Non Observability in the Random Oracle Model

Prabhanjan Ananth and Raghav Bhaskar
Microsoft Research India
Bangalore 560001

Abstract

The Random Oracle Model, introduced by Bellare and Rogaway, provides a method to heuristically argue about the security of cryptographic primitives and protocols. The basis of this heuristic is that secure hash functions are close enough to random functions in their behavior, and so, a primitive that is secure using a random function should continue to remain secure even when the random function is replaced by a real hash function. In the security proof, this setting is realized by modeling the hash function as a random oracle. However, this approach in particular also enables any reduction, reducing a hard problem to the existence of an adversary, to *observe* the queries the adversary makes to its random oracle and to *program* the responses that the oracle provides to these queries. While, the issue of programmability of query responses has received a lot of attention in the literature, to the best of our knowledge, observability of the adversary's queries has not been identified as an artificial artefact of the Random Oracle Model. In this work, we study the security of several popular schemes when the security reduction cannot "observe" the adversary's queries to the random oracle, but can (possibly) continue to "program" the query responses. We first show that RSA-PFDH and Schnorr's signatures continue to remain secure when the security reduction is non observing (NO reductions), which is not surprising as their proofs in the random oracle model rely on programmability. We also provide two example schemes, namely, Fischlin's NIZK-PoK [Fis05] and non interactive extractable commitment scheme, extractor algorithms of which seem to rely on observability in the random oracle model. While we prove that Fischlin's online extractors cannot exist when they are non observing, our extractable commitment scheme continues to be secure even when the extractors are non observing. We also introduce Non Observing Non Programming reductions which we believe are closest to standard model reductions.

1 Introduction

The Random Oracle Model (ROM) was introduced by Bellare and Rogaway in [BR93] as an alternative model to study the security of cryptographic primitives and protocols. In contrast to the Standard model, it assumes the availability of a random function (via an oracle) to all parties (eg. adversary, challenger etc.) in any security game devised to study the security of a cryptographic primitive. The oracle implementing the random function (called the random oracle) returns a randomly chosen value (which it remembers for later) from the range, when queried at a new domain point. For already queried points, it returns the same value that it returned the first time around. The introduction of ROM made it possible to prove the security of many different kinds of cryptographic primitives (including digital signatures, encryption schemes etc.) for which there existed no proofs in the Standard model [PS00]. As truly random functions do not exist in practice, when using these primitives in the real world, the role of random oracle is played by a secure hash function. The heuristic is that secure hash functions are close enough to random oracles in their behavior, and so, the primitives continue to remain secure even under this substitution. This

methodology has resulted in many provably secure (in ROM), and at the same time, practical and efficient schemes.

Since no real world hash function is truly random, proofs in ROM have been a subject of debate by cryptographers. Public key encryption and Signature schemes have been devised such that they can be proved secure in the ROM but which become insecure as soon as the random oracle is instantiated by any real hash function [CGH04]. However, no real attacks have been demonstrated against any practical scheme that has been proved secure in ROM. This ambiguity about the *reasonability* of ROM has been of great interest to cryptographers. Furthermore, from a practical viewpoint, random oracle heuristic is the only basis for arguing the security of some of the most efficient cryptographic schemes (e.g. [PS00, BR, BR96]). Therefore, it is of fundamental importance to understand why certain cryptographic schemes can be proved secure in the ROM while no proof of security exists for them in the Standard model.

In ROM, the challenger (or the reduction) simulates the random oracle for the adversary in the security game. This new power to simulate the random oracle for the adversary enables the challenger to solve some hard problem, thus ruling out the possibility of the adversary winning in the security game. The ability to simulate the random oracle seems to "artificially" augment the capabilities of the challenger (in comparison to a Standard model challenger) in the following two ways:

- The challenger can now observe the input points at which the adversary makes queries to the random oracle. We will refer to this ability of the challenger as *observability*.
- The challenger can now control the response of the random oracle at these input points, often embedding instances of some hard problem in the response. We will refer to this ability of the challenger as *programmability*.

Both these additional capabilities of the reduction are very artifical when compared to Standard model reductions. Neither do we know of hash functions that can support such complicated programming nor do we know of a way of observing an algorithm's queries to a hash function when the reduction is black-box. The possibility of programming the random oracle has been exploited in constructing many security reductions and thus the programmability aspect of the ROM has attracted much attention (see [Nie, FLR$^+$10]). We explain some of that work later. On the other hand, even though observability is often criticized (explicitly in [Nie]) for providing the challenger with an unreasonable ability[1], to the best of our knowledge, no formal study of this capability of the reduction has been done. Perhaps one of the most important reasons for the lack of this study is a general perception that observability is crucial to every proof in the ROM and that nothing can be achieved without it, thereby leaving no motivation to study reductions that limit observability. In this work, we study the role of observability in the construction of security reductions for several schemes. We call such reductions, in which the communication between the adversary and the random oracle is hidden from the reduction, as Non Observing reductions.

1.1 Non Observing reductions.

While in ROM, reductions often work by providing a simulation of the random oracle to the adversary, we want Non Observing reductions to operate in the presence of an external random oracle. All entities make their queries to this external random oracle which is independent of the reduction. All communication between the random oracle and the adversary is hidden from the reduction. As our focus is on restricting the observability capability of the reductions, we do let

[1]Standard model reductions would never get to see the queries made by the adversary to the random oracle.

the reduction control the responses returned by the random oracle to the adversary (as long as the returned responses are uniformly distributed in the range of the random function). As shown in Figure 1, the reduction first sends a Turing machine to the external random oracle which uses this machine to respond to the queries as follows. On receiving a query, the external random oracle, inputs "Next" to the Turing machine, sent by the reduction, to obtain an output r which it sends as a response to the query. Note, that the reduction may still be able to figure out the number of queries made by the adversary. Non-Observing reductions, the way we have defined, deliberately have a fair bit of programming capability as our focus is to identify security reductions which crucially rely on Observability.

Our first result is about online extractors[2] for NIZK-POK proposed by Fischlin in [Fis05]. The Fischlin transformation converts an interactive ZK-PoK (with some special properties) into NIZK-PoK with online extractors in the ROM. An online extractor, as defined in [Fis05], can output the witness given a (acceptable) proof and the queries made by the prover to the random oracle (*i.e.* with no rewinding). Such a NIZK-PoK can be easily converted into a secure signature scheme. We introduce the notion of Non Observing extractors, which can program the responses but not observe the queries made by the adversarial prover, and then prove that they do not exist for NIZK-PoKs obtained from the Fischlin transformation, thus also ruling out possibility of proving the security of the resulting signature scheme. Our proof idea can be extended to rule out the online extractor of [Pas03b] as well though we do not discuss this in our work. Thus, our result clearly proves that Observability is crucial for the existence of Fischlin extracor, as expected by looking at its construction.

Our second result demonstrates the existence of extractable commitment schemes with non observing extractors. The security of the extractable commitment schemes as studied in [Pas03a] seem to rely on the fact that the extractors can observe the set of query-response pairs. We show that the capability to observe is not necessary by constructing a secure extractable commitment scheme where the extractor is not allowed to observe the queries made by the committer but only allowed to program the random oracle.

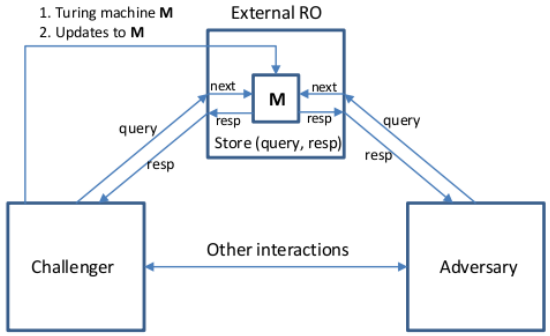
We also show (in Appendix B) that both RSA-PFDH and Schnorr signatures remain secure (under the notion of Existential Forgery with Chosen Message Attack) with Non Observing reductions. This confirms the fact that our Non Observing reductions have enough programming capabilities so as to be able to *rewind* and *embed*.

1.2 Related work.

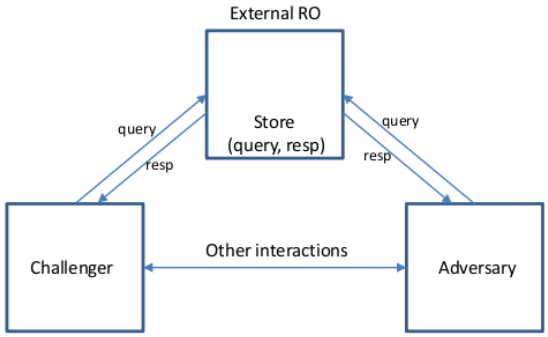
As part of this ongoing scrutiny of ROM, weaker versions of the Random Oracle model have been proposed. We briefly survey a few of them that have been proposed since [CGH04]:

- Micali and Reyzin [MR98] initiated the study of the security of signing with weak hashing by considering hash functions for which an adversary can fix arbitrarily the input-output values at polynomially many inputs.
- Nielsen [Nie] proposed a variant of the random oracle model where the random oracle is not programmable. In this model, one cannot program or set the value that the random oracle returns to any arbitrary value. He establishes separation results between proofs in the (programmable) random oracle model and non-programmable random oracle model

[2]From the point of view of using the external random oracle, extractors are no different from reductions in our results.



a) Non Observing Reductions



b) Non Observing Non Programming Reductions

Table 1: Non Observing reductions can return responses to adversary's queries without actually observing the query. Non Observing Non Programming reductions can neither observe the queries nor influence the responses.

- Liskov [Lis06] proposed models for weak hash functions where there exist the random oracle and the additional oracles that break some properties of the ROM. He listed several such oracles that provide, for example, collisions. He also proposed a general construction of a hash function from weak hash functions. Pasini and Vaudenay [PV] applied Liskov's idea to the security analysis of digital signature schemes. They considered the security of hash-then-sign type signature schemes in the random oracle model with an additional oracle that returns first-preimages. Numayama, Isshiki and Tanaka [NIT] studied the security of the Full Domain Hash signature scheme, as well as three variants thereof in weakened random oracle models.

- Mironov [Mir] relaxed the collision-resistant requirement of hash functions in hash-and-sign constructions, without addressing the need for a random oracle. He notably revisited two popular signature schemes, DSA and PSS-RSA, and proposed variants based only on the target-collision resistant property of the underlying hash function. Their proofs of security, while still dependent on random oracles, only require *short-input ones*. In [HK], Halevi and Krawczyk proved similar results.

- Unruh [Unr] pointed out the fact that it might be more realistic to consider random oracles with auxiliary input. The auxiliary input models the fact that adversary at times has access to certain information about the hash function (e.g. collisions) before the initiation of the protocol. He showed that the RSA-OAEP encryption scheme [BR] is secure in the random oracle model even in the presence of oracle-dependent auxiliary inputs

- [FLR$^+$10] was the first work to examine the programmability by (black-box) reductions in ROM and proposed three variants of reduction in ROM: Fully-Programming Reductions, Non-Programming reductions and Randomly-Programming Reductions. Our work is similar in spirit but for the property of observability in black-box reductions in ROM.

2 Non Observing Reductions

2.1 Modeling issues

In this section, we formally define Non Observing reductions. As discussed in the introduction, a non observing reduction cannot observe the interaction between the adversary and the oracle, though it can continue to 'influence' the query responses. Modelling how the reduction can influence the responses returned by the oracle machine while ensuring that the reduction can get no 'information' about the queries of the adversary is tricky. One way of modelling it is as follows: The reduction sends a (stateful) Turing machine M to the oracle. On receiving a query q from the adversary, M is executed by the oracle machine on input q to obtain an element r. The value r is then sent to the adversary as a response. Also, M needs to be programmed in such a way that for a repeat query, the same response is returned. Unfortunately, this approach does not work because consider a machine M designed in a way, such that on receiving a 'special' query from the reduction, all the queries made by the adversary till then are revealed. Thus, even though the reduction did not actually "see" the queries made by the adversary it still gets information about the queries made by the adversary through the machine M. The problem with the previous approach was that the response returned to a query could depend upon all the queries made till then. We can rectify this problem by forcing the condition that the machine M should return responses from a list which is fixed before the adversary makes any query. We allow the reduction to program this list before the random oracle is initiated. This again suffers from the following problem: After the execution of the adversary, the reduction can make special queries in such a way that the queries made by the adversary are leaked. This can be done by associating a (query number n, bit position p, bit value b) tuple with each response in the list. The reduction can then make a special query indicating the desired query number and bit position, and M can then return the response associated with the bit value of the p^{th} bit of the n^{th} query. From this, the reduction can figure out all the queries made by the adversary. The main problem with this approach is that the machine M is allowed to maintain state. We can fix this problem by requiring M to be stateless. Thus, the reduction sends a stateless machine M along with a list L such that for every query q the following is done: $M(q, L)$ is executed to obtain r. It is verified whether r is in L before sending it to the adversary. This seems to do better than the previous approaches, in that the machine can no longer return responses which are correlated to the queries made till that point. But even this approach does not work! For two different queries q_1 and q_2, $M(q_1, L)$ and $M(q_2, L)$ can give the same answer which might help the adversary in distinguishing between a pure random oracle from M because in a pure random oracle model this can happen only with negligible probability. Another problem with this approach is that the returned response can still leak information about some bits of the query. To circumvent these problems, we present a model, defined in the next subsection, which seems to not only capture non observability reasonably well but also is a more natural transition from (programmable and observable) ROM.

2.2 Our model

A Non Observing reduction works in the presence of an external random oracle which is beyond its control. The adversary and other parties make queries directly to this external random oracle and receive the responses from it as well. All communication between the adversary and the random oracle is hidden from everyone else. For ever instantiation of a random oracle, the reduction (or challenger) can send a Turing machine M to the oracle. This machine is used by the external random oracle to answer its queries in the following manner: When a random oracle receives a query, which it has never answered before, it forwards a request to manchine M using the "Next"

message. The machine M, at this point, can provide any response to the oracle as long as it is uniformly distributed in the range of the random function represented by the oracle. The reduction is also allowed to send updates to M during the lifetime of its exection. The reduction can also directly send updates to the external random oracle in the form of (*query*, *response*) pairs of its choice. Both these update procedures, allow the reduction to dynamically adjust its responses to the random oracle queries. As the reduction may be interacting with the adversary through other channels (for instance it may be simulating a signature oracle for the adversary), it can therefore use all of its knowledge from these other channels in returning the response. On receiving a response from the reduction, the oracle stores the (query, response) pair for future use and returns the response to the adversary. The reduction can also send queries of its own to the random oracle. In case of it being a fresh query, the oracle will answer it just like an adversary's fresh query, else it will return a response using the stored (query, response) pairs. Thus, our Non Observing reduction can do almost all what a reduction in ROM can do, but has no visibility of the adversary's queries ever. We show this in Figure 1. Before we formally define the model, we present some preliminaries.

For a cryptographic primitive S with the security property Π_{S}, let $\mathsf{Adv}^{\Pi}_{\mathsf{S}}(\mathcal{A})$ be defined as the success probability of the adversary $\mathcal{A}$ in violating the security property Π_{S}. For an oracle adversary $\mathcal{A}^{\mathcal{O}}$, the probability $\mathsf{Adv}^{\Pi}_{\mathsf{S}}(\mathcal{A}^{\mathcal{O}})$ is taken over the random coins of the oracle $\mathcal{O}$ as well. More formally, $\mathsf{Adv}^{\Pi}_{\mathsf{S}}(\mathcal{A}^{\mathcal{O}})$ is the weighted average of the Adv parameters when instantiated with specific oracles and the weights correspond to the probabilities that the corresponding specific oracles are chosen. This definition can analogously be extended to setting of multiple oracles. We say that an adversary $\mathcal{A}$ Π_{S}-*breaks* S if there exists a polynomial $p(\cdot)$ such that $\mathsf{Adv}^{\Pi}_{\mathsf{S}}(\mathcal{A}) > 1/p(n)$ for infinitely many n. Consider cryptographic primitives S and f with security properties Π_{S} and Π_f respectively. Let $\mathcal{A}$ denote an attacker on the primitive S and let $\mathcal{R}$ denote a reduction from f to S, *i.e.*, $\mathcal{R}$ uses $\mathcal{A}$ to attack primitive f. We will denote by $\mathcal{O}$ to denote a random oracle that chooses a response uniformly at random from its range and by $\mathcal{O}(M)$ an oracle that uses the machine M produced by reduction $\mathcal{R}$ to provide the responses to its queries. We will use $\mathcal{R}^{\boxed{\mathcal{A}^{\mathcal{O}(M)}}}$ to denote the fact that the interaction between $\mathcal{A}$ and $\mathcal{O}(\mathcal{M})$ is hidden from $\mathcal{R}$ and $\mathcal{R}^{\boxed{\mathcal{A}^{\mathcal{O}(M)}},\mathcal{O}(M)}$ to denote such a reduction with its own oracles access to $\mathcal{O}(\mathcal{M})$.

Definition 1. *There exists a black-box Non Observing reduction from f to S if there exists a PPT ITM machine $\mathcal{R}$ that outputs M with the property that if $\mathcal{A}^{\mathcal{O}(M)}$ Π_{S}-breaks S, making $q_{\mathcal{O}}$ queries to the oracle $\mathcal{O}(M)$, then $\mathcal{R}^{\boxed{\mathcal{A}^{\mathcal{O}(M)}},\mathcal{O}(M)}(w)$ Π_f-breaks f, where $\mathcal{O}(M)$ is computationally indistinguishable from $\mathcal{O}$.*

We define a Non Observing Non Programming reduction as a Non Observing reduction that can no longer provide the responses to the oracle. Thus, such a reduction works with oracle $\mathcal{O}$ rather than $\mathcal{O}(M)$. See Table 1 for a diagramatic representation of the reduction.

Definition 2. *There exists a black-box Non Observing Non Programming reduction from f to S if there exists a PPT ITM machine $\mathcal{R}$ with the property that if $\mathcal{A}^{\mathcal{O}}$ Π_{S}-breaks S, making $q_{\mathcal{O}}$ queries to the oracle $\mathcal{O}$, then $\mathcal{R}^{\boxed{\mathcal{A}^{\mathcal{O}}},\mathcal{O}}(w)$ Π_f-breaks f.*

3 Non Observing Online Extractors

In this section, we explore online extractors in the non observability framework. The construction of online extractors in the non interactive zero knowledge proof of knowledge (NIZKPoK) was studied by [Fis05] though it was first discussed in [SG98]. Informally, an online extractor can extract a

witness for an input instance, given a proof (accepted by an honest verifier) and all the queries made by the prover to the random oracle. The online extractors deviate from the traditional extractors in that rewinding is not necessary to extract the witness. As remarked in [Fis05], rewinding leads to loose security reductions and hence online extractors can be useful to obtain tight security results.

The formal definition of online extractors for a non interactive zero knowledge proof of knowledge is given below.

Definition 3 (Online Extractor)**.** *[Fis05] There exists a probabilistic polynomial time algorithm K such that the following holds for any algorithm A. Let $\mathcal{O}$ be a random oracle, $(x, \pi) \leftarrow A^{\mathcal{O}}(k)$ and $Q_{\mathcal{O}}(A)$ be the sequence of queries of A to $\mathcal{O}$. Let $w \leftarrow K(x, \pi, Q_{\mathcal{O}}(A))$. Then as a function of k,*

$$Pr[(x, w) \notin W_k \wedge V^{\mathcal{O}}(x, \pi) = 1] \approx 0$$

In the above definition, the online extractor does not have any power to choose the random oracle. In other words, the extractor is not allowed to program the random oracle. We first describe the result from [Fis05] which gives the construction of online extractors. [Fis05] gave a transformation, termed as Fischlin transformation, to convert an interactive proof of knowledge (defined as Fiat Shamir Proof of knowledge; see definition 8 in Appendix) to a NIZK-PoK which has an online extractor.

We give an informal description of the Fischlin transformation. Fischlin transformation converts a 3-message interactive ZKPoK (P_{FS}, V_{FS}) to a non interactive ZKPoK (P^H, V^H), where H is the random oracle, as follows. P^H executes logarithmically many copies, denoted by r, of the underlying prover P_{FS}. In each execution it gets the commitment com_i from P_{FS}. In the i^{th} execution, P^H then sequentially checks whether there exists any challenge ch_i from 0 to $2^t - 1$ such that $H(\mathsf{com}, i, ch_i, resp_i)$ has all its last b bits as 0, where b is typically logarithmic in the security parameter and $resp_i$ is the response returned by the i^{th} copy of P_{FS} on challenge ch_i. If no such challenge exists then P^H picks the challenge ch_i for which $H(\mathsf{com}, i, ch_i, resp_i)$ is minimum among all other challenges. Finally, P^H composes the proof $(com_i, ch_i, resp_i)_{1 \leq i \leq r}$. The verifier V_{FS} on input $x, (com_i, ch_i, resp_i)_{1 \leq i \leq r}$ checks whether V_{FS} accepts the proof $(x, com_i, ch_i, resp_i)$ for all $1 \leq i \leq r$. And also, it checks whether the last b bits of summation of $H(\mathsf{com}, i, ch_i, resp_i)$ is at most logarithmic in the security parameter. We formally describe the Fischlin transformation below.

Definition 4 (Fischlin Transformation)**.** *[Fis05] Let H be a random oracle. Let (P_{FS}, V_{FS}) be an interactive Fiat-Shamir proof of knowledge with challenges of $\ell = \ell(k) = O(log(k))$ bits for relation W. Define the parameters b, r, S, t (as functions of k) for the number of test bits, repetitions, maximum sum and trial bits such that $br = \omega(logk)$, $2^{t-b} = \omega(logk)$, b, r, $t = O(logk)$, $S = O(r)$ and $b \leq t \leq \ell$. Define the following non-interactive proof system for relation W in the random oracle model, where the random oracle maps to b bits.*

- ***Prover.*** *The prover P^H on input (x,w) first runs the prover $P_{FS}(x, w)$ in r independent repetitions to obtain r commitments $com_1, \cdots, com_r$. Let $\mathsf{com} = (com_1, \cdots, com_r)$. Then P_H does the following, either sequentially or in parallel for each repetition i. For each $ch_i = 0, 1, 2, \cdots, 2^t - 1$ (viewed as t-bit strings) it lets P_{FS} compute the final responses $resp_i = resp_i(ch_i)$ by rewinding, until it finds the first one such that $H(x, \mathsf{com}, i, ch_i, resp_i) = 0^b$; if no such tuple is found then P^H picks the first one for which the hash value is minimal among all 2^t hash values. The prover finally outputs $\pi = (com_i, ch_i, resp_i)_{i=1,2,\ldots,r}$.*

- ***Verifier.*** *The verifier V^H on input x and $\pi = (com_i, ch_i, resp_i)_{i=1,2,\ldots,r}$ accepts if and only if $V_{1,FS}(x, com_i, ch_i, resp_i) = 1$ for each $i = 1, 2, \cdots, r$, and if $\Sigma_{i=1}^{r} H(x, com, ch_i, resp_i) \leq S$.*

We first give an intuitive description of the proof of online extractability of the above transformation. Consider an adversary who, on input x, has produced a proof π without having a witness for x. Let Q be the set of queries made by the adversary to the random oracle. We first claim that there cannot exist two queries $(\mathsf{com}, i, ch_i, resp_i)$ and $(\mathsf{com}, i, ch_i^*, resp_i^*)$ in Q such that both $(com_i, ch_i, resp_i)$ and $(com_i, ch_i^*, resp_i^*)$ are accepted by the verifier V_{FS}. If there existed two such queries then by the special soundness property of (P_{FS}, V_{FS}), the witness can be extracted. The extraction can be done by the online extractor since he can observe the queries made by the adversary. Hence, once the commitment tuple is fixed the adversary can query the random oracle for one particular challenge ch_i for i from 1 to r. Let s_i be the value output by the random oracle for the challeng e ch_i. Using simple probability arguments it can be shown that the summation of s_i for all the repetitions is negligible. Thus, for a given commitment tuple adversary succeeds with negligible probability in producing an accepting proof corresponding to that tuple. Since, adversary can try only polynomially many commitment tuples he can succeed in producing an accepting proof only with negligible probability.

The capability of the online extractor to extract the witness comes from crucial fact that the extractor can observe the queries made by the prover. If the extractor is not allowed to see the queries made by the prover and not allowed to program the random oracle then it can be seen that the extractor cannot extract the witness from the proof. In other words, there does not exist an online extractor for any NIZKPoK that neither programs the random oracle nor observes the queries made by the prover. The reason is that, if such an extractor were to exist then a malicious verifier can simply run the extractor to get the witness thus contradicting the zero knowledge property of the protocol. The same is not clear when the extractor is allowed to program the random oracle. More precisely, we want to understand whether there exist online extractors for NIZKPoK which are allowed do some limited programming of the random oracle but not allowed to observe. We term this class of online extractors as non observing online extractors and formally define them below.

Definition 5 (Non Observing Online Extractors)**.** *There exists a probabilistic polynomial time algorithm $K = (K_1, K_2)$ such that for large enough k the following holds for any algorithm A. There exists a polynomial $p(k)$ such that $(M, aux) \leftarrow K_1(k, p(k))$ and $(x, \pi) \leftarrow A^{\mathcal{O}(M)}(k)$ making $q_{\mathcal{O}} \leq p(k)$ queries to the oracle $\mathcal{O}(M)$. Then we have that $w \leftarrow K_2^{\mathcal{O}(M)}(x, \pi, M, aux)$. Then as a function of k,*

$$Pr[(x, w) \notin W_k \wedge V^{\mathcal{O}(M)}(x, \pi) = 1] \approx 0$$

We first show that there is a NIZKPoK which has a non observing extractor in the random oracle model. Consider a NIZKPoK (P, V) in the common reference string model. Construction of NIZKPoK in the common reference string model has been well studied in literature. See [DSP92] for one such example. The fact that (P, V) is a NIZKPoK means that it has an extractor $E = (E_0, E_1)$ which executes as follows. On input security parameter, E_0 produces a pair of strings (σ, aux). Let π be an acceptable proof produced by an adversary A on input x as well as σ. Then, E_1 on input (x, π, σ, aux) outputs a witness w for x with non negligible probability. We construct (P^*, V^*) in the random oracle model from (P, V) as follows. P^* on input x, queries the random oracle on the point x to get the response σ. P^* then executes $P(x, \sigma)$ to obtain π. The verifier V^*, on receiving π, first queries x to the random oracle to get σ and then executes $V(x, \pi, \sigma)$. V^* then outputs whatever V outputs. We can construct a non observing online extractor $E^* = (E_0^*, E_1^*)$ as follows. Consider an adversarial prover A. Let $q_{\mathcal{O}}$ be the number of queries made by the adversary. E_0^* executes E_0 for $q_{\mathcal{O}}$ times to obtain the strings $\Big((\sigma_1, aux_1), \ldots, (\sigma_{q_{\mathcal{O}}}, aux_{q_{\mathcal{O}}})\Big)$. E_0^* then constructs a Turing machine M which on being invoked for the i^{th} time with input *next* outputs the string σ_i. E_0^* then sends

the Turing machine M to the random oracle. After receiving the proof π from an adversary A, E_1^* then queries x to the random oracle to obtain σ_i and then it executes $E_1(x, \pi, \sigma_i, aux_i)$ to obtain w. From the extractability property of (P, V), it follows that if π is an acceptable proof then E^* outputs a witness for x with non negligible probability. This shows the existence of a NIZKPoK having non observing extractors.

The natural question to ask now is whether this is true for all NIZKPoK in the random oracle model. That is, whether there exists non observing online extractors for all NIZKPoK. We show that this is not true. In fact, we show that all the NIZKPoKs that are obtained from the Fischlin transformation do not have non observing online extractors. We formalize this result in the following theorem. To understand the theorem given below, we refer the reader to the definitions of special zero knowledge and one way instance generators in Appendix A.

Theorem 1. *Consider a relation W having a one-way instance generator $\mathcal{I}$. Let (P^H, V^H) be a non-interactive zero-knowledge proof of knowledge obtained by applying the Fischlin transformation to an interactive Fiat-Shamir proof of knowledge, (P_{FS}, V_{FS}) defined for the relation W. Then, there does not exists a Non Observing extractor for (P^H, V^H)* [3].

Proof: We show that if a Non Observing online extractor K exists for (P^H, V^H), then we can construct an algorithm B, termed as inverter, which does the following. It takes as input x where x is produced by the one way instance generator, $\mathcal{I}$. It then outputs a witness w for x with non-negligible probability such that $(x, w) \in W$. This contradicts the fact that $\mathcal{I}$ is a one-way instance generator for the relation W. We now give the construction of B. The algorithm B on input x executes the following steps.

Step 1) B first executes K_1 to get the Turing machine M which is passed on to the random oracle. It then makes $2^t r$ queries to a copy of M to obtain a list L. The reason why the size of the list is set to $2^t r$ is because the honest prover makes at most $2^t r$ queries. Note, the list L is the same as the first $2^t r$ responses returned by $\mathcal{O}(M)$.

Step 2) B chooses the r challenges $ch_i, i \in [1, r]$ by looking up the list L. As B has access to the list L of hash responses, it can figure out the exact challenges which an honest prover will include in his proof by imitating the honest prover's strategy. B considers the first 2^t elements in the list L which $K_1(1^k)$ outputs. If there is an element whose least significant b bits are 0 (if there are many such elements pick the one with the least index in L), then B assigns $ch_1^{'}$ to be its index else assign $ch_1^{'}$ to be the index of the minimum among the first 2^t elements. If $(ch_1^{'})^{th}$ element corresponds to an element whose least significant b bits are 0, then B repeats the process to compute $ch_2^{'}$ starting from the $(\beta_1^{'} + 1)^{th}$ element of L. Whereas if $(ch_1^{'})^{th}$ element corresponds to the smallest element in the first 2^t elements, repeat the above process starting from the $(2^t + 1)^{th}$ element of the list. Thus, using the above approach B computes $ch_i^{'}$, for all $i \in [1, r]$. Assign ch_i to be $ch_i^{'}$ if $i = 1$ else $ch_i = ch_i^{'} - ch_{i-1}^{'}$.

Step 3) B executes the special zero knowledge simulator, Z, of the interactive Fiat-Shamir proof of knowledge, (P_{FS}, V_{FS}), at (x, ch_i, YES) to obtain com_i and $resp_i$ for all $i \in [1, r]$. Let $\mathsf{com} = (com_1, \ldots, com_r)$.

Step 4) B makes $ch_r^{'}$ queries to the random oracle $\mathcal{O}(M)$ as follows. At query numbers $ch_i^{'}, \forall i \in [1, r]$ it queries the oracle with $(x, \mathsf{com}, i, ch_i, resp_i)$. At all other points it queries the oracle at

[3] We say that there does not exist a non observing online extractor for a proof system if for every PPT extractor there exists a PPT adversarial prover such that the probability that the adversarial prover produces an accepting proof and at the same time the non observing extractor cannot extract a witness is non-negligible.

$(x, \mathsf{com}, 1, 0, 0)$, where com is a r-sized vector with each element chosen randomly from the commitment space.

Finally, B produces $\pi_B = ((com_1, ch_1, resp_1), (com_2, ch_2, resp_2), \ldots, (com_r, ch_r, resp_r))$ as the proof. The following lemma proves that no probabilistic polynomial time algorithm (even with access to the Turing machine M) can distinguish proof π_B from a proof $\pi_{P^{\mathcal{O}(L)}}$ produced by an honest prover $P^{\mathcal{O}(M)}$, where $P^{\mathcal{O}(M)}$ is same as the prover P^H but with random oracle H replaced with $\mathcal{O}(M)$.

Lemma 1. *Let D be a probabilistic polynomial time algorithm. The following two distributions are indistinguishable.*

- *$K_1(1^k, 2^t r) \rightarrow M$. B is executed with input (x, M) and oracle access to $\mathcal{O}(M)$ which then outputs π_B. Output $D^{\mathcal{O}(M)}(x, M, \pi_B)$.*
- *$K_1(1^k, 2^t r) \rightarrow M$. $P^{\mathcal{O}(M)}$ is executed with input x, w and oracle access to $\mathcal{O}(M)$ which then outputs $\pi_{P^{\mathcal{O}(M)}}$. Output $D^{\mathcal{O}(M)}(x, M, \pi_{P^{\mathcal{O}(M)}})$.*

Proof. B's strategy of producing the challenges and the special zero-knowledge property ensure that the distributions of π_B is computationally indistinguishable from $\pi_{P^{\mathcal{O}(M)}}$ and thus D cannot distinguish the proof transcript π_B from $\pi_{P^{\mathcal{O}(M)}}$. But as the queries made to the random oracle by B are different from that of an honest prover, D could try guessing the queries. We show below that this happens with negligible probability. Let the set of queries made by $P^{\mathcal{O}(M)}$ (respectively, B) to $\mathcal{O}(M)$ during its execution be $Query_{P^{\mathcal{O}(M)}}$ (resp. $Query_B$). Denote the set of respones returned by $\mathcal{O}(M)$ corresponding to $Query_{P^{\mathcal{O}(M)}}$ (resp. $Query_B$) by $Resp_{P^{\mathcal{O}(M)}}$ (resp. $Resp_B$). To prove the theorem, we first make the claim that $Resp_{P^{\mathcal{O}(M)}}$ is in fact the same as $Resp_B$. This follows from the description of B and $P^{\mathcal{O}(M)}$. Let $\pi_{P^{\mathcal{O}(M)}} = (com_i, ch_i, resp_i)_{1 \leq i \leq r}$ and $\pi_B = (com_i^{'}, ch_i^{'}, resp_i^{'})_{1 \leq i \leq r}$. We then claim that in both the cases, if the distinguisher makes a query q to $\mathcal{O}(M)$ then q belongs to $(Query_{P^{\mathcal{O}(M)}} \backslash \{(com_1, \cdots, com_r, i, ch_i, resp_i) : 1 \leq i \leq r\})$ (resp. q belongs to $(Query_B \backslash \{(com_1^{'}, \cdots, com_r^{'}, i, ch_i^{'}, resp_i^{'})\})$ with negligible probability. To prove this claim, consider the following cases.

Case 1. $P^{\mathcal{O}(M)}$: Without loss of generality let q be equal to $(com_1, \cdots, com_r, i, ch, resp)$ for some $i \in \{1, \ldots, r\}$. Since $q \in (Query_{P^{\mathcal{O}(M)}} \backslash \{(com_1, \cdots, com_r, i, ch_i, resp_i) : 1 \leq i \leq r\})$, it should happen that $(com_i, ch, resp)$ is an accepting transcript (this is because the honest prover follows the protocol and hence all its queries correspond to accepting transcripts). We now have two accepting transcripts $(com_i, ch_i, resp_i)$ (from $\pi_{P^{\mathcal{O}(M)}}$) and $(com_i, ch, resp)$ using which we can extract a witness for x. Using this observation, we can construct a polynomial time procedure which can extract a witness from the input instance. Now, we make the observation that we could have executed the zero knowledge simulator to obtain π_{Sim} (which is indistinguishable from $\pi_{P^{\mathcal{O}(M)}}$) and then using the strategy of D (in a non black box way) to find q we could then extract a witness for the input instance. Since such an approach gives us a probabilistic polynomial time algorithm to compute the witness, our assumption that the considered relation has a one-way instance generator is violated.

Case 2. B: Consider the query $q' = (com_1^{''}, \ldots, com_r^{''}, 0, 0, 0)$ in the set $(Query_B \backslash \{(com_1^{'}, \cdots, com_r^{'}), i, ch_i^{'}, resp_i^{'}\})$. The probability that $q' = q$ is negligible since $com_1^{''}, \ldots, com_r^{''}$ is picked uniformly at random.

From the above two cases it can be inferred that the distributions $D^{\mathcal{O}(M)}(x, L, \pi_{P^{\mathcal{O}(M)}})$ and $D^{\mathcal{O}(M)}(x, L, \pi_B)$ are indistinguishable. □

Now, consider the following probabilistic polynomial time algorithm.

Input: Instance x obtained as the output of the one-way instance generator $\mathcal{I}$.
Output: Witness w.
1. $K_1(1^k, 2^t r) \to M$.
2. $B^{\mathcal{O}(M)}(x, M) \to \pi_B$.
3. $K_2^{\mathcal{O}(M)}(x, M, \pi_B) \to w$.

Using Lemma 3 and the construction of B, it can be seen that the above algorithm outputs w with non-negligible probability for input x such that $(x, w) \in W$ contradicting the assumption that W has a one-way instance generator. Thus, Non Observing online extractors do not exist for (P^H, V^H).

4 Extractable commitment schemes

The notion of extractable commitment schemes has been studied [Pas03a, ACP09] in the common reference string model as well as the random oracle model. Extractable commitments are commitment schemes equipped with an additional algorithm, called the *extractor*, which can recover the committed value given the commitment as well as the trapdoor to the CRS (in the CRS model) or given access to the queries to the random oracle made by the committer to generate the commitment. In this section, we study extractable commitments in the random oracle model. If the extractor is allowed to observe the queries made by the committer then there is a simple commitment scheme as described in [Pas03a]. We give an example of an non interactive extractable commitment scheme where the extractor is allowed the program the random oracle but not allowed to observe the queries made by the committer to the random oracle. We now define the notion of extractable commitment schemes in the random oracle model when the extractor is non-observing. For preliminaries on commitment schemes, refer to Appendix D.

Definition 6. *Consider a non-interactive commitment scheme (C, R) defined in the random oracle model where C is the committer and R is the receiver. We say (C, R) is an extractable commitment scheme with non observing extractors if there exists a PPT extractor $K = (K_1, K_2)$ which does the following. The algorithm K_1 on input security parameter ouputs a Turing machine M along with auxillary information aux. Let the output of the committer with access to $\mathcal{O}(M)$ be the commitment c. Then, K_2 on input (c, M, aux) and access to the random oracle $\mathcal{O}(M)$ outputs m with probability negligibly close to the probability that the committer succeeds in decommitting to m.*

We now describe our extractable commitment scheme. Consider the random oracle H mapping from $\{0,1\}^*$ to $\{0,1\}^n$, where n is some polynomial in the security parameter. Let $k_1 < n$ and k_2 be polynomials in the security parameter.

ExtCom.
Commit phase:
On input m, the committer picks a value R from $\{0,1\}^{k_2}$ uniformly at random. It then sends the queries $(m, R, 1), \ldots, (m, R, l)$ to H to receive the responses $(h'_1, \ldots, h'_l)$, where l is the length of m. It then picks a non-zero key K from the space $\{0,1\}^{n-k_1} \backslash \{0^{n-k_1}\}$ uniformly at random [4]. It then computes $(h_1, \ldots, h_l)$ as follows: $h_i = h'_i$ if $m_i = 0$ (m_i denotes the i^{th} bit of m) else

[4]This means that it picks a *non-zero* key from the space $\{0,1\}^{n-k_1}$ uniformly at random.

$h_i = h'_i \oplus (K||0^{k_1})$. It then sends $(h_1, \ldots, h_l)$ as the commitment.

Reveal phase:
The committer sends (m, R, K) as the decommitment. Let the input received by the receiver during the commit phase be $(h_1, \ldots, h_l)$. The receiver accepts if the following conditions are satisfied.
1. $K \neq 0$.
2. $H(m, R, i) = h_i$ if $m_i = 0$, else $H(m, R, i) \oplus (K||0^{k_1}) = h_i$.

We now show that the above commitment scheme satisfies both the hiding and the binding properties. We first show that the commitment scheme satisfies computational hiding property. Observe that the only way for the adversary to distinguish the commitments corresponding to two different messages is when it queries the random oracle on the message which is contained in the commitment. Since the adversary runs in polynomial time the probability that it guesses R (picked by the committer) correctly is negligible, since R is polynomial in the security parameter. This shows that no PPT adversary can distinguish commitments corresponding to two different messages. We now show that the extractable commitment scheme satisfies the binding property. Let m_1 and m_2 be two distinct messages which correspond to the opening of a commitment $c = (h_1, \ldots, h_l)$. Without loss of generality, assume that the i^{th} bit of m_1 is different from the i^{th} bit of m_2. Also assume that the i^{th} bit of m_1 is 1 while the i^{th} bit of m_2 is 0. Since c can be opened to both m_1 and m_2, this means that $H(m_1, R, i) \oplus (K||0^{k_1}) = H(m_2, R, i)$. In other words, the last k_1 bits of $H(m_1, R, i)$ is the same as the last k_1 bits of $H(m_2, R, i)$. But this can happen only with probability $\frac{1}{2^{k_1}} = negl(k)$ which in turn means that with negligible probability the commitment c can be opened to both m_1 and m_2 for any two distinct messages m_1 and m_2. This shows that the commitment scheme satisfies binding property.

The following theorem shows that the above described commitment scheme is a secure extractable commitment scheme even when the extractor is non observing, as per Definition 6.

Theorem 2. ***ExtCom*** *is an extractable commitment scheme secure in the random oracle model. Further, the extractor in the commitment scheme is non-observing.*

Proof. We demonstrate the existence of an extractor K for the commitment scheme **ExtCom** which succeeds in extracting the message from the commitment with non-negligible probability. To do this, we crucially use the fact that the extractor can program the random oracle. Further, the extractor we construct is a non-observing one: the extractor cannot see the interaction between the oracle and the adversary. We now define the extractor K which is decomposed into algorithms K_1 and K_2.

K_1 picks a list L of responses of length q_h uniformly at random. It then constructs a machine M which does the following. On being invoked for the i^{th} time with the *next* query, M outputs the i^{th} entry in the list L. Note that M is a stateful machine and so can store the number of times it has been invoked till now. K_1 then sends M to the oracle $\mathcal{O}$. We now define K_2: On input a commitment $(h_1, \ldots, h_l)$ along with the Turing machine M and oracle access to $\mathcal{O}(M)$, K_2 does the following. It executes $M(next)$ for q_h times to get a list L. It then computes a message m such that for all $i = 1, \ldots, l$ it assigns the i^{th} bit of m to be 0 *if* h_i is found in L else it assigns m_i to be 1. It then outputs m.

We claim that, with overwhelming probability the output of K_2 is the same as the message decommitted by the sender during the reveal phase. More precisely, if the sender successfully decommits to m then K outputs m' such that $m' = m$ with overwhelming probability. To prove this claim, we first consider the event that $m_i^{'} \neq m_i$ for some i from $1, \ldots, l$. To show that this event is negligible, observe that it suffices to show that the event $resp_i = resp_j \oplus (K||0^{k_1})$ is negligible for

any non-zero K, where $resp_i, resp_j$ are any two responses returned by the machine M. This follows from the following two cases: if $resp_i = resp_j$ then $resp_i \neq resp_j \oplus (K||0^{k_1})$ since K is non-zero and if $resp_i \neq resp_j$ then $resp_i$ can be same as $resp_j \oplus (K||0^{k_1})$ with negligible probability since the probability that the last k_1 bits of $resp_i$ and $resp_j$ are the same is $\frac{1}{2^{k_1}}$ (because the responses returned by M are picked uniformly at random). This proves that $resp_i = resp_j \oplus (K||0^{k_1})$ with negligible probability which further proves that the probability that $m' \neq m$ is negligible. This completes the proof. □

These ideas can be extended further, in a straightforward manner, to construct plaintext aware encryption schemes having non observing plaintext extractors.

References

[ACP09] M. Abdalla, C. Chevalier, and D. Pointcheval. Smooth projective hashing for conditionally extractable commitments. *Advances in Cryptology-CRYPTO 2009*, pages 671–689, 2009.

[BR] M. Bellare and P. Rogaway. Optimal Asymmetric Encryption. In *Advances in Cryptology* - EUROCRYPT *'94*, pages 92–111.

[BR93] M. Bellare and P. Rogaway. Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In *Proceedings of the First ACM Conference on Computer and Communications Security*, pages 62–73, 1993.

[BR96] Mihir Bellare and Phillip Rogaway. The exact security of digital signatures - how to sign with rsa and rabin. In *EUROCRYPT*, pages 399–416, 1996.

[CGH04] R. Canetti, O. Goldreich, and S. Halevi. The Random Oracle Methodology, Revisited. *J. Assoc. Comput. Mach.*, 51(4):557–594, 2004.

[DSP92] A. De Santis and G. Persiano. Zero-knowledge proofs of knowledge without interaction. In *Foundations of Computer Science, 1992. Proceedings., 33rd Annual Symposium on*, pages 427–436. IEEE, 1992.

[Fis05] Marc Fischlin. Communication-efficient non-interactive proofs of knowledge with online extractors. In *In CRYPTO 2005*, pages 152–168. Springer-Verlag, 2005.

[FLR+10] Marc Fischlin, Anja Lehmann, Thomas Ristenpart, Thomas Shrimpton, Martijn Stam, and Stefano Tessaro. Random oracles with(out) programmability. In *ASIACRYPT*, pages 303–320, 2010.

[GMR88] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Comput.*, 17(2):281–308, 1988.

[HK] S. Halevi and H. Krawczyk. Strengthening Digital Signatures Via Randomized Hashing. In *Advances in Cryptology* - CRYPTO *2006*, pages 41–59.

[Lis06] M. Liskov. Constructing an Ideal Hash Function from Weak Ideal Compression Functions. In *Selected Areas in Cryptography, 13th Annual International Workshop, SAC 2006*, pages 358–375, 2006.

[Mir] I. Mironov. Collision-Resistant No More: Hash-and-Sign Paradigm Revisited. In *9th International Workshop on Practice and Theory in Public Key Cryptography, PKC 2006*, pages 140–156.

[MR98] S. Micali and L. Reyzin. Signing with Partially Adversarial Hashing. Technical Report 575, MIT/LCS/TM, 1998.

[MvOV97] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography.* Discrete Mathematics and its Applications. CRC Press, Boca Raton, FL, 1997.

[Nie] J. B. Nielsen. Separating Random Oracle Proofs from Complexity Theoretic Proofs: The Non-committing Encryption Case. In *Advances in Cryptology* - Crypto *2002*, pages 111–126.

[NIT] A. Numayama, T. Isshiki, and K. Tanaka. Security of Digital Signature Schemes in Weakened Random Oracle Models. In *11th International Workshop on Practice and Theory in Public Key Cryptography, PKC 2008*, pages 268–287.

[Pas03a] R. Pass. On deniability in the common reference string and random oracle model. *Advances in Cryptology-CRYPTO 2003*, pages 316–337, 2003.

[Pas03b] Rafael Pass. On deniability in the common reference string and random oracle model. In *In proceedings of CRYPTO 03, LNCS series*, pages 316–337. Springer-Verlag, 2003.

[PS00] D. Pointcheval and J. Stern. Security Arguments for Digital Signatures and Blind Signatures. *J. Cryptology*, 13(3):361–396, 2000.

[PV] S. Pasini and S. Vaudenay. Hash-and-Sign with Weak Hashing Made Secure. In *Information Security and Privacy: 12th Australasian Conference, ACISP 2007*, pages 338–354.

[PV05] Pascal Paillier and Damien Vergnaud. Discrete-log-based signatures may not be equivalent to discrete log. In *ASIACRYPT*, pages 1–20, 2005.

[RSA83] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems (reprint). *Commun. ACM*, 26(1):96–99, 1983.

[SG98] V. Shoup and R. Gennaro. Securing threshold cryptosystems against chosen ciphertext attack. *Advances in CryptologyEUROCRYPT'98*, pages 1–16, 1998.

[Unr] D. Unruh. Random Oracles and Auxiliary Input. In *Advances in Cryptology* - Crypto *2007*, pages 205–223.

A Preliminaries

Let $\lambda \in \mathbb{N}$ be the security parameter. We say that a function is *negligible* in λ if it is asymptotically smaller than the inverse of any fixed polynomial. More precisely, a function $\eta(\lambda)$ from non-negative integers to reals is called negligible in λ if for every constant $c > 0$, $\exists \lambda_c$ such that $\forall \lambda > \lambda_c$, $|\eta(\lambda)| < \lambda^{-c}$. Otherwise, $\eta(\lambda)$ is said to be *non-negligible* in λ.

A.1 Signatures

Definition 7 (Digital Signatures Scheme.)**.** *A signature scheme [GMR88] consists of three algorithms* $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ *for key generation, signing and verification, respectively. More concretely,*

- ***Key Generation.*** $\mathsf{Gen}(1^\lambda)$ *generates a public key secret key pair,* $(\mathtt{pk}, \mathtt{sk})$*.*
- ***Signing.*** $\mathsf{Sign}(\mathtt{sk}, m)$ *outputs a signature* σ *on message* m*.*
- ***Verification.*** $\mathsf{Verify}(\mathtt{pk}, m, \sigma)$ *outputs a bit.*

A signature scheme is said to be complete *if, for every* $(\mathtt{pk}, \mathtt{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$*, any message* $m \in \{0,1\}^\lambda$ *and any* $\sigma \leftarrow \mathsf{Sign}(\mathtt{sk}, m)$*, we have* $\mathsf{Verify}(\mathtt{pk}, m, \sigma) = 1$*.*

Attack and Forgery types: An adversary can broadly mount two kinds of attacks against signature schemes: *Key-only attack* (KOA, also called no-message attack) and *Chosen message attack* (CMA). In the first attack, the attacker knows only the public key of the signer while in the latter, the attacker can also obtain signatures on messages of his choice adaptively. The result of the attacks by the adversary are classified as follows:

1. Total Break - The adversary learns the secret key of the signer.
2. Universal Forgery (UF) - The adversary can produce a valid signature for any given message.
3. Existential Forgery(EF) - The adversary can produce a new message signature pair.

Thus, by combining the attack type and the attack result, one can talk about various levels of security for digital signatures. For instance, a (ε, τ)-universal forger under key-only attack is an adversary who, knowing only the public key, can produce a signature on any given message with probability ε in time at most τ. An (ε, τ, q_h)-universal forger under key-only attack is the same adversary *in the Random Oracle Model* who, makes at most q_h hash queries to the random oracle. For details refer to [MvOV97, PV05].

Security notions are obtained by coupling an adversarial goal with an attack model. We distinguish between several notions for which general results are immediate, as shown on Figure 1. We refer the reader to the extensive cryptographic literature for a more formal definition of these security notions.

Existential forgeries	$\mathsf{EF\text{-}KOA}[\mathcal{S}]$	$\Rightarrow$	$\mathsf{EF\text{-}CMA}[\mathcal{S}]$
	$\Uparrow$		$\Uparrow$
Universal forgeries	$\mathsf{UF\text{-}KOA}[\mathcal{S}]$	$\Rightarrow$	$\mathsf{UF\text{-}CMA}[\mathcal{S}]$
	$\Uparrow$		$\Uparrow$
Breakability	$\mathsf{BK\text{-}KOA}[\mathcal{S}]$	$\Rightarrow$	$\mathsf{BK\text{-}CMA}[\mathcal{S}]$
Goal vs. Attack	**Key only**		**Chosen message**

Figure 1: Major security notions for signature schemes. $\mathcal{S}$ denotes an arbitrary signature scheme and $P_1 \Leftarrow P_2$ means that P_1 is polynomially reducible to P_2. Security notions are defined by their underlying problem *e.g.* $\mathsf{UF\text{-}KOA}[\mathcal{S}]$ denotes the problem of computing a universal forgery under a key-only attack.

Definition 8. *A Fiat Shamir proof of knowledge (with $l(k)$-bit challenges) for relation W is pair (P, V) of probabilistic polynomial time algorithms $P = (P_0, P_1), V = (V_0, V_1)$ with the following properties.* [*Completeness.*] *For any parameter k, any $(x, w) \in W_k$, any $(P(x, w), V_0(x)) \to (\alpha, \beta, \gamma)$ it holds $V_1(x, \alpha, \beta, \gamma) = 1$.*
[*Commitment Entropy.*] *For parameter k, for any $(x, w) \in W_k$, the min-entropy of $P_0(x, w) \to \alpha$ is superlogarithmic in k.*
[*Public Coin.*] *For any k, any $(x, w) \in W_k$ any $\alpha \leftarrow P_0(x, w)$ the challenge $V_0(x, \alpha) \to \beta$ is uniform on $\{0, 1\}^{l(k)}$.*
[*Unique responses.*] *For any probabilistic polynomial time algorithm A, for parameter k and $A(k) \to (x, \alpha, \beta, \gamma, \gamma')$ we have, as a function of k,*

$$Pr[V_1(x, \alpha, \beta, \gamma) = V_1(x, \alpha, \beta, \gamma') = 1 \wedge \gamma \neq \gamma'] \approx 0$$

[*Special Soundness.*] *There exists a probabilistic polynomial time algorithm K, the knowledge extractor, such that for any k, any $(x, w) \in W_k$, any pairs $(\alpha, \beta, \gamma), (\alpha, \beta', \gamma')$ with $V_1(x, \alpha, \beta, \gamma) = V_1(x, \alpha, \beta', \gamma') = 1$ and $\beta \neq \beta'$, for $K(x, \alpha, \beta, \gamma, \beta', \gamma') \to w'$ it holds $(x, w') \in W_k$.*
[*Honest-Verifier Zero-Knowledge.*] *There exists a probabilistic polynomial time algorithm Z, the zero-knowledge simulator, such that for any pair of probabilistic polynomial time algorithms $D = (D_0, D_1)$ the following distributions are computationally indistinguishable:*

- *Let $D_0(k) \to (x, w, \delta)$ and $(P(x, w), V_0(x)) \to (\alpha, \beta, \gamma)$ if $(x, w) \in W_k$ and $\perp \to (\alpha, \beta, \gamma)$ otherwise. Output $D_1(\alpha, \beta, \gamma, \delta)$.*
- *Let $D_0(k) \to (x, w, \delta)$ and $Z(x, YES) \to (\alpha, \beta, \gamma)$ if $(x, w) \in W_k$ and $Z(x, NO) \to (\alpha, \beta, \gamma)$. Output $D_1(\alpha, \beta, \gamma, \delta)$.*

Definition 9 (Special Zero-Knowledge)**.** *There exists a probabilistic polynomial-time algorithm X, the special zero-knowledge simulator, such that for any pair of probabilistic polynomial-time algorithms $D = (D_0, D_1)$ the following distributions are computationally indistinguishable: (-) Let $(x, w, ch, \delta) \leftarrow D_0(k)$ and $(com, ch, resp) \leftarrow (P(x, w), V_0(x, ch))$ if $(x, w) \in W_k$ and $(com, ch, resp) \leftarrow \perp$ else. Output $D_1(com, ch, resp, \delta)$. (-) Let $(x, w, ch, \delta) \leftarrow D_0(k)$ and $(com, ch, resp) \leftarrow Z(x, ch, YES)$ if $(x, w) \in W_k$ and $(com, ch, resp) \leftarrow Z(x, ch, NO)$ else. Output $D_1(com, ch, resp, \delta)$.*

Definition 10. *A relation W is said to have a one-way instance generator $\mathcal{I}$ if for any parameter k algorithm $\mathcal{I}$ returns in probabilistic polynomial time $(x, w) \in W_k$, but such that for any probabilistic polynomial time algorithm, termed as inverter, I, for $(x, w) \in \mathcal{I}(1^k)$ and $I(x) \to w'$ the probability $P((x, w') \in W_k)$ is negligible in k.*

B Schnorr Signatures are secure without observability

In this section we provide a positive result arguing the security of the Schnorr signature scheme. We define below the Discrete Log problem and the Schnorr signature scheme and then show that Schnorr signatures are secure against existential forgery under chosen message attack (EF-CMA). We refer the reader to Appendix A for an introduction to signature schemes and related security models.

Definition 11 (DL Problem)**.** *Let $\mathbb{G} = \langle g \rangle$ be a group of prime order q generated by g. Given $r \in \mathbb{G}$, computing $k \in \mathbb{Z}_q$ such that $r = g^k$ is known as the Discrete Log (DL) problem over the group $\mathbb{G}$.*

A probabilistic algorithm $\mathcal{A}$ is said to be an (ε, τ)-solver for DL if

$$\Pr_{k \xleftarrow{\$} \mathbb{Z}_q} \left[\mathcal{A}(g^k) = k \right] \geq \varepsilon,$$

where the probability is taken over the random tape of $\mathcal{A}$ and random choices of k and $\mathcal{A}$ stops after time at most τ.

The (ε, τ)-discrete log assumption (for group $\mathbb{G}$) says that no (ε, τ)-solver can exist for DL over $\mathbb{G}$. The (asymptotic) *DL-assumption* says that the (ε, τ)-discrete log assumption holds whenever $\tau = \text{poly}(\log q)$ and ε is a non-negligible function of $\log q$.

Definition 12 (Schnorr Signature Scheme)**.** *Let p and q be primes such that $q \mid (p-1)$ Let g be a generator of the cyclic subgroup $\mathbb{G}$ of order q in $\mathbb{Z}_p^*$. Let H be a secure hash function with range $\{1, \ldots, q-1\}$. The Schnorr signature scheme consists of the following three algorithms:*

1. Key Generation: *Choose a random x with $0 < x < q$. x is the private key and $y := g^x$ is the public key.*
2. Signing: *Given the input message m, choose a random $k \mod q$. Let $c := H(m, g^k)$, and $s := k + cx$. Return (c, s) as the signature.*
3. Verification: *Given the message m and the signature pair (c, s), calculate $r = g^s y^{-c}$. Let $c' = H(m, r)$. If $c = c'$ then return true else return $false$.*

We now show that there exists a Non Observing reduction from the Discrete Log assumption to any adversary (making atmost q_h hash queries) that generates existential forgeries under key only attack. To give an intuition behind our idea we start by recalling the original proof of Schnorr in the random oracle model. The key idea in the proof [PS00] is to execute the adversary twice using the same input and random tape. This adversary will make hash queries and the reduction needs to respond to these queries. In both executions, the reduction responds with the same hash responses till some chosen query called the "forking point". Starting from this "forking point" the reduction uses different hash responses in the two executions. With certain probability both executions of the adversary will generate forgery at the "forking" point and this allows the reduction to compute the discrete log. As for the adversary's signature query for some message m, the reduction returns a randomly chosen tuple (s, c) as the signature and then updates the random oracle to return the response c when queried at $(m; g^s y^{-c})$.

Our positive results relies on the observation that in the proof of security of Schnorr signatures, the reduction does not need to actually know the queries made by the adversary to the random oracle. It just needs to be able to achieve a "forking" situation. Furthermore, this effect can be achieved by programming two separate random oracles for the two executions, that respond with the same value up till the "chosen query."

We start by recalling the Splitting Lemma. We refer the reader to [PS00] for details on the proof. Then we provide a formal proof for the security of the Schnorr signature scheme in non observing reduction model.

Lemma 2 (The Splitting Lemma.)**.** *Let $A \subset X \times Y$ such that $Pr[(x, y) \in A] \geq \varepsilon$. For,*

$$B = \left\{ (x, y) \in X \times Y \mid Pr_{y' \in Y}[(x, y') \in A] \geq \frac{\varepsilon}{2} \right\} \tag{1}$$

then $Pr[B] \geq \frac{\varepsilon}{2}$

The philosophy of splitting lemma is that when a subset A is "large" in a product space $X \times Y$, then it has many "large" sections.

Theorem 3. *There exists a Non Observing reduction from the (τ', ε')-secure Discrete-Log problem to any adversary that $(\varepsilon, \tau, q_h, q_s)$-breaks existentially unforgeability (under chosen message attack) of the Schnorr signature scheme such that*

$$\varepsilon \approx \mathsf{poly}\left(\varepsilon', q_h\right) \text{ and } \tau \approx \mathsf{poly}\left(\tau', \varepsilon, q_h, q_s\right)$$

where q_h is the number of random oracle queries and q_s is the number of signature queries that the Schnorr forger makes.

Proof. We closely follow the ideas from [PS00]. Given an existential Schnorr forger oracle machine $\mathcal{A}$, which receives the public key pk and makes up to q_h random oracle queries and up to q_s signature queries, we construct a Non Observing reduction $\mathcal{R} = (\mathcal{R}_1, \mathcal{R}_2)$ that solves the Discrete-Log problem. The reduction will execute the adversary twice. We start by describing $\mathcal{R}_1$, the first part of the reduction. For the two executions of the adversary, $\mathcal{R}_1$ generates machines M_1 and M_2 that it will send to the two instances of the random oracle. Execution of the adversary instantiated with the oracle $\mathcal{O}(M_1)$ when using machine M_1 is denoted by $\mathcal{A}^{\boxed{\mathcal{O}(M_1)}}$. Similarly, adversary instantiated with the oracle $\mathcal{O}(M_2)$ when using machine M_2 is denoted by $\mathcal{A}^{\boxed{\mathcal{O}(M_2)}}$.

We start by describing how $\mathcal{R}_1$ generates the machines M_1 and M_2. $\mathcal{R}_1$ samples a random value $j \in [q_h]$ and q_h random hash responses $\rho_1, \rho_2 \ldots \rho_{q_h}$. It creates a list L_1 with these values $\rho_1, \rho_2 \ldots \rho_{q_h}$. This list is used by machine M_1 to return the query responses and in that order. Similarly, it initializes machine M_2 with list $L_2 = \{\rho_1, \rho_2 \ldots \rho_{j-1}, \rho'_j \ldots \rho'_{q_h}\}$, where $\rho'_j \ldots \rho'_{q_h}$ are freshly chosen.

Reduction $\mathcal{R}_2$ chooses fresh randomness ϖ and executes $\mathcal{A}^{\boxed{\mathcal{O}(M_1)}}$ with input pk and with randomness ϖ. It then executes $\mathcal{A}^{\boxed{\mathcal{O}(M_2)}}$ with the same input pk and the same randomness ϖ. Observe that the reduction $\mathcal{R}_2$ remains oblivious of all the queries that the adversaries make to their oracles. However, since we execute $\mathcal{A}^{\boxed{\mathcal{O}(M_1)}}$ and $\mathcal{A}^{\boxed{\mathcal{O}(M_2)}}$ with the same input and randomness we can assume that in both executions the adversary will make the same first query to the respective oracle. Further, if it is given same responses then its future queries, in the two executions, will also be the same. In our setting, the first $j-1$ responses provided by L_1 and L_2 will be the same and therefore the first j queries made by the adversary will necessarily have to be the same. The queries made by the adversaries after this point might vary. In effect, we have achieved the effect of rewinding even though the reduction $\mathcal{R}_2$ remains oblivious of actual queries made by both $\mathcal{A}^{\boxed{\mathcal{O}(M_1)}}$ and $\mathcal{A}^{\boxed{\mathcal{O}(M_2)}}$.

To answer the signature queries of the adversary, the reduction exploits the fact that it can send dynamic updates to the random oracle. Thus, when the signature oracle is queried at point m, the reduction randomly chooses s and c from $[1, p-1]$, and returns (s, c) as the signature. At the same time, it sends an update to the random oracle to return c when queried at point $(m, g^s \cdot y^{-c})$.

If both executions $\mathcal{A}^{\boxed{\mathcal{O}(M_1)}}$ and $\mathcal{A}^{\boxed{\mathcal{O}(M_2)}}$ return valid signatures on the j^{th} query we can use the two forgeries to solve the Discrete-Log problem. In all other cases our reduction aborts.

Probability Analysis. Let ν be the probability that the adversary returns a forgery on a hash query that it made to its oracle at some point. The probability that the adversary comes up with a forgery involving a hash query for which the random oracle was never queried is $\frac{1}{q-1}$. Thus $\nu = \varepsilon - \frac{1}{q-1} \geq 6\varepsilon/7$. This means that over the random coins of $\varpi, \rho_1 \ldots \rho_{q_h}$ the probability of

success the adversary returning forgery involving some hash query is at least ν. Therefore, there exists a $\beta \in [q_h]$ such that over the random coins of $\varpi, \rho_1 \ldots \rho_{q_h}$ the probability of success of the adversary returning forgery involving the β^{th} hash query is at least $\frac{\nu}{q_h}$. We say that a value $\varpi, \rho_1 \ldots \rho_{\beta-1}$ is "good" if an adversary on input $\varpi, \rho_1 \ldots \rho_{q_h}$ successfully outputs a forgery on the β^{th} hash query with probability at least $\frac{\nu}{2q_h}$, where the probability is taken over the random choices of $\rho_\beta \ldots \rho_{q_h}$. Now using Lemma 2, at least $\frac{\nu}{2q_h}$ fraction of the values $\varpi, \rho_1 \ldots \rho_{\beta-1}$ will be "good." Finally, this yields that the over all success probability of our reduction in solving discrete log is at least $\frac{1}{q_h} \cdot \left(\frac{\nu}{2q_h}\right)^3$, which is non negligible. For a more detailed analysis, refer to [PS00]. □

C RSA based Signature Schemes.

RSA [RSA83] public system consists of a public key (N, e) and a secret key (N, d), where N is the product of two $\lambda/2$ bit primes, and $e, d \in \mathbb{Z}^*_{\phi(N)}$ satisfying $ed \equiv 1 \mod \phi(N)$. The RSA function $f : \mathbb{Z}^*_N \to \mathbb{Z}^*_N$ is defined by $f(x) = x^e \mod N$ and its inverse $f^{-1} : \mathbb{Z}^*_N \to \mathbb{Z}^*_N$ is defined by $f^{-1}(y) = y^d \mod N$ $(x, y \in \mathbb{Z}^*_N)$. Let $(N, e, d) \leftarrow \mathcal{RSA}(1^\lambda)$ be the RSA key generation algorithm.

Definition 13 (RSA Problem)**.** *Given (N, e) and $y \in \mathbb{Z}^*_N$ computing $x^e \mod N$ is known as the RSA inversion problem.*

A probabilistic algorithm $\mathcal{A}$ is said to be an (ε, τ)-solver for the RSA inversion problem if

$$\Pr_{(N,e,d)\leftarrow\mathcal{RSA}(1^\lambda), x\xleftarrow{\$}\mathbb{Z}^*_N}[(\mathcal{A}(N, e, x^e)) = x] \geq \varepsilon,$$

where the probability is taken over the random tape of $\mathcal{A}$, random choices of $\mathcal{RSA}$ and x and $\mathcal{A}$ stops after time at most τ.

The (ε, τ)-RSA inversion assumption says that no (ε, τ)-solver can exist for RSA inversion problem. The (asymptotic) *RSA inversion assumption* says that the (ε, τ)-RSA inversion assumption holds whenever $\tau = \text{poly}(\log q)$ and ε is a non-negligible function of $\log q$.

Definition 14 (Probabilistic Full Domain Hash (PFDH) signatures.)**.** *This signature scheme is parameterized by a length parameter k_0. Let $\mathcal{H} : \{0,1\}^* \to \mathbb{Z}^*_N$ be a secure hash function. The PFDH signature scheme consists of the following three algorithms:*

1. Key Generation: *Let $(N, e, d) \leftarrow \mathcal{RSA}(1^\lambda)$. Further let (N, d) be the private key and (N, e) be the public key.*

2. Signing: *Given the input message m, and the secret key (N, d), choose a random $r \xleftarrow{\$} \{0,1\}^{k_0}$. Compute $s := (H(m, r))^d \mod N$ and return (r, s) as the signature.*

3. Verification: *Given the message m and the signature (r, s), check if $s^e = H(m, r) \mod N$. If it is indeed the case then return true else return $false$.*

When $k_0 = 0$ then the above described signature scheme is in fact deterministic and is referred to as the Full Domain Hash (FDH) signature scheme.

Theorem 4. *Assuming that the RSA inversion problem is (τ', ϵ')-secure, the PFDH signature scheme with parameter k_0 is $(\epsilon, \tau, q_h, q_s)$-existentially unforgeable under chosen message attack (under Non Observing reductions) in the random oracle model such that:-*

$$\epsilon \geq \epsilon' q_h \text{ and } \tau = \tau' + poly(\lambda, q_h, q_s)$$

where q_h is the number of the hash queries and q_s is the number of signature queries that the forger makes.

Proof. Given an existential forger oracle machine $\mathcal{A}$, which receives the public key pk and makes up to q_h queries and upto q_s signature queries, we construct a Non Observing reduction $\mathcal{R} = (\mathcal{R}_1, \mathcal{R}_2)$. $\mathcal{R}_1$ is responsible for generating a machine M that will be used by it to return responses to the random oracle for the adversary. $\mathcal{R}_1$ generates a list L as follows: $\mathcal{R}_1$ samples $q_s + q_h$ elements $\mu_1, \ldots, \mu_{q_s+q_h} \in Z_N^*$. It picks a number i uniformly at random from $[q_s + q_h]$. It then writes the elements $\mu_1^e, \ldots, \mu_{i-1}^e, y\mu_i^e, \mu_{i+1}^e, \ldots, \mu_{q_s+q_h}^e$ on L, where y is the RSA inversion challenge. The machine M is then programmed to return the reponses to its queries from list L and in that order. We next describe the simulation of signature queries. Reduction $\mathcal{R}_2$ then instantiates the adversary with input (N, e) and oracle access to $\mathcal{O}(M)$. On receiving a message m, the reduction picks a bit r of k_0 bits uniformly at random and queries $O(M)$ with (m, r). If the response returned by $O(M)$ is of the form μ_j^e then reduction returns μ_j to the adversary else it aborts.
Outputting an inverse: The reduction returns a forgery (r, s) on message m with randomness r such that it never received a signature on this message. Let the response of $O(\mathcal{R}_1)$ to the query (m, r) be h. Either h is of the form μ_j^e or $y\mu_j^e$ for some $j \in [q_s + q_h]$. If $h = y\mu_j^e$ then $\mathcal{R}_2$ outputs $\frac{s}{\mu_j}$ as RSA inverse of y.
Probability analysis: The reduction succeeds only if the adversary produces a forgery corresponding to the element $y\mu_i^e$. But the adversary can choose this element with probability $\frac{1}{q_s+q_h}$. Hence, the probability that the reduction outputs an RSA inverse of y is $\frac{\epsilon'}{q_s+q_h}$.

□

D Commitment schemes

In this section, we recall the definition of commitment schemes in the random oracle model. A commitment scheme consists of three PPT algorithms: Commit, Decommit and Verify which are as described below. The committer executes the Commit algorithm during the commit phase and it executes the Decommit algorithm during the reveal phase. Consider a random oracle H.

- Commit(m, r): It takes the message m and chooses randomness r to derive a commitment c using the random oracle H.
- Decommit(c): The commitment c is opened by outputting m and r.
- Verify(m, r, c): It verifies whether c is indeed the output of (m, r) using the random oracle H.

A commitment scheme is said to satisfy two main properties: namely, hiding and binding. The computational hiding property says that distributions of the commitments corresponding to two different messages are computationally indistinguishable. The computational binding property says that a probabilistic polynomial time committer can open a commitment to two different values only with negligible probability.