randomness τ_v^1 generates a first verifier message ρ_v and compute a commitment $C_t = Com(0)$ to 0. V sends ρ_v and C_t.

3. V and P execute the BGGL protocol in which V uses random tape τ_v^2 and proves that C_t is a commitment to 0. In addition, in each verifier step in this subprotocol, P generates a ZAP proof along with each verifier message for the following OR statement:

   (a) The current message is produced by an honest verifier of the BGGL protocol using random tape r_p, or,

   (b) $x \in L$

4. V sets $(\tau_v^3, \tau_v^4) = f_{\gamma_v^2}(hist)$, where $hist$ is the history so far except those ZAP proofs. Using randomness τ_v^3, V sends a commitment $C_v = Com(\tau_v^3)$ to P. In the remaining steps, V uses randomness τ_v^4.

5. P sets $\tau_p = f_{\gamma_p^2}(hist)$. Using random tape τ_p, P and V execute (P_{RC}, V_{RC}) in which P proves $x \in L$, except that for every V_{RC}'s message, we have V give an additional ZAP proof for the following OR statements:

   (a) the current message is produced by an honest verifier of (P_{RC}, V_{RC}) using random tape τ_v^3, or,

   (b) C_t is a commitment to 1.

   V accepts if only if V' accepts the transcript of (P_{RC}, V_{RC}).

Remark. In [DGS09], the actual transformation of resettably-sound concurrent ZK argument into a resettably-sound resettable ZK argument takes two steps: 1) transform the resettably-sound concurrent ZK argument into a *hybrid* sound *hybrid* zero knowledge argument; 2) transform a hybrid sound hybrid zero knowledge protocol into a resettably-sound resettable zero knowledge protocol. The second step is done by simply having each party refresh their randomness via a pseudorandom function. Here for the sake of simplicity and keeping the proof short, we merge these two steps into a single transformation (and refer the reader to [DGS09] for a detailed formal presentation).

Theorem 4.1 The protocol (P, V) is a resettably-sound resettable zero knowledge.

Proof sketch. The proof of this theorem is similar in spirit to the one appeared in [DGS09]. Here we just give a proof outline.
The *completeness* is obvious.

Resettable-Soundness. For a given cheating prover P^* for (P, V) and a NO instance $x \notin L$, we can construct a series of hybrid verifiers to show the cheating probability is negligible just like the hybrid verifiers V_1, V_2, V_3, V_4 and V_5 we set up in the previous section. Whenever a hybrid verifier needs to rewind in some target sessions with a specific first prover message C_p, it always computes a commitment C_t to 1 in its first step, and then runs the simulator for the BGGL protocol to prove that C_t is a commitment to 0 in all sessions having the same first prover message C_t[6]; Whenever it produces a fake first message a of the underlying 3-round WI protocol in (P_{RC}, V_{RC}), it uses the witness for "C_t is a commitment to 1" to execute ZAP for the correctness of message a. Similar to the analysis presented in previous section, it is not hard to show that, if all building blocks are secure, the above protocol (P, V) is resettably-sound.

Resettable ZK. Note that the BGGL protocol is resettably-sound, and hence for any malicious resetting verifier, if an execution of BGGL protocol in step 3 is accepting, the message C_t sent in step 2 is guaranteed to be a commitment to 0 (except with negligible probability). As a consequence, all verifier's messages sent in the subprotocol (P_{RC}, V_{RC}) are determined by the commitment C_v sent in step 4 and the session history of (P_{RC}, V_{RC}) due to the fact that ZAP is resettably-sound, that is, for a fixed session prefix until step 4, all subexecutions of (P_{RC}, V_{RC}) are *identical*. This observation enables us to adopt essentially the same simulation strategy of S which works for *concurrent* adversary and prove the property of resettable zero knowledge. Given a resetting verifier V^*, our simulator S' proceeds as follows. For all sessions, S' follows the honest prover strategy until step 4. When reaching the subprotocol (P_{RC}, V_{RC}), S' acts as the simulator S for (P_{RC}, V_{RC}). For those solved sessions, S' uses the relevant secret key as witness to carry out the final ZAP. When an unsolved session reaches the end of the 3-round WI protocol in (P_{RC}, V_{RC}), S' applies the extraction strategy of S to extract a secret key. We can perform a similar analysis and show that S' will run in expected polynomial time and its output is distinguishable from that in the real interaction. □

[6]Note that, all subexecutions of BGGL protocol in these sessions are actually *identical*, due to the resettable-soundness of ZAP and the instance x to be proven is a NO instance. This is why the simulator for BGGL protocol in the standalone setting works in this specific resettable setting.

References

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

[BGGL01] Boaz Barak, Oded Goldreich, Shafi Goldwasser, and Yehuda Lindell. Resettably-sound zero-knowledge and its applications. In *FOCS*, pages 116–125, 2001.

[BLV03] Boaz Barak, Yehuda Lindell, and Salil P. Vadhan. Lower bounds for non-black-box zero knowledge. In *FOCS*, pages 384–393, 2003.

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

[CPV04] Giovanni Di Crescenzo, Giuseppe Persiano, and Ivan Visconti. Constant-round resettable zero knowledge with concurrent soundness in the bare public-key model. In *CRYPTO*, pages 237–253, 2004.

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

[DL07a] Yi Deng and Dongdai Lin. Instance-dependent verifiable random functions and their application to simultaneous resettability. In Naor [Nao07], pages 148–168.

[DL07b] Yi Deng and Dongdai Lin. Resettable zero knowledge with concurrent soundness in the bare public-key model under standard assumption. In *Inscrypt*, pages 123–137, 2007.

[DN00] Cynthia Dwork and Moni Naor. Zaps and their applications. In *FOCS*, pages 283–293, 2000.

[DOPS04] Yevgeniy Dodis, Shien Jin Ong, Manoj Prabhakaran, and Amit Sahai. On the (im)possibility of cryptography with imperfect randomness. In *FOCS*, pages 196–205. IEEE Computer Society, 2004.

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

[GO94] Oded Goldreich and Yair Oren. Definitions and properties of zero-knowledge proof systems. *J. Cryptology*, 7(1):1–32, 1994.

[GS09] Vipul Goyal and Amit Sahai. Resettably secure computation. In Antoine Joux, editor, *EUROCRYPT*, volume 5479 of *Lecture Notes in Computer Science*, pages 54–71. Springer, 2009.

[KLRZ08] Yael Tauman Kalai, Xin Li, Anup Rao, and David Zuckerman. Network extractor protocols. In *FOCS*, pages 654–663. IEEE Computer Society, 2008.

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

[MR01a] Silvio Micali and Leonid Reyzin. Min-round resettable zero-knowledge in the public-key model. In *EUROCRYPT*, pages 373–393, 2001.

[MR01b] Silvio Micali and Leonid Reyzin. Soundness in the public-key model. In *CRYPTO*, pages 542–565, 2001.

[Nao07] Moni Naor, editor. *Advances in Cryptology - EUROCRYPT 2007, 26th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Barcelona, Spain, May 20-24, 2007, Proceedings*, volume 4515 of *Lecture Notes in Computer Science*. Springer, 2007.

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

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

[YZ07] Moti Yung and Yunlei Zhao. Generic and practical resettable zero-knowledge in the bare public-key model. In Naor [Nao07], pages 129–147.

[ZDLZ03] Yunlei Zhao, Xiaotie Deng, Chan H. Lee, and Hong Zhu. Resettable zero-knowledge in the weak public-key model. In *EUROCRYPT*, pages 123–139, 2003.