Some results concerning security in the Random Oracle Model

by

Victor Glazer

A thesis submitted in conformity with the requirements
for the degree of Master of Science
Graduate Department of Computer Science
University of Toronto

Abstract

Some results concerning security in the Random Oracle Model

Victor Glazer

Master of Science

Graduate Department of Computer Science

University of Toronto

2005

The Random Oracle Model (ROM) is a setting where all parties, including the adversary, have black-box access to a "truly random function" (the random oracle). In this thesis, we present two results concerning security in the ROM. First, we show that, for every canonical identification scheme, the corresponding Fiat-Shamir signature scheme is secure in the ROM. Previously, only "non-trivial" canonical identification schemes were known to yield Fiat-Shamir signature schemes which are secure in the ROM. Second, we show how to modify a certain discrete logarithm-based public-key encryption scheme so that it becomes CCA2-secure in the ROM. In conclusion, we review several "uninstantiability" results which demonstrate that security in the ROM does *not* guarantee "real-world" security, and briefly survey a number of signature and public-key encryption schemes which *are* secure in the "real world".

For Eleyna

Acknowledgements

First and foremost, I would like to thank my supervisor, Charles Rackoff. Working with Charlie has been an amazing experience. This thesis is as much his as mine, in that he both helped me decide what problems to tackle and provided crucial insights into their solution.

I would also like to thank my second reader, Ian F. Blake, for his helpful comments and suggestions.

I am grateful to Allan Borodin and Faith Fich for introducing me to the manifold joys of theoretical Computer Science, and to Rudi Mathon and Christina Christara for encouraging me to pursue graduate studies.

Last but not least, I want to thank my awesome wife Eleyna and my parents Michael and Rimma. I wouldn't have made it through two long years of graduate school without their constant love and support.

Contents

Chapter 1

Introduction

Modern cryptography has computational complexity at its foundation. In order to gain confidence in the security of a cryptographic construction, we show that *every* polynomially-bounded adversary which succeeds in "breaking" it can be used to solve a computational problem widely believed to be "hard on average", say integer factorization ([Len00]) or the discrete logarithm problem ([Odl00]). This means that no such "breaker" exists, provided the problem in question is indeed "hard".

Cryptographers make two kinds of "hardness" assumptions: ones asserting the *difficulty* of specific (usually number-theoretic) problems, and ones asserting the *existence* of secure cryptographic primitives such as "one-way functions" (see Chapter 2, Section 2) or "trapdoor permutations" (see Chapter 2, Section 4). Assumptions of the first kind enable us to prove the security of constructions which are efficient enough to be practical. Unfortunately, the particular problem we assume to be "hard" might later turn out to be "easy", rendering the protocol insecure. Assumptions of the second kind (often called "general assumptions") often lead to constructions which are too inefficient to be of practical interest. Although such constructions are in a sense mere "proofs of concept", their security is guaranteed as long as *any* secure primitives of the relevant kind exist.

Broadly speaking, cryptographic problems fall into two categories: "public-key" and "private-key". In both cases, two or more people wish to securely interact over an insecure channel, which is either controlled or monitored by the adversary.

In the private-key setting, the participants share a common secret pri, unknown to the adversary. The intuition is that they are "friends who trust each other". In contrast, in the public-key setting each person has associated with him both private information pri, known only to himself, and public information pub, known to everyone (including the adversary). Here the intuition is that the participants are "mutually mistrustful strangers". Many important cryptographic problems, including "signature schemes" (see Chapter 2, Section 5) and "encryption schemes" (see Chapter 2, Section 7), come in both public-key and private-key flavours.

Today we have extremely efficient private-key constructions which are secure if "block ciphers" such as DES ([NIS99]) and AES ([NIS01]) are "pseudorandom", as well as fairly inefficient private-key constructions which are secure if "one-way functions" exist ([GGM84a], [GGM84b], [GGM86], [HILL99]). It could therefore be argued that private-key cryptography is now largely "an engineering problem". Unfortunately, that is not yet the case for public-key cryptography.

Beginning in the late eighties, much work was done on formulating the "right" definitions of public-key security and showing that constructions which are secure according to these definitions can be obtained from "trapdoor permutations". Such constructions were available by the early nineties ([GMR88], [Rom90], [NY90], [RS92]), but from a practical standpoint their efficiency left a lot to be desired. Numerous attempts were also made to come up with efficient constructions which are secure under some variant of the popular "RSA" (factoring-related) and "Diffie-Hellman" (discrete log-related) assumptions ([Bon99],[MW00]), but they were not successful. Lacking viable alternatives, practitioners mostly relied on ad hoc approaches of dubious security, many of which were eventually broken ([Bri85], [Ble98]).

In [BR93], Mihir Bellare and Phillip Rogaway introduced the "Random Oracle Methodology" in an effort to bridge the gap between cryptographic theory and practice. Informally, the Random Oracle Model, or ROM for short, is a setting where all parties (including the adversary) have black-box access to a "truly random function" (the random oracle). Although it has other applications in complexity theory, notably to Micali's non-interactive "CS proofs" ([Mic94], [Mic00]), the ROM is usually encountered in the context of public-key cryptography.

The Random Oracle Methodology is a two-step procedure for obtaining practical public-key constructions. In the first step, one designs an efficient construction which is secure in the ROM under some standard hardness assumption, say the "Computational Diffie-Hellman" assumption. Because of the many nice properties enjoyed by random oracles, this generally isn't too difficult. In the second step, one "instantiates" the random oracle $\mathcal{R}$ using a "cryptographic hash function" h; a reasonable choice for h might be SHA-256 ([NIS04]). Thereafter, whenever $\mathcal{R}$ is queried on a string s, the answer is $h(s)$. A heuristic justification for this step is that good cryptographic hash functions hopefully behave "a lot like" random oracles. However, as we will see in Chapter 5, $\mathcal{R}$ should strictly speaking be instantiated using an *ensemble* $\mathcal{H} = \{\mathcal{H}_n\}_{n \in \mathbb{N}}$ of hash function families (see Chapter 2, Section 3) rather than a single function h.

Because in the first step we have the powerful random oracle primitive at our disposal, it is natural to question the need to make any hardness assumptions at all. However, as shown in [IR89], if a "key exchange protocol" which is secure in the ROM exists then $P \neq NP$. Since it is easy to securely exchange a key using a secure public-key encryption scheme, proving that such a scheme is secure in the ROM without making any additional assumptions is therefore prohibitively difficult. But if we are willing to make additional assumptions, why not simply make one strong enough to eliminate the need for the random oracle altogether? Ronald Cramer and Victor Shoup developed a fairly efficient public-key encryption scheme ([CS98]) which is secure in the "real world" under just such a "non-standard-yet-plausible" hardness assumption, namely the "Decisional Diffie-

Hellman assumption".

On the other hand, no hardness assumptions are necessary in order to show that signature schemes which are secure in the ROM exist. Since random oracles are one-way (see Chapter 2, Section 8), we can replace the one-way function evaluations in Rompel's construction ([Rom90]) with $\mathcal{R}$ queries. While the resulting construction is admittedly quite inefficient (in the sense that it requires many random oracle queries), it appears that one can't do much better without making hardness assumptions.

As for signature schemes which are secure in the ROM under standard hardness assumptions such as the "RSA assumption", for example the schemes presented in [BR93] and [BR94], their benefits are less clear today. Although they are considerably more efficient than both constructions which are secure in the "real world" under standard assumptions ([DN94],[Cr96]) and constructions which are secure in the ROM unconditionally, we now have constructions of comparable efficiency which are secure in the "real world" under "non-standard-yet-plausible" assumptions like the "Strong RSA assumption" ([CS99], [GHR99], [Fis03]).

What sort of security does the Random Oracle Methodology guarantee? Informally, hash functions are efficiently evaluable and thus have a short description, which means that they cannot be "truly random". It is therefore unclear why security should be preserved when the random oracle is "instantiated" using a hash function. Nonetheless, security in the ROM was at first believed to provide "strong evidence" of real-world security. However, in [CGH98] Canetti, Goldreich and Halevi exhibited a signature scheme and a public-key encryption scheme which are secure in the ROM yet insecure in the "real world", *no matter what hash function is used to "instantiate" the random oracle*; such schemes are said to be "uninstantiable" (see Chapter 5). From a theoretical standpoint, this result conclusively demonstrated that security in the ROM *does not* imply real-world security. However, since Canetti et al.'s constructions were rather contrived and quite inefficient, practitioners remained unconvinced.

Several additional uninstantiability results have emerged since, arguably the most significant being Goldwasser and Tauman-Kalai's proof ([GTK03]) that there exist uninstantiable "Fiat-Shamir signature schemes" (see Chapter 3, Section 1 for an overview of Fiat-Shamir signature schemes). Like Canetti et al.'s, Goldwasser and Tauman-Kalai's constructions are contrived and inefficient. Worse still, their actual proof has a somewhat non-constructive flavour (see Chapter 5, Section 4). However, since Fiat-Shamir signature schemes are widely used in practice, Goldwasser and Tauman-Kalai's result can be viewed as dealing the Random Oracle Methodology a more severe blow than Cenetti et al.'s.

Chapter Outline

Chapter 1 is this introduction.

Chapter 2 contains definitions of the relevant cryptographic primitives, including one-way functions, signature schemes, identification schemes, trapdoor permutations and public-key encryption schemes.

Chapter 3 concerns the security of Fiat-Shamir signature schemes in the ROM. We first present an earlier result ([AABN02]) demonstrating that every "passively secure non-trivial canonical identification scheme" yields a "Fiat-Shamir signature scheme" which is secure in the ROM. We then show that, for "actively secure" schemes, the "non-triviality" assumption is not necessary. Namely, we prove that, for every "actively secure canonical identification scheme" (non-trivial or not), the corresponding Fiat-Shamir signature scheme is secure in the ROM.

Chapter 4 describes a certain public-key encryption scheme which is "CCA2-secure" in the ROM. We first present an earlier version of the scheme, proposed in [BR97], which was initially claimed to be CCA2-secure in the ROM under the "Computational Diffie-Hellman assumption". It was later pointed out in [ABR01a] that the original proof of security was flawed. We then show how to modify the scheme so that it is indeed CCA2-secure in the ROM under the "Computational Diffie-Hellman" assumption.

Chapter 5 sketches Canetti, Goldreich and Halevi's seminal result that "uninstantiable" signature and public-key encryption schemes exist ([CGH98]) and presents Maurer, Renner and Holenstein's recent simple proof thereof ([MRH04]). Goldwasser and Tauman-Kalai's proof that there exist uninstantiable Fiat-Shamir signature schemes ([GTK03]) is also discussed.

Chapter 6 briefly surveys a number of practical signature and public-key encryption schemes which are secure in the "real world" under either standard or non-standard-yet-quite-plausible hardness assumptions.

Chapter 2

Preliminaries

1 Negligible and non-negligible functions

A function $\mu : \mathbb{N} \to \mathbb{R}$ is *negligible* (in n) if it goes to zero faster than any inverse polynomial $\frac{1}{p(n)}$ in n. In other words, for every $c \in \mathbb{N}$ there exists an $n_0 \in \mathbb{N}$ such that $\mu(n) < \frac{1}{n^c}$ for all $n \geq n_0$. If μ is not negligible it is said to be *non-negligible* (in n). In that case there exists a $d \in \mathbb{N}$ such that $\mu(n) > \frac{1}{n^d}$ for infinitely many n (not necessarily contiguous).

If definitional robustness is desired, negligible and non-negligible functions are a natural choice for formalizing the intuitive notions of "insignificant" and "significant" probabilities when dealing with polynomial-time adversaries.

2 One-way functions

One-way functions are a cryptographic primitive of fundamental importance. Informally, a function mapping strings to strings is *one-way* if it is "easy to evaluate" but "hard to invert on average". Formally, a function $f : \{0,1\}^* \to \{0,1\}^*$ is *one-way* if it is computable in deterministic polynomial time and, for every probabilistic polynomial-time "inverter" INV, $p_{INV}(n)$ is negligible. Here $p_{INV}(n)$ is the probability that, given

1^n and $y = f(x)$ for a random $x \in \{0,1\}^n$, INV outputs an $x' \in \{0,1\}^n$ such that $f(x') = y$; $p_{INV}(n)$ is taken over the choice of $x \in \{0,1\}^n$ and the random bits of INV.

Observe that if $P = NP$ then every function f computable in deterministic polynomial time can be easily inverted by non-deterministically guessing an $x' \in \{0,1\}^n$ such that $f(x') = y$. Proving the existence of one-way functions is therefore no easier than proving $P \neq NP$.

3 Hash function ensembles

A *hash function* h is simply an efficiently-evaluable function mapping $\{0,1\}^*$ to $\{0,1\}^n$, where n is some security parameter. "Cryptographic" hash functions such as SHA-256 ([NIS04]) are informally believed to "hide all information about their input". More rigorously, hash functions are often assumed to be "collision resistant" or "collision intractable", meaning that it's infeasible to find two domain elements which have the same image under h. Formally, however, it doesn't make sense to assert that collisions in SHA-256 or any other fixed hash function are hard to find, since they can always be built into the code of the finder machine. Instead, we prefer to talk about hash function ensembles.

A *hash function ensemble* $\mathcal{H} = \{\mathcal{H}_n\}_{n\in\mathbb{N}}$ is a collection of hash function families $\mathcal{H}_n = \{h_s : \{0,1\}^* \to \{0,1\}^n\}_{s\in\{0,1\}^n}$[1]. $\mathcal{H}$ is *efficiently evaluable* in the sense that there exists a (deterministic) polynomial-time Turing machine $M_\mathcal{H}$ such that $M_\mathcal{H}(s,x) = h_s(x)$ for all $s \in \{0,1\}^n$ and $x \in \{0,1\}^*$. We say that $\mathcal{H}$ is *collision resistant* if, for every probabilistic polynomial time "collision finder" F who is given a randomly chosen $s \in \{0,1\}^n$, the probability $p_F(n)$ that F outputs $x_1, x_2 \in \{0,1\}^*$, $x_1 \neq x_2$ such that $h_s(x_1) = h_s(x_2)$ is negligible; here $p_F(n)$ is taken over the choice of s and the random bits of F.

It is worth pointing out that collision-resistance implies a kind of one-wayness (see Section 2). Suppose that we have a probabilistic polynomial-time inverter INV who,

[1]In general, h_s maps $\{0,1\}^*$ to $\{0,1\}^{\ell(n)}$, where $\ell(n) \leq n^c$ for some c. However, we will usually assume that $\ell(n) \equiv n$ to simplify the presentation.

given a randomly chosen $s \in \{0,1\}^n$ and $y = h_s(x) \in \{0,1\}^n$ for a randomly chosen $x \in \{0,1\}^{n+1}$, outputs with non-negligible (in n) probability an $x' \in \{0,1\}^{n+1}$ such that $h_s(x') = y$; here the probability is taken over the choice of s and x, as well as the random bits of INV. Notice that, provided $x' \neq x$, (x, x') is a collision in h_s. Also, since $\frac{|\{0,1\}^{n+1}|}{|\{0,1\}^n|} = 2$, h_s maps two domain elements to each codomain element on average. We can use INV to construct a collision finder F as follows. Given a randomly chosen $s \in \{0,1\}^n$, F randomly chooses (say without replacement) $x_1, \ldots, x_{n^c} \in \{0,1\}^{n+1}$ and simulates INV to obtain $x'_i = INV(s, x_i)$, $1 \leq i \leq n^c$. It can be shown that, if c is "large enough", the probability that there exists an $1 \leq i \leq n^c$ such that $h_s(x'_i) = h_s(x_i)$ and $x'_i \neq x_i$ is non-negligible (in n).

4 Trapdoor permutations

Impagliazzo and Rudich show in [IR89] that proving secure public-key encryption schemes (see Section 7) exist assuming only that one-way functions exist is no easier than proving $P \neq NP$. On the other hand, if trapdoor permutations exist then so do secure public-key encryption schemes ([RS92]).

Informally, a bijection mapping n-bit strings to n-bit strings is a *trapdoor permutation* if it is "easy to evaluate" and "hard to invert on average", yet "easy to invert" given some additional information.

Formally, a *trapdoor permutation* $\mathcal{F}$ consists of three polynomial-time algorithms: a key generator G and two function evaluators, f and f'. G is probabilistic, whereas f and f' are both deterministic. Given 1^n and some random bits, G outputs a pair of keys (k, k'). Associated with every pair of keys (k, k') is a pair of functions $(f_k, f'_{k'})$, each mapping n-bit strings to n-bit strings; f_k and $f'_{k'}$ are both injective (and therefore surjective), and $f'_{k'} = f_k^{-1}$.

For every pair of keys (k, k') generated by running G on 1^n (together with some random bits) and every $x \in \{0,1\}^n$, $f(k,x) = f_k(x)$ and $f'(k',x) = f'_{k'}(x)$. Moreover, f_k is a one-way function in the following sense. For every probabilistic polynomial-time "inverter" INV, $p_{INV}(n)$ is negligible in n. Here $p_{INV}(n)$ is the probability that, given a key k (generated by running G on 1^n and some random bits) and $y = f_k(x) \in \{0,1\}^n$ for a random $x \in \{0,1\}^n$, INV outputs $x = f'_{k'}(y)$; $p_{INV}(n)$ is taken over the choice of $s \in \{0,1\}^n$, as well as the random bits of INV and G (that is, the choice of k).

5 Signature schemes and message authentification codes (MACs)

A *signature scheme* SIG consists of three polynomial-time algorithms: a key generator GEN, a signer $SIGN$ and a verifier VER. Although in general all three may be probabilistic, we will assume for convenience that GEN and $SIGN$ are probabilistic, whereas VER is deterministic (this is nearly always the case in practice).

GEN, $SIGN$ and VER work as follows.

- Given 1^n and some random bits, GEN outputs a pair of keys (pub, pri), where pub is the public key and pri is the private key. Although in general $|pri| \leq n^c$ for some c, we will usually assume that $|pri| = n$ to simplify the presentation.

- Given 1^n, pri, a message $m \in \{0,1\}^*$ and some random bits, $SIGN$ outputs a signature $\sigma_m \in \{0,1\}^{p(n)}$ of m, where $p(\cdot)$ is some polynomial.

- Given 1^n, pub, a message m and a supposed signature $\alpha \in \{0,1\}^{p(n)}$ or m, VER outputs either 1, indicating he thinks α is a valid signature of m, or 0, indicating he thinks it is not.

Denote the output of $SIGN$ given 1^n, pri, a message m and some random bits by $SIGN_{pri}(m)$, and the output of VER given 1^n, pub, m and a supposed signature α of

m by $VER_{pub}(m, \alpha)$. We require that VER accept all signatures output by $SIGN$, so that for all n, all key pairs (pub, pri) generated by running GEN on 1^n and some random bits, all messages m and all signatures $\sigma_m = SIGN_{pri}(m)$, $VER_{pub}(m, \sigma_m) = 1$.

Informally, SIG is *secure* if no probabilistic polynomial-time "forger" F who knows *pub* has a significant probability of coming up with a valid signature σ^* of a new message $m^* \in \{0,1\}^*$, even after being shown the signatures of polynomially many messages of his choice. Since F adaptively chooses the messages whose signatures he is shown and wins if he successfully signs *any* new message (even a "silly" one such as the empty string λ), this sort of security for signature schemes is sometimes called "security against existential forgery under adaptive chosen-message attack".

Formally, F is equipped with a "signature oracle" $\mathcal{S}$; given a message $m \in \{0,1\}^*$, $\mathcal{S}$ outputs a signature $\sigma_m = SIGN_{pri}(m)$ of m. SIG is *secure* if, for every probabilistic polynomial-time forger $F^{\mathcal{S}}$, $p_F(n)$ is negligible in n. Here $p_F(n)$ is the probability that, given 1^n and a public key *pub* (generated by running GEN on 1^n and some random bits), $F^{\mathcal{S}}$ outputs a pair (m^*, σ^*) such that $\mathcal{S}$ has not been queried on m^* and $VER_{pub}(m^*, \sigma^*) = 1$; $p_F(n)$ is taken over the random bits of GEN (that is, the choice of (pub, pri)), $F^{\mathcal{S}}_{pub}$ and VER_{pub}, as well as the randomness of $\mathcal{S}$ (that is, the random bits of $SIGN$).

Building on the results of [GMR88], [BM88] and [NY89], Rompel showed in [Rom90] that if one-way functions exist, then so do secure signature schemes. It's not hard to show that the converse also holds, namely that if secure signature schemes exist, then so do one-way functions. Notice that if SIG is a secure signature scheme, then the function f_{GEN} mapping the random bits r of GEN to the public key *pub* is one-way. Since GEN runs in polynomial time, f_{GEN} is efficiently evaluable. If f_{GEN} were easy to invert on average, then a forger F who is given *pub* could compute *pri* with high probability, thereby completely breaking the security of SIG. Thus f_{GEN} is a one-way function. Notice that this means that if one-way functions do not exist, then neither do secure signatures schemes.

Message authentification codes or MACs, as they are commonly referred to, are essentially private-key signature schemes. This time there is only one (private) key, k, which is chosen randomly and given to both the signer and the verifier. As with signature schemes, the standard notion of security for MACs is "security against existential forgery under adaptive chosen-message attack". Although the forger still has access to a signature oracle $\mathcal{S}$, this time he is obviously not given the private key k (which is built into $\mathcal{S}$); the forger's success probability is taken over his random bits and the choice of k.

6 Identification schemes

An *identification scheme* ID consists of three probabilistic polynomial-time algorithms: a key generator G, a prover P and a verifier V. P and V are "linked interactive machines", meaning that they can "interact" by sending messages back and forth between each other. Informally, P's goal is to convince V that he knows some secret, for example the private key generated by G. Although identification schemes are interesting in their own right, our interest in them stems from the fact that they are a source of signature schemes secure in the ROM ([AABN02]).

Each message exchanged between P and V is called a *round*. If V's messages consist solely of random bits, then ID is said to be *public-coin*. Three-round, public-coin identification schemes are called *canonical*. This thesis only deals with canonical identification schemes, so let ID be canonical. Observe that the prover always goes last, because otherwise he wouldn't be able to respond to the verifier's last challenge. Since the prover and the verifier alternate rounds, this means that P first sends a message to V, then V challenges P, and finally P responds to V's challenge.

Formally, ID works as follows. First, G is run on 1^n and some random bits to obtain a pair of keys (PK, SK), where PK is the public key and SK is the private key. Let P_{SK}

denote the behaviour of P when given 1^n, SK and some random bits, and V_{PK} denote the behaviour of V when given 1^n and PK. P_{SK} first sends a *commitment* CMT to V_{PK}, to which V_{PK} replies with a *challenge* CH consisting of the entire contents of his random tape. P_{SK} then sends a *response* RSP to V_{PK}, at which point V_{PK} makes a deterministic decision to either accept or reject (see Figure 2.1). For reasons which will become clear later, it is convenient to assume that all of P_{SK}'s commitments are of length $\ell(n)$, where ℓ is some polytime-computable function of the security parameter n. This is always the case in practice.

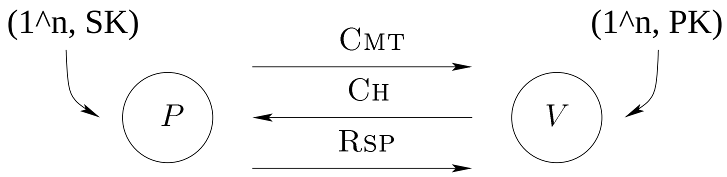

Figure 2.1: The interaction between P and V

Since the behaviour of V_{PK} is completely determined once his random tape CH is fixed, we may think of V_{PK} as a deterministic function accepting or rejecting "transcripts" of the form (m_1, CH, m_2), where m_1 and m_2 are the first and second messages received by V_{PK}, respectively; V_{PK} may interact with an adversary who is not P_{SK}, so these need not equal CMT and RSP. We require that P_{SK} always convince V_{PK} to accept, so that $V_{PK}(\text{CMT}, \text{CH}, \text{RSP}) = 1$ for all CMT and RSP produced by P_{SK}.

We are interested in two notions of security for identification schemes: *passive security* and *active security*.

Informally, ID is *passively secure* if no probabilistic polynomial-time "impersonator" I who knows PK (but not SK) has a significant probability of convincing V_{PK} to accept when interacting with him in the role of P_{SK}, even after seeing polynomially many transcripts of conversations between P_{SK} and V_{PK}. This weak type of security for identification schemes is called "passive" because I_{PK} passively monitors the conversation between P_{SK} and V_{PK} without interfering with it.

Formally, I is equipped with a "transcript oracle" $\mathcal{T}$. Every time $\mathcal{T}$ is queried, it generates a transcript $(\textsc{Cmt}, \textsc{Ch}, \textsc{Rsp})$ by running P_{SK} and V_{PK} on some random bits. $I_{PK}^{\mathcal{T}}$ is given 1^n and a public key PK (generated by running G on 1^n and some random bits), together with some random bits. $I_{PK}^{\mathcal{T}}$ first obtains polynomially many transcripts by repeatedly querying $\mathcal{T}$. Next, $I_{PK}^{\mathcal{T}}$ sends a commitment $\textsc{Cmt}'$ to V_{PK}, receiving a challenge $\textsc{Ch}$ in reply. $I_{PK}^{\mathcal{T}}$ then responds to the challenge by sending $\textsc{Rsp}'$ to V_{PK}. ID is *passively secure* if, for every passive probabilistic polynomial-time impersonator $I_{PK}^{\mathcal{T}}$, the probability $p_I(n)$ that $V_{PK}(\textsc{Cmt}', \textsc{Ch}, \textsc{Rsp}') = 1$ is negligible in n; $p_I(n)$ is taken over the random bits of G (that is, the choice of (PK, SK)), $I_{PK}^{\mathcal{T}}$ and V_{PK} (that is, the choice of $\textsc{Ch}$), as well as the randomness of $\mathcal{T}$ (that is, the random bits of P_{SK} and V_{PK}).

Informally, ID is *actively secure*, or simply *secure*, if no probabilistic polynomial-time "impersonator" I who knows PK (but not SK) has a significant probability of convincing V_{PK} to accept when interacting with him in the role of P_{SK}, even after arbitrarily interacting with P_{SK} in the role of V_{PK} polynomially many times. This strong type of security for identification schemes is called "active" because I_{PK} actively interacts with P_{SK} rather than merely monitoring P_{SK}'s conversation with V_{PK}.

Formally, we think of I_{PK}, who is given 1^n and a public key PK (generated by running G on 1^n and some random bits), together with some random bits, as operating in two "phases". In the first phase, I_{PK} interacts with P_{SK} (in the role of V_{PK}) by sending him polynomially many adaptively chosen challenges; note that I_{PK} is not constrained to choose his challenges randomly. In the second phase, I_{PK} interacts with V_{PK} (in the role of P_{SK}) as follows. I_{PK} first sends a commitment $\textsc{Cmt}''$ to V_{PK}, receiving a random challenge $\textsc{Ch}$ in reply. I_{PK} then responds to the challenge by sending $\textsc{Rsp}''$ to V_{PK}. ID is *secure* if, for every active probabilistic polynomial-time impersonator I_{PK}, the probability $p_I(n)$ that $V_{PK}(\textsc{Cmt}'', \textsc{Ch}, \textsc{Rsp}'') = 1$ is negligible in n; $p_I(n)$ is taken over the random bits of G (that is, the choice of (PK, SK)), I_{PK}, V_{PK} (that is, the choice of $\textsc{Ch}$) and P_{SK}.

Note that if V_{PK}'s challenge $\textsc{Ch}$ is too short, $|\textsc{Ch}| = \log_2(n)$ say, then the size of the challenge space is only $2^{|\textsc{Ch}|} = n$. An impersonator I_{PK} in possession of even a single valid transcript $(\textsc{Cmt}, \textsc{Ch}, \textsc{Rsp})$, obtained through either interacting with P_{SK} or querying $\mathcal{T}$, can in this case break the security of ID as follows. I_{PK} sends $\textsc{Cmt}$ to V_{PK}, receives a challenge $\textsc{Ch}'$ from V_{PK} and sends $\textsc{Rsp}$ to V_{PK} in response. Since V_{PK} accepts whenever $\textsc{Ch}' = \textsc{Ch}$, which happens with probability $\frac{1}{n}$ (and possibly even if $\textsc{Ch}' \neq \textsc{Ch}$), I_{PK}'s success probability is non-negligible. In order for ID to hope to satisfy either of the above two definitions of security, the challenge space must therefore be of size super-polynomial in n, meaning that $|\textsc{Ch}| = \omega(\log n)$.

Observe that passive security is strictly weaker than active security, since every actively secure ID is also passively secure, but not vice versa. Active security implies passive security because, for every (passive) impersonator $I^{\mathcal{T}}_{PK}$ who breaks the passive security of ID, there is a corresponding (active) impersonator I_{PK} who breaks the active security of ID: I_{PK} simply simulates $I^{\mathcal{T}}_{PK}$, taking care to accumulate enough valid transcripts during the first phase (by choosing the challenges he sends to P_{SK} randomly) to answer all of $I^{\mathcal{T}}_{PK}$'s $\mathcal{T}$ queries; I_{PK}'s success probability is identical to that of $I^{\mathcal{T}}_{PK}$.

To see that passive security does *not* imply active security, consider the following (admittedly rather contrived) modification ID' of an arbitrary passively secure identification scheme ID (such identification schemes exist if one-way functions do, as we'll see below); we may assume without loss of generality that $|\textsc{Ch}| = n$. ID' is identical to ID, except that whenever the new prover P'_{SK} receives the challenge $\bar{0} = 0^n$, he responds by revealing the private key SK. ID' remains passively secure, since a passive impersonator $I^{\mathcal{T}}_{PK}$ whose running time is bounded above by some polynomial $p(\cdot)$ in the security parameter n will see a transcript containing SK with probability at most $\frac{p(n)}{2^n}$, which is negligible. However, it is completely trivial for an active impersonator I_{PK} to break the (active) security of ID': I_{PK} sends $\bar{0}$ to P'_{SK} to obtain the secret key SK in the first phase, then simulates P'_{SK} in order to correctly respond to V'_{PK}'s challenge in the second phase.

Finally, observe that secure identification schemes exist if and only if one-way functions do. To see that if secure identification schemes exist then so do one-way functions, let $ID = (G, P, V)$ be an arbitrary secure canonical identification scheme. The function f_G mapping the random bits r of G to the public key PK must be one-way, since otherwise an impersonator could completely break the security of ID (see Section 5). To see that secure canonical identification schemes exist if one-way functions do, we need only show how to convert an arbitrary secure signature scheme into a secure canonical identification scheme (recall that secure signature schemes exist if one-way functions do).

We can easily obtain a canonical identification scheme $ID = (G, P, V)$ from any signature scheme $SIG = (GEN, SIGN, VER)$; V simply challenges P to sign a random n-bit message $\textsc{Ch}$ and accepts only if $\textsc{Rsp}$ is a valid signature of $\textsc{Ch}$. Specifically, G is the same as GEN (so that $(pub, pri) = (PK, SK)$), $\textsc{Cmt} = \lambda$, $\textsc{Rsp} = SIGN_{SK}(\textsc{Ch})$ and $V_{PK}(\lambda, \textsc{Ch}, \textsc{Rsp}) = VER_{PK}(\textsc{Ch}, \textsc{Rsp})$.

It's not too hard to see that if SIG is secure (as a signature scheme) then ID is secure (as an identification scheme). An active impersonator I which successfully breaks the security of ID first gets to see the signatures of polynomially many messages of his choice and then successfully signs a random message $\textsc{Ch}$, whose signature he almost certainly hasn't already seen (because the challenge space is of super-polynomial size); a polynomial-time forger $F^{\mathcal{S}}$ with access to a signature oracle $\mathcal{S}$ can easily simulate I, thereby breaking the security of SIG.

7 Public-key encryption schemes

A *public-key encryption scheme* PKE consists of three polynomial-time algorithms: a key generator GEN, an encryptor ENC and a decryptor DEC. GEN and ENC are probabilistic (our definition of security will crucially depend on the fact that ENC is probabilistic), whereas DEC is deterministic.

Informally, the setup is that a person A wants to securely communicate with some stranger B he knows nothing about, except for his name and address. To this end, A generates a pair of keys (pub, pri) using GEN, sends the public key pub to B and keeps the private key pri for himself. To communicate a message m to A, B obtains an encryption e_m of m using ENC and sends e_m to A; A then decrypts e_m using DEC.

For reasons of modularity and efficiency, public-key encryption schemes are in practice almost always used solely to securely exchange a "short" private key k, whose length we'll assume to be equal to the security parameter n for convenience. Once both A and B are in possession of k, they can securely communicate using highly efficient "private-key encryption". Thus, unlike in the case of signature schemes, where we insisted that $SIGN$ be able to sign messages of arbitrary length, here we will only require ENC to be able to encrypt n-bit messages.

Formally, GEN, ENC and DEC work as follows.

- Given 1^n and some random bits, GEN outputs a pair of keys (pub, pri), where pub is the public key and pri is the private key.

- Given 1^n, pub, a message $m \in \{0,1\}^n$ and some random bits, ENC outputs an encryption $e_m \in \{0,1\}^{p(n)}$ of m, where $p(\cdot)$ is some polynomial.

- Given 1^n, pri and a supposed encryption α, DEC either outputs a message $m \in \{0,1\}^n$ or a special symbol $\perp$ indicating a failure to decrypt.

Denote the output of ENC given 1^n, pub, $m \in \{0,1\}^n$ and some random bits by $ENC_{pub}(m)$, and the output of DEC given 1^n, pri and $\alpha \in \{0,1\}^{p(n)}$ by $DEC_{pri}(\alpha)$. We require that DEC correctly decrypt all encryptions produced by ENC, so that for all n, all key pairs (pub, pri) generated by running GEN on 1^n and some random bits, all messages $m \in \{0,1\}^n$ and all encryptions $e_m = ENC_{pub}(m)$, $DEC_{pri}(e_m) = m$.

We are interested in two notions of security for public-key encryption schemes: *semantic security* and *chosen-ciphertext security*.

Informally, PKE is *semantically secure* ([GM84]) if no probabilistic polynomial-time "eavesdropper" E who knows pub (but not pri) and passively monitors the channel between A and B can "learn" even a single bit of information about a message m through seeing its encryption e_m.

Formally, E is given 1^n and a public key pub (generated by running GEN on 1^n and some random bits) and chooses two distinct n-bit messages, m_0 and m_1. A bit b is then chosen randomly (but not shown to E), and E is given an encryption $e_b = ENC_{pub}(m_b)$ of m_b. E next computes for a while, finally outputting a bit b'. Let $p_E(n)$ be the probability that $b' = b$, meaning that E correctly determined b; $p_E(n)$ is taken over the random bits of E and GEN (that is, the choice of (pub, pri)), as well as the choice of b. PKE is *semantically secure* if, for every probabilistic polynomial-time eavesdropper E, $|\frac{1}{2} - p_E(n)|$ is negligible in n (in other words, $p_E(n)$ doesn't significantly differ from $\frac{1}{2}$, the probability of randomly guessing b).

Note that PKE cannot be semantically secure if ENC is deterministic. In order to break the semantic security of PKE, an eavesdropper E (who knows pub) simply computes $\eta_0 = ENC_{pub}(\bar{0})$ and $\eta_1 = ENC_{pub}(\bar{1})$, where $\bar{0} = 0^n$ and $\bar{1} = 0^{n-1}1$, then sets $m_0 = \bar{0}$ and $m_1 = \bar{1}$. Once E receives e_b, he outputs 0 if $e_b = \eta_0$ and 1 if $e_b = \eta_1$ (these are the only two possibilities because ENC is deterministic). Since E always outputs b correctly (so that $b' = b$ with probability 1), $|\frac{1}{2} - p_E(n)| = \frac{1}{2}$, which is certainly non-negligible.

Informally, PKE is *secure against (adaptive) chosen-ciphertext attack* or *CCA2-secure* ([RS92]) if no probabilistic polynomial-time adversary ADV who knows pub (but not pri) and has complete control over the channel between A and B can "learn" even a single bit of information about a message m through seeing its encryption e_m. What does it mean for ADV to have "complete control" over the channel between A and B? Intuitively, ADV intercepts all encryptions or "ciphertexts" sent by A to B and sends B whatever he likes instead.

Formally, ADV is given 1^n and a public key pub (generated by running GEN on 1^n and some random bits) and equipped with a "decryption oracle" $\mathcal{D}$, which outputs $DEC_{pri}(\alpha)$ when queried on a ciphertext $\alpha \in \{0,1\}^{p(n)}$. $\mathcal{D}$ is meant to capture the intuition that ADV can effectively force B to decrypt any ciphertext of his choosing (of course the answer may well be $\perp$; we think of "ciphertexts" that decrypt to $\perp$ as being malformed).

$ADV^{\mathcal{D}}$ queries $\mathcal{D}$ on α_0 and receives $DEC_{pri}(\alpha_0)$, queries $\mathcal{D}$ on α_1 and receives $DEC_{pri}(\alpha_1)$, and so on. Since $ADV^{\mathcal{D}}$ may in general choose his queries based on $\mathcal{D}$'s previous answers, this is an adaptive attack. Eventually, $ADV^{\mathcal{D}}$ chooses two distinct n-bit messages, m_0 and m_1. A bit b is then chosen randomly (but not shown to $ADV^{\mathcal{D}}$), and $ADV^{\mathcal{D}}$ is given an encryption $e_b = ENC_{pub}(m_b)$ of m_b.

$ADV^{\mathcal{D}}$ now gets to query $\mathcal{D}$ on some additional ciphertexts, whose choice may in general depend on e_b. Naturally, we don't allow $ADV^{\mathcal{D}}$ to query $\mathcal{D}$ on e_b itself, since $DEC_{pri}(e_b)$ uniquely determines b (because $m_0 \neq m_1$). Alternatively, once $ADV^{\mathcal{D}}$ receives e_b we could forbid him from querying $\mathcal{D}$ altogether; security against this type of "lunchtime attack" is called *CCA1 security* ([NY90]). However, that would arguably be too restrictive, since in practice CCA2 security is almost always broken by querying $\mathcal{D}$ on ciphertexts "related to" (though not the same as) e_b.

$ADV^{\mathcal{D}}$ next computes for a while, finally outputting a bit b'. Let $p_{ADV}(n)$ be the probability that $b' = b$, meaning that $ADV^{\mathcal{D}}$ correctly determined b; $p_{ADV}(n)$ is taken over the random bits of $ADV^{\mathcal{D}}$ and GEN (that is, the choice of (pub, pri)), as well as the choice of b (the decryption oracle $\mathcal{D}$ is deterministic). PKE is *CCA2 secure* if, for every probabilistic polynomial-time adversary $ADV^{\mathcal{D}}$, $|\frac{1}{2} - p_{ADV}(n)|$ is negligible in n (in other words, $p_{ADV}(n)$ doesn't significantly differ from $\frac{1}{2}$, the probability of randomly guessing b).

Observe that semantic security is strictly weaker than CCA2 security, since every CCA2-secure PKE is also semantically secure, but not vice versa. CCA2 security implies semantic security because, for every eavesdropper E who breaks the semantic security of PKE, there is a corresponding adversary $ADV^{\mathcal{D}}$ who breaks the CCA2 security of PKE: $ADV^{\mathcal{D}}$ simply simulates E, ignoring the decryption oracle $\mathcal{D}$; $ADV^{\mathcal{D}}$'s success probability is identical to that of E. This implies that PKE cannot be CCA2-secure if ENC is deterministic — we already know that such a PKE is not semantically secure, and we just showed every CCA2-secure public-key encryption scheme is.

To see that semantic security does *not* imply CCA2 security, consider the following (admittedly rather contrived) modification PKE' of an arbitrary semantically secure public-key encryption scheme PKE. PKE' is identical to PKE, except that the new encryptor ENC' appends an additional bit, say 0 for concreteness, to every encryption; this bit is ignored by the new decryptor DEC'. PKE' remains semantically secure, since, for every eavesdropper E' who breaks the semantic security of PKE', there is a corresponding eavesdropper E who breaks the semantic security of PKE: E simulates E' to obtain a pair of messages (m_0, m_1), receives an encryption e_b and gives e_b0 to E', accepting if and only if E' accepts; E's success probability is identical to that of E'.

However, it is completely trivial for an adversary $ADV^{\mathcal{D}}$ to break the CCA2 security of PKE': $ADV^{\mathcal{D}}$ sets $m_0 = \bar{0}$ and $m_1 = \bar{1}$, receives $e_b = ENC'_{pub}(m_b)$, and queries $\mathcal{D}$ on e_b1 to obtain a decryption m'. He then outputs 0 if $m' = \bar{0}$ and 1 if $m' = \bar{1}$ (these are the only two possibilities, since DEC' ignores the trailing bit). Since $ADV^{\mathcal{D}}$ always outputs b correctly (so that $b' = b$ with probability 1), $|1 - p_{ADV}(n)| = \frac{1}{2}$, which is certainly non-negligible. Although our highly artificial modification of PKE may seem like a cheat, in practice public-key encryption schemes fail to be CCA2-secure for the same reason as PKE'. Namely, they are "malleable", which informally means that an adversary $ADV^{\mathcal{D}}$ can "malleate" (that is, modify) e_b into some related encryption e'_b such that b can be computed from $\mathcal{D}(e'_b)$ ($ADV^{\mathcal{D}}$ is allowed to query $\mathcal{D}$ on e'_b since $e'_b \neq e_b$).

8 The Random Oracle Model (ROM)

The Random Oracle Model, or ROM for short, is a setting where all parties have access to a "random oracle" $\mathcal{R}$. The ROM was formally introduced in the context of cryptography in [BR93].

One way to think of $\mathcal{R}$ is as a randomly chosen function mapping $\{0,1\}^*$ to $\{0,1\}^{n\dagger}$. However, the set of all such functions is (countably) infinite, and we prefer not to talk about sampling such sets. Instead, we view $\mathcal{R}$ as choosing his answers "on-line". When queried on $q \in \{0,1\}^*$, $\mathcal{R}$ first checks whether q is a "new query", meaning that he hasn't been queried on q before. If so, he randomly chooses a response $ans \in \{0,1\}^n$ to q, writes ans down somewhere and then outputs it. Otherwise (namely in the case that $\mathcal{R}$ has been queried on q already), he looks up and outputs his previous response to q; this ensures that identical queries receive an identical response (which is the case when $\mathcal{R}$ is viewed as a function).

We next show that, in some appropriate sense at least, random oracles are one-way (see Section 2 for a definition of one-wayness). Let $p_{INV}(n)$ denote the probability that a probabilistic polynomial-time inverter $INV^{\mathcal{R}}$ who is given $y = \mathcal{R}(x) \in \{0,1\}^n$ for a randomly chosen $x \in \{0,1\}^n$ outputs an $x' \in \{0,1\}^n$ such that $\mathcal{R}(x') = y$; $p_{INV}(n)$ is taken over the choice of x, the random bits of $INV^{\mathcal{R}}$ and the randomness of $\mathcal{R}$. Observe that y yields no information about x, since it is distributed uniformly over $\{0,1\}^n$ no matter what x is. Denote the strings $INV^{\mathcal{R}}$ queries $\mathcal{R}$ on by $x_1, \ldots, x_{q_{\mathcal{R}}}$, and set $y_i = \mathcal{R}(x_i)$ for $1 \le i \le q_{\mathcal{R}}$; notice that $q_{\mathcal{R}} \le n^c$ for some c, because $INV^{\mathcal{R}}$ runs in (strict) polynomial time. $INV^{\mathcal{R}}$ wins if there is an $1 \le i \le q_{\mathcal{R}}$ such that either $x_i = x$ or $x_i \neq x$ but $y_i = y$ anyway. Applying the union bound, we see that this happens with probability at most $\frac{2q_{\mathcal{R}}}{2^n} \le \frac{2n^c}{2^n}$, which is negligible.

†In general, $\mathcal{R}$ maps $\{0,1\}^*$ to $\{0,1\}^{\ell(n)}$, where $\ell(n) \le n^c$ for some c. However, we will usually assume that $\ell(n) \equiv n$ to simplify the presentation.

Chapter 3

On the security of Fiat-Shamir signature schemes in the ROM

1 Fiat-Shamir signature schemes: an overview

In their seminal 1986 paper ([FS87]), Amos Fiat and Adi Shamir proposed a new, highly efficient signature scheme based on a certain canonical identification scheme closely related to the protocols presented in [GMR85] and [FMR96] (for definitions of signature schemes and canonical identification schemes, see Sections 5 and 6 of Chapter 2). Such signature schemes are now called "Fiat-Shamir signature schemes", whereas Fiat and Shamir's approach itself is referred to as the "Fiat-Shamir paradigm". Essentially, their idea was as follows. In order to sign a message m, simply simulate the prover, replacing the verifier's random challenge with $h(m)$, where h is some "cryptographic hash function" (actually, this isn't quite right, as we'll see below). The resulting transcript then serves as a signature of m.

Fiat and Shamir showed that the signature scheme in question is secure if h is "truly random", provided that taking square roots modulo $N = pq$, where p and q are unknown "large" primes, is hard (a standard hardness assumption). In modern terminology, Fiat and Shamir effectively showed that the signature scheme is secure in the Random Oracle Model, or ROM (see Chapter 2, Section 8 for a discussion of the ROM), under a standard hardness assumption. Although this may strike one as a rather weak security guarantee, no practical signature schemes provably secure under standard hardness assumptions were known at the time. Today, a number of highly efficient signature schemes provably secure under such "non-standard-yet-plausible" hardness assumptions as the "strong RSA assumption" and the "strong Computational Diffie-Hellman assumption" are available ([GHR99], [CS99], [Fis03], [BB04]).

Various other Fiat-Shamir signature schemes which are provably secure in the ROM under standard hardness assumptions have been described over the years ([MS90], [Oka93], [Sho96], [GaJ03]), but until fairly recently it was not known whether *every* (actively) secure canonical identification scheme yields a Fiat-Shamir signature scheme secure in the ROM. While Abdalla et al. showed in [AABN02] that secure "non-trivial" canonical identification schemes yield Fiat-Shamir signature schemes secure in the ROM (informally, a canonical identification scheme is "non-trivial" if the prover's commitment distribution has "high entropy"), they left open the question of whether secure "trivial" canonical identification schemes do. In Section 5, we prove that every secure canonical identification scheme, trivial or not, does indeed yield a Fiat-Shamir signature scheme secure in the ROM.

However, as we will see in Chapter 5, security in the ROM is no guarantee of real-world security. In [GTK03], Goldwasser and Tauman show that there exist Fiat-Shamir signature schemes which, although secure in the ROM, are "uninstantiable" (see Chapter 5, Section 4). Such schemes are not secure in the "real world", no matter what hash function ensemble is used to "instantiate" the random oracle.

2 The Fiat-Shamir transform

Let $ID = (G, P, V)$ be a canonical identification scheme and h be a "cryptographic hash function". The function mapping ID and h to the corresponding Fiat-Shamir signature scheme $SIG_h(ID)$ is sometimes called the "Fiat-Shamir transform". Since this thesis is primarily concerned with security in the ROM, we will only present the transform's ROM version, which maps ID to $SIG(ID) = (G, SIGN^{\mathcal{R}}, VER^{\mathcal{R}})$.

Given 1^n, a private key SK (generated by running G on 1^n together with some random bits), a message $m \in \{0,1\}^*$ and some random bits, the signer $SIGN^{\mathcal{R}}$ proceeds as follows. He first simulates P_{SK} to obtain a commitment $\textsc{Cmt}$ and computes a challenge $\textsc{Ch}_m = \mathcal{R}(\textsc{Cmt}, m)$ by querying $\mathcal{R}$; note that the challenge depends on the message to be signed. $SIGN^{\mathcal{R}}$ then simulates P_{SK} on $\textsc{Ch}_m$ to obtain a response $\textsc{Rsp}$ and outputs $\sigma_m = (\textsc{Cmt}, \textsc{Rsp})$ as the signature of m. (Recall that P_{SK} denotes the behaviour of P when given 1^n, SK and some random bits r. Specifically, P computes a commitment $\textsc{Cmt}$ as a function of 1^n, SK and r, receives a challenge $\textsc{Ch}$, and then computes a response $\textsc{Rsp}$ as a function of 1^n, SK, r and $\textsc{Ch}$).

Given 1^n, a public key PK (generated by running G on 1^n together with some random bits), a message $m \in \{0,1\}^*$ and a supposed signature (α, γ) of m, the verifier $VER^{\mathcal{R}}$ simply computes $\beta = \mathcal{R}(\alpha, m)$ by querying $\mathcal{R}$ and outputs $V_{PK}(\alpha, \beta, \gamma)$ (Recall that V_{PK} is a deterministic function of (α, β, γ)).

Our goal is to show that if ID is secure then $SIG(ID)$ is secure in the ROM. By security in this setting we mean the ordinary security for signature schemes (that is, security against existential forgery under adaptive chosen-message attack), except that the forger F now has access to the random oracle $\mathcal{R}$ in addition to the signature oracle $\mathcal{S}$, and his success probability is also taken over the randomness of $\mathcal{R}$.

3 An earlier result

Abdalla, An, Bellare and Namprempre present several results concerning the Fiat-Shamir transform in [AABN02], including a randomized version of the transform and applications to "forward-secure signature schemes". However, we are only interested in the following result of theirs: for every passively secure "non-trivial" canonical identification scheme ID, the corresponding Fiat-Shamir signature scheme $SIG(ID)$ is secure in the ROM. Since active security implies passive security for identification schemes, this means that every secure "non-trivial" canonical identification scheme yields a Fiat-Shamir signature scheme secure in the ROM.

Informally, ID is "non-trivial" if the prover's commitment distribution has "high entropy". Formally, let $\mathcal{P}_{SK} = \{p_i\}_{i=0}^{k}$ denote P_{SK}'s commitment distribution and define the *min-entropy* of $\mathcal{P}_{SK}$ by $H_{\min}(\mathcal{P}_{SK}) = -\log_2(p_{\max})$, where $p_{\max} = \max\{p_i\}_{i=0}^{k}$ is the largest probability mass in $\mathcal{P}_{SK}$. ID is *non-trivial* if $\min\{H_{\min}(\mathcal{P}_{SK}) : SK \leftarrow G(1^n)\} = \omega(\log n)$, meaning that the minimum min-entropy of $\mathcal{P}_{SK}$, taken over all private keys SK (generated by running G on 1^n and some random bits), is super-logarithmic in the security parameter n. It can be shown that in this case the probability of seeing the same commitment more than once in polynomially many trials is negligible, so that, for all practical purposes, P_{SK}'s commitments don't repeat. Canonical identification schemes which are not non-trivial are said to be *trivial*.

Let ID be a non-trivial canonical identification scheme. Suppose that $F^{\mathcal{R},\mathcal{S}}$ is a polynomial-time forger who breaks the security of $SIG(ID)$ in the ROM, and denote his (non-negligible) success probability by $p_F(n)$. $F_{PK}^{\mathcal{R},\mathcal{S}}$ is given 1^n, PK and some random bits, and his goal is to output a new message m^* (i.e. one he hasn't queried $\mathcal{S}$ on) together with a valid signature $\sigma^* = (\text{CMT}^*, \text{RSP}^*)$ of m^*.

We may assume, without loss of generality, that $F_{PK}^{\mathcal{R},\mathcal{S}}$ doesn't query $\mathcal{R}$ on any string more than once, since that would yield no new information (because $\mathcal{R}$'s responses would all be identical). We may additionally assume, again wlog, that $F_{PK}^{\mathcal{R},\mathcal{S}}$ doesn't query $\mathcal{R}$ on strings whose length is less than $\ell(n)$; recall that all of P_{SK}'s commitments are of length $\ell(n)$. $\mathcal{R}$ queries involving such "short strings" can safely be answered randomly, since there is no interplay between them and $\mathcal{S}$ queries (more on this interplay later). This assumption ensures that every $s \in \{0,1\}^*$ $F_{PK}^{\mathcal{R},\mathcal{S}}$ queries $\mathcal{R}$ on can be parsed as $(\textsc{Cmt}, m)$, where $\textsc{Cmt} \in \{0,1\}^{\ell(n)}$ and $m \in \{0,1\}^*$. Finally, it will be convenient for us to assume that $F_{PK}^{\mathcal{R},\mathcal{S}}$ queries $\mathcal{R}$ on $(\textsc{Cmt}^*, m^*)$ at some point during his execution; following the terminology of [AABN02], we refer to this special $\mathcal{R}$ query as the "crucial query". There is no loss of generality in assuming that the forger makes the crucial query, since every $F^{\mathcal{R},\mathcal{S}}$ who doesn't can easily be converted into a corresponding forger $\hat{F}^{\mathcal{R},\mathcal{S}}$ who does: $\hat{F}_{PK}^{\mathcal{R},\mathcal{S}}$ obtains m^* and $\sigma^* = (\textsc{Cmt}^*, \textsc{Rsp}^*)$ by simulating $F_{PK}^{\mathcal{R},\mathcal{S}}$, queries $\mathcal{R}$ on $(\textsc{Cmt}^*, m^*)$ and then outputs (m^*, σ^*). Since the additional $\mathcal{R}$ query doesn't affect the choice of m^* and σ^*, $\hat{F}^{\mathcal{R},\mathcal{S}}$'s success probability is identical to that of $F^{\mathcal{R},\mathcal{S}}$.

We now describe a polynomial-time impersonator $I^{\mathcal{T}}$ who, given 1^n, PK and some random bits, breaks the passive security of ID (in the real world) by simulating $F_{PK}^{\mathcal{R},\mathcal{S}}$. Let $q_{\mathcal{R}}(n)$ denote the number of times $F_{PK}^{\mathcal{R},\mathcal{S}}$ queries $\mathcal{R}$. Since $F_{PK}^{\mathcal{R},\mathcal{S}}$ runs in (strict) polynomial time, $q_{\mathcal{R}}(n) \leq n^c$ for some c (in the worst case, $F_{PK}^{\mathcal{R},\mathcal{S}}$ does nothing but query $\mathcal{R}$, each query taking a single step).

$I_{PK}^{\mathcal{T}}$ begins by randomly choosing an index $i \in \{1, \ldots, q_{\mathcal{R}}(n)\}$; as we'll see later, i is not revealed to $F_{PK}^{\mathcal{R},\mathcal{S}}$ in the course of the simulation. Since we've assumed both that $F_{PK}^{\mathcal{R},\mathcal{S}}$ makes the crucial query and that he never queries $\mathcal{R}$ on the same string more than once, i is the index of the crucial query with probability $\frac{1}{q_{\mathcal{R}}(n)} \geq \frac{1}{n^c}$. A technical but important point is that the distribution of $F^{\mathcal{R},\mathcal{S}}$'s views (namely what he "sees" during the simulation, including his random bits and the answers to his oracle queries) is independent of the choice of i, so that he gets no information about i. If that were not the case, $F_{PK}^{\mathcal{R},\mathcal{S}}$ could exploit his knowledge of i to ensure that $I_{PK}^{\mathcal{T}}$ never guesses the index of the crucial query correctly.

During the simulation, $I_{PK}^{\mathcal{T}}$ responds to all of $F_{PK}^{\mathcal{R},\mathcal{S}}$'s random oracle queries but the i^{th}

with a randomly chosen n-bit string (since $F_{PK}^{\mathcal{R},\mathcal{S}}$ never queries $\mathcal{R}$ on the same string more than once, there is no risk of giving inconsistent answers). The i^{th} query is handled as follows. Suppose that, for his i^{th} random oracle query, $F_{PK}^{\mathcal{R},\mathcal{S}}$ queries $\mathcal{R}$ on some $s \in \{0,1\}^*$ (recall that $|s| \geq \ell(n)$, since $F_{PK}^{\mathcal{R},\mathcal{S}}$ doesn't query $\mathcal{R}$ on "short strings"). $I_{PK}^{\mathcal{T}}$ parses s as $(\textsc{Cmt}^*, m^*)$, sends $\textsc{Cmt}^* \in \{0,1\}^{\ell(n)}$ to V_{PK}, and receives a challenge $\textsc{Ch}^* \in \{0,1\}^n$ in reply. He then gives $\textsc{Ch}^*$ to $F_{PK}^{\mathcal{R},\mathcal{S}}$ as the answer to $\mathcal{R}(s)$ and continues his simulation. This ensures that $\textsc{Rsp}^*$ is a correct answer to $\textsc{Ch}^*$, so that $(\textsc{Cmt}^*, \textsc{Ch}^*, \textsc{Rsp}^*)$ is a valid transcript.

Whenever $F_{PK}^{\mathcal{R},\mathcal{S}}$ asks to see a signature of a message m, $I_{PK}^{\mathcal{T}}$ queries $\mathcal{T}$ to obtain a valid transcript $(\textsc{Cmt}, \textsc{Ch}, \textsc{Rsp})$ and gives $(\textsc{Cmt}, \textsc{Rsp})$ to $F_{PK}^{\mathcal{R},\mathcal{S}}$. To ensure that future $\mathcal{R}$ queries are answered consistently, $I^{\mathcal{T}}$ then sets $\mathcal{R}(\textsc{Cmt}, m)$ to $\textsc{Ch}$. Observe that if $(\textsc{Cmt}, \textsc{Rsp})$ is to be a legitimate signature of m, we must have $\mathcal{R}(\textsc{Cmt}, m) = \textsc{Ch}$, so that every $\mathcal{S}$ query effectively involves an implicit $\mathcal{R}$ query. But what if $\mathcal{R}$ has been queried on $(\textsc{Cmt}, m)$ already? Unless we are very lucky and $\textsc{Ch}$ matches the value previously assigned to $\mathcal{R}(\textsc{Cmt}, m)$ (which happens with probability $\frac{1}{2^n}$, since $\textsc{Ch} \in \{0,1\}^n$ is chosen randomly), this prevents $\mathcal{R}$ from being well-defined. We may thus view a new commitment $\textsc{Cmt}$ as being added to the (notional) set of "forbidden commitments" every time $\mathcal{R}$ is queried on $s \in \{0,1\}^*$ — simply parse s as $(\textsc{Cmt}, m)$. Notice that the size of this "forbidden commitment set" is polynomial in the security parameter n, because $q_{\mathcal{R}}(n) \leq n^c$. Since ID is non-trivial (which informally means that the number of commitments is super-polynomial in n), the probability that a randomly chosen commitment belongs to the "forbidden commitment set" is therefore negligible in n, so this event can be safely ignored for the purposes of our analysis.

Eventually, $F_{PK}^{\mathcal{R},\mathcal{S}}$ outputs a message m^* together with a purportedly valid signature $\sigma^* = (\textsc{Cmt}^*, \textsc{Rsp}^*)$ of m^*. $I_{PK}^{\mathcal{T}}$ then sends $\textsc{Rsp}^*$ to V_{PK} as the answer to the challenge $\textsc{Ch}^*$. Since $p_F(n)$ is non-negligible, there is a d such that $p_F(n) > \frac{1}{n^d}$ for infinitely many n. What is the probability that $I_{PK}^{\mathcal{T}}$ breaks the security of ID, namely that

$V_{PK}(\textsc{Cmt}^*, \textsc{Ch}^*, \textsc{Rsp}^*) = 1$? If $I_{PK}^{\mathcal{T}}$ correctly guesses the index of the crucial query and there are no "commitment collisions" (recall that these only occur with negligible probability), then his simulation of $F_{PK}^{\mathcal{R},\mathcal{S}}$ is perfect; in that case his success probability is just $p_F(n)$. Since the choice of i is independent of the simulation, we get:

$$\begin{aligned}\Pr[V_{PK}(\textsc{Cmt}^*, \textsc{Ch}^*, \textsc{Rsp}^*) = 1] &\approx \frac{1}{q_{\mathcal{R}}(n)} \cdot p_F(n) \geq \frac{1}{n^c} \cdot p_F(n) \\ &> \frac{1}{n^c} \cdot \frac{1}{n^d} = \frac{1}{n^{c+d}} \text{ for infinitely many } n.\end{aligned}$$

$I_{PK}^{\mathcal{T}}$ therefore breaks the passive security of ID, so that $SIG(ID)$ is secure in the ROM if ID is passively secure. ■

4 The non-triviality assumption

It's not hard to show that passive security of ID is a necessary condition for $SID(ID)$ to be secure in the ROM, meaning that ID is passively secure whenever $SIG(ID)$ is secure in the ROM; Abdalla et al. claim that non-triviality is also necessary. To support this claim, they show that, subject to an assumption, there exists a passively secure trivial identification scheme which yields a Fiat-Shamir signature scheme that is *not* secure in the ROM. However, below we show that, subject to a different assumption, there exists a passively secure trivial canonical identification scheme ID' such that $SIG(ID')$ *is* secure in the ROM. Thus, in some sense at least, the non-triviality assumption is *not* necessary.

Let $\mathcal{F} = (G, f, f')$ be a trapdoor permutation (see Chapter 2, Section 4 for the relevant definitions), and consider the following identification scheme $ID' = (G, P', V')$. ID''s key generator G is identical to that of $\mathcal{F}$, so $PK = k$ and $SK = k'$. Informally, we think of ID' as a two-round scheme: the verifier V' challenges the prover P' to invert f_k on a random string $\textsc{Ch}$, accepting if and only if P' does so successfully. Formally, ID' is a canonical (three-round) scheme where P''s commitment $\textsc{Cmt}$ is fixed, say $\textsc{Cmt} = \lambda$ for concreteness. The verifier V', who knows k, accepts a transcript $(\textsc{Ch}, \textsc{Rsp})$ if and only if $f_k(\text{RSP}) = \text{CH}$ (since the commitment λ is fixed, it may be omitted from the transcript). We refer to such schemes as "hyper-trivial", since not only is the entropy of their commitment distribution "low", it's actually zero.

First, we show that ID' is passively secure. Observe that, although for completeness we establish it directly, the passive security of ID' also follows from the fact that $SIG(ID')$ is secure in the ROM (as shown below). Suppose that $I^{\mathcal{T}}$ is a polynomial-time passive impersonator who breaks the security of ID', and denote his (non-negligible) success probability by $p_I(n)$. We use $I^{\mathcal{T}}$ to construct a polynomial-time inverter INV who breaks the one-wayness of f_k.

Recall that INV is given 1^n, k, $y \in \{0,1\}^n$ and some random bits, and his goal is to output an $x \in \{0,1\}^n$ such that $f_k(x) = y$. INV_k simulates $I_k^{\mathcal{T}}$ as follows. Whenever $I_k^{\mathcal{T}}$ queries the transcript oracle $\mathcal{T}$, INV_k randomly chooses $x' \in \{0,1\}^n$, computes $y' = f_k(x') \in \{0,1\}^n$ and gives (y', x') to $I_k^{\mathcal{T}}$. Since f_k is a bijection, setting y' to $f_k(x')$ for a random x' is equivalent to setting x' to $f'_{k'}(y')$ for a random y', so (y', x') has exactly the right distribution. Once $I_k^{\mathcal{T}}$ outputs λ to signal he is ready to be challenged, INV_k gives him y, receiving RSP' in reply. INV_k then outputs RSP' as his guess at $f'_{k'}(y)$. Since INV_k's simulation of $I_k^{\mathcal{T}}$ is perfect, $f_k(\text{RSP}') = y$ with probability $p_I(n)$, which is non-negligible. INV_k therefore breaks the one-wayness of f_k, so that ID' is passively secure.

Next, we show that $SIG(ID')$ is secure in the ROM. Suppose that $F^{\mathcal{R},\mathcal{S}}$ is a polynomial-time forger who breaks the security of $SIG(ID')$ in the ROM, and denote his (non-negligible) success probability by $p_F(n)$. We use $F^{\mathcal{R},\mathcal{S}}$ to construct a polynomial-time inverter INV who breaks the one-wayness of f_k.

Recall that $F^{\mathcal{R},\mathcal{S}}$ is given 1^n, k and some random bits, and his goal is to output a new message m^* together with a signature RSP^* such that $f_k(\text{RSP}^*) = \mathcal{R}(m^*)$. Whenever $F_k^{\mathcal{R},\mathcal{S}}$ queries $\mathcal{S}$ on a message $m \in \{0,1\}^*$, he is given $f'_{k'}(\mathcal{R}(m))$. As in Section 3, we assume, without loss of generality, that $F_k^{\mathcal{R},\mathcal{S}}$ doesn't query $\mathcal{R}$ on "short strings" (i.e.

strings whose length is less than $\ell(n)$), doesn't query $\mathcal{R}$ on the same string more than once, and makes the "crucial query" $\mathcal{R}(m^*)$. We additionally assume that $F_k^{\mathcal{R},\mathcal{S}}$ doesn't query $\mathcal{S}$ on the same message more than once. There is no loss of generality in making this assumption, because signing in $SIG(ID')$ is deterministic (since P' is deterministic). Let $q_{\mathcal{R}}(n)$ and $q_{\mathcal{S}}(n)$ denote the number of times $F_k^{\mathcal{R},\mathcal{S}}$ queries $\mathcal{R}$ and $\mathcal{S}$, respectively, and suppose that the running time of $F_k^{\mathcal{R},\mathcal{S}}$ is bounded above by n^c; such a c must exist because $F_k^{\mathcal{R},\mathcal{S}}$ runs in strict polynomial time. Observe that $q_{\mathcal{R}}(n) + q_{\mathcal{S}}(n) \leq n^c$, since in the worst case $F_k^{\mathcal{R},\mathcal{S}}$ queries an oracle at every step of his execution.

Recall that INV is given 1^n, k, $y \in \{0,1\}^n$ and some random bits, and his goal is to output an $x \in \{0,1\}^n$ such that $f_k(x) = y$. As in Section 3, INV_k first randomly chooses an index $i \in \{1, \ldots, q_{\mathcal{R}}(n)\}$; i represents INV_k's guess at the index of the crucial query, and won't be revealed to $F_k^{\mathcal{R},\mathcal{S}}$ in the course of the simulation. Since $F_k^{\mathcal{R},\mathcal{S}}$ gets no information about i, INV_k guesses the index of the crucial query correctly with probability $\frac{1}{q_{\mathcal{R}}(n)} \geq \frac{1}{n^c}$.

Before beginning the simulation proper, INV_k generates n^c "transcripts" $(y_1, x_1), \ldots, (y_{n^c}, x_{n^c})$ by randomly choosing $x_j \in \{0,1\}^n$ and setting $y_j = f_k(x_j)$ for $1 \leq j \leq n^c$; since f_k is a bijection, this is equivalent to randomly choosing y_j and setting x_j to $f'_{k'}(y_j)$. The idea of generating transcripts ahead of time is key, since it later enables INV_k to consistently answer $F_k^{\mathcal{R},\mathcal{S}}$'s oracle queries; recall from Section 3 that every $\mathcal{S}$ query effectively involves an implicit $\mathcal{R}$ query, and this time we can't rely on the non-triviality assumption to bail us out. A similar technique is used to prove our main result in Section 5.

INV_k now begins his simulation of $F_k^{\mathcal{R},\mathcal{S}}$. As in Section 3, the i^{th} random oracle query is treated specially. Since in this case the commitment λ is fixed, INV_k simply answers the query with y (the string he is trying to invert f_k on). The rest of $F_k^{\mathcal{R},\mathcal{S}}$'s oracle queries are handled as follows. Each time $F_k^{\mathcal{R},\mathcal{S}}$ queries an oracle on a new message $m \in \{0,1\}^*$, INV_k associates an unused transcript (y_j, x_j) with m; all oracle queries regarding m are answered using (y_j, x_j). Specifically, INV_k sets $\mathcal{R}(m)$ to y_j and $\mathcal{S}(m)$ to x_j. Note that INV_k won't run out of transcripts, because $F_k^{\mathcal{R},\mathcal{S}}$ queries his oracles on at most n^c distinct messages.

$F_k^{\mathcal{R},\mathcal{S}}$ eventually outputs RSP^* (purportedly a signature of m^*), which INV_k then outputs as his guess at $f'_{k'}(y)$. Since INV_k's simulation of $F_k^{\mathcal{R},\mathcal{S}}$ is perfect and the choice of i is independent of it, INV_k succeeds with probability at least $\frac{1}{n^c} \cdot p_F(n)$, which is non-negligible. INV_k therefore breaks the one-wayness of f_k, so that $SIG(ID')$ is secure in the ROM.

Here we have only shown that ID', which is hyper-trivial, yields a Fiat-Shamir signature scheme that is secure in the ROM. However, a similar argument demonstrates that every passively secure canonical identification scheme whose prover is deterministic (trivial or not) does.

5 Our result

In this section, we prove the following theorem.

Theorem. *For every (actively) secure canonical identification scheme $ID = (G, P, V)$, the corresponding Fiat-Shamir signature scheme, $SIG(ID) = (G, SIGN^{\mathcal{R}}, VER^{\mathcal{R}})$, is secure in the ROM.*

Proof. Suppose that $F^{\mathcal{R},\mathcal{S}}$ is a polynomial-time forger who breaks the security of $SIG(ID)$ in the ROM, and denote his (non-negligible) success probability by $p_F(n)$. We use $F^{\mathcal{R},\mathcal{S}}$ to construct an active impersonator I who breaks the security of ID.

Recall that $F^{\mathcal{R},\mathcal{S}}$ is given 1^n, PK and some random bits, and his goal is to output a new message m^* together with a valid signature $(\text{CMT}^*, \text{RSP}^*)$ of m^*. As in previous sections, we make a number of "regularity assumptions" about $F_{PK}^{\mathcal{R},\mathcal{S}}$, without loss of generality: $F_{PK}^{\mathcal{R},\mathcal{S}}$ doesn't query $\mathcal{R}$ on "short strings" (i.e. strings whose length is less than $\ell(n)$), doesn't query $\mathcal{R}$ on the same string more than once, and makes the "crucial query" $\mathcal{R}(\textsc{Cmt}^*, m^*)$ at some point during his execution. Note, however, that $F_{PK}^{\mathcal{R},\mathcal{S}}$ may query $\mathcal{S}$ on the same message more than once. Since signing in $SIG(ID)$ is probabilistic, this makes perfect sense and could yield useful information.

Recall that I is given 1^n, PK and some random bits. As per the definition of active security (see Chapter 2, Section 6), I_{PK} first gets to interact with P_{SK} polynomially many times in the role of V_{PK}. Each time, I_{PK} receives a commitment $\textsc{Cmt} \in \{0,1\}^{\ell(n)}$ from P_{SK}, sends a (not necessarily random) challenge $\textsc{Ch} \in \{0,1\}^n$ to P_{SK}, and then receives a response $\textsc{Rsp}$ from P_{SK}. Next, I_{PK} sends a commitment $\textsc{Cmt}'$ to V_{PK} — this marks the end of his "interactive" phase — receiving a random challenge $\textsc{Ch}'$ in reply. His goal is to output a response $\textsc{Rsp}'$ such that $V_{PK}(\textsc{Cmt}', \textsc{Ch}', \textsc{Rsp}') = 1$.

Let $q_{\mathcal{R}}(n)$ and $q_{\mathcal{S}}(n)$ denote the number of times $F_{PK}^{\mathcal{R},\mathcal{S}}$ queries $\mathcal{R}$ and $\mathcal{S}$, respectively, and set $q(n) = q_{\mathcal{R}}(n) + q_{\mathcal{S}}(n)$. Also, suppose that the running time of $F_{PK}^{\mathcal{R},\mathcal{S}}$ is bounded above by n^c; such a c must exist since $F_{PK}^{\mathcal{R},\mathcal{S}}$ runs in strict polynomial time. Observe that $q(n) \leq n^c$, since in the worst case $F_{PK}^{\mathcal{R},\mathcal{S}}$ queries an oracle at every step of his execution.

Before beginning his simulation of $F_{PK}^{\mathcal{R},\mathcal{S}}$, our impersonator I_{PK} obtains $q(n)$ "transcript blocks" $\mathcal{B}_1, \ldots, \mathcal{B}_{q(n)}$, each consisting of $q_{\mathcal{S}}(n)$ transcripts, by interacting with P_{SK}. A new transcript is added to a given block $\mathcal{B}_k$ as follows. I_{PK} first receives a commitment $\textsc{Cmt} \in \{0,1\}^{\ell(n)}$ from P_{SK} (it's chosen according to P_{SK}'s commitment distribution, $\mathcal{P}_{SK}$). I_{PK} next needs to decide what challenge $\textsc{Ch} \in \{0,1\}^{\ell(n)}$ to send to P_{SK}. If $\textsc{Cmt}$ does not appear in any of the transcripts already contained in $\mathcal{B}_k$, I_{PK} chooses $\textsc{Ch}$ randomly. Otherwise, he sets $\textsc{Ch}$ to the challenge associated with $\textsc{Cmt}$ (since $\textsc{Ch}$ repeats whenever $\textsc{Cmt}$ does, every commitment in $\mathcal{B}_k$ is associated with some particular challenge). After sending $\textsc{Ch}$ to P_{SK}, I_{PK} receives a response $\textsc{Rsp}$; the transcript $(\textsc{Cmt}, \textsc{Ch}, \textsc{Rsp})$ is then added to $\mathcal{B}_k$. Once the transcript blocks have been generated, I_{PK} randomly chooses an index $i \in \{1, \ldots, q_{\mathcal{R}}(n)\}$; i represents I_{PK}'s guess at the index of the crucial query, and won't be revealed to $F_{PK}^{\mathcal{R},\mathcal{S}}$ in the course of the simulation.

I_{PK} now begins his simulation of $F_{PK}^{\mathcal{R},\mathcal{S}}$. Notice that, thanks to our "regularity assumptions" above, every oracle query made by $F_{PK}^{\mathcal{R},\mathcal{S}}$ can be unambiguously associated with some message $m \in \{0,1\}^*$. Let $m_1, m_2, m_3, \ldots$ be the *distinct* messages associated with $F_{PK}^{\mathcal{R},\mathcal{S}}$'s oracle queries (there are at most $q(n)$ such messages). I_{PK} answers $\mathcal{R}$ and $\mathcal{S}$ queries associated with the k^{th} distinct message m_k using transcripts contained in the k^{th} block $\mathcal{B}_k$. The idea is to ensure that I_{PK} can answer as many $\mathcal{S}(m_k)$ queries as necessary — there will be at most $q_{\mathcal{S}}(n)$ — in a way that is consistent with his answers to queries of the form $\mathcal{R}(\textsc{Cmt}, m_k)$, $\textsc{Cmt} \in \{0,1\}^{\ell(n)}$. Specifically, I_{PK} answers $F_{PK}^{\mathcal{R},\mathcal{S}}$'s j^{th} $\mathcal{S}(m_k)$ query with $(\textsc{Cmt}, \textsc{Rsp})$, where $(\textsc{Cmt}, \textsc{Ch}, \textsc{Rsp})$ is the j^{th} transcript contained in $\mathcal{B}_k$. His answer to queries of the form $\mathcal{R}(\textsc{Cmt}, m_k)$ depends on whether the commitment $\textsc{Cmt} \in \{0,1\}^{\ell(n)}$ appears in any of the transcripts in $\mathcal{B}_k$. If so, I_{PK} sets $\mathcal{R}(\textsc{Cmt}, m_k)$ to $\textsc{Ch} \in \{0,1\}^n$, the challenge associated with $\textsc{Cmt}$. Otherwise, he randomly chooses an $r \in \{0,1\}^n$ and sets $\mathcal{R}(\textsc{Cmt}, m_k)$ to r.

As in previous sections, the i^{th} random oracle query is handled specially. Suppose that, for his i^{th} random oracle query, $F_{PK}^{\mathcal{R},\mathcal{S}}$ queries $\mathcal{R}$ on some $s \in \{0,1\}^*$ (recall that $|s| \geq \ell(n)$, since $F_{PK}^{\mathcal{R},\mathcal{S}}$ doesn't query $\mathcal{R}$ on "short strings"). $I_{PK}^{\mathcal{T}}$ parses s as $(\textsc{Cmt}', m')$, sends $\textsc{Cmt}' \in \{0,1\}^{\ell(n)}$ to V_{PK}, and receives a challenge $\textsc{Ch}' \in \{0,1\}^n$ in reply. He then gives $\textsc{Ch}'$ to $F_{PK}^{\mathcal{R},\mathcal{S}}$ as the answer to $\mathcal{R}(s)$ and continues his simulation.

Eventually, $F_{PK}^{\mathcal{R},\mathcal{S}}$ outputs a message m^* together with a purported signature $(\textsc{Cmt}^*$, $\textsc{Rsp}^*)$ of m^*. I_{PK} then sends $\textsc{Rsp}^*$ to V_{PK} as the answer to the challenge $\textsc{Ch}'$. If I_{PK} guessed the index of the crucial query correctly, then $m^* = m'$ and $\textsc{Cmt}^* = \textsc{Cmt}'$, so that $\mathcal{R}(\textsc{Cmt}^*, m^*) = \textsc{Ch}'$. In that case, I_{PK}'s simulation of $F_{PK}^{\mathcal{R},\mathcal{S}}$ is perfect. Let A denote the event that $V_{PK}(\textsc{Cmt}', \textsc{Ch}', \textsc{Rsp}^*) = 1$ and B denote the event that $i = i^*$, where i^* is the true index of the crucial query. Since $p_F(n) > \frac{1}{n^d}$ for some d and infinitely many n (because $F^{\mathcal{R},\mathcal{S}}$ breaks the security of $SIG(ID)$ in the ROM), we get:

$$\begin{aligned}\Pr[A] &\geq \Pr[A, B] = \Pr[A \mid B] \cdot \Pr[B] \\ &= p_F(n) \cdot \frac{1}{q_{\mathcal{R}}(n)} \geq \frac{p_F(n)}{n^c} > \frac{1}{n^{c+d}} \text{ for infinitely many } n.\end{aligned}$$

I_{PK} therefore breaks the security of ID, so that $SIG(ID)$ is secure in the ROM. ■

Chapter 4

A public-key encryption scheme CCA2-secure in the ROM

1 Public-key encryption in the ROM: an overview

In [BR93], Bellare and Rogaway proposed the following public-key encryption scheme, which they showed to be CCA2-secure in the ROM (see Sections 7 and 8 of Chapter 2 for the relevant definitions). Let (G, f, f') be a trapdoor permutation (see Section 4 of Chapter 2). The key generator $GEN^{\mathcal{R}}$ simulates G to obtain a pair of keys (k, k') and sets $pub = k$, $pri = k'$. Let n be the security parameter. To encrypt a message $m \in \{0,1\}^n$, $ENC_k^{\mathcal{R}}$ chooses $r \in \{0,1\}^n$ randomly, computes $y = f_k(r)$ and sets e_m to $(y, \mathcal{R}(r) \oplus m, \mathcal{R}(r, m))$. Given a purported encryption $e = (\alpha, \beta, \gamma)$, $DEC_{k'}^{\mathcal{R}}$ computes $r = f'_{k'}(\alpha)$ and sets $m = \beta \oplus \mathcal{R}(r)$. If $\gamma \neq \mathcal{R}(r, m)$, he outputs $\perp$, indicating a failure to decrypt; otherwise, he outputs m as the decryption of e. Intuitively, r is hard to find because f_k is hard to invert. Together with the randomness of $\mathcal{R}$, this implies that $\mathcal{R}(r) \oplus m$ yields no information about m, which guarantees semantic security. The "authentification code" $\mathcal{R}(r, m)$ ensures that encryptions are difficult to "malleate".

One drawback of the above scheme is that encryptions are of length about $3n$, or roughly three times the length of the message being encrypted. To address this issue, Bellare and Rogaway introduced the highly influential Optimal Asymmetric Encryption Padding or OAEP scheme in [BR94], a simplified version[1] of which is described next. Let $\mathcal{F} = (G, f, f')$ be a trapdoor permutation. As before, the key generator $GEN^{\mathcal{R}}$ simulates G to obtain a pair of keys (k, k') and sets $pub = k$, $pri = k'$. Let n be the security parameter. To simplify the presentation, it will be convenient to assume that f_k and $f'_{k'}$ map $2n$ bits to $2n$ bits (as opposed to n bits to n bits). To encrypt a message $m \in \{0,1\}^n$, $ENC_k^{\mathcal{R}}$ chooses $r \in \{0,1\}^n$ randomly, computes $\rho(m) = (m \oplus \mathcal{R}(r), r \oplus \mathcal{R}(m \oplus \mathcal{R}(r)))$ and sets $e_m = f_k(\rho(m))$; ρ is sometimes called the *padding function*. Given a purported encryption $\alpha \in \{0,1\}^{2n}$, $DEC_{k'}^{\mathcal{R}}$ computes $(\beta, \gamma) = f'_{k'}(\alpha)$, where $|\beta| = |\gamma| = n$, and sets $r = \gamma \oplus \mathcal{R}(\beta)$. He then outputs $m = \beta \oplus \mathcal{R}(r) \in \{0,1\}^n$ as the decryption of α; notice that every α is a valid encryption of *some* m, so that $DEC_{k'}^{\mathcal{R}}$ never outputs $\perp$. This scheme was shown to be "plaintext aware" in [BR94], where it was also claimed (without proof) that plaintext awareness implies "security against chosen-ciphertext attack" (it's not entirely clear whether the authors had CCA1 or CCA2 security in mind).

After Bleichenbacher showed in [Ble98] that version 1.5 of RSA Security's PKCS #1 standard ([RSA93]) is vulnerable to chosen-ciphertext attack, RSA-OAEP (a concrete implementation of the OAEP scheme where the role of $\mathcal{F}$ is played by the RSA function) served as the basis for version 2.0 of the standard ([RSA98]). RSA-OAEP was subsequently also incorporated into IEEE's public-key cryptography standard, IEEE P1363-2000 ([IEE00]). However, in [Sho01] Shoup pointed out that, although OAEP is indeed *CCA1*-secure, there is an (additional, not random) oracle relative to which $\mathcal{F}$ remains one-way but OAEP fails to be *CCA2*-secure. Since standard "black-box" security reductions relativize — that is, hold relative to *every* oracle — any reduction from inverting $\mathcal{F}$ to breaking the CCA2 security of OAEP would therefore have to be "non-black-box",

[1]The real scheme makes use of two independent random oracles, $\mathcal{G}$ and $\mathcal{H}$, and has three security parameters.

meaning that it would need to somehow depend on the specifics of $\mathcal{F}$. Shoup also proposed a modification of OAEP, called OAEP+, which he proved to be CCA2-secure in the ROM. In [FOPS01], Fujisaki, Okamoto, Pointcheval and Stern proved that OAEP is CCA2-secure under the stronger assumption that $\mathcal{F}$ is "partial domain one-way", as opposed to "full domain one-way" or simply one-way (also see [FOPS04] for the journal version). Since RSA is "random self-reducible" — loosely, this means that being able to invert it on a large fraction of the inputs allows one to invert it on every single input — it is "partial domain one-way" if and only if it is one-way. Thus, Shoup's result notwithstanding, RSA-OAEP is in fact CCA2-secure under the RSA assumption.

In [Bon01], Boneh observed that the OAEP padding function ρ may be viewed as two rounds of a "Feistel network" and proposed two simpler, more elegant single-round padding functions. When used in conjunction with either RSA or Rabin's modular squaring function, these new paddings yield encryption schemes which are CCA2-secure in the ROM (under the assumption that RSA is hard to invert and factoring is hard, respectively). Interestingly, Boneh recommends using the Rabin function in preference to RSA where his paddings are concerned, since it has better "reduction efficiency".

In [CS98], Cramer and Shoup described an efficient, practical public-key encryption scheme (its efficiency is comparable to that of RSA-OAEP) which is CCA2-secure in the "real world" (and hence also in the ROM) under the non-standard-yet-highly-plausible "Decisional Diffie-Hellman" assumption. This important result is perhaps the main reason why there hasn't been a great deal of work done on discrete logarithm-based public-key encryption schemes which are CCA2-secure in the ROM.

2 The original scheme

In [BR97], Bellare and Rogaway proposed a discrete logarithm-based public-key encryption scheme called the Diffie-Hellman Integrated Encryption Scheme or DHIES, which they claimed is CCA2-secure in the ROM under the "Computational Diffie-Hellman" assumption (a standard discrete log-type hardness assumption). However, in [ABR01b] Abdalla, Bellare and Rogaway conceded that DHIES is unlikely to be CCA2-secure in the ROM under the above assumption (see also [ABR01a]). Instead, they proved that DHIES is CCA2-secure in the "real world" under a strong, non-standard Diffie-Hellman-type assumption called the "Hash Diffie-Hellman" assumption; their new focus on "real-world" security (as opposed to security in the ROM) was likely a response to Cramer and Shoup's 1998 discovery ([CS98]) of a practical public-key encryption scheme which is CCA2-secure in the "real world" under the non-standard-yet-plausible "Decisional Diffie-Hellman" assumption. Although its security rests on a rather less believable assumption, DHIES is somewhat more efficient than Cramer and Shoup's scheme. We now give an informal description of the DHIES encryption scheme.

Let G be a cyclic multiplicative group of order $p-1$, where p is some n-bit prime (n being the security parameter). For concreteness, think of G as $\mathcal{Z}_p^* = \{1, 2, \ldots, p-1\}$ (here the group operation is multiplication mod p). We will need to assume that membership in G is efficiently testable, which is the case for $\mathcal{Z}_p^*$. It will also sometimes be convenient to treat the elements of G as strings over $\{0, 1\}$, say via their binary encoding.

Fix a generator $g \in G$, so that $G = \{g, g^2, \ldots, g^{p-1}\}$; g is implicitly given to all participants, as are p and 1^n. Informally, the Computational Diffie-Hellman assumption (often abbreviated as the CDH assumption) with respect to G says that $g^{uv} \in G$ is hard to compute from $g^u \in G$ and $g^v \in G$. Formally, the CDH assumption holds for G if, for every probabilistic polynomial-time adversary A who is given $g^u \in G$ and $g^v \in G$ for randomly chosen $u, v \in \{1, \ldots, p-1\}$, the probability $p_A(n)$ that A outputs $g^{uv} \in G$ is negligible; here $p_A(n)$ is taken over the random bits of A as well as the choice of u and v.

We will need a secure MAC $\mathcal{M} = (SIGN, VER)$ (see Section 5 of Chapter 2) and a secure "private-key encryption scheme" $\mathcal{E} = (ENC, DEC)$ (the latter haven't actually been formally defined in this thesis). $\mathcal{M}$ must also be "non-malleable" in the following sense: given a signature σ of some message m, it is infeasible to find another signature $\sigma' \neq \sigma$ of m. Both secure non-malleable MACs and secure "private-key encryption schemes" can be implemented using "pseudorandom function generators" ([GGM84b]), which exist if one-way functions (see Section 2 of Chapter 2) do ([GGM84a], [GGM86]).

To generate a matching public/private key pair, randomly choose a $v \in \{1, \ldots, p-1\}$ and set $pub = g^v \in G$, $pri = v$.

To encrypt a message $m \in \{0,1\}^n$ given a public key g^v, first randomly choose $u \in \{1, \ldots, p-1\}$ and compute $g^u \in G$, $(g^v)^u = g^{uv} \in G$. Next, query the random oracle $\mathcal{R}$ on (g^u, g^{uv}) (notice that here we are treating elements of G as binary strings) to obtain two keys $k_1, k_2 \in \{0,1\}^n$ (here we assume for convenience that $\mathcal{R} : \{0,1\}^* \to \{0,1\}^{2n}$). Finally, compute $s = ENC_{k_1}(m)$, $t = SIGN_{k_2}(s)$ and output $e_m = (g^u, s, t)$ as the encryption of m.

To decrypt a purported ciphertext $e = (\alpha, \beta, \gamma)$ given a private key v, proceed as follows. If $\alpha \notin G$, output $\perp$, indicating a failure to decrypt. Otherwise, compute $\alpha^v \in G$ and query $\mathcal{R}$ on (α, α^v) to obtain $k_1, k_2 \in \{0,1\}^n$. If $VER_{k_2}(\beta, \gamma) = 0$, output $\perp$. Otherwise, output $DEC_{k_1}(\beta)$ as the decryption of e.

Next, we briefly argue why DHIES is unlikely to be CCA2-secure in the ROM (without actually *proving* that it isn't). The standard way to demonstrate that a public-key encryption scheme is CCA2-secure is to show that it is both semantically secure (see Chapter 2, Section 7) and "plaintext-aware". Informally, a public-key encryption scheme is plaintext-aware if the decryption oracle $\mathcal{D}$ is useless to the CCA2 adversary ADV, meaning that ADV is only able to get $\mathcal{D}$ to decrypt ciphertexts he could have decrypted himself. More formally, we say that a public-key encryption scheme is *plaintext-aware in the ROM* if, for every probabilistic polynomial-time adversary $ADV^{\mathcal{R},\mathcal{D}}$, there is a corresponding probabilistic polynomial-time adversary $A^{\mathcal{R}}$ such that the difference between the success probabilities of $ADV^{\mathcal{R},\mathcal{D}}$ and $A^{\mathcal{R}}$ is negligible (in n). $A^{\mathcal{R}}$ essentially simulates $ADV^{\mathcal{R},\mathcal{D}}$, computing $\mathcal{D}$'s answers himself based on $ADV^{\mathcal{R},\mathcal{D}}$'s view. Below, we provide evidence that DHIES fails to have this property.

Recall that $ADV^{\mathcal{R},\mathcal{D}}$ is given $g^v \in G$, chooses a pair of messages $m_0, m_1 \in \{0,1\}^n$ and then sees an encryption $e_b = (g^u, s = ENC_{k_1}(m_b), t = SIGN_{k_2}(s))$, where $k_1 k_2 = \mathcal{R}(g^u, g^{uv})$ and $b \in \{0,1\}$, $u \in \{1, \ldots, p-1\}$ are chosen randomly. Since $ADV^{\mathcal{R},\mathcal{D}}$ is not allowed to query $\mathcal{D}$ on e_b, he must modify at least one of g^u, s and t.

If $ADV^{\mathcal{R},\mathcal{D}}$ queries $\mathcal{D}$ on $e' = (g^{u'}, s', t')$, where either $s' \neq s$ or $t' \neq t$ (or both), $A^{\mathcal{R}}$ can safely respond with $\perp$. This is almost certainly the right answer, because in order for e' to be a valid ciphertext $ADV^{\mathcal{R},\mathcal{D}}$ would need to either sign a new message (if $s' \neq s$), or come up with another signature of an old message (if $s' = s$); the former is infeasible because $\mathcal{M}$ is secure, whereas the latter is infeasible because $\mathcal{M}$ is non-malleable. If, on the other hand, $ADV^{\mathcal{R},\mathcal{D}}$ queries $\mathcal{D}$ on $(g^{u'}, s, t)$ for some $u' \neq u$, the correct answer is again almost certainly $\perp$, provided that $g^{u'v} \neq g^{uv}$. Although we haven't done so, we could ensure that is the case by stipulating that $|G| = q$ for some prime q.

However, as we shall now see, $\mathcal{D}$ allows $ADV^{\mathcal{R},\mathcal{D}}$ to determine, given any $\alpha, \beta \in G$, whether $\alpha^v = \beta$ (recall that $ADV^{\mathcal{R},\mathcal{D}}$ does *not* know v). First, $ADV^{\mathcal{R},\mathcal{D}}$ computes $k_1 k_2 = \mathcal{R}(\alpha, \beta)$ and creates an encryption $e' = (\alpha, s' = ENC_{k_1}(m), t' = SIGN_{k_2}(s'))$ of some message $m \in \{0,1\}^n$, say $\bar{0}$ for definiteness. Next, he queries $\mathcal{D}$ on e'. It is easy to show that $\mathcal{D}(e') = m$ (as opposed to $\perp$) if and only if $\alpha^v = \beta$. Since it is by no means clear how $A^{\mathcal{R}}$ would emulate such a functionality, a stronger assumption than CDH is apparently required to ensure that DHIES is CCA2-secure in the ROM.

3 Our modification

We now describe a modified version of DHIES, called DHIES+, which is provably secure in the ROM under the CDH assumption. Although in what follows we only show how to encrypt a single bit in order to simplify the presentation, our scheme can be easily extended to encrypt n-bit messages with the aid of a secure MAC and a secure private-key encryption scheme.

Let G be a cyclic multiplicative group of order q, where q is some n-bit prime (n being the security parameter). For concreteness, think of G as a subgroup of $\mathcal{Z}_p^*$, where $p > q$ is some other prime. Notice that here, unlike in Section 2, $|G|$ is prime. This will be important in Section 3.2 below. Fix a generator $g \in G$, so that $G = \{g, g^2, \ldots, g^q\}$; g is once again implicitly given to all participants, as are q and 1^n.

To generate a matching public/private key pair, randomly choose a $v \in \{1, \ldots, q\}$ and set $pub = g^v \in G$, $pri = v$.

To encrypt a bit b given a public key g^v, first randomly choose $u \in \{1, \ldots, q\}$ and compute $g^u \in G$, $(g^v)^u = g^{uv} \in G$. Next, query $\mathcal{R}$ to obtain $s_0 s_1 r = \mathcal{R}(g^{uv})$, where $s_0, s_1, r \in \{0,1\}^n$ (here we assume for convenience that $\mathcal{R} : \{0,1\}^* \rightarrow \{0,1\}^{3n}$). Finally, compute $t = u \oplus r$ and output $e_b = (g^u, s_b, t)$ as the encryption of b.

To decrypt a purported ciphertext $e = (\alpha, \beta, \gamma)$ given a private key v, proceed as follows:

1. If $\alpha \notin G$, output $\perp$, indicating a failure to decrypt.

2. Compute $\alpha^v \in G$ and query $\mathcal{R}$ to obtain $s_0 s_1 r = \mathcal{R}(\alpha^v)$.

3. Set $u = \gamma \oplus r$ and compute $g^u \in G$. If $\alpha \neq g^u$, output $\perp$.

4. If $\beta \notin \{s_0, s_1\}$, output $\perp$. Otherwise, output a b such that $\beta = s_b$ as the decryption of e.

Remark. Notice that there is a small probability ($\frac{1}{2^n}$, to be exact) that $s_0 = s_1$, in which case we won't be able to decrypt e correctly. This unlikely occurrence can be avoided by making the scheme slightly more complicated, but we won't go into the details here.

We will prove that DHIES+ is CCA2-secure in the ROM in two stages. First, we'll show that it is semantically secure in the ROM under the CDH assumption. Next, we'll show that it is plaintext-aware in the ROM.

3.1 Semantic security

Intuitively, DHIES+ is semantically secure in the ROM under the CDH assumption because s_b provides the adversary with no information about b unless he can compute $g^{uv} \in G$, which is infeasible if the Diffie-Hellman problem is hard for G. More formally, suppose that $ADV^{\mathcal{R}}$ is a probabilistic polynomial-time adversary who breaks the semantic security (see Chapter 2, Section 7) of DHIES+ in the ROM. Since in this case there are only two possible plaintexts (namely 0 and 1), there is no need to let $ADV^{\mathcal{R}}$ choose m_0 and m_1. Instead, he simply gets a public key $g^v \in G$ and an encryption $e_b = (g^u, s_b, t)$ of a randomly chosen bit b, where $s_0 s_1 r = \mathcal{R}(g^{uv})$ and $t = u \oplus r$. Denote $ADV^{\mathcal{R}}$'s success probability by $p_{ADV}(n)$ and the total number of times he queries $\mathcal{R}$ during his execution by $c(n)$; $p_{ADV}(n)$ is taken over $ADV^{\mathcal{R}}$'s random bits, as well as the randomness of $\mathcal{R}$ and the choice of b, u and v. Since $ADV^{\mathcal{R}}$ runs in strict polynomial time, $c(n) \leq n^c$ for some c. We may assume without loss of generality that $ADV^{\mathcal{R}}$ never queries $\mathcal{R}$ on the same string more than once, since $\mathcal{R}$'s response would be identical.

We use $ADV^{\mathcal{R}}$ to construct a probabilistic polynomial-time solver S who, given $g^u \in G$ and $g^v \in G$ for randomly chosen $u, v \in \{1, \ldots, q\}$, outputs $g^{uv} \in G$ with non-negligible probability. S first randomly chooses $i \in \{1, \ldots, c(n)\}$ and $s, t \in \{0,1\}^n$. He then simulates $ADV^{\mathcal{R}}$ on $(g^v, (g^u, s, t))$, answering all of $ADV^{\mathcal{R}}$'s $\mathcal{R}$ queries randomly. As soon as $ADV^{\mathcal{R}}$ asks his i^{th} random oracle query, $\mathcal{R}(m)$, S ends the simulation and outputs m as the value of g^{uv} (if $ADV^{\mathcal{R}}$ terminates before asking the i^{th} query or $m \notin G$, S outputs some dummy value such as g^u).

Let A be the event that $ADV^{\mathcal{R}}$ succeeds and B be the event that $ADV^{\mathcal{R}}$ queries $\mathcal{R}$ on $g^{uv} \in G$ at some point. Observe that

$$
\begin{aligned}
p_{ADV}(n) = \Pr[A] &= \Pr[A, B] + \Pr[A, \overline{B}] \\
&= \Pr[A, B] + \Pr[A \mid \overline{B}] \cdot \Pr[\overline{B}] \\
&\leq \Pr[A, B] + \Pr[A \mid \overline{B}],
\end{aligned}
$$

and note that $\Pr[A \mid \overline{B}] = \frac{1}{2}$, because in that case s_b yields no information about b. Since $ADV^{\mathcal{R}}$ breaks the semantic security of DHIES+ in the ROM, $p_{ADV}(n) > \frac{1}{2} + \frac{1}{n^d}$ for some d and infinitely many n. We therefore have:

$$\begin{aligned}\Pr[A, B] &\geq p_{ADV}(n) - \Pr[A \mid \overline{B}] = p_{ADV}(n) - \frac{1}{2} \\ &> \frac{1}{2} + \frac{1}{n^d} - \frac{1}{2} = \frac{1}{n^d} \text{ for infinitely many } n.\end{aligned}$$

Now denote the success probability of S (taken over his random bits, as well as the choice of u and v) by $p_S(n)$, and let C be the event that $ADV^{\mathcal{R}}$'s i^{th} random oracle query is $\mathcal{R}(g^{uv})$. Since $i \in \{1, \ldots, c(n)\}$ is chosen uniformly (and independently of the simulation of $A^{\mathcal{R}}$), where $c(n) \leq n^c$, we get:

$$\begin{aligned}p_S(n) = \Pr[A, B, C] = \Pr[C \mid A, B] \cdot \Pr[A, B] &= \frac{1}{c(n)} \cdot \Pr[A, B] \\ &> \frac{1}{c(n)} \cdot \frac{1}{n^d} \geq \frac{1}{n^c} \cdot \frac{1}{n^d} = \frac{1}{n^{c+d}} \text{ for infinitely many } n.\end{aligned}$$

This shows that $p_S(n)$ is non-negligible, so that DHIES+ is semantically secure in the ROM under the CDH assumption. ■

3.2 Plaintext awareness

Informally, DHIES+ is plaintext-aware in the ROM because the fact that u (and not merely g^u) is incorporated into the ciphertext e enables the simulator to not only determine if the adversary knows the decryption of e, but to actually decrypt it himself (albeit with negligible error).

More formally, let $ADV^{\mathcal{R},\mathcal{D}}$ be a probabilistic polynomial-time adversary who is given a public key $g^v \in G$ and attempts to break the CCA2 security (see Chapter 2, Section 7) of DHIES+ in the ROM. Suppose for simplicity that there is no "lunchtime attack" phase, so that $ADV^{\mathcal{R},\mathcal{D}}$ gets an encryption $e_b = (g^u, s_b, u \oplus r)$ of a random bit b (where $s_0 s_1 r = \mathcal{R}(g^{uv})$), queries the decryption oracle $\mathcal{D}$ on a bunch of strings $a_0, a_1, a_2, \ldots$, and finally outputs a bit b'. We may assume that $a_i \in \{0,1\}^{3n}$ (so that $a_i = (\alpha_i, \beta_i, \gamma_i)$ for some $\alpha_i, \beta_i, \gamma_i \in \{0,1\}^n$), since $\mathcal{D}$'s reply would definitely be $\perp$ otherwise. Let $p_{ADV}(n)$ denote the probability that $b' = b$; $p_{ADV}(n)$ taken over $ADV^{\mathcal{R},\mathcal{D}}$'s random bits and the randomness of $\mathcal{R}$, as well as the choice of v, u and b. We must exhibit a probabilistic polynomial-time adversary $A^{\mathcal{R}}$, also given g^v and e_b, such that $|p_{ADV}(n) - p_A(n)|$ is negligible; here $p_A(n)$ is the probability that $A^{\mathcal{R}}$ correctly outputs b, taken over his random bits and the randomness of $\mathcal{R}$, as well as the choice of v, u and b.

Let $A^{\mathcal{R}}$ simulate $ADV^{\mathcal{R},\mathcal{D}}$, answering his $\mathcal{D}(a_i)$ queries as follows. $A^{\mathcal{R}}$ checks whether $ADV^{\mathcal{R},\mathcal{D}}$ has previously queried the random oracle $\mathcal{R}$ on a string w such that $(\alpha_i, \beta_i, \gamma_i) = (g^{u'}, s'_m, u' \oplus r')$, where $w = g^{u'v}$, $s'_0 s'_1 r' = \mathcal{R}(w)$, $u' \in \{1, \ldots, q\}$ and m is a bit; he is basically just trying to determine if $ADV^{\mathcal{R},\mathcal{D}}$ already knows the decryption of a_i. $A^{\mathcal{R}}$'s answer is m if such a w exists and $\perp$ otherwise.

First, observe that whenever $A^{\mathcal{R}}$ answers $\mathcal{D}(a_i)$ with m, so does $\mathcal{D}$, since $\mathcal{D}(g^{u'}, s'_m, u' \oplus r') = m$ provided that $s'_0 s'_1 r' = \mathcal{R}(g^{u'v})$ and $u' \in \{1, \ldots, q\}$. It remains to show that if $A^{\mathcal{R}}$'s answer is $\perp$, then so is $\mathcal{D}$'s (almost certainly, anyway). We consider the following two exhaustive cases.

- **Case #1:** $\alpha_i \neq g^u$

  If $\alpha_i \notin G$ then $\mathcal{D}(a_i) = \perp$ and we are done. Let us therefore suppose that $\alpha_i = g^{u'}$ for some $u' \in \{1, \ldots, q\}$, $u' \neq u$, so that $a_i = (g^{u'}, \beta_i, \gamma_i)$. Set $s'_0 s'_1 r' = \mathcal{R}(g^{u'v})$. Since G has prime order, we are guaranteed that $g^{u'v} \neq g^{uv}$. This matters because we would otherwise have $s'_b = s_b$, $r' = r$, and $ADV^{\mathcal{R},\mathcal{D}}$ has information about both s_b and r by virtue of having seen $e_b = (g^u, s_b, u \oplus r)$.

  - **Subcase #1a:** $ADV^{\mathcal{R},\mathcal{D}}$ has queried $\mathcal{R}$ on $g^{u'v}$

    Since $A^{\mathcal{R}}$'s answer was $\perp$, either $\beta_i \notin \{s'_0, s'_1\}$ or $\gamma_i \neq u' \oplus r'$. In both cases, $\mathcal{D}(a_i) = \perp$.

  - **Subcase #1b:** $ADV^{\mathcal{R},\mathcal{D}}$ hasn't queried $\mathcal{R}$ on $g^{u'v}$

    In this case s'_0, s'_1 and r' are completely random (note that this assertion is justified only because $g^{u'v} \neq g^{uv}$). The probability that a_i is a valid encryption, so that $\mathcal{D}(a_i) \neq \perp$, is therefore

$$\underbrace{\frac{2}{2^n}}_{s'_0 \text{ or } s'_1} \cdot \underbrace{\frac{1}{2^n}}_{r'} = \frac{2}{4^n},$$

which is certainly negligible.

- **Case #2:** $\alpha_i = g^u$
  - **Subcase #2a:** $\gamma_i = u \oplus r$

    If $\beta_i \notin \{s_0, s_1\}$ then $\mathcal{D}(a_i) = \perp$ and we are done, so suppose that $\beta_i = s_{\bar{b}}$ (recall that $ADV^{\mathcal{R},\mathcal{D}}$ isn't allowed to query $\mathcal{D}$ on $e_b = (g^u, s_b, u \oplus r)$). Since $A^{\mathcal{R}}$'s answer was $\perp$, we know that $ADV^{\mathcal{R},\mathcal{D}}$ hasn't queried $\mathcal{R}$ on g^{uv}, and consequently has no information about $s_{\bar{b}}$ (because $ADV^{\mathcal{R},\mathcal{D}}$ hasn't seen $s_{\bar{b}}$, we may view it as not having been chosen yet). The probability that a_i is a valid encryption (so that $\mathcal{D}(a_i) \neq \perp$) is therefore $\frac{1}{2^n}$ in this case, which is again negligible.
  - **Subcase #2b:** $\gamma_i \neq u \oplus r$

    In this case $\mathcal{D}(a_i) = \perp$, so we are done.

We can therefore conclude that the difference between $p_A(n)$ and $p_{ADV}(n)$ is negligible, which means that DHIES+ is plaintext-aware in the ROM. Since we have already shown in Section 3.1 that DHIES+ is semantically secure in the ROM under the CDH assumption, this completes our proof that it is CCA2-secure in the ROM under the CDH assumption. ■

Chapter 5

Uninstantiability

1 The random oracle methodology and "uninstantiable" schemes

As originally proposed in [BR93] and applied in practice, the Random Oracle Methodology involves taking a construction which is secure in the ROM (usually under a standard hardness assumption) and "instantiating" the random oracle $\mathcal{R}$ using a "cryptographically strong" hash function $h : \{0,1\}^* \rightarrow \{0,1\}^n$; whenever $\mathcal{R}$ is queried on $m \in \{0,1\}^*$, the answer is $h(m)$. However, a hash function *ensemble* $\mathcal{H} = \{\mathcal{H}_n\}_{n\in\mathbb{N}}$, $\mathcal{H}_n = \{h_s : \{0,1\}^* \rightarrow \{0,1\}^n\}_{s\in\{0,1\}^n}$ (see Chapter 2, Section 3) must strictly speaking be used instead. The reason is that, as we'll see shortly, it isn't hard to come up with signature and public-key encryption schemes (see sections 5 and 7 of Chapter 2, respectively) which are secure in the ROM yet become hopelessly insecure if $\mathcal{R}$ queries are answered using a fixed function h. We therefore call a signature or public-key encryption scheme *uninstantiable* if it is secure in the ROM (possibly under some hardness assumption) yet insecure in the real world, no matter what hash function ensemble $\mathcal{H}$ is used to instantiate the random oracle $\mathcal{R}$.

Since random oracles are one-way (see Chapter 2, Section 8), the results of [Rom90] imply that signature schemes which are secure in the ROM exist unconditionally[1]. Given a hash function h, we can obtain a signature scheme which is secure in the ROM but cannot be instantiated using h by modifying any signature scheme which is secure in the ROM as follows. Given a message $m \in \{0,1\}^*$, the new signer computes a signature σ of m as before and then checks whether $\mathcal{R}(\bar{0}) = h(\bar{0})$. If so, he outputs (pri, σ); otherwise, he outputs $(\bar{0}, \sigma)$. Given a message m and a purported signature α of m, the new verifier parses α as (β, γ), where $|\beta| = n$, and then checks whether γ is a valid signature of m as before. Observe that our modification does not violate the correctness of the scheme, since every signature output by the signer is accepted by the verifier, whether $\mathcal{R}(\bar{0}) = h(\bar{0})$ or not. The modified scheme also remains secure in the ROM, because the probability that $\mathcal{R}(\bar{0}) = h(\bar{0})$ (taken over the randomness of $\mathcal{R}$) is $\frac{1}{2^n}$. However, once $\mathcal{R}$ is instantiated using h, all the forger has to do to learn pri — thereby completely breaking the scheme's security — is query his signature oracle on some string (say λ for concreteness).

The above approach can be readily adapted to yield a public-key encryption scheme which is secure in the ROM, but cannot be instantiated using h. To obtain such a scheme, simply take any public-key encryption scheme which is secure in the ROM (as noted in Chapter 1, in light of the results of [IR89] a hardness assumption of some sort will almost certainly be necessary here) and modify it as follows. Given a message $m \in \{0,1\}^n$, the new encryptor computes an encryption e of m as before and then checks whether $\mathcal{R}(\bar{0}) = h(\bar{0})$. If so, he outputs (m, e); otherwise, he outputs $(\bar{0}, e)$. Given a purported encryption α, the new decryptor first parses α as (β, γ), where $|\beta| = n$, then decrypts γ as before. Observe that our modification once again doesn't violate the correctness of the scheme, since every encryption output by the encryptor is correctly decrypted by the decryptor, whether $\mathcal{R}(\bar{0}) = h(\bar{0})$ or not. It is easy to see that the modified scheme remains secure in the ROM, but becomes completely insecure if $\mathcal{R}$ is instantiated using h.

[1]A subtle but important technical point to note here is that both Rompel's construction and his proof are of the "black-box" variety.

2 The first uninstantiability result

Canetti, Goldreich and Halevi first showed that uninstantiable signature and public-key encryption schemes exist in [CGH98]. Their key insight was that, for every hash function ensemble $\mathcal{H} = \{\mathcal{H}_n\}_{n\in\mathbb{N}}$, $\mathcal{H}_n = \{h_s : \{0,1\}^* \to \{0,1\}^n\}_{s\in\{0,1\}^n}$, there exists a binary relation $R^{\mathcal{H}} = \bigcup_{n\in\mathbb{N}}\{(s, h_s(s))\}_{s\in\{0,1\}^n}$ with the following two properties:

(i) There is a (deterministic) polynomial-time machine $M_{\mathcal{H}}$ which, given any $s \in \{0,1\}^n$, outputs an $x \in \{0,1\}^*$ such that $(x, h_s(x)) \in R^{\mathcal{H}}$.

(ii) For every probabilistic polynomial-time "finder" $F^{\mathcal{R}}$ who is given 1^n, the probability (taken over the random bits of $F^{\mathcal{R}}$ and the randomness of $\mathcal{R}$) that $F^{\mathcal{R}}$ outputs an $x \in \{0,1\}^*$ such that $(x, \mathcal{R}(x)) \in R^{\mathcal{H}}$ is negligible in n.

$R^{\mathcal{H}}$ obviously satisfies property (i), since $M_{\mathcal{H}}$ can simply output s itself as x. To see that $R^{\mathcal{H}}$ satisfies property (ii), observe that $(x, \mathcal{R}(x)) \in R^{\mathcal{H}} \Leftrightarrow \mathcal{R}(x) = h_x(x)$. $F^{\mathcal{R}}$'s success probability is therefore at most $\frac{q_{\mathcal{R}}}{2^n}$, where $q_{\mathcal{R}}$ is the (polynomially bounded) number of times he queries $\mathcal{R}$. Notice that $R^{\mathcal{H}}$ is also polynomial-time decidable in the following sense: to determine whether $(x, y) \in R^{\mathcal{H}}$, one need only compute $y' = h_x(x)$ (this can be done in polynomial time, because $\mathcal{H}$ is efficiently evaluable) and check if $y = y'$.

Given any hash function ensemble $\mathcal{H}$, we can use $R^{\mathcal{H}}$ to obtain a signature scheme which is secure in the ROM yet becomes insecure when the random oracle $\mathcal{R}$ is instantiated using $\mathcal{H}$. The idea is to take a signature scheme which is secure in the ROM (as pointed out in Section 1, such schemes exist unconditionally) and modify it as follows. Given a message $m \in \{0,1\}^*$, the new signer computes a signature σ of m as before and then checks whether $(m, \mathcal{R}(m)) \in R^{\mathcal{H}}$ (this can be done in polynomial time, since $R^{\mathcal{H}}$ is polynomial-time decidable). If so, he outputs (pri, σ); otherwise, he outputs $(\bar{0}, \sigma)$. Given a message m and a purported signature α of m, the new verifier parses α as (β, γ), where $|\beta| = n$, and then checks whether γ is a valid signature of m as before. Observe that our modification does not violate the correctness of the scheme, since every signature output by the signer is accepted by the verifier, whether $(m, \mathcal{R}(m)) \in R^{\mathcal{H}}$ or not. The modified scheme also remains secure in the ROM, since property (ii) above guarantees that any forger has only a negligible probability of finding an m such that $(m, \mathcal{R}(m)) \in R^{\mathcal{H}}$. However, once $\mathcal{R}$ is instantiated using $\mathcal{H}$, the forger, who is given s, need only query his signature oracle $\mathcal{S}$ on s to obtain pri.

We next use diagonalization to go from schemes which cannot be instantiated using some specific ensemble $\mathcal{H}$ to schemes which cannot be instantiated using *any* ensemble. Recall from Section 3 of Chapter 2 that every hash function ensemble $\mathcal{H}$ can be identified with its polynomial-time "evaluator" Turing machine $M_{\mathcal{H}}$. We can therefore effectively enumerate all hash function ensembles by enumerating all polynomial-time Turing machines and padding or truncating their output as necessary. Let $M_{\mathcal{U}}$ be the universal Turing machine doing the enumerating, and denote the corresponding "universal" ensemble by $\mathcal{U} = \{\mathcal{U}_n\}_{n \in \mathbb{N}}$. Since the running time of *every* polynomial-time machine cannot be upper-bounded by a *single* polynomial, $M_{\mathcal{U}}$ will need to run in "slightly super-polynomial" time, say $\mathcal{O}(n^{\log n})$ for concreteness. It is easy to see that when $\mathcal{U}$ is substituted for $\mathcal{H}$ in the above construction, the resulting signature scheme is uninstantiable. However, the signer no longer runs in polynomial time, since to determine whether $(m, \mathcal{R}(m)) \in R^{\mathcal{U}}$ he must effectively simulate $M_{\mathcal{U}}$.

Fortunately, the above difficulty can be overcome with the aid of Micali's non-interactive CS proofs ([Mic94], [Mic00]). Let $M'_{\mathcal{U}}$ be a decider for $R^{\mathcal{U}}$. Instead of running $M'_{\mathcal{U}}$ directly to determine whether $(m, \mathcal{R}(m)) \in R^{\mathcal{U}}$, the new signer parses m as (s, π) and checks if π is a valid CS proof that $M'_{\mathcal{U}}$ accepts $(s, \mathcal{R}(s))$ within $\mathcal{O}(n^{\log n})$ steps, where $n = |s| + |\mathcal{R}(s)|$. Since CS proofs can be verified very efficiently, this only takes polynomial time. In the ROM, the scheme remains secure because CS proofs are "computationally sound"[2], meaning that it is infeasible to find a valid proof of a false statement. However,

[2] Interestingly, it is not known whether non-interactive CS proofs are computationally sound in the "real world" under some reasonable complexity assumption.

once $\mathcal{R}$ is instantiated using some ensemble $\mathcal{H}$, the "perfect completeness" property of CS proofs guarantees that a forger can compute a valid π in polynomial time.

Just as in Section 1, the above approach can be readily adapted to yield an uninstantiable public-key encryption scheme.

3 A simple proof of the first result

In [MRH04], Maurer, Renner and Holenstein introduced a new type of reducibility, based on the concept of *indifferentiability*. To motivate their definitions, they gave a simple proof of the existence of uninstantiable signature and public-key encryption schemes. We present a further simplified version of their argument below.

To obtain an uninstantiable signature scheme, modify any signature scheme which is secure in the ROM (as pointed out in Section 1, such schemes exist unconditionally) as follows. Given a message $m \in \{0,1\}^*$, the new signer first computes a signature σ of m as before. He then parses m as $(\langle M \rangle, 1^t)$, where $\langle M \rangle$ describes a (deterministic) Turing machine M under some reasonable encoding, and simulates M on $\langle M \rangle$ for at most t steps. If M outputs $\mathcal{R}(\langle M \rangle)$, the signer outputs (pri, σ); otherwise, he outputs $(\bar{0}, \sigma)$. Given a message m and a purported signature α of m, the new verifier parses α as (β, γ), where $|\beta| = n$, and then checks whether γ is a valid signature of m as before. Observe that our modification does not violate the correctness of the scheme, since every signature output by the signer is accepted by the verifier, whether M outputs $\mathcal{R}(\langle M \rangle)$ within t steps or not. Also note that the new signer runs in polynomial time, since simulating M takes time $\mathcal{O}(t)$ and $t \leq |m|$.

To convince yourself that the modified scheme remains secure in the ROM, consider a function family $\mathcal{F} = \{f_t\}_{t \in \mathbb{N}}$ where each $f_t : \{0,1\}^* \to \{0,1\}^*$ is defined by

$$f_t(\langle M \rangle) = \begin{cases} M(\langle M \rangle) & \text{if } M \text{ halts within } t \text{ steps} \\ \lambda & \text{otherwise} \end{cases}$$

To learn pri using the "trapdoor" we have built into the scheme, a forger would effectively need to find a $t \in \mathbb{N}$ and an $x \in \{0,1\}^*$ such that $f_t(x) = \mathcal{R}(x)$. His probability of finding such a pair (t, x) is at most $\frac{q_\mathcal{R}}{2^n}$ (where $q_\mathcal{R}$ is the number of times he queries $\mathcal{R}$), which is negligible in n.

Once $\mathcal{R}$ is instantiated using a hash function ensemble $\mathcal{H}$, however, it becomes trivial to completely break the scheme's security. Recall that, because $\mathcal{H}$ is efficiently evaluable, there exists a (deterministic) polynomial-time Turing machine $M_\mathcal{H}$ such that $M_\mathcal{H}(s, x) = h_s(x)$ for all $s \in \{0,1\}^n$ and $x \in \{0,1\}^*$ (see Chapter 2, Section 3). Let M_{h_s} denote $M_\mathcal{H}$ with some particular s "hard-coded" into it, so that $M_{h_s}(x) = M_\mathcal{H}(s, x)$ for all $x \in \{0,1\}^*$, and suppose that n^c is an upper bound on the running time of M_{h_s}. When given input $\langle M_{h_s} \rangle$, M_{h_s} halts within n^c steps and outputs $h_s(\langle M_{h_s} \rangle)$. The forger, who is given s, need therefore only query his signature oracle on $(\langle M_{h_s} \rangle, 1^{n^c})$ to learn pri. ■

Just as in Section 1, the above approach can be readily adapted to yield an uninstantiable public-key encryption scheme.

4 An uninstantiability result for Fiat-Shamir signature schemes

The artificiality of [CGH98]'s constructions left open the possibility that "reasonable" signature schemes which are secure in the ROM, and in particular Fiat-Shamir signature schemes, can in fact be instantiated using appropriate hash function ensembles. However, in [GTK03] Goldwasser and Tauman-Kalai showed that there exist uninstantiable Fiat-Shamir signature schemes. It must be remarked that Goldwasser and Tauman-Kalai's construction is, if anything, even more contrived than those of [CGH98]. Barak and Goldreich's Universal Arguments ([BG02]) are used in place of Micali's CS proofs, and Merkle trees ([Mer90]) also make an appearance. Most distressingly, the proof itself has a highly non-constructive, tree-like structure: rather than demonstrate that a single (albeit unnatural) Fiat-Shamir scheme is uninstantiable, Goldwasser and Tauman-Kalai exhibit three such schemes, one of which must be uninstantiable. Nonetheless, from a purely theoretical standpoint, Goldwasser and Tauman-Kalai's result deals a severe blow to the validity of the so-called Fiat-Shamir paradigm (see Chapter 3, Section 1).

Chapter 6

A taste of "real-word" security

In the following two sections, we briefly survey a number of practical signature schemes and public-key encryption schemes which are secure in the "real world" (as opposed to in the ROM) under either standard or nonstandard-yet-quite-plausible hardness assumptions.

1 Signature schemes

- In [DN94], Dwork and Naor proposed a practical signature scheme which is secure — that is, secure against existential forgery under adaptive chosen-message attack (see Chapter 2, Section 5) — under the standard "RSA assumption". Informally, the RSA assumption says that the following problem is hard: given a modulus $n = pq$ where p and q are random primes, a random $y \in \mathcal{Z}_n^*$ and a random exponent e relatively prime to $(p-1)(q-1)$, find an $x \in \mathcal{Z}_n^*$ such that $x^e \equiv y \mod n$. Although it can be shown that this problem would be easy if p and q were given explicitly, it is not known whether factoring n can be reduced to finding x. While their construction is conceptually similar to the "authentification trees" of [GMR88], Dwork and Naor's use of "bushy trees" of high degree and small depth rather than binary trees significantly improves efficiency: for some reasonable settings of the security parameters, signing requires only four tree authentications. A significant drawback of their construction is that all signers and verifiers must share two lists, one consisting of random integers and the other of random primes.

- In [Cr96], Cramer and Damgård described an improved version of Dwork and Naor's signature scheme ([DN94]), secure under the same assumptions. In this new version, signers and verifiers need only share a single list consisting of random primes.

- In [GHR99], Gennaro, Halevi and Rabin presented a rather efficient "hash-and-invert" signature scheme secure under the nonstandard-yet-quite-plausible "strong RSA assumption". Informally, the strong RSA assumption says that the following problem is hard: given a modulus $n = pq$ where p and q are random primes and a random $y \in \mathcal{Z}_n^*$, find an $x \in \mathcal{Z}_n^*$ and an exponent $1 < e < n$ relatively prime to $(p-1)(q-1)$ such that $x^e \equiv y \mod n$; notice that, unlike in the RSA problem, here e is allowed to depend on y. Gennaro, Halevi and Rabin's scheme makes use of "collision-resistant chameleon hash functions", which exist if factoring is hard. For typical settings of the security parameters, it is more than twice as efficient as Cramer and Damgård's scheme ([Cr96]).

- In [CS99], Cramer and Shoup presented another efficient "hash-and-invert" signature scheme secure under the "strong RSA assumption". Their scheme builds on that of Cramer and Damgård ([Cr96]) and is considerably simpler and potentially more efficient than Gennaro, Halevi and Rabin's ([GHR99]). Instead of "collision-resistant chameleon hash functions", it makes use of "universal one-way hash functions" ([NY89]), which exist if one-way functions do. Interestingly, a slight modification of the scheme can be shown to be secure in the ROM under the ordinary RSA assumption.

- In [Fis03], Fischlin described an improved version of Cramer and Shoup's signature scheme ([CS99]), again secure under the "strong RSA assumption". Signing is about thirty percent faster in this new version, and verification is somewhat faster as well. Also, the length of the signatures is nearly halved.

2 Public-key encryption schemes

- In [CS98], Ronald Cramer and Victor Shoup proposed the first practical public-key encryption scheme which is secure — that is, secure against adaptive chosen-ciphertext attack or CCA2-secure (see Chapter 2, Section 7) — under a fairly standard hardness assumption, namely the "Decisional Diffie-Hellman assumption". Informally, the Decisional Diffie-Hellman assumption (often abbreviated as the DDH assumption) holds for a cyclic multiplicative group G of prime order q (say a subgroup of $\mathcal{Z}_p^*$, where $p > q$ is some prime) if, given $g^u \in G$ and $g^v \in G$ for randomly chosen $u, v \in \{1, \ldots, q\}$ (where $g \in G$ is some fixed generator of G), it is hard to distinguish $g^{uv} \in G$ from $g^r \in G$ for a randomly chosen $r \in \{1, \ldots, q\}$. While the Computational Diffie-Hellman or CDH assumption (see Chapter 4, Section 2) asserts that it is hard to compute *all of* g^{uv}, the DDH assumption effectively asserts that it is hard to compute *any bit of* g^{uv}.

- In [Sho00], Victor Shoup presented a "hybrid" encryption scheme which makes use of a "pseudorandom number generator", a collision-resistant hash function (see Chapter 2, Section 3) and a "key encapsulation scheme"; the latter is based on the Cramer-Shoup encryption scheme ([CS98]). A key encapsulation scheme is essentially just a public-key encryption scheme whose security is only guaranteed when the messages being encrypted are random (private keys, for example). The new scheme is somewhat more efficient than [CS98] and is secure under the fairly standard DDH assumption. Interestingly, it is also secure in the ROM under the standard CDH assumption.

- In [KD04], Kurosawa and Desmedt described a new hybrid encryption scheme based on [Sho00]. The scheme is somewhat more efficient (it saves one exponentiation and produces shorter encryptions) and is again secure under the DDH assumption. Kurosawa and Desmedt's key insight was to notice that the underlying key encapsulation scheme need not be CCA2-secure in order for the overall hybrid scheme to be CCA2-secure. However, their proof requires the additional assumption that both the "key derivation function" and the MAC used by the hybrid scheme are secure in a strong, information-theoretic sense. In particular, the key to be exchanged must be statistically close to random, precluding the use of pseudorandom number generators.

- In [GS04], Shoup and Gennaro used the technique of "deferred analysis" to demonstrate that Kurosawa and Desmedt's hybrid scheme ([KD04]) is in fact secure under the DDH assumption provided that both the "key derivation function" and the MAC are secure in the ordinary, computational sense.

Bibliography

[AABN02] Michel Abdalla, Jee Hea An, Mihir Bellare, and Chanathip Namprempre. From identification to signatures via the Fiat-Shamir transform: Minimizing assumptions for security and forward-security. In *Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques*, pages 418–433. Springer-Verlag, 2002.

[ABR01a] Michel Abdalla, Mihir Bellare, and Phillip Rogaway. DHIES: An encryption scheme based on the Diffie-Hellman problem. Available online at `http://www.cs.ucsd.edu/users/mihir/papers/dhies.html`, 2001.

[ABR01b] Michel Abdalla, Mihir Bellare, and Phillip Rogaway. The oracle Diffie-Hellman assumptions and an analysis of DHIES. In *Proceedings of Topics in Cryptology - CT-RSA 2001, The Cryptographer's Track at RSA Conference*, volume 2020 of *Lecture Notes in Computer Science*, pages 143–158. Springer-Verlag, 2001.

[BB04] Dan Boneh and Xavier Boyen. Short signatures without random oracles. In *Proceedings of Advances in Cryptology – EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques*, volume 3027 of *Lecture Notes in Computer Science*, pages 56–73. Springer-Verlag, 2004.

[BG02] Boaz Barak and Oded Goldreich. Universal arguments and their applications. In *Proceedings of the 17th IEEE Annual Conference on Computational Complexity—CCC '02*, pages 162–171. IEEE Computer Society, 2002.

[Ble98] Daniel Bleichenbacher. Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS #1. In *Proceedings of Advances in Cryptology–CRYPTO '98, 18th Annual International Cryptology Conference*, volume 1462 of *Lecture Notes in Computer Science*, pages 1–12. Springer-Verlag, 1998.

[BM88] Mihir Bellare and Silvio Micali. How to sign given any trapdoor function. In *Proceedings of STOC '88: Twentieth Annual ACM Symposium on Theory of computing*, pages 32–42. ACM Press, 1988.

[Bon99] Dan Boneh. Twenty years of attacks on the RSA cryptosystem. *Notices of the American Mathematical Society (AMS)*, 46(2):203–213, 1999.

[Bon01] Dan Boneh. Simplified OAEP for the RSA and Rabin functions. In *CRYPTO '01: Proceedings of the 21st Annual International Cryptology Conference on Advances in Cryptology*, volume 2139 of *Lecture Notes in Computer Science*, pages 275–291. Springer-Verlag, 2001.

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: a paradigm for designing efficient protocols. In *Proceedings of the 1st ACM conference on Computer and communications security—CCS '93*, pages 62–73. ACM Press, 1993.

[BR94] Mihir Bellare and Phillip Rogaway. Optimal assymetric encryption. In *Proceedings of Advances in Cryptology—EUROCRYPT '94, Workshop on the Theory and Application of Cryptographic Techniques*, volume 950 of *Lecture Notes in Computer Science*, pages 92–111. Springer-Verlag, 1994.

[BR97] Mihir Bellare and Phillip Rogaway. Minimizing the use of random oracles in authenticated encryption schemes. In *ICICS '97: International Conference on Information and Communications Security*, volume 1334 of *Lecture Notes in Computer Science*, pages 1–16. Springer-Verlag, 1997.

[Bri85] Ernest F. Brickell. Breaking iterated knapsacks. In *Proceedings of Advances in Cryptology–CRYPTO '84*, volume 196 of *Lecture Notes in Computer Science*, pages 342–358. Springer-Verlag, 1985.

[CGH98] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited (preliminary version). In *Proceedings of STOC '98: Thirtieth Annual ACM Symposium on Theory of Computing*, pages 209–218. ACM Press, 1998.

[Cr96] Ronald Cramer and Ivan Damgård. New generation of secure and practical RSA-based signatures. In *Proceedings of the 16th Annual International Cryptology Conference on Advances in Cryptology—CRYPTO '96*, volume 1109 of *Lecture Notes in Computer Science*, pages 173–185. Springer-Verlag, 1996.

[CS98] Ronald Cramer and Victor Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In *Proceedings of the 18th Annual International Cryptology Conference on Advances in Cryptology—CRYPTO '98*, volume 1462 of *Lecture Notes in Computer Science*, pages 13–25. Springer-Verlag, 1998.

[CS99] Ronald Cramer and Victor Shoup. Signature schemes based on the strong RSA assumption. In *Proceedings of the 6th ACM conference on Computer and Communications Cecurity—CCS '99*, pages 46–51. ACM Press, 1999.

[DN94] Cynthia Dwork and Moni Naor. An efficient existentially unforgeable signature scheme and its applications. In *CRYPTO '94: Proceedings of the*

14th Annual International Cryptology Conference on Advances in Cryptology, volume 839 of *Lecture Notes in Computer Science*, pages 234–246. Springer-Verlag, 1994.

[Fis03] Marc Fischlin. The cramer-shoup strong-RSA signature scheme revisited. In *Proceedings of Public Key Cryptography - PKC 2003, 6th International Workshop on Theory and Practice in Public Key Cryptography*, volume 2567 of *Lecture Notes in Computer Science*, pages 116–129. Springer-Verlag, 2003.

[FMR96] Michael J. Fischer, Silvio Micali, and Charles Rackoff. A secure protocol for the oblivious transfer (extended abstract). *Journal of Cryptology*, 9(3):191–195, 1996.

[FOPS01] Eiichiro Fujisaki, Tatsuaki Okamoto, David Pointcheval, and Jacques Stern. RSA-OAEP is secure under the RSA assumption. In *CRYPTO '01: Proceedings of the 21st Annual International Cryptology Conference on Advances in Cryptology*, volume 2139 of *Lecture Notes in Computer Science*, pages 260–274. Springer-Verlag, 2001.

[FOPS04] Eiichiro Fujisaki, Tatsuaki Okamoto, David Pointcheval, and Jacques Stern. RSA-OAEP is secure under the RSA assumption. *Journal of Cryptology*, 17(2):81–104, 2004.

[FS87] Amos Fiat and Adi Shamir. How to prove yourself: practical solutions to identification and signature problems. In *Proceedings of Advances in cryptology—CRYPTO '86*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194. Springer-Verlag, 1987.

[GaJ03] Eu-Jin Goh and Stanisław Jarecki. A signature scheme as secure as the Diffie-Hellman problem. In *Eurocrypt 2003: Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques'*, volume 2656 of *Lecture Notes in Computer Science*, pages 401–415. Springer-Verlag, 2003.

[GGM84a] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. In *Proceedings of FOCS '84: 25th Annual IEEE Symposium on Foundations of Computer Science*. IEEE Computer Society, 1984.

[GGM84b] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. On the cryptographic applications of random functions. In *Proceedings of CRYPTO '84: Advances in Cryptology*, volume 196 of *Lecture Notes in Computer Science*, pages 276–288. Springer-Verlag, 1984.

[GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *Journal of the ACM*, 33(4):792–807, 1986.

[GHR99] Rosario Gennaro, Shai Halevi, and Tal Rabin. Secure hash-and-sign signatures without the random oracle. In *Proceedings of Advances in Cryptology—EUROCRYPT'99: International Conference on the Theory and Application of Cryptographic Techniques*, volume 1592 of *Lecture Notes in Computer Science*, pages 123–139. Springer-Verlag, 1999.

[GM84] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984.

[GMR85] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof-systems. In *Proceedings of STOC '85: Seventeenth annual ACM Symposium on Theory of computing*, pages 291–304. ACM Press, 1985.

[GMR88] Shafi Goldwasser, Silvio Micali, and Ron L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, 1988.

[GS04] Rosario Gennaro and Victor Shoup. A note on an encryption scheme of Kurosawa and Desmedt. Unpublished manuscript, 2004. Available online at `http://shoup.net/papers/kdnote.pdf`.

[GTK03] Shafi Goldwasser and Yael Tauman Kalai. On the (in)security of the Fiat-Shamir paradigm. In *Proceedings of FOCS '03: 44th Annual IEEE Symposium on Foundations of Computer Science*, pages 102–113. IEEE Computer Society, 2003.

[HILL99] Johan Hastad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28(4):1364–1396, 1999.

[IEE00] IEEE. Standard specifications for public-key cryptography, 2000. Available online at `http://grouper.ieee.org/groups/1363/P1363/index.html`.

[IR89] Russel Impagliazzo and S. Rudich. Limits on the provable consequences of one-way permutations. In *Proceedings of STOC '89: Twenty-first Annual ACM Symposium on Theory of Computing*, pages 44–61. ACM Press, 1989.

[KD04] Kaoru Kurosawa and Yvo Desmedt. A new paradigm of hybrid encryption scheme. In *Proceedings of Advances in Cryptology – CRYPTO 2004: 24th Annual International Cryptology Conference*, volume 3152 of *Lecture Notes in Computer Science*, pages 426–442. Springer-Verlag, 2004.

[Len00] Arjen K. Lenstra. Integer factoring. *Designs, Codes and Cryptography*, 19(2/3):101–128, 2000.

[Mer90] Ralph C. Merkle. A certified digital signature: That antique paper from 1979. In *Proceedings of Advances in Cryptology—CRYPTO '89, 9th Annual International Cryptology Conference*, volume 435 of *Lecture Notes in Computer Science*, pages 218–238. Springer-Verlag, 1990.

[Mic94] Silvio Micali. CS proofs. In *Proceedings of FOCS '94: 35th Annual IEEE Symposium on Foundations of Computer Science*, pages 436–453. IEEE Computer Society, 1994.

[Mic00] Silvio Micali. Computationally sound proofs. *SIAM Journal on Computing*, 30(4):1253–1298, 2000.

[MRH04] Ueli M. Maurer, Renato Renner, and Clemens Holenstein. Indifferentiability, impossibility results on reductions, and applications to the random oracle methodology. In *Proceedings of TCC '04: Theory of Cryptography, First Theory of Cryptography Conference*, volume 2951 of *Lecture Notes in Computer Science*, pages 21–39. Springer-Verlag, 2004.

[MS90] Silvio Micali and Adi Shamir. An improvement of the Fiat-Shamir identification and signature scheme. In *Proceedings of Advances in Cryptology - CRYPTO '88, 8th Annual International Cryptology Conference*, volume 403 of *Lecture Notes in Computer Science*, pages 244–247. Springer-Verlag, 1990.

[MW00] Ueli M. Maurer and Stefan Wolf. The Diffie-Hellman protocol. *Designs, Codes and Cryptography*, 19(2/3):147–171, 2000.

[NIS99] NIST. FIPS 46-3, Data Encryption Standard (DES), 1999. Available online at `http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf`.

[NIS01] NIST. FIPS 197, Advanced Encryption Standard (AES), 2001. Available online at `http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf`.

[NIS04] NIST. FIPS 180-2, Secure Hash Standard (SHS), 2004. Available online at `http://csrc.nist.gov/publications/fips/fips180-2/fips180-2withchangenotice.pdf`; also see `http://csrc.nist.gov/hash_standards_comments.pdf`.

[NY89] M. Naor and M. Yung. Universal one-way hash functions and their cryptographic applications. In *Proceedings of STOC '89: Twenty-first Annual ACM Symposium on Theory of Computing*, pages 33–43. ACM Press, 1989.

[NY90] M. Naor and M. Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *Proceedings of STOC '90: Twenty-second Annual ACM Symposium on Theory of Computing*, pages 427–437. ACM Press, 1990.

[Odl00] Andrew M. Odlyzko. Discrete logarithms: The past and the future. *Designs, Codes and Cryptography*, 19(2/3):129–145, 2000.

[Oka93] Tatsuaki Okamoto. Provably secure and practical identification schemes and corresponding signature schemes. In *Proceedings of the 12th Annual International Cryptology Conference on Advances in Cryptology—CRYPTO '92*, volume 740 of *Lecture Notes in Computer Science*, pages 31–53. Springer-Verlag, 1993.

[Rom90] J. Rompel. One-way functions are necessary and sufficient for secure signatures. In *Proceedings of STOC '90: Twenty-second Annual ACM Symposium on Theory of Computing*, pages 387–394. ACM Press, 1990.

[RS92] Charles Rackoff and Daniel R. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In *Proceedings of the 11th Annual International Cryptology Conference on Advances in Cryptology—CRYPTO '91*, volume 576 of *Lecture Notes in Computer Science*, pages 433–444. Springer-Verlag, 1992.

[RSA93] RSA Security Inc. PKCS#1 RSA cryptography standard, version 2.0, 1993. Available online at `ftp://ftp.rsasecurity.com/pub/pkcs/ps/pkcs-1.ps`.

[RSA98] RSA Security Inc. PKCS#1 RSA encryption standard, version 1.5, 1998. Available online at `ftp://ftp.rsasecurity.com/pub/pkcs/ascii/pkcs-1v2.asc`.

[Sho96] Victor Shoup. On the security of a practical identification scheme. In *Proceedings of Advances in Cryptology - EUROCRYPT '96, International Conference on the Theory and Application of Cryptographic Techniques*, volume 1070 of *Lecture Notes in Computer Science*, pages 344–353. Springer-Verlag, 1996.

[Sho00] Victor Shoup. Using hash functions as a hedge against chosen ciphertext attack. In *EUROCRYPT '00–Proceedings of the International Conference on the Theory and Application of Cryptographic Techniques*, volume 1807 of *Lecture Notes in Computer Science*, pages 275–288. Springer-Verlag, 2000.

[Sho01] Victor Shoup. OAEP reconsidered. In *CRYPTO '01: Proceedings of the 21st Annual International Cryptology Conference on Advances in Cryptology*, volume 2139 of *Lecture Notes in Computer Science*, pages 239–259. Springer-Verlag, 2001.