On The (In)security Of Fischlin's Paradigm

PRABHANJAN ANANTH
Microsoft Research India
prabhanjan.va@gmail.com

RAGHAV BHASKAR
Microsoft Research India
rbhaskar@microsoft.com

VIPUL GOYAL
Microsoft Research India
vipul@microsoft.com

VANISHREE RAO
Department of Computer Science, UCLA
vanishri@cs.ucla.edu

Abstract

The Fiat-Shamir paradigm was proposed as a way to remove interaction from 3-round proof of knowledge protocols and derive secure signature schemes. This generic transformation leads to very efficient schemes and has thus grown quite popular. However, this transformation is proven secure only in the random oracle model. In FOCS 2003, Goldwasser and Kalai showed that this transformation is provably insecure in the standard model by presenting a counterexample of a 3-round protocol, the Fiat-Shamir transformation of which is (although provably secure in the random oracle model) insecure in the standard model, thus showing that the random oracle is uninstantiable. In particular, for every hash function that is used to replace the random oracle, the resulting signature scheme is existentially forgeable. This result was shown by relying on the non-black-box techniques of Barak (FOCS 2001).

An alternative to the Fiat-Shamir paradigm was proposed by Fischlin in Crypto 2005. Fischlin's transformation can be applied to any so called 3-round "Fiat-Shamir proof of knowledge'' and can be used to derive non-interactive zero-knowledge proofs of knowledge as well as signature schemes. An attractive property of this transformation is that it provides online extractability (i.e., the extractor works without having to rewind the prover). Fischlin remarks that in comparison to the Fiat-Shamir transformation, his construction tries to "decouple the hash function from the protocol flow" and hence, the counterexample in the work of Goldwaaser and Kalai does not seem to carry over to this setting.

In this work, we show a counterexample to the Fischlin's transformation. In particular, we construct a 3-round Fiat-Shamir proof of knowledge (on which Fischlin's transformation is applicable), and then, present an adversary against both - the soundness of the resulting non-interactive zero-knowledge, as well as the unforegeability of the resulting signature scheme. Our attacks are successful except with negligible probability for any hash function, that is used to instantiate the random oracle, provided that there is an apriori (polynomial) bound on the running time of the hash function. By choosing the right bound, secure instantiation of Fischlin transformation with most practical cryptographic hash functions can be ruled out.

The techniques used in our work are quite unrelated to the ones used in the work of Goldwasser and Kalai. Our primary technique is to bind the protocol flow with the hash function if the code of the hash function is available. We believe that our ideas are of independent interest and maybe applicable in other related settings.

1 Introduction

The Fiat-Shamir paradigm [FS86] was proposed as a way to remove interaction from 3-round proof of knowledge protocols and derive secure signature schemes. It is a generic transformation which leads to quite efficient schemes thus making it quite popular. The security of this transformation was later analyzed under the ideal assumption that the hash function behaves as a random oracle [BR93, PS00]. Thus, the resulting non-interactive proofs and signature scheme are automatically secure in the random oracle model. Several signature schemes (with the best known ones being [Sch91, GQ88, Oka92]) were constructed following the Fiat-Shamir paradigm. It has also been useful in obtaining forward secure schemes and improving the tightness of security reductions [AABN02, MR02].

Random oracle model can be seen as a methodology to design secure cryptographic systems in two steps: first construct and analyze a scheme assuming only oracle access to the random function. Then, find a suitable hash function and instantiate the previous construction with that to get a real world secure cryptographic system.

For the Fiat-Shamir paradigm, Goldwasser and Kalai [GK03] showed that unfortunately the second step of the design methodology cannot be carried out. In particular, they show that the Fiat-Shamir transformation is uninstantiable in the real world: regardless of the choice of the hash function, the resulting signature scheme is insecure. To do this, they first gave a construction of a 3-round identification scheme based on the non-black-box simulation techniques of Barak [Bar01], and then showed that the resulting signature scheme is universally forgeable for any hash function.

An alternative to the Fiat-Shamir paradigm was proposed by Fischlin [Fis05]. Fischlin's transformation can be applied to any so called 3-round "Fiat-Shamir proof of knowledge" and can be used to derive non-interactive zero-knowledge proofs of knowledge as well as signature schemes. An attractive property of this transformation is that it provides online extractability. In other words, just by observing queries a (possibly malicious) prover makes to the random oracle, an extractor is guaranteed to be able to output the witness of the statement being proven (except with negligible probability). This is in contrast to Fiat-Shamir transformation where an extractor must work by rewinding the prover. This property is quite useful while using the resulting non-interactive schemes in larger systems. Fischlin also shows applications of his transformation in constructing group signature schemes [BBS04].

Even though the purpose of Fischlin's transformation is quite similar to that of Fiat-Shamir, the transformation itself is quite different. Fiat-Shamir transformation is applied on a 3 round public coin proof of knowledge protocol and works as follows. Prover sends a commitment to the verifier, the verifier sends back a random challenge, and, the prover finally responds to that challenge. In the transformed non-interactive protocol, the challenge of the verifier is generated by applying the random oracle to the first message (i.e., the commitment) of the prover. The non-interactive scheme is secure in the random oracle model since getting the challenge from the random oracle is similar to getting it from the external verifier; both challenges will be unpredictable to a malicious prover and trying again any polynomial number of times does not help.

Goldwasser and Kalai [GK03] showed insecurity of the Fiat-Shamir paradigm by relying the breakthrough work of Barak [Bar01]. Indeed it seems (in retrospect) that the non-black-box simulation techniques of Barak fits in quite well to show insecurity of the Fiat-Shamir paradigm:

- In the Fiat-Shamir paradigm, the verifier basically just acts as a hash function (i.e., the verifier message is computed by simply evaluating the hash function on the incoming prover message).
- Hence, having oracle access to the hash function is similar to having a black-box access to the verifier, while, having the code of the hash function directly translates to having non-black-box access to the verifier.
- Barak's techniques yield a zero-knowledge protocol which is secure given only black-box access to the verifier (in other words, a scheme which is *resettably-sound* [BGGL01]), but becomes insecure given non-black-box access to the verifier.

We remark that even though the above idea is the starting point towards showing insecurity of the Fiat-Shamir paradigm in [GK03], this by itself is not sufficient. This is because Barak's techniques do not yield a 3-round protocol. Goldwasser and Kalai make use of a number of additional tools and ideas; please refer to [GK03] for more details.

The above high level idea completely breaks down in the context of Fischlin's transformation. As Fischlin remarks [Fis05], "in comparison to the Fiat-Shamir transformation, this construction somewhat decouples the hash function from the protocol flow". In other words, the prover and the verifier messages of the underlying scheme are computed as specified in the underlying scheme; not by making use of the hash function in any way. The hash function is only used to make some final checks on the resulting transcript of interaction. Hence, as observed by Fischlin, the counterexample in [GK03] does not seem to carry over to this setting [Fis05]. This raises the following natural question

"Is there a concrete hash function using which Fischlin transformation can be securely instantiated?"

Our Results. In this work, we give a partial answer to the above question. More specifically, we prove the following.

> "There does not exist any hash function, whose running time is bounded by an apriori fixed polynomial, using which Fischlin transformation can be securely instantiated."

One can interpret the above result in two different ways. Firstly, the bound on the running time will typically be chosen to be a large polynomial in the security parameter. By choosing a large bound, we can rule out the instantiation of Fischlin transformation with widely used hash functions such as SHA-1. Another interpretation is that, given *any* hash function, we can construct a (3-round Fiat-Shamir proof of knowledge) protocol such the the following holds. When Fischlin's transformation is applied on this protocol and instantiated using this hash function, the resulting signature scheme as well the non-interactive zero-knowledge scheme is completely insecure.

We note that the above does not invalidate the original security proof of Fischlin's transformation in any way (which are provided only in the random oracle model). No claims regarding the security of the transformation, once the hash function is instantiated are made in [Fis05]. Fischlin explicitly acknowledges the possibility of such a result in the introduction of his paper [Fis05].

1.1 Technical Overview.

Before we discuss the techniques involved in our work, we briefly sketch Fischlin's transformation in the following.

Fischlin's Transformation. Fischlin [Fis05] proposed an approach to transform any Fiat-Shamir proof of knowledge (as defined in Section 2) to a non-interactive zero knowledge proof of knowledge in the random oracle model. The basic idea of his transformation is given in the following. The transformed prover (of the non-interactive zero-knowledge scheme) roughly works as follows.

- In the underlying Fiat-Shamir proof of knowledge, the challenge space is restricted to be of polynomial size (i.e., the challenge string of the verifier will be of logarithmic length). The protocol begins by executing a super-constant number of parallel copies of the honest prover of the underlying Fiat-Shamir proof of knowledge.

- For each parallel execution i, the prover computes the commitment α_i (i.e., the first message)

- For all possible (polynomially many) challenges β_i starting from 0 the prover performs the following test. It checks whether a fixed number (depending on the security parameter) of least significant bits of $\mathcal{O}\big((\alpha_1, \ldots, \alpha_r), i, \beta_i, \gamma_i\big)$ are all 0. Here γ_i is the prover's response to the challenge β_i and $\mathcal{O}$ denotes the random oracle. If the test passes, the prover fixes β_i to be the challenge for session i.

- Finally, the transcript $(\alpha_i, \beta_i, \gamma_i)$ corresponding to every execution i is output as the proof.

- Verifier accepts the proof only if: (a) the transcript for every execution is accepted by the verifier of the underlying Fiat-Shamir proof of knowledge, and, (b) for all executions, the least significant bits of the random oracle invocation come out to be all 0 (as described above). [1]

[1]This is actually the simplified version of the final construction in [Fis05]. Our results apply to both the variants.

The above construction retains the completeness property: except with negligible probability, for each execution, there will be at least one challenge β_i for which the random oracle outputs 0 (in the relevant bits).

The construction provides soundness for the following reasons. If the statement is false, for each α_i, there is a single challenge β_i for which a satisfying response γ_i can be given. Consider the vector of first messages chosen by the adversary $(\alpha_1, \ldots, \alpha_r)$. Except with negligible probability, there will exist at least one i such that $\mathcal{O}\big((\alpha_1, \ldots, \alpha_r), i, \beta_i, \gamma_i\big)$ does not have its required bits to be all 0. Once that happens, the adversary will have to change α_i (and hence the vector $(\alpha_1, \ldots, \alpha_r)$ changes). Thus, adversary has to essentially restart its effort to produce a false proof (and again it will only be successful with negligible probability).

Thus, it is crucial to have the entire vector $(\alpha_1, \ldots, \alpha_r)$ as part of the input to the random oracle. Even if the adversary fails to obtain the required 0's even in a single execution, it has to start again from scratch. See section 2 for more details.

Our Ideas. Recall that the verifier accepts the proof only if: (a) the transcript for every execution is accepted by the verifier of the underlying Fiat-Shamir proof of knowledge, and, (b) for all executions, the least significant bits of the random oracle invocation come out to be all 0. Normally, these two tests will be "independent and uncorrelated". This is because no random oracle invocations are involved in the first test (the underlying Fiat-Shamir proof of knowledge protocol does not make use of random oracles). However once the code of the hash function is available, it can be used in the underlying protocol *making the two tests correlated.* In fact, in our construction, the two tests will end up being *identical.* This would allow an adversarial prover to succeed (using the description of the hash function). Below we provide a very high level overview of our main idea.

- Observe that in the final transcript being output, for each session i, adversary needs to include an accepting response for just a single challenge β_i (for which the random oracle output has 0 in all required positions). What if somehow magically, adversary exactly has the capability to come up with an accepting response for just this β_i (note that adversary can have the capability of creating an accepting response just for a single challenge)?

- To achieve the following, we take any Fiat-Shamir proof of knowledge and then add another "mode of execution" to it. In this mode, the prover doesn't need the witness to execute the protocol. However the verifier's test is such that for each α_i, there is a single β_i for which verifier accepts. Hence, the protocol still maintains the special soundness property. This new protocol will be the underlying protocol on which Fischlin's transformation will be applied.

- Now we sketch the test the verifier performs in this second mode. The prover will be free to send any hash function to the verifier as part of α_i. Using this hash function, the verifier is instructed to compute the only acceptable β_i for this α_i. If that is the challenge he chose, the verifier is instructed to accept (and reject otherwise).

- The acceptable β_i is the first possible challenge (lexicographically) such that $H\big((\alpha_1, \ldots, \alpha_i, \ldots \alpha_r), i, \beta_i, \gamma_i\big)$ has 0 in all the required positions (where H is the hash function chosen by the prover).

- Now if the hash function H is the same as the random oracle, we have that the second test (by the verifier of the non-interactive proof) is satisfied for free. Hence, by running in the second mode, soundness of the non-interactive scheme can be violated.

There are several problems with this basic idea. To start with, the verifier of the (interactive) Fiat-Shamir protocol that we constructed is unaware of any other sessions. Whether or not it accepts should only be decided by the transcript of this session. However the test $H\big((\alpha_1, \ldots, \alpha_i, \ldots \alpha_r), i, \beta_i, \gamma_i\big)$ requires the knowledge of the first prover messages in all other sessions (we resolve this problem by having a construction in which a first prover message for one session can be used to compute first prover messages for the other sessions). Secondly, note that the random oracle instantiation could be done by any (possibly adversarial) hash function. Since the transcript of interaction in mode 1 and mode 2 may be easily distinguishable, the hash function may never give output having 0 in the relevant places for mode 2 messages (we solve this problem by employing encryption in a deterministic way using shared public randomness). The final construction is given in Section 3.

We note that once we have an adversary to violate soundness of the non-interactive zero-knowledge scheme, it is also easy to design a forger for the resulting signature scheme.

Further Comments. We note that Fischlin's transformation could still be secure in the following sense. For every protocol (on which Fischlin's transformation can be applied), there exists a hash function, whose running time depends upon the parameters of the protocol (in particular upon the running time of the parties in the protocol) such that the following happens. The signature scheme (and non-interactive zero-knowledge scheme) obtained by applying Fischlin's transformation on this protocol, when instantiated with this hash function, is secure in the plain model. However we note that the hash function used to instantiate the scheme has to be dependent on the protocol. In particular, one cannot use a fixed hash function (such as SHA-256) to instantiate the resulting schemes.

Furthermore, Fischlin's construction could still be instantiated if there are no shared public parameters between the prover and the verifier. As with the counterexample for the Fiat-Shamir transformation [GK03], our construction is in the setting where the prover and the verifier share some public parameters[2]. We also sketch an extension of our main construction to the setting where the prover and the verifier have no prior communication/setup in Section 4. In this setting, our results are only valid for the class of hash function whose output is pseudorandom. Indeed, it is natural to think of the random oracle being instantiated by a pseudorandom function. We leave getting an unrestricted result in this setting as an open problem.

Related Works. A number of works have investigated the difference in the settings: where one only has oracle access to a primitive v/s having the full code of the primitive. These lines of research include ones on program obfuscation [BGI$^+$01, GK05], non-black-box simulation techniques [Bar01, Pas04, DGS09], uninstantiabilty of constructions in the random oracle model [CGH04], etc.

2 Fischlin Transformation

In this section, we shall review the Fischlin transformation. We begin by stating the preliminaries. Throughout the paper, we denote the security parameter by k. A function $f : \mathbb{N} \to \mathbb{R}^+ \cup \{0\}$ is said to be negligible (in its input) if, for any constant $c > 0$, there exists a constant, $k_0 \in \mathbb{N}$, such that $f(k) < (1/k)^c$ for any $k > k_0$. We let $f(k) = negl(k)$ denote that $f(k)$ is a negligible function. We say that function is non-negligible if it is not negligible; namely, we say that $f(\cdot)$ is non-negligible in its input if there is constant c > 0 such that for infinitely many k, it holds $f(k) \geq (1/k)^c$. For a probabilistic polynomial time algorithm $\mathcal{A}$, we use the notation $y \leftarrow \mathcal{A}(x)$ to denote $\mathcal{A}$ outputting y on input x. We use the notation $\mathsf{Pr}[E] \gtrsim 1$ to indicate that the probability of the event E is negligibly close to 1. Similarly, the notation $\mathsf{Pr}[E] \gtrsim 0$ is used to indicate that the probability of the event E is neglibly close to 0.

In this paper, we scrutinize the "real-world'' security of protocols that are proven secure in the random oracle model. In the random oracle model, all the parties have access to a purely random function (i.e., a mapping that maps every input to an output that is distributed uniformly random in a range space whose size is dependent on the security parameter. We denote the random oracle by $\mathcal{O}$.

Fischlin transformation converts a 3-round zero-knowledge proof of knowledge, termed as Fiat-Shamir proof of knowledge, to a non-interactive zero-knowledge proof of knowledge proven secure in the random oracle model. In what follows we shall review the formal definitions for both Fiat-Shamir proof of knowledge as well as non-interactive zero-knowledge proof of knowledge as defined in [Fis05].

Definition 1. *A Fiat-Shamir proof of knowledge (with $O(\log(k))$-bit challenges) for a witness relation W is a 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 $(\alpha, \beta, \gamma) \leftarrow (P(x, w), V_0(x))$ it holds* $V_1(x, \alpha, \beta, \gamma) = \mathsf{Accept}$.

[2]Note that none of the parties need to trust the public parameters for their security.

[*Commitment Entropy.*] *For parameter k, for any $(x, w) \in W_k$, the min-entropy of $\alpha \leftarrow P_0(x, w)$ is superlogarithmic in k.*

[*Public Coin.*] *For any k, any $(x, w) \in W_k$, any $\alpha \leftarrow P_0(x, w)$, the challenge $\beta \leftarrow V_0(x, \alpha)$ is uniform on $\{0, 1\}^{l(k)}$.*

[*Unique responses.*] *For any probabilistic polynomial time algorithm A, for parameter k and $(x, \alpha, \beta, \gamma, \gamma') \leftarrow A(k)$, we have, as a function of k,*

$$Pr[V_1(x, \alpha, \beta, \gamma) = V_1(x, \alpha, \beta, \gamma') = \mathsf{Accept} \wedge \gamma \neq \gamma'] \approx 0$$

[*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 $V_1(x, \alpha, \beta, \gamma) = V_1(x, \alpha, \beta', \gamma') = \mathsf{Accept}$ and $\beta \neq \beta'$, for $w' \leftarrow K(x, \alpha, \beta, \gamma, \beta', \gamma')$ 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 $(x, w, \delta) \leftarrow D_0(k)$ and $(\alpha, \beta, \gamma) \leftarrow (P(x, w), V_0(x))$ if $(x, w) \in W_k$ and $(\alpha, \beta, \gamma) \leftarrow \perp$ otherwise. Output $D_1(\alpha, \beta, \gamma, \delta)$.*
- *Let $(x, w, \delta) \leftarrow D_0(k)$ and $(\alpha, \beta, \gamma) \leftarrow Z(x, \mathsf{YES})$ if $(x, w) \in W_k$ and $(\alpha, \beta, \gamma) \leftarrow Z(x, \mathsf{NO})$. Output $D_1(\alpha, \beta, \gamma, \delta)$.*

Definition 2. *A pair (P, V) of probabilistic polynomial time algorithms is called a non-interactive zero-knowledge proof of knowledge for relation W with an online extractor (in the random oracle model) if the following holds.*
[*Completeness*] *For any oracle $\mathcal{O}$, any $(x, w) \in W_k$ and any $\pi \leftarrow P^{\mathcal{O}}(x, w)$, we have $Pr[V^{\mathcal{O}}(x, \pi) = \mathsf{Accept}] \gtrsim 1$.*
[*Zero-knowledge*] *There exist a pair of probabilistic polynomial time algorithms $Z = (Z_0, Z_1)$, 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 $\mathcal{O}$ be a random oracle, $(x, w, \delta) \leftarrow D_0^{\mathcal{O}}(k)$, and $\pi \leftarrow P^{\mathcal{O}}(x, w)$ where $(x, w) \in W_k$. Output $D_1^{\mathcal{O}}(\pi, \delta)$.*
- *Let $(\mathcal{O}_0, \sigma) \leftarrow Z_0(k)$, $(x, w, \delta) \leftarrow D_0^{\mathcal{O}_0}(k)$ and $(\mathcal{O}_1, \pi) \leftarrow Z_1(\sigma, x)$. Output $D_1^{\mathcal{O}_1}(\pi, \delta)$.*

[*Online Extractor.*] *There exists a probabilistic polynomial time algorithm K, the online extractor, 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 $\mathcal{O}$ and $\mathcal{O}$'s answers. Let $w \leftarrow K(x, \pi, Q_{\mathcal{O}}(A))$. Then, as a function of k,*

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

We are now ready to give a formal description of the Fischlin transformation.

Fischlin Transformation. Let (P_{FS}, V_{FS}) be an interactive Fiat-Shamir proof of knowledge with challenges of $l = l(k) = O(\log(k))$ bits for a relation W. Define the parameters b, r, S, t as the number of test bits, repetitions, maximum sum and trial bits such that $br = \omega(\log(k))$, $t - b = \omega(\log(k))$,$b, r, t = O(\log(k))$, $S = O(r)$ and $b \leq t \leq l$. 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 $P^{\mathcal{O}}$ on input (x, w), first runs the prover $P_{FS}(x, w)$ in r independent repetitions to obtain r commitments $(\alpha_1, \ldots, \alpha_r)$. Then $P^{\mathcal{O}}$ does the following, either sequentially or in parallel for each repetition i. For each $\beta_i = 0, 1, \ldots, 2^t - 1$ it lets P_{FS} compute the final responses γ_i by rewinding, until it finds the first one such that $\mathcal{O}(x, (\alpha_1, \ldots, \alpha_r), i, \beta_i, \gamma_i) = 0^b$, if no such tuple is found then $P^{\mathcal{O}}$ picks the first one for which the hash value is minimal among all 2^t hash values. The prover finally outputs $\pi = (\alpha_i, \beta_i, \gamma_i)_{i=1,\ldots,r}$.

Verifier: The verifier $V^{\mathcal{O}}$ on input x and $\pi = (\alpha_i, \beta_i, \gamma_i)_{i=1,\ldots,r}$ accepts if and only if $V_{1,FS}(x, \alpha_i, \beta_i, \gamma_i) = \mathsf{Accept}$ (*first test*) for each $i \in [r]$ and if $\sum_{i=1}^{r} \mathcal{O}(x, (\alpha_1, \ldots, \alpha_r), i, \beta_i, \gamma_i) \leq S$ (*second test*).

We shall now briefly review the proof of security (in the random oracle model) of the Fischlin transformation. We shall begin by arguing completeness. We need to show that the (honest) prover fails to convince the verifier only with negligible probability. From the completeness property of the underlying Fiat-Shamir proof of knowledge, the proof produced by the honest prover passes the first test with probability 1. It can be shown that the probability that the proof passes the second test is negligibly close to 1 by the following two basic arguments:

- Probability that at least in one of the r repititions the smallest hash value that the prover obtains is $> S$ is negligible. Hence, with all but negligible probability the sum of the hash values $\leq rS$.

- By a basic combinatorial argument, the sum of the hash values $> S$ and $\leq rS$ only with negligible probability. Hence, the sum is $\leq S$ with all but negligible probability.

From this, it can be seen that the honest prover passes the second test with probability negligibly close to 1.

We now prove that the protocol satisfies online extractability (which in turn implies soundness). Consider an adversarial prover who produces a proof given just the input instance. The claim is that except with negligible probability the proof is rejected by the verifier. Consider a particular commitment tuple $(\alpha_1, \ldots, \alpha_r)$. Observe that in the queries made by the adversarial prover to the random oracle there cannot be two accepting transcripts of the form $((\alpha_1, \ldots, \alpha_r), i, \beta, \gamma)$ and $((\alpha_1, \ldots, \alpha_r), i, \beta', \gamma')$ because then the special soundness property of the underlying Fiat-Shamir proof of knowledge would imply that the adversary has the witness for the input instance. Hence, corresponding to each repetition i and commitment tuple $(\alpha_1, \ldots, \alpha_r)$, the adversary can query the random oracle for at most one challenge β_i. Let s_i be the value output by the random oracle for this particular β_i. With negligible probability, the summation of s_i over all the repetitions is at most S. This is because, there are only polynomially many (in the security parameter) possible tuples $(s_1^{'}, \ldots, s_r^{'})$ whose summation is at most S and since s_i is picked uniformly at random, the probability that the sum of s_i is at most S is $poly(k).negl(k)$ which is also negligible in k. This means that the adversary has negligible probability of succeeding for a given $(\alpha_1, \ldots, \alpha_r)$. Since, the adversary can try only polynomially many such commitment tuples, it is only with negligible probability that it can produce an accepting proof.

Fischlin showed that the non-interactive zero-knowledge proof of knowledge obtained from his construction can be used to construct signature schemes which are secure. The signature scheme derived from his construction was shown to be existentially unforgeable against adaptive chosen message attacks in the random oracle model.

In this paper, we give a construction of 3-round Fiat-Shamir proof of knowledge protocol such that the resulting protocol obtained after applying Fischlin transformation does not satisfy soundness in the real-world. And hence, note that the security of the signature scheme built over this non-interactive protocol (which is the output of Fischlin transformation) also breaks down.

3 Our construction

Our goal is to construct a Fiat-Shamir proof of knowledge (P^*, V^*) for some witness relation such that the non-interactive protocol obtained after applying the Fischlin transformation to it is insecure when the random oracle is instantiated with a hash function ensemble containing functions whose running time is apriori bounded by some polynomial. We now give the intuition about the construction of proof of knowledge (P^*, V^*). We first consider a Fiat-Shamir proof of knowledge (P, V) from which we build (P^*, V^*). The verifier V^* is basically V with its verdict function extended so as to also accept in the case when the challenge equals the output of some pre-determined function, denoted by least (defined later). The function least takes the first message as input and returns a challenge. As the verifier chooses the challenge uniformly at random, it is only with low probability that the challenge equals the output of least of the first message. This, together with the fact that (P, V) is sound, implies that (P^*, V^*) satisfies soundness property. However, in the non-interactive protocol obtained by applying Fischlin transformation to (P^*, V^*), denoted by $(P^{\mathcal{O}}, V^{\mathcal{O}})$, the prover himself is supposed to compute the challenge messages. However, the probability that any adversarial prover succeeds in producing an accepting proof is negligible from the security proof of the Fischlin transformation. But when the random oracle $\mathcal{O}$ in $(P^{\mathcal{O}}, V^{\mathcal{O}})$ is instantiated by a hash function h drawn from a hash function ensemble to get (P_h, V_h) we can construct an adversary to violate the soundness of (P_h, V_h) as follows. The adversarial prover first makes the least function dependent on machine M, which implements the instantiated hash function h, in such a way that the first test and the second test become identical. The adversary then produces an accepting proof by setting the challenges in each repetition to be the output of the least function of the first message and thus succeeds in passing the first test and hence the second test too. We now give details of the construction below:

Let (P, V) be a Fiat-Shamir proof of knowledge for a relation W. We use two main tools for our construction, namely, a CPA symmetric encryption scheme $E = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ and a pseudorandom function family $\mathcal{F}$. Before we describe the protocol, we make the following assumption: In all the executions of the protocol, the prover and the verifier have access to a string which is generated by a Setup algorithm. The Setup algorithm takes 1^k as input and executes KeyGen to obtain SK. It further chooses a key K uniformly at random to choose a function from the pseudorandom function family $\mathcal{F}$. Finally, $(\mathsf{SK}, \mathsf{K})$ is output by Setup.[3] The output of the Setup is used in the following way. Each time the prover or the verifier needs to encrypt a message m, they proceed as follows. Compute $f_{\mathsf{K}}(m)$ (where f_{K} is the function in $\mathcal{F}$ corresponding to key K) to obtain r. To encrypt m, execute the algorithm Enc with inputs m, SK and r. Unless explicitly mentioned, by $\mathsf{Enc}(m)$ we mean that m is encrypted using key SK and randomness $f_{\mathsf{K}}(m)$. This means that $\mathsf{Enc}(m)$ gives the same ciphertext every time it is executed. If we intend to use a different randomness, we use the notation $\mathsf{Enc}(m : R)$ to mean that m is encrypted using the randomness R. When the prover or the verifier wants to decrypt a message m they execute Dec with input m and key SK. Jumping ahead, we need to encrypt messages because the hash function used to instantiate the random oracle might have the code of the verifier V embedded in it. In this case the hash function may output values depending whether the input transcripts are accepting or rejecting. To make our security proof go through we need to make sure the hash function does not have the capability to distinguish the transcripts. We can ensure this by encrypting the messages of the prover (The Setup algorithm is considered to be a part of the interaction between a specific prover and a verifier; the hash function used to instantiate the random oracle is independent of the output of the Setup algorithm). We now proceed to describe the protocol.

As discussed before, we will first consider a Fiat-Shamir proof of knowledge (P, V). We assume that the prover P in (P, V) can be decomposed into P_0 and P_1, where $P_0(x, w)$ outputs the commitment α and $P_1(x, w, \alpha, \beta)$ outputs γ. Similarly, the verifier V can be decomposed into V_0 and V_1 such that V_0 interacts with P (by outputting β on input some (x, α)) to produce the transcript (α, β, γ) and then V accepts if and only if $V_1(\alpha, \beta, \gamma)$ accepts. We use the symbols r, b, t as defined in the Fischlin construction (c.f. Section 2). We denote the least significant l bits of $M(\boldsymbol{y})$ by $M(\boldsymbol{y})^{(l)}$.

Our protocol is parameterized by a polynomial p_{hash}. We are now ready to describe the protocol (P^*, V^*) for the relation W.

Protocol (P^*, V^*):

1. P^*: Run P_0 on (x, w) to obtain α. Define $\alpha^* = \mathsf{Enc}((\alpha, i, \mathsf{bit}, \mathsf{M}))$, where each of $i, \mathsf{bit}, \mathsf{M}$ is set to 0, with their lengths being $\log(r)$ bits, 1 bit, and $|x|$ bits, respectively[4]. Send α^* to V^*.

[3]We note that neither the prover nor the verifier needs to place any trust in the setup algorithm for their security. The reason to have $(\mathsf{SK}, \mathsf{K})$ as the public parameters (as opposed to the part of protocol messages) will become clear later on.

[4]Hereafter, unless specified otherwise, we maintain that the lengths of bit, i are specified above, and in every instance where we set M to 0, it is of length $|x|$.

[Note: Looking ahead, in the protocol obtained by first applying Fischlin transformation to (P^*, V^*) and then instantiating it with the hash function, the adversary will set i to be the repetition number, bit *to be 1 and M to be the hash function instantiating the random oracle.]*

2. V^*: Execute $\mathsf{Dec}(\alpha^*)$ to obtain α_1 which is then parsed as $(\alpha, i, \mathsf{bit}, M)$. Run V_0 on input (x, α) to obtain β. Send β to P^*.

3. P^*: Run P_1 on input (x, w, α, β) to obtain γ. Send $\gamma^* = \mathsf{Enc}(\gamma)$ to V^*.

V^* then decides to accept or reject the transcript $(x, \alpha^*, \beta, \gamma^*)$ by executing the following.

i. Let $\alpha_1 \leftarrow \mathsf{Dec}(\alpha^*)$ and $\gamma \leftarrow \mathsf{Dec}(\gamma^*)$.

ii. Parse α_1 to be $(\alpha, i, \mathsf{bit}, \mathsf{M})$.

iii. If $\mathsf{bit} = 0$ then Accept if and only if $V_0(x, \alpha, \beta, \gamma)$ accepts and $\gamma^* = \mathsf{Enc}(\gamma)$.
[Note: Recall that $\mathsf{Enc}(m)$ *is the encryption of m using the randomness $f_{\mathsf{K}}(m)$. Hence, the check $\gamma^* =$* $\mathsf{Enc}(\gamma)$ *ensures that γ^* is indeed the encryption of γ using the randomness $f_{\mathsf{K}}(\gamma)$. Looking ahead, this will be helpful to make the protocol satisfy the unique responses property.]*

iv. Else, do the following. If M is not a valid Turing machine then Reject. Otherwise, Accept if both the following conditions hold:

- $\beta = \mathsf{least}(x, \alpha, i, \mathsf{M})$.
- $\gamma^* = \mathsf{Enc}((i, \beta))$.

where the least procedure is defined below.

$\mathsf{least}(x, \alpha, i, \mathsf{M})$:

```
1. $\mathsf{min} \leftarrow 2^b + 1$
2. $\beta \leftarrow null$
3. For $j = 0$ to $2^t - 1$:
4.     $\boldsymbol{y} \leftarrow \Big(x, \big(\mathsf{Enc}((\alpha, 1, 1, \mathsf{M})), \ldots, \mathsf{Enc}((\alpha, r, 1, \mathsf{M})), i, j, \mathsf{Enc}((i, j))\big)\Big)$
5.     Execute $\mathsf{M}(\boldsymbol{y})$ upto $p_{\mathsf{hash}}(|\boldsymbol{y}|)$ steps
6.     If $\mathsf{M}(\boldsymbol{y})$ terminates within $p_{\mathsf{hash}}(|\boldsymbol{y}|)$ steps:
7.         $\mathsf{hash} \leftarrow \mathsf{M}(\boldsymbol{y})^{(b)}$
8.         If $\mathsf{min} > \mathsf{hash}$:
9.             $\mathsf{min} \leftarrow \mathsf{hash}$
10.            $\beta \leftarrow j$
11. Return $\beta$.
```

The least algorithm does the following. It checks for what values of j from 0 to $2^t - 1$, the last b bits of $\mathsf{M}\Big(x, (\mathsf{Enc}((\alpha, 1, 1, \mathsf{M})), \ldots, \mathsf{Enc}((\alpha, r, 1, \mathsf{M})), i, j, \mathsf{Enc}((i, j))\Big)$ takes the minimum value among all possible 2^t values provided M terminates within $p_{\mathsf{hash}}\Big(\Big|x, \mathsf{Enc}((\alpha, 1, 1, \mathsf{M})), \ldots, \mathsf{Enc}((\alpha, r, 1, \mathsf{M})), i, j, \mathsf{Enc}((i, j))\Big|\Big)$ steps. If there are many values of j for which the hash function maps to the minimum then it picks the one which is the smallest. Observe that in the Fischlin construction, the non-interactive prover $P^{\mathcal{O}}$ would implicitly run the least algorithm as follows. It rewinds the prover in the Fiat-Shamir proof of knowledge until it finds the smallest β such that the hash value when applied on the entire transcript maps to a minimum. This observation was the main intuition behind our definition of the least algorithm.

We show that (P^*, V^*) satisfies all the properties of Fiat-Shamir proof of knowledge.

Lemma 1. *(P^*, V^*) is a Fiat-Shamir proof of knowledge for the relation W.*

Proof. Before we show that (P^*, V^*) satisfies all the properties of Fiat-Shamir proof of knowledge, we first make the observation that both the prover P^* and the verifier V^* run in polynomial time.

[Completeness] For any $(\alpha^*, \beta, \gamma^*)$ resulting from the interaction between P^* and V^* on input x, we have that $\alpha^* = \mathsf{Enc}((\alpha, i, \mathsf{bit}, \mathsf{M}))$ and $\gamma^* = \mathsf{Enc}(\gamma)$ with each of $i, \mathsf{bit}, \mathsf{M}$ being 0. V^* accepts $(\alpha^*, \beta, \gamma^*)$ only if $V(x, \alpha, \beta, \gamma)$ accepts. Thus, the completeness of (P^*, V^*) follows from the completeness property of (P, V).

[Special Soundness] Let K be a knowledge extractor for (P, V). We show that (P^*, V^*) satisfies special soundness by constructing a knowledge extractor K^* that uses K as follows. For any $(x, w) \in W$, on input $(x, \alpha^*, \beta_1, \gamma_1^*, \beta_2, \gamma_2^*)$ such that $V^*(x, \alpha^*, \beta_1, \gamma_1^*) = V^*(x, \alpha^*, \beta_2, \gamma_2^*) = \mathsf{Accept}$ and $\beta_1 \neq \beta_2$, K^* does the following. It decrypts α^* to obtain $(\alpha, i, \mathsf{bit}, \mathsf{M})$. Similarly it decrypts γ_1^* and γ_2^* to obtain γ_1 and γ_2 respectively. Then, K^* outputs whatever $K(x, \alpha, \beta_1, \gamma_1, \beta_2, \gamma_2)$ outputs. To see that K^* is indeed a knowledge extractor for (P^*, V^*), consider the following two cases.

- $\mathsf{bit} = 0$: Here, $V^*(x, \alpha^*, \beta_1, \gamma_1^*) = V^*(x, \alpha^*, \beta_2, \gamma_2^*) = \mathsf{Accept}$ only if $V(x, \alpha, \beta_1, \gamma_1) = V(x, \alpha, \beta_2, \gamma_2) = \mathsf{Accept}$. Hence the special soundness property is satisfied because the special soundness of (P, V) ensures that for such an input $(x, \alpha, \beta_1, \gamma_1, \beta_2, \gamma_2)$, K outputs w' such that $(x, w') \in W$.
- $\mathsf{bit} = 1$: In this case, V^* accepts both inputs $(\alpha^*, \beta_1, \gamma_1^*)$ and $(\alpha^*, \beta_2, \gamma_2^*)$ only if both β_1 and β_2 are equal to $\mathsf{least}(\alpha, i, 1, \mathsf{M})$. This contradicts the assumption that $\beta_1 \neq \beta_2$.

[Commitment entropy] The first message of P^* contains α which has the same distribution as the first message of P and hence the commitment entropy property is satisfied.

[Public coin] This follows from the description of V^*.

[Unique responses] For any probabilistic polynomial-time algorithm $\mathcal{A}$ and $(x, \alpha^*, \beta, \gamma_1^*, \gamma_2^*) \leftarrow \mathcal{A}(1^k)$, where $\alpha^* = \mathsf{Enc}((\alpha, i, \mathsf{bit}, \mathsf{M}))$, $\mathsf{Enc}(\gamma_1) = \gamma_1^*$ and $\mathsf{Enc}(\gamma_2) = \gamma_2^*$. We claim that the following is negligible in k:

$$\mathsf{Pr}[V^*(\alpha^*, \beta, \gamma_1^*) = V^*(\alpha^*, \beta, \gamma_2^*) = \mathsf{Accept} \;\&\; \gamma_1^* \neq \gamma_2^*].$$

To prove this claim, consider the following cases.

- $\mathsf{bit} = 0$: Observe that in this case, $V^*(x, \alpha^*, \beta, \gamma_1^*) = \mathsf{Accept}$ only if $V(x, \alpha, \beta, \gamma_1) = \mathsf{Accept}$, and $V^*(x, \alpha^*, \beta, \gamma_2^*) = \mathsf{Accept}$ only if $V(x, \alpha, \beta, \gamma_2) = \mathsf{Accept}$. Also, γ_1 is equal to γ_2 only if γ_1^* is equal to γ_2^*. This is because of the following reason. γ_1^* is the encryption of γ_1 using the randomness $f_{\mathsf{K}}(\gamma_1)$ and γ_2^* is the encryption of γ_2 using the randomness $f_{\mathsf{K}}(\gamma_2)$. And hence if γ_1 were to be equal to γ_2 then this would imply that γ_1^* equals γ_2^*. Combining the above arguments we have the following. Conditioned on $\mathsf{bit} = 0$,

  $$\mathsf{Pr}[V^*(\alpha^*, \beta, \gamma_1^*) = V^*(\alpha^*, \beta, \gamma_2^*) = \mathsf{Accept} \;\&\; \gamma_1^* \neq \gamma_2^*] = \mathsf{Pr}[V(\alpha, \beta, \gamma_1) = V(\alpha, \beta, \gamma_2) = \mathsf{Accept} \;\&\; \gamma_1 \neq \gamma_2]$$

  Since (P, V) satisfies the unique responses property, $\mathsf{Pr}[V(\alpha, \beta, \gamma_1) = V(\alpha, \beta, \gamma_2) = \mathsf{Accept} \;\&\; \gamma_1 \neq \gamma_2]$ is negligible and hence the claim follows.
- $\mathsf{bit} = 1$: Note that in this case, one of the conditions that needs to be satisfied for V^* to accept $(\alpha^*, \beta, \gamma_1^*)$ (and $(\alpha^*, \beta, \gamma_2^*)$) is that $\gamma_1^* = \mathsf{Enc}((i, \beta))$ (resp., $\gamma_2^* = \mathsf{Enc}((i, \beta))$). This implies that $\gamma_1^* = \gamma_2^*$ and hence the following holds conditioned on $\mathsf{bit} = 1$.

  $$\mathsf{Pr}[V^*(\alpha^*, \beta, \gamma_1^*) = V^*(\alpha^*, \beta, \gamma_2^*) = \mathsf{Accept} \;\&\; \gamma_1^* \neq \gamma_2^*] \;=\; 0$$

[Honest Verifier Zero-knowledge] To prove that (P^*, V^*) satisfies honest verifer zero-knowledge property, we construct a zero-knowledge simulator Z^* for (P^*, V^*) as follows. Let Z be a zero-knowledge simulator for the protocol (P, V). On input $(x, \mathsf{membership})$, where $\mathsf{membership} \in \{\mathsf{yes}, \mathsf{no}\}$, Z^* runs $Z(x, \mathsf{membership})$ to obtain (α, β, γ). Z^* outputs $(\mathsf{Enc}((\alpha, i, \mathsf{bit}, \mathsf{M})), \beta, \mathsf{Enc}(\gamma))$, where i, bit and M are set to 0.

To prove that Z^* is a zero-knowledge simulator for (P^*, V^*), we first assume for contradiction that there exists a distinguisher $D^* = (D_0^*, D_1^*)$ such that the statistical distance between following two distributions is non-negligible.

- $\mathsf{Dist}^*_{\mathsf{real}}$: Let $(x, w, \mathsf{state}) \leftarrow D_0^*(1^k)$ and $(\alpha^*, \beta, \gamma^*) \leftarrow (P^*(w), V^*)(x)$ if $(x, w) \in W$, and $(\alpha^*, \beta, \gamma^*) \leftarrow \perp$ otherwise. Output $D_1^*(\alpha^*, \beta, \gamma^*, \mathsf{state})$.
- $\mathsf{Dist}^*_{\mathsf{sim}}$: Let $(x, w, \mathsf{state}) \leftarrow D_0^*(1^k)$ and $(\alpha^*, \beta, \gamma^*) \leftarrow Z^*(x, \mathsf{yes})$ if $(x, w) \in W$, and $(\alpha^*, \beta, \gamma^*) \leftarrow Z^*(x, \mathsf{no})$ otherwise. Output $D_1^*(\alpha^*, \beta, \gamma^*, \mathsf{state})$.

Then, we construct a distinguisher $D = (D_0, D_1)$ that contradicts the honest verifier zero-knowledge property of (P, V) as follows. $D_0(1^k)$ runs $D_0^*(1^k)$ to obtain (x, w, state) and outputs the same. Once D_1 receives (α, β, γ), if $(\alpha, \beta, \gamma) \neq \perp$ then it outputs $D_1^*(\mathsf{Enc}((\alpha, i, \mathsf{bit}, \mathsf{M})), \beta, \mathsf{Enc}(\gamma), \mathsf{state})$, where i, bit and M are set to 0, else it outputs $D_1^*(\perp, \mathsf{state})$. Now consider the following distributions.

- $\mathsf{Dist}_{\mathsf{real}}$: Let $(x, w, \mathsf{state}) \leftarrow D_0(1^k)$ and $(\alpha, \beta, \gamma) \leftarrow (P(w), V)(x)$ if $(x, w) \in W$, and $(\alpha, \beta, \gamma) \leftarrow \perp$ otherwise. Output $D_1(\alpha, \beta, \gamma, \mathsf{state})$.
- $\mathsf{Dist}_{\mathsf{sim}}$: Let $(x, w, \mathsf{state}) \leftarrow D_0(1^k)$ and $(\alpha, \beta, \gamma) \leftarrow Z(x, \mathsf{yes})$ if $(x, w) \in W$, and $(\alpha, \beta, \gamma) \leftarrow Z(x, \mathsf{no})$ otherwise. Output $D_1(\alpha, \beta, \gamma, \mathsf{state})$.

Now, from the way Z^* and D are constructed, the distribution $\mathsf{Dist}_{\mathsf{real}}$ is the same as $\mathsf{Dist}^*_{\mathsf{real}}$. Similarly, the distribution $\mathsf{Dist}_{\mathsf{sim}}$ is the same as $\mathsf{Dist}^*_{\mathsf{sim}}$. This implies that the statistical distance between $\mathsf{Dist}_{\mathsf{real}}$ and $\mathsf{Dist}_{\mathsf{sim}}$ is also non-negligible, a contradiction. □

3.1 On the Insecurity of (P_h, V_h)

The non-interactive zero-knowledge proof of knowledge $(P^{*\mathcal{O}}, V^{*\mathcal{O}})$ obtained by applying the Fischlin transformation to (P^*, V^*) is sound in the random oracle model. This follows from the fact that (P^*, V^*) is Fiat-Shamir proof of knowledge (Theorem 3) and any protocol obtained by applying Fischlin transformation to a Fiat-Shamir proof of knowledge is secure in the random oracle model. In this section, we show that when the random oracle is instantiated by a hash function h, whose worst case running time is at most the polynomial p_{hash} in the size of its inputs, the protocol (P_h, V_h), which is obtained by instantiating the random oracle $\mathcal{O}$ in $(P^{*\mathcal{O}}, V^{*\mathcal{O}})$, is not sound. Typically, p_{hash} is chosen to be a polynomial of degree c, for a large constant c. The following theorem rules out the secure instantiation of the Fischlin construction with most of the practical hash functions.

Theorem 1. *Let (P^*, V^*) be the 3-round Fiat-Shamir proof of knowledge, for a witness relation W and the corresponding language $L = \{x : (x, w) \in W\}$, as described above. Let $(P^{*\mathcal{O}}, V^{*\mathcal{O}})$ be the non-interactive zero-knowledge proof of knowledge obtained by applying the Fischlin transformation to (P^*, V^*). Then, for any hash function h, that is used to instantiate the random oracle $\mathcal{O}$, and whose running time is at most $p_{\mathsf{hash}}(|\boldsymbol{y}|)$ for any input $\boldsymbol{y} \in \{\mathbf{0}, \mathbf{1}\}^*$, the resulting protocol (P_h, V_h) is not sound. In other words, there exists an adversary $\mathcal{A}$ such that $\mathsf{Pr}[(x, \pi) \leftarrow A(1^k) \ : \ V^h(x, \pi) = 1$ and $x \notin L]$ is non-negligible.*

Proof. Let h be any hash function whose running time is at most $p_{\mathsf{hash}}(|\boldsymbol{y}|)$ for any input $\boldsymbol{y} \in \{0, 1\}^*$. Let M be a Turing machine which computes the hash function h. Now to prove that (P_h, V_h) does not satisfy soundness, we construct a PPT adversary $\mathcal{A}$ which on input M outputs (x, π) for $x \notin L$ such that $V_h(x, \pi) = \mathsf{Accept}$ with probability negligibly close to 1.
The adversary $\mathcal{A}$ is described as follows.

- On input M, choose a string $x \notin L$ such that $|x| = |\mathsf{M}|$.

- Pick α uniformly at random from the commitment space. Then compute β_i as $\beta_i = \mathsf{least}(\alpha, i, 1, \mathsf{M})$ for all $i \in [r]$.

- Check whether $\sum_{i=1}^{r} \mathsf{M}\big(x, (\mathsf{Enc}(\alpha, 1, 1, \mathsf{M})), \ldots, (\mathsf{Enc}(\alpha, r, 1, \mathsf{M})), i, \beta_i, \mathsf{Enc}((i, \beta_i))\big)^{(b)} \leq S$. If so, then output the proof π as $\pi = (\mathsf{Enc}(\alpha, i, 1, \mathsf{M}), \beta_i, \mathsf{Enc}(i, \beta_i))_{1 \leq i \leq r}$; else abort.

We first give the intuition as to why the adversary succeeds. To show that the adversary succeeds, we need to show that the adversary passes both the tests of V_h with non-negligible probability. It can be observed that in the case of the adversary $\mathcal{A}$, by its construction, the first test and the second test of V_h becomes identical. In other words, the adversary succeeds if and only if the sum of hash values evaluated on the proof produced by the adversary is at most S (second test). Unlike the case of the random oracle model, it is tricky to bound the probability [5] that the sum of the hash values is at most S when we are using a real world hash function. To show that the probability of this event is non-negligible, we first observe that by the completeness property, the honest prover passes the second test with non-negligible probability. In other words, with non-negligible probability, the sum of the outputs of the hash function on the proof produced by the honest prover is at most S. If we show that the hash function cannot distinguish a proof produced by an honest prover from the proof produced by the adversary $\mathcal{A}$ then this would imply that the sum of hash function outputs on the proof produced by $\mathcal{A}$ is at most S with non-negligible probability. This would mean that the adversary succeeds the second test, and hence the first test, with non-negligible probability which will in turn prove the theorem. We now describe the technical details of the proof.

We show that the adversary succeeds with non-negligible probability in two steps. Firstly, in Lemma 2 below, we show that the set of first (commitments) and last messages (responses) of the adversary $\mathcal{A}$ is computationally indistinguishable from that of an honest prover. This is to ensure that the hash function cannot distinguish whether its input corresponds to a proof produced by an honest prover or a proof produced by $\mathcal{A}$. Then, using Lemma 2, we show in Lemma 3 that the adversary aborts with negligible probability. Then observing that the adversary does not abort if and only if the adversary produces an accepting proof we conclude that the adversary succeeds with non-negligible probability.

Lemma 2. *The following two distributions are computationally indistinguishable.*

- $\mathsf{D}_x^{(0)} = \left\{ \Big(\mathsf{Enc}(\alpha_1, 0^{|r|}, 0^{|\mathsf{bit}|}, 0^{|x|}), \ldots, \mathsf{Enc}(\alpha_r, 0^{|r|}, 0^{|\mathsf{bit}|}, 0^{|x|}), i, \beta, \mathsf{Enc}\big(P_1(x, w, \alpha_i, \beta)\big)\Big)_{1 \leq i \leq r,\, 0 \leq \beta \leq 2^t - 1} \right\}$ [6] *, defined over the random coins used to generate* $\mathsf{SK}, \mathsf{K}, \alpha_1, \ldots, \alpha_r$.

- $\mathsf{D}_x^{(1)} = \left\{ \Big(\mathsf{Enc}(\alpha, 1, 1, \mathsf{M}), \ldots, \mathsf{Enc}\big(\alpha, r, 1, \mathsf{M}), i, \beta, \mathsf{Enc}((i, \beta))\big)\Big)_{1 \leq i \leq r,\, 0 \leq \beta \leq 2^t - 1} \right\}$*, defined over the random coins used to generate* $\mathsf{SK}, \mathsf{K}, \alpha$.

Note that we assume here the distinguisher does not get access to either the PRF key K *or the decryption key* SK*, as the distinguishers we are interested in are the hash functions.*

Proof. To prove the above lemma, we first prove the following claim and then show that the lemma follows as a consequence of the claim. Before that, we introduce the following notation. Let T be a tuple of messages $(m_1, \ldots, m_r)$. By $\mathsf{Enc}(T)$ we mean the tuple $(\mathsf{Enc}(m_1), \ldots, \mathsf{Enc}(m_r))$.

Claim 1. Consider the following two tuples of messages

$$T_1^{(l)} = (m_1^{(1)}, \ldots, m_l^{(1)})$$

$$T_2^{(l)} = (m_1^{(2)}, \ldots, m_l^{(2)}),$$

[5]The probability that a hash function maps to a value is calculated over the random coins used to pick the hash function from the ensemble and also over the random coins used to generate the input to the hash function.

[6]Recall that P_1 is part of the prover P in the Fiat-Shamir proof of knowledge (P, V).

such that all the messages in both the tuples are of length polynomial in the security parameter. Further, the above two tuples satisfy the property that $m_i^{(1)} \neq m_j^{(1)}$ for $i \neq j$ and $i, j \in \{1, \ldots, l\}$. Then we claim that the following distributions are computationally indistinguishable.

$$\{\mathsf{Enc}(m_1^{(1)}), \ldots, \mathsf{Enc}(m_l^{(1)})\}$$

$$\{\mathsf{Enc}(m_1^{(2)}), \ldots, \mathsf{Enc}(m_l^{(2)})\}$$

The distinguisher does not get access to either the PRF key K and the secret key SK.

Proof. We prove this by induction on the length of the tuples. When a pair of tuples T_1 and T_2 contains just one message then the indistinguishability of $\mathsf{Enc}(T_1)$ and $\mathsf{Enc}(T_2)$ follows from the semantic security of the encryption scheme as well as the fact that the output of the PRF is computationally indistinguishable from the output of a purely random function. Throughout this proof, whenever we talk about indistinguishability we assume that the distinguisher does not have access to either the PRF key K or the secret key SK. Consider any two tuples $T_1^{(l-1)}$ and $T_2^{(l-1)}$, each of length $l-1$ for some integer $l > 2$, such that $m_i^{'(1)} \neq m_j^{'(1)}$ where $m_i^{'(k)}$ is the i^{th} message in the tuple $T_k^{(l-1)}$, for $k = 1, 2$ and for all i, j with $i \neq j$. By induction hypothesis, we will assume that the distributions $\{\mathsf{Enc}(T_1^{(l-1)})\}$ and $\{\mathsf{Enc}(T_2^{(l-1)})\}$ are computationally indistinguishable. We now show that for a pair of tuples $T_1^{(l)} = (m_1^{(1)}, \ldots, m_l^{(1)})$ and $T_2^{(l)} = (m_1^{(2)}, \ldots, m_l^{(2)})$, such that $m_i^{(1)} \neq m_j^{(1)}$, the distributions $\{\mathsf{Enc}(T_1^{(l)})\}$ and $\{\mathsf{Enc}(T_2^{(l)})\}$ are computationally indistinguishable. To show this, we assume that there exists a distinguisher D_l which distinguishes the distributions $\{\mathsf{Enc}(T_1^{(l)})\}$ and $\{\mathsf{Enc}(T_2^{(l)})\}$ with non-negligible probability. We then construct a distinguisher D_{l-1} to distinguish the distributions $\{\mathsf{Enc}(T_1^{(l-1)})\}$ and $\{\mathsf{Enc}(T_2^{(l-1)})\}$ with non-negligible probability, thus contradicting the hypothesis. The distinguisher D_{l-1} on input a tuple $\mathsf{Enc}(T^{(l-1)})$, first picks a message m, of length polynomial in the security parameter, uniformly at random. It then inputs the tuple $(\mathsf{Enc}(T^{(l-1)}), \mathsf{Enc}(m))$ to the distinguisher D_l. The distinguisher D_{l-1} then outputs whatever D_l outputs. With probability negligibly close to 1, m is different from the other messages in the tuple $T^{(l-1)}$. Conditioned on the event that m is different from the other messages in $T^{(l-1)}$, we have that the success probability of the distinguisher D_{l-1} is the same as the success probability of the distinguisher D_l thus contradicting the fact that the distributions $\{\mathsf{Enc}(T_1^{(l-1)})\}$ and $\{\mathsf{Enc}(T_2^{(l-1)})\}$ are computationally indistinguishable. This shows that the distributions $\{\mathsf{Enc}(T_1^{(l)})\}$ and $\{\mathsf{Enc}(T_2^{(l)})\}$ are computationally indistinguishable which proves the above claim.

Consider the following two tuples of messages.

1. $M_x^{(0)} = \Big(\big(m_1^{(0)}, \ldots, m_r^{(0)}\big), i, \beta, m_{(i,\beta)}^{(0)} \Big)_{\substack{1 \leq i \leq r, \\ 0 \leq \beta \leq 2^t - 1}} = \Big(\big((\alpha_1, 0^{|i|}, 0^{|b|}, 0^{|x|}), \ldots, (\alpha_r, 0^{|i|}, 0^{|b|}, 0^{|x|}) \big), i, \beta, \gamma_{(i,\beta)}^{(0)} \Big)_{\substack{1 \leq i \leq r, \\ 0 \leq \beta \leq 2^t - 1}}$,
where $\gamma_{(i,\beta)}^{(0)}$ is the output of $P_1(x, w, \alpha_i, \beta)$.
2. $M_x^{(1)} = \Big(\big(m_1^{(1)}, \ldots, m_r^{(1)}\big), i, \beta, m_{(i,\beta)}^{(1)} \Big)_{\substack{1 \leq i \leq r, \\ 0 \leq \beta \leq 2^t - 1}} = \Big(\big((\alpha, 1, 1, \mathsf{M}), \ldots, (\alpha, r, 1, \mathsf{M}) \big), i, \beta, (i, \beta) \Big)_{\substack{1 \leq i \leq r, \\ 0 \leq \beta \leq 2^t - 1}}$.

To prove the lemma we first make the following observations. In $M_x^{(1)}$, for $i \neq j$, the message $m_i^{(1)} (= (\alpha, i, 1, \mathsf{M}))$ is not equal to $m_j^{(1)} (= (\alpha, j, 1, \mathsf{M}))$. Moreover, $m_i^{(1)}$ is not equal to $m_{(j,\beta)}^{(1)}$ for $i = 1, \ldots, r$ and for $j = 1, \ldots, r$. Also, $i \neq i'$ or $\beta \neq \beta'$, $m_{(i,\beta)}^{(1)} (= (i, \beta))$ is not equal to $m_{(i',\beta')}^{(1)} (= (i', \beta'))$.

Similarly, in $M_x^{(0)}$, for $i \neq j$, $m_i^{(0)} (= (\alpha_i, 0, 0, 0))$ is not equal to $m_j^{(0)} (= (\alpha_j, 0, 0, 0))$ with probability negligibly close to 1; this follows from the commitment entropy property of the protocol (P, V). Further, for the same reason $m_i^{(0)}$ is not equal to $m_{(j,\beta)}^{(0)}$ with probability negligibly close to 1. We state the following claims to show that $m_{(j,\beta)}^{(0)}$ is equal to $m_{(j',\beta')}^{(0)}$ for either $j \neq j'$ or $\beta \neq \beta'$ with only negligible probability.

Claim 2. Let α be the output of $P_0(x, w)$. For any distinct β and β', $\Pr[P_1(x, w, \alpha, \beta) = P_1(x, w, \alpha, \beta')]$ is negligible.

Proof. Consider the set $S^{(x,w)}_{collision} = \{\alpha : \exists(\beta, \beta')$ such that $\Pr[P_1(x, w, \alpha, \beta) = P_1(x, w, \alpha, \beta')]$ is non-negligible$\}$. Let $\alpha \leftarrow P_0(x, w)$. To prove the above claim, we need to prove that $\alpha \in S^{(x,w)}_{collision}$ with negligible probability.

Assume for contradiction that $\alpha \in S^{(x,w)}_{collision}$ with non-negligible probability. Then we construct an algorithm which computes a witness for the given input with non-negligible probability thus leading to a contradiction. The algorithm is defined as follows.

- Run the zero-knowledge simulator $Z(x, \mathsf{YES})$ to obtain (α, β, γ).
- Choose β' uniformly at random.
- Check whether (α, β', γ) is an accepting transcript. If so, then output $K(x, \alpha, \beta, \beta', \gamma, \gamma)$, where K is the extractor; else abort.

We now claim that the above algorithm outputs a witness w such that $(x, w) \in R$ with non-negligible probability. The message α in the transcript generated by Z belongs to $S^{(x,w)}_{collision}$ with non-negligible probability (otherwise we can construct a distinguisher to violate the zero-knowledge property). Conditioned on the event that $\alpha \in S^{(x,w)}_{collision}$, we have $P_1(x, w, \alpha, \beta_1) = P_1(x, w, \alpha, \beta_1^{'})$ with non-negligible probability for some challenges $\beta_1, \beta_1^{'}$. Since (β, β') is picked uniformly at random from the challenge space which is of size polynomial in k, we have $\beta = \beta_1$ and $\beta' = \beta_1^{'}$ with non-negligible probability. This means that $P_1(x, w, \alpha, \beta) = P_1(x, w, \alpha, \beta^{'})$ with non-negligible probability. Since, (α, β, γ) and (α, β', γ) are both accepting transcripts, from the special soundness of (P^*, V^*) we have that $K(x, \alpha, \beta, \beta', \gamma)$ outputs a valid witness. Summing up, the algorithm outputs a witness with non-negligible probability. This completes the proof of Claim 1.

Claim 3. Let α and α' be the output of $P_0(x, w)$ on two different executions. For any β, β', $\Pr[P_1(x, w, \alpha, \beta) = P_1(x, w, \alpha', \beta')]$ is negligible.
Proof. Assume for contradiction that the above claim is false; i.e., $\exists\, \beta, \beta'$ $\Pr[P_1(x, w, \alpha, \beta) = P_1(x, w, \alpha', \beta')]$ is non-negligible. Then we design an efficient algorithm that when given an input $x \in L$ outputs a witness w with non-negligible probability such that $(x, w) \in R$.

On input $x \in L$, the algorithm proceeds as follows.

- Execute $Z(x, \mathsf{YES})$ to obtain (α, β, γ). Execute $Z(x, \mathsf{YES})$ to obtain $(\alpha', \beta', \gamma')$.
- Sample β'' uniformly at random. If $\beta'' = \beta$ then abort.
- Check whether $(\alpha, \beta'', \gamma')$ is an accepting transcript. If it is not an accepting transcript, then abort. Else, output $K(x, \alpha, \beta, \beta'', \gamma, \gamma')$.

By arguments similar to the ones in Claim 1, we can show that the algorithm outputs a valid witness with non-negligible probability which leads to a contradiction. This completes the proof of Claim 2.

From the above observations we deduce that all the messages contained in the ciphertexts in the tuple $\mathsf{Enc}(M_x^{(0)})$ (resp., $\mathsf{Enc}(M_x^{(1)})$) are different. We can now invoke Claim 1 to show that $D_x^{(0)}$ is indistinguishable from $D_x^{(1)}$. This completes the proof of Lemma 2.

□

We now show that the adversary $\mathcal{A}$ produces an accepting proof with non-negligible probability. Before we show this, observe that whenever $\mathcal{A}$ does not abort it produces an accepting proof. Hence, it suffices to bound the probability with which $\mathcal{A}$ aborts. In the following lemma, we show that $\mathcal{A}$ aborts with negligible probability, thus proving the theorem.

Lemma 3. *The adversary $\mathcal{A}$ aborts only with negligible probability.*

Proof. To prove this claim, we consider a modified version of the proof system (P_h, V_h), denoted by (P', V'). The modification is done in such a way that the probability that V_h accepts the proof produced by P_h is at most the probability that V' accepts the proof produced by P'. Also, we modify the adversarial procedure $\mathcal{A}$ to obtain $\mathcal{A}'$. The probability that $\mathcal{A}$ does not abort is the same as the probability that $\mathcal{A}'$ can produce a proof such that V' accepts. Using Lemma 2, we prove that the output distribution of P' and $\mathcal{A}'$ are computationally indistinguishable using which we show that the success probability of P' and the success probability of $\mathcal{A}'$ are negligibly close to each other. This further proves that the probability that $\mathcal{A}$ succeeds and the probability that P_h succeeds are negligibly close to each other. The lemma then follows. We are now ready to describe the technical details of the proof.

We first recall the construction of (P_h, V_h). The prover P_h on input (x, w) first runs the r copies of the prover P^* to obtain $(\alpha_1^*, \ldots, \alpha_r^*)$. Then by rewinding it finds the smallest β_i for every repetition i such that $\sum_{i=1}^{r} \mathsf{M}\big(x, (\alpha_1^*, \ldots, \alpha_r^*), i, \beta_i, \gamma_i^*\big)^{(b)} \leq S$, where M represents the Turing machine that computes the hash function h. And, the verifier V_h performs the following two checks: whether all the r transcripts are accepted by the verifier V^* and if $\sum_{i=1}^{r} \mathsf{M}\big(x, (\alpha_1^*, \ldots, \alpha_r^*), i, \beta_i, \gamma_i^*\big)^{(b)} \leq S$.

As mentioned in the beginning of the proof, we modify (P_h, V_h) to obtain (P', V') whose description is given below. We assume that the prover P' has access to $(\mathsf{SK}, \mathsf{K})$, which is the output of Setup, while the verifier V' does not have access to $(\mathsf{SK}, \mathsf{K})$.

$\mathsf{Prover}\ P'$: The prover P' on input (x, w), first runs the prover $P^*(x, w)$ in r independent repetitions to obtain r commitments $(\alpha_1^*, \ldots, \alpha_r^*)$. Then P' does the following, either sequentially or in parallel for each repetition i. For each $\beta_i = 0, 1, \ldots, 2^t - 1$, P' computes the final responses $P^*(x, w, \alpha_i^*, \beta_i) = \gamma_{i,\beta_i}^*$ by rewinding. The prover outputs $\pi = \Big((\alpha_1^*, \ldots, \alpha_r^*), i, \beta, \gamma_{i,\beta}^*\Big)_{1 \leq i \leq r,\ 0 \leq \beta \leq 2^t - 1}$.

$\mathsf{Verifier}\ V'$: The verifier V' on input x and $\pi = \Big((\alpha_1^*, \ldots, \alpha_r^*), i, \beta, \gamma_{i,\beta}^*\Big)_{1 \leq i \leq r,\ 0 \leq \beta \leq 2^t - 1}$ executes the following. For each repetition, it chooses the smallest $\beta_i \in \{0, \ldots, 2^t - 1\}$ for which $\mathsf{M}\big(x, (\alpha_1, \ldots, \alpha_r), i, \beta_i, \gamma_i\big)^{(b)}$ is the minimum among all the 2^t hash values. It accepts if and only if $\sum_{i=1}^{r} \mathsf{M}(x, (\alpha_1, \ldots, \alpha_r), i, \beta_i, \gamma_i)^{(b)} \leq S$.

We also consider a modified version of the adversary $\mathcal{A}$, denoted by $\mathcal{A}'$, which is defined as follows. On input x, $\mathcal{A}'$ picks α uniformly at random. Then it sends as proof, $\pi = \Big(\mathsf{Enc}((\alpha, 1, 1, M)), \ldots, \mathsf{Enc}((\alpha, r, 1, M)), i, \beta, \mathsf{Enc}\big((i, \beta)\big)\Big)_{\substack{1 \leq i \leq r, \\ 0 \leq \beta \leq 2^t - 1}}$ to verifier V'.

Now observe that $\Pr[V'(x, \pi') = \mathsf{Accept}\ :\ \pi' \leftarrow P'(x, w)] \geq \Pr[V_h(x, \pi) = \mathsf{Accept}\ :\ \pi \leftarrow P_h(x, w)]$. Similarly, it can be seen that $\Pr[V'(x, \pi') = 1\ :\ \pi \leftarrow \mathcal{A}'(x)] = \Pr[\mathcal{A} \text{ does not abort}]$. Further, note that the distribution $\mathsf{D}_x^{(1)}$ is identical to the distribution $\{\pi' : \pi' \leftarrow \mathcal{A}'(x)\}$. Similarly, the distribution $\mathsf{D}_x^{(0)}$ is identical to the distribution $\{\pi' : \pi' \leftarrow P'(x, w)\}$. Putting these arguments together, we have that the following quantity to be at most $negl(k)$.

$$\big|\Pr[V'(x, \pi') = \mathsf{Accept}\ :\ \pi' \leftarrow P'(x, w)] - \Pr[V'(x, \pi') = \mathsf{Accept}\ :\ \pi' \leftarrow \mathcal{A}'(x)]\big|$$

This is because, otherwise V' would act as a distinguisher distinguishing $\mathsf{D}_x^{(0)}$ and $\mathsf{D}_x^{(1)}$ contradicting Lemma 2. We can use Lemma 2 here since V' does not have access to either the PRF key K or the decryption key SK. This further implies that $|\Pr[V_h(x, \pi) = \mathsf{Accept}\ :\ \pi \leftarrow P_h(x, w)] - \Pr[\mathcal{A} \text{ does not abort}]| \leq negl(k)$. Hence, the probability that adversary $\mathcal{A}$ does not abort is negligibly close to 1. This concludes the proof of Lemma 3. □

□

4 Simplified Construction for Pseudorandom Hash Functions

In this section, we present a simpler construction to demonstrate the insecurity of the Fischlin transformation with respect to hash functions which behave as pseudorandom functions (i.e., the output of such a hash function for any input is indistinguishable from random). As in the main construction, we restrict our attention to the case when the worst case running time of the hash function is at most some fixed polynomial in the size of the input. More formally, the insecurity arguments hold only for those hash functions whose running time is at most $p_{\mathsf{hash}}(|\boldsymbol{y}|)$ on input $\boldsymbol{y}$, where p_{hash} is a polynomial. Unlike the main construction, the construction presented below does not require any initial setup.

Let (P, V) be any 3-round Fiat-Shamir proof of knowledge for some witness relation W. We extend (P, V) to obtain a 3-round Fiat-Shamir proof of knowledge (P^*, V^*) as we shall describe shortly.

Let $P = (P_0, P_1)$ and $V = (V_0, V_1)$. On input (x, w), P_0 generates α, and on input (x, w, α, β) P_1 generates γ. Also, V_0 and V_1 are such that V_0 interacts with P to produce a transcript (α, β, γ) by generating β uniformly at random and then V accepts if and only if $V_1(x, \alpha, \beta, \gamma)$ accepts. The protocol (P^*, V^*) is described below.

1. P^*: Run P_0 on (x, w) to obtain α. Define $\alpha^* = (\alpha, i, \mathsf{bit}, \mathsf{M})$, where each of $i, \mathsf{bit}, \mathsf{M}$ is set to 0, with their lengths being $\log(r)$ bits, 1 bit, and and $|x|$ bits, respectively[7]. Send α^* to V^*.

2. V^*: Run V_0 to obtain β. Send β to P^*.

3. P^*: Run P_1 on input (x, w, α, β) to obtain γ. Send γ to V^*.

4. V^*: Parse α^* as $(\alpha, i, \mathsf{bit}, \mathsf{M})$. If $\mathsf{bit} = 0$ then Accept if $V_0(x, \alpha, \beta, \gamma)$ accepts. If $\mathsf{bit} = 1$ then check if M is a valid Turing Machine. If M is not a valid TM then Reject. Else, Accept if all the following conditions hold:

   - $\beta = \mathsf{least}(x, \alpha, i, 1, \mathsf{M})$
   - $\gamma = 0$

where the function least is defined as follows.

```
least($x, \alpha, i$, M):

1. min ← $2^b + 1$
2. $\beta \leftarrow null$
3. For $j = 0$ to $2^t - 1$:
4.     $\boldsymbol{y} \leftarrow \Big(x, ((\alpha, 1, 1, \mathsf{M}), \ldots, (\alpha, r, 1, \mathsf{M})), i, j, 0\Big)$
5.     Execute M($\boldsymbol{y}$) upto $p_{\mathsf{hash}}(|\boldsymbol{y}|)$ steps
6.     If M($\boldsymbol{y}$) terminates within $p_{\mathsf{hash}}(|\boldsymbol{y}|)$ steps:
7.         hash ← M$(\boldsymbol{y})^{(b)}$
8.         If min > hash:
9.             min ← hash
10.            $\beta \leftarrow j$
11. Return $\beta$.
```

The functionality of the least algorithm defined above is very similar to the one defined in the main construction with the main difference being that the first message and the last messages are not encrypted in the algorithm described above unlike the least algorithm defined in the main construction. Further, the symbols r, b, t are as described in the Fischlin construction (c.f. Section 2). We now show that (P^*, V^*) satisfies all the properties of Fiat-Shamir proof of knowledge.

[7]Hereafter, unless specified otherwise, we maintain that the lengths of bit, i, M are as specified above.

4.1 On the Security of (P^*, V^*)

Theorem 2. *(P^*, V^*) is a Fiat-Shamir proof of knowledge for the relation W.*

Proof. [Completeness] For any $(\alpha^*, \beta, \gamma)$ resulting from the interaction between P^* and V^*, we have that $\alpha^* = (\alpha, i, \mathsf{bit}, \mathsf{M})$ with each of $i, \mathsf{bit}, \mathsf{M}$ being 0 and with (α, β, γ) generated as per the description of (P, V). Since, if V accepts (α, β, γ) then V^* accepts $(\alpha^*, \beta, \gamma)$, completeness of (P, V) implies that of (P^*, V^*).

[Special Soundness] Let K be a knowledge extractor for (P, V). We show that (P^*, V^*) satisfies special soundness by constructing a knowledge extractor K^* that uses K as follows: For any $(x, w) \in W_k$, on input $(x, \alpha^*, \beta, \gamma, \beta', \gamma')$, where $\alpha^* = (\alpha, i, \mathsf{bit}, \mathsf{M})$, $\beta \neq \beta'$, and $V^*(x, (\alpha, i, \mathsf{bit}, \mathsf{M}), \beta, \gamma) = V^*(x, (\alpha, i, \mathsf{bit}, \mathsf{M}), \beta', \gamma') = \mathsf{Accept}$, K^* outputs $K(x, \alpha, \beta, \gamma, \beta', \gamma')$. To see that K^* is indeed a knowledge extractor for (P^*, V^*), consider the following two cases.

- $\mathsf{bit} = 0$: Here, $V^*(x, (\alpha, i, \mathsf{bit}, \mathsf{M}), \beta, \gamma) = V^*(x, (\alpha, i, \mathsf{bit}, \mathsf{M}), \beta', \gamma') = \mathsf{Accept}$ implies that $V(x, \alpha, \beta, \gamma) = V(x, \alpha, \beta', \gamma') = \mathsf{Accept}$. Since special soundness of (P, V) ensures that for an input $(x, \alpha, \beta, \gamma, \beta', \gamma')$, K outputs w' such that $(x, w') \in W_k$, K^* also outputs a witness.

- $\mathsf{bit} = 1$: In this case, V^* accepts on both inputs $((\alpha, i, \mathsf{bit}, \mathsf{M}), \beta, \gamma)$ and $((\alpha, i, \mathsf{bit}, \mathsf{M}), \beta', \gamma')$ only if both β and β' are equal to $\mathsf{least}(\alpha, i, 1, \mathsf{M})$ which leads to a contradiction to $\beta \neq \beta'$.

[Commitment entropy] The first message of P^* contains α which has the same distribution as the first message of P and hence the commitment entropy property is satisfied.
[Public coin] This follows directly from the description of V^*.

[Unique responses] For any probabilistic polynomial-time algorithm $\mathcal{A}$, and $(x, (\alpha, i, \mathsf{bit}, \mathsf{M}), \beta, \gamma, \gamma') \leftarrow \mathcal{A}(1^k)$, we claim that the following is negligible in k:

$$\mathsf{Pr}[V^*((\alpha, i, \mathsf{bit}, \mathsf{M}), \beta, \gamma) = V^*((\alpha, i, \mathsf{bit}, \mathsf{M}), \beta, \gamma') = \mathsf{Accept} \,\&\, \gamma \neq \gamma'].$$

We establish the claim under the following two cases.

- $\mathsf{bit} = 0$: Note that, $V^*(x, (\alpha, i, \mathsf{bit}, \mathsf{M}), \beta, \gamma) = \mathsf{Accept}$ implies that $V(x, \alpha, \beta, \gamma) = \mathsf{Accept}$, and also, $V^*(x, (\alpha, i, \mathsf{bit}, \mathsf{M}), \beta, \gamma') = \mathsf{Accept}$ implies that $V(x, \alpha, \beta, \gamma') = \mathsf{Accept}$. Since $\mathsf{Pr}[V(\alpha, \beta, \gamma) = V(\alpha, \beta, \gamma') = \mathsf{Accept} \,\&\, \gamma \neq \gamma']$ is negligibly close to 0, we have $\mathsf{Pr}[V^*((\alpha, i, \mathsf{bit}, h), \beta, \gamma) = V^*((\alpha, i, \mathsf{bit}, h), \beta, \gamma') = \mathsf{Accept} \,\&\, \gamma \neq \gamma' | \mathsf{bit} = 0]$ is negligibly close to 0.

- $\mathsf{bit} = 1$: Observe that, $V^*(x, (\alpha, i, \mathsf{bit}, \mathsf{M}), \beta, \gamma) = \mathsf{Accept}$ implies that $\gamma = 0$, and also, $V^*(x, (\alpha, i, \mathsf{bit}, \mathsf{M}), \beta, \gamma') = \mathsf{Accept}$ implies that $\gamma = 0$, thus giving us $\gamma = \gamma'$.

[Honest Verifier Zero-knowledge] To prove that (P^*, V^*) is an HVZK protocol, we construct a special zero-knowledge simulator Z^* for (P^*, V^*) as follows. Let Z be a special zero-knowledge simulator for the protocol (P, V). On input $(x, \beta, \mathsf{memebership})$, where $\mathsf{memebership} \in \{\mathsf{yes}, \mathsf{no}\}$, Z^* runs $Z(x, \beta, \mathsf{memebership})$ to obtain (α, β, γ). If $(\alpha, \beta, \gamma) = \perp$, then Z^* also outputs $\perp$; otherwise, it outputs $((\alpha, i, \mathsf{bit}, \mathsf{M}), \beta, \gamma)$, where i, bit and M are set to 0.

To prove that Z^* is a special zero-knowledge simulator for (P^*, V^*), assume for contradiction that there exists a distinguisher $D^* = (D_0^*, D_1^*)$ such that the statistical distance, $\epsilon(k)$, between following two distributions is non-negligible.

- $\mathsf{Dist}^*_{\mathsf{real}}$: Let $(x, w, \mathsf{state}) \leftarrow D_0^*(1^k)$ and $(\alpha^*, \beta, \gamma) \leftarrow (P^*(w), V^*)(x)$ if $(x, w) \in W_k$, and $(\alpha^*, \beta, \gamma) \leftarrow \perp$ otherwise. Output $D_1^*(\alpha^*, \beta, \gamma, \mathsf{state})$.

- $\mathsf{Dist}^*_{\mathsf{sim}}$: Let $(x, w, \mathsf{state}) \leftarrow D_0^*(1^k)$ and $(\alpha^*, \beta, \gamma) \leftarrow Z^*(x, \mathsf{yes})$ if $(x, w) \in W_k$, and $(\alpha^*, \beta, \gamma) \leftarrow Z^*(x, \mathsf{no})$ otherwise. Output $D_1^*(\alpha^*, \beta, \gamma, \mathsf{state})$.

Then, we construct a distinguisher $D = (D_0, D_1)$ against the HVZK property of (P, V) as follows. $D_0(1^k)$ runs $D_0^*(1^k)$ to obtain (x, w, state) and outputs the same. Once D_1 receives (α, β, γ), if $(\alpha, \beta, \gamma) \neq \perp$ then it outputs $D_1^*((\alpha, i, \mathsf{bit}, \mathsf{M}), \beta, \gamma, \mathsf{state})$, where i, bit and M are set to 0; otherwise, it sets $(\alpha^*, \beta, \gamma) \leftarrow \perp$ and outputs $D_1^*(\alpha^*, \beta, \gamma, \mathsf{state})$. Now consider the following distributions.

- $\mathsf{Dist}_{\mathsf{real}}$: Let $(x, w, \mathsf{state}) \leftarrow D_0(1^k)$ and $(\alpha, \beta, \gamma) \leftarrow (P(w), V)(x)$ if $(x, w) \in W_k$, and $(\alpha, \beta, \gamma) \leftarrow \perp$ otherwise. Output $D_1(\alpha, \beta, \gamma, \mathsf{state})$.
- $\mathsf{Dist}_{\mathsf{sim}}$: Let $(x, w, \mathsf{state}) \leftarrow D_0(1^k)$ and $(\alpha, \beta, \gamma) \leftarrow Z(x, \mathsf{yes})$ if $(x, w) \in W_k$, and $(\alpha, \beta, \gamma) \leftarrow Z(x, \mathsf{no})$ otherwise. Output $D_1(\alpha^*, \beta, \gamma, \mathsf{state})$.

Now, from the way Z^* and D are constructed, the output of $\mathsf{Dist}_{\mathsf{real}}$ is the same as the output of $\mathsf{Dist}^*_{\mathsf{real}}$. Similarly, the output of $\mathsf{Dist}_{\mathsf{sim}}$ is the same as the output of $\mathsf{Dist}^*_{\mathsf{sim}}$. This implies that the statistical distance between $\mathsf{Dist}_{\mathsf{real}}$ and $\mathsf{Dist}_{\mathsf{sim}}$ is also $\epsilon(k)$, a contradiction.

□

4.2 On the Insecurity of (P_h, V_h)

Let $\mathcal{F}_{p_{\mathsf{hash}}}$ be a pseudorandom function family such that each function in the family can be be evaluated on any input in time p_{hash} in the size of its inputs, where p_{hash} is a polynomial. Now we shall show that for every such hash function that is used to instantiate the random oracle in the non-interactive ZK PoK that is obtained by applying Fischlin transformation to (P^*, V^*), the resulting protocol does not satisfy the soundness.

In order to model a pseudorandom hash function family, one may look at the function to take a randomly chosen secret key also as an input. Thus, to instantiate the random oracle, first a key K is picked uniformly at random. Let f_K be the pseudorandom hash function corresponding to K in $\mathcal{F}_{p_{\mathsf{hash}}}$. Henceforth, we shall denote f_K by h for a simpler notation. Let (P_h, V_h) be the protocol obtained by applying Fischlin transformation to (P^*, V^*). The following theorem shows that the protocol (P_h, V_h) does not satisfy the soundness property.

Theorem 3. *Let (P^*, V^*) be the 3-round Fiat-Shamir proof of knowledge described above for a witness relation W with corresponding language L and let $(P^{\mathcal{O}}, V^{\mathcal{O}})$ be the non-interactive zero-knowledge proof of knowledge protocol obtained by applying Fischlin transformation to (P^*, V^*). Then for any pseudorandom hash function h in $\mathcal{F}_{p_{\mathsf{hash}}}$ that is used to instantiate the random oracle $\mathcal{O}$, the resulting protocol (P_h, V_h) does not satisfy soundness. In other words, there exists an adversary $\mathcal{A}$ that outputs (x, π) such that $\Pr[V_h(x, \pi) = \mathsf{Accept}$ and $x \notin L]$ is non-negligible.*

Proof. Let M be an efficient Turing machine that computes the hash function h. Now to prove that (P_h, V_h) does not satisfy soundness, we construct a PPT adversary $\mathcal{A}$ that outputs (x, π) such that $x \notin L$ and $V_h(x, \pi) = \mathsf{Accept}$ with probability negligibly close to 1.

The adversary $\mathcal{A}$ is described as follows.

- On input M, choose a string $x \notin L$ such that $|x| = |\mathsf{M}|$.
- Then compute β_i as $\beta_i = \mathsf{least}(\alpha, i, 1, \mathsf{M})$ for all $i \in [r]$.
- Check whether $\sum_{i=1}^{r} \mathsf{M}\big(x, (\alpha, 1, 1, \mathsf{M}), \ldots, (\alpha, r, 1, \mathsf{M}), i, \beta_i, 0\big) \leq S$. If so, then output the proof π as $\pi = ((\alpha, i, 1, \mathsf{M}), \beta_i, 0)_{1 \leq i \leq r}$ else it aborts. Observe that when the adversary does not abort, then the proof produced by the adversary is accepted by the verifier. So it suffices to analyse the probability with which the adversary does not abort.

We claim that with probability negligibly close to 1, the adversary does not abort. From now on, we will perform the probability analysis assuming that h is a purely random function instead of a pseudorandom function. Note that if we prove that adversary aborts with negligible probability assuming that h is a purely random function then this would imply that the probability with which the adversary aborts is negligible when h is a pseudorandom function. To show that the probability that the adversary does not abort is negligibly close to 1 we use arguments similar to the ones used in [Fis05].

Let $\mathrm{s_i} := \mathsf{M}(x, (\alpha_1^*, \ldots, \alpha_r^*), i, \beta_i, \gamma_i)$ represent the random variable for the hash value corresponding to the the i^{th} execution. We shall first bound the value of each $\mathrm{s_i}$ and then we proceed to bound the value for the sum. We have,

$$\mathsf{Pr}[\exists i : \ \mathrm{s_i} > S] \leq r \cdot \left(1 - (S+1)2^{-b}\right)^{2^t} \leq r \cdot e^{-(S+1)2^{t-b}}.$$

The first inequality follows from the fact that the probability that for each repitition the output of the hash function is at most S is at least $(S+1)2^{-b}$. This gives us that, in one execution, the probability of the hash value exceeding S is negligible. This, in turn, implies that the sum of the hash values exceeds rS is also negligible, since r is logarithmic. Hence, hereafter, we present our arguments conditioned on the event that $\sum_{i=1}^{r} \mathrm{s_i} \leq rS$.

Recall that the adversary $\mathcal{A}$ aborts if the sum $T := \sum_{i=1}^{r} \mathrm{s_i}$ exceeds rS, with each $\mathrm{s_i} \geq 0$. For any such $T = S+1, S+2, \ldots, rS$ there are at most $\binom{T+r-1}{r-1}$ possible values of $\mathrm{s_1}, \ldots, \mathrm{s_r}$ that will sum up to T. This can be upper bounded by

$$\binom{T+r-1}{r-1} \leq \left(\frac{e(rS+r-1)}{r-1}\right)^{r-1} \leq (e(2S+1))^{r-1} \leq e^{r \ln(e(2S+1))}.$$

Now, the probability that we obtain such a sum for a given partition, $\mathrm{s_1} = s_1, \ldots, \mathrm{s_r} = s_r$, is at most

$$\begin{aligned}\prod_{i=1}^{r} \mathsf{Pr}[\mathrm{s_i} = s_i] \leq \prod_{i=1}^{r} \mathsf{Pr}[\mathrm{s_i} \geq s_i] \leq \prod_{i=1}^{r} (1 - s_i 2^{-b})^{2^t} \\ = \prod_{i=1}^{r} e^{-s_i 2^{t-b}} = e^{-(\sum s_i)2^{t-b}} = e^{-T2^{t-b}} \leq e^{-(S+1)2^{t-b}}\end{aligned}$$

Since the parameters $r = O(\log(k))$ and $2^{t-b} = \omega(\log(k))$, and since $\ln(2S+1) \leq S+1$, the probability that the adversary gets a sum T such that $S < T \leq rS$ is at most $\exp(r \ln(e(2S+1)) - (S+1)2^{t-b})$ which is negligible. Thus, the adversary aborts only with negligible probability. In other words, with probability negligibly close to 1, $\mathcal{A}$ produces an accepting proof.

$\square$

References

[AABN02] Michel Abdalla, Jee Hea An, Mihir Bellare, and Chanathip Namprempre. From identification to signatures via the fiat-shamir transform: Minimizing assumptions for security and forward-security. In *EUROCRYPT*, pages 418–433, 2002.

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

[BBS04] Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In *CRYPTO*, pages 41–55, 2004.

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

[BGI$^+$01] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In *CRYPTO*, pages 1–18, 2001.

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *ACM Conference on Computer and Communications Security*, pages 62–73, 1993.

[CGH04] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited. *J. ACM*, 51:557–594, July 2004.

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

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

[FS86] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *CRYPTO*, pages 186–194, 1986.

[GK03] Shafi Goldwasser and Yael Tauman Kalai. On the (in)security of the fiat-shamir paradigm. In *FOCS*, pages 102–113, 2003.

[GK05] S. Goldwasser and Yael Tauman Kalai. On the impossibility of obfuscation with auxiliary input. In *Foundations of Computer Science, 2005. FOCS 2005. 46th Annual IEEE Symposium on*, pages 553 – 562, oct. 2005.

[GQ88] Louis C. Guillou and Jean-Jacques Quisquater. A "paradoxical" indentity-based signature scheme resulting from zero-knowledge. In *CRYPTO*, pages 216–231, 1988.

[MR02] Silvio Micali and Leonid Reyzin. Improving the exact security of digital signature schemes. *J. Cryptology*, 15(1):1–18, 2002.

[Oka92] Tatsuaki Okamoto. Provably secure and practical identification schemes and corresponding signature schemes. In *CRYPTO*, pages 31–53, 1992.

[Pas04] Rafael Pass. Bounded-concurrent secure multi-party computation with a dishonest majority. pages 232–241, 2004.

[PS00] David Pointcheval and Jacques Stern. Security arguments for digital signatures and blind signatures. *J. Cryptology*, 13(3):361–396, 2000.

[Sch91] Claus-Peter Schnorr. Efficient signature generation by smart cards. *J. Cryptology*, 4(3):161–174, 1991.