Adaptive Proofs of Knowledge in the Random Oracle Model

David Bernhard[1(✉)], Marc Fischlin[2], and Bogdan Warinschi[1]

[1] University of Bristol, Bristol, UK
{bernhard,bogdan}@compsci.bristol.ac.uk
[2] Technische Universität Darmstadt, Darmstadt, Germany
marc.fischlin@cryptoplexity.de

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 [5,22,31,50] 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

J. Katz (Ed.): PKC 2015, LNCS 9020, pp. 629–649, 2015.
DOI: 10.1007/978-3-662-46447-2_28

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 [47].

Shoup and Gennaro [47] 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 [13,17,36, 38,43,52], 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 [43, 51], is that the scheme is non-malleable (NM-CPA) [12].

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 [26] 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 [25] 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 [47]. 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 [14].

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. A longer discussion can be found in the full version of our paper. The notion of simulation-soundness of zero-knowledge proofs has been introduced for proofs of membership by Sahai

[41], showing that the Naor-Yung paradigm [36] yields CCA secure encryptions in the common reference string model. In the context of proofs of knowledge, De Santis and Persiano [19] 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 [33], Chase and Lysanskaya [15], and by Dodis et al. [20], 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. [12] and Faust et al. [21]; 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 [21] 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 [9] 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.

2.1 Proof Schemes

Below we give the common definition for proof schemes for NP relations R. For Fiat-Shamir proof schemes we occasionally also need to parametrise over the underlying number-theoretic group $\mathbb{G}(\lambda)$ in a straightforward way, but we omit this for sake of representational simplicity.

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.

2.2 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 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.*

2.3 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 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

Game G_1	**Game** G_2
initialise():	initialise():
//potentially generate parameters	//potentially generate parameters
$\mathcal{A}$ issues prove(x, w):	$\mathcal{A}$ issues prove(x, w):
if $\neg R(x, w)$ then return $\perp$	if $\neg R(x, w)$ then return $\perp$
$\pi \leftarrow \mathcal{P}^{\mathsf{RO}}(x, w)$	$\pi \leftarrow \mathcal{S}.\mathsf{prove}(x)$
return π	return π
$\mathcal{A}$ issues ro(x):	$\mathcal{A}$ issues ro(x):
return $\mathsf{RO}(x)$	return $\mathcal{S}.\mathsf{ro}(x)$

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.

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 [26].

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.*

The above definition is less general than the one first proposed by Bellare and Goldreich [5]. 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

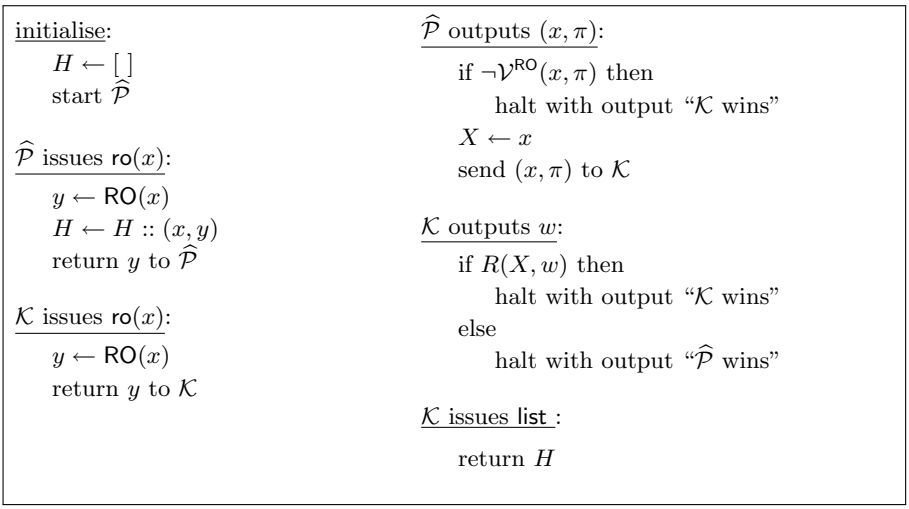

Fig. 2. The game G defining proofs of knowledge in the random oracle model. Capital-X is part of the game's internal state that persists between calls (so that the extractor's witness is verified against the same statement that the prover provided earlier).

the prover making a valid proof is no longer well-defined, since it also depends on the extractor's response to earlier proofs.

2.4 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.

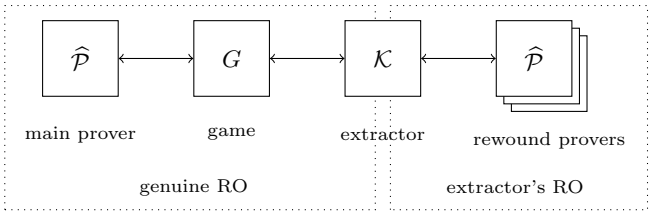

Fig. 3. Extending the straight-line proof of knowledge game to the rewinding case

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.*

2.5 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 [41] 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) [33]. 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 [50] 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*

initialise:
$H \leftarrow [\,]$; $\Pi \leftarrow [\,]$
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 $\mathcal{K}$

$\mathcal{K}$ calls list:
return (H, Π)

$\widehat{\mathcal{P}}$ outputs (x, π):
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 $R(X, w)$ then
halt with output "$\mathcal{K}$ wins"
else
halt with output "$\widehat{\mathcal{P}}$ wins"

$\widehat{\mathcal{P}}$ 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. 4. The game G defining SSE in the random oracle model.

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.

$\mathcal{K}$ calls $\widehat{\mathcal{P}}$ for the first time:
start $\widehat{\mathcal{P}}$

$\mathcal{K}$ sends a value z:
send z to $\widehat{\mathcal{P}}$

$\widehat{\mathcal{P}}$ calls $\mathsf{ro}(x)$:
send ("ro", x) to $\mathcal{K}$

$\widehat{\mathcal{P}}$ outputs (x, π):
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.

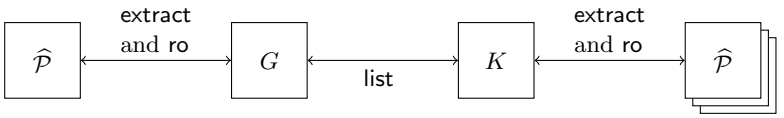

Fig. 6. The adaptive proof game and the queries that the various algorithms can exchange

3.1 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.

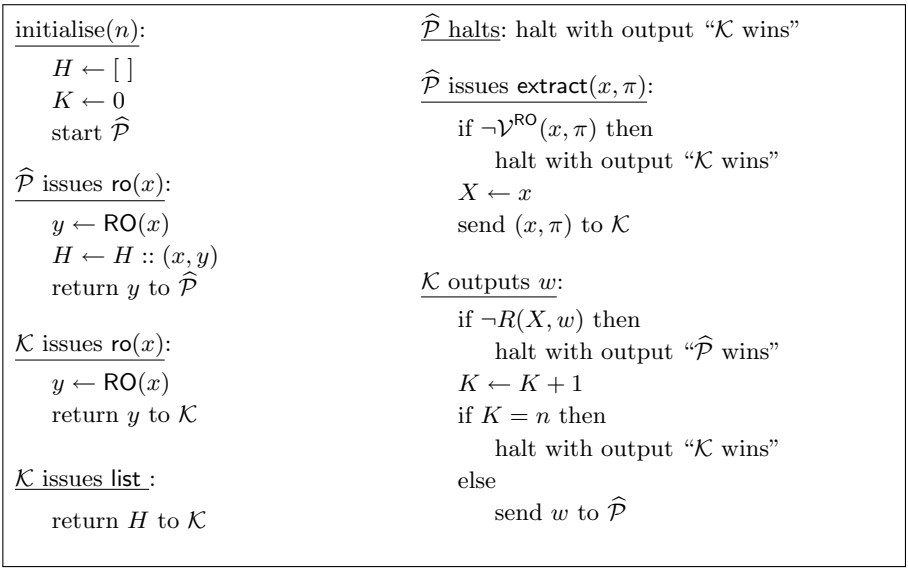

Fig. 7. The game G for adaptive proofs with parameter n

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.

3.2 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.

initialise(n):
$H \leftarrow [\,]$; $\Pi \leftarrow [\,]$
$K \leftarrow 0$
start $\widehat{\mathcal{P}}$

$\widehat{\mathcal{P}}$ issues ro(x):
$C \leftarrow$ "prover"; $I \leftarrow x$
send x to $\mathcal{S}$.ro

$\mathcal{K}$ issues ro(x):
$C \leftarrow$ "extractor"
send x to $\mathcal{S}$.ro

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

$\mathcal{S}$.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

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.

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 here.

4.1 Adaptive Proofs Exist

First, we establish that simulation-sound adaptive proofs in the random oracle model exist. An existing construction due to Fischlin [26] 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.

The following theorem shows that Fischlin proofs are adaptively secure.

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

4.2 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* $\mathfrak{E}.\mathsf{Encrypt}$*.*

$\mathsf{KeyGen}()$:	$\mathsf{Encrypt}(pk, m)$:		$\mathsf{Decrypt}(sk, c)$:
$(pk, sk) \leftarrow \mathfrak{E}.\mathsf{KeyGen}()$	$r \leftarrow\!\!\$\ RS$		parse c as (e, π)
return (pk, sk)	$c \leftarrow \mathfrak{E}.\mathsf{Encrypt}(pk, m; r)$		if $\mathfrak{P}.\mathcal{V}((pk, e), \pi) = 0$ then
	$w \leftarrow \mathsf{W}(pk, c, m, r)$		return $\perp$
	$\pi \leftarrow \mathfrak{P}.\mathcal{P}((pk, c), w)$		$m \leftarrow \mathfrak{E}.\mathsf{Decrypt}(sk, e)$
	return (c, π)		return m

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.

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 $G^s = A + \mathcal{H}(pk, e, A) \cdot u$

(mod 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 $\mathsf{W}(pk,(u,v),m,r) := r$.

4.3 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.

4.4 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 [47]. 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.

Acknowledgments. 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 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. Abe, M.: Combining Encryption and Proof of Knowledge in the Random Oracle Model. The Computer Journal **47**(1), 58–70 (2004)
2. Abdalla, M., Bellare, M., Rogaway, P.: The oracle diffie-Hellman assumptions and an analysis of DHIES. In: Naccache, D. (ed.) CT-RSA 2001. LNCS, vol. 2020, p. 143. Springer, Heidelberg (2001)
3. Adida, B.: Helios: web-based open-audit voting. In: 17th USENIX security symposium, pp. 335–348. Helios website (2008). http://heliosvoting.org paper: http://www.usenix.org/events/sec08/tech/full_papers/adida/adida.pdf
4. Bagherzandi, A., Cheaon, J.H., Jarecki, S.: Multisignatures secure under the discrete logarithm assumption and a generalized forking lemma. In: CCS 2008, pp. 449–458. ACM press (2008)
5. Bellare, M., Goldreich, O.: On defining proofs of knowledge. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 390–420. Springer, Heidelberg (1993)
6. Bellare, M., Goldreich, O.: On probabilistic versus deterministic provers in the definition of proofs of knowledge. In: Goldreich, O. (ed.) Studies in Complexity and Cryptography. LNCS, vol. 6650, pp. 114–123. Springer, Heidelberg (2011)
7. Bellare, M., Namprempre, C., Pointcheval, D., Semanko, M.: The One-More-RSA-Inversion Problems and the Security of Chaum's Blind Signature Scheme. J. Cryptology **16**(3), 185–215 (2003). Springer

8. Bellare, M., Neven, G.: Multi-signatures in the plain public-key model and a general forking lemma. In: Proceedings of ACM Conference on Computer and Communications Security, pp. 390–399 (2006)
9. Bellare, M., Rogaway, P.: Random oracles are practical: a paradigm for designing efficient protocols. In: ACM Conference on Computer and Communications Security, pp. 62–73. ACM (1993)
10. Bellare, M., Rogaway, P.: The security of triple encryption and a framework for code-based game-playing proofs. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 409–426. Springer, Heidelberg (2006). Full version of 27 November 2008 (Draft 3.0) at eprint.iacr.org/2004/331
11. Bellare, M., Sahai, A.: Non-malleable encryption: equivalence between two notions, and an indistinguishability-based characterization. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 519–536. Springer, Heidelberg (1999)
12. Bernhard, D., Pereira, O., Warinschi, B.: How not to prove yourself: pitfalls of the Fiat-Shamir Heuristic and applications to Helios. In: Wang, X., Sako, K. (eds.) ASIACRYPT 2012. LNCS, vol. 7658, pp. 626–643. Springer, Heidelberg (2012)
13. Blum, M., Feldman, P., Micali, S.: Non-interactive zero-knowledge and its applications. In: Proceedings of the twentieth annual ACM symposium on theory of computing (STOC 1990), pp. 103–112 (1988)
14. Canetti, R., Goldreich, O., Goldwasser, S., Micali, S.: Resettable zero-knowledge. In: STOC, pp. 235–244. ACM Press (2000)
15. Chase, M., Lysyanskaya, A.: On signatures of knowledge. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 78–96. Springer, Heidelberg (2006)
16. Cramer, R., Shoup, V.: A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, p. 13. Springer, Heidelberg (1998)
17. Damgård, I.B.: Towards practical public key systems secure against chosen ciphertext attacks. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 445–456. Springer, Heidelberg (1992)
18. De Santis, A., Di Crescenzo, G., Ostrovsky, R., Persiano, G., Sahai, A.: Robust non-interactive zero knowledge. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, p. 566. Springer, Heidelberg (2001)
19. De Santis, A., Persiano, G.: Zero-knowledge proofs of knowledge without interaction (extended abstract). In: FOCS, pp. 427–436 (1992)
20. Dodis, Y., Haralambiev, K., López-Alt, A., Wichs, D.: Efficient public-key cryptography in the presence of key leakage. In: Abe, M. (ed.) ASIACRYPT 2010. LNCS, vol. 6477, pp. 613–631. Springer, Heidelberg (2010)
21. Faust, S., Kohlweiss, M., Marson, G.A., Venturi, D.: On the non-malleability of the Fiat-Shamir transform. In: Galbraith, S., Nandi, M. (eds.) INDOCRYPT 2012. LNCS, vol. 7668. Springer, Heidelberg (2012)
22. Feige, U., Fiat, A., Shamir, A.: Zero-knowledge proofs of identity. Journal of Cryptology **1**(2), 77–94 (1988)
23. Feige, U., Lapidot, D., Shamir, A.: Multiple non-interactive zero knowledge proofs based on a single random string (extended abstract). In: FOCS, pp. 308–317 (1990)
24. Feige, U., Shamir, A.: Zero knowledge proofs of knowledge in two rounds. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 526–544. Springer, Heidelberg (1990)
25. Fiat, A., Shamir, A.: How to prove yourself: practical solutions to identification and signature problems. In: Odlyzko, A.M. (ed.) CRYPTO 1986. LNCS, vol. 263, pp. 186–194. Springer, Heidelberg (1987)

26. Fischlin, M.: Communication-efficient non-interactive proofs of knowledge with online extractors. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 152–168. Springer, Heidelberg (2005)
27. Fouque, P.-A., Pointcheval, D.: Threshold cryptosystems secure against chosen-ciphertext attacks. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, p. 351. Springer, Heidelberg (2001)
28. Fujisaki, E., Okamoto, T.: Secure Integration of Asymmetric and Symmetric Encryption Schemes. J. Cryptology **26**(1), 80–101 (2013)
29. Garay, J.A., MacKenzie, P.D., Yang, K.: Strengthening Zero-Knowledge Protocols Using Signatures. J. Cryptology **19**(2), 169–209 (2006). Springer
30. Garg, S., Bhaskar, R., Lokam, S.V.: Improved bounds on security reductions for discrete log based signatures. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 93–107. Springer, Heidelberg (2008)
31. Goldwasser, S., Micali, S., Rackoff, C.: The Knowledge Complexity of Interactive Proof Systems. SIAM J. Comput. **18**(1), 186–208 (1989)
32. Goldreich, O., Ostrovsky, R.: Software Protection and Simulation on Oblivious RAMs. J. ACM **43**(3), 431–473 (1996)
33. Groth, J.: Simulation-sound NIZK proofs for a practical language and constant size group signatures. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 444–459. Springer, Heidelberg (2006)
34. Kiltz, E., Malone-Lee, J.: A General Construction of IND-CCA2 Secure Public Key Encryption. IMA Int. Conf. 152–166 (2003)
35. Lindell, Y.: A Simpler Construction of CCA2-Secure Public-KeyEncryption under General Assumptions. J. Cryptology **19**(3), 359–377 (2006). Springer
36. Naor, M., Yung, M.: Public-key cryptosystems provably secure against chosen ciphertext attacks. In: Proceedings of the twenty-second annual ACM symposium on theory of computing (STOC 1990), pp. 42–437 (1990)
37. Pointcheval, D., Stern, J.: Security Arguments for Digital Signatures and Blind Signatures. J. Cryptolog **13**(3), 361–396 (2000). Springer
38. Rackoff, C., Simon, D.R.: Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 433–444. Springer, Heidelberg (1992)
39. Sahai, A.: Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In: Proceedings of the 40th annual symposium on foundations of computer science (FOCS 1999), pp. 543–553 (1999)
40. Schnorr, C.P.: Efficient signature generation for smart cards. Journal of cryptology **4**, 161–174 (1991). Springer
41. Schnorr, C.-P., Jakobsson, M.: Security of signed elgamal encryption. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, p. 73. Springer, Heidelberg (2000)
42. Seurin, Y.: On the exact security of schnorr-type signatures in the random oracle model. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 554–571. Springer, Heidelberg (2012)
43. Seurin, Y., Treger, J.: A robust and plaintext-aware variant of signed elgamal encryption. In: Dawson, E. (ed.) CT-RSA 2013. LNCS, vol. 7779, pp. 68–83. Springer, Heidelberg (2013)
44. Shoup, V.: A Proposal for an ISO Standard for Public Key Encryption. Version 2.1 (2001). www.shoup.net
45. Shoup, V., Gennaro, R.: Securing threshold cryptosystems against chosen ciphertext attack. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 1–16. Springer, Heidelberg (1998)

46. Shoup, V., Gennaro, R.: Securing Threshold Cryptosystems against Chosen Ciphertext Attack. J. Cryptology **15**(2), 75–96 (2002). Springer
47. Tompa, M., Woll, H.: Random self-reducibility and zero knowledge interactive proofs of possession of information. In: FOCS, pp. 472–482 (1987)
48. Tsiounis, Y., Yung, M.: On the security of elgamal based encryption. In: Imai, H., Zheng, Y. (eds.) PKC 1998. LNCS, vol. 1431, p. 117. Springer, Heidelberg (1998)
49. Wee, H.: Zero knowledge in the random oracle model, revisited. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 417–434. Springer, Heidelberg (2009)
50. Groth, J., Sahai, A.: Efficient non-interactive proof systems for bilinear groups. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 415–432. Springer, Heidelberg (2008)
51. Wikström, D.: Simplified submission of inputs to protocols. In: Ostrovsky, R., De Prisco, R., Visconti, I. (eds.) SCN 2008. LNCS, vol. 5229, pp. 293–308. Springer, Heidelberg (2008)
52. Zheng, Y., Seberry, J.: Practical approaches to attaining security against adaptively chosen ciphertext attacks. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740. Springer, Heidelberg (1992)