Constant-Round Resettable Zero Knowledge with Concurrent Soundness in the Bare Public-Key Model

Giovanni Di Crescenzo[1], Giuseppe Persiano[2], and Ivan Visconti[3]

[1] Telcordia Technologies, Piscataway, NJ, USA
giovanni@research.telcordia.com

[2] Dip. di Informatica ed Appl., Univ. di Salerno, Baronissi, Italy
giuper@dia.unisa.it

[3] Département d'Informatique, École Normale Supérieure, Paris, France
ivan.visconti@ens.fr

Abstract. In the bare public-key model (BPK in short), each verifier is assumed to have deposited a public key in a file that is accessible by all users at all times. In this model, introduced by Canetti *et al.* [STOC 2000], constant-round black-box concurrent and resettable zero knowledge is possible as opposed to the standard model for zero knowledge. As pointed out by Micali and Reyzin [Crypto 2001], the notion of soundness in this model is more subtle and complex than in the classical model and indeed four distinct notions have been introduced (from weakest to strongest): one-time, sequential, concurrent and resettable soundness.

In this paper we present the first *constant-round* concurrently sound resettable zero-knowledge argument system in the bare public-key model for $\mathcal{NP}$. More specifically, we present a 4-round protocol, which is optimal as far as the number of rounds is concerned. Our result solves the main open problem on resettable zero knowledge in the BPK model and improves the previous works of Micali and Reyzin [EuroCrypt 2001] and Zhao *et al.* [EuroCrypt 2003] since they achieved concurrent soundness in stronger models.

1 Introduction

The classical notion of a zero-knowledge proof has been introduced in [1]. Roughly speaking, in a zero-knowledge proof a prover can prove to a verifier the validity of a statement without releasing any additional information. In order to prove that a zero-knowledge protocol does not leak information it is required to show the existence of a probabilistic polynomial-time algorithm, referred to as *Simulator*, whose output is indistinguishable from the output of the interaction between the prover and the verifier. Since its introduction, the concept of a zero-knowledge proof and the simulation paradigm have been widely used to prove the security of many protocols. More recently, it has been recognized that in several practical settings the original notion of zero knowledge (which in its original formulation only considered one prover and one verifier that carried out the proof procedure in isolation) was insufficient. For example, the notion of *concurrent zero knowledge* [2] formalizes security in a scenario in which several verifiers access concurrently the same prover and maliciously coordinate their actions so to extract information from the prover. Motivated by considerations regarding smart cards, the notion of *resettable zero knowledge* (rZK, in short) was introduced in [3]. An rZK proof remains "secure" even if the verifier is allowed to tamper with the prover and to reset the prover in the middle of a proof to any previous state and then asks different questions. It is easy to see that concurrent zero knowledge is a special case of resettable zero knowledge and, currently, rZK is the strongest notion of zero knowledge that has been studied. Unfortunately, if we only consider black-box zero knowledge, constant-round concurrent zero knowledge is only possible for trivial languages (see [4]). Moreover, the existence of a constant-round concurrent zero-knowledge argument in the non-black-box model (see [5] for the main results in the non-black-box model) is currently an open question. Such negative results have motivated the introduction of the *bare public-key model* [3] (BPK, in short). Here each possible verifier deposits a public key `pk` in a public file and keeps private the associated secret information `sk`. From then on, all provers interacting with such a verifier will use `pk` and the verifier cannot change `pk` from proof to proof. Canetti *et al.* [3] showed that constant-round rZK is possible in the BPK model. However, the fact that the verifier has a public key means that it is vulnerable to an attack by a malicious prover that opens several sessions with the same verifier in order to violate the soundness condition. This is to be contrasted with the standard models for interactive zero knowledge [1] or non-interactive zero knowledge [6] where, as far as soundness is concerned, it does not matter whether a malicious prover is interacting once or multiple times with the same verifier.

Indeed, in [7], Micali and Reyzin pointed out, among other contributions, that the known constant-round rZK arguments in the BPK model did not seem to be sound if a prover was allowed to concurrently interact with several instances of the same verifier. In other words, the known rZK arguments in the BPK were not *concurrently sound.*

Micali and Reyzin gave in [7] a 4-round argument system which is sequentially sound (*i.e.*, the soundness holds if a prover can play only sequential sessions) and probably is not concurrently sound, and they also showed that the same holds for the five-round protocol of Canetti *et al.* [3]. Moreover they proved that resettable soundness cannot be achieved in the black-box model. In [8], Barak *et al.* used non-black-box techniques in order to obtain a constant-round rZK argument of knowledge but their protocol enjoys only sequential soundness.

In order to design a concurrently sound resettable zero-knowledge argument system, Micali and Reyzin proposed (see [9]) the upper bounded public-key (UPK, in short) model in which a honest verifier possesses a counter and uses the same private key no more than a fixed polynomial number of times. A weaker model than the UPK model but still stronger than the BPK model is the weak public-key (WPK, in short) model introduced in [10]. In this model an honest verifier can use the same key no more than a fixed polynomial number of times for each statement to be proved.

Other models were proposed in order to achieve constant-round concurrent zero knowledge. In particular, in [2, 11] a constant-round concurrent zero-knowledge proof system is presented by relaxing the asynchrony of the model or the zero-knowledge property. In [12] a constant-round concurrent zero-knowledge proof system is presented by requiring a pre-processing stage in which both the provers and the verifiers are involved. In [13] a constant-round concurrent zero-knowledge proof is presented assuming the existence of a trusted auxiliary string. All these models are considered stronger than the BPK model.

Our results. In this paper we present the first *constant-round concurrently* sound *resettable* zero-knowledge argument system in the BPK model for $\mathcal{NP}$. In particular we show a 4-round argument that is optimal in light of a lower bound for concurrent soundness proved in [7]. We stress that our result is the best one can hope for in terms of combined security against malicious provers and verifiers if we restrict ourselves to black-box zero knowledge, since in this setting simultaneously achieving resettable soundness and zero knowledge has been shown to be possible only for languages in BPP by [7]. Our construction employs the technique of *complexity leveraging* used in the previous results [3, 7, 10] in order to prove the soundness of their protocols and is based on the existence of a verifiably binding cryptosystem semantically secure against subexponential adversaries. The existence of cryptographic primitives secure against subexponential adversaries is used also in [3, 7, 10] and the existence of a constant-round black-box rZK argument system in the BPK model assuming only cryptographic primitives secure against polynomial-time adversaries is an interesting open question.

Finally, we describe a simple 3-round sequentially sound and sequential zero-knowledge argument system in the BPK model for all $\mathcal{NP}$.

2 Definitions

The BPK model. The Bare Public-Key (BPK, in short) model assumes that:

1. there exists a public file F that is a collection of records, each containing a public key;
2. an (honest) prover is an interactive deterministic polynomial-time algorithm that takes as input a security parameter 1^n, F, an n-bit string x, such that $x \in L$ and L is an NP-language, an auxiliary input y, a reference to an entry of F and a random tape;
3. an (honest) verifier V is an interactive deterministic polynomial-time algorithm that works in the following two stages: 1) in a first stage on input a security parameter 1^n and a random tape, V generates a key pair $(\mathtt{pk}, \mathtt{sk})$ and stores $\mathtt{pk}$ in one entry of the file F; 2) in the second stage, V takes as input $\mathtt{sk}$, a statement $x \in L$ and a random string, V performs an interactive protocol with a prover, and outputs "accept" or "reject";
4. the first interaction of each prover starts after that all verifiers have completed their first stage.

Definition 1. *Given an NP-language L and its corresponding relation R_L, we say that a pair $\langle P, V\rangle$ is complete for L, if for all n-bit strings $x \in L$ and any witness y such that $(x, y) \in R_L$, the probability that V interacting with P on input y, outputs "reject"is negligible in n.*

Malicious provers in the BPK model. Let s be a positive polynomial and P^* be a probabilistic polynomial-time algorithm that takes as first input 1^n.

P^* is an *s-sequential malicious* prover if it runs in at most $s(n)$ stages in the following way: in stage 1, P^* receives a public key $\mathtt{pk}$ and outputs an n-bit string x_1. In every even stage, P^* starts from the final configuration of the previous stage, sends and receives messages of a single interactive protocol on input $\mathtt{pk}$ and can decide to abort the stage in any moment and to start the next one. In every odd stage $i > 1$, P^* starts from the final configuration of the previous stage and outputs an n-bit string x_i.

P^* is an *s-concurrent malicious* prover if on input a public key $\mathtt{pk}$ of V, can perform the following $s(n)$ interactive protocols with V: 1) if P^* is already running i protocols $0 \leq i < s(n)$ he can start a new protocol with V choosing the new statement to be proved; 2) he can output a message for any running protocol, receive immediately the response from V and continue.

Attacks in the BPK model. In [7] the following attacks have been defined.

Given an s-sequential malicious prover P^* and an honest verifier V, a *sequential attack* is performed in the following way: 1) the first stage of V is run on input 1^n and a random string so that a pair $(\mathtt{pk}, \mathtt{sk})$ is obtained; 2) the first stage of P^* is run on input 1^n and $\mathtt{pk}$ and x_1 is obtained; 3) for $1 \leq i \leq s(n)/2$ the $2i$-th stage of P^* is run letting it interact with V that receives as input $\mathtt{sk}, x_i$ and a random string r_i, while the $(2i + 1)$-th stage of P^* is run to obtain x_i.

Given an s-concurrent malicious prover P^* and an honest verifier V, a *concurrent attack* is performed in the following way: 1) the first stage of V is run on input 1^n and a random string so that a pair $(\mathtt{pk}, \mathtt{sk})$ is obtained; 2) P^* is run on input 1^n and $\mathtt{pk}$; 3) whenever P^* starts a new protocol choosing a statement, V is run on inputs the new statement, a new random string and $\mathtt{sk}$.

Definition 2. *Given a complete pair $\langle P, V\rangle$ for an NP-language L in the BPK model, then $\langle P, V\rangle$ is a concurrently (resp. sequentially) sound interactive argument system for L if for all positive polynomial s, for all s-concurrent (resp s-sequential) malicious prover P^*, for any false statement "$x \in L$" the probability that in an execution of a concurrent (resp. sequential) attack V outputs "accept" for such a statement is negligible in n.*

The strongest notion of zero knowledge, referred to as resettable zero knowledge, gives to a verifier the ability to rewind the prover to a previous state. This is significantly different from a scenario of multiple interactions between prover and verifier since after a rewinding the prover uses the same random bits.

We now give the formal definition of a black-box resettable zero-knowledge argument system for $\mathcal{NP}$ in the bare public-key model.

Definition 3. *An interactive argument system $\langle P, V\rangle$ in the BPK model is black-box* resettable zero-knowledge *if there exists a probabilistic polynomial-time algorithm S such that for any probabilistic polynomial time V^*, for any polynomials s, t, for any $x_i \in L$, $|x_i| = n$, $i = 1, \ldots, s(n)$, V^* runs in at most t steps and the following two distributions are indistinguishable:*

1. *the output of V^* that generates F with $s(n)$ entries and interacts (even concurrently) a polynomial number of times with each $P(x_i, y_i, j, r_k, F)$ where y_i is a witness for $x_i \in L$, $|x_i| = n$ and r_k is a random tape for $1 \leq i, j, k \leq s(n)$;*
2. *the output of S interacting with V^* on input $x_1, \ldots, x_{s(n)}$.*

Moreover we define such an adversarial verifier V^* as an (s,t)-resetting malicious verifier.

An important tool used this paper is that of a non-interactive zero-knowledge argument system.

Definition 4. *A pair of probabilistic polynomial-time algorithms (*NiPk,NiVk*) is a* non-interactive zero-knowledge argument system *for an $\mathcal{NP}$ language L if there exists a polynomial $k(\cdot)$,*

1. *(Completeness) for all $x \in L$, with $|x| = n$ and NP-witness y for $x \in L$,*

$$Pr[\sigma \overset{R}{\leftarrow} \{0,1\}^{k(n)}; \Pi \leftarrow \textsc{NiPk}(x, y, \sigma) : \textsc{NiVk}(x, \Pi, \sigma) = 1] = 1.$$

2. *(Soundness) for all $x \notin L$*

$$Pr[\sigma \overset{R}{\leftarrow} \{0,1\}^{k(n)}; \exists \Pi : \textsc{NiVk}(x, \Pi, \sigma) = 1]$$

   is negligible.
3. *(Simulatability) there exists a probabilistic polynomial-time algorithm S such that the family of distributions*

$$\{(\sigma, \Pi) \overset{R}{\leftarrow} S(x) : (\sigma, \Pi)\}_{x \in L} \, and \, \{\sigma \overset{R}{\leftarrow} \{0,1\}^{k(n)}; \Pi \overset{R}{\leftarrow} \textsc{NiPk}(x, y, \sigma) : (\sigma, \Pi)\}_{x \in L}$$

   are computationally indistinguishable.

We assume, without loss of generality, that a random reference string of length n is sufficient for proving theorems of length n (that is, we assume $k(n) = n$).

3 Concurrently Sound rZK Argument System for $\mathcal{NP}$ in the BPK Model

In this section we present a constant-round concurrently sound resettable zero-knowledge argument in the BPK model for all $\mathcal{NP}$ languages.

In our construction we assume the existence of an encryption scheme that is secure with respect to sub-exponential adversaries and that is *verifiably binding*. We next review the notion of semantic security adapted for sub-exponential adversaries and present the notion of a *verifiably binding* cryptosystem.

An encryption scheme is a triple of efficient algorithms $\mathtt{PK} = (\mathtt{G}, \mathtt{E}, \mathtt{D})$. The key generator algorithm $\mathtt{G}$ on input a random k-bit string r (the security parameter) outputs a pair $(\mathtt{pk}, \mathtt{sk})$ of public and private key. The public key $\mathtt{pk}$ is used to encrypt a string m by computing $\mathtt{E}(\mathtt{pk}, m; r)$ where r is a random string of length $|m|$.

Semantic security [14] is defined by considering the following experiment for encryption scheme $\mathtt{PK} = (\mathtt{G}, \mathtt{E}, \mathtt{D})$ involving a two-part adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$. The key generator $\mathtt{G}$ is run on a random k-bit string and keys $(\mathtt{pk}, \mathtt{sk})$ are given in output. Two $\textsc{poly}(k)$-bit strings ω_0 and ω_1 are returned by $\mathcal{A}_0$ on input $\mathtt{pk}$. Then b is taken at random from $\{0, 1\}$ and an encryption ξ of ω_b is computed. We say that adversary $\mathcal{A}$ is *successful for* $\mathtt{PK}$ if the probability that $\mathcal{A}_1$ outputs b on input $\mathtt{pk}, \omega_0, \omega_1$ and ξ is non-negligibly (in k) greater than $1/2$. We say that $\mathtt{PK}$ is *η-secure* if no adversary running in time $o(2^{k^\eta})$ is successful. The classical notion of semantic security is instead obtained by requiring that no polynomial-time adversary is successful.

Roughly speaking, a *verifiably binding* cryptosystem $\mathtt{PK}$ is a cryptosystem for which 1) given a string $\mathtt{pk}$ and an integer k, it is easy to verify that $\mathtt{pk}$ is a legal public key with security parameter k and 2) to each ciphertext corresponds at most one plaintext.

More formally,

Definition 5. *An η-secure encryption scheme* $\mathtt{PK} = (\mathtt{G}, \mathtt{E}, \mathtt{D})$ *is* verifiably binding *iff:*

1. *(binding): for any probabilistic polynomial-time algorithm $\mathcal{A}$ it holds that*

$$Pr[(\mathtt{pk}, m_0, m_1, r_0, r_1) \leftarrow \mathcal{A}(1^k) : \mathtt{E}(\mathtt{pk}, m_0; r_0) = \mathtt{E}(\mathtt{pk}, m_1; r_1)]$$

   is negligible in k;
2. *(verifiability): there exists a probabilistic polynomial-time algorithm* $\textsc{Ver}$ *such that if* $\mathtt{pk}$ *belongs to the output space of* $\mathtt{G}$ *on input a k-bit string then* $\textsc{Ver}(\mathtt{pk}, 1^k) = 1$*;* $\textsc{Ver}(\mathtt{pk}, 1^k) = 0$ *otherwise.*

Assumptions. To prove the properties of our protocol we make the following complexity theoretic assumptions:

1. The existence of an η-secure verifiably binding encryption scheme $\mathtt{PK} = (\mathtt{G}, \mathtt{E}, \mathtt{D})$ for some $\eta > 0$.

   We briefly note that the El Gamal encryption scheme [15] is verifiably binding since an exponentiation in Z_q^* is one to one and it can be easily verified that a positive integer q is a prime.
2. The existence of a one-to-one length-preserving one-way function $f : \{0, 1\}^* \rightarrow \{0, 1\}^*$ which, in turn, implies the existence of a pseudo-random family of functions $\mathcal{R} = \{R_s\}$.
3. The existence of a non-interactive zero-knowledge proof system (NIZK, in short) $(\textsc{NiPm}, \textsc{NiVm})$ for an $\mathcal{NP}$-complete language.

4. The existence of a 3-round witness indistinguishable argument of knowledge $\mathtt{WI} = (\mathtt{WI}_1, \mathtt{WI}_2, \mathtt{WI}_3)$ for a specific polynomial-time relation that we define in the following way. Let f be a one-to-one length-preserving one-way function and let $\mathtt{PK}$ be an η-secure verifiably binding encryption scheme. Then define the polynomial-time relation $\mathcal{C} = C(\mathtt{PK}, f)$ as consisting of all pairs $((\mathtt{pk}, v), (\mathtt{wit}))$, where $\mathtt{pk}$ is a public key of the output space of $\mathtt{G}$ and v is a string and either $\mathtt{wit} = \mathtt{sk}$ and $(\mathtt{pk}, \mathtt{sk})$ is in the output space of $\mathtt{G}$ or $\mathtt{wit} = u$ and $f(u) = v$.

Before describing our protocol formally, let us try to convey the main idea behind it. Fix an $\mathcal{NP}$ language L and let x be the input statement. The prover generates a puzzle (in our construction, the puzzle consists of a string v and solving the puzzle consists in finding the inverse $f^{-1}(v)$ of the one-to-one length-preserving one-way function f) and sends it to the verifier. The verifier uses $\mathtt{WI}$ to prove knowledge of the private key $\mathtt{sk}_i$ associated to her public key $\mathtt{pk}_i$ or knowledge of the solution of the puzzle given to her by the prover. Moreover, the prover and the verifier play a coin tossing protocol, based on the encryption scheme $\mathtt{PK}$ to generate a reference string for the NIZK proof that $x \in L$.

In our implementation of the FLS-paradigm [16], in the interaction between the prover and the verifier, the verifier will use his knowledge of the private key to run $\mathtt{WI}$. In order to prove concurrent soundness, we show an algorithm $\mathcal{A}$ that interacts with a (possibly) cheating prover P^* and breaks an η-secure encryption scheme in time $o(2^{k^\eta})$. The puzzle helps algorithm $\mathcal{A}$ in simulating the verifier with respect to a challenge public key $\mathtt{pk}$ for which it does not have access to the private key. Indeed, $\mathcal{A}$ instead of proving knowledge of the private key associated to $\mathtt{pk}$ proves knowledge of the solution of the puzzle by performing exhaustive search. By carefully picking the size of the puzzle (and thus the time required to solve it) we can make sure $\mathcal{A}$ runs in time $o(2^{k^\eta})$.

Note that when $\mathcal{A}$ inverts the one-to-one length-preserving one-way function and computes the witness-indistinguishable argument of knowledge, it runs in subexponential time in order to simulate the verifier without performing rewinds. Straight-line quasi-polynomial time simulatable argument systems were studied in detail in [17], where this relaxed simulation notion is used to decrease the round complexity of argument systems. We use a similar technique but for subexponential time simulation of arguments of knowledge.

If the steps described above were executed sequentially, we would have an 8-round protocol (one round for the prover to send the puzzle, three rounds for the coin tossing, three rounds for the witness-indistinguishable argument of knowledge, and one round for the NIZK). However, observe that the coin-tossing protocol and the 3-round witness-indistinguishable argument of knowledge can be performed in parallel thus reducing the the round complexity to 5 rounds. Moreover, we can save one more round, by letting the prover send the puzzle in parallel with the second round of the witness indistinguishable argument of knowledge. To do so, we need a special implementation of this primitive since, when the protocol starts, only the size of the statement is known and the statement itself is part of the second round. Let us now give the details of our construction.

The public file. The public file F contains entries consisting in public keys with security parameter k for the public-key cryptosystem $\mathtt{PK}$.

Private inputs. The private input of the prover consists of a witness y for $x \in L$. The private input of the verifier consists of the secret key $\mathtt{sk}_i$ corresponding to the public key $\mathtt{pk}_i$.

The protocol. Suppose that the prover wants to prove that $x \in L$ and denote by $n = \text{POLY}(k)$ the length of x. We denote by i the index of the verifier in the public file so that the verifier knows the private key $\mathtt{sk}_i$ associated with the i-th public key $\mathtt{pk}_i$ of the public file F.

In the first round V randomly picks an n-bit string σ_v that will be used as V's contribution to the reference string for the non-interactive zero-knowledge protocol. V compute the encryption ξ of σ_v using an n-bit string r_v as randomness and by using public key $\mathtt{pk}_i$. Moreover, V runs $\mathtt{WI}_1$ in order to compute the first message a_1 of the witness-indistinguishable argument of knowledge. Then V sends (ξ, a_1) to P. In the second round P verifies that $\mathtt{pk}_i$ is a legal public key for $\mathtt{PK}$ with k as security parameter and then computes its contribution to the random string to be used for the non-interactive argument by picking a random seed s and computing $(u, \sigma_p) = R_s(x \circ y \circ F \circ \xi \circ a_1 \circ i)$ ("$\circ$" denotes concatenation) where $\{R_s\}$ is a family of pseudorandom functions. The string u has length $k' \leq k$ (to be determined later) whereas σ_p has length n and is P's contribution for the reference string. P runs $\mathtt{WI}_2$ to compute the second message a_2 of the witness-indistinguishable argument of knowledge. Moreover P computes $v = f(u)$ where f is a one-to-one length-preserving one-way function and sends (σ_p, a_2, v) to the verifier. In the third round of the protocol V uses his knowledge of the private key to run $\mathtt{WI}_3$ obtaining a_3, so that she proves that she knows either the private key associated with $\mathtt{pk}_i$ or $f^{-1}(v)$. V then sends a_3, σ_v and r_v to P. In the last round of the protocol P verifies that the witness-indistinguishable argument of knowledge is correct and that ξ is an encryption of σ_v. Then P runs algorithm NIPM on input x and using $\sigma = \sigma_p \oplus \sigma_v$ as reference string obtaining a proof Π_p that is sent to V. A more formal description of the protocols is found in Figure 1.

Theorem 1. *If there exists an η-secure verifiably binding encryption scheme, a one-to-one length-preserving one-way function then there exists a* constant-round concurrently sound resettable zero-knowledge *argument for all languages in $\mathcal{NP}$ in the BPK model.*

Proof. Consider the protocol found in Figure 1.

Completeness. If $x \in L$ then P can always compute the proof Π and V accepts it.

Concurrent soundness. Assume by contradiction that the protocol is not concurrently sound. Thus there exists an s-concurrent malicious prover P^* that by,

Common input: the public file F, n-bit string $x \in L$ and index i that specifies the i-th entry of F. Public key $\mathtt{pk}_i$ has security parameter k.
P's private input: a witness y for $x \in L$.
V's private input: private key $\mathtt{sk}_i$.

V-round-1:

1. randomly pick $\sigma_v \leftarrow \{0,1\}^n$ and $r_v \leftarrow \{0,1\}^n$;
2. compute $\xi = \mathtt{E}(\mathtt{pk}_i, \sigma_v; r_v)$ and $a_1 = \mathtt{WI}_1(1^k)$;
3. send (ξ, a_1) to P;

P-round-2:

1. verify that $\mathtt{pk}_i$ is a public key with security parameter k for $\mathtt{PK}$;
2. randomly pick $s \leftarrow \{0,1\}^n$ and compute $R = R_s(x \circ y \circ F \circ \xi \circ a_1 \circ i)$; let u be the string consisting of the first k' bits of R and σ_p the string consisting of the next n bits of R;
3. compute $a_2 = \mathtt{WI}_2(a_1)$;
4. compute $v = f(u)$ where f is a one-to-one length preserving one-way function;
5. send (σ_p, a_2, v) to V;

V-round-3:

1. verify that v is a k'-bit string;
2. set $\sigma = \sigma_p \oplus \sigma_v$;
3. run algorithm $\mathtt{WI}_3$ on input instance $(\mathtt{pk}_i, v)$, messages a_1, a_2 using $\mathtt{sk}_i$ as a witness and obtaining a_3;
4. send (σ_v, a_3, r_v) to P;

P-round-4:

1. verify that $\xi = \mathtt{E}(\mathtt{pk}_i, \sigma_v; r_v)$;
2. set $\sigma = \sigma_p \oplus \sigma_v$;
3. verify that (a_1, a_2, a_3) is the correct transcript of the 3-round witness indistinguishable argument on input instance $(\mathtt{pk}_i, v)$;
4. run NiPm on input instance x, y as a witness and σ as reference string obtaining proof Π;
5. send Π to V;

V-decision: verify that Π is a proof by running algorithm NiVm on input x, Π and σ.

Fig. 1. The 4-round concurrently sound rZK argument system for $\mathcal{NP}$ in the BPK model. The values k and k' are determined as functions of n in the proof of concurrent soundness.

concurrently interacting with V, has non-negligible probability $p(n)$ of making the verifier accept some $x \notin L$ of length n. We assume we know the index of the session j^* in which the prover will succeed in cheating (this assumption will be later removed) and exhibit an algorithm $\mathcal{A}$ that has black-box access to P^* (*i.e.*, $\mathcal{A}$ simulates the work of a verifier V) and breaks the encryption scheme $\mathtt{PK}$ in $o(2^{k^\eta})$ steps, thus reaching a contradiction.

We now describe algorithm $\mathcal{A}$. $\mathcal{A}$ runs in two stages. First, on input the challenge public key $\mathtt{pk}$, $\mathcal{A}$ randomly picks two strings ω_0 and ω_1 of the same length as the length of the reference string used by (NiPm,NiVm) for inputs of length n. Then $\mathcal{A}$ receives as a challenge an encryption $\tilde{\xi}$ of ω_b computed using public key $\mathtt{pk}$ and $b \in \{0,1\}$. $\mathcal{A}$'s task is to guess $b \in \{0,1\}$ with a non-negligible advantage over 1/2 (we assume that b is randomly chosen).

For all the sessions, $\mathcal{A}$ interacts with the an s-concurrent prover P^* mounting a concurrent attack, and simulates the verifier by computing the two messages as explained below. When $\mathcal{A}$ reaches session j^*, $\mathcal{A}$ outputs her guess for bit b.

1. Session $j \neq j^*$.
   At V-round-1, $\mathcal{A}$ sends an encryption ξ of a randomly chosen string σ_v computed with r_v as randomness and sends the first round of the witness-indistinguishable argument of knowledge a_1. Upon receiving message (σ_p, a_2, v) from P^*, $\mathcal{A}$ inverts the one-to-one length-preserving one-way function f on v obtaining $u = f^{-1}(v)$ by performing exhaustive search in $\{0,1\}^{k'}$. $\mathcal{A}$ then computes a_3 by running $\mathtt{WI}_3$ on input instance $(\mathtt{pk}_i, v)$ and witness u and sends to P^* the triple (σ_v, a_3, r_v).
   Note that $\mathcal{A}$ plays round V-round-1 identically to the honest verifier while $\mathcal{A}$ plays round V-round-3 by using a different witness w.r.t. V for the non-interactive zero-knowledge argument of knowledge that however is concurrent witness indistinguishable.
2. Session j^*.
   At V-round-1, $\mathcal{A}$ computes the first message of the witness-indistinguishable argument of knowledge a_1 and sets ξ equal to the challenge encryption $\tilde{\xi}$. Then $\mathcal{A}$ sends (ξ, a_1) to V.
   At V-round-3, $\mathcal{A}$ cannot continue with this session since she does not know the decryption of ξ (remember that $\xi = \tilde{\xi}$) and thus can not play the third round. However, by assumption P^* can produce with non-negligible probability a string Π^* that is accepted by NiVm on input x and reference string $\rho_b^* = \omega_b \oplus \sigma_p$. Let τ be an upper bound on the length of such a non-interactive zero-knowledge argument. $\mathcal{A}$ checks, by exhaustive search, if there exists $\Pi_0 \in \{0,1\}^\tau$, such that NiVm accepts Π_0 on input x and ρ_0^* as reference string. Then $\mathcal{A}$ searches for a string $\Pi_1 \in \{0,1\}^\tau$ by considering ρ_1^* as reference string. If a proof Π_0 is found and no proof Π_1 is found then $\mathcal{A}$ outputs 0; in the opposite case $\mathcal{A}$ outputs 1; otherwise (that is, if both or neither proof exists) $\mathcal{A}$ randomly guesses the bit b.
   We note that the distribution of the first message of session j^* is still identical to the distribution of the honest verifier's message.

Let us now show that the probability that $\mathcal{A}$ correctly guesses b is non-negligibly larger that 1/2. We have that

$$\begin{aligned} Pr[A \text{ outputs } b] &= Pr[\exists \Pi_b \wedge \nexists \Pi_{1-b}] + \frac{1}{2}\left(Pr[\exists \Pi_b \wedge \exists \Pi_{1-b}] + Pr[\nexists \Pi_b \wedge \nexists \Pi_{1-b}]\right) \\ &= \frac{1}{2} + \frac{1}{2}\left(Pr[\exists \Pi_b \wedge \nexists \Pi_{1-b}] - Pr[\nexists \Pi_b \wedge \exists \Pi_{1-b}]\right) \\ &= \frac{1}{2} + \frac{1}{2}\left(Pr[\exists \Pi_b \wedge \nexists \Pi_{1-b} \wedge x \notin L] - Pr[\nexists \Pi_b \wedge \exists \Pi_{1-b} \wedge x \notin L]\right). \end{aligned}$$

The last equality follows from the observation that, by the completeness of the NIZK, the events $\not\exists \Pi_{1-b}$ and $\not\exists \Pi_b$ can happen only if $x \notin L$. Now, we have

$$\begin{aligned} Pr[A \text{ outputs } b] &= \frac{1}{2} + \frac{1}{2}\big(Pr[\exists \Pi_b \wedge x \notin L] - Pr[\exists \Pi_b \wedge \exists \Pi_{1-b} \wedge x \notin L] - \\ &\quad Pr[\not\exists \Pi_b \wedge \exists \Pi_{1-b} \wedge x \notin L]\big) \\ &= \frac{1}{2} + \frac{p(n)}{2} - \frac{Pr[\exists \Pi_{1-b} \wedge x \notin L]}{2}. \end{aligned}$$

Now, since the string ω_{1-b} is picked at random and P^* has no information about it, the string ρ^*_{1-b} is random and thus, by the soundness of (NiPm,NiVm), $Pr[\exists \Pi_{1-b} \wedge x \notin L]$ is negligible. Therefore, the probability that $\mathcal{A}$ correctly guesses b is non-negligibly larger than 1/2.

We note that algorithm $\mathcal{A}$ takes time $\text{POLY}(n) \cdot (2^\tau + 2^{k'})$. Writing τ as $\tau = n^\gamma$, for some constant γ, we pick k and k' so that $n^\gamma < k^{\eta/2}$ and $k' < k^{\eta/2}$. We thus have that $\mathcal{A}$ breaks an η-secure verifiably binding cryptosystem in time bounded by $\text{POLY}(k^{\eta/2})(2^{k^{\eta/2}} + 2^{k^{\eta/2}}) = o(2^{k^\eta})$.

Therefore the existence of $\mathcal{A}$ contradicts the η-security of the cryptosystem.

In our proof we assumed that $\mathcal{A}$ knows the value j. If this is not the case that $\mathcal{A}$ can simply guess the values and the same analysis applies and the probability that $\mathcal{A}$ correctly guesses b decreases by a polynomial factor.

Resettable Zero Knowledge. Let V^* be an (s,t)-resetting verifier. We now present a probabilistic polynomial-time algorithm $S \equiv S^{V^*}$ that has black-box access to V^* and whose output is computationally indistinguishable from the view of the interactions between P and V^*.

We start with an informal discussion. The construction of S is very similar to the construction of the simulator for the constant-round (sequentially sound) resettable zero-knowledge argument for any NP language and in the BPK model, given in [3] (protocol 6.2). In particular, note that both the protocol of Figure 1 and protocol 6.2 in [3] can be abstractly described as follows. The prover and the verifier run a 3-round argument of knowledge, where the verifier, acting as a prover, proves knowledge to the prover, acting as verifier, of some trapdoor information. Knowledge of the trapdoor information allows for efficient simulation of the interaction between the prover and the verifier. In [3], the trapdoor information is the private key associated with the verifier's public key. In our protocol, the trapdoor information is either the private key associated with the verifier's public key (for the real verifier) or the inverse of an output of a one-to-one length-preserving one-way function sent from the prover to the verifier. Note that just to obtain round optimality we use a special witness-indistinguishable argument of knowledge where the statement is known only after that the second round is played while its size is known from the beginning. Due to this difference, our simulator only differs from the one of [3] in the fact that we need to prove that when the simulator runs the extractor of the argument of knowledge, with high probability it extracts the verifier's private key (rather than $f^{-1}(v)$). The rest of the construction of our simulator is conceptually identical to that of [3], but we still review a more precise description here for completeness.

First of all, without loss of generality, we make the following two simplifying assumptions. Recall that, since our protocol is a resettable zero-knowledge argument system, V^* is allowed to reset the prover. However, in [3] Canetti *et al.* proved that in such a setting a verifier that concurrently interacts with many incarnations of the prover does not get any advantage with respect to a sequential (resetting) verifier (that is, a verifier that runs a new session only after having terminated the previous one). Thus in this proof we will consider V^* as a sequential (resetting) verifier. A second assumption is that we can define S for a modification of our protocol in which the prover uses a truly random function rather than a pseudo-random one to compute her random bits. Proving that the two views are computationally indistinguishable is rather standard.

S runs the first stage of V^* so that the public file composed by $s(n)$ entries is obtained. In the second stage, the aim of the simulator is to obtain the private keys corresponding to the public keys of the public file. Let $V^*(F)$ be the state of V^* at the end of the first stage.

In the following, we say that a session is *solved* by S if S has the private key corresponding to the public key used by V^* in this session. The work of S in the second stage of the simulation is composed by at most $s(n)+1$ sequential phases. In each phase, either S has a chance of terminating the simulation or S learns one more private key. At the end of each phase S rewinds V^* to state $V^*(F)$. The simulation ends as soon as S manages to solve all sessions of a phase.

We describe now the work of S during a phase. Once a session is started, S receives the first message from V^*. Then there are two cases. If the session is solved by S then S can simulate the prover; otherwise, S tries to obtain the private key used in this session so that all future sessions involving this verifier will be solved by S.

Specifically, first consider the simpler case of a solved session. We distinguish two sub-cases. First, we consider the sub-case where the first message in the session (ξ, a_1) has not appeared before for the same incarnation of the prover, *i.e.*, (ξ, a_1) has not appeared before for the same prover oracle accessed by V^* with the same random tape, same witness and same theorem. Then S runs the simulator for (NiPm,NiVm) on input x and obtains a pair (σ^*, Π^*) and then forces σ equal to σ^* in the following way. Since S knows the verifier's secret-key (we are assuming in this sub-case that the session is solved), S can decrypt ξ and thus obtain the string σ_v computed by the verifier at the first round. Thus S sets $\sigma_p = \sigma_v \oplus \sigma^*$. Consequently, in round P-round-4, S will send "proof" Π^* (that is computationally indistinguishable from the proof computed by the real prover). We use here the binding property of the encryption scheme since S must decrypt ξ obtaining the same value σ_v that will be sent by V^* in round V-round-3.

Now we consider the sub-case where the first message in the session (ξ, a_1) has already appeared in such a phase for the same incarnation of the prover. Here S sends the same strings σ_p, a_2 and the same k'-bit string v that was sent in the previous session containing (ξ, a_1) as first message for the same incarnation of the prover. Even for the case of the third message of a session that has already appeared for the same incarnation of the prover, S replies with the same round P-round-4 played before.

We now consider the harder case of a session which is not solved by S. In this case S uses the argument of knowledge of V^* to obtain the private key used in this session. Specifically, in any unsolved session, the simulator uses the extractor E associated with the witness-indistinguishable argument of knowledge used by the verifier.

Recall that we denote by (ξ, a_1) the first message sent by the verifier in the current session, by $\mathtt{pk}_i$ the verifier's public key and by $v = f(u)$ the puzzle sent by the simulator when simulating the prover's first message. We now distinguish three possible cases.

Case 1: The message (ξ, a_1) has not yet appeared in a previous session for the same incarnation of the prover and the extractor E obtains $\mathtt{sk}_i$ as witness. Note that S obtains the verifier's private key by running E. This is the most benign of the three cases since the session is now solved.

Case 2: The message (ξ, a_1) has not yet appeared in a previous session for the same incarnation of the prover and the extractor E obtains $f^{-1}(v)$ as witness. Note however that the value v has been chosen by S itself. If this case happens with non-negligible probability then we can use V^* to invert the one-way function f. We stress that this case is the only conceptual difference between our proof and the proof of rZK of protocol 6.2 in [3].

Case 3: The message (ξ, a_1) has already appeared in a previous session for the same incarnation of the prover. Note that since we are assuming that the current session is not solved by S, this means that in at least one previous session, V^* sent (ξ_1, a_1) but then did not continue with such a session. This prevents S from simulating as in case 2 since the simulation would not be correct. (Specifically, as discussed in [3], in a real execution of the argument, the pseudo-random string used as random string for the prover's first message is determined by the previous uncompleted session (the input of R_s is the same in both cases and the seed s is taken from the same random string) and therefore cannot be reset by S to simulate this case by running an independent execution of E.) This problem is bypassed precisely as in [3]. That is, S tries to continue the simulation from the maximal sequence of executions which does not contain (ξ, a_1) as a first step of the verifier for such an incarnation of the prover, using a new random function.

The same analysis in [3] shows that this simulation strategy ends in expected polynomial time and returns a distribution indistinguishable from a real execution of the argument. □

3-*Round WI Argument of Knowledge.* As already pointed out above, we can save one round (and thus obtain a 4-round argument system instead of 5-round one) by having the prover send the puzzle after the verifier has started the witness-indistinguishable argument of knowledge. In this argument of knowledge, the verifier acts as a prover and shows knowledge of either the secret key associated with his private key or of a solution of the puzzle. Consequently, the input statement of such an argument of knowledge is not known from the start and actually, when the first message is produced, only its length is known.

Next we briefly describe such an argument of knowledge by adapting to our needs the technique used by [16] to obtain a non-interactive zero-knowledge proof system for Hamiltonicity.

1. The prover commits to n randomly generated Hamiltonian cycles (each edge is hidden in a committed adjacency matrix of degree n);
2. the graph G is presented to the prover and the verifier and verifier sends an n-bit random challenge;
3. if the i-th bit of the challenge is 0 then the prover opens the i-th Hamiltonian cycle;
4. if the i-th bit of the challenge is 1 then the prover sends a permutation π_i and shows that each edge that is missing in the graph $\pi_i(G)$ corresponds to a commitment of 0 in the i-th committed Hamiltonian cycle.

Completeness, soundness and witness indistinguishability can be easily verified. The protocol is an argument of knowledge since an extractor that rewinds the prover and changes the challenge obtains a Hamiltonian cycle of G.

4 Sequentially Sound Sequential Zero Knowledge for $\mathcal{NP}$ in the BPK Model

In this section we give a 3-round sequentially sound sequential zero-knowledge argument in the BPK model for any language in $\mathcal{NP}$.

Assumptions. We start by listing the tools and the complexity-theoretic assumptions we need for the construction of this section.

1. We assume the existence of an η-secure signature scheme $\mathtt{SS} = (\mathtt{SigG}, \mathtt{Sig}, \mathtt{Ver})$. Here $\mathtt{SigG}$ denotes the key generator algorithm that receives the security parameter k (in unary) and returns a pair $(\mathtt{pk}, \mathtt{sk})$ of public keys; $\mathtt{Sig}$ is the signature algorithm that takes as input a message m and a private key $\mathtt{sk}$ and returns a signature s of m; and $\mathtt{Ver}$ is the signature verification algorithm that takes a message m, a signature s and a public key $\mathtt{pk}$ and verifies that s is a valid signature.
   The scheme $\mathtt{SS}$ is η-secure in the sense that no algorithm running in time $o(2^{k^\eta})$ that has access to a signature oracle but not to the private key can forge the signature of a message m for which it has not queried the oracle.
   It is well known that if sub-exponentially strong one-way functions exist then it is possible to construct secure signature schemes [18].
   We assume that signatures of k-bit messages produced by using keys with security parameter k have length k. This is not generally true as for each signature scheme we have a constant a such that signatures of k-bit messages have length k^a but this has the advantage of not overburdening the notation. It is understood that all our proofs continue to hold if this assumption is removed.

2. We assume the existence of a one-round perfectly binding computationally hiding γ-extractable commitment scheme. The scheme is γ-extractable in the sense that there exists an extractor algorithm E that on input a commitment, computes in time $O(2^{k^\gamma})$ the committed value.
   Such a commitment schemes are known to exist under the assumption of the existence of sub-exponentially strong one-to-one length-preserving one-way functions.
3. We also assume the existence of ZAPs for all $\mathcal{NP}$ (see [19]).

In sums, our construction is based on the existence of subexponentially strong one-to-one length preserving one-way functions and one-way trapdoor permutations.

We start by briefly describing the main idea of our protocol. The prover and the verifier play the following game: the prover picks a random message m_1, computes a commitment $\tilde{m}_1$ of m_1 and asks the verifier to sign $\tilde{m}_1$; the verifier signs the commitment and sends back to the prover such a signature and a message m_2. Finally the prover, constructs an extractable commitment `com` of a random message and proves to the verifier using a ZAP that either $x \in L$ or `com` is the extractable commitment of a signature of a commitment of m_2. Let us now informally argue about sequential soundness and sequential zero-knowledge of the argument system described. For the sequential soundness, we observe that, since m_2 is chosen at random by the verifier for each sequential execution of the protocol, it is unlikely that the prover knows the signature of a commitment of m_2. For the zero-knowledge property instead, the simulator, once m_2 is received, rewinds $V^\star$ and opens a new session with the verifier in which he sets $m_1 = m_2$, computes a commitment of $m_1 = m_2$ and sends it to the verifier that thus produces a signature of a commitment of m_2. Going back to the original session, the simulator has a witness for the ZAP and can thus complete the simulation.

Theorem 2. *If there exist subexponentially strong one-to-one length-preserving one-way functions and trapdoor permutations then there exists a* 3*-round sequentially sound sequential zero-knowledge argument for* $\mathcal{NP}$ *in the* BPK *model.*

Proof. Completeness and *Sequential soundness* can be easily proved. For the *Sequential Zero Knowledge,* we now describe a simulator S. We consider a malicious verifier V^* that in the first stage outputs the public file F and in the second stage interacts with P by considering $s(n)$ possible theorems and $s(n)$ possible entries of F. However V^* is now a sequential verifier and thus he cannot run twice the same incarnation of P, neither he can run two concurrent sessions with P. Thus the simulation proceeds session by session and we can focus only in the simulation of a generic session.

Let V_1^* be the state of V^* at the beginning of a given session. The simulator sends in the first round a message that is distributed identically w.r.t. the one of the prover. Then V^* replies by sending a message m_2, let V_2^* the state of V^* in such a step. The simulator rewinds V^* to state V_1^* and plays again the first round but this time he sets $m_1 = m_2$. The simulator repeats this first round with a different randomness as long as the verifier sends a valid second message that therefore contains a signature of a commitment of m_2. The simulator can use the signature of a commitment of m_2 as witness for the third round of the original proof, that can be given by rewinding V^* to state V_2^*. More precisely, S rewinds V^* to state V_2^* and computes $\tilde{a}$ as a commitment of a commitment of m_2 and $\tilde{b}$ as a commitment of the previously received signature. Then S has a witness for playing the ZAP.

The previously described rewind strategy allows the simulator to complete the simulation in expected polynomial-time and, moreover, the indistinguishability of the ZAP and the hiding of the commitment scheme guarantee that the distribution of the output is computationally indistinguishable from an interaction between a real prover and $V^\star$.

We remark that it is possible to base our construction on primitives secure against polynomial-time adversaries by employing a 3-round witness indistinguishable argument where the statement is chosen by the prover before producing the third message.

5 Conclusions

In an asynchronous environment like the Internet resettable zero-knowledge protocols that are not concurrently sound in the BPK model cannot be considered secure and previous concurrently sound protocols required stronger assumptions than the BPK model.

In this work we have positively closed one of the main open problems regarding zero knowledge in the BPK model. We have shown that a constant-round concurrently sound resettable zero-knowledge argument system in the BPK model exists. In particular, we have shown a 4-round protocol which is optimal for the black-box model.

Acknowledgments

We would like to thank the anonymous referee for his/her very useful remarks on our submission.

References

1. Goldwasser, S., Micali, S., Rackoff, C.: The Knowledge Complexity of Interactive Proof-Systems. SIAM J. on Computing **18** (1989) 186–208
2. Dwork, C., Naor, M., Sahai, A.: Concurrent Zero-Knowledge. In: Proceedings of the 30th ACM Symposium on Theory of Computing (STOC '98), 409–418
3. Canetti, R., Goldreich, O., Goldwasser, S., Micali, S.: Resettable Zero-Knowledge. In: Proceedings of the 32nd ACM Symposium on Theory of Computing (STOC '00), 235–244
4. Canetti, R., Kilian, J., Petrank, E., Rosen, A.: Black-Box Concurrent Zero-Knowledge Requires $\omega(\log n)$ Rounds. In: Proceedings of the 33st ACM Symposium on Theory of Computing (STOC '01), 570–579

5. Barak, B.: How to Go Beyond the Black-Box Simulation Barrier. In: Proceeding of the 42nd Symposium on Foundations of Computer Science, (FOCS '01), 106–115
6. Blum, M., De Santis, A., Micali, S., Persiano, G.: Non-Interactive Zero-Knowledge. SIAM J. on Computing **20** (1991) 1084–1118
7. Micali, S., Reyzin, L.: Soundness in the Public-Key Model. In: Advances in Cryptology – Crypto '01. Volume 2139 of Lecture Notes in Computer Science. Springer-Verlag, 542–565
8. Barak, B., Goldreich, O., Goldwasser, S., Lindell, Y.: Resettably-Sound Zero-Znowledge and its Applications. In: Proceeding of the 42nd Symposium on Foundations of Computer Science, (FOCS '01), 116–125
9. Micali, S., Reyzin, L.: Min-round Resettable Zero-Knowledge in the Public-key Model. In: Advances in Cryptology – Eurocrypt '01. Volume 2045 of Lecture Notes in Computer Science. Springer-Verlag (2001) 373–393
10. Zhao, Y., Deng, X., Lee, C., Zhu, H.: Resettable Zero-Knowledge in the Weak Public-Key Model. In: Advances in Cryptology – Eurocrypt '03. Volume 2045 of Lecture Notes in Computer Science. Springer-Verlag (2003) 123–139
11. Goldreich, O.: Concurrent Zero-Knowledge with Timing, Revisited. In: Proceedings of the 34th ACM Symposium on Theory of Computing (STOC '02), ACM (2002) 332–340
12. Di Crescenzo, G., Ostrovsky, R.: On Concurrent Zero-Knowledge with Preprocessing. In: Advances in Cryptology – Crypto '99. Volume 1666 of Lecture Notes in Computer Science. Springer-Verlag (1999) 485–502
13. Damgard, I.: Efficient Concurrent Zero-Knowledge in the Auxiliary String Model. In: Advances in Cryptology – Eurocrypt '00. Volume 1807 of Lecture Notes in Computer Science. Springer-Verlag (2000) 418–430
14. Goldwasser, S., Micali, S.: Probabilistic encryption. J. of Comp. and Sys. Sci. **28** (1984) 270–299
15. El Gamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. In: Advances in Cryptology – Crypto '84. Volume 196 of Lecture Notes in Computer Science. Springer Verlag (1984) 10–18
16. Feige, U., Lapidot, D., Shamir, A.: Multiple Non-Interactive Zero Knowledge Proofs Under General Assumptions. SIAM J. on Computing **29** (1999) 1–28
17. Pass, R.: Simulation in Quasi-Polynomial Time and Its Applications to Protocol Composition. In: Advances in Cryptology – Eurocrypt '03. Volume 2045 of Lecture Notes in Computer Science. Springer-Verlag (2003) 160–176
18. Rompel, J.: One-Way Functions are Necessary and Sufficient for Digital Signatures. In: Proceedings of the 22nd ACM Symposium on Theory of Computing (STOC '90). (1990) 12–19
19. Dwork, C., Naor, M.: Zaps and their applications. In: IEEE Symposium on Foundations of Computer Science. (2000) 283–293