Improved Setup Assumptions for 3-Round Resettable Zero Knowledge

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

[1] Telcordia Technologies, Piscataway, NJ, USA. `giovanni@research.telcordia.com`
[2] Dip. di Inf. ed Appl., Univ. di Salerno, Baronissi, Italy. `giuper@dia.unisa.it`
[3] Dép. d'Inf., École Normale Supérieure, Paris, France. `ivan.visconti@ens.fr`

Abstract. In the *bare public-key* model, introduced by Canetti *et al.* [STOC 2000], it is only assumed that each verifier deposits during a set-up phase a public key in a file accessible by all users at all times. 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. Indeed Micali and Reyzin have introduced four different notions which are called (from weaker to stronger): one-time, sequential, concurrent and resettable soundness. In this paper we introduce the *counter public-key model* (the `cPK` model for short), an augmentation of the bare public-key model in which each verifier is equipped with a counter and, like in the original bare public-key model, the key of the verifier can be used for any polynomial number of interactions with provers. In the `cPK` model, we give a *three-round concurrently-sound resettable zero-knowledge* argument of membership for NP. Previously similar results were obtained by Micali and Reyzin [EuroCrypt 2001] and then improved by Zhao *et al.* [EuroCrypt 2003] in models in which, roughly speaking, each verifier is still equipped with a counter, but the key of the verifier could only be used for a fixed number of interactions.

1 Introduction

The notion of Zero Knowledge, put forth by Goldwasser, Micali and Rackoff [1], has proved to be a fundamental concept in the area of complexity-based cryptography. The original notions of security with respect to malicious provers (formalized by the *soundness* requirement) and the security with respect to malicious verifiers (captured by the *zero-knowledge* requirement) only considered a prover and a verifier acting in isolation. Recently, the case in which provers and verifiers are part of a large system (and thus prover-verifier interactions may be interleaved) has been considered and stronger notions of soundness and zero knowledge have been proposed. In a sequence of papers the notions of concurrent zero knowledge [2] and resettable zero knowledge [3] were introduced and protocols in the standard model were provided [4,5,6,3].

An important measure of an efficiency of a system is the number of rounds needed. Lower bounds for the number of rounds for concurrent and resettable zero knowledge have shown that these strong notions of security cannot be implemented, in the standard model, using a constant number of rounds if black-box zero knowledge is sought [7]. Canetti *et al.* [3] were thus motivated to introduce the *bare public-key* model (the BPK model for short) in which, during a set-up stage, each verifier stores in a public file his public key to be used in all subsequent interactions and keeps secret the associated private key. In this model constant-round concurrent and resettable zero-knowledge arguments for all $\mathcal{NP}$ were shown to exist in [3].

Other models have been proposed in order to achieve similar results by focusing on black-box concurrent zero knowledge, in particular the results of [2,8,9] in the timing model, those of [10,11] in the common reference string model, those of [13] in the preprocessing model, and those of [12] in some partially synchronous model.

Among the different proposed models, the BPK model has the following advantages: 1) it is not based on any trusted third party; 2) no timing assumption is made; 3) the set-up stage is non-interactively performed only by the verifiers. Consequently, the public-key model is, from among the currently proposed models, the one that makes the least set-up assumptions and, in particular, it is weaker than the widely accepted *Public Key Infrastructure* model.

Subsequently to the introduction of the BPK model, Micali and Reyzin [14] noticed that, unlike in the standard model for interactive zero knowledge, distinct notions of soundness arise depending on whether the verifier's public key is used for once (one-time soundness), for polynomially many sequential arguments (sequential soundness), for polynomially many concurrently interleaved arguments (concurrent soundness), or whether the prover is allowed to reset the verifier to a given state during the interaction (resettable soundness). However, they showed that resettably sound zero knowledge cannot be achieved in the black-box model for non-trivial languages. Consequently, for black-box zero knowledge, the strongest possible notion is that of a *concurrently sound resettable zero-knowledge* argument. In [14], Micali and Reyzin showed that in the BPK model, concurrent soundness cannot be achieved in less than four rounds. Moreover they showed that the argument system of Canetti *et al.* presented in [3] is only sequentially sound and the same holds for the four-round resettable zero-knowledge argument presented in [14]. Recently, the existence of a constant-round concurrently-sound resettable zero-knowledge argument in the BPK model has been proved by [15] where a 4-round concurrently-sound resettable zero-knowledge argument in the BPK model has been given for all $\mathcal{NP}$ languages.

Prior to the work of [15], augmented variations of the BPK model had been presented in which constant-round concurrently-sound resettable zero knowledge could be achieved. These proposals are interesting, even in light of the result of [15], since they achieve three-round concurrently-sound resettable zero knowledge which is remarkable as no non-trivial (black-box) zero knowledge can be achieved in less than three rounds in any variation of the BPK model [14,16]. In particular, three-round concurrently sound resettable zero-knowledge arguments for all $\mathcal{NP}$ are possible in the *upper-bounded* public-key (UPK, for short) model (see [17]) and in the *weak* public-key (WPK, for short) model (see [18]). In the

UPK model, each public key can be used for a fixed number of arguments to be determined during set-up. The verifier is equipped with a counter to keep track of the number of arguments he has been involved in. In the WPK model, instead, there is a fixed upper bound on the number of times the verifier can be involved in sessions regarding the same statement. Thus, also in this model, the verifier must have a counter (actually, one for each possible statement).

Our Contribution. In this paper we introduce the *counter* public-key (cPK for short) model, that is a model weaker than the WPK (and thus of the UPK) model and only slightly stronger than the BPK model. The cPK model, like the UPK of Micali and Reyzin and the WPK of Zhao *et al.*, is an extension of the BPK model in which the verifier is equipped with a counter that, roughly speaking, keeps track of the number of sessions that she has been involved with. However, unlike the UPK model and the WPK model, each public key of the verifier can be used any polynomial number of times exactly like in the original BPK model. Therefore, our proposed model, although slightly stronger than the original BPK model, can be considered much weaker than the UPK model and the WPK model. Indeed, in the cPK model the verifier has no bound on the number of proofs he can engage with the provers while in both UPK and WPK models once the bound is reached, soundness is not guaranteed to hold.

We first present a *three-round concurrently sound resettable zero-knowledge argument of membership* for $\mathcal{NP}$ in the cPK model. This construction improves the previous works of [17,18] that achieved the same result but in stronger models. We notice that, in the BPK model, concurrent soundness requires 4 rounds. Our protocol is based on the existence of sub-exponentially hard primitives, as in all previous works for obtaining a constant-round resettable zero-knowledge argument in any public-key model.

Our second construction is a *three-round concurrently sound concurrent zero-knowledge argument of knowledge* for all $\mathcal{NP}$ relations in the cPK model under standard intractability assumptions. We notice that, in the black-box model, resettable zero-knowledge arguments of knowledge exist only for trivial languages and thus concurrent zero knowledge is the strongest notion of zero knowledge that can be achieved when arguments of knowledge are sought.

2 The cPK Model

The cPK 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 $\mathcal{NP}$-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 pk in one entry of the file F; 2) in the second stage, V takes as input the private key sk, a counter value c, a statement "$x \in L$" and a random string, then V performs an interactive protocol with a prover , and outputs "accept" or "reject";
4. interactions between prover and verifier start after all verifiers have completed their first stage.

Definition 1. *Given an $\mathcal{NP}$-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 cPK *model.* We will give argument systems that are sound with respect to s-concurrent malicious provers, for any positive polynomial s. An s-concurrent malicious prover $P^\star$ for the complete pair $\langle P^\star, V\rangle$ is a probabilistic polynomial-time algorithm that takes as input V's public key pk, and, if $P^\star$ is concurrently running i sessions, for $0 < i \leq s(n)$, $P^\star$ can pick a new statement x_{i+1} and a value c_{i+1} of the counter and start a new session with V on input x_{i+1} and c_{i+1}. The only restriction is that, for each value c of the counter, $P^\star$ can only start one session with value c. Also, we allow the malicious prover to schedule his messages in the concurrent sessions in any way he wants and, for each message of $P^\star$, V's reply is immediately received.

We stress here that our definition of malicious prover is the same used by L. Reyzin (see [16]) for the UPK model. Instead, in [18], the value of the counter is assumed to be private to the verifier and the malicious prover has no way of manipulating it. Moreover, we stress that in the cPK model there is no bound on the number of sessions in which the verifier can be involved, thus the model is weaker than the WPK and UPK models and very close to the standard BPK model.

Given an s-concurrent malicious prover $P^\star$ 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) $P^\star$ is run on input 1^n and pk; 3) whenever $P^\star$ starts a new protocol choosing a statement, V is run on inputs the new statement, a new random string and sk.

Definition 2. *Given a complete pair $\langle P, V\rangle$ for an $\mathcal{NP}$-language L in the* cPK *model, then $\langle P, V\rangle$ is a* concurrently sound *interactive argument system in the* cPK *model for language L if, for all positive polynomial s, for all s-concurrent malicious prover $P^\star$, for any false statement "$x \in L$" the probability that in an execution of a concurrent 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 reset the prover to a previous state. This is significantly different from a scenario of multiple interactions between prover and verifier since after a reset, the prover uses the same random bits. We now give the formal definition of a black-box resettable zero-knowledge argument system with concurrent soundness for $\mathcal{NP}$ in the counter public-key model.

Definition 3. *An interactive argument system $\langle P, V\rangle$ in the* cPK *model is black-box* resettable zero-knowledge *if for any polynomial $t(\cdot)$, and for any probabilistic adversary $V^\star$ running in time $t(\cdot)$, there exists a probabilistic polynomial-time algorithm S such that for any polynomial $s(\cdot)$, for any $x_1, \cdots, x_{s(n)} \in L$ of length n, the following two distributions are indistinguishable:*

1. *the output of $V^\star$ consisting of a public file F with $s(n)$ entries and the transcript of a polynomial number of (even concurrent) interactions with each $P(x_l, y_l, r_g, F, i)$ where y_l is a witness for "$x_l \in L$", $|x_l| = n$, r_g is a random tape and i specifies an entry of the public file, for $1 \le i, l, g \le s(n)$;*
2. *the output of S that has black-box access to $V^\star$ on input $x_1, \ldots, x_{s(n)}$.*

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

3 Cryptographic Tools

We review the cryptographic tools that we will use in our constructions. We start from the notions of an η-secure digital signature scheme and of a γ-secure commitment scheme.

Definition 4. *An η-secure* digital signature scheme SS *is a triple of algorithms* $\mathtt{SS} = (\mathtt{G}, \mathtt{Sig}, \mathtt{Ver})$ *such that*

1. **correctness:** *for all messages $m \in \{0,1\}^k$,*

$$Pr[(\mathtt{pk}, \mathtt{sk}) \leftarrow \mathtt{G}(1^k); \hat{m} \leftarrow \mathtt{Sig}(m, \mathtt{sk}) : \mathtt{Ver}(m, \hat{m}, \mathtt{pk}) = 1] = 1.$$

2. **unforgeability:** *for all algorithms A running in time $o(2^{k^\eta})$ it holds that*

$$Pr[(\mathtt{pk}, \mathtt{sk}) \leftarrow \mathtt{G}(1^k); (m, \hat{m}) \leftarrow A^{\mathcal{O}(\mathtt{sk})}(pk) : m \notin \mathtt{Query}\,\text{and}\,\mathtt{Ver}(m, \hat{m}, \mathtt{pk}) = 1]$$

is negligible in k where $\mathcal{O}(\mathtt{sk})$ is a signature oracle *that on input a message returns as output a signature of the message and* Query *is the set of signature requests submitted by A to $\mathcal{O}$.*

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.

Standard secure signature schemes exist under the assumption of the existence of one-way functions [19]. In our case we need the existence of functions that are one-way with respect to subexponential-time adversaries.

Definition 5. *An γ-secure* bit commitment scheme *is a pair of algorithms* $(\mathtt{Com}, \mathtt{Dec})$ *such that*

1. **correctness:** *for all* $b \in \{0,1\}$ *and for all* k,

$$Pr[(\texttt{com}, \texttt{dec}) \leftarrow \texttt{Com}(b, 1^k) : \texttt{Dec}(\texttt{com}, \texttt{dec}, b) = 1] = 1;$$

2. **perfect binding:** *for all* k*, the set of strings* com *of length* k *for which there exist strings* $\texttt{dec}_0, \texttt{dec}_1$ *such that* $\texttt{Dec}(\texttt{com}, \texttt{dec}_0, 0) = 1$ *and* $\texttt{Dec}(\texttt{com}, \texttt{dec}_1, 1) = 1$ *is empty;*
3. **computationally hiding:** *the ensembles of random variables*

$\{[(\texttt{com}, \texttt{dec}) \leftarrow \texttt{Com}(0, 1^k) : \texttt{com}]\}_{k>0}$ and $\{[(\texttt{com}, \texttt{dec}) \leftarrow \texttt{Com}(1, 1^k) : \texttt{com}]\}_{k>0}$

are indistinguishable with respect to algorithms running in time $o(2^{k^\gamma})$*;*
4. **extractability:** *there exists an extractor algorithm* E *running in time* 2^{k^γ} *such that, for all commitments* com *computed by a probabilistic polynomial-time committer adversary* A*, if* A *succeeds in decommitting* com *as* b *with non-negligible probability, then* $E(\texttt{com}) = b$ *with overwhelming probability.*

The above definitions can be easily extended to the case in which we wish to commit to a string (instead of committing to a bit). Such commitment scheme exists, for instance under the assumption that there exist permutations that are one-way with respect to polynomial-time adversaries but such that they can be inverted in subexponential time. In [20], these type of commitment schemes are used in order to achieve straight-line extractability in superpolynomial time.

Finally, we review the notion of a ZAP[21].

Definition 6. *A triple of polynomial-time algorithms* (ZG, ZP, ZV) *is a* ZAP *for the NP-language* L *with polynomial-time relation* R_L *iff:*

1. **Completeness***: given a witness* y *for "*$x \in L$*" and* $z = ZG(1^k)$ *then* $ZV(x, z, ZP(x, y, z)) = 1$ *with probability* 1*.*
2. **Soundness***: for all* $x \notin L$*, with overwhelming probability over* $z = ZG(1^k)$*, there exists no* z' *such that* $ZV(x, z, z') = 1$*.*
3. **Witness-Indistinguishability***: let* y_1, y_2 *such that* $(x, y_1) \in R_L$ *and* $(x, y_2) \in R_L$*. Then* $\forall z$*, the distributions on* $ZP(x, y_1, z)$ *and on* $ZP(x, y_2, z)$ *are computationally indistinguishable.*

In [21] a ZAP is presented under the assumption that non-interactive zero-knowledge proofs exist, thus the existence of ZAPs is implied by the existence of one-way trapdoor permutations.

Since we will need ZAPs to be secure with respect to subexponentially strong adversaries, we need subexponentially strong versions of these assumptions.

4 Three-Round Arguments in the cPK Model

In the cPK model we show that there exist three-round arguments of *membership* for all $\mathcal{NP}$ languages that are concurrently sound and black-box resettable zero knowledge. We stress that a concurrently sound resettable zero-knowledge argument in the BPK model requires at least four rounds (see [14]) while the previously presented three-round protocols require stronger models than the cPK model (see [17,18]).

Our construction assumes the existence of η-secure signature schemes, γ-secure commitment schemes, pseudo-random family of functions (which can be constructed assuming the existence of one-way functions) and the existence of ZAPs secure with respect to subexponential-time adversaries. We use subexponentially strong cryptographic primitives since we crucially use complexity leveraging for our result.

We then show how to obtain three-round arguments of knowledge for all polynomial-time relations that are concurrently sound and black-box concurrent zero-knowledge under standard complexity assumptions. We stress that black-box resettable zero-knowledge arguments of knowledge are not possible [22,14], that concurrent soundness needs four rounds in the BPK model, and that three rounds is optimal for zero-knowledge in any public-key model.

4.1 Three-Round RZK Argument of Membership in the cPK Model

In this section we present our construction for the three-round argument of membership for all $\mathcal{NP}$ in the cPK that is concurrently sound and resettable zero-knowledge. Throughout the section, L will be a fixed $\mathcal{NP}$ language.

Our proof system will follow the FLS paradigm for zero knowledge [23] and we next define the auxiliary language $\Lambda = \Lambda(L)$ that we are going to use.

Definition 7. *The 8-tuple $\tau = (x, c, \tilde{a}_1, \tilde{a}_2, \tilde{b}_1, \tilde{b}_2, k, \mathtt{pk})$ belongs to the language Λ if*

- *$x \in L$ or*
- *there exist $a_1, a_2, \alpha_1, \alpha_2, b_1, b_2, \beta_1, \beta_2$ such that*
  1. *$\mathtt{pk}$ is a public key in the output space of $\mathtt{G}(1^k)$;*
  2. *$a_1 \neq a_2$;*
  3. *$(\tilde{a_1}, \alpha_1) = \mathtt{Com}(a_1, 1^k)$ and $(\tilde{a_2}, \alpha_2) = \mathtt{Com}(a_2, 1^k)$;*
  4. *$(\tilde{b_1}, \beta_1) = \mathtt{Com}(b_1, 1^k)$ and $(\tilde{b_2}, \beta_2) = \mathtt{Com}(b_2, 1^k)$;*
  5. *$\mathtt{Ver}(a_1 \circ c, b_1, \mathtt{pk}) = 1$ and $\mathtt{Ver}(a_2 \circ c, b_2, \mathtt{pk}) = 1$.*

Informally speaking, $\tau = (x, c, \tilde{a}_1, \tilde{a}_2, \tilde{b}_1, \tilde{b}_2, k, \mathtt{pk})$ belongs to Λ if x belongs to L or if, for $i = 1, 2$, $\tilde{b}_i$ is the commitment of a valid signature b_i (with respect to $\mathtt{pk}$) of the concatenation of message a_i committed to by $\tilde{a}_i$ concatenated with c.

Assumptions. In our construction we assume the existence of the following cryptographic tools.

1. an η-secure digital signature scheme $\mathtt{SS} = (\mathtt{G}, \mathtt{Sig}, \mathtt{Ver})$;
2. a γ-secure commitment scheme $(\mathtt{Com}, \mathtt{Dec})$;
3. a pseudo-random family of functions $\mathcal{F}$;
4. a ZAP (ZG, ZV, ZP) for the language Λ.

High-level overview. Let k be the security parameter. The public entry of a verifier contains a public key pk for a secure signature scheme and the first message z of ZAP for Λ. The actual proof that $x \in L$ consists of a first round where the prover sends a random message m to the verifier. The verifier replies with the current value c (in unary) of the counter and a signature $\hat{m}$ of $m \circ c$. The prover first verifies that $\hat{m}$ is a valid signature and then constructs 4 commitments $\tilde{a}_1, \tilde{a}_2, \tilde{b}_1, \tilde{b}_2$ of 0^k. Finally the prover computes the second message of the ZAP in which he proves that $x \in L$ or that $\tilde{b}_1$ and $\tilde{b}_2$ are the commitments of valid signatures b_1, b_2 of messages $a_1 \circ c$ and $a_2 \circ c$ such that $a_1 \neq a_2$ and a_1 and a_2 are the messages committed in $\tilde{a}_1$ and $\tilde{a}_2$.

Let us now informally argue the properties of our construction. For the concurrent soundness we observe that even if the prover opens polynomially many concurrent sessions with the verifier, he will receive signatures of messages relative to different values of the counter. In particular, the prover will never see the signature of two messages with the same value of the counter as suffix. We will use a γ-secure commitment scheme along with a ZAP in order to show that a prover that proves false statements can be used by a superpolynomial-time algorithm in order to break a subexponentially strong assumption. Such a telescopic use of the hardness of different cryptographic assumptions is referred to as *complexity leveraging* and its power is gaining interest. The use of an *extractable* commitment along with a ZAP is also discussed and used in [20].

For the resettable zero knowledge property, the simulator while interacting with the verifier V^* will try, by rewinding $V^\star$, to get signatures for two messages with the same value of the counter as suffix. More precisely, to simulate the proof that $x \in L$, the simulator will first ask for the signature of message $m \circ c$ (where c is the current value of the counter), then he starts a look-ahead (by rewinding $V^\star$) in order to obtain the signature of a new message $m' \circ c$. Once the signature is obtained, the look-ahead ends and the simulator goes back to the previous original execution since it is now able to successfully run the third round.

The crucial observation to show that the simulation ends in expected polynomial time is that the values of the counters cannot be greater than the running time of the adversarial verifier (since $V^\star$ sends the value of the counter in unary). More precisely, each look-ahead starts after the first signature corresponding to a given counter value and to a given public key has been received by S. Since the number of public keys is polynomially bounded (the size of the public file does not change after the preprocessing stage) and the running-time of an (s,t)-resetting verifier is bounded by the polynomial t, we have that the number of look-aheads is polynomial. Moreover, each look-ahead starts because a given counter value has been sent by $V^\star$ on input a given transcript. Therefore, the expected number of rewinds that will be needed in order to obtain again the same counter (in the look-ahead the corresponding signature is asked for a different message) is the inverse of the probability that $V^\star$ plays such a value. Finally, by observing that the simulation between two rewinds can be run by S in polynomial time, we have that the simulator runs in expected polynomial-time.

Formal description. Let k be the security parameter. The verifier runs the key generator $\mathtt{G}$ for the η-secure signature scheme on input 1^k obtaining the pair $(\mathtt{pk}, \mathtt{sk})$. Moreover, the verifier runs ZG on input 1^k and obtains z that will be used as the first round of the ZAP for the language Λ. The entry of the verifier in the public file consists of the pair $(\mathtt{pk}, z)$. The private key $\mathtt{sk}$ associated with $\mathtt{pk}$ is kept secret by the verifier. Moreover, the verifier initializes her counter c by setting $c = 0$. The protocol is found in Figure 1.

Common input: security parameter k, public file $F\{(\mathtt{pk}, z)\}$ and instance x.
P's private input: a witness y for $x \in L$.
V's private input: private key $\mathtt{sk}$ and a counter c.

P-round-1:

1. randomly pick seed s;
2. compute $w = \mathcal{F}(s, x \circ y \circ \mathtt{pk})$ and use w as randomness;
3. randomly pick $m \leftarrow \{0,1\}^k$ and sends it to V;

V-round-2:

1. increment c;
2. compute $\hat{m_c} = \mathtt{Sig}(m \circ c, \mathtt{sk})$;
3. send $(1^c, \hat{m_c})$ to P;

P-round-3:

1. verify that $\mathtt{Ver}(m \circ c, \hat{m_c}, \mathtt{pk}) = 1$;
2. randomly pick seed s';
3. compute $w' = \mathcal{F}(s', x \circ y \circ \mathtt{pk} \circ \hat{m_c} \circ c)$ and use w' as randomness;
4. use a γ-secure commitment function $\mathtt{Com}$ to compute commitments $\tilde{a}_1, \tilde{a}_2, \tilde{b}_1, \tilde{b}_2$ where $\tilde{a}_j = \mathtt{Com}(0^k, 1^{k_1})$ and $\tilde{b}_j = \mathtt{Com}(0^k, 1^{k_1})$ for $j = 1, 2$;
5. compute $Z = ZP((x, c, \tilde{a}_1, \tilde{a}_2, \tilde{b}_1, \tilde{b}_2, k, \mathtt{pk}), y, z)$;
6. send $\tilde{a}_1, \tilde{a}_2, \tilde{b}_1, \tilde{b}_2$ and Z to V;

V-decision: verify that the ZAP is valid by running ZV on input instance $\tau = (x, c, \tilde{a}_1, \tilde{a}_2, \tilde{b}_1, \tilde{b}_2, k, \mathtt{pk})$, initial ZAP-message z and ZAP-reply Z.

Fig. 1. The 3-round concurrently sound rZK argument for $\mathcal{NP}$ in the $\mathtt{cPK}$ model. The value k_1 is chosen based on η, γ and k (see the proof of concurrent soundness).

Theorem 1. *If, for some positive η and γ, there exist an η-secure digital signature scheme, a γ-secure commitment scheme and sub-exponentially strong ZAPs for all $\mathcal{NP}$ languages then there exists a* 3-round concurrently sound resettable zero-knowledge *argument system for any language $L \in \mathcal{NP}$ in the* $\mathtt{cPK}$ *model.*

Proof. Consider the protocol in Figure 1. Completeness follows by inspection.

Concurrent soundness. Assume by contradiction that the protocol is not concurrently sound; then there exists a malicious prover $P^\star$ that in a concurrent attack has a non-negligible probability of making verifier V accept x for some $x \notin L$.

We assume we know the session $j^\star$ of the verifier in which the prover will succeed in cheating. This assumption will be later removed. In order to obtain a contradiction we show an algorithm $\mathcal{A}$ running in time $o(2^{k^\eta})$ that breaks the η-secure digital signature scheme $\mathtt{SS}$ used in the construction. Algorithm $\mathcal{A}$ receives a signature public key $\mathtt{pk}$, obtained by running $\mathtt{G}$ on input 1^k, has access to $P^\star$ and to a signing oracle $\mathcal{O}$ for a public key $\mathtt{pk}$ and outputs a signature of a message for which the oracle had not been queried.

We now describe algorithm $\mathcal{A}$. On input challenge public key $\mathtt{pk}$, $\mathcal{A}$ performs the set-up procedure and builds the public entry of the verifier as $(\mathtt{pk}, z)$ where z is the output of algorithm ZG on input 1^k. Algorithm $\mathcal{A}$ starts the interaction with the prover $P^\star$ and, for all sessions j constructs the message to be sent in the second round of the protocol by following the verifier's algorithm and by resorting to the oracle $\mathcal{O}$ to compute signatures of messages $m \circ c$, for messages m received from $P^\star$. At the end of session $j^\star$, since $x \notin L$ then, by the soundness of the ZAP (ZG, ZP, ZV), it must be the case that $P^\star$ has exhibited commitments $\tilde{a}_1, \tilde{a}_2$ of two different messages a_1, a_2 and commitments $\tilde{b}_1, \tilde{b}_2$ of two signatures b_1, b_2 such that, b_1 is a signature of a_1 and b_2 is a signature of a_2. Moreover, messages a_1 and a_2 have the value of the counter chosen by $P^\star$ for the $j^\star$-th session as suffix. Then $\mathcal{A}$ in time $O(\mathtt{POLY}(k_1) \cdot 2^{{k_1}^\gamma})$ breaks the secrecy of the commitments and obtains the two messages along with their corresponding signatures. Now, $\mathcal{A}$ has queried the oracle for public key $\mathtt{pk}$ once for each value of the counter (we remind the reader that the adversary $P^\star$, for each value of the counter, is allowed to run the verifier at most once) and thus $\mathcal{A}$ has not queried the oracle for at least one of $a_1 \circ c$ or $a_2 \circ c$. By picking k_1 such that $k_1^\gamma < k^\eta$, we have that $\mathcal{A}$ runs in time $o(2^{k^\eta})$ and we have reached a contradiction.

In our proof we assumed that $\mathcal{A}$ knows the value $j^\star$. If this is not the case that $\mathcal{A}$ can simply guess the value and the same analysis applies since this decreases only by a polynomial factor the probability of breaking the digital signature scheme. Moreover $\mathcal{A}$ can also try to break all the commitments of all sessions, since the running time will still be $o(2^{k^\eta})$.

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

The simulator S receives from $V^\star$ requests that can be described wlog by a quadruple (x, i, r, v), where x denotes the input instance for language L, i denotes the index of the public key with respect to which the interaction has to be simulated, r is the index of the random tape that must be used in the simulation, v is the index of the message that S must send (and, for our specific protocol, $v = 1$ or $v = 3$). We remark that the resetting adversary $V^\star$ is allowed to reset the prover to any previous state and even request that a different random tape has to be used (however, $V^\star$ is not allowed to feed the prover with a random tape of its choice).

Simulator S maintains several data structures which will be implicitly used in its description below and performs all the consistency checks requested by the protocol (for example, that the signatures received are valid). In addition, S also builds a table $S(i, c)$ of signatures with entry (i, c) holding signatures of messages with suffix c computed with respect to the i-th public key.

The interaction between $V^\star$ and S consists of essentially four types of requests. Indeed, two rounds of the argument system are played by the prover and we distinguish two cases depending on whether or not the same round has been requested since the last rewind by the verifier.

1. The request is $(x, i, r, 1)$ and it is the first such request since last rewind. In this case, S follows exactly the prover's algorithm using the r-th random tape as source of randomness.
2. The request is $(x, i, r, 1)$ and such request has already been presented to S by $V^\star$ since last rewind. In this case, S re-plays the same message used in the previous round.
3. The request is $(x, i, r, 3)$ and S has already received such a request since the last rewind. In this case, S re-plays the same message used in the previous round.
4. The request is $(x, i, r, 3)$ and S has not received such a request since the last rewind. Let c be the value of the counter declared by $V^\star$ in the message just preceding the request. We have two sub-cases:
   (a) $S(i, c)$ contains two (or more) signatures. In this case S uses two signatures from $S(i, c)$ as witness to compute the last message of the ZAP.
   (b) $S(i, c)$ contains one signature. S has obtained the signature in the second round played by $V^\star$ (the case in which $|S(i, c)| = \emptyset$ and S has to play the third round cannot happen). In this case S needs to obtain a second signature with suffix c in order to be able to compute the last message of the ZAP. Thus, S starts a look-ahead for (i, c). More precisely, S rewinds $V^\star$ to the state just after he has sent the first request $(x, i, r, 1)$ (notice that since $V^\star$ is a resetting adversary, there could be several such requests) and uses a new random string r' instead of r. S will repeat such a rewind strategy until a rewind ends by appending a second entry to (i, c). As we shall argue, the simulation will halt in expected polynomial-time as the number of pairs (i, c) (and thus the number of look-aheads) is bounded by $s(n)t(n)$ which is a polynomial (we are considering by definition an (s, t)-resetting verifier).

The views are indistinguishable. The first message played by S in each session has exactly the same distribution of the one played by the prover since S simply runs the prover's algorithm. We stress that even though after each rewind S changes one randomness r_g for a given $g \in \{1, \ldots, s(n)\}$, $V^\star$ is not aware of such an update since its view does not go back with respect to the last rewind.

The third round played by S has the following two differences with respect to the one played by the prover.

1. The prover commits to junk bits (the 0^k strings) while the simulator commits to a pair of different messages with the same suffix, along with their signatures with respect to a given public key. By the computational hiding of the commitment scheme, $V^\star$ does not distinguish the commitments of S from the commitments of the prover. More formally, if $V^\star$ distinguishes with non-negligible probability the commitments of S from the commitment of P, then $V^\star$ can be used to contradict the hiding of the commitment scheme.
2. The prover uses y such that $(x, y) \in R_L$ in order to compute the auxiliary witness for running ZP on input the auxiliary statement "$\tau \in \Lambda$". Instead, the simulator uses his knowledge of the different messages with c as suffix along with their signatures to compute the auxiliary witness for "$\tau \in \Lambda$". Both the prover and S follows the honest prover algorithm for the ZAP by running ZP on a good witness for the auxiliary statement. Therefore, an adversarial verifier $V^\star$ that distinguishes the witness used by the simulator from the one used by an honest prover can be used to contradict the witness-indistinguishability of the ZAP. Note that in our implementation of the ZAP the prover uses as randomness a pseudorandom string of both z and the message sent by the verifier. Therefore, as remarked in [21,22], this implementation of ZAP preserves witness-indistinguishability even in case of reset attacks, i.e., the implemented ZAP is a resettable witness-indistinguishable proof system.

The simulation ends in polynomial time. S has to compute two messages for each session. Note that for the first message, S always performs a straight-line simulation since the first round of a session is played by running the prover algorithm, and since no witness is needed, it can be computed by S without rewinding $V^\star$.

The analysis is more complicated for the second message. First of all, observe that the simulator starts a new look-ahead procedure only after receiving a request $(x, i, r, 3)$. Such a request is immediately preceded by a message from $V^\star$ containing one valid signature for a pair (i, c) for which $S(i, c)$ was empty before the request (for otherwise, no look-ahead procedure would be started since S has at least 2 valid signatures). In other words, the simulator starts a look-ahead procedure only after receiving a useful signature. However, observe that both the number of entries in the public file (and thus the number of possible values of i) and the number of possible values of the counter are bounded by the running time of the adversary $V^\star$ that is assumed to be polynomially bounded. Next, we argue that the contribution of each entry to the expected work of the simulator is also polynomially bounded. Roughly speaking, the contribution of each pair (i, c) is equal to the probability that counter c appears in a session with public key $\mathtt{pk}_i$ times the number of rewinds needed to have a new session with the same public key and the same value of the counter in which we ask for the signature of a different message. It is easy to see that this quantity is polynomially bounded.

4.2 Three-Round CZK Argument of Knowledge in the cPK Model

In this section we present a three-round concurrently-sound concurrent zero-knowledge argument of *knowledge* in cPK for any language $L \in \mathcal{NP}$ under standard intractability assumptions.

The argument of knowledge that we present is derived from the argument of membership of the previous section by replacing the ZAP with a 3-round witness-indistinguishable proof system to prove a statement of the form "$\alpha \vee \beta$" where α is known at the beginning of the protocol while β is known only in the last round. Additionally, we need witness extraction with respect to α. An implementation of this primitive can be given by using a variation of the protocol presented in [25] (this is also used in [26]).

To prove the properties of our construction, we assume the existence of signature and commitment schemes secure with respect to polynomial-time adversaries, and the existence of the mentioned 3-round witness-indistinguishable proof system of membership π for any NP language (which can, in turn, be based on the existence of one-way permutations). We use this proof system as a black box and denote the messages computed in it as (wi_1, wi_2, wi_3), where wi_1 is sent by the prover, wi_2 is sent be the verifier and wi_3 is sent by the prover again. Note that in order to prove that $x \in L$, for some NP language L, the message wi_1 can be computed in polynomial time given only the length value $|x|$ (that is, neither x nor a witness for it is necessary).

The public file. Let k be the security parameter. The i-th entry of the public file F consists of a randomly generated public key $\mathtt{pk}$ with security parameter k for the secure signature scheme $\mathtt{SS}$ and of the first round of a two-round computationally-binding perfectly-hiding commitment scheme

Private inputs. For the statements "$x \in L$", the private input of the prover consists of a witness y for $x \in L$. The private input of the verifier consists of the private key $\mathtt{sk}$ corresponding to the public key $\mathtt{pk}$ and a counter c.

The protocol. Suppose that the prover wants to prove that $x \in L$ and that the verifier knows the private key $\mathtt{sk}$ of the i-th public key $\mathtt{pk}$ of the public file F.

In the first round P randomly picks string m of length k, computes wi_1 according to the proof system π, and sends the pair (m, π) to V. Then V increments c and uses the private key $\mathtt{sk}$ to compute a digital signature $\hat{m_c}$ of $m \circ c$, computes message wi_2 and sends the triplet $(1^c, \hat{m_c}, wi_2)$ to P. In the last round P verifies that $\hat{m_c}$ is a valid signature of $m \circ c$ with $\mathtt{pk}$ and computes the commitments $\tilde{a}_1, \tilde{a}_2, \tilde{b}_1, \tilde{b}_2$ of 0^k. Then P computes message wi_3 according to proof system π by using instance $(x, c, \tilde{a}_1, \tilde{a}_2, \tilde{b}_1, \tilde{b}_2, k, \mathtt{pk})$ and string y as the input and witness for π, respectively. P sends $wi_3, \tilde{a}_1, \tilde{a}_2, \tilde{b}_1, \tilde{b}_2$ to V. Finally, V verifies that (wi_1, wi_2, wi_3) is valid by running the verifier's accepting predicate in proof system π, using as input the instance $(x, c, \tilde{a}_1, \tilde{a}_2, \tilde{b}_1, \tilde{b}_2, k, \mathtt{pk})$.

Theorem 2. *If there exist one-way permutations, then there exists a three-round concurrently-sound concurrent zero-knowledge argument of knowledge for $\mathcal{NP}$ in the* cPK *model.*

The proof of Theorem 2 is omitted from this extended abstract.

5 Conclusion

In this paper we have presented a 3-round concurrently-sound resettable zero-knowledge argument system in the cPK model that improves the previous works of Micali and Reyzin [17] and Zhao *et al.* [18]. The cPK model is only a slight variation of the BPK model, and we have shown that it can be used to go beyond the barrier of four rounds needed for concurrent soundness in the BPK model. Our result makes a big step for closing the gap between a public-key model that admits three-round concurrently-sound resettable zero-knowledge arguments and the BPK model. Moreover, we have shown a 3-round concurrently-sound concurrent zero-knowledge argument of knowledge in the cPK model under standard intractability assumptions.

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), ACM (1998) 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), ACM (2000) 235–244
4. Richardson, R., Kilian, J.: On the Concurrent Composition of Zero-Knowledge Proofs. In: Advances in Cryptology – Eurocrypt '99. Volume 1592 of LNCS, (1999) 415–431
5. Kilian, J., Petrank, E.: Concurrent and Resettable Zero-Knowledge in Poly-Logarithmic Rounds. In: Proceedings of the 33rd ACM Symposium on Theory of Computing (STOC '01), ACM (2001) 560–569
6. Kilian, J., Petrank, E., Rackoff, C.: Lower Bounds for Zero Knowledge on the Internet. In: Proceedings of the 39th Symposium on Foundations of Computer Science, (FOCS '98). (1998) 484–492
7. 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), ACM (2001) 570–579
8. Dwork, C., Sahai, A.: Concurrent Zero-Knowledge: Reducing the Need for Timing Constraints. In Krawczyk, H., ed.: Advances in Cryptology – Crypto '98. Volume 1462 of LNCS, (1998) 442–457
9. 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
10. Blum, M., De Santis, A., Micali, S., Persiano, G.: Non-Interactive Zero-Knowledge. SIAM J. on Computing **20** (1991) 1084–1118
11. Damgard, I.: Efficient Concurrent Zero-Knowledge in the Auxiliary String Model. In: Advances in Cryptology – Eurocrypt '00. Volume 1807 of LNCS, (2000) 418–430

12. Di Crescenzo, G., Removing Complexity Assumptions from Concurrent Zero-Knowledge Proofs, In: Proc. of Cocoon 2000. LNCS (2000).
13. Di Crescenzo, G., Ostrovsky, R., On Concurrent Zero-Knowledge with Pre-processing, In: Advances in Cryptology – Crypto '99. Volume 1666 of LNCS, (1999)
14. Micali, S., Reyzin, L.: Soundness in the Public-Key Model. In: Advances in Cryptology – Crypto '01. Volume 2139 of LNCS, (2001) 542–565
15. Di Crescenzo, G., Persiano, G., Visconti, I.: Constant-Round Resettable Zero Knowledge with Concurrent Soundness in the Bare Public-Key Model. In: Advances in Cryptology – Crypto '04. Volume 3152 of LNCS, (2004) 237–253
16. Reyzin, L.: Zero-Knowledge with Public Keys, Ph.D. Thesis. MIT (2001)
17. Micali, S., Reyzin, L.: Min-Round Resettable Zero-Knowledge in the Public-key Model. In: Advances in Cryptology – Eurocrypt '01. Volume 2045 of LNCS, (2001) 373–393
18. Zhao, Y., Deng, X., Lee, C., Zhu, H.: ResettableZero-Knowledge in the Weak Public-Key Model. In: Advances in Cryptology – Eurocrypt '03. Volume 2045 of LNCS, (2003) 123–139
19. 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
20. Pass, R.: Simulation in Quasi-Polynomial Time and Its Applications to Protocol Composition. In: Advances in Cryptology – Eurocrypt '03. Volume 2045 of LNCS, (2003) 160–176
21. Dwork, C., Naor, M.: Zaps and their applications. In: IEEE Symposium on Foundations of Computer Science. (2000) 283–293
22. 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), (2001) 116–125
23. Feige, U., Lapidot, D., Shamir, A.: Multiple Non-Interactive Zero Knowledge Proofs Under General Assumptions. SIAM J. on Computing **29** (1999) 1–28
24. De Santis, A., Persiano, G.: Zero-Knowledge Proofs of Knowledge Without Interaction. In: Proceedings of the 33rd Symposium on Foundations of Computer Science, (FOCS '92). (1992) 427–436
25. Lapidot, D., Shamir, A.: Publicly Verifiable Non-Interactive Zero-Knowledge Proofs. In: Advances in Cryptology – Crypto '90. Volume 537 of LNCS, (1991) 353–365
26. Katz, J., Ostrovsky, R.: Round-Optimal Secure Two-Party Computation. In: Advances in Cryptology – Crypto '04. Volume 3152 of LNCS, (2004).