Adaptive Proofs have Straightline Extractors (in the Random Oracle Model)

Abstract. The concept of *adaptive security* for proofs of knowledge was recently studied by Bernhard et al. They formalised adaptive security in the ROM and showed that the non-interactive version of the Schnorr protocol obtained using the Fiat-Shamir transformation is not adaptively secure unless the one-more discrete logarithm problem is *easy*. Their only construction for adaptively secure protocols used the Fischlin transformation [3] which yields protocols with *straight-line extractors*. In this paper we provide two further key insights. Our main result shows that any adaptively secure protocol must have a straight-line extractor: even the most clever rewinding strategies cannot offer any benefits against adaptive provers.

Then, we show that any Fiat-Shamir transformed Σ-protocol is not adaptively secure unless a related problem which we call the Σ-one-wayness problem is *easy*. This assumption concerns not just Schnorr but applies to a whole class of Σ-protocols including e.g. Chaum-Pedersen and representation proofs. We also prove that Σ-one-wayness is hard in the generic group model. Taken together, these results suggest that Fiat-Shamir transformed Σ-protocols should not be used in settings where adaptive security is important.

1 Introduction

Non-interactive zero-knowledge (NIZK) proof schemes are proofs of knowledge (PoK) if they admit an extractor such that, for every prover creating a statement/proof pair that verifies, the extractor can return a witness to the statement. In this paper we study the relation between variants of the proof of knowledge property with different kinds of extractors. An extractor is *straight-line* if it only sees a single execution of the prover (and learns the RO queries/answers that the prover makes), or *rewinding* if it is allowed to launch and interact with further copies of the prover (with the same coins used to produce the statement) before returning a witness.

The distinction between straightline and rewinding extractors may be crucial in applications since for a rewinding extractor it is not clear how many times does it have to rewind to extract all witnesses from a prover who makes a sequence of n proofs. Shoup and Gennaro [1] first encountered this problem in the context of proving CCA security of a particular public-key encryption scheme; the "obvious" approach ends up rewinding 2^n times which leads to an inefficient reduction. Clearly, this problem disappears for a straight-line extractor.

Bernhard et al. [12] proposed a notion of *adaptive* proofs that lie somewhere between proofs with inefficient rewinding strategies and straight-line PoKs: a proof scheme is adaptively secure if there is an extractor that can rewind, but must efficiently extract even from provers who make sequences of proofs. The notion is called adaptive because the extractor must return a witness for the first proof to the prover before the prover makes the second one, and so on.

The main theorem of the cited work shows that the usual Fiat-Shamir-Schnorr proof scheme is *not* an adaptive proof unless the one-more discrete logarithm assumption is easy.

This result essentially separates the usual PoK notion from adaptive proofs, but the separation relies on an inefficient interactive assumption, it is specific to a proof system for a specific problem (discrete logarithm), and a specific class of proofs (those obtained from Sigma protocols via the Fiat-Shamir transform). It also leaves open the question whether any adaptive proofs exist that are not also straight-line[1].

Our contribution. In this paper we obtain a full characterization of adaptive proofs in the random oracle model and we leverage this result to provide more general results regarding the limitations of the Fiat-Shamir transform.

ADAPTIVE PROOFS $\equiv$ STRAIGHTLINE EXTRACTORS. Our main contribution answers Bernhard et al.'s open question negatively. It holds for all non-interactive proof schemes in the ROM, whether or not they are derived from Sigma protocols:

Theorem 1 (Informal). *Consider an arbitrary non-interactive proof of knowledge system in the ROM. If the proof system has an efficient adaptive extractor (against adaptive provers) then it also has a straight-line extractor.*

The immediate consequence of this theorem is that when designing PoKs for an adaptive setting, one cannot rely on rewinding and instead one should ensure the existence of a straightline extractor. While a general strategy is to employ Fischlin's transformation [3], one may still want to rely on the more efficient construction that uses the Fiat-Shamir transformation whenever possible – the impossibility result of Bernhard et al. [12] only applies to Fiat-Shamir-Schnorr proofs.

LIMITATIONS OF THE FIAT-SHAMIR TRANSFORM. We show that the Fiat-Shamir transformation has intrinsic limitaitons. In particular, we generalize the results of [12] in two distinct directions. On the one hand, we show that it holds for arbitrary Sigma protocols for proving knowledge of preimages of linear functions (including Schnorr, Chaum–Pedersen and representation proofs). More interestingly, we weaken the condition under which these proofs are not adaptively secure from one-more discrete logarithm (resp. one-more one-wayness [2]) to the following assumption: a dishonest verifier in a single execution of the Sigma protocol cannot extract the witness. We call this assumption *Σ-one-wayness.* Our result thus improves from a "q-type" assumption, which does not admit an efficient game, to an efficient game with only three rounds.

[1] Straight-line proofs are trivially adaptively secure.

This theorem answers the main open question of [12] and hints at a shortcoming of proofs based on the Fiat–Shamir transform: if used in a setting where the prover gets to adaptively chose statements an extractor would have to be (more or less) straight-line. However, if this is the case, the proof may not be that interesting anyway as the underlying witness is not well-protected:

Theorem 2 (Informal). *Suppose there is a straight-line extractor for a Fiat-Shamir transformed Sigma protocol Σ (to prove knowledge of a preimage of some linear function f). Then a dishonest verifer can extract a witness (a preimage under f) in a single run of Σ.*

Taken together, these results imply that Fiat-Shamir transformed Sigma protocols are not adaptively secure in any setting in which they might be useful. Take the Schnorr protocol as an example: if Fiat-Shamir-Schnorr is adaptively secure then either discrete logarithms are easy in the relevant group, in which case the Schnorr protocol is redundant, or the Schnorr protocol provably helps a dishonest verifier to extract the discrete log of the statement — in which case Schnorr is certainly not zero-knowledge.

The theorem of Bernhard et al. [12] contains an adaptive prover who makes a sequence of n proofs such that each one depends on all previous ones. The straightforward rewinding strategy — rewind on every proof to extract — ends up rewinding 2^n times since the rewound provers make new proofs which again have to be rewound. A combinatorial argument then shows that any strategy that rewinds fewer than 2^n times must have taken a discrete logarithm to find out the witness for one of the proofs output by the prover. The problem is that we do not know where, and also if we inject a challenge in one proof then we end up having to simulate all other proofs in the experiment. So far the solution to this problem was to reduce to the one-more discrete logarithm problem.

Our proof technique for Theorem 2 (formally, Theorem 14) is to take any prover and turn it into an adaptive prover who makes a chain of n proofs, together with some bookkeeping. We then show that any adaptive extractor against this prover must either take exponential time or reduce to a straight-line extractor against the original prover. Applying this theorem to the honest prover, we get a reduction to Σ-one-wayness. We summarize these results in the following table (FSS=Fiat-Shamir-Schnorr, DLOG=discrete logarithm, OMDL=one-more discrete logarithm).

property	breaks if FSS has
one-way (e.g. DLOG)	straight line extractor [10]
Σ-one-way	adaptive extractor (new)
one-more one-way (e.g. OMDL)	adaptive extractor [12]

In conclusion, we suggest that adaptive proofs are not a new class of proofs but rather another description of the class of straight-line extractable proofs; and Fiat-Shamir transformed Sigma protocols for "useful" functions are *not* in this class.

Weaker assumptions. The obvious next question is whether one could strengthen our results even further and only rely on one-wayness of ϕ (e.g. in the case of Schnorr, DLOG) rather than Σ-one-wayness. We show using a meta-metareduction that no algebraic metareduction [4] from a programming extractor to one-wayness (e.g. DLOG) can exist, unless one-wayness is already easy. All previous metareductions in this area [10, 12] including ours to Σ-one-wayness are algebraic: the only operations they perform on elements of the target group are group operations.

A generalization of the Generic Group Model (GGM). To strengthen trust in the Σ-one-wayness hypothesis on which our impossibility rely we provide a justification in the generic group model. Interestingly, the first problem that one needs to face here is that the existing approaches to formalizing and using GGM is not suitable: in brief, an adversary that interacts with the Σ protocol for some problem gets to see not only group elements but also some information related to the exponents of these group elements – this ability is not considered the standard GGM formalizations. We suggest one approach to deal with this issue and use the resulting model to formally justify Σ-one-wayness.

Related work. One-more type assumptions were introduced by Bellare et al. [2]. Their value in proving schemes secure is subject to some debate, as explained by Koblitz and Menezes [8] who also gave the first "weakened" one-more assumption. Problems with forking-based proofs were first noted by Shoup and Gennaro [1]; Paillier and Vergnaud [4] developed separation results for Schnorr-based signatures using metareductions that formed the first formal proof of a limitation of Schnorr-based techniques. Both Brown [6] and Bresson et al. [7] concurrently applied separation techniques to one-more problems. Fischlin and Fleischhacker [9] were the first to consider limitations of metareductions via meta-metareductions. The most recent results that motivated this paper are Seurin and Treger [10] who gave a very simple metareduction from a non-programming extractor for Schnorr proofs to discrete log; and Bernhard et al. [12] who introduced adaptive proofs.

2 Preliminaries

Notation. $f : A \to B$ is a function with domain A and range B; $\mathcal{A} : A \twoheadrightarrow B$ is a randomised algorithm on the same domain and range. We write security games in a language based on Bellare and Rogaway's code-based game-playing [5]. $y \leftarrow f(x)$ is assignment, $x \twoheadleftarrow R$ is uniform random sampling. $T[i]$ is the element at index i of table T.

An interactive, randomised algorithm $\mathcal{A}$ has access to a random string r and an input/output interface. It maintains its state between calls. A security game is such an algorithm that may at some point output "win" or "lose", which terminates the entire execution. We say that a security property is given by a game, to mean that the property holds if no efficient adversary can cause the game to output "win" with more than negligible probability in some underlying security parameter.

Σ-PROTOCOLS — Let k be a field. Let $\mathcal{W}, \mathcal{X}$ be k-vector spaces and let $\phi : \mathcal{W} \to \mathcal{X}$ be a k-linear map. Suppose further that one can sample uniformly from k and $\mathcal{W}$.

Definition 3. *The Σ-protocol Σ_ϕ is the following interactive protocol for a prover P to prove knowledge of a preimage under ϕ to a verifier V.*

$$\begin{array}{rcl}
P(w \in \mathcal{W}, x = \phi(w)) & & V \\
r \twoheadleftarrow \mathcal{W}; a \leftarrow \phi(r) & \xrightarrow{(x,a)} & \\
 & \xleftarrow{c} & c \twoheadleftarrow k \\
s \leftarrow r + c \cdot w & \xrightarrow{s} & \phi(s) \overset{?}{=} a + c \cdot x
\end{array}$$

We choose to let P transmit the statement x to V as part of the first round of the proof — of course, V may also know x in advance. The verifier accepts if the equation $\phi(s) = a + c \cdot x$ holds in $\mathcal{X}$, in which case we call (x, a, c, s) an accepting transcript. Instances of this template protocol include:

- Schnorr: $\mathcal{W} = k = GF(p)$, $\mathcal{X}$ is some group G of order p with a generator g (e.g. over an elliptic curve) and $\phi(w) = g^w$.
- Chaum-Pedersen: $\mathcal{W} = k$, $\mathcal{X} = G \times G$ for a group as above and $\phi(w) = (g^w, h^w)$ for two different generators g, h of G.
- Representation: $\mathcal{W} = k^n, \mathcal{X} = G$ and $\phi(w_1, \ldots, w_n) = \prod_{i=1}^n {g_i}^{w_i}$ for some known set of generators $\{g_i\}_{i \in I}$ of G.

Σ-protocols according to our definition automatically satisfy:

- Special soundness: if (x, a, c, s) and (x, a, c', s') are accepting transcripts with $c \neq c'$ then $1/(c - c') \cdot (s - s')$ is a preimage[2] of x under ϕ.
- Soundness: if $x' \in \mathcal{X} \setminus \mathrm{Im}[\phi]$, then a cheating prover gets a verifier to accept with probability at most $1/|k|$.
- Honest-verifier zero-knowledge: a verifier who choses c as prescribed (at least, independently of a) gains no information from the protocol beyond the fact that the prover knows a preimage of x under ϕ.

PROOF SCHEMES — A (non-interactive) proof scheme for a relation ρ on sets $X \times W$ consists of a proof space Π and a pair of algorithms $\texttt{prove} : X \times W \twoheadrightarrow \Pi$ and $\texttt{verify} : X \times \Pi \to \{0, 1\}$ (i.e. `verify` is deterministic). An element $\pi \in \Pi$ satisfying $\texttt{verify}(x, \pi) = 1$ is called a valid proof for x. For any (x, w) satisfying ρ, if $\pi \leftarrow \texttt{prove}(x, w)$ then we require $\texttt{verify}(x, \pi) = 1$. We further assume that there is an algorithm $\texttt{sample} :\twoheadrightarrow X \times W$ that produces elements uniformly distributed in ρ (as a subset of $X \times W$). In the random oracle model (ROM), both `prove` and `verify` may call a function H that is modelled as a random oracle in security proofs. The relation ρ itself does not use H.

[2] The inversion $1/(c - c')$ is in the field k where it exists due to $c \neq c'$; the dot in this formula is field-vector multiplication.

Fiat-Shamir — The Fiat-Shamir transformation turns Σ-protocols into non-interactive proof schemes that are full zero-knowledge proofs of knowledge in the random oracle model. The idea is simply to replace the verifier's challenge c by a hash over the statement x and the commitment a.

Definition 4. *Let $\phi : \mathcal{W} \to \mathcal{X}$ be a k-linear function where $\mathcal{W}$ and k are efficiently sampleable. Suppose that H is a function with domain (including) $\mathcal{X} \times \mathcal{X}$ and range k. Then the Fiat-Shamir transformed Σ-protocol $\mathcal{F}_\phi$ is the following proof scheme for sets $(\mathcal{X}, \mathcal{W})$ and relation $\rho(x, w) = 1 \iff \phi(w) = x$. The proof space is $\Pi = \mathcal{X} \times \mathcal{W}$ and the algorithms are*

- `sample()`*: pick $w \twoheadleftarrow \mathcal{W}$, set $x \leftarrow \phi(w)$ and return (x, w).*
- `prove(`x, w`)`*: pick $r \twoheadleftarrow \mathcal{W}$, set $a \leftarrow \phi(r)$, $c \leftarrow H(x, a)$ and $s \leftarrow r + cw$. The proof is $\pi = (a, s)$.*
- `verify(`$x, (a, s)$`)`*: check that $\phi(s) = a + H(x, a) \cdot x$.*

3 Variations on the theme of one-wayness

Σ-protocols are only useful when the function ϕ is hard to invert: otherwise, they are trivially zero-knowledge proofs of knowledge, but so is the protocol in which the prover just sends the statement x to the verifier. For the same reasons, if ϕ is easy to invert then $\mathcal{F}_\phi$ is adaptively secure too. This shows that we cannot hope for a theorem of the form "Fiat-Shamir Schnorr is not adaptively secure", since it is adaptively secure e.g. in the group $(\mathbb{Z}_p, +)$ where taking discrete logarithms is easy. Consequently, limitation theorems take the form "if Schnorr is adaptively secure then some property (e.g. OMDL) is easy to break".

We discuss some possible security properties of the function ϕ and introduce Σ-one-wayness. Later on we will show that Fiat-Shamir proofs cannot be adaptively secure unless Σ-one-wayness of ϕ is easy to break (in which case, their use within protocols would be already questionable).

Σ-one-wayness. One-wayness is the first obvious candidate property. Recall that a property defined by a game means that the property (in this case one-wayness of ϕ) holds if it is hard to make the game output "win". The game gives the adversary a uniformly chosen image x; the adversary wins by recovering any preimage w' s.t. $\phi(w') = x$.

Definition 5. *The one-wayness property for a function $\phi : \mathcal{W} \to \mathcal{X}$ is given by the following game.*

```
1  $w \twoheadleftarrow \mathcal{W}; x \leftarrow \phi(w)$
2  output $x$; input $w' \in \mathcal{W}$
3  if $\phi(w') = x$ then return "win" else return "lose" end
```

We propose a new security notion that we call Σ-one-wayness (for linear functions). This says that even a dishonest verifier (who choses c arbitrarily, maybe depending on x and a) cannot extract w from a single run of the protocol. We do not claim that Σ-one-wayness is a sufficient security notion for Σ protocols (it says nothing about extracting partial information on w) but we postulate that it is only deployed if this condition is satisfied. Consequently, we are not proposing a new scheme that is secure under the Σ-one-wayness assumption, but we will claim that if the Fiat-Shamir proof scheme for a function ϕ is adaptively secure then the Σ-one-wayness property for ϕ is easy to break too. Σ-one-wayness is clearly stronger than one-wayness of the function ϕ, but it will turn out to be weaker than one-more one-wayness (which we define in a moment).

Definition 6. *The Σ-one-wayness property for a linear function $\phi : \mathcal{W} \to \mathcal{X}$ is defined by the following game.*

```
$w \twoheadleftarrow \mathcal{W}; x \leftarrow \phi(w)$
$r \twoheadleftarrow \mathcal{W}; a \leftarrow \phi(r)$
output $(x, a)$; input $c \in k$
$s \leftarrow r + c \cdot w$
output $s$; input $w' \in \mathcal{W}$
if $\phi(w') = x$ then return "win" else return "lose" end
```

If Σ-one-wayness of a function ϕ is easy to break but one-wayness is hard to break, then running the protocol Σ_ϕ could leak a preimage to a dishonest verifier, that said verifier could otherwise not compute by herself. In this case we would discourage the use of the protocol Σ_ϕ (among other things it is certainly not zero-knowledge). If both Σ-one-wayness and one-wayness of ϕ are easy to break then Σ_ϕ is both harmless and useless. So, we think that Σ-one-wayness of ϕ is a necessary condition for the protocol Σ_ϕ to be deployed. Under this condition, we will show that the Fiat-Shamir transformed Σ_ϕ is not adaptively secure.

WEAK ONE-MORE ONE-WAYNESS. For Schnorr, the one-wayness property of the function $\phi(x) = g^x$ is the discrete logarithm property. Bernhard et al. [12] use a stronger assumption known as one-more discrete logarithm, which generalises to one-more one-wayness[2, 7].

One-more one-wayness assumes an invertible function ϕ. The adversary is given two oracles which she can call in any order: a sampling oracle that picks a random preimage $w_i \in \mathcal{W}$ and reveals $x_i = \phi(w_i)$ and an opening oracle that on input x outputs $\phi^{-1}(x)$. To win the game, the adversary must recover the preimages of all samples x_i with fewer calls to the opening oracle than to the sampling oracle.

One-more one-wayness was first discussed by Bellare et al. [2] although the name first appears in a later paper [7]. Unlike the other properties here, it does not admit an efficient security game: the game itself needs to be able to invert ϕ on arbitrary inputs. Koblitz and Menezes [8] discussed a variant that only allows the adversary to open challenges themselves; this is isufficient for our applications. Instead we propose a new property: weak one-more one-wayness fixes this problem by restricting the adversary to asking linear combinations of the sampled challenges. Your task is still to recover all w_i with fewer queries to the linear combination oracle than to the sampling oracle. The requirement for ϕ to be invertible can also be dropped again.

Definition 7. *The weak and normal one-more one-wayness properties for a bijection $\phi : \mathcal{W} \to \mathcal{X}$ are given by the following games. In each game the adversary can call the* `sample` *and* `open` *oracles many times, in any order. The adversary wins the game if it can provide preimages under ϕ for all samples that it had obtained and yet it made fewer opening queries than sample queries.*

weak one-more one-wayness:

```
sample():
    $n \leftarrow n + 1$
    $w[n] \twoheadleftarrow \mathcal{W}$
    return $\phi(w[n])$

open($c_1, \dots, c_n \in k^n$):
    return $\sum_{i=1}^{n} c_i \cdot w_i$
```

one-more one-wayness:

```
sample():
    (same as weak version)



open($x \in \mathcal{X}$):
    return $\phi^{-1}(x)$
```

The strong problem clearly reduces to the weak one. A weak adversary can still obtain a preimage of a particular sample by submitting the vector with 1 at the appropriate position and 0 elsewhere.

The point of the weak one-more one-wayness property is that the theorem by Bernhard et al. [12] can trivially be strengthened to show that Fiat-Shamir-Schnorr is not adaptively secure even under the weak one-more discrete logarithm property: their reduction only ever makes opening queries on elements that are linear combinations of samples with known coefficients. This is not surprising since their reduction is trying to "simulate" Schnorr proofs on sample elements.

Σ-one-wayness reduces to weak one-more one-wayness, even to a weaker version where the number of samples is additionally bounded at 2 and only a single linear combination query is allowed. Thus, we end up with a hierarchy of one-wayness / Σ-one-wayness / weak one-more one-wayness / one-more one-wayness, in order of increasing strength.

4 Adaptive proofs

In this section we recall the notion of adaptive proofs and introduce templates for a couple of provers that form the basis of the results we prove in the next section.

Definition 8. *A prover is an algorithm $\mathcal{P}$ that outputs a statement/proof pair (x, π); in the ROM a prover may make random oracle calls. We assume that there is a uniformly sampleable space of random strings R associated to each* *prover and we write $\mathcal{P}(r)$ to mean running prover $\mathcal{P}$ on random string $r \in R$. In the ROM, we write $(x, \pi, l) \leftarrow \mathcal{P}(r)$ to mean that we also return the list l of random oracle queries made by this execution of the prover on random string r.*

A proof scheme is sound with error ε if for any prover $\mathcal{P}$, the probability of producing a pair (x, π) such that $\texttt{verify}(x, \pi) = 1$ but no w exists making $\rho(x, w)$ hold, is at most ε.

A proof scheme in the random oracle model is straight-line extractable with error ε if there is an extractor $\mathcal{K}$ as follows. For any prover $\mathcal{P}$, pick $r \twoheadleftarrow R$ and execute $(x, \pi, l) \leftarrow \mathcal{P}(r)$. If $\texttt{verify}(x, \pi) = 1$ w.r.t.[3] l then with probability at least $1 - \varepsilon$, $\mathcal{K}(x, \pi, l)$ returns a w such that $\rho(x, w)$ holds. It follows immediately that an extractable proof scheme with error ε is also sound with at most the same error.

A proof scheme in the ROM has a programming straight-line extractor with error ε if there is an extractor $\mathcal{K}$ as follows. Let any prover $\mathcal{P}$ interact with $\mathcal{K}$ in the sense that $\mathcal{K}$ answers $\mathcal{P}$'s random oracle queries. If $\mathcal{P}$ outputs (x, π) such that $\texttt{verify}(x, \pi) = 1$ w.r.t. the oracle queries made by $\mathcal{P}$, then with probability at least $1 - \varepsilon$, $\mathcal{K}$ outputs a w such that $\rho(x, w)$ holds.

A straight-line extractor, whether programming or not, may have black-box access to further copies of the prover in the following sense: it may start these copies and control all interaction with them, including picking the random string and answering all random oracle queries. As motivation for this (established) notion of straightline extractors, consider an extractor who is trying to extract from a honest prover. The code of the honest prover is known, so the extractor can always simulate the honest prover on inputs of its choice. The extractor cannot see the random string of the "main" copy of the honest prover from which it is trying to extract, however.

A rewinding extractor can, in addition to the capabilities of a straight-line extractor, launch further copies of the prover $\mathcal{P}$ with the *same random string* as the one that the extractor is trying to extract from, and answer their random oracle queries. The difference between straight-line and rewinding extractors is thus that rewinding extractors can run further copies of the prover that behave identically to the "main" one, as long as they receive the same inputs and outputs, and thus "fork" the prover instance from which they are trying to extract.

ADAPTIVE PROOFS. We present the adaptive proof game of Bernhard et al. [12]. The game is an interactive algorithm with two interfaces for an adaptive prover and an extractor. An adaptive prover for a proof scheme $(\Pi, \texttt{prove}, \texttt{verify})$ w.r.t. (X, W, ρ) is an algorithm that can repeatedly output pairs $(x, \pi) \in X \times \Pi$ and expect witnesses $w \in W$ in return. After a number of such interactions, the adaptive prover halts. In the random oracle model, an adaptive prover may also ask random oracle queries.

[3] We say $\texttt{verify}(x, \pi) = 1$ w.r.t. l if all elements on which $\texttt{verify}$ queries the oracle on input (x, π) are contained in l, and $\texttt{verify}$ outputs 1 on these inputs if given the appropriate responses in l.

```
initialise:
    $Q \leftarrow [\ ]$
    $K \leftarrow 0$
    $r \twoheadleftarrow R$
    run $\mathcal{P}(r)$

$\mathcal{P}$ asks ro($x$):
    $y \leftarrow$ ro($x$)
    $Q \leftarrow Q :: (x, y)$
    send $y$ to $\mathcal{P}$

$\mathcal{P}$ outputs $(x, \pi)$:
    if not verify($x, \pi$) then
        $\mathcal{K}$ wins; halt.
    end
    $\Xi \leftarrow x$
    send $(x, \pi, Q)$ to $\mathcal{K}$

$\mathcal{K}$ asks ro($x$):
    $y \leftarrow$ ro($x$)
    send $y$ to $\mathcal{K}$

$\mathcal{K}$ outputs $w$:
    if $\rho(\Xi, w)$ then
        $K \leftarrow K + 1$
        if $K = n$ then
            $\mathcal{K}$ wins; halt.
        end
        send $w$ to $\mathcal{P}$
    else
        $\mathcal{P}$ wins; halt.
    end
$\mathcal{P}$ halts:
    $\mathcal{K}$ wins; halt.
```

Fig. 1. The adaptive proof game. The extractor also has access to further copies of $\mathcal{P}(r)$ using coins r sampled from an appropriate size domain R.

An adaptive extractor is an algorithm $\mathcal{K}$ that can interact with an adaptive prover: it repeatedly takes pairs $(x, \pi) \in X \times \Pi$ such that $\texttt{verify}(x, \pi) = 1$ as input and returns witnesses $w \in W$ such that $\rho(x, w) = 1$. In addition, an adaptive extractor may be rewinding, i.e. launch further copies of the adaptive prover that run on the same random string as the main one (managed by the game) and answer all their queries. The adaptive proof game does not check the correctness of witnesses for the other copies of the prover.

Definition 9. *A proof scheme* $(\Pi, \texttt{prove}, \texttt{verify})$ *is an n-proof (with error ε) in the random oracle model if there is an adaptive extractor $\mathcal{K}$ such that for any adaptive prover $\mathcal{P}$, the extractor wins the game in Figure 1 (with probability at least $1 - \varepsilon$). The adaptive extractor $\mathcal{K}$ may launch and interact with further copies of the adaptive prover $\mathcal{P}$ on the same random string r as the main one, without the game mediating between them.*

In the n-proof game in Figure 1, the prover $\mathcal{P}$ is trying to find a claim (x, π) that verifies, but from which the extractor $\mathcal{K}$ cannot extract a witness w. The extractor is trying to extract witnesses from all claims made by the prover.

The game uses three global variables. K stores the number of witnesses that the extractor has found so far. If this counter reaches n, the extractor wins and the scheme is an n-proof. Q stores a list of all the prover's random oracle queries so far. These are provided[4] to the extractor along with each of the prover's claims. Ξ stores the last statement that appeared in one of the prover's claims; it is used to check the validity of a witness returned by the extractor.

A n-proof for $n = 1$ is simply a proof of knowledge in the ROM: the prover makes a single claim (a pair containing a statement x and a proof π) and the extractor wins if it can obtain a witness. A proof scheme is an adaptive proof if there is an adaptive extractor that works for any polynomially bounded parameter n.

Canonical provers. The canonical prover $\mathcal{P}_C$ samples a statement/witness pair and creates a proof. We write R_C for the randomness space of the canonical prover and $\mathcal{P}_C(r)$ to denote running the canonical prover on the random string $r \in R_C$.

Definition 10 (canonical prover). *Let $(\Pi, \texttt{prove}, \texttt{verify})$ be a proof scheme for (X, W, ρ) where r is uniformly sampleable via an algorithm* `sample`*. The canonical prover $\mathcal{P}_C$ for this scheme is the following algorithm.*

$$(x, w) \leftarrow \texttt{sample()};\ \pi \leftarrow \texttt{prove}(x, w);\ \texttt{return}\ (x, \pi)$$

If required, the canonical prover can also return the list l of all random oracle queries made during its execution (by `prove`*).*

Since an extractor is supposed to work against any (efficient) prover, to argue that an extractor cannot exist it is enough to show that one cannot extract from the canonical prover. To deal with adaptive extractors, we propose the following construction of an adaptive chain prover $\mathcal{P}^n$ from any prover $\mathcal{P}$. It follows the idea of Shoup and Gennaro [1] in making a chain of "challenges" (in this case proofs) where each challenge depends on all previous ones and then asking queries on them *in reverse order*. This way, the obvious rewinding extractor using special soundness will take exponential time.

To make each challenge depend on previous ones, we use a function F to update the random string for each of the n contained copies of $\mathcal{P}$ based on the (random oracle) state of the previous copy. The final parameter l returned by $\mathcal{P}$ is the list of all random oracle queries made by this copy. Recall that $\mathcal{P}(r)$ means run prover $\mathcal{P}$ on random string $r \in R$ where R is the randomness space for this prover.

Definition 11 (adaptive chain prover). *Let $(\Pi, \texttt{prove}, \texttt{verify})$ be a proof scheme for (X, W, ρ). Let $\mathcal{P}$ be any prover (in the ROM) and let R be its randomness space. Let L be the space of possible random oracle input/output transcripts. Let $F : R \times L \to R$ be a function (which does not depend on the random oracle). The adaptive chain prover $\mathcal{P}^n$ of order n w.r.t. function F is the algorithm in Figure 2, taking an $r \in R$ as input.*

[4] The original definition gave the extractor an extra `list` oracle to query the prover's random oracle list. Our presentation is equivalent.

```
$\mathcal{P}^n(r)$:
for $i$ = 1, $n$ do
    $(x_i, \pi_i, l) \leftarrow \mathcal{P}(r)$
    $r \leftarrow F(r, l)$
end
for $i$ = $n$, 1, -1 do
    output $(x_i, \pi_i)$
    input $w'$
    if not $\rho(x_i, w')$ then
        halt
    end
end
```

Fig. 2. The adaptive chain prover $\mathcal{P}^n$.

Later on, we will take F to be a (pseudo-)random function. This has the effect that two copies of $\mathcal{P}^n$ that get identical answers to their random oracle queries will behave identically, but two copies of $\mathcal{P}^n$ that "fork" will behave as copies of $\mathcal{P}$ with *independent* random strings from the forking point onwards.

The intuition behind this construction is that having access to copies of $\mathcal{P}$ on some uniformly random string r' cannot help you extract from a copy $\mathcal{P}(r)$, as long as r and r' are independent — certainly, an extractor could always simulate such copies herself if the code of $\mathcal{P}$ is known. We will use this idea to show that forking a copy of $\mathcal{P}^n$ is no help in extracting from the proofs made later on by another copy.

5 Limitations of the Fiat-Shamir transformation

We recall the hierarchy of security definitions for functions (the last is the strongest): one-way / Σ-one-way / weak one-more one-way / one-more one-way.
Known limitations. Seurin and Treger [10] proved that Fiat-Shamir-Schnorr cannot have a non-programming straight-line extractor unless the underlying function is not one-way (i.e. one can take discrete logarithms). The following theorem generalizes this result to the case of arbitrary Σ-protocols.

Theorem 12. *Suppose there is a non-programming straight-line extractor K for the proof scheme $\mathcal{F}_\phi$. Then ϕ is not one-way. Specifically, there is an algorithm breaking one-wayness with approximately the same running time and success probability as the extractor K has against the canonical prover $\mathcal{P}_C$.*

Let x be a challenge from the one-way game for ϕ; we need to find a w' such that $\phi(w') = x$. We simulate a proof: pick $s \twoheadleftarrow \mathcal{W}$, $c \twoheadleftarrow k$ and $a \leftarrow \phi(s) - c \cdot x$. Then we give the extractor the statement x, proof (a, s) and a list of random oracle queries consisting of the entry $RO(x, a) = c$. These elements are identically distributed to what the extractor would see in an execution with the canonical prover $\mathcal{P}_C$ for Σ_ϕ. We pass any witness w' returned by the extractor on to the challenger to win with the same success probability. q.e.d.

Bernhard et al. [12] showed that substituting an adaptive extractor for a straight-line one gets a similar result for the one-more one-wayness assumption on the function ϕ — the proof of this theorem is nontrivial however. While their original proof only concerned Fiat-Shamir-Schnorr, a close inspection of the proof shows that it works for any Σ protocol and that the weak one-more assumption is sufficient too. In summary.

Theorem 13. *Suppose that there is an efficient adaptive extractor for $\mathcal{F}_\phi$. Then ϕ is not weak one-more one-way.*

We will not re-prove the theorem here. As to running time, as long as the extractor makes fewer than 2^n queries when running against a particular prover then the reduction to weak one-more one-wayness runs in the same time as the extractor, but its success probability is the inverse of the number of copies of the prover that the extractor causes to be invoked. Although Bernhard et al. [12] only prove the theorem for the case of the Fiat-Shamir-Schnorr protocol, their reduction is "black box" in the sense that it only needs to be able to sample and open instances of the underlying Sigma protocol. The exact same proof will work for our generalisation. We can write the prover in the cited theorem as $(\mathcal{P}_C)^n$, the adaptive chain prover derived from the canonical prover. This will allow us to conclude that any extractor against the chain prover implies a straight-line extractor against the canonical prover $\mathcal{P}_C$.

New Results. If we switch to a programming straight-line extractor, we can show a separation result under the Σ-one-wayness assumption.

Theorem 14. *Suppose there is a programming straight-line extractor K for $\mathcal{F}_\phi$. Then ϕ is not Σ-one-way. Specifically, there exists a reduction with approximately the same running time and the same success probability as the extractor K against the canonical prover $\mathcal{P}_C$.*

We simulate the canonical prover towards the extractor. Receive x, a from the Σ-one-way challenger and ask the random oracle query $c \leftarrow RO(x, a)$ (which the extractor answers). Then send c to the Σ-one-way challenger to get s and send (x, a, s) to the extractor. Again, whenever the extractor provides the correct w', we win against the challenger. q.e.d.

This result is new, but not surprising — Fiat-Shamir transformed Sigma protocols are not supposed to be straight-line extractable and the Σ-one-wayness property is constructed exactly to make this reduction work. The value of said property is that we can also use it for adaptive extractors.

Our main contribution in this paper is a new theorem that says all adaptive proofs in the ROM admit a straight-line extractor.

Theorem 15. *Consider any non-interactive ROM proof scheme with an adaptive extractor $\mathcal{K}$. Suppose that, running against any n-prover $\widehat{\mathcal{P}}$, the extractor $\mathcal{K}$ causes at most $f(n) < 2^n$ copies of the prover to be run in the experiment and answers all extraction queries of the main run correctly with probability at least $p(n) > 0$. Then there is a programming straight-line extractor against any non-adaptive prover $\mathcal{P}$ with success probability $p(n)/(n \cdot f(n))$.*

Applying this theorem to Fiat-Shamir transformed Sigma protocols gets us the following insight.

Corollary 16. *Suppose that the Fiat-Shamir transformed Sigma protocol $\mathcal{F}_\phi$ is adaptively secure. Then ϕ is not Σ-one-way secure.*

For the corollary, note that we have a programming straight-line extractor against the canonical prover $\mathcal{P}_C$ by applying Theorem 15 to the prover $(\mathcal{P}_C)^n$. The result then follows from Theorem 14.

We sketch the proof here and provide the full argument in the appendices. Let $\mathcal{P}$ be a non-adaptive prover. We construct a simulator $\mathcal{S}^n$ that is indistinguishable from the black box providing "rewinding" access for multiple copies of $\mathcal{P}^n$. The point of the simulator is that it shares state "between the copies". We then guess which instance of the prover (specifically, which proof) the extractor is going to answer without "forking" and inject the Σ-one-wayness challenge into it. The same combinatorial argument as in the proof of Bernhard et al. [12] shows that such an instance must exist if the extractor launches fewer than 2^n copies of the prover.

The core of such a simulation argument is to keep track of a history of each instance of the prover, since two copies of the prover with identical histories must behave identically towards the extractor. In $(\mathcal{P}_C)^n$, this history is implicitly tracked in the randomness r used for each copy of $\mathcal{P}_C$, however a collision in the random oracle could lead to two copies with different histories "merging". Our simulator computes an explicit history instead, namely the list of all random oracle queries so far.

As in Bernhard et al. [12] we define an event E that occurs whenever a copy of the prover gets its extraction query answered without having a "partner" (another copy, from which the witness was extracted by forking and special soundness). The novelty in our proof is that because we have cast the prover as a chain $(\mathcal{P}_C)^n$ with suitable state tracking, we can show that event E implies not only a "break" of the chain prover but also of one of the contained canonical provers $\mathcal{P}_C$. We then show that, if the simulator guessed correctly, event E implies that the simulator can solve its Σ-one-wayness challenge. This is a much weaker assumption than one-more one-wayness.

6 Generic hardness of Σ-one-wayness

In this section we show that Fiat-Shamir transformed Sigma $\mathcal{F}_\phi$ is not adaptively secure in the generic setting. Thus if we want to build a protocol where we need an adaptively secure proof (such as to get CCA encryption), we would not use $\mathcal{F}_\phi$. By Corollary 16 we just need to prove that ϕ is $\Sigma-$one-way secure in the generic group model (GGM). Again, let $\mathcal{X}$, $\mathcal{W}$ be vector spaces over k and $\phi : \mathcal{W} \to \mathcal{X}$ be a k-linear map. We assume that k is a finite field and $\mathcal{X}, \mathcal{W}$ are finite dimensional, since sampling uniformly from an infinite set does not make much sense. Let $b_1, b_2, ..., b_n$ be basis vectors of $\mathcal{X}$, where $\dim(\mathcal{X}) = n$ and

$\{b_1, ..., b_s\}$ be a basis for $\mathrm{Im}(\phi)$. For each $1 \leq i \leq s$ denote a_i to be an element of $\mathcal{W}$ so that $\phi(a_i) = b_i$.

The generic group model is a model which analyses success of algorithms against representations of groups which do not reveal any information to adversary. There are many ways to formalise this idea [13–15]. We will follow the definition provided by Shoup [15]. Here, adversary is given access to images of elements of a group under a random injective map $\sigma : \mathcal{X} \to S \subset \{0,1\}^*$, called an *encoding function*. Group operations can be computed by making oracle queries. The adversary is given access to two oracles ADD and INV:

$$ADD(\sigma(x), \sigma(y)) = \sigma(x+y),\ INV(\sigma(x)) = \sigma(-x).$$

Note that the adversary cannot get any information from the representation $\sigma(x)$ of element x. A *generic algorithm* $\mathcal{A}$ for $\mathcal{X}$ on S is a probabilistic algorithm that takes as input an encoding list $(\sigma(x_1), \sigma(x_2), ..., \sigma(x_l))$ where each $x_i \in \mathcal{X}$ and σ is an encoding function of $\mathcal{X}$ on S. As the algorithm executes, it makes queries to ADD or INV oracles and then appends outputs of the queries to the encoding list. The output of $\mathcal{A}$ is a bit string denoted as $\mathcal{A}(\sigma; x_1, ..., x_l)$. We also want to extend the inteface of the model and introduce the $FSS_{r,w}$ oracle: $FSS_{r,w}(c) = r + cw$ for some $r, w \in \mathcal{W}$ and $c \in k$. We may assume that when oracles encounter some $\sigma_1 \notin Im(\sigma)$, they return an error message.

Similarly to most of the proofs of generic hardness of computational problems, we use the Schwartz-Zippel lemma to estimate the probability of a generic algorithm winning the $\Sigma-$one-wayness game. However, we are interested in the version of this theorem when we sample uniformly from a vector space rather than a field.

Lemma 17. *Let V be a finite dimensional vector space over k and F be a non-zero linear polynomial in $k[X_1, ..., X_t]$. Then, for $x_1, x_2, ..., x_t$ selected at random independently and uniformly from V the probability that $F(x_1, x_2, ..., x_t) = 0$ is at most $1/|V|$.*

Proof. Note that V is isomorphic to k^m, where $m = \dim(V)$. Hence we can assume without loss of generality that $V = k^m$. Let $x_1, ..., x_t$ be uniformly random and independent elements of V and denote $x_{i,j}$ to be the $j-$th component of x_i. Therefore, since F is linear we have that:

$$\begin{aligned}\Pr(F(x_1, x_2, ..., x_n) = 0) &= \Pr(F(x_{1,1}, ..., x_{t,1}) = 0, ..., F(x_{1,m}, ..., x_{t,m}) = 0) \\ &= \prod_{i=1}^{m} \Pr(F(x_{1,i}, ..., x_{t,i}) = 0) \\ &\leq (1/|k|)^m = 1/|V|\end{aligned} \tag{1}$$

by the Schwartz-Zippel lemma.

The next theorem establishes generic hardness of $\Sigma-$one-wayness.

Theorem 18. *Let w, r be random elements of $\mathcal{W}$ and $\mathcal{A}$ be a generic algorithm for $\mathcal{X}$ on $S \subset \{0,1\}^*$ that makes at most m queries to ADD and INV oracles and exactly one query to $FSS_{r,w}$ oracle. Then the probability that $\phi(\mathcal{A}(\sigma; b_1, ..., b_n, x, y)) = x$ is $O((m+n)^2/|\mathcal{X}| + |ker(\phi)|/|\mathcal{W}|)$, where $x = \phi(w)$ and $y = \phi(r)$.*

Full proof is provided in the appendix. Here we skip the details and only show brief sketch of it. We use techniques suggested by Shoup [4] and introduce a Polynomial Simulation Model (PSM). We define indeterminants X and Y and keep lists of polynomials $F_i's$ in $k[X, Y]$. Each polynomial F_i has corresponding values $z_i \in \mathcal{W} \cup \{\perp\}$ and $\sigma_i \in S$. We can think of them as preimages of $F_i(x, y)$ w.r.t. ϕ and $\sigma(F_i(x, y))$ respectively. At the beginning we set $F_i = b_i$ for $i = 1, 2, ..., n$, $F_{n+1} = X, F_{n+2} = Y$, $z_i = a_i$ for $i = 1, 2, ..., s$ and $\perp$ for $i = s + 1, ..., n+2$, and $\sigma_1, \sigma_2, ..., \sigma_{n+2}$ are chosen at random, but $\sigma_i \neq \sigma_j$ for $1 \leq i < j \leq n$. When query to ADD or INV oracle occurs, we perform the same operation on our polynomials corresponding to the input of the query. If extraction query is made, we return a random value from $\mathcal{W}$. We have to remember to update $z_i's$ after every query.

Note that if for every $F_i \neq F_j$ we have $F_i(x, y) \neq F_j(x, y)$ (i.e. no *collision* occurs) then adversary is not able to spot the difference between the GGM and PSM. We use Lemma 17 to find upper bound on the probability of such an event. Then we easily argue that adversary cannot win the game in the model we just defined by showing that $z_{n+1} = \perp$.

Theorem 18 implies that every generic algorithm $\mathcal{A}$, which wins Σ−one-wayness with high probability, must perform at least $\Omega(\alpha\sqrt{|\mathcal{X}|})$ group operations, where $\alpha = \sqrt{1 - |\ker(\phi)|/|\mathcal{W}|}$. In particular, if $\mathcal{W}$ is large then we get the lower bound $\Omega(\sqrt{|\mathcal{X}|})$ for IES by choosing $\phi(w) = g^w$.

7 Reducing to DLOG?

Given a non-programming straight line extractor in the ROM for a Fiat-Shamir transformed Sigma protocol $\mathcal{F}_\phi$ we can break one-wayness of ϕ; for a programming extractor or an adaptive extractor we can break Σ-one-wayness. This raises the question, can we break one-wayness given a programming extractor? Our answer is negative. We give the argument for the case of Schnorr proofs where one-wayness is the discrete logarithm (DLOG) problem; this also implies that there can be no generic metareduction to one-wayness for any Sigma protocol.

The metareductions in the theorems of Seurin and Treger [10], Bernhard et al. [12] and this paper are all algebraic (in the sense of Paillier and Vergnaud) [4] over the vector space[5] $\mathcal{X}$, the range of the function ϕ. We therefore consider it a meaningful result to show that no algebraic metareduction to DLOG can exist (unless DLOG is already easy).

[5] Paillier and Vergnaud defined the algerbraic model for groups; one can interpret a $GF(p)$ vector space as an Abelian group to use their definition of the algebraic model.

Theorem 19. *If there is an algebraic metareduction from a programming straight-line extractor for Fiat-Shamir-Schnorr proofs to the DLOG problem then there is also a meta-metareduction breaking the DLOG problem directly with approximately the same success probability.*

The proof is in the appendix. The idea is that a metareduction M gets to see two bases in the group: the generator g and the challenge h from its DLOG challenger. Since we assumed a programming extractor, M must ask its random oracle queries to its extractor interface where our meta-metareduction will answer them. Any statement output by M (to the extractor) therefore has the form $(g^a h^b)$ for some (a, b) which are available to our meta-metareduction by use of the algebraic model. Proofs of the form $(a, 0)$ are independent of the challenge h; intuitively they should not help to compute the discrete logarithm of h so we just return the witness a. The first time M outputs a proof with a statement of the form (a, b) with $b \neq 0$, we fork M on the relevant random oracle query and use special soundness to find the discrete logarithm of h to basis g.

8 Conclusions

Bernhard et al. introduced adaptive proofs, setting up a hierarchy of (1) proofs of knowledge (2) adaptive proofs and (3) straight-line extractable proofs, with a separation between (1) and (2). While useful for proving limitations of Sigma protocols, we have showed that adaptive proofs are not a new class of proof after all: all adaptively secure proofs admit a straight-line extractor against the canonical prover.

Along the way we have generalised previous results from Schnorr's protocol to Sigma protocols. In addition, we have weakened the counter-assumption from one-more one-wayness, which is a "q-type" interactive assumption (adversary gets an unbounded number of sample queries) and is not efficiently realisable to Σ-one-wayness, which both has a constant number of steps and an efficient security game.

Our result shows that the Fiat-Shamir transformation and Sigma protocols in general may be even weaker than previously thought. Namely, the non-interactive proof scheme $\mathcal{F}_\phi$ only achieves adaptive security if a single execution of the interactive protocol Σ_ϕ against a dishonest verifier already leaks the secret witness with non-negligible probability. In such a case, the use of any Sigma-protocol based scheme for ϕ is highly questionable.

References

1. V. Shoup and R. Gennaro. Securing Threshold Cryptosystems Against Chosen-Ciphertext Attack. In: Advances in Cryptology (Eurocrypt '98), LNCS 1403, pages 1–16, 1998.
2. M. Bellare, C. Namprempre, D. Pointcheval and M. Semanko. The One-More-RSA-Inversion Problems and the Security of Chaum's Blind Signature Scheme. eprint 2001/002. Originally appeared as "The Power of RSA Inversion Oracles and the Security of Chaum's RSA-Based Blind Signature Scheme" in Financial Cryptography, LNCS 2339, pages 319–338, Springer 2001.
3. 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.
4. P. Paillier and D. Vergnaud. Discrete-Log Based Signatures may not be Equivalent to Discrete Log. In: Asiacrypt '05, LNCS 3788, pages 1–20, Springer 2005.
5. M. Bellare and P. Rogaway. Code-based game-playing proofs and the security of triple encryption. In: Advances in Cryptology — Eurocrypt '06, LNCS 4004, pages 409–426, 2006. The title cited is from the latest version on eprint at `http://eprint.iacr.org/2004/331`.
6. D. Brown. Irreducibility to the One-More Evaluation Problems: More May Be Less. eprint 2007/435.
7. E. Bresson, J. Monnerat and D. Vergnaud. Separation Results on the "One-More" Computational Problems. In: CT-RSA '08, LNCS 4964, pages 71–87, Springer 2008.
8. N. Koblitz and A. Menezes. Another look at non-standard discrete log and Diffie-Hellman problems. In: Journal of Mathematical Cryptology, vol. 2 issue 4, pages 311–326, 2008. eprint 2007/442.
9. M. Fischlin and N. Fleischhacker. Limitations of the Meta-Reduction Technique: The Case of Schnorr Signatures. In: Eurocrypt '13, LNCS 7881, pages 444–460, Springer 2013. eprint 2013/140.
10. Y. Seurin and J. Treger. A Robust and Plaintext-Aware Variant of Signed ElGamal Encryption. In: CT-RSA LNCS 7779, pages 68–83, Springer 2013.
11. David Bernhard. Zero-Knowledge Proofs in Theory and Practice. PhD thesis, University of Bristol, 2014. Available at `www.cs.bris.ac.uk/$\sim$bernhard/papers.html`
12. David Bernhard, Marc Fischlin and Bogdan Warinschi. Adaptive Proofs of Knowledge in the Random Oracle Model. PKC '15, LNCS 9020, Springer, pages 629–649, 2015.
13. Maurer, Ueli. Abstract models of computation in cryptography. IMA International Conference on Cryptography and Coding. Springer Berlin Heidelberg, 2005.
14. Nechaev, Vassiliy Ilyich. Complexity of a determinate algorithm for the discrete logarithm. Mathematical Notes 55.2 (1994): 165-172.
15. Shoup, Victor. Lower bounds for discrete logarithms and related problems. International Conference on the Theory and Applications of Cryptographic Techniques. Springer Berlin Heidelberg, 1997.

game	description	loss*
$\mathcal{B}$ / $\mathcal{P}^n$	initial game	–
$\mathcal{B}_1$	single copy of F	$\equiv$
$\mathcal{B}_2$	memoize F	$\equiv$
$\mathcal{B}_3$	switch F for random function	$\equiv$ or $\Delta_{\text{PRF}}(F)$
$\mathcal{B}_4$	lazy sampling	$\equiv$
$\mathcal{B}_5$ / $\mathcal{S}^n$	change `random` inputs	p_{coll}

* $\equiv$: games are equivalent. $\Delta_{\text{PRF}}(F)$: distinguishing advantage between F and a random function. p_{coll}: probability of a collision.

Fig. 3. Game-hops to construct the simulator $\mathcal{S}^n$.

A Proof of Theorem 15

Recall that a prover $\mathcal{P}$ is an algorithm that may take some random input $r \in R$ (for a randomness space R specific to the prover), may make random oracle queries and eventually outputs a statement/proof pair (x, π). For any prover $\mathcal{P}$, the prover $\mathcal{P}^n$ is the construction described in Definition 11. If a proof scheme is adaptively secure, there is an extractor $\mathcal{K}$ that can win the adaptive game against any n-prover. In particular it wins against $\mathcal{P}^n$ for any (non-adaptive) prover $\mathcal{P}$. We show how use this extractor to build a straight-line extractor against $\mathcal{P}$. Recall that in the extraction game the knowledge extractor needs to extract all witnesses corresponding to a run of the malicious adaptive prover and that in the process the extractor has access to a number of copies of the prover, all using the same coins as the main run.

A.1 The simulator $\mathcal{S}^n$

We will construct a simulator $\mathcal{S}^n$ that is indistinguishable from a black box providing access to multiple copies of $\mathcal{P}^n$. The proof of this fact uses a number of game-hops summarised in Figure 3. We begin with a black box $\mathcal{B}$ that provides access to multiple copies of $\mathcal{P}^n$, all running with the same random string r and containing their own copy of the function F.

Hop 1. We replace all copies of F with one single copy, shared between the instances of $\mathcal{P}^n$, to get a new black box $\mathcal{B}_1$. Since functions are stateless, deterministic, non-interactive (F may not depend on the random oracle) and their output is completely determined by their inputs on each call, this transformation does not change the behaviour of the black box. Call the new algorithm thus obtained $\mathcal{B}_1$.

Hop 2. Next, we memoize access to F: whenever a call is made on fresh inputs, we call the function F on these inputs and store the inputs and the output in a table R. Whenever a call is made on inputs already in the table, we return the stored output directly without calling F. This gives us a new box $\mathcal{B}_2$. The properties of a function mentioned in the last step imply that this transformation too does not change the behaviour of the black box.

```
oracle random($r, l$)
    if not $T[(r,l)]$ then
        $T[(r,l)] \twoheadleftarrow R$
    end
    return $T[(r,l)]$
```

Fig. 4. A random function $R \times L \to R$.

Hop 3. We switch F for a random function $R \times L \to R$, giving a new black box $\mathcal{B}_3$. Since F was a pseudorandom function, any environment distinguishing $\mathcal{B}_2$ from $\mathcal{B}_3$ reduces to a distinguisher between F and a random function with the same distinguishing advantage.

Hop 4. A random function has the property that its outputs on any two distinct inputs are independent and uniformly random in its codomain. We can replace all remaining calls to the random function by its memoizer drawing fresh, uniform randomness from R itself, so-called lazy sampling. This does not change the behaviour of the black box. Call this memoizer algorithm `random`. We give its current operation in Figure 4 and call the new black box $\mathcal{B}_4$.

Hop 5. The only use that `random` makes of its parameters r, l is to answer repeated calls with the same parameters consistently. A call `random`(r, l) occurs when a new random string r' is required for a copy of $\mathcal{P}$ to be run as part of a copy of $\mathcal{P}^n$; when this call occurs, l is the list of all random oracle calls made by the last copy of $\mathcal{P}$ that is part of the copy of $\mathcal{P}^n$ making the call and r is a function of the initial randomness r_0 and the lists l of all previous copies of $\mathcal{P}$ that were part of the copy of $\mathcal{P}^n$ making the call.

We replace these parameters (r, l), in both `random` and its caller $\mathcal{P}^n$, by the list of all random oracle queries that all copies of $\mathcal{P}$ simulated by this copy of $\mathcal{P}^n$ have made so far. For $\mathcal{P}^n$, this means that it must keep a list T of all oracle queries made so far and after every copy of $\mathcal{P}$ finishes, append the list l of this copy to its list T. Call the black box with this change made to all its copies of $\mathcal{P}^n$, algorithm $\mathcal{B}_5$. The code with which $\mathcal{B}_5$ simulates a copy of $\mathcal{P}^n$ is in Figure 5. Algorithms $\mathcal{B}_4$ and $\mathcal{B}_5$ are indistinguishable if R is large enough, as the following results show.

Claim. For an adversary who causes up to m copies of $\mathcal{P}$ to be launched, the probability of distinguishing $\mathcal{B}_4$ and $\mathcal{B}_5$ is at most $(m + m^2)/|R|$.

Corollary 20. *If n is a security parameter, m is polynomial in n and $|R| \approx 2^n$ then the distinguishing advantage between $\mathcal{B}_4$ and $\mathcal{B}_5$ is negligible in n.*

The proof of Claim A.1 is an induction over all calls to `random` that occur in an execution trace. Before the first call, the table T will be empty so the first value returned from `random` will be uniformly distributed in R in both $\mathcal{B}_4$ and $\mathcal{B}_5$.

```
algorithm random($t$) -- called only as a subroutine from prover.
    if not $T[t]$ then -- if undefined at this point
        $T[t] \twoheadleftarrow R$
    end
    return $T[t]$

algorithm prover($r$)
    for $i$ = 1, $n$ do
        $(x_i, \pi_i, l) \leftarrow \mathcal{P}(r)$
        $Q \leftarrow Q :: l$
        $r \leftarrow$ random($Q$)
    end
    for $i$ = $n$, 1, -1 do
        $w' \leftarrow$ extract($x_i, \pi_i$)
        if not $\rho(x_i, w')$ then
            halt
        end
    end
```

Fig. 5. The lazy simulator $\mathcal{S}_n$. Each prover $\mathcal{P}^n$ is simulated by a copy of `prover(`r`)`; algorithm `random` is shared between all provers.

For any following call, there are two cases. The first is that the call is fresh, i.e. `random` ends up drawing a new random value. In both $\mathcal{B}_4$ and $\mathcal{B}_5$, these values are uniformly distributed in R and independent of previous values.

The second case is that a call is made on a value already recorded in T. In $\mathcal{B}_5$, this can only happen if the current and previous caller (who caused the relevant T-entry to be made) represent copies of $\mathcal{P}^n$ with the exact same random oracle history, since the index into T is the entire random oracle history of the current copy. In $\mathcal{B}_4$, two copies of $\mathcal{P}^n$ with the same history will also trigger the same `random` call, but it is also possible that a r-value collides with a previously chosen one. If this happens, the next copy of $\mathcal{P}$ in the current copy of $\mathcal{P}^n$ will run on the same randomness r as some previous copy of $\mathcal{P}$ in another[6] copy of $\mathcal{P}^n$, even though the random oracle histories of the two $\mathcal{P}^n$ copies are not the same. This is the only case in which the output distributions of $\mathcal{B}_4$ and $\mathcal{B}_5$ can differ.

We compute the probability of a collision in the output of `random`. The probability of hitting r_0 is $1/|R|$. The probability of a collision between two calls when there are m calls in total is bounded by $m^2/|R|$ (the "birthday bound"). Together, this means that an adversary triggering at most m copies of $\mathcal{P}$ (each of which leads to one `random` call) triggers a collision with probability at most $(m + m^2)/|R|$. q.e.d.

[6] This may also be the same copy of $\mathcal{P}^n$ at an earlier point in time, when it had a different (shorter) random oracle history.

Before we continue, we define some terms that we will use in the following claims and proofs.

Simulator $\mathcal{S}^n$. The collection of $\mathcal{B}_5$ and all copies of the adaptive prover $\mathcal{P}$ that it launches is called the simulator $\mathcal{S}^n$.

Collision. In an execution of $\mathcal{S}^n$ with initial randomness r_0, a collision is the event that either a random sample drawn in `random` hits the value r_0 or two such samples hit the same value.

Partner. A copy of $\mathcal{P}$ simulated by $\mathcal{S}^n$ which has completed its execution with random string r and random oracle history s has a partner if there is some other copy of $\mathcal{P}$ with the same random string r but a different random oracle history $s' \neq s$. (Partnership is symmetric but not reflexive.)

Event E. In an execution of $\mathcal{S}^n$, the event E occurs if the environment with which $\mathcal{S}^n$ is interacting answers an extraction query for a copy of $\mathcal{P}$ that does not have a partner.

Depth. The depth of an extraction query in an execution of the prover $\mathcal{P}^n$ is the index of this query in the execution history: the first extraction query has depth 1, the second depth 2 etc.

A.2 Rewinding is exponential

The rest of the proof is a case distinction on an event that we call event E. If event E does not occur in an execution of $\mathcal{S}^n$ (and there are no collisions, the probability of which we already bounded above) then any successful extractor must take exponential time. Informally, if event E occurs then the extractor has taken a discrete logarithm somewhere; formally we build a metareduction that straight-line extracts from a single prover $\mathcal{P}$ and succeeds with the same probability as that of event E occurring.

Definition 21. *The event E occurs in an execution of $\mathcal{S}^n$ if the extractor answers an extraction query for a copy of $\mathcal{P}$ that does not have a partner: no other copy of $\mathcal{P}$ has been simulated that used the same random string r as the current one but got a different answer to at least one random oracle query.*

First, we deal with the case that event E does not occur. For a completed copy of $\mathcal{P}$, we say that it used parameters (r, s) if it ran on input r and s is the list of its random oracle queries and responses.

Lemma 22. *Consider a run of the simulator $\mathcal{S}^n$ and assume that neither a collision nor event E have occurred. Suppose that some simulated copy of $\mathcal{P}^n$ gets a correct answer to an extraction query at some depth k, for a copy of $\mathcal{P}$ with random string r and random oracle history s. Then there are at least 2^{k-1} copies of $\mathcal{P}^n$ that have been simulated by $\mathcal{S}^n$ so far which have completed their first loop and used (r, s) for the $(n - k + 1)$-st pass through this loop.*

Corollary 23. *As long as collisions and event E do not occur in a run of $\mathcal{S}^n$, any corretly answered extraction query at depth k implies that the extractor has* *launched at least 2^k copies of the prover $\mathcal{P}^n$. In particular, if the extractor wins the adaptive proof game against $\mathcal{S}^n$ then there must have been at least 2^n copies of $\mathcal{P}^n$ that were simulated by $\mathcal{S}^n$.*

We prove the corollary first. For the extractor to win, either a prover must have made an invalid proof — but the canonical prover never does this — or the extractor must have answered all the main prover's n queries correctly. In particular it has answered the main prover's query at depth n correctly, and this query must have a partner (or else we would have event E).

By Lemma 22, both the query and its partner must each have 2^{k-1} associated copies of $\mathcal{P}^n$. These copies cannot overlap: suppose the extractor answers some query for parameters (r, s) with partner (r, s') and $s \neq s'$. Then there are 2^{k-1} copies of the prover with (r, s) as their parameters for the $(n-k+1)$-st copy of $\mathcal{P}$ and 2^{k-1} copies of the prover $\mathcal{P}^n$ with (r, s') in the same position, for a total of 2^k copies of $\mathcal{P}^n$. q.e.d.

This is the proof of Lemma 22, by induction on k. For $k = 1$ the lemma says that there is at least one copy of $\mathcal{P}^n$ which ran on the parameters leading to the query in question, which is certainly true. Since the extraction queries are made in reverse order, a query at depth k corresponds to the $(n-k+1)$-st copy of $\mathcal{P}$ in a copy of $\mathcal{P}^n$. Suppose that case k is proven and consider an extraction query at depth $k+1$ with parameters (r, s) in some copy of $\mathcal{P}^n$. Since this copy of $\mathcal{P}^n$ has progressed to its $(k+1)$-st query, it must have got an answer to its k-th query, let's call the parameters in this k-th query (r', s'). Since this query was answered without event E occurring, it must have a partner. By the induction hypothesis, there are 2^{k-1} copies of $\mathcal{P}^n$ around that had their $(n-k+1)$-st copy of $\mathcal{P}$ run on parameters (r', s'). This means that all these copies must have used the same random strings and oracle histories for queries $1, \ldots, n-k$ or we would have a collision on r'. In particular, they used the same parameters for query $n-k = n-(k+1)+1$; these parameters correspond to the $(k+1)$-st extraction query so they must be (r, s). This gives us one half of the required 2^k copies to prove the induction step.

Let's look at the copy of $\mathcal{P}^n$ that made the extraction query on the partner (r', s'') of (r', s'). By the previous claim, this query must also have been at depth k or we would have a collision. Using the induction hypothesis, we find a further 2^{k-1} copies of $\mathcal{P}^n$ that have (r', s'') at their $(n-k+1)$-st position. These copies are distinct from the ones we found in the last step since their $(n-k+1)$-st random oracle history element is s'' which is distinct from s'. But by the same reasoning as above, these copies must have (r, s) at position $n-k$ so we have found 2^k copies with this property, concluding the induction and the proof of Lemma 22. q.e.d.

A.3 Adaptive proofs are straight line

Finally, we give a metareduction that works when event E does occur and the metareduction guesses correctly where to hide the challenge.

Lemma 24. *There is a metareduction M as follows. Take any adaptive extractor $\mathcal{K}$ that, in the n-proof game for any n, launches at most $f(n)$ copies of the adaptive prover, wins with probability at least $p(n)$; event E occurs with probability $e(n)$ given that the extractor wins. Then the metareduction M with black-box access to $\mathcal{K}$ is a programmable straight-line extractor against any non-adaptive prover $\mathcal{P}$ (with randomness space R) for the scheme in question with success probability at least*

$$\frac{e(n)p(n)}{n \cdot f(n)} - \frac{nf(n) + n^2 f(n)^2}{|R|}$$

Assume that F is a (pseudo-) random function $R \times L \to R$ and that $\mathcal{K}$ is an adaptive extractor as defined in the lemma. Consider an execution of $\mathcal{K}$ against the simulator $\mathcal{S}^n$ for the prover $\mathcal{P}^n$. The probability of a collision is bounded by $(m + m^2)/|R(n)|$ where m is the number of random values r drawn in the entire execution; since there are at most $f(n)$ simulated adaptive provers $\mathcal{P}^n$ each of which simulates n canonical provers, we have $m = n \cdot f(n)$.

We choose one of the up to $n \cdot f(n)$ provers $\mathcal{P}$ at random and replace it with the challenge instance from which we want to straight-line extract. The extractor may be choosing the random oracle responses of this instance if it is not part of the "main" copy of $\mathcal{P}^n$, this is fine as we are constructing a programming extractor. If there is no collision, with probability $e(n)/(n \cdot f(n))$ we placed the challenge in a prover that never obtains a partner and so we never have to rewind the challenge prover (if we guessed wrongly, we abort the metareduction). With probability at least $p(n)$, the execution will return the correct witness to the challenger (otherwise the extractor cannot win) and so the metareduction obtains the correct witness for the challenge copy of the prover $\mathcal{P}$, making it a straight-line extractor. q.e.d.

Now suppose that the extractor $\mathcal{K}$ is efficient. This means that (at least from some n_0 onwards) we have $f(n) < 2^n$, in which case the extractor winning implies event E has occurred by the corollary to Lemma 22. So we must have $e(n) = 1$ (from some n_0 onwards). If further $|R(n)|$ is exponentially large in n then the collision term is negligible in n — this is certainly the case for Sigma protocols where R must encompass at least the domain of the function ϕ; if this is not large enough then ϕ cannot even be one-way. This leaves us with a probability $p(n)/(n \cdot f(n)) - negl.$ that the metareduction succeeds.

B Proof of Theorem 18

First of all, we define a Polynomial Simulation Model.

Definition 25. *Polynomial Simulation Model (PSM) is a model defined in Figure 6 which provides the same interface as Generic Group Model.*

We simulate oracle queries as follows: let V be the set of indeterminants of size $v + 2$. At any step in the simulation, the algorithm has computed a list $F[1], ..., F[t]$ of linear polynomials in $k[V_1, ..., V_{v+2}]$, where $V = \{V_1, ..., V_{v+2}\}$, along with a list $z[1], ..., z[t]$ of values in $k \cup \{\perp\}$, list $e[1], ..., e[t]$ of values in S and also a polynomial D and a value $z_D \in \mathcal{W}$. One can observe that if we make a group operation in GGM, we can make an analogous operation on polynomials in PSM. We might think of $e[i]$ and $z[i]$ as representations of $\sigma(F[i](x,y))$ and preimage of $F[i](x,y)$ w.r.t. ϕ respectively.

We want to show that view of adversary in both models is identical, unless a *collision* occurs - this term will be introduced later. The main result is as follows:

Claim. Denote C as sequence of coin flips made by $\mathcal{A}$ and let $\mathcal{A}^C_{GGM}$ be an output of $\mathcal{A}$ in the generic group model. Similarly, we define $\mathcal{A}^C_{PSM}$ for the polynomial simulation model. Let $\alpha = |\Pr[\phi(\mathcal{A}^C_{GGM}) = x] - \Pr[\phi(\mathcal{A}^C_{PSM}) = x]|$. Then

$$\alpha = O((m+n)^2/|\mathcal{X}|).$$

Proof. Let us run $\mathcal{A}$ in GGM and PSM and let $N = \{\sigma_1, ..., \sigma_\alpha\}$ be the set of values σ_i in k such that when $\mathcal{A}$ used σ_i as input of oracle queries then σ_i had not been already on the encoding list (in GGM). We introduce a notion of collision.

Definition 26. *Let $x_1, ..., x_\alpha$ be elements of k so that $\sigma(x_i) = \sigma_i$ and $\mathcal{E}$ be an event such that there exist i, j so that $F[i] \neq F[j]$ and $F[i](x, y, x_1, ..., x_\alpha) = F[j](x, y, x_1, ..., x_\alpha)$. We say that $\mathcal{E}$ is a collision.*

First, we will find an upper bound on probability that a collision occurs.

Lemma 27. *$Pr[\mathcal{E}] = O((m+n)^2/|\mathcal{X}|)$.*

Proof. Let us choose i, j such that $F[i] \neq F[j]$. Note that the total degree of $F[i] - F[j]$ is at most 1 and $F[i](x, y, x_1, ..., x_\alpha) = F[j](x, y, x_1, ..., x_\alpha)$ is equivalent to $F[i](x, y, x_1, ..., x_\alpha) - F[j](x, y, x_1, ..., x_\alpha) = 0$. This occurs with probability at most $1/|\mathcal{X}|$ by Lemma 1. By checking all possible pairs we obtain an upper bound for $\Pr[\mathcal{E}]$, which is $O((m+n)^2/|\mathcal{X}|)$.

We are interested in view of adversary when $\bar{\mathcal{E}}$ holds i.e. when there is no collision. The following lemma states that view of adversary in GGM and PSM is identical given $\bar{\mathcal{E}}$.

Lemma 28. *$Pr[\phi(\mathcal{A}^C_{GGM}) = x|\bar{\mathcal{E}}] = Pr[\phi(\mathcal{A}^C_{PSM}) = x|\bar{\mathcal{E}}]$.*

Proof. Clearly, the probability of getting initial values for $\sigma(b_1), ..., \sigma(b_n), \sigma(x)$ and $\sigma(y)$ in both models are the same. Hence we just need to show that for any sequence of oracle queries, the probability that adversary gets exact values from oracles in GGM and PSM are equal. We formalise this idea: define $(D_1, D_2, .., D_t; d_1, ..., d_t)_X$ to be an event that $\mathcal{A}$ makes queries $D_1, ..., D_t$ (in this particular order) in model X and for query D_i, an oracle outputs d_i. We claim that for any queries $D_1, ..., D_t$ and any $d_1, ..., d_t$ we have

$$\Pr[(D_1, .., D_t; d_1, ..., d_t)_{GGM}] = \Pr[(D_1, .., D_t; d_1, ..., d_t)_{PSM}].$$

We will prove this statement by induction on t. Clearly the statement holds for $t = 0$. Suppose that it holds for some non-negative t.
Let $P_X = \Pr[(D_{t+1}; d_{t+1})_X | (D_1, .., D_t; d_1, ..., d_t)_X]$. Note that:

$$\Pr[(D_1, .., D_{t+1}; d_1, ..., d_{t+1})_X] = P_X \cdot \Pr[(D_1, .., D_t; d_1, ..., d_t)_X].$$

By inductive hypothesis we just need to show that $P_{GGM} = P_{PSM}$. Consider the cases for D_{t+1}:
Case 1 - $D_{t+1} = ADD(a, b)$ or $D_{t+1} = INV(a)$: since these two cases are very similar, we will only consider the case when the next query is an addition query. In PSM we have $a = e[i]$ and $b = e[j]$ for some i, j. Now, if polynomial $F[i]+F[j]$ has already been added to the list F i.e. $F[l] = F[i] + F[j]$, then it returns $e[l]$. Note that $e[l]$ has been an output of a query made before, hence $e[l] = d_u$ for some $u \leq t$. In GGM, however, adversary can expand the preimages of a and b in terms of $x, y, x_1, ..., x_\alpha$. Let p, q be elements of $\mathcal{X}$ so that $\sigma(p) = a$ and $\sigma(q) = b$. Then $\mathcal{A}$ can write down $a = f(x, y, x_1, ..., x_\alpha)$ and $b = g(x, y, x_1, ..., x_\alpha)$ for some linear polynomials f, g. The key observation is that $\{F[i] - f, F[j] - g\} \subset \text{Span}(D - z_D)$, where D and z_D are defined in PSM. Similarly, $\mathcal{A}$ can write down $c = h(x, y, x_1, ..., x_\alpha)$ for some linear polynomial h, where $\sigma(c) = e[l]$, and we have $F[l] - h \in \text{Span}(D - z_D)$, assuming no collision occurs. Hence we get $c = h(x, y, x_1, ..., x_\alpha) = f(x, y, x_1, ..., x_\alpha) + g(x, y, x_1, ..., x_\alpha) = p + q$. Therefore, given $\bar{\mathcal{E}}$ the ADD oracle will also return $e[l]$. Now, if $F[i] + F[j]$ has not been added to the list then in PSM the ADD oracle returns a random element in $S \backslash e$. So if $d_{k+1} \in S \backslash e$, then $P_{PSM} = 1/|S \backslash e|$ and 0 otherwise. Since no collision occurs, the oracle in GGM will also return a random value from S, distinct from the elements on the encoding list. Hence we conclude that $P_{PSM} = P_{GGM}$.
Case 2 - $D_{t+1} = FSS_{r,w}(c)$: when the oracle in PSM is given c, she returns a random value in $\mathcal{W}$. Therefore $P_{PSM} = 1/|\mathcal{W}|$. Now, we need to prove that $P_{GGM} = 1/|\mathcal{W}|$. Clearly, it is equivalent to showing that there are exactly $|\mathcal{W}|$ solutions (counting with order) to the equation: $r + cw = d_{t+1}$, where w, r are unknown. Note that we can choose w in $|\mathcal{W}|$ ways and then we just adjust r to get a solution. Hence $P_{GGM} = |\mathcal{W}|/|\mathcal{W}|^2 = 1/|\mathcal{W}| = P_{PSM}$. Thus the lemma holds.

Finally, by Lemmas 27 and 28 we get:

$$\begin{aligned}\alpha &= |\Pr[\phi(\mathcal{A}^C_{GGM}) = x|\mathcal{E}] \cdot \Pr[\mathcal{E}] + \Pr[\phi(\mathcal{A}^C_{GGM}) = x|\bar{\mathcal{E}}] \cdot \Pr[\bar{\mathcal{E}}] \\ &\quad - \Pr[\phi(\mathcal{A}^C_{PSM}) = x|\mathcal{E}] \cdot \Pr[\mathcal{E}] + \Pr[\phi(\mathcal{A}^C_{PSM}) = x|\bar{\mathcal{E}}] \cdot \Pr[\bar{\mathcal{E}}]| \\ &= \Pr[\mathcal{E}] \cdot |(\Pr[\phi(\mathcal{A}^C_{GGM}) = x|\bar{\mathcal{E}}] - \Pr[\phi(\mathcal{A}^C_{PSM}) = x|\bar{\mathcal{E}}])| \\ &\leq \Pr[\mathcal{E}] = O((m + n)^2/|\mathcal{X}|).\end{aligned} \tag{2}$$

Now, in order to show that a generic algorithm cannot win the $\Sigma-$one-wayness game in PSM with high probability, we just need the following lemma:

Lemma 29. *When the generic algorithm $\mathcal{A}$ terminates, we have $z[n + 1] = \perp$.*

Proof. Suppose that $z[n+1] \neq \perp$. We can define $W = \mathrm{Span}\{b_1, b_2, ..., b_s, D\}$ to be a subspace of $k[X, Y, X_1, ..., X_v]$ with $\dim(W) = s+1$. On the other hand, note that for every i we have $z[i] \neq \perp \iff F[i] \in W$. Indeed, this follows straight from the design of PSM. Thus $F[n+1] = X \in W$. Therefore, $F[n+2] = Y = (Y + cX) - cX = D - cX \in W$, for some $c \in k$. Note that $F[1], F[2], ..., F[s], F[n+1], F[n+2]$ are clearly linearly independent, so $s + 2 \leq \dim(W) = s+1$, contradiction.

The lemma above implies that a generic algorithm $\mathcal{A}$ is not able to win the game in Polynomial Simulation Model unless it randomly guesses the witness. We deduce that $\Pr[\phi(\mathcal{A}^{C}_{PSM}) = x] = O(|\ker(\phi)|/|\mathcal{W}|)$. So by this observation and Claim B the statement of Theorem 18 holds.

C Proof of Theorem 19

An algorithm is algebraic over a group G in the sense of Paillier and Vergnaud [4] if the only operations that it applies to group elements are the group operation and derived ones (e.g. inversion, exponentiation). We chose to define Sigma protocols over vector spaces rather than groups; one can interpret a $GF(p)$ vector space as an Abelian group to use the algebraic model. Vector addition becomes the group operation and scalar multiplication becomes group exponentiation; for the Schnorr protocol one recovers the usual definition over a group G with generator g (but loses sight of the fact that the linearity of $x \mapsto g^x$ is what makes the Sigma protocol work).

The algebraic model says that for any algebraic algorithm that has seen some group elements $g_1, \ldots, g_n \in G$ and outputs a value $\gamma \in G$, we can extract coefficients $c_1, \ldots, c_n$ such that $\gamma = \prod_{i=1}^{n} g_i^{c_i}$ (writing the group operation multiplicatively). In the language of vector spaces, any output can be expressed as a linear combination of the input elements.

Fix a group G with generator g. Suppose that there is an algebraic metareduction M that solves DLOG given access to a programming straight-line extractor for Schnorr proofs. Such an extractor has the following interface: random oracle queries are answered like a random oracle and on input a statement/proof pair $(x, (a, s))$ that is valid w.r.t. the random oracle queries so far, K outputs the witness w s.t. $x = g^w$.

M has access to the group generator g and can obtain a single group element h from its DLOG challenger. M wins if it can return the discrete logarithm u such that $h = g^u$. M may also have access to one or many copies of the extractor K (the argument is the same in both cases). For every group element γ that M outputs (to K), we assume that values α, β are available such that $\gamma = g^\alpha h^\beta$. We write this as $\gamma = [\alpha, \beta]$. Since the extractor K never outputs group elements, M cannot learn any more "bases" in the group G.

Now consider the following meta-metareduction μ. Let a challenge h be obtained from a DLOG challenger and give it to M if/when it requests a DLOG challenge. Whenever M outputs a valid proof $(x, (a, s))$ for extraction, if the group element x gives us a representation $[d, 0]$ (i.e. independent of h) then just return d as the witness. The first time M outputs a valid proof with x represented as $[\alpha, \beta]$ for $\beta \neq 0$, output the relevant $(x, (a, s))$ and halt. The meta-metareduction μ also answers random oracle queries on behalf of M by calling an external random oracle. Should M output a guess at u without ever making an extraction query depending on h, then μ returns u to the DLOG challenger — in this case M must have taken the discrete logarithm itself.

We claim that M cannot distinguish μ from an extractor K until the point where M outputs a proof depending on h. Indeed, the distribution of all elements "from K" is identical: random oracle responses are consistent and random; returned witnesses are the (unique) discrete logs of the relevant statements x.

Consider the case where M does not extract the discrete logarithm u of h without making at least one proof depending on h. Then the algorithm μ (containing M) is itself a random oracle model prover: it makes RO calls and outputs a proof $(x, (a, s))$, namely the first of M's proofs that depends on h i.e. x has representation $g^\alpha h^\beta$ with $\beta \neq 0$. We apply the forking lemma to μ to obtain the discrete logarithm w of x, with overwhelming probability. From this we can compute the discrete logarithm of h as $(w - \alpha)/\beta$ (in the field $GF(p)$ where p is the group order).

It follows that if M is a metareduction from a programming extractor to DLOG with advantage α then μ is a meta-metareduction to DLOG directly with advantage $c\alpha$ (c is the forking lemma factor — a non-negligible constant). In other words, a reduction from a programming extractor to DLOG can only exist if DLOG is already easy.

The very same argument works for a metareduction M from an adaptive extractor for Fiat-Shamir-Schnorr to DLOG: if M never outputs a proof depending on the challenge h then we can simulate the extractor; otherwise, let μ be the algorithm that runs M and the simulated extractor up to the first proof that does depend on h, then apply the forking lemma.

```
$initialise$:
    $V \leftarrow [X, Y]$
    $v \leftarrow 0$
    $F \leftarrow [b_1, b_2, ..., b_n, X, Y]$
    $z \leftarrow [a_1, a_2, ..., a_s]$
    for $i = s+1, ..., n+2$
        $z \leftarrow z :: \perp$
    $\sigma_1, \sigma_{n+1}, ..., \sigma_{n+2} \twoheadleftarrow S$
    $e \leftarrow [\sigma_1]$
    for $i = 2, ..., n$
        $e \leftarrow e :: (a \twoheadleftarrow S \backslash e)$
    $e \leftarrow e :: [\sigma_{n+1}, \sigma_{n+2}]$
    $D = 0 \in k[X, Y]$
    $z_D = 0 \in \mathcal{W}$
    $(\sigma_1, \sigma_2, ..., \sigma_{n+2})$


$newvar(s)$:
    $v \leftarrow v + 1$
    $V \leftarrow V :: X_v$
    $F \leftarrow F :: X_v$
    $z \leftarrow z :: \perp$
    $e \leftarrow e :: s$


$FSS_{r,w}(c)$:
    $r \twoheadleftarrow \mathcal{W}$
    $z_D \leftarrow r$
    $D \leftarrow Y + cX$
    for $i = 1, ..., |F|$
        if $update(f) \neq \perp$
            $z[i] \leftarrow update(f)$
    return $r$

$update(f)$:
    if $f = qD + \sum_{i=1}^{s} p_i b_i$ for some $p_1, ..., p_s, q \in k$ then
        return $q z_D + \sum_{i=1}^{s} p_i a_i$
    else
        return $\perp$

$add(a, b)$:
    if $a \notin e$ then
        $newvar$(a)
    if $b \notin e$
        $newvar$(b)
    $f \leftarrow F[i] + F[j]$, where $e[i] = a$ and $e[j] = b$
    if $f = F[l]$ for some $l$ then
        return $e[l]$
    else
        $F \leftarrow F :: f$
        $z \leftarrow z :: update(f)$
        $s \twoheadleftarrow S \backslash e$
        $e \leftarrow e :: s$
        return $s$

$inv(a)$:
    if $a \notin e$ then
        $newvar$(a)
    $f \leftarrow -F[i]$, where $e[i] = a$
    if $f = F[l]$ for some $l$ then
        return $e[l]$
    else
        $F \leftarrow F :: f$
        $z \leftarrow z :: update(f)$
        $s \twoheadleftarrow S \backslash e$
        $e \leftarrow e :: s$
        return $s$
```

Fig. 6. Polynomial Simulation Model