Adaptive Proofs of Knowledge in the Random Oracle Model

David Bernhard[1] and Marc Fischlin[2] and Bogdan Warinschi[1]

[1]University of Bristol [2]Technische Universität Darmstadt

Abstract. We formalise the notion of adaptive proofs of knowledge in the random oracle model, where the extractor has to recover witnesses for multiple, possibly adaptively chosen statements and proofs. We also discuss extensions to simulation soundness, as typically required for the "encrypt-then-prove" construction of strongly secure encryption from IND-CPA schemes. Utilizing our model we show three results:

(1) Simulation-sound adaptive proofs exist.
(2) The "encrypt-then-prove" construction with a simulation-sound adaptive proof yields CCA security. This appears to be a "folklore" result but which has never been proven in the random oracle model. As a corollary, we obtain a new class of CCA-secure encryption schemes.
(3) We show that the Fiat-Shamir transformed Schnorr protocol is *not* adaptively secure and discuss the implications of this limitation.

Our result not only separates adaptive proofs from proofs of knowledge, but also gives a strong hint why Signed ElGamal as the most prominent encrypt-then-prove example has not been proven CCA-secure without making further assumptions.

1 Introduction

Proofs of knowledge [34,24,52,6] are a generic tool to ensure correct operation in many cryptographic constructions, including voting protocols, e-cash systems, or group signatures. More generally, they can turn passively secure multi-party protocols into actively secure ones. The value of proofs of knowledge in security arguments is that whenever a participant makes a proof of knowledge on some statement as part of a protocol, one can "hop" into an alternate, virtual world in which the participant outputs the witness along with the statement. This approach of pretending that each proof makes its witness available in a security argument relies on the extractor that exists by definition of a proof of knowledge: when a participant outputs a proof, we "freeze" the protocol, and invoke the extractor to get the witness. This extraction is usually carried out by rewinding the party and branching into another protocol execution. Then we resume the protocol with the witness now being available.

The problem with the "freeze-extract-resume" approach is that its implementation can easily become expensive. Each extraction and its rewinding can double the running time of a reduction such that, if a participant makes a badly

nested "chain" of n proofs, a naive approach ends up with an exponential running time of 2^n to get all the witnesses. This is certainly true for interactive proofs of knowledge, but also in the case of non-interactive proofs of knowledge in the random oracle model. Such random-oracle based proofs are paramount if efficiency is vital, especially in the form of Fiat-Shamir transformed Sigma protocols a.k.a. "Schnorr-type" proofs. In this context the rewinding problem was first mentioned explicitly by Shoup and Gennaro [50].

Shoup and Gennaro [50] required nested proofs in the random oracle model for the construction of CCA secure public-key encryption from IND-CPA secure encryption via the encrypt-then-prove approach (e.g., for signed ElGamal). The idea behind this approach, gradually refined in a sequence of works [15,39,43,19,57,48], is to attach to each ciphertext a proof of knowledge of the message. Intuitively, if one has to know the message to create a ciphertext, a decryption oracle should be redundant, so encrypt-then-prove should lift CPA security to CCA security. Unfortunately, there is no general proof of this intuition which also covers the setting with random oracles. Currently, the best result for signed ElGamal, without making additional "knowledge type" assumptions as in [53,46], is that the scheme is non-malleable (NM-CPA) [14].

Adaptive Proofs of Knowledge. Our notion and formalisation of *adaptive* proofs of knowledge allows to capture the case of having to extract from multiple proofs, possibly chosen adaptively by a malicious prover. We focus on the case of non-interactive proofs in the random oracle. As a first step, we will cast the (single-round) proof of knowledge property as a game between a prover (or attacker) and an extractor. The prover wins the game if it makes a statement and a valid proof but such that the extractor cannot find a witness. The extractor wins the game if it can return a witness (or if the prover does not produce a valid proof). A proof scheme is a proof of knowledge if there is an extractor that wins against any prover with overwhelming probability.

For extending our simple game to the adaptive case, the prover can now produce many statement/witness pairs in rounds and the scheme is an adaptive proof if the extractor can find all witnesses (for a prover who makes a polynomially bounded number of queries). The game is adaptive because the extractor must return each found witness to the prover before the prover makes her next query.

In addition to adaptive proofs, we define *simulation-sound* adaptive proofs of knowledge. These proofs are obtained by the exact same change that extends proofs of knowledge to simulation-sound proofs of knowledge: in addition to producing statements and proofs of her own, the prover can simultaneously ask the zero-knowledge simulator for proofs on valid statements of her choice. Of course, the prover cannot ask the extractor to extract a witness from a simulated proof.

Our results. After we provide a formalisation of adaptive proofs we can argue about instantiations and applications. We provide three main results in this regard:

(1.) *Simulation-sound adaptive proofs exist.* We discuss that the construction of straight-line proofs of knowledge by Fischlin [28] satisifies our notion. Fischlin's transformation is an alternative to the common Fiat-Shamir transformation and allows any Sigma protocol with unique responses to be turned into a non-interactive proof.
(2.) *Adaptive simulation-sound proofs yield CCA security.* We propose that adaptive proofs are to proof schemes what CCA security is to encryption schemes. Only an adaptive proof gives you a formal guarantee that the intuition behind proofs of knowledge still works when they are used over multiple rounds of a protocol.

We prove that the encrypt-then-prove construction using an IND-CPA encryption scheme and a simulation sound adaptive proof yields CCA security. Our proof is to our knowledge the first proof of CCA security that considers a potentially rewinding reduction in an adaptive case. While our proof follows the same high-level direction as proofs of existing CCA schemes (using the reduction to answer decryption queries), the need to handle rewinding without causing an exponential blow-up makes for a complicated argument. We develop a new proof technique called coin splitting to deal with some of the problems that arise.
(3.) *Fiat-Shamir-Schnorr is not adaptively secure.* We prove that the most common and efficient construction of proofs of knowledge via the Fiat-Shamir transformation [27] is not adaptively secure. Our proof constructs a prover who makes a "chain" of n Fiat-Shamir-Schnorr statement/proof pairs, following the ideas of Shoup and Gennaro [50]. We then show that any extractor that wins the adaptive proof game against this prover either reduces to breaking the one-more discrete logarithm problem or launches $\Omega(2^n)$ copies of the prover. The key technical tools in the proof are the meta-reduction paradigm and a technique which we term coin splitting.

Coin splitting allows us to perform a kind of hybrid argument on attackers which have rewinding black-box access to several copies of the same algorithm. We can change these copies individually as long as we can argue that the attacker does not notice the difference. Coin splitting is a technique to show that some changes which we make to individual copies are indeed indistinguishable to an attacker who cannot break a more basic security assumption. The idea of this technique originates in papers on resettable zero-knowledge [16].

Related work. We recall some related work here and discuss that so far no previous work has given a profound answer to the issue of adaptive simulation-sound proofs of knowledge in the random oracle model. We describe how our work first into a larger context more comprehensively in Appendix A. The notion of simulation-soundness of zero-knowledge proofs has been introduced for proofs of membership by Sahai [44], showing that the Naor-Yung paradigm [39] yields CCA secure encryptions in the common reference string model. In the context of proofs of knowledge, De Santis and Persiano [21] already augmented ciphertexts by proofs in the common reference string model to aim at CCA security, albeit their argument seems to miss simulation-soundness as an important ingredient. This property has been considered in works by Groth [36], Chase and Lysanskaya

[17], and by Dodis et al. [22], but once more in the common reference string model only. The first formal definitions of simulation-sound proofs of knowledge in the random oracle model were concurrently given by Bernhard et al. [14] and Faust et al. [23]; both works show that proofs derived via Fiat-Shamir transform meet this notion. Both formulations, however, consider an extractor that needs only to extract from non-adaptively chosen proofs, and in case of [23] only once (for security of their signature construction). In conclusion, our work here fills in a gap allowing to argue about important properties of adaptivity and simulation-soundness of proofs of knowledge in the random oracle model.

2 Zero-knowledge proofs

In this section we discuss zero-knowledge proofs of knowledge and simulation soundness in the random oracle model (ROM). Our central idea for zero-knowledge and its extension of simulation soundness is a game between two players, a malicious prover $\widehat{\mathcal{P}}$ and an extractor $\mathcal{K}$. The prover's aim is to produce a statement and a proof that verifies such that the extractor cannot extract a witness from this proof. The extractor's goal is to extract a witness from the proof that the prover provides.

We use a code-based game-playing model à la Bellare and Rogaway [11] to define adaptive proofs of knowledge. The game mediates between all involved players, e.g., between the adversarial prover and the extractor and possibly also the simulator in case of simulation soundness. The game starts by running the initialisation algorithm which ends by specifying to which player it wishes to transfer control to first. Executions are token-based: at any time, either the game or one of the players is running ("holding the token") and a call from one participant to another yields control. All game variables are global and persist between calls so the game may maintain state between calls. The game eventually declares the winner among the prover and the extractor.

To ensure termination, we assume strict polynomial-time provers and extractors (in the size of implicit parameters such as the size of the groups over which the proofs are constructed). Our notions could also be achieved by having the game "time out" if one player does not reply in a bounded number of time steps, though this would require a more involved model of concurrency. The important property is in any case that all players in the game must, on receiving an input, eventually produce an output. In particular, a prover cannot prevent a "perfect" extractor from winning a game by entering an infinite loop instead of producing a proof.

Proof schemes. A cryptographic group $\mathbb{G}$ consists of a group (in the sense of group theory) of prime order q with a distinguished generator g. A group generation algorithm $\mathsf{GrpSetup}$ is an algorithm that takes a security parameter $\lambda \in \mathbb{N}$ (in unary encoding, written 1^λ) and outputs a cryptographic group $\mathbb{G}(\lambda)$. A relation over cryptographic groups is a relation R' parametrised by the "interface" of a group. For example, in any cryptographic group $\mathbb{G}$ one can define the discrete logarithm relation $R'(\mathbb{G}) := \{(x, w) \mid x \in \mathbb{G} \wedge x = g^w\}$ where g is

the distinguished generator of $\mathbb{G}$. For a group generator $\mathsf{GrpSetup}$ and a relation R' over groups one can define the relation $R = \bigcup_{\mathbb{G}(\lambda)} R'(\mathbb{G}(\lambda))$ for a sequence of groups $\mathbb{G}(\lambda)$, $\lambda \in \mathbb{N}$; proof schemes typically prove relations R of this kind that are in class NP.[1] An algorithm over cryptographic groups is given by pseudocode that uses the "interface" of groups. For example, the algorithms $\mathcal{P}$ and $\mathcal{V}$ below are algorithms over groups: they can be implemented for any given group[2] $\mathbb{G}$.

Definition 1. *A non-interactive proof scheme for a relation R over groups consists of two algorithms $(\mathcal{P}, \mathcal{V})$ over groups. $\mathcal{P}$ may be randomised, $\mathcal{V}$ must be deterministic. For any pair $(x, w) \in R$, if $\pi \leftarrow \mathcal{P}(x, w)$ then $\mathcal{V}(x, \pi)$ must output "true".*

The elements of the relation R are called statement and witness. $\mathcal{P}$ is called the prover and its outputs π are called proofs. $\mathcal{V}$ is called the verifier and a statement/proof pair on which $\mathcal{V}$ outputs "true" is called a valid proof. In the random oracle model, both $\mathcal{P}$ and $\mathcal{V}$ may call the random oracle but the relation R' itself must be independent of any oracles. The last condition in the definition of proof schemes is called correctness and says that proofs produced by the genuine prover are valid. In the random oracle model, the prover and verifier in the definition of the correctness property have access to the same random oracle, i.e. the oracle answers consistently for both algorithms.

Our definitions of properties for proof schemes are centered around a game with multiple interfaces to which various parties such as provers, extractors or simulators may connect. We give our games as collections of algorithms where each algorithm has both a name and a description of the interface to which it applies. A `return` statement in an algorithm terminates the algorithm and outputs the return value on the same interface that called the algorithm. Where an algorithm should terminate and send a value on a different interface, we use the keyword `send` instead. The keyword `halt` in the code of a game terminates not just an algorithm but the entire game — when this occurs, the game will announce a winner.

Zero-knowledge. A proof scheme is called zero-knowledge if there is an algorithm $\mathcal{S}$, called the simulator, which is able to produce proofs indistinguishable from genuine ones after seeing only the statement but no witness. Informally, $\pi \leftarrow \mathcal{P}(x, w)$ and $\pi' \leftarrow \mathcal{S}(x)$ should be indistinguishable for any pair $(x, w) \in R$.

In the (programmable) random oracle model we define zero-knowledge in such a way that the simulator is responsible for the random oracle (if present). Formally, we treat the prover $\mathcal{P}$ as an interactive algorithm that may issue oracle queries and get responses, and that eventually outputs a proof. A simulator is a stateful interactive algorithm $\mathcal{S}$ that can respond to two kinds of queries: a prove query which takes a value x as input and should produce a proof π as

[1] Note that it would be pointless to ask whether R' is in class NP or not, as $R'(\mathbb{G})$ is a finite object for any finite group $\mathbb{G}$.

[2] If we wanted to be really precise we could write $\mathcal{P}(\mathbb{G})(x, w)$ to capture the dependency on a group. We choose not to use subscripts in order to avoid subscripted subscripts when the group itself depends on a security parameter.

output, and a ro query which takes an arbitrary $x \in \Sigma^*$ as input and returns a $y \in \Sigma^*$. A simulator does not have access to a random oracle, but simulates its own random oracle towards its "clients". A proof scheme is zero-knowledge in the random oracle model if the following two games are indistinguishable:

- The first game internally runs a random oracle RO . On input a pair (x, w), if $R(x, w)$ does not hold then the game returns $\perp$ and halts. If the relation holds, the game runs $\pi \leftarrow \mathcal{P}(x, w)$ and returns π. The prover $\mathcal{P}$ uses the game's random oracle. The adversary may then query the game's random oracle (which $\mathcal{P}$ used) directly, as often as she wants.
- The second game does not run a random oracle. On input a pair (x, w), again if $R(x, w)$ does not hold the game returns $\perp$ and halts. Otherwise, the game runs $\pi \leftarrow \mathcal{S}(x)$ and returns π to the adversary. The adversary may then issue random oracle queries which the game delegates to the simulator's random oracle.

We specify this property using pseudocode in Figure 1. We use the following notation: An oracle is a stateful process which other processes can access via a well-defined set of queries. If $\mathcal{O}$ is an oracle and q is one if its supported queries then we write $\mathcal{O}.\mathsf{q}(x)$ to denote the invocation of this query with parameter x. We write $x \leftarrow\!\!\$\ S$ for selecting x uniformly at random from the set S and $y \leftarrow\!\!\$\ \mathcal{A}^{O_1,\ldots,O_n}(x)$ for calling the (potentially randomised) algorithm $\mathcal{A}$ on input x to get output y. The superscripts denote the oracles that $\mathcal{A}$ can use while it is running. Sometimes, we will allow these oracles to call each other directly (for example if several oracles need access to a random oracle) and to issue a command halt that halts the entire execution.

To maintain random oracle queries in later definitions we write [ ] for the empty list and $L :: l$ to concatenate element l to list L. When L is a list of pairs, we define $L(x)$ to be y such that (x, y) is the first element in L of the form $(x, \cdot)$. If no such element exists then $L(x)$ is defined to be $\perp$.

In Figure 1 we give the games G_1 and G_2 and the methods that the adversary can call. Since it will be helpful later on to give each kind of query a name, we call the adversary's initial query with parameters (x, w) a prove query. Similarly, we call the two operations that a simulator $\mathcal{S}$ admits prove and ro queries.

At the moment, our code may seem like an unnecessarily complicated way of stating a simple property. This level of formalism will become necessary when we move on to adaptive proofs however.

Definition 2. *A proof scheme $(\mathcal{P}, \mathcal{V})$ is zero knowledge in the random oracle model for a relation R if there exists a simulator $\mathcal{S}$ satisfying the following condition. For any security parameter λ let $\delta(\lambda)$ be the distinguishing advantage of any efficient adversary between the games G_1 and G_2 of Figure 1 and for relation R and simulator $\mathcal{S}$. Then $\delta(\lambda)$ is negligible as a function of λ.*

Proofs of knowledge. A proof scheme is a proof of knowledge if there is an extractor $\mathcal{K}$ such that for any prover $\widehat{\mathcal{P}}$ which can make a statement/proof pair that verifies, $\mathcal{K}$ can deliver an associated witness. Formalising this statement

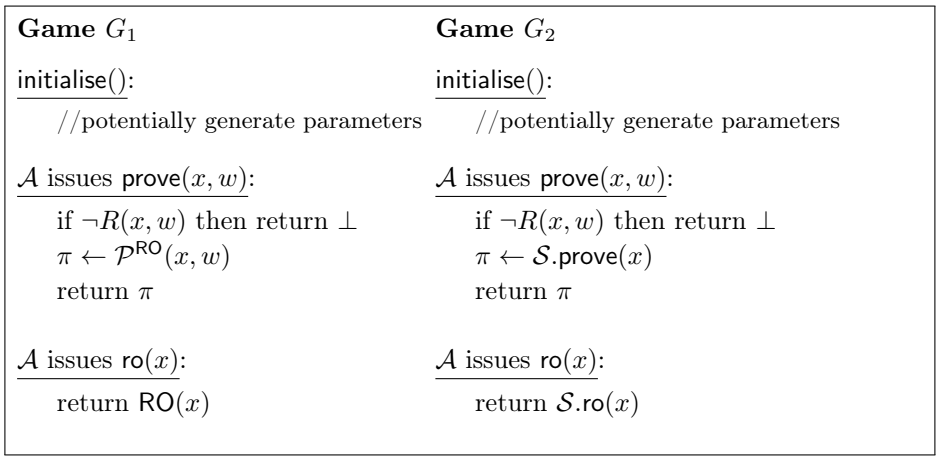


Fig. 1: Games for zero-knowledge (ZK) in the random oracle model. A scheme $(\mathcal{P}, \mathcal{V})$ is ZK if the two games G_1 and G_2 are indistinguishable. The adversary $\mathcal{A}$ may issue prove once and ro any number of times. RO is a random oracle.

requires that we not only take care of random oracles but also the extractor's ability to "fork" the prover.

We first consider the non-rewinding case as a warm-up. A prover $\widehat{\mathcal{P}}$ is a randomized interactive algorithm that may make random oracle queries and eventually outputs a pair (x, π). A non-rewinding extractor $\mathcal{K}$ is an algorithm that takes a pair (x, π) as input, may make random oracle queries and eventually outputs a value w. We consider the game G that runs a random oracle RO internally and connects a prover and an extractor as in Figure 2. A proof scheme is an R-proof of knowledge if there is an extractor $\mathcal{K}$ such that for every prover $\widehat{\mathcal{P}}$, the game mediating between the two algorithms outputs "$\mathcal{K}$ wins" with overwhelming probability.

The game as in Figure 2, in which both the prover $\widehat{\mathcal{P}}$ and the extractor $\mathcal{K}$ can access a random oracle and where the extractor is supposed to find a witness for a valid proof produced by the prover, is actually too demanding to be useful: It basically says that anyone is able to extract a witness from the proof. To derive some sensible notion we give the extractor some advantage and allow it to inspect the random oracle queries made by the prover. That is, the extractor $\mathcal{K}$ can make an extra query list in response to which the game G returns the list H. This gives us a notion of straight-line proofs in the random oracle model which is actually sufficient for capturing the approach used by Fischlin [28].

Definition 3. *A proof scheme $(\mathcal{P}, \mathcal{V})$ is a straight-line proof of knowledge in the ROM w.r.t. a relation R if there is an extractor $\mathcal{K}$ such that for any prover $\widehat{\mathcal{P}}$, the game in Figure 2 augmented with a* list *query that allows $\mathcal{K}$ to see the list H returns "$\mathcal{K}$ wins" with overwhelming probability.*

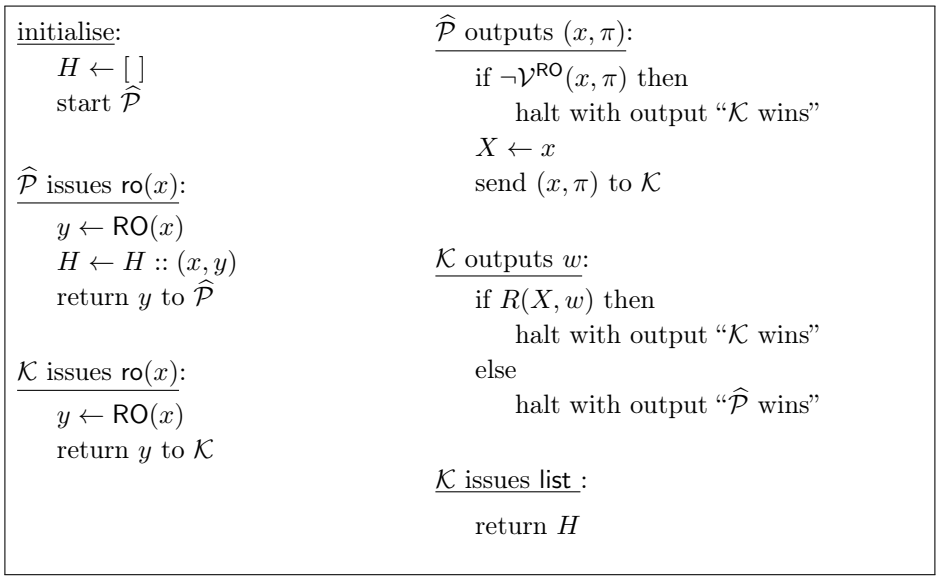


Fig. 2: The game G defining proofs of knowledge in the random oracle model. Capital-X is part of the game's internal state that persists between calls (so that the extractor's witness is verified against the same statement that the prover provided earlier).

The above definition is less general than the one first proposed by Bellare and Goldreich [6]. There the authors relate the extractor's success probability to that of the prover (in producing a valid proof), whereas our definition lets the extractor win by default if the prover does not make a proof. However, our notion generalises more easily to the adaptive setting where the probability of the prover making a valid proof is no longer well-defined, since it also depends on the extractor's response to earlier proofs.

Rewinding Extractors. The next standard notion that we formalise in our game-based model is that of a rewinding extractor in the ROM, running the prover multiple times. We model the extractor's rewinding ability by giving the extractor access to further copies of the prover, initialised with the same random string as the main incarnation of the prover's algorithm which connects to the proof of knowledge game. We call these further copies "rewound copies". Although all copies of the prover share the same random string, this string is not available to the extractor directly. This prevents the extractor from just simulating the prover on its own and reading off any witness used to make a proof.

The rewound copies of the prover connect to the extractor directly as sketched in Figure 3. In particular, the extractor is responsible for answering the random oracle queries for the rewound copies and can use this ability to "fork" them at any point. In order to apply the forking strategy to proofs made by the main prover, the extractor may make use of the list H that records all random oracle queries and answers for the main execution.

The game itself is the same as for non-rewound provers. For example, for the prover in Schnorr's protocol, one extraction strategy is to start a rewound copy of the prover and run it up until the point that it asks the oracle query which the main prover used to make its proof. Then, the extractor gives the rewound copy a different answer to the oracle query and hopes to obtain a new proof with the same commitment, in order to extract via special soundness. If the main prover made other oracle queries before the "target" one, then the extractor looks these up in the list H and gives the rewound copy the same answers when it makes the relevant queries.

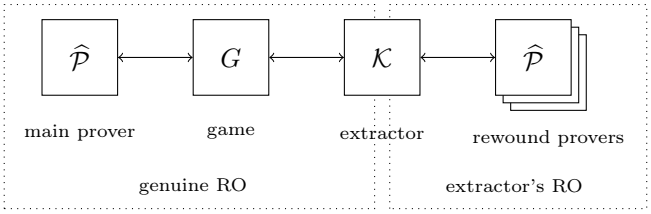


Fig. 3: Extending the straight-line proof of knowledge game to the rewinding case.

Definition 4. *A proof scheme is a rewinding proof of knowledge in the ROM if it satisfies the conditions of Definition 3 (proof of knowledge) for an extractor $\mathcal{K}$ that has black box access to further copies of the main prover with the same random string.*

Simulation soundness and Extractability. Simulation soundness is a property of some zero-knowledge proofs, where even after seeing a simulated proof you cannot construct a new proof of a false statement. Simulation soundness was introduced by Sahai [44] for proofs of statements; unlike proofs of knowledge these do not require an extractor. Sahai used simulation soundness to show that the Naor-Yung "double encryption" transformation can be used to obtain CCA secure encryption. Naor-Yung is not an encrypt-then-prove construction. The latter use only a single encryption but require a proof of knowledge; their security arguments make use of the extractor.

In fact, encrypt-then-prove requires a proof scheme for which one can apply both the zero-knowledge simulator and the proof-of-knowledge extractor in the same security argument. The formal property that models this is called simulation-sound extractability (SSE) [36]. Specifically, the extractor must still work even if the simulator has been invoked, as long as one does not try to extract from a simulated proof. Simulation soundness is often challenging to achieve outside the random oracle model. The proof scheme of Groth and Sahai [55] for example, even in the instantiations that are proofs of knowledge, operates with a setup parameter that can be constructed in two ways: either

one can simulate proofs or one can extract, but not both simultaneously. In the random oracle model simulation soundness is typically easier to achieve, e.g., it comes almost for free with Schnorr-type proofs. However, it takes some care to formalise this property as the simulator works under the condition that it controls the random oracle. Hence, the extractor must now succeed w.r.t. the simulator's random oracle in this case.

We model simulation-sound extractability by taking the game for proofs of knowledge and giving the prover the extra ability to ask `prove` queries just like in the zero-knowledge game. These queries are always answered by the zero-knowledge simulator and their proof replies are banned from being handed over to the extractor. The SSE game runs the simulator and delegates random oracle queries to it. The result is the game G in Figure 4. The list Π keeps track of simulated proofs. If the prover returns a simulated proof (on the same statement as it used in the related proof query), it loses the game. The state C is required for a bit of extra bookkeeping since the random oracle is now external to the game. By $\mathcal{V}^{\mathcal{S}.\mathsf{ro}}$ we mean that the game G runs the verifier $\mathcal{V}$ and uses the simulator's random oracle to answer any oracle queries made by the verifier. In other words, in the SSE game even the notion of a valid proof depends on the simulator. Unlike the prover $\widehat{\mathcal{P}}$ which is one of the players in our game, the simulator $\mathcal{S}$ is assumed to always produce valid proofs by the zero-knowledge property. The extractor's list query returns both the random oracle queries and the proof queries made by the main prover so far — this allows the extractor to make a rewound copy of the prover run in identical executions as in the main copy.

In addition to the main game G, we define an auxiliary game $\widehat{G}$ that sits between the extractor $\mathcal{K}$ and its rewound provers (there is one copy of $\widehat{G}$ for each rewound prover). The task of $\widehat{G}$ is to "sanitize" prove queries made by rewound provers. When a rewound prover makes such a query, the extractor must play the role of the simulator — after all, the extractor is already simulating the rewound prover's random oracle. (The extractor may run a copy of the simulator $\mathcal{S}$ internally.) However, provers make prove queries containing both a statement x and a witness w whereas the simulator only ever gets to see x. The auxiliary game $\widehat{G}$ strips the witness from these proof queries. Otherwise, $\widehat{G}$ acts as a channel between $\mathcal{K}$ and a rewound copy of $\widehat{\mathcal{P}}$. This is slightly tedious to write in our notation; we make the convention that $\widehat{G}$ prefixes a string to every message from $\widehat{\mathcal{P}}$ to $\mathcal{K}$ to indicate whether the value is meant to be a random oracle, extraction or proof query. Messages (responses) flowing in the other direction can always be passed on unchanged — the prover will hopefully remember what its last query was when it gets a response.

Definition 5. *A proof scheme $(\mathcal{P}, \mathcal{V})$ is simulation sound in the ROM for a relation R if it satisfies the following conditions. An s-prover is an algorithm $\widehat{\mathcal{P}}$ that can ask random oracle and proof queries and eventually outputs an extraction query containing a statement/proof pair.*

- *The proof scheme is zero-knowledge w.r.t. R for a simulator $\mathcal{S}$ and a proof of knowledge w.r.t. R for an extractor $\mathcal{K}$.*

– *For every s-prover $\widehat{\mathcal{P}}$, if we connect $\mathcal{K}$ to $\widehat{\mathcal{P}}$ through the game G of Figure 4 and give $\mathcal{K}$ access to further rewound copies of the prover (with the same random string) through the auxiliary game $\widehat{G}$ of Figure 5 then with overwhelming probability the game G returns "$\mathcal{K}$ wins".*

3 Adaptive Proofs of Knowledge

Given our game-centric view of proofs of knowledge we can extend the approach to adaptive proofs of knowledge. An adaptive proof is simply a proof scheme where the extractor can still win if the prover is given multiple turns to make proofs. The adaptive part is that the game hands the extractor's witness in each turn back to the prover before the prover must take her next turn. Should a prover be able to produce a proof for which she does not know the witness, she could then use the extractor's ability to find a witness to help make her next proof. The intuition is essentially the same for the cases with and without simulation soundness. We first introduce adaptive proofs formally without simulation soundness using so-called n-proofs, where n is a parameter describing the number of rounds the prover plays. In a later step we add simulation soundness.

ADAPTIVE PROOFS AND n-PROOFS. Let $(\mathcal{P}, \mathcal{V})$ be a proof scheme for a relation R. An adaptive prover $\widehat{\mathcal{P}}$ in the ROM is a component that can make two kinds of queries, repeatedly and in any order. The first are random oracle queries; these are self-explanatory. The second are extraction queries which take a statement and a proof as parameters. The response to an extraction query is a witness. (Correctness conditions will be enforced by the game, not the prover.) Adaptive provers may also halt. For example, a non-adaptive prover can be seen as an adaptive prover that halts after its first extraction query.

An adaptive extractor $\mathcal{K}$ is a component that can make list and random oracle queries and receive and process extraction queries from an adaptive prover. In addition, an extractor may have black-box access to further rewinding copies of the adaptive prover (with the same random string) and answer all of their queries.

The n-proof game takes a parameter n as input and connects an adaptive prover and extractor. It runs up to n rounds in which the prover may make a statement and proof and the extractor must return a witness. The extractor wins if it can deliver all n witnesses or if the prover halts earlier than this, or fails to make a valid proof. The extractor loses if it does not supply a valid witness to one of the first n extraction queries.

Definition 6. *A proof scheme is an n-proof in the ROM for a relation R if there exists an extractor $\mathcal{K}$ such that for every adaptive prover $\widehat{\mathcal{P}}$ the game G of Figure 7 when connected to $\widehat{\mathcal{P}}$ and $\mathcal{K}$ returns "$\mathcal{K}$ wins" with overwhelming probability.*

If $\mathcal{K}$ also has access to further copies of $\widehat{\mathcal{P}}$ with the same random string then we call the proof scheme a rewinding n-proof, otherwise we call it a straight line n-proof.

If for every polynomial $p(x)$ there is an extractor $\mathcal{K}_{p(x)}$ making a particular scheme a $p(n)$-proof, we say that the scheme is an adaptive proof.

Simulation-sound adaptive proofs. Adding simulation soundness to adaptive proofs works the same way as for non-adaptive proofs. Adaptive s-provers may make random oracle, proof and extraction queries (the simulation sound n-proof game only limits the number of extraction queries, not proof queries). We give the new algorithms in Figure 8. Random oracle calls from the main prover go to the simulator; simulated proofs are logged and provided on request to the extractor (via a list query) and are banned from extraction queries. The rewinding copies of the prover are connected to the extractor through the same games $\widehat{G}$ as in the non-adaptive case: extraction queries and witnesses found by the extractor are simply passed back and forth. Only the witnesses in prove queries are stripped out.

Consider a proof scheme $(\mathcal{P}, \mathcal{V})$ that is zero-knowledge for a relation R with simulator $\mathcal{S}$. The simulation sound n-proof experiment for this scheme, an adaptive s-prover $\widehat{\mathcal{P}}$ and an extractor $\mathcal{K}$ is the following experiment. Connect the prover $\widehat{\mathcal{P}}$, the simulator $\mathcal{S}$ and the extractor $\mathcal{K}$ to the game G of Figure 8. Let $\mathcal{K}$ have black box access to further copies of $\widehat{\mathcal{P}}$ mediated by $\widehat{G}$ as in Figure 5 that forwards all messages in both directions except that it strips witnesses from proof queries.

Definition 7. *Let $(\mathcal{P}, \mathcal{V})$ be a proof scheme for a relation R that is zero-knowledge with simulator $\mathcal{S}$. The scheme is a simulation sound n-proof if there is an extractor $\mathcal{K}$ such that for any adaptive s-prover $\widehat{\mathcal{P}}$, the simulation sound n-proof experiment returns "$\mathcal{K}$ wins" with overwhelming probability. If the extractor works for all polynomially bounded n, we call the scheme an adaptive simulation sound proof.*

Discussion. We can classify the types of proofs that we have introduced with the following table. "poly" means polynomially many and means that the number of queries is not bounded by the games involved but an efficient algorithm will be able to launch at most polynomially many queries (in some initial input or security parameter) in the first place. We see no reason to study explicit bounds on the number of proof queries as our simulators are non-rewinding.

type	# extraction queries	# proof queries
PoK	1	none
ss-PoK	1	poly
n-proof	n	none
ss-n-proof	n	poly
adaptive proof	poly	none
ss adaptive proof	poly	poly

The adaptive property of our proofs is strictly stronger than allowing the prover to make one extraction query with a vector of many statement/proof

pairs. Namely, the multiforking lemma of Bagherzandi et al. [5] shows that Fiat-Shamir-Schnorr proofs are extractable for one such "parallel" query yet we will show that such proofs are not adaptive proofs (under the one-more discrete logarithm assumption). This separation has an analogue in the world of public-key encryption that we will discuss soon. Sahai [44] shows that security against one "parallel" decryption query is equivalent to non-malleability (NM-CPA) which is known to be strictly weaker than CCA security where adaptive decryption queries are allowed. It is also known that adding a proof of knowledge (of the message and randomness) to an IND-CPA secure encryption scheme can be used to construct non-malleable encryption yet this technique is not guaranteed to achieve CCA security.

After establishing that simulation sound adaptive proofs exist by studying a construction due to Fischlin [28], we show two main results. The negative result is that the Fiat-Shamir transformed Schnorr protocol is not adaptively secure.

Adaptive proofs bear an interesting relationship to CCA security of encryption. Although adaptivity is an interesting question for many applications in particular in multiparty computation, almost all known constructions of CCA secure public-key encryption use some form of non-interactive proof of knowledge in their ciphertexts[3]. This is sometimes known as the encrypt-then-prove construction.

Our conceptual understanding of our new notion is that simulation sound adaptive proofs are to proof schemes what CCA is to encryption. Under this comparison, normal (simulation sound) proofs of knowledge are the equivalent notion to non-malleability (NM-CPA). We validate this connection by showing that the encrypt-then-prove construction with an adaptively secure proof is indeed CCA secure, assuming the encryption scheme was IND-CPA secure. Together with Fischlin's proof scheme this yields a new CCA secure encryption scheme. Our proof is to our knowledge the first proof of a CCA secure construction with a potentially rewinding extractor, leading to many complexities not present in the proofs of existing schemes.

On the negative side, Shoup and Gennaro [50] first observed that adding a Fiat-Shamir-Schnorr proof to ElGamal encryption cannot be proven CCA secure the "obvious" way. This construction is called Signed ElGamal or TDH0 and if it were CCA, it would outperform all other known CCA public-key schemes and at the same time allow for homomorphic[4] operations e.g. for tallying in an electronic voting scheme. The problem with Signed ElGamal is that the usual rewinding extractor suffers an exponential blow-up when used adaptively. While Shoup and Gennaro found an example of this, they admit that they could neither prove nor disprove CCA security of Signed ElGamal; this has since become a

[3] The exception is the original construction via the Naor-Yung transformation [39] but this comes at the cost of having to encrypt twice.

[4] CCA secure schemes cannot of course be homomorphic in the usual sense. However, an encrypt-then-prove ciphertext consists of two parts, a "small" ciphertext (usually ElGamal) and a proof; after checking the proof, one may in some cases discard the proofs and work homomorphically with only the small ciphertexts.

long-standing open problem. In this work, we give the first formal proof of an inherent limitation in the Fiat-Shamir-Schnorr construction. Under the one-more discrete logarithm assumption, we show that any extractor for such proofs in the random oracle model must make at least $\Omega(2^n)$ "forking" executions to extract from the n-prover originally sketched by Shoup and Gennaro. It follows that Fiat-Shamir transformed Sigma protocols are not adaptive proofs.

The big open question is still whether Signed ElGamal (ElGamal with a Fiat-Shamir-Schnorr proof; we formalise this in the next section) is CCA secure. Our results rule out any proof of CCA security of Signed ElGamal that depends on or implies adaptive security of the contained Fiat-Shamir-Schnorr PoK, which would otherwise seem the most straightforward way to obtain CCA security. We cannot rule out proofs using other techniques however. Certainly, Schnorr and Jakobsson [46] have given a proof in a combination of the random oracle and generic group models. Tsiounis and Yung [53] have done the same under a "knowledge assumption" for which however we know no justification that does not invlove both generic groups and random oracles. Interestingly, if the Schnorr-Jakobsson technique were applicable without generic groups then Signed ElGamal would not only be CCA but also PA2 "plaintext aware" yet Seurin and Treger [48] have shown that Signed ElGamal cannot be PA2 in the random oracle model (without generic goups) unless ElGamal is insecure (CDH is easy) in the first place, in which case the whole construction can never be CCA secure[5]. Seurin and Treger too admit that their results cannot rule out a proof of CCA security of Signed ElGamal by other means.

4 Overview of our results

In this section we briefly discuss our main results. Since the proofs of our theorems are long and contain many technical/formal details that are not particularly enlightening, we have chosen to give only an overview of our results in the body of the paper and place the proofs in the appendices.

Adaptive proofs exist. First, we establish that simulation-sound adaptive proofs in the random oracle model exist. An existing construction due to Fischlin [28] is adaptively secure. Fischlin gives a transformation of Sigma protocols to non-interactive proof schemes as an alternative to the more common Fiat-Shamir transformation. We present Fischlin's transformation and a proof that it is adaptively simulation sound in Appendix C. Here, we sketch the construction.

Sigma protocols use a property called special soundness to achieve their security properties. Special soundness says that if you see any two proofs with the same statement and commitment but different challenges (which implies different responses) then you can compute a witness from the "difference" of the

[5] PA2 together with IND-CPA implies CCA but PA2 on its own says very little: the scheme in which ciphertexts are just plaintexts in the clear is PA2 and if the Schnorr-Jakobsson extractor worked without generic groups we would have the same situation for ElGamal.

two protocols. The rewinding extractor for Fiat-Shamir proofs exploits special soundness. Fiat-Shamir provers in the random oracle model obtain their challenges from the random oracle. The extractor runs a second copy of the prover identical to the main one from which it is trying to extract but since the extractor plays the random oracle towards the second copy of the prover, the extractor can give the second prover a different challenge. If the second prover makes a proof, the extractor can take the first and second prover's proofs and apply special soundness. The formal argument that this works (with overwhelming probability for a prover who almost always succeeds in making a proof) is called the Forking Lemma [10].

Fischlin's idea is to impose an additional constraint on proofs: a number of low-order bits in the hash of the entire proof must be all zero. This forces the prover to make several attempts until she finds a proof that meets the constraint. The number of bits that must be zero is small enough that a prover can succeed with a reasonable number of attempts and large enough that with high probability, a prover will need more than one attempt. The prover must check each attempt by sending the whole proof to the random oracle. A real verifier will only ever see one proof and not the prover's intermediate attempts; an extractor however can see all of the prover's random oracle queries. Fischlin shows that with high probability, a prover must send two candidate proofs to the random oracle before she finds a proof satisfying the new constraint and from two such proofs, an extractor can extract a witness by special soundness. Since the prover is doing the "rewinding" herself, the extractor for Fischlin proofs is straight line, i.e. never needs to rewind the prover.

The following theorem (which we prove in Appendix C) establishes that Fischlin proofs are adaptive.

Theorem 1. *A Fischlin-transformed Sigma protocol with special soundness is a simulation-sound adaptive proof in the random oracle model.*

Encrypt-then-prove. Our main positive result is that the encrypt-then-prove transformation does what it is intuitively supposed to do — boost IND-CPA to CCA — if the proof scheme is a simulation-sound adaptive proof. To define the transformation we first clarify a point about NP languages. In the introduction, we said that encrypt-then-prove uses a proof of the "randomness and message" used to encrypt. This is not precise enough for a formal definition. This informal statement would give us a proof over a relation $R_1 : \{(c, (m, r)) \mid c = \mathsf{Encrypt}(pk, m; r)\}$ where statements are ciphertexts and witnesses are message/randomness pairs. However, Signed ElGamal (which we will define soon) uses a Schnorr proof which is a proof of knowledge of a discrete logarithm, namely the randomness in an ElGamal ciphertext. This would suggest a relation $R_2 : \{(c, r) \mid c = \mathsf{Encrypt}(pk, m; r)\}$. Of course, the point of the proof is that the message can be computed from a ciphertext and its randomness, but that is not the same thing as the formal definition of the proof's NP relation. In addition, since the NP relation and proof depends on the public key as an extra parameter, when we define the transformation formally we are actually working with a

parametrised family of relations. Further, the encrypt-then-prove transformation still works if one adds extra features to the proof. For example, the Helios voting scheme for example uses encrypt-then-prove ciphertexts that additionally prove that the encrypted message is a valid vote.

We address all these problems with an abstract definition of compatibility between encryption and proof schemes; any schemes that meet this definition can be used in the encrypt-then-prove transformation. Our definition also means that we will not have a concrete NP relation to work with in our main theorem. Instead, compatibility says that the NP relation can be anything that supports the two features we need: from a witness you can compute a message and from the list of all inputs used to form a ciphertext, you can derive a witness.

Definition 8. *An encryption scheme* $\mathfrak{E} = (\mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ *and a proof scheme* $\mathfrak{P} = (\mathcal{P}, \mathcal{V})$ *for relation* R *are compatible if there are efficient algorithms* M *and* W *such that:*

1. *For any tuple* (pk, c, w) *of public key, ciphertext and witness such that* $R((pk, c), w)$ *holds, the value* $m := \mathsf{M}(pk, c, w)$ *is the message such that* c *is an encryption of* m *under public key* pk*.*
2. *For any tuple* (pk, c, m, r) *of public key, ciphertext, message and random string, the value* $w := \mathsf{W}(pk, c, m, r)$ *is a witness for which* $R((pk, c), w)$ *holds.*

Definition 9. *Let* $\mathfrak{E}$ *and* $\mathfrak{P}$ *be compatible encryption and proof schemes (for a relation* R *and algorithms* M*,* W*. The encrypt-then-prove transformation of* $\mathfrak{E}$ *and* $\mathfrak{P}$ *is the encryption scheme in Figure 9 where* RS *is the space of random strings for* $\mathfrak{E}.\mathsf{Encrypt}$*.*

Signed ElGamal. As an example, we present the Signed ElGamal scheme. Signed ElGamal is ElGamal encryption with a Fiat-Shamir-Schnorr proof. It operates over a cyclic group $\mathbb{G}$ of prime order q with a generator G. To generate keys, pick a random $sk \leftarrow\!\!\$\ \mathbb{F}_q$ and set your public key to $pk \leftarrow G^{sk}$. To encrypt a message $m \in \mathbb{G}$, pick a random $r \leftarrow\!\!\$\ \mathbb{F}_q$ and create an ElGamal ciphertext $e \leftarrow (G^r, pk^r \cdot m)$. Then pick another random value $a \leftarrow\!\!\$\ \mathbb{F}_q$ and create the Schnorr commitment $A \leftarrow G^a$, challenge $c \leftarrow \mathcal{H}(pk, e, A)$ and response $s \leftarrow a + c \cdot r \pmod q$. The ciphertext is (e, A, s). To decrypt a ciphertext (e, A, s) with secret key sk, parse e as a pair (u, v) and check that $G^s = A + \mathcal{H}(pk, e, A) \cdot u \pmod q$. If this fails, the ciphertext is invalid. If it succeeds, the decryption is $m \leftarrow v/u^{sk}$. The relation R for Signed ElGamal is $R((pk, (u, v)), r) :\Leftrightarrow u = G^r$. Here (u, v) is an ElGamal ciphertext and $(pk, (u, v))$ is a statement consisting of a public key/ciphertext pair. The maps to make the encryption scheme and proof compatible are $\mathsf{M}(pk, (u, v), w) := v/pk^w$ and $\mathsf{W}(pk, (u, v), m, r) := r$.

Simulation-sound adaptive proofs yield CCA. Our main positive result expresses the intuition behind encrypt-then-prove.

Theorem 2. *Let* $\mathfrak{E}$ *be an IND-CPA secure encryption scheme and let* $\mathfrak{P}$ *be a compatible simulation-sound adaptive proof scheme in the random oracle model.*

Then the encrypt-then-prove transformation of these schemes is a CCA secure encryption scheme in the random oracle model.

As a corollary we immediately obtain a new CCA secure encryption scheme.

Corollary 1. *The encrypt-then-prove transformation of ElGamal using Fischlin-Schnorr proofs is CCA secure.*

The final step of the proof follows the basic intuition behind all encrypt-then-prove constructions. We reduce CCA security to IND-CPA security. Our reduction sends the two challenge messages to the IND-CPA game for the basic scheme, gets a ciphertext back and simulates a proof on it to create the challenge ciphertext of the encrypt-then-prove construction. When the CCA attacker makes a decryption query with an encrypt-then-prove ciphertext, the reduction invokes the extractor using the IND-CPA ciphertext component as the statement and the proof component as the proof. The witness contains the encrypted message which the reduction returns to the attacker. Since we are simulating and extracting in the same reduction, we require simulation sound extractability.

Unfortunately, this idea does not explain how the reduction is supposed to deal with the extractor requesting a further copy of the attacker. Worse still, the "prover" that we are simulating towards the extractor is a combination of the attacker and the IND-CPA challenger. We definitely cannot clone or rewind our challenger. To our knowledge, our proof is the first proof of a CCA construction that involves rewinding.

After an intial step in which we simulate all proofs on challenge ciphertexts, most of the proof is an argument how and why a single reduction can provide the extractor with a consistent simulation of multiple copies of the same algorithm. We call this technique coin splitting. It works on two principles. (1.) Keep track of which copies are "clones" of other copies. If a copy C is getting the exact same queries that another copy D has already answered, let C simply replay D's answers. (2.) Make sure that all cases not handled by the last point involve fresh, independent randomness. Then the reduction can simply draw fresh random values from one source to simulate all copies.

Coin splitting lets us use our one IND-CPA challenge for the extractor's main prover and simulate our own challenges for the rewinding provers. To the extractor, all this will look just like fresh randomness each time.

FIAT-SHAMIR-SCHNORR IS NOT ADAPTIVELY SECURE. Our third result is negative. It separates proofs of knowledge from adaptive proofs and shows that Fiat-Shamir Schnorr is an example that separates the two notions.

Theorem 3. *The Fiat-Shamir-Schnorr (FSS) proof scheme is not adaptively secure under the one-more discrete logarithm assumption. Specifically, for any n there is a prover $\widehat{\mathcal{P}}$ who makes a sequence of n FSS proofs. For any extractor $\mathcal{K}$ who can win the adaptive proof experiment against $\widehat{\mathcal{P}}$, either $\mathcal{K}$ calls at least 2^n rewinding copies of $\widehat{\mathcal{P}}$ or there is a reduction that solves the one-more discrete logarithm problem in the underlying group with a comparable success rate to $\mathcal{K}$.*

The prover in question follows the same ideas as Shoup and Gennaro's CCA attacker [50]. While the cited work gave the attacker as an example why the "obvious" proof fails, it did not show any inherent limitation of the Fiat-Shamir technique; it did not show that this limitation cannot be overcome by using a different proof technique. Our paper is the first to give a proof that Fiat-Shamir transformed sigma protocols have an unavoidable limitation.

The core of the proof is a kind of metareduction. For any extractor that wins the adaptive proof game against our prover, we construct a second "meta-extractor" that finds a discrete logarithm to one of the Schnorr challenges involved if the extractor's rewinding strategy satisfies certain conditions that we call "event E". We then replace the prover by a one-more discrete logarithm challenger to deal with the other proofs that may arise from rewinding copies. We reuse our coin splitting technique to argue that this game hop is undetectable by the extractor.

We use a combinatorial argument to show that any extractor who does not make at least 2^n calls to different rewinding copies of the prover must trigger event E. The gist of the argument is that we map the prover's n queries to a path in a depth-n binary tree where each node represents a discrete-logarithm problem. Each query that the extractor answers successfully reveals certain nodes in this tree and answering all n queries implies that the entire tree is revealed. We treat each rewinding operation that the extractor performs as exploring a path in the tree. If the extractor explores the complete tree we get the claimed 2^n rewinding operations. The event E is that the extractor "jumps" from one leaf in the tree to another without exploring the path in between. If this occurs, our reduction solves the one-more discrete logarithm problem. It is impossible, even for a computationally unbounded extractor, to reveal the whole tree (answer all n queries) without either exploring the whole tree (taking exponential time) or performing at least one jump (solving a one-more discrete logarithm instance).

5 Conclusion

It has been known for a long time [50] that Fiat-Shamir proofs can be problematic in adaptive settings. Our notion of adaptive proofs captures the kind of proofs we would like to have in such settings (and which are achievable via the Fischlin transformation). For example, simulation sound adaptive proofs capture in more detail than any previous work how Signed ElGamal and more generally encrypt-then-prove is supposed to be CCA secure. We also provide the first proof of an inherent limitation of the Fiat-Shamir transformation: it is not adaptively secure (under a one-more assumption on the underlying one-way function). The question of the exact security of Signed ElGamal remains open and we plan to address it in future work with similar techniques to the ones in this paper.

Adaptive Proofs outside the ROM.. In this paper, we have focused entirely on non-interactive proofs in the random oracle model. A good question is, can we generalise to e.g. common reference string-based proof schemes?

The game for adaptive proofs could easily be cast in other models. If additional "static" setup parameters such as a common reference string (CRS) are required, these can be generated once in the initialisation algorithm and shared between all parties (the extractor and any provers that it may invoke). If a random oracle is not required, the corresponding algorithms can simply be removed from the game. A CRS with an extraction trapdoor could also be generated in the initialisation algorithm: the extractor would get the CRS and the trapdoor, the provers would get the CRS only. Since the provers obtain their CRS from the game directly, the extractor cannot tamper with it. However, this whole discussion misses a point: adaptivity was never much of a problem with CRS-based proofs; their extractor is straight-line in all cases that we know of. Indeed, the only feature of some proof schemes that causes problems with adaptivity is the rewinding extractor, for which we know of no sensible instances except in the random oracle model. We would expect all non-ROM non-interactive proofs of knowledge to be adaptively secure, so we question whether there is much value in a notion of adaptive proofs outside the ROM.

We believe that the most interesting proof schemes are simulation sound extractable (SSE) ones, whether adaptive or not: such proofs seem to be required to boost security in an encrypt-then-prove construction. Typically, ROM proofs come with the SSE property for free. Simulation-sound extraction however can be a big problem in the CRS model. Consider Groth-Sahai proofs: one can set up either a hiding CRS with a potential simulation trapdoor or a binding CRS with a potential extraction trapdoor[6]. To achieve SSE, one would need both trapdoors at once; indeed the basic Groth-Sahai proofs are sound and zero-knowledge but not SSE as far as we are aware.

With this problem in mind, we do not at the moment have a sensible generalisation of the simulation-sound n-proof game that admits CRSs. The open questions for us are first, whether it would make sense to demand that simulation and extraction trapdoors are simultaneously available for some form of CRS; secondly, whether it would not give the extractor too much power if it were allowed to see both the simulation and extraction trapdoors (which might let the extractor interfere too much with the main prover); thirdly, if the extractor does not get a simulation trapdoor, how it should answer proof queries of rewinding provers. Finally, we are unsure whether rewinding provers serve any purpose outside the ROM — removing rewinding would solve the preceding two questions, at the cost of a slightly weaker notion, but leave the first question open how to simulate and extract in the same experiment). We do not know of any efficient non-ROM schemes that would acheive such a notion, however one chooses to settle these questions.

Acknowledgements. We thank the current and earlier reviewers of this paper for their helpful comments. This work has been supported in part by the European Union under the Seventh Framework Programme (FP7/2007-2013), grant

[6] Ignoring for the moment the distinction between proofs of statements and proofs of knowledge — only the latter support extraction and not all instances of Groth-Sahai proofs have this property.

agreement 609611 (PRACTICE) and the ERC Advanced Grant ERC-2010-AdG-267188-CRIPTO. Marc Fischlin is supported by the Heisenberg grant Fi 940/3-2 of the German Research Foundation (DFG).

References

1. Masayuki Abe. Combining Encryption and Proof of Knowledge in the Random Oracle Model. The Computer Journal, volume 47, number 1, pages 58-70, 2004.
2. Michel Abdalla, Mihir Bellare, and Phillip Rogaway. The Oracle Diffie-Hellman Assumptions and an Analysis of DHIES. CT-RSA, Lecture Notes in Computer Science, volume 2020, Springer, pages 143-158, 2001.
3. B. Adida. Helios: Web-based open-audit voting. In: 17th USENIX security symposium, Pages 335-348, 2008. Helios website: `http://heliosvoting.org` paper: `http://www.usenix.org/events/sec08/tech/full_papers/adida/adida.pdf`
4. P. Baecher, C. Brzuska and M. Fischlin. Notions of Black-Box Reductions, Revisited. In: Advances in Cryptology — Asiacrypt '13, LNCS 8269, pages 296–315, 2013.
5. A. Bagherzandi, J. H. Cheaon and S. Jarecki. Multisignatures Secure under the Discrete Logarithm Assumption and a Generalized Forking Lemma. In: CCS '08, pages 449–458, ACM press 2008.
6. M. Bellare and O. Goldreich. On Defining Proofs of Knowledge. In: Advances in Cryptology (CRYPTO' 92), LNCS 740, pages 390–420, 1992.
7. Mihir Bellare and Oded Goldreich. On Probabilistic versus Deterministic Provers in the Definition of Proofs of Knowledge, Studies in Complexity and Cryptography, Lecture Notes in Computer Science, volume 6650, Springer, pages 114-123, 2011.
8. Mihir Bellare and Oded Goldreich. Strong Proofs of Knowledge, Studies in Complexity and Cryptography, Lecture Notes in Computer Science, volume 6650, Springer, pages 54-58, 2011.
9. Mihir Bellare, Chanathip Namprempre, David Pointcheval, and Michael Semanko. The One-More-RSA-Inversion Problems and the Security of Chaum's Blind Signature Scheme. J. Cryptology, volume 16, number 3, Springer, pages 185-215, 2003.
10. M. Bellare and G. Neven. Multi-signatures in the plain public-key model and a general forking lemma. In: Proceedings of ACM Conference on Computer and Communications Security, pages 390–399, 2006.
11. Mihir Bellare and Phillip Rogaway. Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. ACM Conference on Computer and Communications Security, pages 62-73, ACM, 1993.
12. M. Bellare and P. Rogaway. Code-Based Game-Playing Proofs and the Security of Triple Encryption. Full version of November 27, 2008 (Draft 3.0) at `eprint.iacr.org/2004/331`. Originally published in: Advances in Cryptology (Eurocrypt '06), LNCS 4004, Springer, pages 409–426, 2006.
13. M. Bellare and A. Sahai. Non-Malleable Encryption: Equivalence between Two Notions, and an Indisinguishability-Based Characterization. In: Advances in Cryptology (Crypto '99), LNCS 1666, Springer, pages 519–536, 1999.
14. D. Bernhard, O. Pereira and B. Warinschi. How not to Prove Yourself: Pitfalls of Fiat-Shamir and Applications to Helios. In: Advances in Cryptology — Asiacrypt '12, LNCS 7658, pages 626–643, 2012.

15. M. Blum, P. Feldman, and S. Micali. Non-Interactive Zero-Knowledge and Its Applications. In Proceedings of the twentyeth annual ACM symposium on theory of computing (STOC '90), pages 103-112, 1988.
16. Ran Canetti, Oded Goldreich, Shafi Goldwasser and Silvio Micali. Resettable zero-knowledge, STOC, ACM Press, pages 235–244, 2000.
17. M. Chase and A. Lysyanskaya. On Defining Signatures of Knowledge. In: Advances in Cryptology (Crypto '06), LNCS 4117, pages 78–96, 2006.
18. R. Cramer and V. Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In: Proceedings of the 18th Annual International Cryptology Conference on Advances in Cryptology (CRYPTO '98), pages 13–25, 2008.
19. Ivan Damgard. Towards Practical Public Key Systems Secure Against Chosen Ciphertext Attacks. CRYPTO, Lecture Notes in Computer Science, volume 576, Springer, pages 445-456, 1992.
20. Alfredo De Santis, Giovanni Di Crescenzo, Rafail Ostrovsky, Giuseppe Persiano, and Amit Sahai. Robust Non-interactive Zero Knowledge. CRYPTO, Lecture Notes in Computer Science, volume 2139, Springer, pages 566-598, 2001.
21. Alfredo De Santis and Giuseppe Persiano, Zero-Knowledge Proofs of Knowledge Without Interaction (Extended Abstract), FOCS, pages 427-436, 1992.
22. Yevgeniy Dodis, Kristiyan Haralambiev, Adriana López-Alt, and Daniel Wichs, Efficient Public-Key Cryptography in the Presence of Key Leakage, ASIACRYPT, Lecture Notes in Computer Science, volume 6477, Springer, pages 613-631, 2010.
23. Sebastian Faust, Markulf Kohlweiss, Giorgia Azzurra Marson, and Daniele Venturi. On the Non-malleability of the Fiat-Shamir Transform. Indocrypt, Lecture Notes in Computer Science, Springer, 2012.
24. Uriel Feige, Amos Fiat, and Adi Shamir. Zero-knowledge proofs of identity. Journal of Cryptology, volume 1, number 2, pages 77–94, 1988.
25. Uriel Feige, Dror Lapidot and Adi Shamir. Multiple Non-Interactive Zero Knowledge Proofs Based on a Single Random String (Extended Abstract). FOCS, pages 308-317, 1990.
26. Uriel Feige and Adi Shamir. Zero Knowledge Proofs of Knowledge in Two Rounds, CRYPTO, Lecture Notes in Computer Science, volume 435, Springer, pages 526-544, 1989.
27. A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In: Proceedings on advances in cryptology (CRYPTO '86), pages 186–194, 1986.
28. M. Fischlin. Communication-Efficient Non-Interactive Proofs of Knowledge with Online Extractors. In: Proceedings of the 25th annual international cryptology conference on advances in cryptology (CRYPTO '05), pages 152–168, 2005.
29. M. Fischlin and N. Fleischhacker. Limitations of the Meta-Reduction Technique: The Case of Schnorr Signatures. EUROCRYPT, Lecture Notes in Computer Science, Springer, 2013.
30. P. Fouque and D. Pointcheval. Threshold Cryptosystems Secure against Chosen-Ciphertext Attacks. In: Advances in Cryptology (Asiacrypt '01), LNCS 2248, pages 351–368, 2001.
31. Eiichiro Fujisaki, Tatsuaki Okamoto. Secure Integration of Asymmetric and Symmetric Encryption Schemes. J. Cryptology 26(1), pages 80–101, 2013.
32. Juan A. Garay, Philip D. MacKenzie, and Ke Yang. Strengthening Zero-Knowledge Protocols Using Signatures. J. Cryptology, volume 19, number 2, Springer, pages 169-209, 2006.

33. S. Garg, R. Bhaskar, and S.V. Lokam. Improved bounds on security reductions for discrete log based signatures. CRYPTO, volume 5157, Lecture Notes in Computer Science, Springer, pages 93–107, 2008.
34. Shafi Goldwasser, Silvio Micali and Charles Rackoff. The Knowledge Complexity of Interactive Proof Systems. SIAM J. Comput., volume 18, number 1, pages 186–208, 1989.
35. Oded Goldreich, Rafail Ostrovsky. Software Protection and Simulation on Oblivious RAMs. J. ACM 43(3), pages 431–473, 1996.
36. Jens Groth. Simulation-Sound NIZK Proofs for a Practical Language and Constant Size Group Signatures. ASIACRYPT, Lecture Notes in Computer Science, volume 4284, Springer, pages 444-459, 2006.
37. Eike Kiltz, John Malone-Lee. A General Construction of IND-CCA2 Secure Public Key Encryption. IMA Int. Conf., pages 152–166, 2003.
38. Yehuda Lindell. A Simpler Construction of CCA2-Secure Public-KeyEncryption under General Assumptions. J. Cryptology, volume 19, number 3, pages 359-377, Springer, 2006.
39. M. Naor and M. Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In: Proceedings of the twenty-second annual ACM symposium on theory of computing (STOC '90), pages 42–437, 1990.
40. P. Paillier, P and D. Vergnaud. Discrete-log-based signatures may not be equivalent to discrete log. ASIACRYPT, volume 3788, Lecture Notes in Computer Science, Springer, pages 1–20, 2005.
41. Rafael Pass. On Deniability in the Common Reference String and Random Oracle Model. CRYPTO, Lecture Notes in Computer Science, volume 2729, Springer, pages 316-337, 2003.
42. David Pointcheval and Jacques Stern. Security Arguments for Digital Signatures and Blind Signatures. J. Cryptolog, volume 13, number 3, Springer, pages 361-396, 2000.
43. Charles Rackoff and Daniel R. Simon. Non-Interactive Zero-Knowledge Proof of Knowledge and Chosen Ciphertext Attack. CRYPTO, Lecture Notes in Computer Science, volume 576, Springer, pages 433-444, 1991.
44. A. Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In: Proceedings of the 40th annual symposium on foundations of computer science (FOCS '99), pages 543–553, 1999.
45. C.P. Schnorr. Efficient signature generation for smart cards. In: Journal of cryptology, Volume 4, Springer, pages 161–174, 1991.
46. C.P. Schnorr and M. Jakobsson. Security of Signed ElGamal Encryption. In: Proceedings of the 6th International Conference on the Theory and Application of Cryptology and Information Security: Advances in Cryptology (ASIACRYPT '00), Springer, pages 73–89, 2000.
47. Y. Seurin. On the exact security of Schnorr-type signatures in the random oracle model. EUROCRYPT, volume 7237, Lecture Notes in Computer Science, Springer, pages 554–571, 2012.
48. Y. Seurin and J. Treger. A Robust and Plaintext-Aware Variant of Signed ElGamal Encryption. CT-RSA, Lecture Notes in Computer Science, volume 7779, Springer, pages 68-83, 2013.
49. V. Shoup. A Proposal for an ISO Standard for Public Key Encryption. Version 2.1. Available at `www.shoup.net`, 2001.
50. V. Shoup and R. Gennaro. Securing Threshold Cryptosystems Agains Chosen-Ciphertext Attack. In: Advances in Cryptology (Eurocrypt '98), LNCS 1403, pages 1–16, 1998.

51. Victor Shoup and Rosario Gennaro. Securing Threshold Cryptosystems against Chosen Ciphertext Attack. J. Cryptology, volume 15, number 2, Springer, pages 75-96, 2002.
52. Martin Tompa and Heather Woll. Random Self-Reducibility and Zero Knowledge Interactive Proofs of Possession of Information, FOCS, pages 472–482, 1987.
53. Y. Tsiounis and M. Yung. On the security of ElGamal-based encryption. In: International Workshop on Practice and Theory in Public Key Cryptography (PKC '98), pages 117–134, 1998.
54. Hoeteck Wee. Zero Knowledge in the Random Oracle Model, Revisited. ASIACRYPT, Lecture Notes in Computer Science, volume 5912, Springer, pages 417-434, 2009.
55. J. Groth and A. Sahai. Efficient non-interactive proof systems for bilinear groups. In: Advances in Cryptology (Eurocrypt '08), LNCS 4965, pages 415–432, 2008.
56. D. Wikström. Simplified Submission of Inputs to Protocols. In: Security and Cryptography for Networks, 6th International Conference, SCN 2008, pages 293–308, 2008.
57. Yuliang Zheng and Jennifer Seberry. Practical Approaches to Attaining Security against Adaptively Chosen Ciphertext Attacks (Extended Abstract). In Advances in Cryptology (Crypto '92), pages 292–304, Springer, 1992.

A Comparison with Previous Work

We briefly survey the results most closely related to this paper.

Non-interactive zero-knowledge proofs. Non-interactive zero-knowledge (NIZK) proof schemes can be classified as common reference string (CRS) or random oracle model (ROM) schemes.[7] In the ROM one can further distinguish between rewinding and straight-line extractors; CRS extractors are always straight-line. One can further distinguish proofs of knowledge from proofs of membership.

Historically, CRS proof schemes were discovered first and the theory of NIZK schemes has mainly concerned itself with the CRS model. However, the only NIZK proofs for which we are aware of practical deployments are ROM based, as they are more efficient than CRS ones. Due to the non-rewinding extraction strategy for proofs in the CRS model, some ideas which work on CRS NIZK schemes do not carry over to ROM proofs. This paper studies NIZK proofs of knowledge in the random oracle model directly.

NIZK and CCA. Many authors have studied the use of NIZK proofs to obtain CCA security [15,39,21,43,19,57,28,14]. Encrypt-then-prove goes back to the works of Rackoff and Simon [43] and DeSantis and Persiano [21]. The latter define simulation soundness for proofs of knowledge and sketch a proof that encrypt-then-prove with such proofs yields CCA in the CRS model. The folklore statement that "simulation-sound NIZK implies CCA security" presumably relates to the same work.

Naor and Yung [39] presented a double-encryption construction using proofs of membership. Sahai [44] introduced simulation soundness for proofs of membership and proved the Naor-Yung construction CCA-secure with such proofs.

[7] Some schemes use both a CRS and a random oracle.

However, the Naor-Yung paradigm encrypts each message twice in a ciphertext, which is not the encrypt-then-prove construction as we define it. The Naor-Yung construction and many of its refinements [20,38,32,22] also reside in the CRS model.

The first adaptive notion for NIZK was proposed by Groth [36] and by Chase and Lysanskaya [17] in the related context of signatures of knowledge, again in the common reference string model. Dodis et al. [22] showed that a relaxation to true-simulation extractability of non-interactive proofs, where the adversary only sees proofs of true statements, suffices to achieve CCA security in the common reference string model.

ROM proof schemes. Fouque and Pointcheval [30] were the first to define simulation soundness in the random oracle model. Their definition applies to proofs of membership and they transport the Naor-Yung construction from the CRS to the ROM model, where it remains CCA secure. The first formal definitions of simulation sound proofs of knowledge in the random oracle model were concurrently given by Bernhard et al. [14] and Faust et al. [23]; both works show that proofs derived via the Fiat-Shamir transformation meet this notion. Faust et al. [23] also defined simulation sound extractability in the random oracle model. None of these notions are adaptive.

Signed ElGamal. Signed ElGamal is simple, elegant and efficient. It is used in the Helios voting system [3] which is, for example, deployed by the IACR. It is an encrypt-then-prove construction so "folklore" suggests it should be CCA secure. If this were proven, Signed ElGamal would be the best known encryption scheme as it outperforms all other known CCA encryption schemes. The best positive result for Signed ElGamal in the random oracle model to date is due to Bernhard et al. [14] who recently showed that the encrypt-then-prove construction based on Fiat-Shamir proofs is NM-CPA.

IND-CCA security results [53,46] for Signed ElGamal rely on random oracles and either "knowledge assumptions" or the generic group model. A proof of security that relies on the Gap-Diffie-Hellman assumption had been provided by Abe [1], but for a hashed ElGamal variant with Schnorr signatures where parts of the ElGamal ciphertext is now hashed. This demolishes the homomorphic property of the ElGamal encryption, inhibiting further applications like for example Wikström's submission-security [56] which is implicitly used in Helios. The same argument applies to ECIES [49] and DHIES [2].

Yet another variation is to combine ElGamal encryption with other proofs like Chaum-Pedersen signatures, thereby resurrecting chosen-ciphertext security in the random oracle model [48]. In the same paper, Seurin and Treger proved that Signed ElGamal is not plaintext-aware and cannot have a straight-line extractor. This does not resolve the CCA question, because a rewinding, "plaintext-unaware" strategy may still exist. Yet it suggests that a proof along the lines of the ones for other, known CCA secure schemes is impossible and that rewinding is "inherent" in Signed ElGamal. It also proves that the generic-group extractor [46] has no equivalent without the generic group model, as said extractor is straight-line.

B Detailed Definitions

All literature on proofs of knowledge somehow mentions that they operate over NP relations but the degree of formality varies widely. This is not a bad thing: the key points of an argument concerning PoKs can often be understood better without getting lost in details of the execution model and the exact formulation of polynomial-time algorithms. Nonetheless, for the interested reader we present here a highly formal definition of the objects that we consider in this paper.

For example, what exactly does Schnorr's proof scheme prove? We know that it proves knowledge of discrete logarithms in a group and that like all good proof schemes, it is based on a class NP relation. We would expect a relation of the form $R = \{(X, x) \mid g^x = X\}$. For a fixed finite group $\mathbb{G}$, this relation (viewed as a set) is finite however so there is always an algorithm for taking discrete logarithms with a constant overhead of $|\mathbb{G}|$ — formally, such a relation is class P! The common cryptographic approach is to consider a group generator algorithm $\mathsf{GrpSetup}$ that takes a security parameter λ (in unary encoding 1^λ) as input and outputs a group $\mathbb{G}(\lambda)$. Although not commonly spelt out this formally, the NP relation in question is then the union $R = \bigcup_{\lambda \in \mathbb{N}} R(\lambda)$ over the relations for all the groups in the support of the generator algorithm.

This is fine until we come to define a dishonest prover, against which an extractor must succeed (whether for single-use or adaptive PoKs). The statement should read something like "for every prover who produces a valid statement/proof pair, the extractor can extract a witness". We define a valid pair to be a pair on which the verification algorithm $\mathcal{V}$ outputs "true". However, the correctness condition is usually formulated as "for any statement/witness pair in the relation R, running the honest prover $\mathcal{P}$ on this pair produces a proof that verifies (w.r.t. the statement)". Taking these statements together, if we let our NP relation be the union over the support of the group generator algorithm then we give our dishonest provers the ability to choose a group (and security parameter) at runtime; adaptive provers would even be able to vary the group between proofs. This is not the definition that we are looking for.

In what follows, we give a formal definition of NP relations and languages and proof schemes that avoids these problems.

Cryptographic groups and generators. A cryptographic group $\mathbb{G}$ is a group of prime order together with parameters including the group order q and a distinguished generator g. We re-use the notation $\mathbb{G}$ for the carrier set of the group where no confusion can arise. For a particular group $\mathbb{G}$, a local relation $R'(\mathbb{G})$ is any relation that uses the "interface" of a cryptographic group. For example, the local relation for discrete logarithms is $R'(\mathbb{G}) := \{(X, x) \mid X \in \mathbb{G} \wedge 0 \leq x < q \wedge X = g^x\}$. To be really formal, R' is a map[8] from groups to relations.

A group generator is a map GrpSetup from $\mathbb{N}$ to the class of cryptographic groups. Its input is usually called a security parameter and denoted λ; the usual

[8] We prefer the map notation over subscripts since this avoids having to subscript the subscripts when we introduce security parameters.

convention is that security parameters are provided in a unary encoding and written 1^λ so that the generator algorithm becomes efficient in the usual sense. We write $\mathbb{G}(\lambda)$ for a group output by the generator algorithm on input 1^λ. For a particular group generator, one can consider the relation $R := \bigcup_{\lambda\in\mathbb{N}} R'(\mathbb{G}(\lambda))$. For a randomised group generator, one would include a second union over all random coins of the generator so that R covers the entire range of the generator.

NP RELATIONS AND LANGUAGES. Let $\Sigma = \{0,1\}$ and $\Sigma^* := \bigcup_{k\in\mathbb{N}_0} \Sigma^k$. A binary relation $R \subseteq \Sigma^* \times \Sigma^*$ is an NP-relation if it is decidable in polynomial time in the size of the first argument. An NP relation R defines the language $\mathcal{L}_R = \{x \mid \exists w : (x,w) \in R\}$; if $x \in \mathcal{L}_R$ and w is such that $(x,w) \in R$ we say that w is a witness for $x \in \mathcal{L}_R$.

For a group generator $\mathsf{GrpSetup}$ and a local relation R' on cryptographic groups, the relation R obtained as described in the previous section is of class NP if there is a polynomial p such that for every group $\mathbb{G}$, the following conditions hold.

1. Let $\mathcal{L}'(\mathbb{G}) := \{X \mid \exists w\ (X,w) \in R'(\mathbb{G})\}$ be the induced (localised) language over $\mathbb{G}$. Then every $X \in \mathcal{L}'(\mathbb{G})$ has an element w with $|w| \le p(|X|)$ such that $R'(\mathbb{G})(X,w)$ holds.
2. There is an algorithm that on input a group $\mathbb{G}$ and a pair (X,w) decides $R'(\mathbb{G})$ in at most $p(|X|)$ steps.

Taking a group as input means that the algorithm can obtain the group order and distinguished generator and can perform group operations and inversions for a cost of one step each.

PROOF SCHEMES. A proof scheme for a (local) relation over cryptographic groups is formally a triple $(R', \mathcal{P}, \mathcal{V})$ where all components are maps from the class of cryptographic groups to local relations resp. algorithms over these groups. In other words, the relation $R'(\mathbb{G})$ and the algorithms $\mathcal{P}(\mathbb{G}), \mathcal{V}(\mathbb{G})$ may use the group order q, the distinguished generator g and group operations and inversions. It is a fact of basic Algebra that this allows one to compute a map $\mathbb{Z} \times \mathbb{G} \to \mathbb{G} : (z,h) \mapsto h^z$ which is a group isomorphism for any fixed h and $0 \le z < q$.

The correctness condition too can be formulated locally: for any group $\mathbb{G}$, if $(X,x) \in R'(\mathbb{G})$ and $\pi \leftarrow \mathcal{P}(\mathbb{G})(X,x)$ then $\mathcal{V}(\mathbb{G})(X,\pi)$ should return "true". Note that at this stage one could demand that the relation is polytime decidable in the length of its first argument alone (for some polynomial p fixed once for all groups) but it does not make sense to talk of an NP relation yet.

For example, the Fiat-Shamir-Schnorr proof scheme has the following algorithms. The algorithms are "generic" as they operate over any cryptographic group. This does not prevent the algorithms from, i.e. acting differently for particular choices of q but it does mean that details of the representation of the underlying group (such as whether it is implemented as a subgroup of $\mathbb{Z}_p^\times$ or the group of rational points on an elliptic curve) are abstracted away.

$\mathcal{P}(X,x)$**:** $a \leftarrow_\$ \mathbb{Z}_q; A \leftarrow g^a; c \leftarrow \mathcal{H}(X,A); s \leftarrow a + c \cdot x \pmod q; \pi \leftarrow (A,s)$; return π.

$\mathcal{V}(X, \pi)$: $(A, s) \leftarrow \pi; c \leftarrow \mathcal{H}(X, A)$; If $g^s = A \cdot X^c$ then return "true" else return "false".

PROOFS OF KNOWLEDGE. To define asymptotic security for proofs of knowledge, the objects we consider are formally 4-tuples $(\mathsf{GrpSetup}, R', \mathcal{P}, \mathcal{V})$ consisting of a group generator algorithm, a local relation and generic prover and verifier algorithms. For such a tuple and an extractor $\mathcal{K}$ which is formally a map from the class of cryptographic groups to algorithms over these groups, one can define the extraction probability $\eta(\mathbb{G}, \widehat{\mathcal{P}}, \mathcal{K})$ as the probability that $\mathcal{K}(\mathbb{G})$ wins the proof of knowledge game against $\widehat{\mathcal{P}}(\mathbb{G})$ over group $\mathbb{G}$.

We say that the (maps from groups to) algorithms $(\mathcal{P}, \mathcal{V})$ are a proof of knowledge for the local relation R' with respect to the group generator $\mathsf{GrpSetup}$ if for any $\lambda \in \mathbb{N}$ the following condition holds: there is an efficient (randomised, strict polynomial-time in the size of its input) extractor $\mathcal{K}$ such that for any efficient prover $\widehat{\mathcal{P}}$ the quantity $PoK(\lambda) := \eta(\mathsf{GrpSetup}(1^\lambda), \widehat{\mathcal{P}}, \mathcal{K})$ is overwhelming as a function of λ. Here the probability is taken over all "coins" involved in the experiment, including those of the group generator.

This definition guarantees that we can first fix a group (by picking a value for λ) and then run an experiment in which the prover and extractor are constrained to this group while still obtaining a valid asymptotic definition in which we can ask for a quantity to be negligible (namely the probability that the extractor loses).

THE ONE-MORE DISCRETE LOGARITHM PROBLEM. Formally, we consider the experiment $\mathbf{Exp}^{\mathsf{omdl}}_{\mathcal{A},\mathbb{G}}(\lambda)$ defined in Figure 10. The notation follows the code-based game-playing model of Bellare and Rogaway [12]: there are procedures called initialise and finalise and oracles called chal and dlog. The experiment begins with the game running the initialisation algorithm and handing its return value to the adversary. The adversary may then call the two oracles in any order and as many times as she wishes before she returns an output value which is passed to the finalisation algorithm — in our case a set of candidate discrete logarithms for the challenges. The output of the finalisation is taken to be the game result. The adversary's aim is to make the game result beome 1 and the OMDL assumption holds in a family of groups if no efficient adversary can make the game return 1 with more than negligible probability.

The experiment involves an adversary $\mathcal{A}$ that works against group $\mathbb{G}$ defined by $\mathsf{GrpSetup}$. The adversary has access to two oracles, one that provides fresh random group elements, and one that returns the discrete logarithm for aribtrary group elements. For the OMDL experiment, the game output is the boolean value that indicates whether the number of challenges obtained from the challenge oracle is strictly greater than the number of discrete logarithm queries made, and that the adversary has returned the discrete logarithms of all of the challenges she has received.

For an adversary $\mathcal{A}$ we define its advantage against the one more discrete logarithm problem by $\mathbf{Adv}^{\mathsf{omdl}}_{\mathcal{A},\mathbb{G}}(\lambda) = \mathbf{Pr}[\mathbf{Exp}^{\mathsf{omdl}}_{\mathcal{A},\mathbb{G}}(\lambda) = 1]$ and we say that the

problem is hard with respect to a group generator $\mathsf{GrpSetup}$ if for any efficient adversary $\mathcal{A}$, its advantage is negligible.

C Fischlin's Simulation-Sound Adaptive Proof

The results of our paper show that the most popular way to construct proofs of knowledge in the random oracle model (the Fiat-Shamir transformation) may not lead to non-interactive proofs of knowledge that offer all the guarantees that are desired and expected. A natural question is whether one can avoid the rewinding extractors which led to the problems that we have shown. At Crypto 2005, Fischlin [28] presented an alternative to the Fiat-Shamir transformation with an *online extractor*, which fulfils this requirement. It is intuitively clear that proof systems with non-rewinding extractors yield adaptive proofs. We confirm this intuition for the construction of [28] which we actually show to also be simulation sound adaptive proof.

The Protocol. The basic idea of Fischlin's transformation is to select the verifier's challenge in a Σ-protocol such that a number of low-order bits in the hash of the proof are all zero, or at least "small enough". This forces the prover to ask repeated values of potential challenges to the random oracle until he finds one with a suitable response; from any two such queries, one can extract a witness using special soundness.

We begin with a Σ-protocol with a challenge space of ℓ bits size, such that $\ell = \mathcal{O}(\log \lambda)$ where λ is the security parameter. We can associate the following algorithms to a Σ-protocol:

Commit takes some inputs and produces a commitment $comm$ and auxiliary information aux (to generate the response later). This algorithm models what the honest prover does at the beginning of the protocol.

$\mathsf{Respond}(ch, aux)$ produces the response that the honest prover gives after receiving challenge ch. All other information that he needs is contained in aux. We assume that $\mathsf{Respond}$ is deterministic.

$\mathsf{Verify}(y, comm, ch, resp)$ verifies a protocol execution with respect to some initial input y. This algorithm is run by the verifier at the end of the protocol. We assume that Verify is deterministic.

The transformation relies on parameters b, r, S, t subject to the following conditions.

b is the bit-size of the range of a random oracle $H : \{0,1\}^* \to \{0,1\}^b$. We require $b = \mathcal{O}(\log \lambda)$.

r is the number of repetitions of the basic Σ-protocol. We require $r = \mathcal{O}(\log \lambda)$ and $b \cdot r = \omega(\log \lambda)$.

S is the maximum sum of hash values for an accepting proof. We require $S = \mathcal{O}(r)$.

t is the size of challenges. We require $t = \mathcal{O}(\log \lambda)$, $b \leq t \leq \ell$ and $2^{t-b} = \omega(\log \lambda)$.

The prover and verifier execute the following algorithms which we write $(\mathsf{Prove}, \mathsf{Verify})$ instead of $(\mathcal{P}, \mathcal{V})$ in this section, to be consistent with the original. By $i \leftarrow \mathsf{argmin}_{j \in I} f(j)$ we mean iterate over all values of I and set i to be the value of $j \in I$ for which $f(j)$ is minimal. If the minimum is attained for several values, pick the first such one.

procedure Prove (x, w)

100 for $i = 1, \ldots, r$ do

101 $(com_i, aux_i) \leftarrow \mathsf{Commit}(x, w)$

102 for $i = 1, \ldots, r$ do

103 $ch_i \leftarrow \mathsf{argmin}_{j \in [0, 2^t - 1]} H(x, (com_k)_{k=1}^r, i, j, \mathsf{Respond}(j, aux_i))$

104 $resp_i \leftarrow \mathsf{Respond}(ch_i, aux_i)$ // We assumed that $\mathsf{Respond}$ is deterministic so we get the same value as above.

105 $\pi \leftarrow (com_i, ch_i, resp_i)_{i=1}^r$

procedure Verify (x, π)

200 for $i = 1, \ldots, r$ do // Verify each sigma proof individually.

201 if $\mathsf{Verify}(x, com_i, ch_i, resp_i) = 0$ then

202 return 0

203 if $\sum_{i=1}^r H(x, (com_k)_{k=1}^r, i, ch_i, resp_i) \leq S$ then return 1 else return 0

For the security proofs of this scheme we refer the reader to the original paper [28]. This protocol has a straight-line (non-rewinding) extractor: given only the hash queries made by any prover that produced an accepting proof, with overwhelming probability there are two queries $H(x, (com_k)_{k=1}^r, i, ch_i, resp_i)$ and $H(x, (com_k)_{k=1}^r, i, ch_i', resp_i')$ such that $ch_i \neq ch_i'$. If the original Σ-protocol has special soundness and satisfies some property about unique responses, this suffices to compute a witness w efficiently.

Simulation sound adaptive proofs. In this section we argue that Fischlin's construction yields simulation sound adaptive proofs.

Theorem 4. *Let* $(\mathsf{Commit}, \mathsf{Respond}, \mathsf{Verify})$ *be a* Σ*-protocol with special soundness and unique responses. Then the Fischlin transformation of this protocol is a simulation sound adaptive proof.*

Proof. We first show that the system in question is an adaptive proof. The extractor $\mathcal{K}$ stores all hash input/output pairs that the main invocation of the adversary makes. The extractor does not launch any other invocations. When the extractor $\mathcal{K}$ receives an extraction query, it searches for values in the hash list from which it can extract a witness to the given proofs using special soundness and the fact that responses are unique.

Fischlin has showed that for every proof, the probability of finding such values is overwhelming. Therefore, the probability of success remains non-negligible even after polynomially many repetitions, whether in sequence or in parallel. More precisely, let $q(\lambda)$ be a bound on the number of proofs the extractor must

extract from in total. Let $\varepsilon(\lambda)$ be the probability that the extractor fails on a single proof. We know that $\varepsilon(\lambda)$ is negligible, so from some λ_0 onwards we can assume $q(\lambda) < 1/\varepsilon(\lambda)^2$. But $\lim_{\lambda\to\infty}\left(1-1/\lambda^2\right)^\lambda = 1$, so the success probability of the extractor converges to 1. From some λ_0 onwards we can therefore assume that the extractor succeeds with constant (greater than zero) and hence non-negligible probability.[9] To argue simulation soundness, we simply note that the simulator given in the original paper [28] can be used to simulate proofs even in the presence of an extractor, as it only needs to program the random oracle on freshly chosen values with high entropy.

D Proof of Theorem 2

We will prove Theorem 2 by code-based game-playing. This requires us to set up code for all algorithms that are involved.

Security of encryption: an encryption scheme is a triple of algorithms $(\mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ and has the IND-CPA or CCA security properties if for any efficient adversary $\mathcal{A}$, the probability of winning the following games is negligibly close to 1/2. The adversary may call challenge once and RO many times; in the IND-CPA game she may not call decrypt whereas in the CCA game she may do this as often as she likes.

procedure initialise

100 $(pk, sk) \leftarrow \mathsf{KeyGen}()$
101 return pk

procedure finalise (b)

150 return $b = u$ // False if b has not been defined yet (by challenge).

The challenge oracle may only be called once.

procedure challenge (m_0, m_1)

200 $b \leftarrow \{0,1\}$
201 $c^* \leftarrow \mathsf{Encrypt}(pk, m_b)$
202 return c^*

The decryption oracle is only available in the CCA game.

procedure Decrypt (c)

300 if $c = c^*$ then // False if c^* is not defined yet.
301 return $\perp$
302 return $\mathsf{Decrypt}(sk, c)$

Random oracle (ro) queries are handled by a random oracle as defined in the preliminaries.

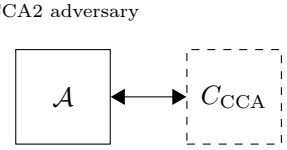
Here is an outline of the proof, which proceeds in six "hops":

[9] The proof so far can be extended to any non-interactive proof scheme with a straight-line extractor that takes the transcript of all queries made by the adversary so far as input and has an overwhelming probability of extraction.

1. Rewrite the challenger as a collection of individual algorithms, so it becomes easier to swap them out (and make an indistinguishability argument).
2. Switch the proof on the challenge ciphertext from a real to a simulated one. This is in preparation for delegating the $\mathfrak{E}$-part of the challenge ciphertext to the IND-CPA challenger, when we will no longer have the witness necessary to make the required proof.
3. Use the extractor $\mathcal{K}$ to answer decryption queries. This means we need to provide implementations of the rewinding provers for $\mathcal{K}$, with all the instances of this oracle sharing the same random coins. This complexity was not present in previous proofs of CCA security since they did not rely on rewinding extractors.
4. Remove the code that passes a witness to the challenge ciphertext between the reduction that creates said ciphertext and the game, which checks the witness. This is a technical step necessary to make the last step work, in which we use the IND-CPA challenger.
5. Coin-splitting. The simulation sound adaptive experiment gives us an extractor that expects access to multiple copies of the same prover, all running with the same random string. We wish to inject the IND-CPA challenge in one copy only (as we only get one challenge query), so we give an argument why this is allowed.
6. Reduce to IND-CPA, now that we have done all the work necessary to make this step possible.

To help the reader through the proof, we present each step in a separate section, following the five-part layout of motivation, new/changed code, justification, diagram and conclusion.

Proof. Suppose that $\mathcal{A}$ is an adversary against the IND-CCA2 property of the transformed scheme.



Our starting point: an adversary which can win the CCA game with some non-negligible advantage. In all the following diagrams, we will draw the "game" against which the adversary has this distinguishing advantage with a dashed border.

Hop 1: Building a Reduction.

We know that $\mathcal{A}$ has non-negligible advantage against C_{CCA}. In a first step, we rewrite the challenger as a collection of different algorithms for the various tasks (key generation, challenge, decryption) controlled by a reduction $\mathcal{R}$. $\mathcal{R}$ will later connect $\mathcal{A}$ and $\mathcal{K}$, reducing the CCA property of the transformed scheme to the simulation sound adaptive property of the proof and the IND-CPA property of the basic encryption scheme, hence the name. This step is purely to make the following steps easier, and does not change the advantage of the adversary $\mathcal{A}$.

Reduction $\mathcal{R}_1$ and algorithms I (for initial setup and key generation), C (for the proof on the challenge) and D (for decryption). Oracles ro, challenge and Decrypt of $\mathcal{A}$ are handled by $\mathcal{R}$ which delegates part of the work to the aforementioned algorithms.

procedure $\mathcal{R}$.initialise

100 $pk \leftarrow I.\mathsf{setup}()$
101 return pk

procedure $\mathcal{R}$.finalise (u)

150 return $b = u$ // False if b has not been defined yet (by challenge).

The challenge oracle may only be called once.

procedure challenge (m_0, m_1)

200 $b \leftarrow \{0, 1\}$
201 $r \leftarrow\!\!\$\ R_{\mathrm{Enc}}$
202 $c^- \leftarrow \mathfrak{E}.\mathsf{Encrypt}(pk, m_b; r)$ // $\mathfrak{E}$ is the IND-CPA encryption scheme.
203 $w^- \leftarrow \mathsf{W}(pk, c^-, m_b, r)$ // w^- is the witness to the statement that we're going to make a proof on, where W is the algorithm that exists by Definition 8 and turns a message/randomness pair into a witness.
204 $\pi^- \leftarrow C.\mathsf{prove}(pk, c^-, w^-)$
205 $c^* \leftarrow (c^-, \pi^-)$
206 return c^*

procedure Decrypt (c)

300 if $c = c^*$ then // False if c^* is not defined yet.
301 return $\perp$
302 $m \leftarrow D(c)$
303 return m

procedure ro (x)

400 return $\mathsf{RO}(x)$

procedure I.setup

600 $(pk, sk) \leftarrow \mathsf{KeyGen}()$
601 return pk

procedure I.keys

650 return (pk, sk)

procedure C.prove (pk, c, w)

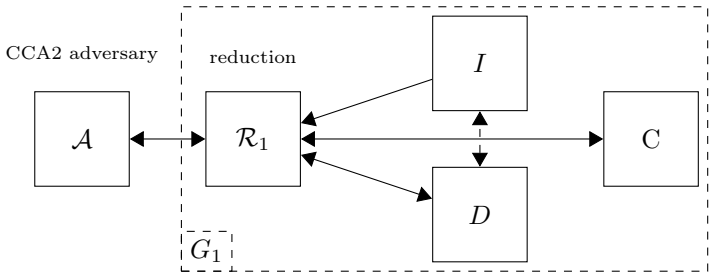
700 return $\mathsf{prove}(pk, c, w)$

Random oracle queries are handled by C using a random oracle.

procedure D (c)

800 $(pk, sk) \leftarrow I.\mathsf{keys}()$
801 return $\mathsf{Decrypt}(sk, c)$

One can easily verify by expanding all calls made by $\mathcal{R}_1$ that all calls from $\mathcal{A}$ are handled exactly the same as by the CCA challenger. In other words, this hop does not change the adversary's advantage.



The first game-hop: rebuilding the challenger into a collection of algorithms which we collectively call game G_1.

Let $\mathbf{Adv}(\mathcal{A}, G_1)$ be the advantage of the adversary $\mathcal{A}$ against G_1, i.e. $1/2$ plus the probability that $\mathcal{A}$ guesses the bit b correctly when interacting with G_1. (The bit b is created by $\mathcal{R}_1$ in line 200.) Let $\mathbf{Adv}(\mathcal{A}, \mathrm{CCA})$ be his advantage against the CCA challenger. Then for all adversaries $\mathcal{A}$, we have shown

$$\mathbf{Adv}(\mathcal{A}, G_1) = \mathbf{Adv}(\mathcal{A}, \mathrm{CCA})$$

Hop 2: Simulating Challenge Proofs.

For our next game-hop, we replace the proof on the challenge ciphertext with a simulated one, in preparation for the time when we no longer have a witness to the challenge ciphertext. Recall that we have a simulator S that responds to calls $\mathsf{prove}(X)$ by simulating a proof π on X, and at the same time manages a random oracle. The simulator expects these queries without a witness. However, to justify this step we will need to reduce to the zero-knowledge experiment, in which we do have to provide witnesses to our simulation queries. We introduce an adapter E that checks and strips these witnesses. (We will get rid of the witnesses completely in a later hop.)

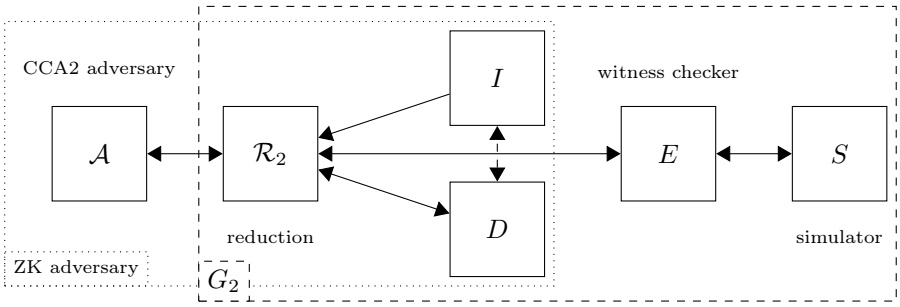
Changes to $\mathcal{R}$ for the second hop, and the new adapter E. In this and all future hops, we underline important changes to the code since the last hop.

procedure $\mathcal{R}.\mathsf{challenge}\ (m_0, m_1)$
200 $b \leftarrow \{0,1\}$
201 $r \leftarrow\!\!\$\ R_{\mathrm{Enc}}$
202 $c^- \leftarrow \mathfrak{E}.\mathsf{Encrypt}(pk, m_b; r)$
203 $w^- \leftarrow \mathsf{W}(pk, c^-, m_b, r)$
204 $\pi^- \leftarrow \underline{E.\mathsf{prove}(pk, c^-, w^-)}$
205 $c^* \leftarrow \underline{(c^-, \pi^-)}$
206 return c^*
procedure $E.\mathsf{prove}\ (pk, c, w)$

```
250   if ¬R_pk(c, w) then
251       return ⊥
252   return S.prove(pk, c)                                   // No w!
procedure R.ro (x)
400   return E.ro(x)
procedure E.ro (x)
450   return S.ro(x)
```

We wish to show that the second game-hop is undetectable from the point of view of the adversary. To do this, we note that the only changes we have made are routing ro calls and the one proof creation in the challenge procedure to S (the latter through E) instead of C.

If we view the ensemble of $\mathcal{A}$, $\mathcal{R}$, I and D as an algorithm with a single interface that can be connected to either E or C (see the figure below), we have essentially built a distinguisher for the zero-knowledge experiment in Definition 2. All we must do now is let this distinguisher output 1 if $\mathcal{A}$ guessed the bit b correctly and 0 otherwise. We omit writing out the code for this argument.



Using the simulator for challenge ciphertexts. The new adapter E checks and strips witnesses from prove queries as required by the ZK game. Everything in the dashed box is the game G_2 against which we show that $\mathcal{A}$ still has a non-negligible advantage; the dotted box can be used as a distinguisher for the zero-knowledge property to justify this step.

Let $\mathbf{Adv}(\mathcal{A}, G_2)$ be the advantage of adversary $\mathcal{A}$ against the above game (still the probability that he guesses b correctly, plus one half). The hop from G_2 to G_3 switches a subsystem that creates real proofs for one that creates simulated proofs. Distinguishing these two subsystems is exactly the zero-knowledge game. So let Δ_{ZK} be the distinguishing advantage against the zero-knowledge game. Then we have shown

$$\mathbf{Adv}(\mathcal{A}, G_1) \leq \mathbf{Adv}(\mathcal{A}, G_2) + \Delta_{\mathrm{ZK}}$$

Hop 3: Decryption.

In this hop, we will introduce $\mathcal{K}$ and use it to answer decryption queries. The construction that we have built so far (minus the simulator) will become

the main prover $\widehat{\mathcal{P}}$ with which $\mathcal{K}$ will interact via the simulation sound adaptive proof game. The extractor $\mathcal{K}$ will be able to invoke further "rewinding" copies of our prover but we do not care about these (yet).

This hop is not a "game-hop" in the usual sense. We do not change a small part of the game's code and argue that this makes only a negligible difference to $\mathcal{A}$. Rather, we completely replace the whole game and argue that it makes no difference at all to $\mathcal{A}$. So while we are still giving the common "game-hop" argument that the new game, i.e. everything but the main invocation of $\mathcal{A}$, is indistinguishable from the old game, due to the different nature we prefer to call the current hop an *environment hop*, as we are placing $\mathcal{A}$ in a completely new environment. Let $\mathcal{R}_3$ be $\mathcal{R}_2$ with the following changes.

Changes to $\mathcal{R}$ for the third hop.
procedure $\mathcal{R}.\mathsf{challenge}\ (m_0, m_1)$
200 $b \leftarrow \{0,1\}$
201 $r \leftarrow\!\!\$\ R_{\mathrm{Enc}}$
202 $c^- \leftarrow \mathfrak{E}.\mathsf{Encrypt}(pk, m_b; r)$
203 $w^- \leftarrow \mathsf{W}(m_b, r)$
204 $\pi^- \leftarrow \underline{\mathsf{prove}(pk, c^-, w^-)}$ // On external interface — will go to adaptive proof game.
205 $c^* \leftarrow (c^-, \pi^-)$
206 return c^*
procedure $\mathsf{ro}\ (x)$
400 return $\underline{\mathsf{ro}(x)}$ // On external interface.
procedure $\mathsf{Decrypt}\ (c)$
300 if $c = c^*$ then // False if c^* is not defined yet.
301 return $\perp$
302 return $\underline{\mathsf{extract}(c)}$ // On external interface.

Let $\widehat{P}$ be the ensemble consisting of $\mathcal{A}$ and $\mathcal{R}_3$, that takes a public key as input (on what was formerly the I interface) and has an external interface on which it can output proof, extraction and RO queries. We consider the simulation sound adaptive proof game of Definition 7 with $\mathcal{K}$ playing against $\widehat{P}$.

The main invocation of $\mathcal{A}$, which will be accessible to $\mathcal{K}$ as part of the main prover, will be provided with access to a simulator by the game. We wish to argue that this main invocation of $\mathcal{A}$ cannot distinguish the third game from the second; we do not care what the extractor does with the other invocations of the prover $\widehat{P}$ and their instances of $\mathcal{A}$. The external interface of $\mathcal{A}$ allows for the following queries.

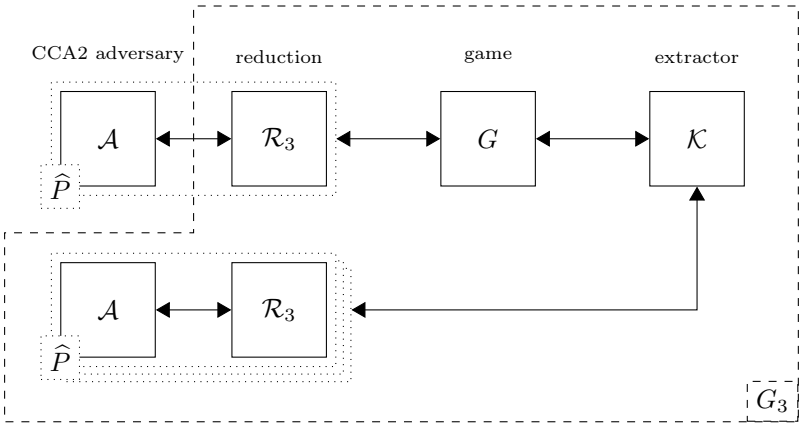
Public key This is now produced by the new reduction, but according to the same distribution as before — the algorithm that ends up creating the key is still KeyGen.

Challenge The only difference in the handling of challenge queries is that the adaptive game's simulation oracle is now in charge of making the simulated proof on the challenge ciphertext. But this oracle just delegates to the simulator, so the proof is still made exactly the same way (with the same distribution) as before.

Decryption Here, we note that a ciphertext can only decrypt to one plaintext. So, there is only one possible answer to a decryption query; therefore it does not matter how this answer is generated by $\mathcal{K}$.

Random Oracle Random oracle queries end up at the simulator S, just like they did in the last game. So the distribution of these does not change either.

In conclusion, all answers to queries by the main invocation of $\mathcal{A}$ are given with the same distributions as they were in the last game-hop.



Connecting the challenger $\mathcal{K}$. The entire previous game (minus S) has become the prover $\widehat{P}$ in the top-left corner; further copies of this prover are available to $\mathcal{K}$ (on the lower row). The game G_3 is still "everything except $\mathcal{A}$", but now that means everything except the main invocation of $\mathcal{A}$.

The main observation is that the view of adversary $\mathcal{A}$ in G_3 is the same as the view of $\mathcal{A}$ in G_2, provided that $\mathcal{K}$ successfully answers all decryption queries.

For a given malicious prover $\widehat{P}$ and extractor $\mathcal{K}$ we write $\mathbf{Adv}^{\mathsf{ss\text{-}zk}}_{\widehat{P},\mathcal{K}}$ for the probability that the adaptive simulation-sound extraction game in Definition 7 does not output "$\mathcal{K}$ wins", that is $\widehat{P}$ manages to produce a proof for which $\mathcal{K}$ cannot extract a witness. We also write $\Pr[G_2(\mathcal{A}) \Rightarrow 1]$ for the probability that $\mathcal{A}$ successfully guesses the challenge bit in G_2, and similarly for G_3. By the assumption that the proof system used in the construction is simulation-sound adaptive proof of knowledge we may assume the existence of a knowledge extractor $\mathcal{K}$ such that $\mathbf{Adv}^{\mathsf{ss\text{-}zk}}_{\widehat{P},\mathcal{K}}$ is negligible. We write $\mathcal{K}$ wins for the event that $\mathcal{K}$ successfully answers all of the queries of malicious prover $\mathcal{P}$ in G_3 and $\mathcal{K}$ wins for the complementary event. Notice that by our construction we have that $\Pr[\mathcal{K}\ \mathsf{loses}] = \mathbf{Adv}^{\mathsf{ss\text{-}zk}}_{\widehat{P},\mathcal{K}}$ and $\Pr[\mathcal{K}\ \mathsf{wins}] = (1 - \mathbf{Adv}^{\mathsf{ss\text{-}zk}}_{\widehat{P},\mathcal{K}})$ (as the execution of the extractor is as in the game for simulation sound adaptive extractability). We have the following:

$$
\begin{aligned}
\mathbf{Adv}(\mathcal{A}, G_3) &= \left| \Pr[G_3(\mathcal{A}) \Rightarrow 1] - \frac{1}{2} \right| \\
&= \left| \Pr[G_3(\mathcal{A}) \Rightarrow 1 \wedge \mathcal{K} \text{ wins}] + \Pr[G_3(\mathcal{A}) \Rightarrow 1 \wedge \mathcal{K} \text{ loses}] - \frac{1}{2} \right| \\
&= \left| \Pr[\mathcal{K} \text{ wins}] \cdot \Pr[G_3(A) \Rightarrow 1 \mid \mathcal{K} \text{ wins}] + \Pr[\mathcal{K} \text{ loses}] \cdot \Pr[G_3(\mathcal{A}) \Rightarrow 1 \mid \mathcal{K} \text{ loses}] - \frac{1}{2} \right| \\
&= \left| (1 - \mathbf{Adv}^{\mathsf{ss\text{-}zk}}_{\widehat{P},\mathcal{K}}) \cdot \Pr[G_3(A) \Rightarrow 1 \mid \mathcal{K} \text{ wins}] + \mathbf{Adv}^{\mathsf{ss\text{-}zk}}_{\widehat{P},\mathcal{K}} \cdot \Pr[G_3(\mathcal{A}) \Rightarrow 1 \mid \mathcal{K} \text{ loses}] - \frac{1}{2} \right| \\
&= \left| \left(\Pr[G_2(\mathcal{A}) \Rightarrow 1] - \frac{1}{2} \right) - \mathbf{Adv}^{\mathsf{ss\text{-}zk}}_{\widehat{P},\mathcal{K}} \cdot (\Pr[G_3(A) \Rightarrow 1 \mid \mathcal{K} \text{ wins}] - \Pr[G_3(A) \Rightarrow 1 \mid \mathcal{K} \text{ wins}]) \right| \\
&= \left| \mathbf{Adv}(\mathcal{A}, G_2) - \mathbf{Adv}^{\mathsf{ss\text{-}zk}}_{\widehat{P},\mathcal{K}} \cdot (\Pr[G_3(A) \Rightarrow 1 \mid \mathcal{K} \text{ wins}] - \Pr[G_3(A) \Rightarrow 1 \mid \mathcal{K} \text{ wins}]) \right|
\end{aligned}
$$

If we write $\Delta_{\text{ss-aPoK}}(\mathcal{A})$ for the negligible function that represents the advantage of $\widehat{P}$ (based on $\mathcal{A}$ against the extractor $\mathcal{K}$ guaranteed by the security of the proof system employed in our construction, i.e. an extractor for which $\Delta_{\text{ss-aPoK}}$ is neligible) we have that

$$
\mathbf{Adv}(A, G_3) \geq \mathbf{Adv}(\mathcal{A}, G_2) - 2 \cdot \Delta_{\text{ss-aPoK}}(\mathcal{A}) \tag{1}
$$

Hop 4: Dropping the Witness.

Currently, when an instance of $\mathcal{A}$ makes a challenge query, $\mathcal{R}_3$ makes a proof query to get a simulated proof on it, but must provide a witness which is checked and then discarded by the adaptive proof game. We change the code of this game so that proof queries do not require a witness any more. We simply drop the witness check. This is legitimate if we can show that the check was never going to fail in the first place. Since the game, apart from checking and stripping witnesses, just forwards proof queries to the simulator, we are not changing the values returned from proof queries so $\mathcal{A}$ and $\mathcal{K}$ will not notice this change.

procedure $\mathcal{R}$.challenge (m_0, m_1)
200 $b \leftarrow \{0,1\}$
201 $r \leftarrow\!\!\$\ R_{\text{Enc}}$
202 $c^- \leftarrow \mathfrak{E}.\mathsf{Encrypt}(pk, m_b; r)$
203 $\underline{\pi^- \leftarrow \mathsf{prove}(pk, c^-)}$
204 $c^* \leftarrow (c^-, \pi^-)$
205 return c^*
procedure G.prove (x)
900 $X' \leftarrow x$
901 send x to $\mathcal{S}$.prove

procedure $G.\mathsf{return\text{-}proof}\ (\pi)$
950 $\quad \Pi \leftarrow \Pi :: (X', x)$ // Called when $\mathcal{S}$ returns π to the game.
951 $\quad$ send π to $\widehat{\mathcal{P}}$

We perform the same change on the auxiliary games $\widehat{G}$ that handle proof queries for the rewinding provers: we drop the witness from the parameter list of the proof queries and remove the witness-checking code. In other words, the prover and extractor now use the same syntax for proof queries: the only parameter is the statement.

This game-hop will not affect $\mathcal{A}$ as long as no prove query is ever made on a false statement, i.e. on an x for which no w exists such that $R(pk, x, w)$ holds. But the only proof query ever made by $\mathcal{R}$ is for the challenge ciphertext, and we can inspect the code of $\mathcal{R}$ to see that it is always well-formed. Let $\mathbf{Adv}(\mathcal{A}, G_4)$ be the advantage of adversary $\mathcal{A}$ against the above game. Then we have

$$\mathbf{Adv}(\mathcal{A}, G_4) = \mathbf{Adv}(\mathcal{A}, G_3)$$

Hop 5: Coin-Splitting.

The simulation sound adaptive proof game says that $\mathcal{K}$ can extract witnesses to proofs as long as he is given access to many copies of the same prover $\widehat{P}$ using the same random string. In our reduction to the IND-CPA challenger for $\mathfrak{E}$ however, we will have access to only one instance of the challenger. We will use our one instance of the challenger to create the challenge for the main invocation of $\mathcal{A}$, as this is the one that we are arguing to have a non-negligible advantage in guessing the challenge bit. This means that all the rewinding copies of $\mathcal{A}$ will not be identical to the main one and we have to justify why this reduction will work.

The argument, which we call *coin-splitting* (as opposed to coin-fixing), relies on two observations. First, if an invocation I of $\widehat{P}$ is doing exactly the same as another invocation J has done earlier (in this case we call I a *prefix* of J), then I can just replay the queries that J has already sent. Secondly, if two invocations ever get different answers to the same query, in which case we say that they have been *forked* at the point where this first happens, then we will show that they can essentially use independent random coins from then onwards and $\mathcal{K}$ will still work.

Informally, the reasoning is as follows. Suppose $\widehat{P}$, instead of using its random coins directly, uses them to seed a PRG and uses the input and output of all queries to update this PRG; when it needs a random coin, it draws it from the PRG. Then the moment two invocations are forked, they behave towards $\mathcal{K}$ as if they were using independent random coins, unless $\mathcal{K}$ can distinguish the PRG from a true random generator.

We leave a general formulation of this principle for future work. For the current paper, we reuse the technique from the metareduction in Appendix E, but instead of using a truly random function Ψ we can work with a pseudorandom

function. The reason is that here we work only with efficient algorithms and could make another game hop to a truly random function, unlike in the case of the OMDL problem where we have a "hidden" superpolynomially powerful support through the DL oracle and cannot perform such a game hop. In addition, we only need it to generate the random coins r and bit b used to produce the challenge ciphertext (essentially, the random coins that the IND-CPA challenger will use in the next game-hop).

This allows us to simulate all copies of the prover with a single reduction which can pick fresh random coins for the challenge queries in each invocation of $\widehat{P}$ that it will simulate, subject to a *prefix rule*. More formally, call an invocation I a *prefix* of another invocation J if the list of all inputs and outputs that I has received so far is a prefix of the list of inputs and outputs that J has received. If two invocations have received identical inputs and outputs, they are both prefixes of each other. Call an invocation I a *strict prefix* of J if I is a prefix of J but J is not a prefix of I. Call a query made by an invocation that is not a strict prefix of any other a *fresh* query.

We make the following changes.

1. All invocations of $\mathcal{R}$ ask a random oracle query $\mathsf{ro}(m_0, m_1)$ on the challenge messages immediately before creating the challenge ciphertext. (We will see in a minute that this exempts the main invocation from the prefix rule below.)
2. All invocations except the main one obey the *prefix rule*: if any such invocation is a prefix of another at the point when it needs to draw a bit b and randomness r for encrypting the challenge message, it draws the same b and r as the invocations of which it is a prefix.
3. If any invocation of $\mathcal{R}$ is not a prefix of any other at the point when it must create a challenge ciphertext, it draws fresh random b and r *independently of all previous invocations.*

We assume that id is the identifier of the current invocation, and that there is a well-defined way to check for "being a prefix". Let $\leftleftarrows\$$ denote sampling fresh randomness independently of all other executions, i.e. from a source distinct from the common random string.

procedure $\mathcal{R}.\mathsf{challenge}\ (m_0, m_1)$

200 $z \leftarrow S.\mathsf{ro}(m_0, m_1)$
201 if $id =$"main" then // If we are the main invocation.
202 $b \leftleftarrows\$ \{0,1\}$
203 $r \leftleftarrows\$ R_{\mathrm{Enc}}$
204 $c^- \leftarrow \mathfrak{E}.\mathsf{Encrypt}(pk, m_b; r)$
205 $challenge[id] \leftarrow c^-$ // Save our choice. id is our identifier.
206 else if $\exists I$: we are a prefix of I then
207 $c^- \leftarrow challenge[I]$ // Replay challenge.
208 else

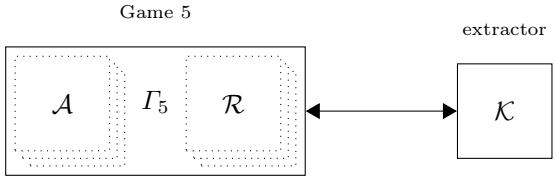
```
209       b ⟸$ {0,1}
210       r ⟸$ R_Enc
211       c⁻ ← 𝔈.Encrypt(pk, m_b; r)
212       challenge[id] ← c⁻
213   π⁻ ← prove(pk, c⁻)   // This is not replayed, as it becomes a call to
      𝒦.
214   c* ← (c⁻, π⁻)
215   return c*
```

We need to prove two points: why the main invocation of $\mathcal{A}$ cannot detect this game-hop and why $\mathcal{K}$ still works despite having to interact with a construction outside the scope of the simulation sound adaptive proof game. However, we do not have to argue that $\mathcal{K}$ cannot detect this game-hop: as long as $\mathcal{K}$ "believes" that he is interacting with some valid multi-prover $\widehat{P}$, we do not mind if $\mathcal{K}$ has a different view to the last game as $\mathcal{K}$ must work for all adaptive provers.

We have not changed the reduction's interaction with the main invocation of $\mathcal{A}$ at all (we can assume that $\mathcal{R}$ discards the result of the extra RO query), so we are fine here. For the second argument, assume for a moment that $\mathcal{R}$ internally had a PRF F with a seed drawn from his random coins, and that $\mathcal{R}$ created the random values b and r by applying the PRF to the transcript of all previous inputs and outputs to S and $\mathcal{K}$ so far. Consider the second point in time when an invocation in this game makes a challenge query and label this invocation J; call the invocation that previously made its challenge query I. An analogous argument applies to all further challenge queries after the second one. Three cases can occur:

- J is not the main invocation, and is a prefix of I at this point in time.
  In this case, by the prefix rule, J chooses the same challenge ciphertext as I. This is what would happen too if $\mathcal{K}$ were interacting with multiple copies of the same multi-prover $\widehat{P}$.
- J is not the main invocation, and is not a prefix of I when it makes its challenge query.
  In this case, there must have been some point earlier in time when J "forked" from I because it got a different response to I on one of its queries. (Otherwise, since I has already made its challenge query, J would be a prefix of I.)
  In this case, in the thought experiment with the PRF, the result would be that J draws b and r that are indistinguishable to $\mathcal{K}$ from fresh random values created *independently* from the values used by I, despite being a "clone" of I with the same random coins.
- J is the main invocation.
  In this case we claim that with overwhelming probability, J cannot be a prefix of any other invocation due to the extra random oracle query inserted just before making the challenge ciphertext. Suppose that J got a value z from the oracle in response to this query. z came from the simulator, so $\mathcal{K}$

has no control over it. In fact, if $\mathcal{K}$ could distinguish z from a fresh random value, then the entire construction minus S would be able to distinguish S from a random oracle and honest prover, breaking the ZK game.
Therefore, when $\mathcal{K}$ answered the same query for I, he must have chosen a response z' where $z = z'$ holds only with negligible probability. So with overwhelming probability, J is not a prefix of I and is free to choose a fresh challenge ciphertext by the argument above.



We assume the reader is familiar by now with the idea of G_5 being everything except the main invocation of $\mathcal{A}$. We narrow our view a bit and let Γ_5 be everything that is connected to the simulation sound adaptive proof game. Γ_5 interacts with $\mathcal{K}$, but unlike in the game definition all prover copies are handled by the same algorithm Γ_5 (let us assume that some convention for addressing is given) that uses some joint state and randomness between the instances of $\widehat{P}$ that it is simulating.

Let $\mathbf{Adv}(\mathcal{A}, G_5)$ be the advantage of the main invocation of the adversary in guessing the random bit of the main invocation of $\mathcal{R}$, in the game consisting of "everything except the main invocation of the adversary" as usual. Let $\Delta'_{\mathrm{ZK}}(\mathcal{A})$ be the distinguishing advantage of this whole construction minus the game's simulator against the ZK game. This is at most the distinguishing advantage Δ_{ZK} of any efficient algorithm against the ZK game. Then we have

$$\mathbf{Adv}(\mathcal{A}, G_5) \geq \mathbf{Adv}(\mathcal{A}, G_4) - \Delta_{\mathrm{ZK}}$$

Hop 6: Reduction to IND-CPA.

We can finally construct an adversary against the IND-CPA challenger of $\mathfrak{E}$. First, we obtain a public key from this challenger and use it to start our latest game, which differs from the previous one in that the reduction $\mathcal{R}$ (which is now finally reducing something) for the main invocation gets the $\mathfrak{E}$-part of the challenge ciphertext from the IND-CPA challenger. If the main invocation of the adversary outputs a bit u, we stop our construction and return this bit to the challenger.

We have already established that the main invocation cannot be a prefix of another at the point when it asks a challenge query; if another invocation is a prefix of the main one when asking its challenge query we can simply replay the same challenge ciphertext without involving the IND-CPA challenger again. The challenge ciphertexts for all other invocations are still created without involving the new challenger.

procedure $\mathcal{R}$.challenge (m_0, m_1)

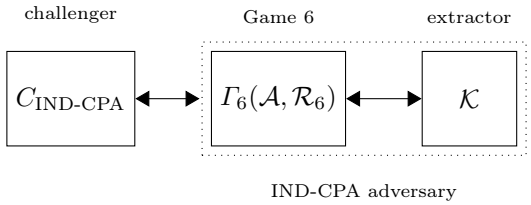
200 $z \leftarrow S.\mathsf{RO}(m_0, m_1)$
201 if $id =$"main" then // If we are the main invocation.
202 $c^- \leftarrow C_{\text{IND-CPA}}.\mathsf{challenge}(m_0, m_1)$
203 $challenge[id] \leftarrow c^-$
204 else if $\exists I$: we are a prefix of I then
205 $c^- \leftarrow challenge[I]$
206 else
207 $b \leftarrow\!\!\$ \{0, 1\}$
208 $r \leftarrow\!\!\$ R_{\text{Enc}}$
209 $c^- \leftarrow \mathfrak{E}.\mathsf{Encrypt}(pk, m_b; r)$
210 $challenge[id] \leftarrow c^-$
211 $\pi^- \leftarrow \mathsf{prove}(pk, c^-)$
212 $c^* \leftarrow (c^-, \pi^-)$
213 return c^*

Algorithm bit is triggered when the main invocation of $\mathcal{A}$ outputs his final bit.

procedure bit (u)

950 return u // Triggers $C_{\text{IND-CPA}}.\mathsf{finalise}\ (u)$ which ends the execution.

As usual, we need to argue that the main invocation of $\mathcal{A}$ cannot detect a difference between this game and the last. This is easy however: we know the code of the IND-CPA challenger (at the beginning of this section) and that it creates the challenge ciphertext in exactly the same way as $\mathcal{R}$ used to.



The reduction to IND-CPA of $\mathfrak{E}$. We have shown that the advantage of the main invocation of $\mathcal{A}$ (inside Γ_6) is still non-negligible, and how it relates to the advantage that $\mathcal{A}$ had against the original CCA challenger.

With the usual notation and with $\mathbf{Adv}_{\text{CPA}}(\mathcal{A})$ the advantage of this whole construction against the IND-CPA challenger, we have

$$\mathbf{Adv}_{\text{CPA}}(\mathcal{A}) = \mathbf{Adv}(\mathcal{A}, G_5)$$

and conclude

$$\mathbf{Adv}_{\text{CPA}}(\mathcal{A}) \geq \mathbf{Adv}_{\text{CCA}}(\mathcal{A}) - (\Delta_{\text{ZK}}(\mathcal{A}) + \Delta'_{\text{ZK}}(\mathcal{A}) + 2 \cdot \Delta_{\text{ss-aPoK}}(\mathcal{A})))$$

which yields the security of the encrypt-then-prove construction.

SUMMARY. We have shown that if $\mathfrak{E}$ is IND-CPA secure and the proof scheme $\mathfrak{P}$ is a zero-knowledge simulation sound adaptive proof then the encrypt-then-prove transformation has CCA security.

The technical tools that we needed in the proof were the zero-knowledge property of $\mathfrak{P}$ to turn an $\mathfrak{E}$-challenge into an encrypt-then-prove one, the simulation sound adaptive property to get the extractor $\mathcal{K}$ to answer decryption queries for us (with the simulation sound part allowing us to use the aforementioned simulated proof) and the coin-splitting paradigm to allow us to inject a single challenge query into what looks like multiple copies of the same algorithm running with the same random coins.

The main difficulties, and the reason that we cannot at the moment run the entire proof using code-based game-playing, is that we do not know either the code of $\mathcal{K}$ nor the order in which it will launch and respond to queries of different invocations of our reduction, neither is the number of such invocations a priori bounded. We see it as an open problem for future research to come up with an analogue of the code-based game-playing framework [12] with syntax and semantics for "token-based concurrent" execution of algorithms and a method to construct proofs such as ours in such a framework, where the main game-hops are not so much changing code within as moving code between different algorithms.

E Proof of Theorem 3

This section is devoted to a proof that Fiat-Shamir-Schnorr is not an adaptively secure proof scheme in the ROM under the one-more discrete logarithm (OMDL) assumption. The general idea of the proof can be carried over to any Fiat-Shamir transformed Sigma protocol; the security assumption becomes a one-more version of the one-wayness assumption on the underlying homomorphism.

OVERVIEW.

We begin by defining a particiular adversary $\hat{P}$ which makes a chain of n proofs following the ideas of Shoup and Gennaro [50]. Our adversary uses a random function Ψ but this is only for illustration purposes (one could also replace Ψ with a pseudorandom function).

Next, we construct a reduction $\mathcal{R}$ which simulates multiple copies of the prover $\hat{P}$. We claim that no extractor $\mathcal{K}$ in the adaptive proof game can distinguish whether it is interacting with $\mathcal{R}$ or multiple copies of $\hat{P}$. The reduction $\mathcal{R}$ does not require a random function. $\mathcal{R}$ reduces to the one-more discrete logarithm problem and the main challenge in this step is ensuring that $\mathcal{R}$ can simulate a potentially unbounded number of copies of $\hat{P}$ using only one OMDL challenger.

In the adaptive proof game between $\mathcal{K}$ and $\mathcal{R}$ (the latter playing all copies of the adversary at once), we define an event NZP. If this event occurs, $\mathcal{R}$ can solve the one-more discrete logarithm instance that is is interacting with.

For any execution of the game that $\mathcal{K}$ wins against $\mathcal{R}$ without event NZP occuring, we show via a combinatorial argument that $\mathcal{K}$ must have interacted

with at least 2^n copies of the adversary (simulated by $\mathcal{R}$). Since an efficient extractor (in the sense of the adaptive proof definition) must win the game with overwhelming probability but cannot take exponential time, we conclude that such an extractor always triggers event NZPand therefore we have an efficient OMDL adversary which always succeeds.

Construction.

In Figure 11 we give the prover $\hat{P}$ and a "hypothetical" OMDL reduction $\mathcal{A}_n$. We cannot use this reduction directly as the extractor has the ability to rewind its adversary, but a reduction cannot rewind the challenger. The reduction $\mathcal{A}_n$ is intended only as an illustration of how our reduction $\mathcal{R}$ will work later.

The prover $\hat{P}$ runs two loops. First, it builds up a chain of n Schnorr statement/proof pairs by picking statements, commitments and challenges (X_i, A_i, c_i). Whenever a random value is required, $\hat{P}$ uses its random function on the entire "history" of random oracle queries so far. This has the effect that if an extractor runs two copies of $\hat{P}$ and "forks" them at any point by giving them different answers to a random oracle query, they will behave from then on as if they had independent sources of randomness.

In the second loop, $\hat{P}$ completes the proofs by computing the responses s_i and asks extraction queries in reverse order (note that the loop counter j runs from n down to 1). Whenever the extractor returns a witness w_i, $\hat{P}$ checks this and halts if it is incorrect. While the main prover does not need to check witnesses (the adaptive proof game does this already), if the extractor tries to give a bad witness to a rewinding copy of the prover then this copy halts and refuses to divulge any further responses s_i which the extractor could use to apply special soundness.

The algorithm $\mathcal{A}_n$ shows the idea behind our reduction $\mathcal{R}$. The Schnorr proof statements and commitments become OMDL challenges and we use one dlog query to compute the response. We can already see that if the extractor correctly answers any extract query of the main copy of the adversary without launching any other copies of the adversary then we must win the OMDL game since we used two challenges but only one discrete logarithm query for the proof in question and the extractor's reply is the second discrete logarithm. We can open all other proofs with one extra dlog query in which case we have obtained $2n$ challenges and solved them all with only $2n-1$ dlog queries, which means that we win the OMDL game. The reduction $\mathcal{R}$ in the next step keeps track of OMDL queries across all copies of the adversary so that we can make a similar argument to win OMDL whenever there are fewer than 2^n copies of the adversary in play.

Reduction.

The key idea in our reduction $\mathcal{R}$ is to track the history H of all challenges (random oracle responses) in each copy of the adversary. We give the code of the reduction in Figure 12. The calls to ro and extract pass control to the extractor

which may choose to activate a different copy of the adversary before returning a value[10].

The reduction shares state between the copies of the adversary that it simulates through the OMDL challenger and two global maps L and Φ, which are shared between all copies of the adversary simulated by $\mathcal{R}$. In the first loop, the adversary makes a chain of n Schnorr statement/commitment pairs. The map L tracks the pair that the adversary makes when its history is H and we write $L[H]$ for the value stored at key H, if any. This allows the reduction to simulate several copies of the adversary consistently: whenever a copy with history H is supposed to make a statement/commitment pair, it looks first in the list H if another copy has already made the required pair and re-uses the same pair if this is the case.

If a copy of the adversary needs to make a fresh statement/commitment pair because it is the first copy to reach history H, it calls the newchallenge subroutine. This draws two OMDL challenges, records them in L and makes an entry in Φ which we will describe in a moment.

In each pass through the second loop, each adversary copy completes a proof, asks and verifies an extraction query. Since the aim of the OMDL game is to open all challenges with fewer dlog calls than challenge calls, consider a potential Φ defined as the number of challenge calls minus the number of discrete logarithm calls made so far at any point in the execution of the reduction $\mathcal{R}$. We will see that this potential can never become negative. If we ever manage to collect all discrete logarithms while the potential is strictly positive, we can win the OMDL game.

The potential Φ can be expressed as the sum of the local potentials ϕ over all pairs (X, A) of statements and commitments made by the reduction. The map Φ tracks this potential and some extra bookkeeping information. Whenever the newchallenge procedure draws two new OMDL challenges, it adds a new entry for them in Φ. Each such entry is a 5-tuple (ϕ, c, s, x, a) where the first element ϕ is the local potential of the pair and the last four elements can take the special value ? (undefined). When a pair is first created, two OMDL challenges have been used to create it and no discrete logarithms for this pair are known yet so the local potential is set to $\phi = 2$ and the other entries are undefined.

The first time we make a proof on a pair (X, A), we use one dlog query to get the required response s and drop the local potential to $\phi = 1$. We also record the challenge c and response c used in the map Φ.

If we need to make a proof on a pair (X, A) at local potential $\phi = 1$, there must have been a previous proof on this pair (or the potential would still be at 2). We have two cases: if we have been given the same challenge as in the previous proof, we just replay the same response. This is why we record challenge/response pairs in the map Φ. If we are given a fresh challenge, this means that the extractor has "forked" two copies of the adversary on this proof and is about to obtain the

[10] For those with knowledge of software engineering terminology: the copies of the adversary simulated by $\mathcal{R}$ are coroutines with shared state and the ro and extract calls are "yield" calls.

witness by special soundness. In this case, we drop the potential to 0 with a dlog query to get x, the actual witness on which we are making our Schnorr proof, and use the previously stored information to find a.

If the extractor forks multiple copies of the adversary on the same proof more than once (i.e. it gives three different copies three different challenges for the same statement/commitment pair) then after the second proof, the local potential will be at 0 but we will have recovered x and a already. Therefore, we do not need any further dlog queries to complete the third proof but can just compute the response the usual way.

Lemma 1. *For any extractor $\mathcal{K}$ connected to the adaptive proof game, our reduction $\mathcal{R}$ is statistically indistinguishable from multiple copies of the adversary $\hat{P}$.*

Proof. The proof is by induction over all queries sent to the extractor $\mathcal{K}$. Before the first time the extractor receives anything, it obviously cannot distinguish anything. The reduction receives a random oracle query from the adversary whenever a rewinding copy of the adversary passes through its first loop. The elements in these queries are completely characterised by the following description:

- If the history of the current copy (making the random oracle query) is a prefix of any other copy's history, then the current copy returns the exact same query as the previous one. After all, all copies run the same algorithm on the same initial random string.
- If the current copy's history is "fresh" — either the current copy has advanced further than any other copy or the extractor has forked it — then the two elements in the random oracle query are uniformly random group elements independent of any previous elements. This is because the elements are drawn using a random function on a fresh input.

By inspection of the code of the reduction, we see that it has the same loop structure and meets the same invariants. The list L ensures the first condition that copies with the same history return the same elements and fresh copies trigger challenge calls which return fresh, uniform group elements by definition of the OMDL problem. Therefore, the pattern of random oracle queries is statistically indistinguishable between the reduction and the adversary copies. The argument for the main copy is identical although the extractor only gets to see the oracle queries in response to list queries to the adaptive game rather than immediately.

For extraction queries the induction argument is even simpler: statement/-commitment pairs of each copy are exactly the ones asked earlier in the random oracle queries (in reverse order) and the responses s_i are completely determined by the statement, commitment and challenge (which came from the extractor).

The event NZP.

Let event NZPin an execution of the reduction $\mathcal{R}$ be the event that $\mathcal{R}$ receives a correct response to an extract query while the local potential of the associated statement/commitment pair is $\phi = 1$ (it can never be 2 as the reduction had to drop it to 1 just to get the response s to ask the extract query in the first place).

Lemma 2. *In any execution, the moment event* NZP *occurs, the reduction* $\mathcal{R}$ *can immediately win the OMDL game that it is playing.*

The only way that the extractor can ever answer an extraction query without triggering event NZPis if it has interacted with a different copy of the adversary, giving it a different challenge for the same pair. This triggers the secondproof algorithm which drops the local potential to 0. This is exactly extraction by forking and special soundness. In other words, event NZPis the event that the extractor finds a witness by some other means than special soundness.

Proof. When the reduction receives the correct w for a pair (X, A) at potential 1, the entry $\Phi[(X, A)]$ is of the form $(1, c, s, ?, ?)$ such that (X, A, c, s) is a correct Schnorr transcript. Since w is the discrete logarithm of X, the reduction can compute $a \leftarrow s - c \cdot x$ and has both discrete logarithms of a pair of challenges at potential 1. The reduction next opens all other challenge pairs: pairs at potential 0 are already opened; for pairs (X', A') at potential 1 the reduction asks $x' \leftarrow \mathsf{dlog}(X')$ and proceeds as before to get a'; for pairs (X'', A'') at potential 2 it just asks $\mathsf{dlog}(X'')$ and $\mathsf{dlog}(A'')$. The result is that the reduction has made exactly one fewer dlog query than challenge query and has all discrete logs, which wins the OMDL game.

Note that the reduction can check the correctness of a candidate w from the extractor itself. If the extractor passes the reduction's "main adversary" an incorrect witness, the reduction loses the adaptive proof game. If a rewinding adversary simulated by the reduction gets an incorrect witness, event NZPis not triggered but this copy halts and so is of no further use to the extractor.

There is one other case that lets the reduction win OMDL immediately: if two challenges returned from the OMDL challenger ever collide. We could simply ignore this event as it happens with negligible probability but even then, since one discrete logarithm call in this case gives the answer to two challenges, the reduction can open all further challenges at a cost of one discrete logarithm call each and win the OMDL game.

Combinatorial argument.

Lemma 3. *Suppose that the extractor* $\mathcal{K}$ *wins an execution of the adaptive proof game against the reduction* $\mathcal{R}$ *without* $\mathcal{R}$ *winning its OMDL game (whether by event* NZP *or a collision in the challenger). Then the extractor must have interacted with at least* 2^n *copies of the adversary (simulated by* $\mathcal{R}$*).*

Proof. Consider an arbitrary execution of $\mathcal{K}$ with $\mathcal{R}$ in which $\mathcal{K}$ wins the adaptive proof game. We identify each instance of the adversary that $\mathcal{R}$ simulates through the sequence of answers it received to its random oracle calls. For example, the

main adversary is identified through $\mathcal{I}_0(c_0, c_1, \ldots, c_n)$: since $\mathcal{K}$ was successful it returned answers to all of the extraction queries (and in particular to all of the random oracle queries). Different random oracle answers imply that $\mathcal{K}$ talks to different copies of the adversary. At the same time, we ignore duplicates: if $\mathcal{K}$ had induced identical executions in two different copies then we count them as one.

Suppose that event NZP does not occur. Then we can construct the following complete binary tree: the nodes of the tree are of the form $(\mathcal{I}, k)$ where $\mathcal{I}$ is an identifier (for a copy of the adversary) and $1 \leq k \leq n$. The nodes of the tree satisfy the invariant that if $(\mathcal{I}, k)$ is present in the tree, then copy $\mathcal{I}$ must have run up to the point where it made its k-th extraction query and received a valid answer. We set the root of the tree to be $(\mathcal{I}_0, n)$: the main adversary completed so $\mathcal{K}$ must have answered all of the n extraction query of $\mathcal{I}_0$.

For each node $(\mathcal{I}, k)$ that occurs in the tree and for which $k > 1$ we recursively add two children: the first child of $(\mathcal{I}, k)$ is $(\mathcal{I}, k-1)$. If an invocation $\mathcal{I}$ asked k extraction queries then it certainly also asked $k-1$ queries and got answers to all previous ones, so the first child meets the required invariant. For the second child of $(\mathcal{I}, k)$, consider the pair of OMDL challenges (X, A) that $\mathcal{I}$ made its k-th extraction query about. Since event NZP has not occurred, there must have been some invocation $\mathcal{I}'$ in which a second extraction query was made on the same (X, A) (as otherwise the potential of (X, A) would be one).

With overwhelming probability the other execution $\mathcal{I}'$ must have been about the same $n-k$ first hash queries (and answers) as $\mathcal{I}$, or else the value (X, A) would not occur in the random oracle call made by $\mathcal{I}'$. Furthermore, $\mathcal{I}'$ can only reduce the potential for (X, A) if it has run up to the k-th extraction query, which implies that it received valid answers for the first $k-1$ queries. The invariant is satisfied, so we can add $(\mathcal{I}', k-1)$ as the second child of $(\mathcal{I}, k)$ to our binary tree.

To summarize, if a node $N' = (\mathcal{I}', k')$ is a child of a node $N = (\mathcal{I}, k)$ then $k' = k-1$ and $\mathcal{I}'$ is an execution that received identical answers to the first $n-k$ hash queries as $\mathcal{I}$. Furthermore, the two children of $N = (\mathcal{I}, k)$ correspond to adversaries that differ in the responses to their $n-k+1$-st hash query. Therefore, for any node $N = (\mathcal{I}, k)$ at level $k > 1$, all identities appearing in the subtree rooted at the first child of N shared the first $n-k+1$ hash answers with $\mathcal{I}$; all identities appearing in the subtree rooted at the second child of N shared the first $n-k$ hash answers with $\mathcal{I}$ but got a different answer to the $n-k+1$-st hash query.

The leaves of the binary tree will thus be of the form $(\mathcal{I}_i, 1)$, where each different $\mathcal{I}_i$ corresponds to a different adversary that $\mathcal{R}$ simulates. The tree has 2^n leaves and it only remains to show that the identities in the leaves are pairwise distinct. Suppose that two of these identities $\mathcal{I}$ and $\mathcal{I}'$ in different leaves are identical. There is a unique path in the binary tree that connects these two leaves and a unique node $N = (\mathcal{I}'', k'')$ on this path with a highest index k'' among all nodes on this path. Without loss of generality, the path from N to $(\mathcal{I}, 1)$ passes through the first child of N, and that to $(\mathcal{I}', 1)$ through the second

child of N. But this is a contradiction: all identities in the subtree rooted at the first child of N differ from those in the second at least in the response to the $n - k'' + 1$-st hash query, therefore they are distinct. We conclude that an extractor $\mathcal{K}$ which does not trigger event NZP must access 2^n distinct copies of the adversary. In other words, for any efficient $\mathcal{K}$ the probability of winning the adaptive proof game against our adversary without triggering event NZP is negligible.

Proof.

Suppose that Fiat-Shamir-Schnorr is an adaptive proof with respect to a group generator $\mathsf{GrpSetup}$ and the discrete logarithm relation. Then there is an efficient extractor $\mathcal{K}$ that can win the adaptive proof game with overwhelming probability against $\hat{P}$ for $n = \lambda$ in the groups $\mathbb{G}(\lambda)$ created by $\mathsf{GrpSetup}$. Our reduction $\mathcal{R}$ and (many copies of) $\hat{P}$ are statistically indistinguishable in any group so with overwhelming probability, $\mathcal{K}$ still wins the adaptive proof game against $\mathcal{R}$ (w.r.t. $\mathsf{GrpSetup}$).

Since $\mathcal{K}$ is efficient, from some λ_0 onwards $\mathcal{K}$ makes strictly fewer than 2^λ queries; in particular it launches fewer than 2^λ copies of the adversary. Since $\mathcal{K}$ triggers event NZPin $\mathcal{R}$ whenever it wins the adaptive proof game with fewer than 2^λ copies of the adversary, we conclude that $\mathcal{R}$ (interacting with $\mathcal{K}$ and the adaptive proof game) wins the OMDL game with overwhelming probability w.r.t $\mathsf{GrpSetup}$.

Actually, since the probability that $\mathcal{R}$ solves OMDL is negligibly close to the probability of $\mathcal{K}$ winning the adaptive proof game (for $\lambda > \lambda_0$), our Theorem even holds for extractors with only non-negligible success probability and yields an OMDL adversary with a comparable success probability. This concludes the proof of Theorem 3.

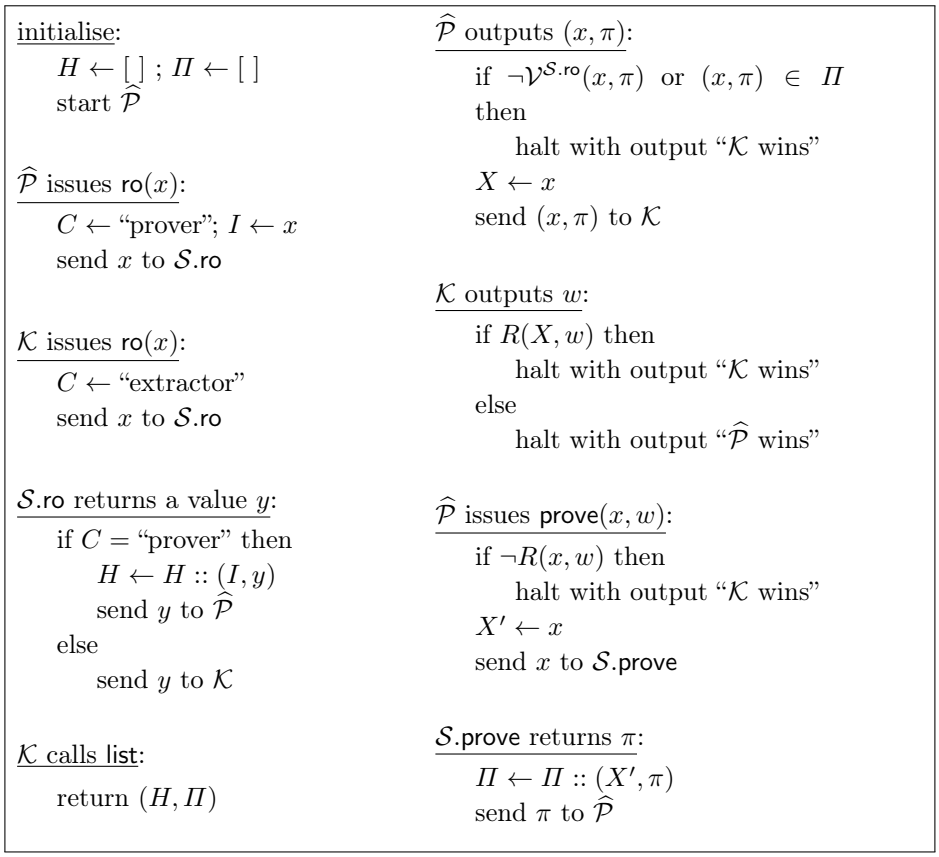


Fig. 4: The game G defining SSE in the random oracle model. It has three interfaces for the main prover $\widehat{\mathcal{P}}$, the extractor $\mathcal{K}$ and the simulator $\mathcal{S}$. The list H tracks the prover's random oracle queries. The state C (for caller) tracks who made the last random oracle query so that the result can be returned to the initiator. I (input) tracks the prover's oracle inputs to update H correctly. X and X' track the prover's outputs and proof requests for verification in a later query.

$\mathcal{K}$ calls $\widehat{\mathcal{P}}$ for the first time:	$\widehat{\mathcal{P}}$ calls $\mathsf{ro}(x)$:
start $\widehat{\mathcal{P}}$	send ("ro", x) to $\mathcal{K}$
$\mathcal{K}$ sends a value z:	$\widehat{\mathcal{P}}$ outputs (x, π):
send z to $\widehat{\mathcal{P}}$	send ("extract", x, π) to $\mathcal{K}$
	$\widehat{\mathcal{P}}$ calls $\mathsf{prove}(x, w)$:
	send ("prove", x) to $\mathcal{K}$

Fig. 5: The auxiliary game $\widehat{G}$ for SSE. It acts mostly as a channel between $\mathcal{K}$ and a rewound prover $\widehat{P}$ except that it strips witnesses from proof queries. We use the convention that $\widehat{G}$ indicates to $\mathcal{K}$ whether a value is for a random oracle, extraction or proof query by prefixing a string.

Fig. 6: The adaptive proof game and the queries that the various algorithms can exchange.

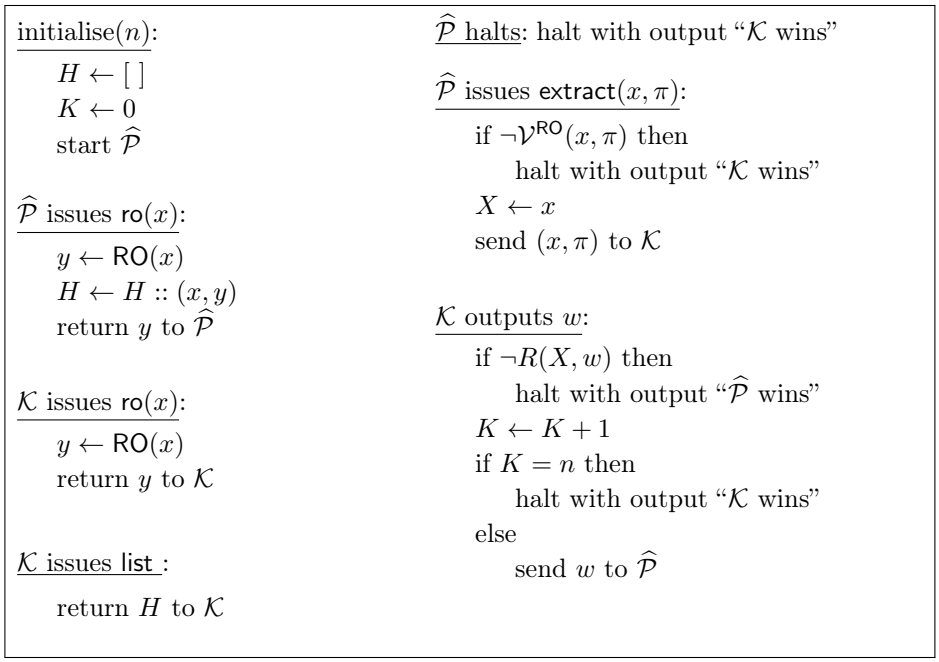


Fig. 7: The game G for adaptive proofs with parameter n.

initialise(n):

$H \leftarrow [\,]$; $\Pi \leftarrow [\,]$
$K \leftarrow 0$
start $\widehat{\mathcal{P}}$

$\widehat{\mathcal{P}}$ issues $\mathsf{ro}(x)$:

$C \leftarrow$ "prover"; $I \leftarrow x$
send x to $\mathcal{S}.\mathsf{ro}$

$\mathcal{K}$ issues $\mathsf{ro}(x)$:

$C \leftarrow$ "extractor"
send x to $\mathcal{S}.\mathsf{ro}$

$\mathcal{S}.\mathsf{ro}$ returns a value y:

if $C =$ "prover" then
  $H \leftarrow H :: (I, y)$
  send y to $\widehat{\mathcal{P}}$
else
  send y to $\mathcal{K}$

$\mathcal{K}$ issues list:

return (H, Π)

$\widehat{\mathcal{P}}$ halts:

halt with output "$\mathcal{K}$ wins"

$\widehat{\mathcal{P}}$ issues $\mathsf{extract}(x, \pi)$:

if $\neg \mathcal{V}^{\mathcal{S}.\mathsf{ro}}(x, \pi)$ or $(x, \pi) \in \Pi$ then
  halt with output "$\mathcal{K}$ wins"
$X \leftarrow x$
send (x, π) to $\mathcal{K}$

$\mathcal{K}$ outputs w:

if $\neg R(X, w)$ then
  halt with output "$\widehat{\mathcal{P}}$ wins"
$K \leftarrow K + 1$
if $K = n$ then
  halt with output "$\mathcal{K}$ wins"
else
  send w to $\widehat{\mathcal{P}}$

$\widehat{\mathcal{P}}$ issues $\mathsf{prove}(x, w)$:

if $\neg R(x, w)$ then
  halt with output "$\mathcal{K}$ wins"
$X' \leftarrow x$
send x to $\mathcal{S}.\mathsf{prove}$

$\mathcal{S}.\mathsf{prove}$ returns π:

$\Pi \leftarrow \Pi :: (X', \pi)$
send π to $\widehat{\mathcal{P}}$

Fig. 8: Simulation sound n-proofs in the random oracle model.

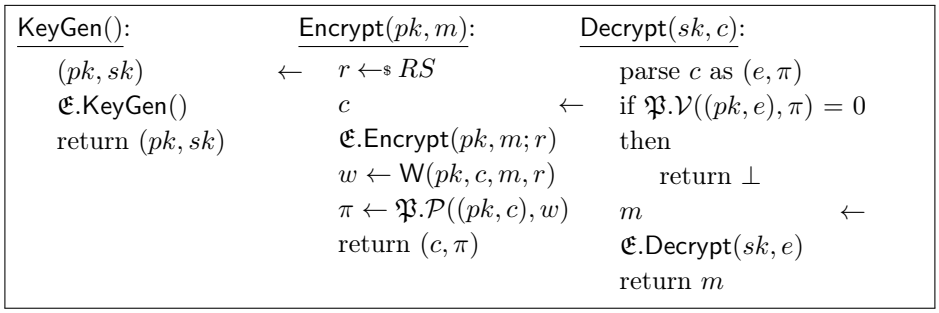


Fig. 9: The encrypt-then-prove transformation of compatible $\mathfrak{E}$ and $\mathfrak{P}$. RS is the space of random strings used by the original encryption algorithm.

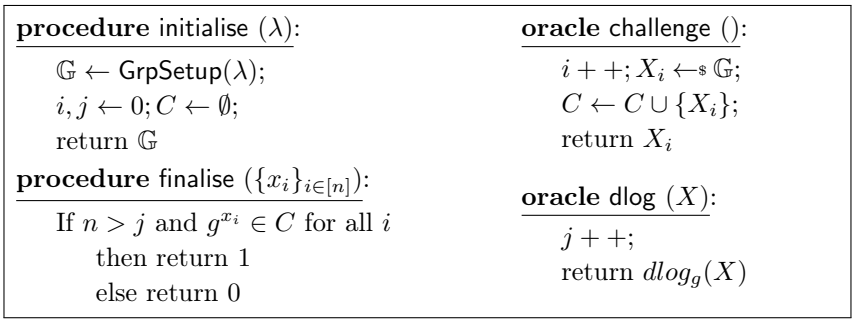


Fig. 10: Experiment for defining the one-more discrete logarithm problem. We demand that $x_i \neq x_j \mod |\mathbb{G}|$ for all $i \neq j$.

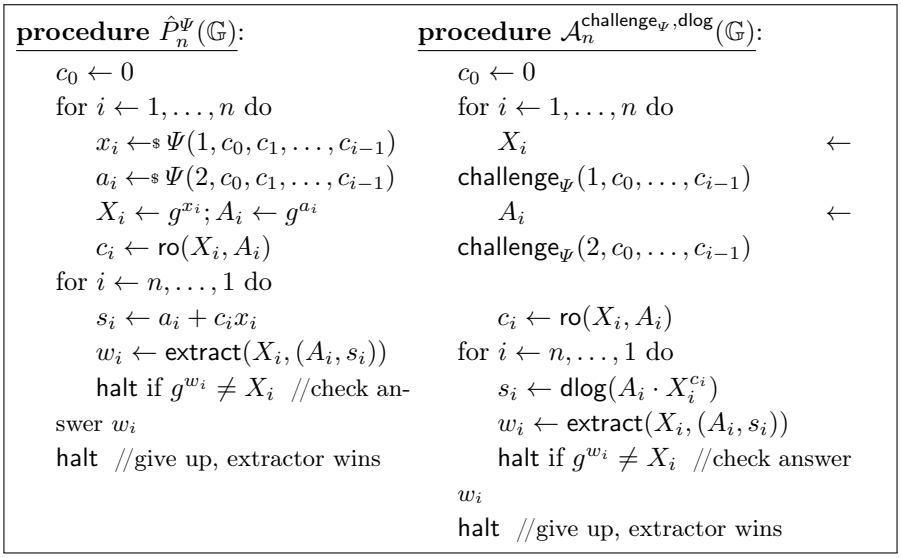


Fig. 11: Adversary $\hat{P}$ is against the adaptive property of the Fiat-Shamir-Schnorr. Adversary $\mathcal{A}_n$ simulates $\hat{P}$ using the oracles from the one more discrete logarithm experiment.

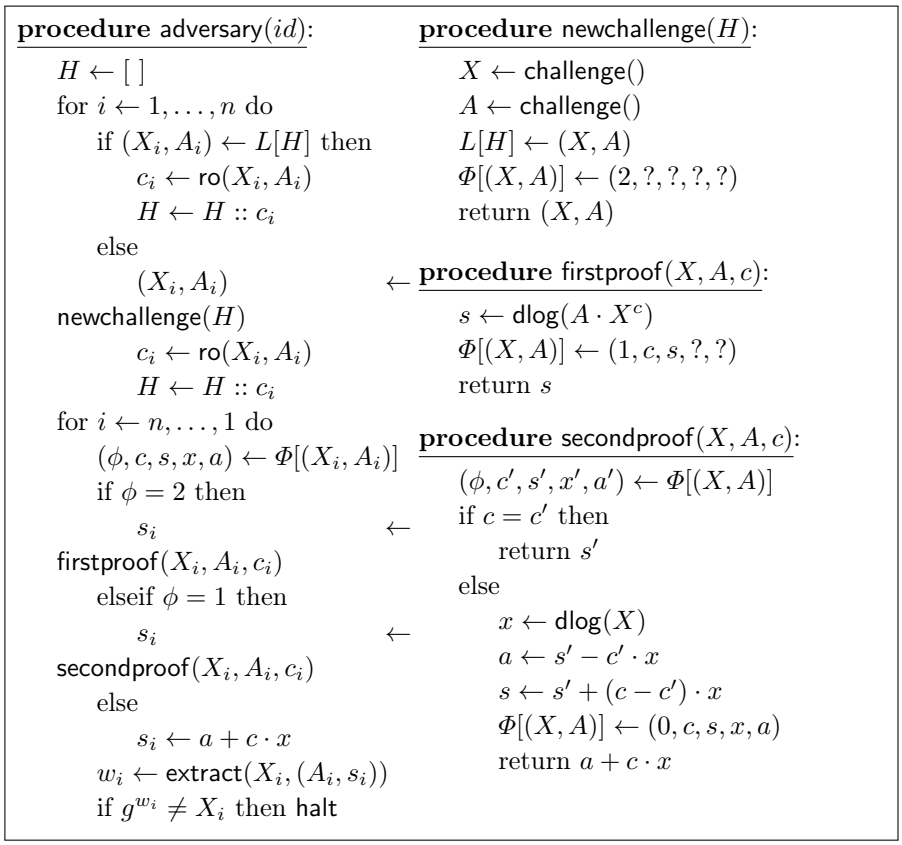


Fig. 12: The reduction $\mathcal{R}$ simulating a copy of $\hat{P}$. The reduction $\mathcal{R}$ itself consists of multiple copies of this algorithm each with a unique identifier id. The variables C, L, Φ are shared between all copies.