IACR Communications in Cryptology
ISSN 3006-5496, Vol. 2, No. 1, 20 pages.

https://doi.org/10.62056/avr-11fgx

Goldreich-Krawczyk Revisited: A Note on the Zero Knowledge of Proofs of Knowledge

Lior Rotem

Stanford University, Stanford, USA

Abstract. The seminal work of Goldreich and Krawczyk (SIAM Journal on Computing) shows that any constant-round public-coin interactive proof for languages not in BPP cannot be black-box zero knowledge. Their result says nothing, however, about proofs (or arguments) of knowledge for languages in BPP. As a special case, their work leaves open the question of whether Schnorr's protocol for proving knowledge of discrete logarithms in cyclic groups is black-box zero knowledge.
In this work we focus on the zero knowledge of *proofs of knowledge*, centering on Schnorr's protocol as a prominent example. We prove two lower bounds, ruling out two different classes of simulators through which Schnorr's protocol can be proven zero knowledge:

1. We prove that if a relation $\mathcal{R}$ has a public-coin interactive proof of knowledge that is black-box zero knowledge and this protocol is compatible with the Fiat-Shamir transform in the random oracle model, then $\mathcal{R}$ must be efficiently searchable. As an immediate corollary, we deduce that Schnorr's protocol cannot be black-box zero knowledge in groups in which discrete log is hard.
2. We define a new class of simulators for Schnorr's protocol, which we call *generic simulators*. A generic simulator is one that works in any cyclic group, and does not use the representation of the specific group in which Schnorr's protocol is instantiated. We prove that Schnorr's protocol cannot have generic simulators.

As an additional contribution, we generalize the original lower bound of Goldreich and Krawczyk, to prove that a language not in BPP cannot have an interactive proof (not necessarily of knowledge) that is both black-box zero knowledge and compatible with the Fiat-Shamir transform in the random oracle model. In conjunction with recent works, this extends the Goldreich-Krawczyk lower bound to public-coin protocols that are not constant-round but have round-by-round soundness, including the parallel repetition of any public-coin interactive proof.

1 Introduction

Zero-knowledge proofs, invented by Goldwasser, Micali, and Rackoff [GMR85], allow to prove a statement without revealing anything about it other than its validity.[1] Loosely speaking, this is typically formalized by requiring the existence of a special algorithm, called the simulator, that simulates the view of any potentially malicious verifier while only getting the statement as input. Zero-knowledge proofs serve as a cornerstone of modern cryptography and have found numerous applications since their inception.

E-mail: lrotem@cs.stanford.edu (Lior Rotem)

[1] For conciseness, in this introduction we use "proofs" to refer to both the statistically- and computationally-sound protocols.

Received: 2025-01-13 Accepted: 2025-03-11

Of particular interest in this work are zero-knowledge proofs *of knowledge* [GMR85, BG93]. These are zero-knowledge proofs that enjoy the additional *knowledge soundness* property: in the context of NP statements, if a prover manages to convince the verifier that an instance x is in some NP language $\mathcal{L}$, then it must "know" an NP witness w for x. The definition is made precise in Section 2. Proofs of knowledge can be non-trivial to construct even for NP languages for which trivial zero-knowledge proofs exist. A prime example of such a situation is Schnorr's proof of knowledge of discrete logarithms [Sch90, Sch91]. Given a cyclic group $\mathbb{G}$ of order p, a generator $g \in \mathbb{G}$, and a group element $h \in \mathbb{G}$, Schnorr's protocol can be used to prove knowledge of $\mathsf{log}_g(h)$. Informally, the protocol proceeds in three rounds:

1. The prover samples a uniformly random exponent $r \leftarrow\!\!\$\ \mathbb{Z}_p$, commits to it by computing $R \leftarrow g^r$, and sends R to the verifier.
2. The verifier samples a random challenge c from some distribution over $\mathbb{Z}_p$.
3. The prover computes $z \leftarrow c \cdot \mathsf{log}_g(h) + r \in \mathbb{Z}_p$ and sends it to the verifier.

The verifier accepts if and only if $g^z = h^c \cdot R$.

In the terminology of NP, Schnorr's protocol is a proof of knowledge for the relation $\mathcal{R}_{\mathsf{dlog}} = \{(h, z) \ : \ g^z = h\}$. However, the NP language $\mathcal{L}_{\mathsf{dlog}}$ induced by the relation is trivial: since g is a generator, every group element $h \in \mathbb{G}$ has a witness z such that $h = g^z$. This means that $\mathcal{L}_{\mathsf{dlog}}$ has a very degenerate zero-knowledge proof: the prover sends nothing and the verifier simply accepts any well-formed group element! Still, as a proof *of knowledge*, Schnorr's protocol has been highly influential in practice, as it served as the basis for Schnorr signatures [Sch90, Sch91, PS96, BN06] and also as a template for many other similar interative proofs in cryptographic groups (for example, [GQ90, CP93, Oka93, CS97, Mau15]; see also [Dam02, BS23] and the references therein).

Public-coin zero-knowledge proofs and their limitations. A very useful subclass of zero-knowledge proofs is that of *public-coin* protocols. In such protocols, all of the verifier's messages are drawn uniformly at random from some message space parametrizing the protocol. Schnorr's protocol mentioned above is an example of a public-coin protocol. Public-coin protocols are advantageous due to the fact that they are publically verifiable: the verifier holds no secrets, and any outside observant can similarly be convinced of the validity of the claim, so long as they trust the verifier to sample their messages properly. Moreover, if a public-coin zero knowledge proof is also *constant round*, then one can apply the Fiat-Shamir transform [FS87] to make it non-interactive with provable security guarantees in the random oracle model [BR93].

In their seminal work, Goldreich and Krawczyk [GK96] proved that public-coin proofs exhibit an inherent tradeoff: if a language $\mathcal{L}$ has a proof system that is simultaneously constant round and black-box zero knowledge, then it must be that $\mathcal{L}$ can be efficiently decided by a probablisitic algorithm (that is, $\mathcal{L}$ is in the class BPP). By *black-box* zero knowledge we mean that the simulator accesses the malicious verifier only in a black-box way. An important corollary of their result is that parallel repetition of the well-known GMW protocol [GMW87] is not black-box zero knowledge (at least unless $\mathsf{NP} = \mathsf{BPP}$). The Goldreich–Krawczyk lower bound, however, says nothing about protocols for languages *inside* of BPP. In particular, it says nothing about Schnorr's protocol, since $\mathcal{L}_{\mathsf{dlog}}$ is trivially in BPP (an algorithm that outputs 1 on any well-formed group element trivially decides the language). This leaves open the following question:

Is Schnorr's protocol zero knowledge?

Note that similarly to the parallel repetition of the GMW protocol, Schnorr's protocol *is* honest verifier zero knowledge, and so the question pertains to zero knowledge against potentially malicious verifiers.

1.1 Our Contributions

In this note, we give strong evidence suggesting Schnorr's protocol might not be zero knowledge. Concretely, we prove two lower bounds on the zero-knowledge of Schnorr's protocol, excluding two different classes of simulators.

A lower bound on black-box zero knowledge. Our first lower bound extends the lower bound of Goldreich and Krawczyk to Schnorr's protocol, and rules out the possibility that it is black-box zero knowledge in discrete-log-hard groups. More concretely, we prove the following theorem.

Theorem 1 (informal)**.** *If schnorr's protocol is black-box zero knowledge in a group $\mathbb{G}$, then there is a PPT algorithm A that solves the discrete log problem in $\mathbb{G}$.*

In fact, we prove a more general lower bound, of which Theorem 1 is an immediate corollary. Our general lower bound applies to any interactive proof of knowledge that is compatible with the Fiat-Shamir transform [FS87]. We say that an interactive proof of knowledge (P, V) is Fiat-Shamir compatible, if the non-interactive proof that is obtained from applying the Fiat-Shamir transform to (P, V) is a non-interctive proof of knowledge in the random oracle model. Schnorr's protocol is Fiat-Shamir compatible [Sch90, Sch91, PS96, BN06, RS21]. In more detail, we prove that if (P, V) is a Fiat-Shamir compatible interactive proof for some relation $\mathcal{R} \subset \mathcal{X} \times \mathcal{W}$, then either it is not black-box zero knowledge, or $\mathcal{R}$ is efficiently searchable. By that, we mean that there is a PPT algorithm that given $x \in \mathcal{X}$ finds a $w \in \mathcal{W}$ such that $(x, w) \in \mathcal{R}$.

Theorem 2 (informal)**.** *Let (P, V) be an interactive proof of knowledge for a relation $\mathcal{R} \subset \mathcal{X} \times \mathcal{W}$. If (P, V) is black-box zero knowledge and Fiat-Shamir compatible, then $\mathcal{R}$ is efficiently searchable.*

Apart from Schnorr's protocol (and other so-called Σ-protocols [Dam02, BS23]), it was recently proved that any protocol that satisfies a generalized notion of special soundness (called $(k_1, \ldots, k_r)$-special soundness) – a notion that applies to many multi-round public-coin protocols – is Fiat-Shamir compatible [AFK22, AFK23]. Hence, by Theorem 2, any such protocol is not black-box zero knowledge.

Corollary: Sequential composition of black-box zero-knowledge protocols. Goldreich and Oren [GO94] proved that black-box zero knowledge is closed under sequential composition. Taken together with Theorem 2, this shows that if a relation $\mathcal{R}$ has an argument system that is both: (1) black-box zero knowledge, and (2) its sequential repetition is Fiat-Shamir compatible, then $\mathcal{R}$ must be efficiently searchable. As a concrete example, the reader may think of Schnorr's protocol with polynomial-sized challenge space. It is not hard to see that in this case, Schnorr's protocol *is* black-box zero knowledge. The corollary says that, assuming discrete log is hard in the underlying group, the sequential composition of Schnorr's protocol with polynomial-sized challenge space is not Fiat-Shamir compatible. This can be seen directly,[2] but the corollary shows that this is true for *any* black-box zero-knowledge protocol for a hard relation.

[2] Let us first think of Schnorr's three-round protocol without sequential repetition, with a polynomial-sized challenge space. A malicious prover, that does not "know" $\mathsf{dlog}_g(h)$, may convince the verifier otherwise by repeatedly invoking the honest-verifier zero knowledge simulator for Schnorr, hoping that the resulted transcript will be consistent with the Fiat-Shamir compiler. Since the challenge space is of polynomial size, this will happen after a polynomial number of invocations of the simulator in expectation. This "attack" can be composed with itself sequentially to show that the sequential composition of Schnorr's protocol with a small challenge space is not Fiat-Shamir compatible.

Generalizing Goldreich-Krawzyck beyond constant-round protocols. As an additional contribution, that follows rather immediately from of Goldreich and Krawzyck [GK96], we observe that their lower bound can be generalized to apply not to just constant-round public coin interactive proofs, but to any interactive proof that is compatible with the Fiat-Shamir transform. By that, we mean that its non-interactive Fiat-Shamir compilation results in a sound non-interactive proof in the random oracle model.[3]

Theorem 3 (informal)**.** *Let (P, V) be an interactive proof for a language $\mathcal{L}$. Then, if (P, V) is black-box zero knowledge and Fiat-Shamir compatible, then $\mathcal{L}$ is in* BPP*.*

Since any constant-round public-coin interactive proof is Fiat-Shamir compatible, the bound of Goldreich and Krawzyck follows from Theorem 3. However, there are examples of public-coin protocols that are Fiat-Shamir compatible while not being constant round. In particular, it was shown that any public-coin interactive proof that satisfies a special notion of soundness called *round-by-round soundness* is Fiat-Shamir compatible [CCH+19, CCH+18, CLW18, Hol19]. Moreover, any public-coin interactive proof can be made round-by-round sound simply by parallel repetition [CCH+19]. Theorem 3 thus shows that the parallel repetition of *any* public coin interactive proof is not black-box zero knowledge. This can be seen as generalizing Goldreich-Krawzyck, whose lower bound rules out that parallel repetition of GMW [GMW87] is black-box zero knowledge.

Finally, note that similarly to Theorem 2, when taken together with Goldreich and Oren [GO94], Theorem 3 shows that the sequential repetition of any protocol for a hard language that is black-box zero-knowledge cannot be Fiat-Shamir compatible.

A lower bound on generic zero knowledge. Our second main lower bound is specific to Schnorr's protocol. We define a new class of simulators for Schnorr's protocol (or any other protocol that can be defined in the generic-group model [Sho97]), which we call *generic simulators*. Technically speaking, a simulator for Schnorr's protocol is generic if it can be defined as a generic algorithm in Shoup's generic group model [Sho97]. Intuitively, a generic simulator is one that works in any cyclic group, and does not use the representation of the specific group in which Schnorr's protocol is instantiated. We say that a protocol satisfies *generic zero knowledge* if any malicious generic verifier has a corresponding generic simulator. Note that some protocols do have simulators that are (non-trivially) generic. For example, if one instantiates the GMW protocol [GMW87] using a generic group-based commitment scheme, then the resulting protocol has a generic simulator.

Equipped with our new definition, we rule out the existence of generic simulators for Schnorr's protocol.

Theorem 4 (informal)**.** *Schnorr's protocol is not generic zero knowledge.*

Theorem 4 is proven by presenting a specific "malicious" generic verifier that cannot possibly have an efficient simulator. We show that if this verifier has an efficient simulator, then this simulator can be transformed into an algorithm for computing discrete log in the GGM, that is too good to be true. By too good to be true, we mean that it violates Shoup's lower bound on the hardness of discrete log in the GGM [Sho97].

Taken together, Theorems 1 and 4 show that if Schnorr's protocol is zero-knowldge (against malicious verifiers), then it must be due to a simulator that simultaneously makes non-black-box use of both the malicious verifier and the representation of the group. Indeed, such a simulator seems unlikely and would have to rely on fundamentally new ideas.

[3] That is in contrast to the compatibility of proofs of knowledge with the Fiat-Shamir transform, that required that the resulting non-interactive proof be *knowledge sound* in the random oracle model.

1.2 Additional Related Work

Our work is closely related to the work Dwork, Naor, Reingold, and Stockmeyer [DNRS03]. Among the many contributions of their work, they made the following observation: if a three-round public-coin interactive proof (P, V) for a language $\mathcal{L}$ can be made non-interactive using the Fiat-Shamir transform in the *standard model* using some concrete hash function H, then (P, V) is not zero knowledge. Theorem 3 can be seen as analogous to the observation of Dwork et al., replacing the standard model with the random oracle model, and zero knowledge with black-box zero knowledge.

Chen, Lombardi, Ma, and Quach [CLMQ21] considered the question of whether the Fiat-Shamir transform must be instantiated with a cryptographic hash function. They considered many facets of this questions. One of their conclusions was that in the GGM, Fiat-Shamir can be applied to Shnorr's protocol with a non-cryptographic hash function. The malicious verifier that we construct in the proof of Theorem 4 can be interpreted as a non-cryptographic hash function (albeit a different one than in [CLMQ21]) that can implement the Fiat-Shamir transform.

Holmgren, Lombardi, and Rothblum [HLR21] proved that under suitable cryptographic assumptions, the Fiat-Shamir transform can be applied to the parallel repetition of (an instantiation of) the GMW protocol (and generalizations thereof). Together with Dwork et al. [DNRS03], this implies that the parallel repetition of the GMW protocol is not zero knowledge.

2 Preliminaries

In this section, we review the basic notation and definitions we rely on in this note. Most of the section is dedicated to defining zero-knowledge proofs and zero-knowledge proofs of knowledge, as well as standard notions in complexity and k-wise independent hash functions. A reader that is familiar with these notions may prefer to skip to Section 3.

Notation and basic conventions. For a binary string $x \in \{0,1\}^*$, we denote its length by $|x|$. For any integer $c \in \mathbb{N}$, we denote by $\{0,1\}^{\leq c}$ the set of all binary strings of length at most c. For a set $\mathcal{X}$, we denote by $x \leftarrow\!\$\, \mathcal{X}$ the process of sampling x from the uniform distribution over $\mathcal{X}$. For a distribution X over $\mathcal{X}$, we write $x \leftarrow\!\$\, X$ to denote the process of sampling an element from $\mathcal{X}$ according to X. For an integer $n \in \mathbb{N}$, we use $[n]$ to denote the set $\{1, \ldots, n\}$. A function $\nu : \mathbb{N} \to \mathbb{R}^+$ is said to be *negligible* if for any polynomial $p(\cdot)$ there exists an integer $N \in \mathbb{N}$ such that $\nu(n) < 1/p(n)$ for all $n > N$. We write that an algorithm is PPT as a shorthand for it being probabilistic and polynomial-time. We say that two ensembles of distributions $X = \{X_\lambda\}_{\lambda \in \mathbb{N}}$ and $Y = \{Y_\lambda\}_{\lambda \in \mathbb{N}}$ are (computationally) indistinguishable if for any PPT algorithm A there is a negligible function $\nu(\cdot)$ such that for every $\lambda \in \mathbb{N}$ it holds that

$$\left| \Pr_{x \leftarrow\!\$ X_\lambda} [A(x) = 1] - \Pr_{y \leftarrow\!\$ Y_\lambda} [A(y) = 1] \right| \leq \nu(\lambda).$$

2.1 Zero-Knowledge Interactive Proofs

We briefly recall the notions of interactive proofs, zero knowledge, and proofs of knowledge. See Goldreich [Gol01] for further discussion.

We start by defining general two-party interactive protocols. A two-party interactive protocol is a pair (A, B) of algorithms, each computing its corresponding next message functions. That is, $A(x, a, m_1, \ldots, m_k; r)$ is a function mapping a shared input x, a private input a, messages $m_1, \ldots, m_k$ exchanged so far, and randomness r, to the next outgoing message m_{k+1} from A to B. The function $B(x, b, m_1, \ldots, m_k; r)$ is similarly defined. The

transcript of the protocol is the concatenation of all messages exchanged by A and B on shared input x and private inputs a and b. For fixed x, a and b it is a random variable over the randomness of A and B. We say that (A, B) is polynomially bounded if there exists a polynomial p such that for all x, a and b it holds that the trascipt is of length at most $p(|x|)$ with probability 1.

An interactive proof is a two-party interactive protocol (P, V), consisting of a prover P and a verifier V. Both parties receive a joint input x, known as the statement, as input. P additionally receives a witness w as a private auxiliary input. At the end of the protocol, V outputs a special message, accept or reject, which terminates the execution. We denote by $\langle P(w), V\rangle(x)$ the output of V when P and V interact on shared input x and P additionally runs on the private input w. Note that for fixed x and w, this is a random variable over the random coins of P and V.

Definition 1. Let $\mathcal{R}$ be a relation, where $\mathcal{R} \subseteq \mathcal{X} \times \mathcal{W}$ for sets $\mathcal{X}$ and $\mathcal{W}$. Let $\mathcal{L} = \{x \in \mathcal{X} \ : \ \exists w \in \mathcal{W} \text{ s.t. } (x, w) \in \mathcal{R}\}$. We say that (P, V) is an *interactive proof* for $\mathcal{L}$ if it satisfies the following conditions:

- **Efficiency:** (P, V) is polynomially bounded and V is a PPT algorithm. If P is also a PPT algorithm, we say that (P, V) has an *efficient prover*.
- **Completeness:** There exists a negligible function $\nu_{\mathsf{cmp}}(\cdot)$ such that for every $(x, w) \in \mathcal{R}$, it holds that

$$\Pr\left[\langle P(w), V\rangle(x) = \mathsf{accept}\right] \geq 1 - \nu_{\mathsf{cmp}}(|x|).$$

  If $\nu_1 = 0$, we say that (P, V) satisfied perfect completeness.
- **Soundness:** For every algorithm P^* there exists a negligible function ν_{snd} such that for every $x \notin \mathcal{L}$, it holds that

$$\Pr\left[\langle P^*, V\rangle(x) = \mathsf{accept}\right] \leq \nu_{\mathsf{snd}}(|x|).$$

We may require that soundness only holds against computationally-bounded provers. We call such protocols *interactive arguments*.

Definition 2. Let $\mathcal{R}$ and $\mathcal{L}$ be defined as in Definition 1. We say that (P, V) is an *interactive argument* for $\mathcal{L}$ if it satisfies efficiency and correctness as in Definition 1 and the soundness condition is replaced by:

- **Soundness:** For every PPT algorithm P^* there exists a negligible function $\nu_{\mathsf{snd}}(\cdot)$ such that for every $x \notin \mathcal{L}$, it holds that

$$\Pr\left[\langle P^*, V\rangle(x) = \mathsf{accept}\right] \leq \nu_{\mathsf{snd}}(|x|).$$

Looking ahead, in this paper we may refer to "proofs" for conciseness when making statements that hold for both proofs and arguments.

Proofs of knowledge. Proofs of knowledge are interactive proofs in which the standard soundness requirement is replaced by "knowledge soundness". A protocol (P, V) for a relation $\mathcal{R}$ is knowledge sound, if there is an efficient oracle-aided algorithm K (known as the extractor) which can "extract" a witness from any prover P^* that convinces the verifier to accept with high enough probability. This extraction is done via black-box access to P^*. Concretely, a protocol (P, V, K) is a proof of knowledge if for any prover P^* which convinces V to accept with some probability ϵ, $K^{P^*}(x)$ outputs a witness w such that $(x, w) \in \mathcal{R}$ with a related probability ϵ', where x is shared input.

It is important to clarify what we mean by the fact that K gets black-box access to P^*. K can initialize many copies of P^* on instances and random coins of its choice, and gets oracle access to these copies. It can then interact with each of these copies, playing the role the honest verifier V. Observe that this in particular allows K to rewind P^* by initializing two copies of it with the same random coins, and "fork" messages of V it sends these copies at a certain point of its choice.

Definition 3. Let $\mathcal{R}$ and $\mathcal{L}$ be as in Definition 1. We say that (P, V, K) is an *interactive proof of knowledge* for $\mathcal{R}$, if it satisfies efficiency and correctness as in Definition 1 and the soundness condition is replaced by:

- **Knowledge soundness:** For every algorithm P^* and for every instance x (either in $\mathcal{L}$ or not), let
$$\epsilon_{P^*,x} := \Pr\left[\langle P^*, V\rangle(x) = \mathsf{accept}\right].$$
Then, there exists a polynomial $p(\cdot,\cdot)$ such that for every P^* and x it holds that
$$\Pr\left[(x,w) \in \mathcal{R} \ : \ w \leftarrow\!\$\ K^{P^*}(x)\right] \geq p(\epsilon_{P^*,x}, |x|^{-1}). \tag{2.1}$$

More over, K is a probabilistic algorithm such that for every P^* and x, $K^{P^*}(x)$ runs in expected time polynomial in $|x|$ and $\epsilon^{-1}_{P^*,x}$.

If Eq. 2.1 is only guaranteed to hold for all PPT algorithms P^*, then we say that (P, V, K) satisfies computational knowledge soundness, or is an interactive argument of knowledge.

Note that if P^* convinces the verifier to accept with non-negligible probability, then the probability that K outputs a valid witness is also non-negligible. Moreover, since we do not restrict x to be in $\mathcal{L}$, knowledge soundness implies soundness per Definition 1 and Definition 2.

Zero knowledge. We now define zero knowledge of interactive proofs and arguments. We only consider computational zero knowledge in this paper. Since we show that certain classes of protocols do not satisfy (certain types of) zero knowledge, considering computational zero knowledge makes these lower bounds stronger.

We denote by $(P(w), V^*)(x)$ the random variable that corresponds to the view of V^* in an interaction with P on shared input x and where P runs on private input w. That is, the view consists of x, the transcript of the interaction, and V^*'s random coins.

Definition 4. Let $\mathcal{R}$ and $\mathcal{L}$ be defined as in Definition 1, and let (P, V) be an interactive proof for a language $\mathcal{L}$. We say that (P, V) is *zero knowledge* if for every non-uniform PPT algorithm V^* there exists a probabilistic expected polynomial-time non-uniform algorithm S, called the simulator, such that the following ensembles are indistinguishable:
$$\{(P(w), V^*)(x)\}_{(x,w)\in\mathcal{R}} \quad \text{and} \quad \{S(x)\}_{(x,w)\in\mathcal{R}}$$

The notion is similarly defined for interactive arguments, and interactive proofs and arguments of knowledge.

We now define *black-box* zero knowledge. The definition is obtained from Definition 4 by requiring that there is a single universal simulator S, that is restricted to access V^* only via oracle access. That is, the simulator may initialize many copies of V^* on random coins of its choice, query them on partial transcripts, and receive V^*'s response computed as a function of the random coins and the partial transcript. For a more exhustive discussion, see [GK96].

Definition 5. Let $\mathcal{R}$ and $\mathcal{L}$ be defined as in Definition 1, and let (P, V) be an interactive proof for a language $\mathcal{L}$. We say that (P, V) is *black-box zero knowledge* if there exists a probabilistic expected polynomial-time algorithm S, such that for every non-uniform PPT algorithm V^*, the following ensembles are indistinguishable:

$$\{(P(w), V^*)(x)\}_{(x,w)\in\mathcal{R}} \quad \text{and} \quad \{S^{V^*(\cdot)}(x)\}_{(x,w)\in\mathcal{R}}$$

The above definition ignores various technicalities that have to do with the random coins of V^*. These technicalities will not be important for us since, jumping ahead, we will only consider black-box zero knowledge against deterministic malicious verifiers. For more detail, see for example [GK96].

2.2 The Fiat-Shamir Compiler

The Fiat-Shamir hueristic [FS87] is a way to transform a public coin interactive proof into a non-interactive (i.e., one message) proof. Recall that a proof is said to be public coin if in each round, the verifier just sends its coin flips for this round, and the verifier's final output is a deterministic function of the instance x and the transcript of the protocol. Note that any public-coin interactive proof can be transformed into one in which the final acceptance/rejection decision made by the verifier is a deterministic function of the instance x and the transcrript t of the protocol. If the verifier uses random coins to make its final decision, then it can send them as a final public-coin message in the protocol. Hence, looking forward, when talking about general public coin protocols, we will assume that the final verifier's decision is deterministic. We will overload notation and write $V(\cdot)$ do denote this deterministic decision function. When the instance x is clear from context, we may abbreviate and write $V(t)$ instead of $V(x, t)$.

We define Fiat-Shamir in the random oracle model, in which all algorithms (including the adversary) have access to an oracle $\mathcal{O}$, whose domain and range will become apparent in a second. Before the protocol is executed, $\mathcal{O}$ is sampled uniformly at random from the set of all functions with these domain and range.

For an interactive public coin protocol (P, V) we define a related non-interactive protocol $(P_{\mathsf{FS}}, V_{\mathsf{FS}})$ in the following manner. Say that (P, V) consists of $2k + 1$ messages, where P sends all odd messages, and V sends all even messages. We denote P's messages by $x_1, \ldots, x_{k+1}$ and V's messages by $y_1, \ldots, y_k$. We assume without loss of generality that P sends the last message in the protocol to be compatible with Schnorr's protocol as sketched above. Note that any protocol in which V sends the last message (for example, to enforce that its decision is a deterministic function of the instance and transcript, as discussed above) can be transformed into a protocol in which P sends the last message by letting it send an arbitrary "dummy" message x_{k+1} and having V ignore it when deciding on its output. Then, $(P_{\mathsf{FS}}, V_{\mathsf{FS}})$ on shared input x and private P_{FS}-input w is defined by:

1. P_{FS} first samples randomness r_P for P and computes $x_1 := P(x, w; r)$. It then queries $\mathcal{O}$ on (x, x_1) to obtain y_1 and computes $x_2 := P(x, w, x_1, y_1)$. This process is then repeated until P_{FS} obtains a full transcript of (P, V). That is, for $i = 2, \ldots, k$: P_{FS} queries $\mathcal{O}$ on $(x, x_1, y_1, \ldots, x_i)$ to obtain y_i, and then computes $x_{i+1} \leftarrow\$ P(x, w, x_1, y_1, \ldots, x_i, y_i)$. The final proof outputted by P_{FS} is $\pi = (x_1, y_1, \ldots, x_r, y_r, x_{k+1})$.

2. On input x and $\pi = (x_1, y_1, \ldots, x_k, y_k, x_{k+1})$, the Fiat-Shamir verifier V_{FS} decides on its output as follows: It first verifies that all y_is are computed correctly; that is, that for all $i \in [k]$ it holds that $y_i = \mathcal{O}(x, x_1, y_1, \ldots, x_{i-1})$ (this is done using V_{FS}'s oracle access to $\mathcal{O}$). If for any i this check fails, V_{FS} outputs reject and terminates. Otherwise, V_{FS} invokes $V(\pi)$ and outputs the same.

2.3 Relations, Languages, and Complexity Classes

Let $\mathcal{X}$ and $\mathcal{W}$ be sets and let $\mathcal{R} \subseteq \mathcal{X} \times \mathcal{W}$ be a relation over these sets. We will denote by $\mathcal{L}_{\mathcal{R}} \subset \mathcal{X}$ the set of all $x \in \mathcal{X}$ for which there is a corresponding $w \in \mathcal{W}$ in the relation:

$$\mathcal{L}_{\mathcal{R}} := \{x \in \mathcal{X} \ : \ \exists w \in \mathcal{W} \text{ s.t. } (x, w) \in \mathcal{R}\}.$$

Let $\mathcal{R}$ be a relation. We say that $\mathcal{R}$ is *efficiently searchable*, if there exists a PPT algorithm A such that on every input $x \in \mathcal{L}_{\mathcal{R}}$ in the language induced by $\mathcal{R}$, A finds a witness w such that (x, w) is in $\mathcal{R}$ with non-negligible probability.

Finally, we make use of the standard complexity classes NP and BPP. The reader is referred to [AB09] for formal definitions.

2.4 k-Wise Independent Hash Functions

A function family $\mathcal{H}$ is said to be uniform over a set $\mathcal{S}$ of elements in its domain, if a uniformly-sampled function $h \leftarrow \mathcal{H}$ acts as a truly random function when evaluated on $\mathcal{S}$. A family that is uniform over all subsets of the domain of size at most k are called *k-wise independent*. This is formally captured via the following definition:

Definition 6. Let $\mathcal{X}$ and $\mathcal{Y}$ be sets and let $\mathcal{S} = \{x_1, \ldots, x_k\} \subseteq \mathcal{X}$ be a subset of size k. We say that a function family $\mathcal{H}$ mapping $\mathcal{X}$ to $\mathcal{Y}$ is *uniform over* $\mathcal{S}$ if for every tuple $(y_1, \ldots, y_k) \in \mathcal{Y}^k$ it holds that

$$\Pr_{h \leftarrow \mathcal{H}}[\forall i \in [k] \ : \ h(x_i) = y_i] = \frac{1}{|\mathcal{Y}|^k}.$$

We say that $\mathcal{H}$ is *k-wise independent* if it is uniform over all subsets of $\mathcal{X}$ of size at most k.

For constructions of k-wise independent hash functions, see for example [Jof74, WC81, CG89, NN90].

3 A Lower Bound for Proofs of Knowledge

In this section, we prove a lower bound on the black-box zero knowledge of any knowledge-sound public-coin proof or argument that is compatible with the Fiat-Shamir transform. Namely, we prove that if a proof (or argument) of knowledge (P, V, K) for a relation $\mathcal{R}$ is black-box zero knowledge and is compatible with Fiat-Shamir, then $\mathcal{R}$ must be easily searchable. In particular, this shows that any Σ-protocol for a "hard" relation $\mathcal{R}$ is *not* black-box zero knowledge.[4] This includes Schnorr's protocol [Sch90, Sch91, PS96, BN06] for proving knowledge of discrete log in discrete log hard groups (and later generalizations thereof; see the introduction for examples).

Before presenting the results of this section, we need to define what it means for a proof of knowledge to be Fiat-Shamir-compatible. We first define the knowledge soundness of non-interactive proofs in the random oracle model. A non-interactive proof (P, V, K) is knowledge sound in the random oracle model if Definition 3 holds with the following changes:

- P and V are now oracle-aided algorithms that get access to a random oracle $\mathcal{O}$. Any malicious prover P^* is also an oracle-aided algorithm that expects access to a random oracle.

[4] A Σ-protocol is a 3-round public-coin protocol with special soundness [Cra97] (see also [BS23]).

- The probability that a malicious prover convinces the verifier to accept is now defined also over the choice of random oracle. That is:

$$\epsilon_{P^*,x} := \Pr\left[\left\langle {P^*}^{\mathcal{O}}, V^{\mathcal{O}}\right\rangle(x) = \mathsf{accept}\right],$$

  where the probability is taken over the random coins of P^* and the choice of $\mathcal{O}$.

- As in the standard model, the knowledge extractor K runs the malicious prover P^* as a (black-box) subroutine. When P^* is an oracle-aided algorithm, it is up to K to simulate this oracle to it.

We can now define the Fiat-Shamir-compatibility of proofs of knowledge. Informally, an interactive proof of knowledge is Fiat-Shamir compatible if its Fiat-Shamir compilation is a non-interactive proof of knowledge in the random oracle model.

Definition 7. Let $\mathcal{R}$ be a relation, where $\mathcal{R} \subseteq \mathcal{X} \times \mathcal{W}$ for sets $\mathcal{X}$ and $\mathcal{W}$. Let (P, V, K) be a public-coin interactive proof of knowledge for $\mathcal{R}$, and let $(P_{\mathsf{FS}}, V_{\mathsf{FS}})$ be the non-interactive proof for $\mathcal{R}$ obtained from (P, V) via the Fiat-Shamir transform. Then, we say that (P, V) is *Fiat-Shamir compatible* if there exists an algorithm K_{FS} such that $(P_{\mathsf{FS}}, V_{\mathsf{FS}}, K_{\mathsf{FS}})$ is an argument of knowledge in the random oacle model.

Equipped with Definition 7, we state the main theorem of the section.

Theorem 5. *Let $\mathcal{R}$ be a relation, where $\mathcal{R} \subseteq \mathcal{X} \times \mathcal{W}$ for sets $\mathcal{X}$ and $\mathcal{W}$, and let (P, V, K) be a public-coin interactive proof of knowledge for $\mathcal{R}$. Suppose that (P, V, K) is black-box zero knowledge and Fiat-Shamir compatible. Then, $\mathcal{R}$ is efficienlty searchable.*

Before proving the theorem, we make two remarks:

1. We will only handle the case where the simulator S guaranteed by the black-box zero knowledge is strictly polynomial time. If S runs in expected polynomial-time we can make it run in strictly polynomial time using a standard execution truncation argument. See [GK96] for more detail.

2. As in [GK96], we will only rely on the fact that S needs to be a simulator for deterministic non-uniform verifiers. Note that this only makes the result stronger, as we are ruling out even a simulator that only works for determinstic verifiers.

We now prove Theorem 5.

Proof. Let $\mathcal{R}$ and (P, V, K) be as in the statement of the theorem. We will construct an algorithm A that efficiently searches the relation $\mathcal{R}$.

By assumption, (P, V, K) is black-box zero knowledge. Let S be the black-box simulator guaranteed by Definition 4. We further assumed that (P, V, K) is Fiat-Shamir compatible, and therefore, there exists an extractor K_{FS} for the Fiat-Shamir compiled variant of the protocol. With these algorithms at hand, we can define the algorithm A. On input $x \in \mathcal{X}$, A does:

1. Invoke the extractor K_{FS} on input x.

2. K_{FS} expects black-box access to a prover P^* for the non-interactive Fiat-Shamir-compiled protocol $(P_{\mathsf{FS}}, V_{\mathsf{FS}})$. A simulates this access as follows:

   (a) K_{FS} can initiate many copies of P^*. When it initiates the ith copy of P^* with input x_i random coins rand_i, K_{FS} invokes $S(x_i; \mathsf{rand}_i)$. We call this invocation the ith copy of S.

(b) For each i, the ith copy of S expects oracle access to a malicious verifier V_i. To simulate this access, whenever the ith copy of S issues a query (a partial transcript) pt to V_i, A forwards pt to K_{FS} as a query made by the ith copy of P^* to the random oracle $\mathcal{O}$. If and when K_{FS} replies with a response c to this query, A forwards c to S as the response by V_i.

(c) Whenever a copy of S outputs a transcript t, A forwards this transcript as a proof outputted by the corresponding copy of P^*.

3. Finally, K_{FS} outputs a witness $w \in \mathcal{W}$. A outputs the same w.

We now turn to bound the success probability of A in finding a witness w such that $(x, w) \in \mathcal{R}$. For ease of notation, we assume that the (honest) verifier's messages in all rounds of the protocol are uniformly random in $\{0,1\}^\ell$ for $\ell = \ell(|x|)$ (the proof readily extends to the general case where each round specifies a different message space). For a function $f : \{0,1\}^* \to \{0,1\}^\ell$ that maps partial transcripts of the protocol (P, V) to verifier messages, we define the "malicious" deterministic verifier V_f that simply decides on its next message by invoking f on the transcript of the execution thus far. Let $\epsilon_{S,f,x}$ denote the probability that on input x, S outputs an accepting transcript when given black-box access to V_f. That is,

$$\epsilon_{S,f,x} := \Pr\left[V(S^{V_f(\cdot)}(x)) = \mathsf{accept}\right]$$

where the probability is taken over the random coins of S. Let $\epsilon_{S,x}$ denote the probability that on input x, S outputs an accepting transcript when given black-box access to V_f for a uniformly random f. That is,

$$\epsilon_{S,x} := \Pr\left[V(S^{V_f(\cdot)}(x)) = \mathsf{accept}\right]$$

where the probability is taken over the random coins of S *and* the random choice of f.

Observe that the distribution of oracle answers induced by the random choice of f is identical to that induced by a random oracle. Hence, by definition of Fiat-Shamir compatibility, there exists a polynomial p such that for every $x \in \mathcal{L}_{\mathcal{R}}$ it holds that

$$\Pr\left[(x, A(x)) \in \mathcal{R}\right] \geq p(\epsilon_{S,x}, |x|^{-1}).$$

The following lemma relates $\epsilon_{S,x}$ to the zero-knowledge of (P, V). The lemma is proven at the end of the section.

Lemma 1. *There exists a negligible function $\nu_{\mathsf{zk}}(\cdot)$ such that for every $x \in \mathcal{L}_{\mathcal{R}}$, it holds that $\epsilon_{S,x} \geq 1 - \nu_{\mathsf{zk}}(|x|)$.*

Therefore, A is an expected polynomial time algorithm that finds a witness w for x with non-negligible probability. By a standard execution trancation argument, A can be made strictly polynomial time with a polynomial loss in its success probability. This implies that $\mathcal{R}$ is efficiently searchable, concluding the proof of the theorem. □

We conclude by proving Lemma 1.[5]

Proof of Lemma 1. Let $x \in \mathcal{L}_{\mathcal{R}}$, let $q = q(|x|)$ be a bound on the number of queries issued by S to the verifier on input x. Moreover, let $c = c(|x|)$ be a bound on the total length of the messages exchanged in (P, V) on input x. We wish to argue that the transcript t outputted by S, when executed by A as a sub-routine, is accepting for x with probability at least $1 - \nu_{\mathsf{zk}}(|x|)$ for some negligible ν_{zk}.

[5]The lemma extends part 2 of Lemma 6.3 from [GK96].

To this end, we define a family of "malicious" verifiers $\{V_h\}_{h\in\mathcal{H}_n}$, where $n = |x|$ and $\mathcal{H}_n$ is a family of q-wise independent hash functions, mapping inputs in $\{0,1\}^{\leq |x|+c(|x|)}$ to outputs in $\{0,1\}^{\ell(n)}$. The verifier V_h is deterministic and decides on its next message by applying h to the input x and the interaction thus far. That is, let $\tilde{x}$ be the instance on which (P, V_h) is invoked, and let $\tilde{x_1}, \tilde{y}_1, \ldots, \tilde{x}_i$ be the partial transcript of the execution so far. Then, V_h's ith message is $\tilde{y}_i = h(\tilde{x}, \tilde{x_1}, \tilde{y}_1, \ldots, \tilde{x}_i)$.

For $(x, w) \in \mathcal{R}$ and a hash function $h \in \mathcal{H}_n$, let $\epsilon_{h,x,w}$ denote the probability, defined over the random coins of P, that an interaction between P and V_h is accepting:

$$\epsilon_{h,x,w} := \Pr\left[\langle P(w), V_h\rangle(x) = \mathsf{accept}\right].$$

By zero knowledge, there exists a negligible function $\nu_{zk}(\cdot)$ such that for every $(x, w) \in \mathcal{R}$ and $h \in \mathcal{H}_n$, it holds that

$$\epsilon_{S,h,x} \geq \epsilon_{h,x,w} - \nu_{\mathsf{zk}}(|x|). \tag{3.1}$$

For every $x \in \mathcal{L}_{\mathcal{R}}$, let $w^*(x)$ denote some canonical witness for x (e.g., the first witness in some lexicographic order). Since $\mathcal{H}_n$ is q-wise independent, it holds that

$$\begin{aligned}
\epsilon_{S,x} &= \mathbb{E}_{h\leftarrow\$\mathcal{H}_n}\left[\epsilon_{S,h,x}\right] && (3.2)\\
&\geq \mathbb{E}_{h\leftarrow\$\mathcal{H}_n}\left[\epsilon_{h,x,w^*(x)} - \nu_{\mathsf{zk}}(|x|)\right] && (3.3)\\
&= \mathbb{E}_{h\leftarrow\$\mathcal{H}_n}\left[\epsilon_{h,x,w^*(x)}\right] - \nu_{\mathsf{zk}}(|x|). && (3.4)
\end{aligned}$$

Finally, by q-wise independence of $\mathcal{H}_n$, the expectation $\mathbb{E}_{h\leftarrow\$\mathcal{H}_n}\left[\epsilon_{h,x,w^*(x)}\right]$ describes the probability that V accepts in an honest execution of (P, V) when the shared instance is x and P gets the additional input $w^*(x)$. By completeness of (P, V), it holds that

$$\mathbb{E}_{h\leftarrow\$\mathcal{H}_n}\left[\epsilon_{h,x,w^*(x)}\right] \geq 1 - \nu_{\mathsf{cmp}}(|x|) \tag{3.5}$$

for some negligible function $\nu_{\mathsf{cmp}}(\cdot)$. Denoting $\nu' = \nu_{\mathsf{zk}} + \nu_{\mathsf{cmp}}$, we obtained that

$$\epsilon_{S,x} \geq 1 - \nu(|x|).$$

Noting that ν is a negligible function concludes the proof of the lemma. □

4 A Lower Bound for Interactive Proofs and Arguments

In this section we generalize the lower bound of Goldreich and Krawczyk [GK96] for interactive proofs and arguments. Namely, we show that if a language $\mathcal{L}$ has *any* interactive argument that is simultaneously black-box zero knowledge and compatible with Fiat-Shamir in the random oracle model, then $\mathcal{L} \in \mathsf{BPP}$. This includes, in particular, the case of constant-round public coin arguments considered by Goldreich and Krawczyk. First, we need to define what we mean when we say an interactive proof is compatible with Fiat-Shamir.

Definition 8. Let (P, V) be an interactive proof for a language $\mathcal{L}$, and let $(P_{\mathsf{FS}}, V_{\mathsf{FS}})$ be the non-interactive proof for $\mathcal{L}$ obtained from (P, V) via the Fiat-Shamir transform. Then, we say that (P, V) is *Fiat-Shamir compatible* if $(P_{\mathsf{FS}}, V_{\mathsf{FS}})$ is computationally sound, when $\mathcal{O}$ is modeled as a random oracle.

We now state the main theorem of this section.

Theorem 6. *Let $\mathcal{L}$ be a language and let (P, V) be an interactive proof for $\mathcal{L}$. Suppose that (P, V) is black-box zero knowledge and Fiat-Shamir compatible. Then, $\mathcal{L} \in \mathsf{BPP}$.*

As in Section 3, we may assume that the simulator S guaranteed by the black-box zero knowledge property runs in strict polynomial time.

Proof. Suppose that (P, V) is black-box zero knowledge, and let S be the universal simulator guaranteed by this assumption. Consider the following algorithm A for deciding the language $\mathcal{L}$. On input $x \in \{0,1\}^n$, A does:

1. Invoke $S(x)$, and for every query $(x', x_1, y_1, \ldots, x_i)$ issued by S to its oracle verifier, reply with a uniformly random $y_i \leftarrow\!\!\$\ \{0,1\}^{\ell(n)}$. Eventually S outputs a transcript $t = (\tilde{x}, \tilde{x_1}, \tilde{y_1}, \ldots, \tilde{x_k}, \tilde{y_k}, \tilde{x_{k+1}})$.

2. Invoke the (honest) verifier V on input t and output the same. That is, output $V(x, t)$.

The following two lemmata establish the correctness of A in deciding $\mathcal{L}$.

Lemma 2. *There exists a negligible function $\nu(\cdot)$ such that for every $x \notin \mathcal{L}$, $A(x)$ outputs 1 with probability at most $\nu(|x|)$.*

Proof. We wish to argue that the transcript t outputted by S is accepting for x with probability at most $\nu(|x|)$ for a negligible $\nu(\cdot)$, and the claim will follow by the definition of A.

Suppose towards contradiction that this is not the case. That is, there exists a polynomial $p(\cdot)$ such that the probability of $V(t) = \mathsf{accept}$ is greater than $1/p(|x|)$. In this case, we break the soundness of the non-interactive Fiat-Shamir variant of (P, V), denoted $(P_{\mathsf{FS}}, V_{\mathsf{FS}})$. To this end, we construct a malicious prover P^*_{FS} convincing V_{FS} to accept with non-negligible probability. On input $x \notin \mathcal{L}$ and given oracle access to a random oracle $\mathcal{O}$, P^*_{FS} invokes the simulator $S(x)$ and simulates to it oracle access to the verifier V^*: Whenever S queries V^* on $(\tilde{x}, \tilde{x}_1, \tilde{y}_1, \ldots, \tilde{x}_i)$, P^* queries $\mathcal{O}$ on $(\tilde{x}, \tilde{x}_1, \tilde{y}_1, \ldots, \tilde{x}_i)$ and replies to S with the response $\tilde{y}_i$ it receives from $\mathcal{O}$. Finally, when S outputs a transcript t, P^*_{FS} outputs the proof $\pi = t$.

We claim that π is accepted by V_{FS} with probability at least $1/p(|x|)$. Observe that by the definition of $\mathcal{O}$, the replies of P^*_{FS} to S's queries to the malicious verifier are uniformly random and independent. Hence, they are distributed the same as the replies of A to S's queries. Note that the view of S is determined by x, its random coins, and the responses to its queries. Hence, the view of S is distributed as its view in its invocation by A. It follows that t is accepting with probability at least $1/p(|x|)$. This means that P^*_{FS} breaks the soundness of $(P_{\mathsf{FS}}, V_{\mathsf{FS}})$, in contradiction to the Fiat-Shamir compatibility of (P, V). This concludes the proof of the claim. □

Lemma 3. *There exists a negligible function $\nu'(\cdot)$ such that for every $x \in \mathcal{L}$, $A(x)$ outputs 1 with probability at least $1 - \nu'(|x|)$.*

The proof of Lemma 3 is identical to that of Lemma 1. Taken together, the two lemmata show that A successfully decides $\mathcal{L}$ with overwhelming probability, implying that $\mathcal{L} \in \mathsf{BPP}$. □

5 A Lower Bound for Generic Zero Knowledge

In this section, we prove the nonexistence of a "generic" simulator for Schnorr's protocol. We start by defining generic simulators, and then prove that such simulators cannot exist for Schnorr's protocol in groups in which discrete log is hard.

5.1 Generic Simulators

We first recall the generic group model (GGM), introduced by Shoup [Sho97], and focus on prime-order groups for simplicity. The GGM is an idealized model in which group elements are represented by "opaque" random bit-strings. Intuitively, the GGM captures group algorithms that work in any group, and do not exploit the underlying representation of group elements in a specific group.

Let $p \in \mathbb{N}$ be a prime and let $\lambda = \lceil \log p \rceil$. Then, a generic group of order p is induced by a random injection σ, mapping integers in $\mathbb{Z}_p$ to bit-strings in $\{0,1\}^{2\lambda}$. A group element is an element in the image of σ. The group operation is induced by the inverse (partial) function σ^{-1} and the group operation of the additive group $\mathbb{Z}_p$. That is, if we denote the group operation by $\circ$, then for every $\ell_1, \ell_2 \in Im(\sigma)$, we have

$$\ell_1 \circ \ell_2 := \sigma(\sigma^{-1}(\ell_1) + \sigma^{-1}(\ell_2)),$$

where addition is in $\mathbb{Z}_p$. For $\ell \in Im(\sigma)$ and $c \in \mathbb{Z}$, we may write ℓ^c to denote

$$\underbrace{\ell \circ \ell \circ \cdots \circ \ell}_{c \text{ times}}$$

Algorithms in the GGM do not have an explicit description of σ. Rather, they access the group operation via an oracle $\mathcal{O}_\sigma$. On input two labels ℓ_1 and ℓ_2 in $\{0,1\}^{2\lambda}$, $\mathcal{O}_\sigma$ replies with $\ell_1 \circ \ell_2$ if $\ell_1, \ell_2 \in Im(\sigma)$ and with $\perp$ otherwise. We will always assume that all algorithms receive the generator $\sigma(1)$ as input, and may not mention this explicitly. In particular, they can compute $\sigma(a)$ for every $a \in \mathbb{Z}_p$ in $O(\log p)$ queries to the group oracle.

Algebraic representation of labels. As many proofs in the GGM, we will rely on the fact that by observing the oracle queries of a generic algorithm A, generic group elements (labels) that A computes can be expressed as a "linear-in-the-exponent" combination of its input elements. Concretely, let $(\ell_1, \ldots, \ell_k)$ denote the input elements to a generic algorithm A, where every $\ell_i \in \{0,1\}^{2\lambda}$. Throughout the execution of A, we associate every label ℓ in $\{0,1\}^{2\lambda}$ with a set of vectors in $\mathbb{Z}_p^k$. We call this set the *representation* of ℓ and denote it by $\mathsf{rep}(\ell)$. Initially, for every $i \in [k]$ we define $\mathsf{rep}(\ell_i) = \{\vec{e_i}\}$, where $\vec{e_i} \in \mathbb{Z}_p^k$ is the vector whose ith entry is 1 and all of its other entries are 0. For every $\ell \in \{0,1\}^{2\lambda}$ not given as input to A, $\mathsf{rep}(\ell)$ is initialized as empty $\mathsf{rep}(\ell) = \emptyset$.

Suppose A issues a query (ℓ_1, ℓ_2) to $\mathcal{O}_\sigma$ and the oracle replies with ℓ_3. Then, we update $\mathsf{rep}(\ell_3)$ as follows: if $\mathsf{rep}(\ell_1)$ or $\mathsf{rep}(\ell_2)$ are $\emptyset$, we do nothing. Otherwise, we add to $\mathsf{rep}(\ell_3)$ all vectors in in $\mathsf{rep}(\ell_1) + \mathsf{rep}(\ell_2)$, where the sum of two sets $S_1 + S_2$ is defined as $\{s_1 + s_2 \ : \ s_1 \in S_1, \ s_2 \in S_2\}$.

Generic protocols and generic simulators. Equipped with the above definition, we can define interactive proofs in the GGM. Such proofs are defined similarly to standard-model interactive proofs (recall Section 2), but now the prover P and the verifier V are oracle-aided algorithms with access to the group oracle $\mathcal{O}_\sigma$. Completeness, soundness, and knowledge soundness can be easily extended to the generic setting as well, but we will not need these extensions to state and prove our result in this section. We will refer to interactive proofs in the GGM as *generic interactive proofs*.

Note that in the GGM, it makes sense to define relations relative to the group oracle. The discrete log relation with respect to the group oracle, defined by $\mathcal{R}_{p,\sigma} = \left\{(h, x) \in \{0,1\}^{2\lambda} \times \mathbb{Z}_p \ : \ \sigma(x) = h\right\}$, is perhaps the simplest example of such a relation. We can consider generic interactive proofs with respect to such oracle-dependent relations, and Schnorr's protocol (which we will shortly define) is an example of that.

With the notion of generic interactive proofs at hand, we can define what we mean when we say that such a protocol has a generic simulator. Loosely, a generic interactive proof

is generic zero-knowledge if for every malicious verifier $(V^*)^{\mathcal{O}_\sigma}$, there is a corresponding generic simulator $(S_{V^*})^{\mathcal{O}_\sigma}$.

Definition 9. Let $(P^{\mathcal{O}_\sigma}, V^{\mathcal{O}_\sigma})$ be an interactive proof for a relation $\mathcal{R}^{\mathcal{O}_\sigma}$. We say that $(P^{\mathcal{O}_\sigma}, V^{\mathcal{O}_\sigma})$ satisfies *generic zero knowledge* if for every PPT oracle-aided algorithm $(V^*)^{\mathcal{O}_\sigma}$ there exists a probabilistic expected polynomial-time oracle-aided algorithm $S^{\mathcal{O}_\sigma}$, called the simulator, such that the following ensembles are indistinguishable for any PPT generic algorithm:

$$\{(P^{\mathcal{O}_\sigma}(w), (V^*)^{\mathcal{O}_\sigma})(x)\}_{(x,w)\in\mathcal{R}^{\mathcal{O}_\sigma}} \quad \text{and} \quad \{S^{\mathcal{O}_\sigma}(x)\}_{(x,w)\in\mathcal{R}^{\mathcal{O}_\sigma}}$$

Two remarks are in order:

1. The distribution ensembles in Definition 9 are implicitly defined over a sequence of oracles in the following way. We write $\mathcal{O}_\sigma$ to refer to a sequence of oracles $\{\mathcal{O}_{\sigma_p}\}_{p\in\mathcal{S}}$, where $\mathcal{S}$ is an infinite sequence of primes $p_1, p_2, \ldots$ and for each i it holds that $p_{i+1} > p_i$. This induces a sequence of relations $\mathcal{R}^{\mathcal{O}_\sigma} = \{\mathcal{R}^{\mathcal{O}_{\sigma_p}}\}_{p\in\mathcal{S}}$. Looking ahead, we can restrict the discussion to relations in which both the instance size and witness size (in bits) are polynomially-related to $\log p \approx \lambda$. Hence, when we will say "polynomial time" there will be no ambiguity: this will be both in terms of instance and witness (which will constitute the inputs to the algorithms we will consider) and in terms of $\log p$.

2. We rely on the convention that the running time of an oracle-aided algorithm serves as an upper bound on the number of oracle queries it issues. Hence, a PPT generic algorithm makes at most a polynomial number of queries to the group oracle.

5.2 Schnorr Has No Generic Simulators

Schnorr's protocol. We briefly recall Schnorr's proof of knowledge of discrete log. We present the protocol as a generic protocol. In this setting, the prover P and the verifier V share a group element $X \in \{0,1\}^{2\lambda}$. P additionally holds an integer $x \in \mathbb{Z}_p$ such that $\sigma(x) = X$. The protocol proceeds as follows:

1. Round 1: P samples a uniformly random $a \leftarrow\!\!\$\ \mathbb{Z}_p$, computes $A \leftarrow \sigma(a)$ via the oracle $\mathcal{O}_\sigma$ and sends A to V.
2. Round 2: V samples a uniformly random $c \leftarrow\!\!\$\ \mathbb{Z}_p$ and sends it to P.
3. Round 3: P computes $\gamma \leftarrow c \cdot x + a$ and sends it to V.
4. Final verification: V checks that $\sigma(\gamma) = X^c \circ A$, where σ and $\circ$ are computed via the oracle $\mathcal{O}_\sigma$. If this check passes, V outputs accept and otherwise, it outputs reject.

Theorem 7. *Schnorr's protocol does not satisfy generic zero knowledge.*

To establish Theorem 7, we prove a lemma that shows that if Schnorr's protocol does have a generic simulator S_{V^*} for every malicious verifier V^*, then there is a generic algorithm that computes discrete logs in the GGM, with the same number of queries as on of these simulators. Shoup's lower bound on the hardness of discrete log in the GGM [Sho97] then implies that every simulator must make at least $\Omega(\sqrt{p})$ queries, and hence cannot be polynomial in $\log p$. We now turn to state and prove our main lemma.

Lemma 4. *Suppose that Schnorr's protocol is generic zero knowledge. Then, there exists a PPT generic verifier V^* such that the following holds. For every generic simulator S for V^*, there exists a discrete log algorithm B such that*

$$\Pr\left[B(\sigma(1), \sigma(x)) = x\right] \geq 1 - \frac{2Q+3}{p-Q} - \epsilon,$$

where $\epsilon = \epsilon(\log p)$ is a negligible function in $\log p$, $Q = Q(p)$ is a bound on the number of GGM queries issued by S, and the probability is taken over the random coins of B and the choice of σ. Moreover, B makes at most Q queries to the GGM oracle.

Proof. Consider the following "malicious" verifier V^*:

1. On instance $X \in \{0,1\}^{2\lambda}$ and first prover message $A \in \{0,1\}^{2\lambda}$, V^* decides on the challenge c by parsing the first λ bits of A as an element of $\mathbb{Z}_p$. This element is the challenge c sent by V^*.

2. Upon receiving the second prover message γ, V^* uses the honest verifier decision rule to determine whether to accept. That is, it accepts if and only if $\sigma(\gamma) = X^c \circ A$.

Now suppose that there exists a generic simulator S for V^*. We claim that such a simulator can be used to compute the discrete log of X with overwhelming probability, and so the number of group operation queries it issues must be exponential in $\log p$.

We construct a generic algorithm B, that on input $(\sigma(1), X = \sigma(x))$ computes x with the same number of queries as S. On input X, B is defined by:

1. Invoke S on input $(\sigma(1), X)$.

2. Whenever S issues a query (ℓ_1, ℓ_2), forward the query to $\mathcal{O}_\sigma$. Let ℓ_3 be the response of the oracle. Forward ℓ_3 as the response to S. If $|\mathsf{rep}(\ell_3)| > 1$, denote by (β, α) and (β', α') be the representations of ℓ_3. Output $x' = (\beta - \beta') \cdot (\alpha' - \alpha)^{-1}$ and terminate. Note that it cannot be the case that $(\alpha, \beta) \neq (\alpha', \beta')$ and $\alpha = \alpha'$.

3. If at any point, S issues a query (ℓ_1, ℓ_2) issued by S such that $\mathcal{O}_\sigma(\ell_1, \ell_2) \neq \perp$ but $\mathsf{rep}(\ell_1) = \emptyset$ or $\mathsf{rep}(\ell_2) = \emptyset$, then abort.
(A aborts if S "gussed" valid labels ℓ_1 and ℓ_2 in the image of σ without obtaining them as inputs/oracle replies).

4. Finally, S outputs a transcript (A', c', γ'). Let (β, α) be the unique representation in $\mathsf{rep}(A')$. Observer that if this step is reached, it must be the case that $\mathsf{rep}(A')$ contains exactly one representation.

5. Abort if any of the following conditions do not hold:

   (a) The transcript (A', c', γ') is accepting.

   (b) c' is the $\mathbb{Z}_p$ element obtained by parsing the first λ bits of A' as an element of $\mathbb{Z}_p$.

   (c) $\alpha + c' \neq 0$.

6. If reached, output $x' = (\gamma' - \beta) \cdot (\alpha + c')^{-1} \in \mathbb{Z}_p$.

We now analyze the success probability of B. If B terminates in Step 2, then it necessarily finds the correct $x' = x$. This is because both (β, α) and (β', α') are representations of the same group element, which implies that $\alpha x + \beta = \alpha' x + \beta'$ and hence $x' = x$. Therefore, we condition the rest of the analysis on the event in which B does not terminate in Step 2.

Note that whenever B does not abort (in either Step 3 or Step 5), it successfully finds $\sigma^{-1}(X)$. This is because (A', c', γ') is an accepting transcript, which means that $A' \circ X^{c'} = \sigma(\gamma')$. In turn, this implies that $\alpha x + \beta + c' x = \gamma'$, and the fact that $x' = x$ follows.

We are left with bounding the probability that B does not abort. Let $\mathsf{E}_{\mathsf{guess}}$ denote the event in which B aborts in Step 3. Since the image of σ is of size less than 2^λ but the label space is of size $2^{2\lambda}$, the probability for $\mathsf{E}_{\mathsf{guess}}$ is at most $Q \cdot 2^{-\lambda} \leq Q/p$. The rest of the analysis is conditioned on $\neg\mathsf{E}_{\mathsf{guess}}$.

For $i \in \{a, b, c\}$, let E_i denote the event in which B aborts in Step 5 due to condition (i). First, note that when interacting with an honest prover, V^* always accepts. Moreover, by definition of V^*, it is always the case that the challenge c is the first λ bits of the first prover message A (interpreted as a $\mathbb{Z}_p$ element). Hence, by zero knowledge, the probability that the event $\mathsf{E}_a \vee \mathsf{E}_b$ occurs is at most $\epsilon(\lambda)$ for some negligible function ϵ.

The event E_c can happen in one of four ways:

1. The first λ bits of $X = \sigma(x)$ encode the integer $1 \in \mathbb{Z}_p$. This happens with probability $1/p$.

2. The first λ bits of $\sigma(1)$ encode the integer $0 \in \mathbb{Z}_p$. This happens with probability $1/p$.

3. S issues a query ℓ_1, ℓ_2 such that $\mathsf{rep}(\ell_1) = (\beta_1, \alpha_1)$, $\mathsf{rep}(\ell_2) = (\beta_2, \alpha_2)$, $\alpha_1 + \alpha_2 = 0 \in \mathbb{Z}_p$, and the first λ bits of $\mathcal{O}_\sigma(\ell_1, \ell_2)$ encode $0 \in \mathbb{Z}_p$. Since S issues at most Q queries, and the label of each response encodes $0 \in \mathbb{Z}_p$ with probability $1/p$, the total probability that this event happens is at most Q/p.

4. S issues a query ℓ_1, ℓ_2 such that $\mathsf{rep}(\ell_1) = (\beta_1, \alpha_1)$, $\mathsf{rep}(\ell_2) = (\beta_2, \alpha_2)$ and the first λ bits of $\mathcal{O}_\sigma(\ell_1, \ell_2)$ encode the $\mathbb{Z}_p$ element $\alpha_1 + \alpha_2$. Note that α_1 are well defined α_2 for all queries issued by S, since we are conditioning the analysis on $\neg\mathsf{E}_{\mathsf{guess}}$ (and hence $\mathsf{rep}(\ell_1) \neq \emptyset$ and $\mathsf{rep}(\ell_2) \neq \emptyset$) and on B not terminating in Step 2 (and hence $|\mathsf{rep}(\ell_1)| \leq 1$ and $|\mathsf{rep}(\ell_2)| \leq 1$).
   The probability for this event is bounded by $Q/(p - Q)$.

Overall, the probability that E_c occurs is less than $(2Q + 2)/(p - Q)$.

Putting everything together, we obtain that the probability that

$$\Pr\left[B(\sigma(1), \sigma(x)) \neq x\right] \quad \leq \quad \frac{2Q + 3}{p - Q} + \epsilon$$

concluding the proof of the lemma. □

Acknowledgments

This work partially supported by the Simons Foundation and a research grant from Protocol Labs.

6 Bibliography

References

[AB09] Sanjeev Arora and Boaz Barak. *Computational complexity: a modern approach*. Cambridge University Press, 2009.

[AFK22] Thomas Attema, Serge Fehr, and Michael Klooß. Fiat-shamir transformation of multi-round interactive proofs. In Eike Kiltz and Vinod Vaikuntanathan, editors, *TCC 2022, Part I*, volume 13747 of *LNCS*, pages 113–142. Springer, Cham, November 2022. doi:10.1007/978-3-031-22318-1_5.

[AFK23] Thomas Attema, Serge Fehr, and Michael Klooß. Fiat-shamir transformation of multi-round interactive proofs (extended version). *Journal of Cryptology*, 36(4):36, October 2023. doi:10.1007/s00145-023-09478-y.

[BG93] Mihir Bellare and Oded Goldreich. On defining proofs of knowledge. In Ernest F. Brickell, editor, *CRYPTO'92*, volume 740 of *LNCS*, pages 390–420. Springer, Berlin, Heidelberg, August 1993. doi:10.1007/3-540-48071-4_28.

[BN06] Mihir Bellare and Gregory Neven. Multi-signatures in the plain public-key model and a general forking lemma. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *ACM CCS 2006*, pages 390–399. ACM Press, October / November 2006. doi:10.1145/1180405.1180453.

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In Dorothy E. Denning, Raymond Pyle, Ravi Ganesan, Ravi S. Sandhu, and Victoria Ashby, editors, *ACM CCS 93*, pages 62–73. ACM Press, November 1993. doi:10.1145/168588.168596.

[BS23] Dan Boneh and Victor Shoup. A graduate course in applied cryptography. *Version 0.6*, 2023. URL: https://toc.cryptobook.us/.

[CCH+18] Ran Canetti, Yilei Chen, Justin Holmgren, Alex Lombardi, Guy N. Rothblum, and Ron D. Rothblum. Fiat-shamir from simpler assumptions. Cryptology ePrint Archive, Paper 2018/1004, 2018. https://eprint.iacr.org/2018/1004. URL: https://eprint.iacr.org/2018/1004.

[CCH+19] Ran Canetti, Yilei Chen, Justin Holmgren, Alex Lombardi, Guy N. Rothblum, Ron D. Rothblum, and Daniel Wichs. Fiat-shamir: from practice to theory. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2019, page 1082–1090, New York, NY, USA, 2019. Association for Computing Machinery. doi:10.1145/3313276.3316380.

[CG89] Benny Chor and Oded Goldreich. On the power of two-point based sampling. *Journal of Complexity*, 5(1):96–106, 1989. doi:10.1016/0885-064X(89)90015-0.

[CLMQ21] Yilei Chen, Alex Lombardi, Fermi Ma, and Willy Quach. Does fiat-shamir require a cryptographic hash function? In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part IV*, volume 12828 of *LNCS*, pages 334–363, Virtual Event, August 2021. Springer, Cham. doi:10.1007/978-3-030-84259-8_12.

[CLW18] Ran Canetti, Alex Lombardi, and Daniel Wichs. Fiat-shamir: From practice to theory, part ii (nizk and correlation intractability from circular-secure fhe). Cryptology ePrint Archive, Paper 2018/1248, 2018. https://eprint.iacr.org/2018/1248. URL: https://eprint.iacr.org/2018/1248.

[CP93] David Chaum and Torben P. Pedersen. Wallet databases with observers. In Ernest F. Brickell, editor, *CRYPTO'92*, volume 740 of *LNCS*, pages 89–105. Springer, Berlin, Heidelberg, August 1993. doi:10.1007/3-540-48071-4_7.

[Cra97] Ronald Cramer. *Modular Design of Secure yet Practical Cryptographic Protocols*. PhD thesis, Universiteit van Amsterdam, 1997.

[CS97] Jan Camenisch and Markus Stadler. Proof systems for general statements about discrete logarithms. *Technical Report/ETH Zurich, Department of Computer Science*, 260, 1997.

[Dam02] Ivan Damgård. On Σ-protocols. *Lecture Notes, University of Aarhus, Department for Computer Science*, 84, 2002. URL: https://www.cs.au.dk/~ivan/Sigma.pdf.

[DNRS03] Cynthia Dwork, Moni Naor, Omer Reingold, and Larry Stockmeyer. Magic functions: In memoriam: Bernard m. dwork 1923–1998. *Journal of the ACM (JACM)*, 50(6):852–921, 2003. doi:10.1145/950620.95062.

[FS87] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *CRYPTO'86*, volume 263 of *LNCS*, pages 186–194. Springer, Berlin, Heidelberg, August 1987. doi:10.1007/3-540-47721-7_12.

[GK96] Oded Goldreich and Hugo Krawczyk. On the composition of zero-knowledge proof systems. *SIAM Journal on Computing*, 25(1):169–192, 1996. doi:10.1137/S0097539791220688.

[GMR85] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof-systems (extended abstract). In *17th ACM STOC*, pages 291–304. ACM Press, May 1985. doi:10.1145/22145.22178.

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to prove all NP-statements in zero-knowledge, and a methodology of cryptographic protocol design. In Andrew M. Odlyzko, editor, *CRYPTO'86*, volume 263 of *LNCS*, pages 171–185. Springer, Berlin, Heidelberg, August 1987. doi:10.1007/3-540-47721-7_11.

[GO94] Oded Goldreich and Yair Oren. Definitions and properties of zero-knowledge proof systems. *Journal of Cryptology*, 7(1):1–32, December 1994. doi:10.1007/BF00195207.

[Gol01] Oded Goldreich. *Foundations of cryptography: volume 1, basic tools*, volume 1. Cambridge university press, 2001.

[GQ90] Louis C. Guillou and Jean-Jacques Quisquater. A "paradoxical" indentity-based signature scheme resulting from zero-knowledge. In Shafi Goldwasser, editor, *CRYPTO'88*, volume 403 of *LNCS*, pages 216–231. Springer, New York, August 1990. doi:10.1007/0-387-34799-2_16.

[HLR21] Justin Holmgren, Alex Lombardi, and Ron D. Rothblum. Fiat-Shamir via list-recoverable codes (or: parallel repetition of GMW is not zero-knowledge). In Samir Khuller and Virginia Vassilevska Williams, editors, *53rd ACM STOC*, pages 750–760. ACM Press, June 2021. doi:10.1145/3406325.3451116.

[Hol19] Justin Holmgren. On round-by-round soundness and state restoration attacks. Cryptology ePrint Archive, Paper 2019/1261, 2019. https://eprint.iacr.org/2019/1261. URL: https://eprint.iacr.org/2019/1261.

[Jof74] Anatole Joffe. On a set of almost deterministic k-independent random variables. *the Annals of Probability*, 2(1):161–162, 1974.

[Mau15] Ueli Maurer. Zero-knowledge proofs of knowledge for group homomorphisms. *DCC*, 77(2-3):663–676, 2015. doi:10.1007/s10623-015-0103-5.

[NN90] Joseph Naor and Moni Naor. Small-bias probability spaces: Efficient constructions and applications. In *22nd ACM STOC*, pages 213–223. ACM Press, May 1990. doi:10.1145/100216.100244.

[Oka93] Tatsuaki Okamoto. Provably secure and practical identification schemes and corresponding signature schemes. In Ernest F. Brickell, editor, *CRYPTO'92*, volume 740 of *LNCS*, pages 31–53. Springer, Berlin, Heidelberg, August 1993. doi:10.1007/3-540-48071-4_3.

[PS96] David Pointcheval and Jacques Stern. Security proofs for signature schemes. In Ueli M. Maurer, editor, *EUROCRYPT'96*, volume 1070 of *LNCS*, pages 387–398. Springer, Berlin, Heidelberg, May 1996. doi:10.1007/3-540-68339-9_33.

[RS21] Lior Rotem and Gil Segev. Tighter security for schnorr identification and signatures: A high-moment forking lemma for Σ-protocols. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part I*, volume 12825 of *LNCS*, pages 222–250, Virtual Event, August 2021. Springer, Cham. doi:10.1007/978-3-030-84242-0_9.

[Sch90] Claus-Peter Schnorr. Efficient identification and signatures for smart cards. In Gilles Brassard, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 239–252. Springer, New York, August 1990. doi:10.1007/0-387-34805-0_22.

[Sch91] Claus-Peter Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, January 1991. doi:10.1007/BF00196725.

[Sho97] Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 256–266. Springer, Berlin, Heidelberg, May 1997. doi:10.1007/3-540-69053-0_18.

[WC81] Mark N Wegman and J Lawrence Carter. New hash functions and their use in authentication and set equality. *Journal of computer and system sciences*, 22(3):265–279, 1981. doi:10.1016/0022-0000(81)90033-7.