Interactive Zero-Knowledge with Restricted Random Oracles

Moti Yung[1] and Yunlei Zhao[2]

[1] RSA Laboratories and Department of Computer Science, Columbia University, New York, NY, USA. moti@cs.columbia.edu
[2] Software School, School of Information Science and Engineering, Fudan University, Shanghai 200433, China. ylzhao@fudan.edu.cn

Abstract. We investigate the design and proofs of zero-knowledge (ZK) interactive systems under what we call the "restricted random oracle model" which restrains the usage of the oracle in the protocol design to that of collapsing protocol rounds a la Fiat-Shamir heuristics, and limits the oracle programmability in the security proofs. We analyze subtleties resulting from the involvement of random oracles in the interactive setting and derive our methodology. Then we investigate the Feige-Shamir 4-round ZK argument for $\mathcal{NP}$ in this model: First we show that a 2-round protocol is possible for a very interesting set of languages; we then show that while the original protocol is not concurrently secure in the public-key model, a modified protocol in our model is, in fact, concurrently secure in the bare public-key model. We point at applications and implications of this fact. Of possible independent interest is a concurrent attack against the Feige-Shamir ZK in the public-key model (for which it was not originally designed).

1 Introduction

The basic random oracle (RO) methodology was originally introduced in [2] as an idealization and abstraction of the Fiat-Shamir heuristics [14] that transforms Σ-protocols (i.e., 3-round public-coin special honest verifier zero-knowledge SHVZK protocols) into signature schemes. The methodology was used to achieve non-interactive schemes (signatures, public-key encryption and non-interactive zero-knowledge NIZK). However, nowadays more and more complicated *interactive* protocols are developed employing random oracle (one example is the recent direct anonymous attestation (DAA) scheme [3] developed for industrial use).

In this work, we show that the design of *interactive* schemes with advanced security notions using the normal random oracle methodology is more subtle than is typically believed. The subtlety lies in the programmability of the RO in security proofs. Namely, in security proofs the simulator defines output of query values and in fact "programs" the RO on any query, in particular *any query it chooses*. We further investigate the usages of RO in protocol designs. For a protocol developed in the RO model, we consider the sensitivity of the security of the protocol when the RO is replaced (realized) by practical (hash)

functions, showing that different usages lead to different sensitivities (say, to future cryptanalysis of the function).

We attempt to minimize the usage of the ROs both in **protocol designs** and in **security proofs**. This is naturally desired property. In our approach we impose the following restrictions:

- We limit the usage of ROs in security proofs by letting the honest player and the simulator (who plays the role of the honest player) use a *non-programmable* RO.
- Furthermore, the non-programmable RO could be replaced by *any* hash function without compromising the security of the *honest* player.
- Finally, we limit the usage of ROs in protocol designs to collapsing Σ-protocols just as in the original Fiat-Shamir methodology.

We note that on one hand, it is desirable to minimize using the truly random function property of real hash functions in protocol designs (due to its idealized nature and its unrealization within certain constructions, e.g., [5, 6, 22, 18, 1]). On the other hand, we justify the original Fiat-Shamir methodology by the simple (yet, we believe important) observation that even very weak hash functions (clearly not collision-resistant and not pseudorandom) can be used to collapse Σ-protocols into non-interactive ones with remarkable security guarantee (not ZK but nevertheless a useful property that can be employed).

In this work, we refer to protocols, which are developed with limiting ROs in security proofs and protocols designs according to our approach (which we motivate herein), as *protocols with restricted ROs*.

Although our approach of restricted ROs is seemingly very limiting, it turns out to be still very powerful for achieving practical *interactive* cryptographic schemes with a (seemingly) better balance between "(idealized) provable" security and implementation efficiency. In particular, we show that a Σ_{OR}-based practical implementation (without $\mathcal{NP}$-reductions) of the Feige-Shamir 4-round ZK arguments (the version that appears in [13]) can give us *generic yet practical* 2-round (that is optimal) ZK systems with restricted ROs.

We then investigate the *concurrent security* of the Feige-Shamir ZK protocol. We show that (perhaps surprisingly) the Feige-Shamir ZK protocol is, in general, *not* concurrently secure in public-key settings (when users possess public keys), by identifying a concurrent attack. Though it may look quite natural to run the Feige-Shamir ZK protocol in public-key models when it is used in practice and perhaps to do it concurrently (even though the protocol was not designed for concurrency), our attack shows that this intuition is wrong. Fortunately, the Feige-Shamir ZK protocol is concurrently secure with restricted ROs in the bare public-key (BPK) model. In this process, we also identify and clarify complications and subtleties in dealing with concurrent adversaries in the setting of *interactive* zero-knowledge with restricted ROs.

We remark that the 2-round ZK systems with restricted ROs (with or without registered public-keys) can be used to construct more complicated interactive systems with restricted ROs. It can also be used to transform a large number of existing interactive schemes, which are developed originally with the normal

random oracle methodology, into schemes with restricted ROs, by adding *at most* one additional round but with seemingly stronger security guarantees.

Finally, as part of this work, two constructions are given which are of independent interest and may be worthy of further explorations: a one-round witness-hiding (WH) protocol for DLP, and a concurrent attack against the Feige-Shamir ZK when it is run in the new setting of public-key models.

2 Preliminaries

In this section we review the major cryptographic tools used, and present a key observation on non-interactive Σ_{OR}-protocols with ROs.

Definition 1 (Σ-protocol [7]). *A 3-round public-coin protocol $\langle P, V\rangle$ is said to be a Σ-protocol for a relation R if the following hold:*

- *Completeness. If P, V follow the protocol, the verifier always accepts.*
- *Special soundness. From any common input x of length n and any pair of accepting conversations on input x, (a, e, z) and (a, e', z') where $e \neq e'$, one can efficiently computes w such that $(x, w) \in R$. Here a, e, z stand for the first, the second and the third message respectively and e is assumed to be a string of length t (that is polynomially related to n) selected uniformly at random in $\{0,1\}^t$.*
- *Special honest verifier zero-knowledge (SHVZK). There exists a probabilistic polynomial-time (PPT) simulator S, which on input x and a random challenge string e, outputs an accepting conversation of the form (a, e, z), with the same probability distribution as the real conversation between the honest P, V on input x.*

Σ-protocols have been proved to be a very powerful cryptographic tool and are widely used. Many basic Σ-protocols have been developed, and the following are Σ-protocol examples for DLP and RSA.

Σ-Protocol for DLP [24]. The following is a Σ-protocol $\langle P, V\rangle$ proposed by Schnorr [24] for proving the knowledge of discrete logarithm, w, for a common input of the form (p, q, g, h) such that $h = g^w \ mod\ p$, where on a security parameter n, p is a uniformly selected n-bit prime such that $q = (p-1)/2$ is also a prime, g is an element in $\mathbf{Z}_p^*$ of order q. It is also actually the first efficient Σ-protocol proposed in the literature.

- P chooses r at random in $\mathbf{Z}_q$ and sends $a = g^r \ mod\ p$ to V.
- V chooses a challenge e at random in $\mathbf{Z}_{2^t}$ and sends it to P. Here, t is fixed such that $2^t < q$.
- P sends $z = r + ew \ mod\ q$ to V, who checks that $g^z = ah^e \ mod\ p$, that p, q are prime and that g, h have order q, and accepts iff this is the case.

Σ-Protocol for RSA [19]. Let n be an RSA modulus and q be a prime. Assume we are given some element $y \in Z_n^*$, and P knows an element w such that $w^q = y \mod n$. The following protocol is a Σ-protocol for proving the knowledge of q-th roots modulo n.

- P chooses r at random in Z_n^* and sends $a = r^q \mod n$ to V.
- V chooses a challenge e at random in Z_{2^t} and sends it to P. Here, t is fixed such that $2^t < q$.
- P sends $z = rw^e \mod n$ to V, who checks that $z^q = ay^e \mod n$, that q is a prime, that $gcd(a, n) = gcd(y, n) = 1$, and accepts iff this is the case.

The OR-proof of Σ-protocols [8]. One basic construction with Σ-protocols allows a prover to show that given two inputs x_0, x_1, it knows a w such that either $(x_0, w) \in R_0$ or $(x_1, w) \in R_1$, but without revealing which is the case. Specifically, given two Σ-protocols $\langle P_b, V_b\rangle$ for R_b, $b \in \{0, 1\}$, with random challenges of (without loss of generality) the same length t, consider the following protocol $\langle P, V\rangle$, which we call Σ_{OR}. The common input of $\langle P, V\rangle$ is (x_0, x_1) and P has a private input w such that $(x_b, w) \in R_b$.

- P computes the first message a_b in $\langle P_b, V_b\rangle$, using x_b, w as private inputs. P chooses e_{1-b} at random, runs the SHVZK simulator of $\langle P_{1-b}, V_{1-b}\rangle$ on input (x_{1-b}, e_{1-b}), and let $(a_{1-b}, e_{1-b}, z_{1-b})$ be the output. P finally sends a_0, a_1 to V.
- V chooses a random t-bit string s and sends it to P.
- P sets $e_b = s \oplus e_{1-b}$ and computes the answer z_b to challenge e_b using (x_b, a_b, e_b, w) as input. He sends (e_0, z_0, e_1, z_1) to V.
- V checks that $s = e_0 \oplus e_1$ and that conversations (a_0, e_0, z_o), (a_1, e_1, z_1) are accepting conversations with respect to inputs x_0, x_1, respectively.

Theorem 1. [8] *The protocol Σ_{OR} above is a Σ-protocol for R_{OR}, where $R_{OR} = \{((x_0, x_1), w) | (x_0, w) \in R_0 \text{ or } (x_1, w) \in R_1\}$. Moreover, for any malicious verifier V^*, the probability distribution of conversations between P and V^*, where w is such that $(x_b, w) \in R_b$, is independent of b. That is, Σ_{OR} is perfectly witness indistinguishable (WI).*

Given access to a random oracle (RO) $\mathcal{O}$, we can transform a Σ-protocol into a non-interactive protocol, which in this work we call non-interactive Σ-protocol in the RO model. On a common input x, an auxiliary input aux and a private witness w, the prover then generates the first message a, queries $\mathcal{O}$ with (x, a, aux) to get the challenge e and then computes the answer z. The proof is then (a, z, aux). To verify such a proof, query $\mathcal{O}$ with (x, a, aux) to get e and then run the verifier of the original Σ-protocol. The transformed non-interactive protocol is *zero-knowledge proof of knowledge* in the random oracle model [2, 25]. The key observation here (which is simple yet powerful) is the following claim:

Claim 1. *The non-interactive Σ_{OR}-protocol in the RO model remains witness-indistinguishable even if the random oracle is replaced by any* real *hash function.*

Proof. Note that the WI property of Σ_{OR} is with respect to *any* malicious verifier. In particular, the WI property holds for a specific malicious verifier that uses a real hash function to generate the challenge in Round-2.

Common input. An element $x \in L$ of length n, where L is an $\mathcal{NP}$-language that admits Σ-protocols.

P's private input. A witness w for $x \in L$.

Random oracle. A random oracle denoted $\mathcal{O}$.

Round-1. The verifier V selects a OWF f_V that admits Σ-protocols, randomly selects two elements in the domain of f_V, x_V^0 and x_V^1, computes $y_V^0 = f_V(x_V^0)$ and $y_V^1 = f_V(x_V^1)$, randomly selects a bit b from $\{0,1\}$, sends to the prover a non-interactive Σ_{OR}-proof on (y_V^0, y_V^1), denoted $\pi_V = (y_V^0, y_V^1, a_V, e_V, z_V, aux_V)$, that it knows either the preimage of y_V^0 or the preimage of y_V^1. The witness used by V in forming π_V is x_V^b. The random challenge e_V is generated by querying $\mathcal{O}$ with $(x, y_V^0, y_V^1, a_V, aux_V)$ where aux_V is the auxiliary information of V that possibly includes a time stamp.

Round-2. The prover P first checks the validity of π_V and aborts if it is not valid. Otherwise, P randomly select a bit b' from $\{0,1\}$, sends back a non-interactive Σ_{OR}-proof on (x, y_V^0, y_V^1, b'), denoted $\pi_P = (a_P, e_P, z_P, b')$, that it knows either a witness for $x \in L$ or the preimage of $y_V^{b'}$. The witness used by P is its private input w. The random challenge e_P is generated by querying the random oracle with (x, a_P, b', π_V).

Verifier's Decision. The verifier checks the validity of π_P and accepts if it is valid, otherwise it rejects.

Figure-1. An insecure ZK protocol in the normal random oracle model

3 Restricted ROs: Motivation and Discussions

In this section, we first provide some motivating examples along with discussions and comments. Then, in light of the motivating examples and discussions, we give some desirable principles for limiting the uses of ROs in security proofs and protocol designs that are naturally derived from the motivation.

3.1 Motivating examples and discussions

A motivating example for limiting the uses of RO in security proofs. Consider the following protocol depicted in Figure-1.

We note that the zero-knowledge property of the protocol of Figure-1 can be easily shown *in the normal random oracle model*, where the ZK simulator simulates (programs) the RO (i.e, the simulator defines the outputs of the random oracle on any queries, in particular *any queries it chooses*). The proof is omitted here due to space limitation. But we argue that this protocol (though simulatable (in the programmable oracle case, which is what the proof shows) is not really ZK even in the random oracle model. Observe the following attack: On an input y_{V^*} that the adversary V^* does not know the preimage of under the OWF f_{V^*} (selected by V^* in Round-1), the adversary V^* additionally randomly selects a x'_{V^*} from the domain of f_{V^*} and computes $y'_{V^*} = f_{V^*}(x'_{V^*})$. Then, V^*

randomly selects a bit b from $\{0,1\}$, sets $y^b_{V^*}$ to be y'_{V^*} and $y^{1-b}_{V^*}$ to be y_{V^*}. Finally, V^* sends to the honest prover a non-interactive Σ_{OR}-proof on $(y^0_{V^*}, y^1_{V^*})$ that it knows either the preimage of $y^0_{V^*}$ or the preimage $y^1_{V^*}$ by using x'_{V^*} as the witness. Then according to the perfect witness indistinguishability of Round-1, with probability $1/2$ the honest prover will select $b' = 1 - b$ in Round-2. That is, with probability $1/2$ V^* will get a non-interactive Σ_{OR}-proof for showing the knowledge of either the witness for $x \in L$ or the preimage of $y^{1-b}_{V^*} = y_{V^*}$, a knowledge of such a witness cannot be generated by V^* alone without interacting with P. In other words, the honest prover divulges (seemingly significantly) valuable "knowledge" to the above malicious verifier.

Note that the above protocol is a very simple interactive protocol in the random oracle model, and so we can easily identify the above simple attack. When we construct much more complicated *interactive* schemes in the random oracle model, the security analyses might be much more complicated and subtle. In light of the above attack, we believe that for interactive protocols in the random oracle model, letting the ZK simulator define the random outputs of the random oracle on queries *it chooses* may be too artificial to reflect the real power of the malicious verifier even in the random oracle model.

According to the above arguments, for proving the security of *interactive* protocols in the random oracle model we should restrict the power of the simulator in defining the random outputs of the random oracle. Also note that in a much more complicated interactive scheme in the random oracle model, where both the prover and the verifier prove using the non-interactive Σ-protocols, the provers may actually be the verifier of the high-level complicated interactive protocol.

Comment: Note that it is easy to check that the protocol depicted in Figure-1 is not zero-knowledge if the simulator (who plays the role of the honest prover) uses a non-programmable random oracle.

Motivating examples for limiting the uses of RO in protocol designs.

Random oracles have been employed in many ways. Consider, for example, the following commitment scheme in the random oracle model employed in [23]: To commit to a message m, the commitment sender randomly picks a random string r and sends $RO(x, r)$. The security (both binding and hiding) can be easily checked in the random oracle model. But in practice when the random oracle RO is replaced by real practical hash functions, the security properties of this commitment scheme are (critically) sensitive to the realization. The zero-knowledge protocol with ROs developed in [23] critically uses the above commitment scheme. This means that for a large complicated cryptographic system built with the ZK protocol of [23] as a building block, the security of the *whole* system will also be (critically) dependent on the assumed random-function property of the underlying hash functions. This means it will be critically sensitive to the realizations of the underlying practical hash functions used, which we would like to avoid in certain critical settings (e.g., if the realization is cryptanalyzed in the future and the binding property is lost, say).

So, what uses of random oracle are less sensitive to the case that the hash function realizing it is cryptanalyzed (perhaps in the future)? We justify the original Fiat-Shamir methodology by showing that even very weak hash functions can be used to collapse Σ-protocols into non-interactive ones with remarkable security guarantee. Consider the following one-round witness hiding (WH) protocol $\langle P, V\rangle$ for DLP.

Common input. (p, q, g, h), where on a security parameter n, p is a uniformly selected n-bit prime such that $q = (p-1)/2$ is also a prime, g and h are elements in $\mathbf{Z}_p^*$ of order q.

P's private input. w such that $h = g^w \ mod\ p$.

The protocol. P chooses r at random in $\mathbf{Z}_q$, computes $a = g^r \mod p$. If a is an even number then let $e = \frac{1}{2}a \mod p$ and if a is an odd number then let $e = \frac{1}{2}(a-1) \mod p$ (this guarantees that $e \in \mathbf{Z}_q$). Then P computes $z = r + ew \mod p$. Finally P sends (a, z) to V.

V's decision. V computes e from a and checks that $g^z = ah^e \ mod\ p$, that p, q are prime and that g, h, a have order q, and accepts iff this is the case.

The above protocol can be viewed as the non-interactive version of Schnorr's Σ-protocol for DLP [24] when the random oracle is replaced by the following hash function H: for any strings x, y in $\{0,1\}^*$ and $e \in \mathbf{Z}_q$, $H(xe0) = H(ye1) = e$. Clearly this hash function is not collision-resistant and not pseudorandom. But this extremely weak hash function still provides remarkable security guarantee for the above transformed non-interactive protocol.

We first note that under the DLP hardness assumption, the above non-interactive protocol is witness hiding (WH) for DLP. Specifically, suppose with non-negligible probability a PPT adversary can produce w from (a, z), then the adversary can also compute $\log_g(a)$ for a random a in $\mathbf{Z}_p^*$ of order q, which violates the DLP hardness assumption.

Now, we consider the soundness. We want to argue that if a malicious prover P^* does not know w, then it should not give the correct pair (a, z) such that $g^z = ah^{\frac{1}{2}a}$ for even a, or $g^z = ah^{\frac{1}{2}(a-1)}$ for odd a. Suppose P^* does not know w but can successfully produce (a, z), then it must be the case that a is a hard instance of DLP and P^* does not know $\log_g(a)$ (since otherwise P^* can compute w from $\log_g a$), which seems infeasible. In particular, based on the following specifically tailored but seemingly hard (and reasonable) assumption the soundness of the above non-interactive protocol holds. (*Note that the WH property does not rely on the new assumption.*)

Hardness assumption: Given (p, q, g, h) of the above form, no PPT algorithm A can with non-negligible probability produce a pair (a, z) such that $g^z = ah^{\frac{1}{2}a}$ for even a or $g^z = ah^{\frac{1}{2}(a-1)}$ for odd a, where $a \in \mathbf{Z}_p^*$ of order q and $z \in \mathbf{Z}_q$. (Note that this assumption implies that a is a hard instance of $\mathbf{Z}_p^*$ and the producer A itself also does not know $\log_g a$.)

Summary: For the security of cryptographic schemes proved with ROs, what may be lost in real world when ROs are replaced by real practical hash functions? The above motivating examples and discussions show that it depends on both, the uses of ROs in security proofs and the uses of ROs in protocol design.

3.2 Principles for restricting uses of ROs in security proofs and protocol designs

In light of the above motivating examples and discussions, we introduce restrictions on uses of RO in *interactive* protocols. We describe the principles in the two-party case, but extensions to the multi-party case are immediate. For a two-party interactive scheme (with restricted ROs), there are two random oracles: $\mathcal{O}_{\mathcal{P}}$ for the prover and $\mathcal{O}_{\mathcal{V}}$ for the verifier. The uses of the ROs in security proofs and protocol design are limited in the following way:

1. For proving prover's security properties (i.e., zero-knowledge), the random oracle $\mathcal{O}_{\mathcal{P}}$ used by the simulator (which plays the role of the honest prover) is *non-programmable* (and the adversary can access $\mathcal{O}_{\mathcal{P}}$ for verifying messages). The random oracle $\mathcal{O}_{\mathcal{V}^*}$ used by the adversary V^* (malicious verifiers) is *programmable* (and the simulator can access $\mathcal{O}_{\mathcal{V}^*}$ for verifying messages). Similarly, for proving verifier's security properties (i.e., soundness), the simulator (playing the role of the honest verifier) uses the *non-programmable* random oracle $\mathcal{O}_{\mathcal{V}}$. The adversary P^* (malicious provers) uses a *programmable* RO $\mathcal{O}_{\mathcal{P}^*}$. Note that this essentially requires that in security proofs the simulator can only define the outputs of the *programmable* random oracle on queries made *by the adversary* in question.
2. Furthermore, the non-programmable random oracles, $\mathcal{O}_{\mathcal{P}}$ and $\mathcal{O}_{\mathcal{V}}$, can be replaced by *any* real (i.e., hash) function without compromising the security of the *honest* players P and V respectively. This requirement essentially says that the restricted random oracle model could be viewed as a "hybrid" between the normal random oracle model and the standard model with real (hash) functions, in the sense that the malicious player still lives in the idealized random oracle world but the honest player could live in real hash function world. In other words, if you are *honest* (e.g. *a trusted authority*) then you could use any real hash function in generating messages from you without compromising your security. Note that, in practice many interactive schemes in the random oracle model (e.g. the DAA protocol of [3]) involve (possibly quite complicated) interactive setup/join protocols between users and *a trusted authority*.
3. As in the original Fiat-Shamir methodology, the random oracles are used only to collapse Σ-protocols into non-interactive ones. This requirement reduces the dependency of the security of the protocol upon the idealized random function property of the realizations of the ROs (e.g., it is not used as a long term commitment, which is naturally desirable in certain critical settings as discussed above).

In the rest of this work, we refer to protocols which are developed with limiting the uses of ROs according to the above principles as *protocols with restricted ROs*.

Remark: In the above description, we only give the general principles of limiting the uses of ROs. When it comes to formally and exactly defining certain cryptographic primitives (e.g., ZK) with restricted ROs, we need to formally

specify (and embed) the above general principles in the specific cryptographic primitives. In particular, the next section provides the formal definition of ZK with restricted ROs.

Comparisons with the work of [23]. We have noted recently the related work of Pass [23] who nicely treats the issue of deniability. [23] observed that non-interactive zero-knowledge in the random oracle model [2] does not preserve deniability and presented a new definition of ZK, named deniable ZK, in the RO model, and constructed a 2-round deniable zero-knowledge protocol from any Σ-protocol.

The approach taken by Pass in [23] for defining and constructing deniable ZK with ROs amounts to the following non-programmable random oracle methodology: all players (including the ZK simulator) access a unique *non-programmable* random oracle. The non-programmable RO methodology is also investigated by Nielsen in the non-committing encryption setting [22]. Below, we provide detailed comparisons between our approach and Pass's non-programmable random oracle methodology. We believe we have some noticeable advantages, though one should welcome various methodologies in this subtle area.

- In Pass's approach, all players access a unique non-programmable RO. But in our approach, there are a pair of ROs: one is non-programmable RO through which messages from the honest player and the simulator (who plays the role of the honest prover) are generated; and one is a programmable RO through which the messages from the adversary in question are generated. Furthermore, the non-programmable RO could be replaced by any real hash function without compromising the honest player's security, a property we do not know how to achieve with Pass's approach.
- We attempted to develop efficient protocols for important cryptographic languages (e.g., DLP and RSA); the efficient protocols are a central reason for which we employ the random oracle idealization, to start with. Solutions for ZK protocols with Pass's approach seems intrinsically inefficient, due to the cut-and-choose technique used, which leads to blow-ups in both computational complexity and communication complexity. Specifically, if the 2-round ZK protocol of [23] is from Schnorr's Σ-protocol for DLP, then on a security parameter n the prover needs to perform $8n$ modular exponentiations and the verifier needs to perform $10n$ exponentiations. For communication complexity, there are about $12n^2$ bits exchanged in total. This inefficiency may violate the spirit of RO protocols as was noted by Pass, who, in fact, suggested as an urgent open problem to find more efficient constructions of zero-knowledge with ROs with the approach. In comparison, our schemes are generic yet practical solutions with restricted ROs and go through only 9 modular exponentiations at each player's side in the BPK model (in the plain model the verifier needs 11 exponentiations), with the DLP as an example.
- In our approach, we further restrict the uses of ROs in protocol designs by limiting the uses of ROs only to collapsing Σ-protocols, while Pass's approach does not. In fact, the ZK protocol with ROs developed in [23] critically uses the commitment scheme $c = RO(m, r)$. This by itself seems fine,

but there is the concern (since we deal with realized idealized objects) that a realization can be broken implying weakened commitment which reveals knowledge. We attempt a design where security properties (WI) for honest parties remain intact even if the realization turns out to be weak.

4 *Generic yet Practical* Round-Optimal Zero-Knowledge with Restricted ROs

Here, we provide the definition of ZK with restricted ROs, show its round-complexity lower-bound, and then present a generic yet practical round-optimal zero-knowledge argument with restricted ROs for any $\mathcal{NP}$-language that admits Σ-protocols (which includes many important languages most relevant to cryptography).

Definition 2 (zero-knowledge argument with restricted ROs). *A pair of interactive machines, $\langle P, V\rangle$, is called a zero-knowledge argument with restricted ROs for a language $L \in \mathcal{NP}$ (with $\mathcal{NP}$-relation R_L), if both machines are polynomial-time and the following conditions hold:*

Completeness: *For any $x \in L$ and its $\mathcal{NP}$-witness w, and any auxiliary input $aux_V \in \{0,1\}^*$, it holds that*

$$\Pr[\langle P^{(\mathcal{O}_P,\mathcal{O}_V)}(w), V^{(\mathcal{O}_V,\mathcal{O}_P)}(aux_V)\rangle(x) = 1] = 1$$

where $\mathcal{O}_P$ and $\mathcal{O}_V$ are two random variables uniformly distributed in $\{0,1\}^{poly(|x|)} \to \{0,1\}^{poly(|x|)}$. The random oracles $\mathcal{O}_P$ and $\mathcal{O}_V$ are used in the following way: random oracles are used only to collapse Σ-protocols (namely, deriving the challenges in such protocols and running them non-interactively), where each player only access its designated random oracle for generating messages from it (and the second random oracle is only accessed for verifying messages from its counterpart).

Computational soundness: *For any $x \notin L$, any PPT interactive machine P^*, and any auxiliary input $aux_{P^*} \in \{0,1\}^*$ and $aux_V \in \{0,1\}^*$, it holds that:*

$$\Pr[\langle P^{*(\mathcal{O}_{P^*},\mathcal{O}_V)}(aux_{P^*}), V^{(\mathcal{O}_V,\mathcal{O}_{P^*})}(aux_V)\rangle(x) = 1] \leq \varepsilon(|x|)$$

where $\varepsilon(\cdot)$ is a negligible function, and $\mathcal{O}_{P^*}$ and $\mathcal{O}_V$ are random variables uniformly distributed in $\{0,1\}^{poly(|x|)} \to \{0,1\}^{poly(|x|)}$. Furthermore, the soundness condition holds even if $\mathcal{O}_V$ is arbitrarily (rather than uniformly) distributed from $\{0,1\}^{poly(|x|)} \to \{0,1\}^{poly(|x|)}$ (i.e., $\mathcal{O}_V$ can be any function).

(Black-box) zero-knowledge: *There exists an expected polynomial-time simulator S such that for every PPT verifier V^*, any $aux_{V^*} \in \{0,1\}^*$, any sufficiently long $x \in L$, the following two ensembles are computationally indistinguishable (where the distinguishing gap is a function in $|x|$):*

– $\{(\mathcal{O}_{V^*}, \mathcal{O}_P, view^{P^{(\mathcal{O}_P,\mathcal{O}_{V^*})}(w)}_{V^{*(\mathcal{O}_{V^*},\mathcal{O}_P)}(aux_{V^*})}(x))\}_{x\in L, aux_{V^*}\in\{0,1\}^*}$ *for arbitrary* w *such that* $(x,w)\in R_L$.
– $\{(\mathcal{O}_P, S^{\mathcal{O}_P}(x, aux_{V^*}))\}_{x\in L, aux_{V^*}\in\{0,1\}^*}$

where $\mathcal{O}_P$ *and* $\mathcal{O}_{V^*}$ *are random variables uniformly distributed in* $\{0,1\}^{poly(|x|)} \rightarrow \{0,1\}^{poly(|x|)}$, *and* $view^{P^{(\mathcal{O}_P,\mathcal{O}_{V^*})}(w)}_{V^{*(\mathcal{O}_{V^*},\mathcal{O}_P)}(aux_{V^*})}(x)$ *is a random variable describing* V^**'s state and all messages exchanged during a joint computation between* P *and* V^* *on common input* x *when* P *has* w *as its auxiliary input and accesses* $(\mathcal{O}_P, \mathcal{O}_{V^*})$, *and* V *has* aux_{V^*} *as its auxiliary input and accesses* $(\mathcal{O}_{V^*}, \mathcal{O}_P)$. Furthermore, the zero-knowledge condition holds even when $\mathcal{O}_P$ is taken arbitrarily (rather than uniformly) from $\{0,1\}^{poly(|x|)} \rightarrow \{0,1\}^{poly(|x|)}$ (i.e., $\mathcal{O}_P$ can be any function).

Comment: Note that in the definition of (black-box) ZK with restricted ROs, the RO $\mathcal{O}_P$ is given (pre-specified) in the above two probability ensembles, which means that S cannot "program" $\mathcal{O}_P$. But, for the RO $\mathcal{O}_{V^*}$, S is allowed to "program" and output a "simulation" of $\mathcal{O}_{V^*}$ in its simulation. Note that the random oracle is actually an infinite object, S thus, for this purpose, has a special fill out function. We refer the reader to [2] for a more formal treatment of "programming" ROs.

We next show, by the Goldreich-Krawczyk technique [17], the impossibility of non-interactive black-box zero-knowledge arguments with restricted ROs for non-trivial languages. Specifically, we give the following proposition (whose proof is omitted here due to space limitation):

Proposition 1. *Suppose an* $\mathcal{NP}$*-language* L *admits a one-round black-box zero-knowledge argument with restricted ROs, then* $L \in \mathcal{BPP}$.

Finally, we show that a protocol based on the Feige-Shamir 4-round ZK argument for $\mathcal{NP}$ renders a generic yet practical 2-round (i.e., optimal) ZK argument with restricted ROs for any language that admits Σ-protocols. The protocol is depicted in Figure-2.

Comment: At a high level, the protocol depicted in Figure-2 can be viewed as a Σ_{OR}-based implementation of the Feige-Shamir 4-round ZK arguments for $\mathcal{NP}$ (the version appearing in [13]) in the RO model. The construction of [13] is a plausible $\mathcal{NP}$-solution and goes through general (inefficient) $\mathcal{NP}$-reductions, whereas here (for this section) we emphasize the fact that the protocol works directly for any language that admits Σ-protocols (a large set that includes, in particular, both DLP and RSA). If the underlying Σ-protocols for the language are practical, then the transformed protocols are also practical. With Σ-protocol for DLP as an example, our scheme goes through 9 modular exponentiations at the prover side and 11 exponentiations at the verifier side.

Theorem 2. *Let* f_V *be* any *one-way function that admits* Σ*-protocol and* L *be a language that admits* Σ*-protocols, the protocol depicted in Figure-2 is a 2-round ZK argument with restricted ROs for* L.

Common input. An element $x \in L$ of length n, where L is an $\mathcal{NP}$-language that admits Σ-protocols.

P's private input. A witness w for $x \in L$.

Random oracles. There are two random oracles $\mathcal{O}_P$ and $\mathcal{O}_V$: $\mathcal{O}_P$ is used by the prover for generating non-interactive Σ-proofs and $\mathcal{O}_V$ is used by the verifier for generating non-interactive Σ-proofs.

Round-1. The verifier V selects a OWF f_V that admits Σ-protocols, randomly selects two elements in the domain of f_V, x_V^0 and x_V^1, computes $y_V^0 = f_V(x_V^0)$ and $y_V^1 = f_V(x_V^1)$, randomly selects a bit b from $\{0,1\}$, sends to the prover a non-interactive Σ_{OR}-proof on (y_V^0, y_V^1), denoted $\pi_V = (y_V^0, y_V^1, a_V, e_V, z_V, aux_V)$, that it knows either the preimage of y_V^0 or the preimage of y_V^1. The witness used by V in forming π_V is x_V^b. The random challenge e_V is generated by querying $\mathcal{O}_V$ with $(x, y_V^0, y_V^1, a_V, aux_V)$, where aux_V is the auxiliary information of V that possibly includes a time-stamp.

Round-2. The prover P first checks the validity of π_V and aborts if it is not valid. Otherwise, P sends back a non-interactive Σ_{OR}-proof on (x, y_V^0, y_V^1), denoted $\pi_P = (a_P, e_P, z_P)$, that it knows either the witness for $x \in L$ or the preimage of either y_V^0 or y_V^1. The witness used by P is its private input (i.e., the $\mathcal{NP}$-witness w). The random challenge e_P is generated by querying $\mathcal{O}_\mathcal{P}$ with (x, a_P, π_V).

Verifier's Decision. The verifier checks the validity of π_P and accepts if it is valid, otherwise it rejects.

Figure-2. The generic yet practical 2-round ZK arguments with restricted ROs

Proof (sketch). Intuitively, P proves $x \in L$ only after it is convinced that the verifier does know the preimage of either y_V^0 or y_V^1 (this means that the verifier can also generate the second-round message *by itself*), so the ZK property of the protocol should hold. In more details, for a malicious verifier V^* who accesses the programmable random oracle $\mathcal{O}_{\mathcal{V}^*}$, the zero-knowledge simulator S first extracts $x_{V^*}^b$ (the witness used by V^* in generating the Round-1 message) by redefining the random oracle $\mathcal{O}_{\mathcal{V}^*}$ on queries made by V^*, then using $x_{V^*}^b$ as the witness S generates a simulated Round-2 message *through its fixed random oracle* $\mathcal{O}_\mathcal{P}$. By the perfect WI property of Round-2, the simulated transcript is indistinguishable from the real transcript. Furthermore, since in security proof we only need the WI property of the non-interactive Σ_{OR}-protocol of Round-2, the fixed random oracle $\mathcal{O}_\mathcal{P}$ can be replaced by *any real function*, as the WI property of Round-2 does hold with respect to any function (*see Claim 1*). That is, the honest prover's security (i.e., ZK) holds even when the honest prover uses any real function (say cryptographic hash based one) in generating non-interactive Σ_{OR}-protocols (of Round-2).

For proving soundness, however, one may argue that seeing the Σ_{OR}-proof π_V sent by the honest verifier in Round-1 (*which could be generated through any real function*) may help a malicious prover P^* to give a false Σ_{OR}-proof π_{P^*} in Round-2. What save us here are the key-pair technique (originally introduced in the Public-Key Encryption setting by Naor and Yung [21]) and the witness indistinguishability of the Σ_{OR}-proof even with real functions. In more details,

suppose a malicious prover P^* can successfully convince of a false statement $x \notin L$ with a non-negligible probability, then we show a PPT algorithm E that will break the one-wayness of f_V. Specifically, on an input y_V E runs P^* as a subroutine and works as follows: E randomly selects x'_V from the domain of f_V, computes $y'_V = f_V(x'_V)$. Then, E randomly selects a bit b from $\{0,1\}$, sets y_V^b be y'_V and y_V^{1-b} be the input y_V. Finally, by using x'_V as the witness E sends to P^* the Round-1 message (generated through the fixed random oracle $\mathcal{O}_V$), claiming that it knows the preimage of either y_V^0 or y_V^1. After receiving a successful Round-2 message from P^* that is generated through the programmable RO $\mathcal{O}_{P^*}$, by rewinding P^* and redefining $\mathcal{O}_{P^*}$ E will extract a preimage of either y_V^0 or y_V^1 (as we assume $x \notin L$). Then by the perfect WI property of Round-1, with probability $1/2$ (conditioned on P^* successfully giving the Round-2 message), the extracted value will be the preimage of $y_V^{1-b} = y_V$, which violates the one-wayness of f_V. Again, in the above security analysis, we only need the WI property of the non-interactive Σ_{OR} of Round-1, and so the fixed random oracle $\mathcal{O}_V$ can be replaced by *any real function*. That is, the honest verifier's security (i.e., soundness) holds even when the honest verifier uses any function in generating the non-interactive Σ_{OR}-protocols of Round-1.

5 Concurrent Security of the Feige-Shamir ZK with Registered Public-Keys and with Restricted ROs

Dealing with concurrent adversaries in the interactive random oracle model turns out to be much more complicated and subtle. The reason is that for messages sent by an adversary we need to rewind the adversary and redefine the outputs of the programmable random oracles to extract the witnesses used by the adversary. But for a concurrent adversary, it can make both concurrent interleaving interactions with the honest player instances and concurrent interleaving oracle queries (across multiple existing sessions). We thus risk an exponential blow-up when tracking back through the interleaving interactions or the interleaving oracle queries across multiple sessions, in the sense that previous simulation efforts (interaction rewinding-s or random oracle redefining-s) will become void. This phenomenon is first observed by Dwork, Naor and Sahai [12] for *interactive* protocols in the standard model in dealing with adversaries that make concurrent interleaving interactions with honest player instances, and observed also by Shoup and Gennaro [25] for *non-interactive* schemes in the random oracle model in dealing with adversaries that make concurrent interleaving oracle queries across multiple sessions in the context of threshold decryption.

To avoid the exponential blow-up in dealing with concurrent adversaries, several computational models have been proposed: the timing model [12, 16], the preprocessing model [10], the common reference string model [9], and the bare public-key model [4].

The bare public-key (BPK) model was introduced by Canetti, Goldreich, Goldwasser and Micali [4] to achieve round-efficient resettable zero-knowledge (rZK) that is a generalization and strengthening of the notion of concurrent

zero-knowledge [12]. A protocol in the BPK model simply assumes that all verifiers have deposited a public key in a public file before any interaction takes place among the users[3]. to all users at all times. Note that an adversary may deposit many (possibly invalid or fake) public keys in it, particularly, without even knowing corresponding secret keys or whether such exist. That is, no trusted third party is assumed in the BPK model. What is essentially guaranteed by the BPK model is only a limitation on the number of different identities that a potential adversary may assume and there are no other assurances. The adversary, in turn, may try to impersonate any user registered in the public-file, but it cannot act on behalf of a non-registered user. The BPK model is thus very simple, and it is, in fact, a weaker version of the frequently used public-key infrastructure (PKI) model (recall that PKI underlies any public-key cryptosystem or any digital signature scheme). Despite its apparent simplicity, the BPK model turns out to be quite powerful in dealing with concurrent adversaries and stronger resetting adversaries.

Soundness in public-key models is more subtle than in the standard model [20]. In public-key models, a verifier V has a secret key SK, corresponding to its public-key PK. A malicious prover P^* could potentially gain some knowledge about SK from an interaction with the verifier. This gained knowledge may help him to convince the verifier of a false theorem in another interaction. Micali and Reyzin [20] showed that under standard intractability assumptions there are four distinct meaningful notions of soundness, i.e., from weaker to stronger, one-time, sequential, concurrent and resettable soundness. In this paper we focus on concurrent soundness which roughly means, for zero-knowledge protocols, that a malicious prover P^* cannot convince the honest verifier V of a false statement even when P^* is allowed multiple interleaving interactions with V.

Due to space limitation, the definitions of concurrent ZK and concurrent soundness in the BPK model with restricted ROs are omitted here, and will be presented in the full version of this work.

5.1 The Feige-Shamir ZK protocol is *not* secure in the public-key setting

Next we show that the Feige-Shamir ZK protocol [13] is, in general, *not* concurrently secure in the public setting (indeed it was not designed for that more modern setting). Specifically, we show a concurrent attack against the Σ_{OR}-based implementation of the Feige-Shamir ZK protocol in the public-key setting (which may be of independent interest).

Consider the version of the protocol depicted in Figure-2 when the pair (y_V^0, y_V^1) is published as the verifier's public-key (i.e., fixed once and for all sessions) and ROs are removed (i.e., random challenges e_V and e_P are not obtained any longer by querying the ROs, but sent by the prover and the verifier respectively). We remark that, at a first glance, *it is quite natural* for the verifier

[3] The BPK model does allow dynamic key registrations (see [4]).

to publish (y_V^0, y_V^1) as its public-key when the Feige-Shamir ZK protocol (especially its Σ_{OR}-based implementation) is used in practice. But, the following attack shows that this intuition is wrong.

Let L (wlog, the $\mathcal{NP}$-complete language Directed Hamiltonian Cycle DHC) be a language that admits Σ-protocols. We show how a malicious prover P^* can convince an honest verifier V (with public-key (y_V^0, y_V^1)) of a false statement "$x \in L$" while $x \notin L$, by concurrently interacting two sessions with V. The message schedule of P^* in the two sessions is specified as follows.

1. P^* interacts with V in the first session and works just as the honest prover does in Phase-1. When P^* moves into Phase-2 of the first session and needs to send V the first-round message, denoted by a_P, of the Σ_{OR}-protocol of Phase-2 of this session on common input (x, y_V^0, y_V^1), P^* suspends the first session and does the following:
   - It first runs the SHVZK simulator (of the underlying Σ-protocol for L) on x to get a simulated conversation, denoted by (a_x, e_x, z_x), for the false statement "$x \in L$".
   - Then, P^* initiates a second session with V; After receiving the first-round message, denoted by a'_V, of the Σ_{OR}-protocol of Phase-1 of the second session on common input (y_V^0, y_V^1) (i.e., V's public-key) , P^* sets $a_P = (a_x, a'_V)$ and suspends the second session.
2. Now, P^* continues the execution of the first session, and sends $a_P = (a_x, a'_V)$ to V as the first-round message of the Σ_{OR}-protocol of Phase-2 of the first session.
3. P^* Runs V further in the first session. After receiving the second-round message of Phase-2 of the first session, denoted by e_P (i.e., the random challenge from V), P^* sets $e'_V = e_P \oplus e_x$ and suspends the first session again.
4. P^* continues the execution of the second session, and sends $e'_V = e_P \oplus e_x$ to V as its random challenge in the second-round of the Σ_{OR}-protocol of Phase-1 of the second session. After receiving the third-round message of Phase-1 of the second session, denoted by z'_V, P^* sets $z_P = ((e_x, z_x), (e'_V, z'_V))$ and suspends the second session again.
5. P^* continues the execution of the first session again, sending the value $z_P = ((e_x, z_x), (e'_V, z'_V))$ to V as the last-round message of the first session.

Note that (a_x, e_x, z_x) is an accepting conversation for showing "$x \in L$", (a'_V, e'_V, z'_V) is an accepting conversation for showing the knowledge of the preimage of either y_V^0 or y_V^1, and furthermore $e_P = e_x \oplus e'_V$. According to the description of Σ_{OR} (presented in Section 2), this means that, from the viewpoint of V, (a_P, e_P, z_P) is an accepting conversation on common input (x, y_V^0, y_V^1) of the Σ_{OR}-protocol of Phase-2 of the first-session, and thus P^* successfully convinced V of a false statement in the first session. We remark that, in general, the above attack also enables P^* to convince V of a true statement $x \in L$ without knowing any $\mathcal{NP}$-witness for $x \in L$.

5.2 The Feige-Shamir ZK is concurrently secure in the BPK model with restricted ROs

As shown, the Feige-Shamir ZK is not concurrently secure in public-key model, but we next show that it is still concurrently secure in the BPK model with restricted ROs (which may make it useful in certain applications which it was not originally designed for).

Specifically, consider the following modified version of the protocol depicted in Figure-2 in the BPK model: there is a key generation phase before any interaction takes place among the users, in which each verifier V_i registers $(y^0_{V_i}, y^1_{V_i})$ in a public-key file F, where $y^0_{V_i} = f_{V_i}(x^0_{V_i})$, $y^1_{V_i} = f_{V_i}(x^1_{V_i})$ and f_{V_i} is a OWF that admits Σ-protocols. For a bit b randomly chosen from $\{0,1\}$, V_i keeps $x^b_{V_i}$ in secret as its secret-key while discarding $x^{(1-b)}_{V_i}$. Then in Round-1 of the modified protocol, by using $x^b_{V_i}$ as the witness, V_i sends a non-interactive Σ_{OR}-proof that it knows the preimage of either $y^0_{V_i}$ or $y^1_{V_i}$. Round-2 remains unchanged.

Theorem 3. *Under any one-way functions that admit Σ-protocols, the above modified protocol is a generic yet practical 2-round concurrently sound concurrent ZK argument with restricted ROs in the BPK model for any language that admits Σ-protocols.*

Below, we present the high-level proof overview of Theorem 3 and identify some complications and subtleties of dealing with concurrent adversaries for *interactive* schemes with ROs.

The simulation procedure for concurrent zero-knowledge is similar to the simulation procedure for resettable zero-knowledge presented in [4]. Specifically, for any concurrent adversary V^* that has as its output a public-key file of the form $F = \{(y^0_{V^*_1}, y^1_{V^*_1}), (y^0_{V^*_2}, y^1_{V^*_2}), \cdots, (y^0_{V^*_q}, y^1_{V^*_q})\}$, the zero-knowledge simulator S runs V^* as a subroutine and works in at most $q+1$ phases. In each phase, S either successfully gets a simulated transcript or "breaks" a new public-key $(y^0_{V^*_i}, y^1_{V^*_i})$, $1 \leq i \leq q$, in the sense that S can extract the corresponding secret-key $x^b_{V^*_i}$. In this process, we identify that dealing with concurrent adversaries for interactive schemes in the random oracle model actually amounts to dealing with resetting adversaries in the standard model. Specifically, in dealing with these resetting adversaries for proving resettable zero-knowledge in the standard model for the sake of extracting the witness used by a malicious resetting verifier in one session (for facilitating the successful simulation), we normally need to rewind the adversary and change the random challenge that has been sent with respect to some message of that session (e.g. the first message of a Σ-protocol), and give back, in turn, a different random challenge. But, the random challenge to be changed in that session may have been "defined" in a previous session, we may thus need to rewind the adversary in a previous session in which the random challenge is defined *for the first time*. Similarly, in dealing with concurrent adversaries for proving concurrent zero-knowledge in the random oracle model, to extract the witness used by the adversary in forming the Round-1 message in

one session, we need to redefine the random output of the programmable random oracle. But the random output of the programmable random oracle used in that session may be obtained by the adversary by querying the random oracle in a previous session, and thus we need to rewind the adversary in the previous session where it made the oracle query in question *for the first time*. We remark that in the proof of concurrent ZK we only need the WI property of the non-interactive Σ_{OR}-proofs (of Round-2) generated through the non-programmable RO $\mathcal{O}_{\mathcal{P}}$ (that does hold even when $\mathcal{O}_{\mathcal{P}}$ is replaced by any function of a proper size). This means that the honest prover's security (i.e., concurrent ZK) holds even when the honest prover uses any function in generating the non-interactive Σ_{OR}-protocols of Round-2.

For concurrent soundness, assume a PPT q-concurrent adversary P^* can successfully convince V with public-key (y_V^0, y_V^1) of a false statement with non-negligible probability p in one of the q concurrent sessions, then we will construct an algorithm E that on an input y in the range of f_V outputs the preimage of y with non-negligible probability $\frac{p^2}{2q}$ in expected polynomial-time, which violates the one-wayness of f_V.

Algorithm E on an input y, first randomly selects an element x' in the domain of f_V, computes $y' = f_V(x')$, randomly selects a bit b from $\{0,1\}$, sets y_b be y' and y_{1-b} be y, publishes (y_0, y_1) as its public-key while keeping x' privately as the corresponding secret-key. Then, E randomly chooses i from $\{1, 2, \cdots, q\}$, and runs P^* by playing the role of the honest verifier (with (y_0, y_1) as its public-key and x' as its secret-key) in any session other than the i-th session. In the i-th session on a common input x_i, suppose P^* successfully gives a Round-2 message, denoted by $(a_{P^*}^{(i)}, e_{P^*}^{(i)}, z_{P^*}^{(i)})$, with respect to a Round-1 message, denoted by $\pi_V^{(i)}$, sent by E in the first-round of the i-th session, where $e_{P^*}^{(i)}$ is the random oracle answer given by E to P^* on a query of the form $(x_i, a_{P^*}, \pi_V^{(i)})$ to the programmable random oracle $\mathcal{O}_{\mathcal{P}^*}$. Then E rewinds P^* to the point that P^* just made the oracle query $(x_i, a_{P^*}, \pi_V^{(i)})$, gives back a new random oracle answer $e_{P^*}^{(i)\prime}$ and runs P^* from the above rewinding point and on. We stress that in the above process all Round-1 messages from E to P^* are generated through the fixed random oracle $\mathcal{O}_{\mathcal{V}}$.

Since we assume that P^* can, with probability p, convince V of a false statement in one of the q concurrent sessions, then conditioned on E correctly guessing the value i, it is easy to see that with probability p^2 E will extract an $\mathcal{NP}$-witness for $x_i \in L$ or a preimage of either y_0 or y_1, which is guaranteed by the special soundness of the Σ_{OR} protocol. Since we further assume that $x_i \notin L$ and E randomly guesses i from $\{1, \cdots, q\}$, we conclude that with probability $\frac{p^2}{q}$ E will output the preimage of either y_0 or y_1. Furthermore, according to the perfect WI property of Σ_{OR}-protocol, we know that with probability $\frac{p^2}{2q}$ E will output a preimage of $y = y_{1-b}$, which violates the one-wayness of f_V. Note that in the above proof we only need the WI property of the non-interactive Σ-proofs (of Round-1) generated through the non-programmable RO $\mathcal{O}_{\mathcal{V}}$ (that does hold even when $\mathcal{O}_{\mathcal{V}}$ is replaced by any properly sized function), which means that the

honest verifier's security (i.e., concurrent soundness) holds even for this type of $\mathcal{O}_{\mathcal{V}}$.

Comment: At a first glance, it seems that the above proof procedure for concurrent soundness can also be applicable to the Feige-Shamir ZK protocol in the public-key model *without ROs* (that is however, as we have shown, not concurrently secure). This subtle point needs further elaboration: The WI property is only guaranteed to be concurrently composable when the *same* protocol is composed concurrently. But in our case, the concurrent adversary P^* actually also runs WI protocols to V (or E) with the player role reversed with respect to the WI protocols from V (or E) to P^* . In general, in this case the concurrent WI property of the Σ_{OR}-protocols from V (or E) to P^* is not guaranteed. This is also the very reason why the Feige-Shamir ZK protocol is not concurrently secure in the public-key model *without ROs*, as shown by our concurrent attack. In contrast, in the (Σ_{OR}-based implementation of) Feige-Shamir ZK protocol in the BPK model *with restricted ROs*, the important fact is that we are, both, working in the random oracle model and the WI (i.e., Σ_{OR}) protocols are *non-interactive*. In more details, suppose in this case the preimage extracted by E is dependent on the witness used by E, then we can show a PPT algorithm E' that violates the WI property of *non-interactive* Σ_{OR}-protocols as follows. For a PPT concurrent adversary P^* and the honest verifier V with public-key (y_0, y_1) who actually is a non-interactive Σ_{OR}-prover on (y_0, y_1) with random challenges generated through the fixed random oracle $\mathcal{O}_V$, E' runs P^* as a subroutine and interacts with V. E' works just as E does but with the following modifications: Whenever E' needs to send a non-interactive Σ_{OR}-proof in Round-1 of a session, E' just interacts with V to get such a proof and sends it to P^*. Note that E' never redefines the fixed random oracle $\mathcal{O}_V$. Clearly, E' can violate the WI property of the *non-interactive* Σ_{OR}-proofs received from V if the extracted preimage (from P^*) is dependent on the witness used by V.

6 A Note on the Applications of the 2-Round ZK with Restricted ROs

The notion of zero-knowledge plays a central role in modern cryptography and we are now at the point where more and more complicated *interactive* schemes with random oracle methodologies are under development (including ones for industrial use). Thus, we expect that the generic yet practical 2-round ZK protocols with restricted ROs (with or without registered public-keys) can be used as a building block in constructing more complicated *interactive* schemes provably secure with restricted ROs.

In particular, we note that the 2-round ZK protocols with restricted ROs can be used to transform a large number of (but not necessarily all) interactive schemes (and non-interactive systems with interactive setup/join protocols like PKI, group signatures or e-cash) developed originally in the normal random oracle model, which use the random oracle only to collapse Σ-protocols, into schemes with provable security using restricted ROs, paying in efficiency *at most* one ex-

tra round, but with seemingly more sound provable security guarantees. The idea is to replace each non-interactive NIZK in the original interactive scheme (developed in the normal interactive RO model) by our 2-round ZK protocols with restricted ROs. The key observation here is that all 2-round ZK protocols with one party playing the role of the prover can share the same Round-1 non-interactive Σ_{OR}-protocol sent by its counterpart. This way, the non-interactive nature of the NIZK-protocols in the original interactive systems can be preserved at the price of at most one additional initiating round on top of the protocol. This general transformation, along with detailed discussions, will be presented in the full version of this work.

Acknowledgments. We are grateful to Yehuda Lindell for referring us to [13] and for valuable discussions and suggestions. We thank the anonymous referees of TCC'06 for valuable and detailed comments and suggestions.

References

1. M. Bellare, A. Boldyreva and A. Palacio. An Uninstantiable Random-Oracle-Model Scheme for a Hybrid-Encryption Problem In *C. Cachin and J. Camenisch (Ed.): Advances in Cryptology-Proceedings of EUROCRYPT 2004, LNCS 3027*, pages 171-188. Springer-Verlag, 2004.
2. M. Bellare and P. Rogaway. Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In*ACM Conference on Computer and Communications Security*, pages 62-73, 1993.
3. E. Brickell, J. Camenisch and L. Chen. Direct Anonymous Attestation. ACM's CCS 2004.
4. R. Canetti, O. Goldreich, S. Goldwasser and S. Micali. Resettable Zero-Knowledge. In *ACM Symposium on Theory of Computing*, pages 235-244, 2000.
5. R. Canetti, O. Goldreich and S. Halevi. The Random Oracle Methodology, Revisited. In *ACM Symposium on Theory of Computing*, pages 209-218, 1998.
6. R. Canetti, O. Goldreich and S. Halevi. On the Random-Oracle Methodology as Applied to Length-Restricted Signature Schemes. In *1st Theory of Cryptography Conference (TCC), LNCS 2951* , pages 40-57, Springer-Verlag, 2004.
7. R. Cramer. Modular Design of Secure, yet Practical Cryptographic Protocols, PhD Thesis, University of Amsterdam, 1996.
8. R. Cramer, I. Damgard and B. Schoenmakers. Proofs of Partial Knowledge and Simplified Design of Witness Hiding Protocols. In *Y. Desmedt (Ed.): Advances in Cryptology-Proceedings of CRYPTO 1994, LNCS 839*, pages 174-187. Springer-Verlag, 1994.
9. I. Damgard. Efficient Concurrent Zero-Knowledge in the Auxiliary String Model. In *B. Preneel (Ed.): Advances in Cryptology-Proceedings of EUROCRYPT 2000, LNCS 1807*, pages 418-430. Springer-Verlag, 2000.
10. G. Di Crescenzo and R. Ostrovsky. On Concurrent Zero-Knowledge with Pre-Processing. In *M. J. Wiener (Ed.): Advances in Cryptology-Proceedings of CRYPTO 1999, LNCS 1666*, pages 485-502. Springer-Verlag, 1999.
11. D. Dolev, C. Dwork and M. Naor. Non-Malleable Cryptography. *SIAM Journal on Computing*, 30(2): 391-437, 2000. Preliminary version appears in STOC'91.

12. C. Dwork, M. Naor and A. Sahai. Concurrent Zero-Knowledge. In *ACM Symposium on Theory of Computing*, pages 409-418, 1998. Full version to appear in Journal of the ACM.
13. U. Feige. Alternative Models for Zero-Knowledge Interactive Proofs. Ph.D. Thesis, Department of Computer Science and Applied Mathematics, Weizmann Institute of Science, Rehovot, Israel, 1990.
14. A. Fiat and A. Shamir. How to Prove Yourself: Practical Solutions to Identification and Signature Problems. In *A. Odlyzko (Ed.): Advances in Cryptology-Proceedings of CRYPTO'86, LNCS 263*, pages 186-194. Springer-Verlag, 1986.
15. U. Feige and Shamir. Zero-Knowledge Proofs of Knowledge in Two Rounds. In *G. Brassard (Ed.): Advances in Cryptology-Proceedings of CRYPTO 1989, LNCS 435*, pages 526-544. Springer-Verlag, 1989.
16. O. Goldreich. Concurrent Zero-Knowledge with Timing, Revisited. In *ACM Symposium on Theory of Computing*, pages 332-340, 2002.
17. O. Goldreich and H. Krawczyk. On the Composition of Zero-Knowledge Proof Systems. *SIAM Journal on Computing*, 25(1): 169-192, 1996.
18. S. Goldwasser and Y. Tauman. On the (In)security of the Fiat-Shamir Paradigm. In *IEEE Symposium on Foundations of Computer Science*, pages 102-115, 2003.
19. L. Guillou and J. J. Quisquater. A Practical Zero-Knowledge Protocol Fitted to Security Microprocessor Minimizing both Transmission and Memory. In *C. G. Gnther (Ed.): Advances in Cryptology-Proceedings of EUROCRYPT 1988, LNCS 330* , pages 123-128, Springer-Verlag, 1988.
20. S. Micali and L. Reyzin. Soundness in the Public-Key Model. In *J. Kilian (Ed.): Advances in Cryptology-Proceedings of CRYPTO 2001, LNCS 2139*, pages 542–565. Springer-Verlag, 2001.
21. M. Naor and M. Yung. Public-Key Cryptosystems Provably Secure Against Chosen Ciphertext Attacks. In *ACM Symposium on Theory of Computing*, pages 427-437, 1990.
22. Jesper Buus Nielsen. Separating Random Oracle Proofs from Complexity Theoretic Proofs: The Non-Committing Encryption Case. In *M. Yung (Ed.): Advances in Cryptology-Proceedings of CRYPTO 2002, LNCS 2442*, pages 111-126, Springer-Verlag, 2002.
23. R. Pass. On Deniabililty in the Common Reference String and Random Oracle Models. In*D. Boneh (Ed.): Advances in Cryptology-Proceedings of CRYPTO 2003, LNCS 2729*, pages 316-337, Springer-Verlag 2003.
24. C. Schnorr. Efficient Signature Generation by Smart Cards. *Journal of Cryptology*, 4(3): 24, 1991.
25. V. Shoup and R. Gennaro. Securing Threshold Cryptosystems Against Chosen Ciphertext Attack. *Journal of Cryptology*, 15(2): 75-96, 2002.