Evaluation Report on the **RSA–OAEP** encryption scheme

Jacques Stern

1 Introduction

This document is an evaluation of the RSA–OAEP encryption scheme, standardized in PKCS # 1 v.2.0 (see [15]). Our work is based on the analysis of documents [15, 24, 25], which provide the specification of the scheme, as well as on various research papers related to each scheme, where security arguments can be found. Among the reseach papers related to the encryption scheme are the seminal work [5] of Bellare and Rogaway on OAEP, and the more recent work by Shoup [27] on one hand, and Fujisaki, Okamoto, Pointcheval and Stern [11] on the other hand.

The present report is organized as follows: firstly, we briefly review the RSA primitive, mainly for notational purposes, and we discuss its relation to the factoring problem. We also recall the strong security notions that are now mandatory for cryptosystems: semantic security and security against adaptive chosen-ciphertext attacks, in the case of encryption. This allows us to discuss the security of initial instantiations of the RSA cryptosystems. Next, we turn to OAEP and provide a complete proof of the security of RSA–OAEP against adaptive chosen-ciphertext attacks, in the random oracle model. We also discuss the practical implications of this proof in terms of key sizes and implementations conformant to PKCS #1 [15, 24]. This is as requested by IPA.

2 The RSA primitive and its security

In this section, we review the original RSA primitive, describe the strong security notions that are currently required, and explain why "plain" RSA cannot meet these requirements.

2.1 The RSA cryptosystem

In modern terms, a public-key encryption scheme on a message space $\mathcal{M}$ consists of three algorithms $(\mathcal{K}, \mathcal{E}, \mathcal{D})$:

- the key generation algorithm $\mathcal{K}(1^k)$ outputs a random pair of private/public keys $(\mathsf{sk}, \mathsf{pk})$, relatively to a security parameter k
- the encryption algorithm $\mathcal{E}_{\mathsf{pk}}(m; r)$ outputs a ciphertext c corresponding to the plaintext $m \in \mathcal{M}$, using random coins r
- the decryption algorithm $\mathcal{D}_{\mathsf{sk}}(c)$ outputs the plaintext m associated to the ciphertext c.

We will occasionnally omit the random coins and write $\mathcal{E}_{\mathsf{pk}}(m)$ in place of $\mathcal{E}_{\mathsf{pk}}(m; r)$. Note that the decryption algorithm is deterministic.

The famous RSA cryptosystem has been proposed by Rivest, Shamir and Adleman [22]. The key generation algorithm of RSA chooses two large primes p, q of equal size and issues the so-called modulus $n = pq$. The sizes of p, q are set in such a way that the binary length $|n|$ of n equals the security parameter k. Additionally, en exponent e, relatively prime to $\varphi(n) = (p-1)(q-1)$ is chosen, so that the public key is the pair (n, e). The private key d is the inverse of e modulo $\varphi(n)$. Variants allow the use of more than two prime factors.

Encryption and decryption are defined as follows:

$$\mathcal{E}_{n,e}(m) = m^e \bmod n \qquad \mathcal{D}_{n,d}(c) = c^d \bmod n.$$

Note that both operations are deterministic and are mutually inverse to each other. Thus, the RSA encryption function is a permutation. It is termed a *trapdoor permutation* since decryption can only be applied given the private key.

The basic security assumption on which the RSA cryptosystem relies is its *one-wayness* (OW): using only public data, an attacker cannot recover the plaintext corresponding to a given ciphertext. In the general formal setting provided above, an encryption scheme is one-way if the success probability of any adversary $\mathcal{A}$ attempting to invert $\mathcal{E}$ (without the help of the private key), is negligible, i.e. asymptotically smaller than the inverse of any polynomial function of the security parameter. Probabilities are taken over the message space $\mathcal{M}$ and the randoin coins Ω. These include both the random coins r used for the encryption scheme, and the internal random coins of the adversary. In symbols:

$$\mathsf{Succ}^{\mathsf{ow}}(\mathcal{A}) = \Pr[(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathcal{K}(1^k), m \in_R \mathcal{M} : \mathcal{A}(\mathsf{pk}, \mathcal{E}_{\mathsf{pk}}(m)) = m].$$

Clearly, the factorization of n allows to invert the RSA encryption function, since d can be computed from p and q. It is unknown whether the converse is true, i.e. whether factoring and inverting RSA are computationnally equivalent. There are indications that it might not be true (see [7]). Thus, the assumption that RSA is one-way might be a stronger assumption than the hardness of factoring. Still, it is a widely believed assumption and the only method to assess the strength of RSA is to check whether the size of the modulus n outreaches the current performances of the various factoring algorithms.

2.2 Advanced security notions

2.2.1 Semantic security

Semantic security, also called *polynomial security/indistinguishability of encryptions*, has been introduced by Goldwasser and Micali [12] : an encryption scheme is *semantically secure* if no polynomial-time attacker can learn any bit of information about the plaintext from the ciphertext, except its length. More formally, an encryption scheme is semantically secure if, for any two-stage adversary $\mathcal{A} = (A_1, A_2)$ running in polynomial time, the advantage $\mathsf{Adv}^{\mathsf{ind}}(\mathcal{A})$ is negligible, where $\mathsf{Adv}^{\mathsf{ind}}(\mathcal{A})$ is formally defined as

$$2 \times \Pr\left[\begin{array}{l}(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathcal{K}(1^k), (m_0, m_1, s) \leftarrow A_1(\mathsf{pk}), \\ b \in_R \{0,1\}, c = \mathcal{E}_{\mathsf{pk}}(m_b) : A_2(m_0, m_1, s, c) = b\end{array}\right] - 1,$$

where the probability space includes the internal random coins of the adversary, and m_0, m_1 are two equal length plaintexts chosen by A_1 in the message-space $\mathcal{M}$.

2.2.2 Non-malleability

Another security notion has been defined in the literature [9], called *non-malleability* (NM). Informally is states that it is impossible to derive, from a given ciphertext, a new ciphertext such that the plaintexts are meaningfully related. We won't dicuss this notion extensively since it has been proven equivalent to semantic security in an extended attack model (see below).

2.2.3 Chosen-ciphertext security

The above definition of semantic security covers passive adversaries. It is a *chosen–plaintext* or CPA attack since the attacker can only encrypt plaintext. In extended models, the adversary is given restricted or non restricted access to various oracles. A *plaintext-checking* oracle receives as its input a pair (m, c) and answers whether c encrypts message m. This gives rise to *plaintext-checking attacks* [20]. A *validity-checking* oracle answers whether its input c is a valid ciphertext or not. The corresponding scenario has been termed *reaction attack* [13]. Its has been spectacularly applied for breaking the famous PKCS #1 v1.5 encryption [6]. Finally, a decryption oracle returns the decryption of any ciphertext c, with the only restriction that it should be different from the challenge ciphertext. When access to the decryption oracle is only granted to A_1, i.e. during the first stage of the attack, before the challenge ciphertext is received, the corresponding scenario is named *indifferent chosen-ciphertext attack* (CCA1) [17]. When the attacker also receives access to the decryption oracle in the second stage, the attack is termed the *adaptive chosen-ciphertext attack* (CCA2) [21]. The security notions defined above and their logical relationships have been discussed

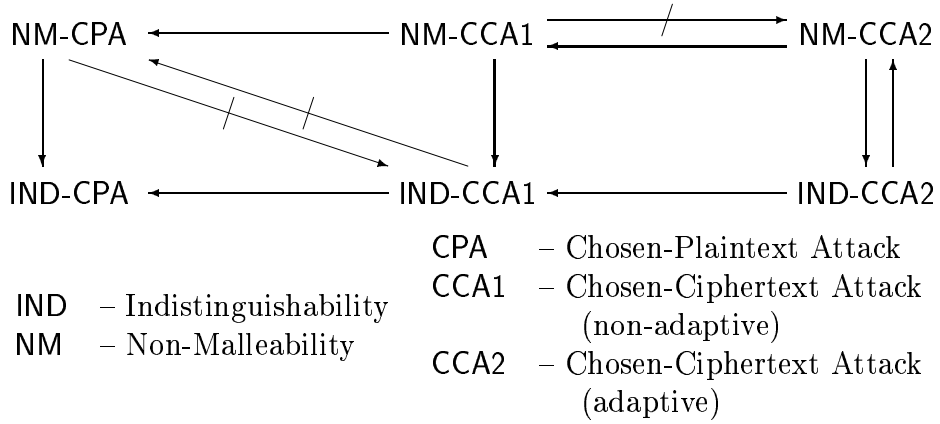

Figure 1: Relations between security notions

at length in [3]. The main results are summarized in the well-known diagram shown on figure 1.

Thus, under CCA2, semantic security and non-malleability are equivalent. This is the strongest security notion currently considered. We restate the definition in a more formal manner: any adversary $\mathcal{A}$ with unrestricted access to the decryption oracle $\mathcal{D}_{\mathsf{sk}}$, has negligible advantage, where the advantage is:

$$\mathsf{Adv}^{\mathsf{ind}}(\mathcal{A}^{\mathcal{D}_{\mathsf{sk}}}) = 2 \times \Pr\left[\begin{array}{l} (\mathsf{pk},\mathsf{sk}) \leftarrow \mathcal{K}(1^k), (m_0, m_1, s) \leftarrow A_1^{\mathcal{D}_{\mathsf{sk}}}(\mathsf{pk}), \\ b \in_R \{0,1\}, c = \mathcal{E}_{\mathsf{pk}}(m_b) : A_2^{\mathcal{D}_{\mathsf{sk}}}(m_0, m_1, s, c) = b \end{array} \right] - 1,$$

2.3 The security of "plain" RSA

By itself, RSA cannot provide a secure encryption scheme for any of the security notions considered in the previous section: semantic security fails because encryption is deterministic and non-malleability cannot hold due to the homomorphic property:

$$\mathcal{E}_{n,e}(m_1) \cdot \mathcal{E}_{n,e}(m_2) = \mathcal{E}_{n,e}(m_1 m_2 \bmod n) \bmod n.$$

Therefore, any RSA-based cryptosystems has to use a padding or encoding method before applying the RSA primitive.

For the sake of completeness, we review the first encoding rule, originally proposed in the PKCS #1 v1.5 standard [23]. Given a modulus n, let k be its size in bytes, $2^{8(k-1)} \leq n < 2^{8k}$. This is a slight twist to our original definition of the security parameter. The standard allows to encrypt an ℓ byte-long message m, for $\ell \leq k - 11$. To perform the encryption, one randomly chooses a $k - 3 - \ell$ byte-long random string r, whose bytes are all non-zero and defines $M = \mathtt{02} \| r \| \mathtt{0} \| m$, a $k - 1$ byte string (see figure 2). String M is turned into an integer $< n$ by standard conversion routines and

0	2	more than 8 bytes non-zero bytes	0	m

Figure 2: PKCS #1 v1.5 Format

thereafter encrypted by means of the RSA function, $C = M^e \bmod n$. When decrypting a ciphertext C, one first applies RSA inversion, $M = C^d \bmod n$, and checks whether the result M matches with the expected format $\mathtt{02}\| * \|\mathtt{0}\| * $. In the positive situation, one outputs the trailing part as the plaintext. Otherwise, the ciphertext is rejected.

The encoding rule seemed to address known weaknesses of plain RSA. However, in 1998, Bleichenbacher [6] described a reaction attack breaking the one-wayness of the underlying RSA scheme. Furthermore, this attack was (almost) realistic against practical implementations of the SSL protocol v3.0. Based on the answer that an SSL server outputs upon receiving an invalid ciphertext, an adversary can select a ciphertext C, such that

$$2 \cdot 2^{8(k-2)} \leq C^d \bmod n < 3 \cdot 2^{8(k-2)}.$$

Furthermore, based on the homomorphic properties of the RSA function, this ciphertext can be related to a target plaintext. Bleichenbacher showed how to tighten the bound by iteration, which leads to recovering the plaintext after a number of queries (a few millions with a 1024-bit modulus).

We will not review Bleichenbacher's attack but, in order to get its flavour, we will assume that the adversary has access to an oracle, which, upon receiving a ciphertext C, returns the answer to the test $C^d \bmod n \stackrel{?}{<} B$, for some fixed bound B. Understanding how the attack works in this simplified setting will be useful at a later point in this report. We assume that n/B is > 2 but still small. Given the target ciphertext, the attacker submits ciphertexts of the form $C(\alpha) = \alpha^e C \bmod n$. If x is obtained from C by inverting the RSA function, $x = C^d \bmod n$, then inverting $C(\alpha)$ yields αx, by the homomorphic properties of the RSA function. The simplified attacks successfully finds

- an integer α such that $B/2 \leq \alpha x < B$
- an integer α_0 such that $n \leq \alpha_0 x < n + B$
- a sequence of integers α_i such that $2^i n \leq \alpha_i x < 2^i n + B$

Observe that α can easily be found by trials and errors. From α, one can find α_0 by testing $C(j\alpha)$, $j > 2$ by means of the oracle, until the test becomes positive. Finally, to get α_{i+1} from α_i, one tests $C(2\alpha_i - j\alpha)$ until a positive answer is returned. When i is large enough, the exact value of x can be found as the integer closest to $\frac{n}{2^i}$.

2.4 The random oracle model

After the attack just described appeared, it became clear that one should provide an encoding rule, such that the resulting encryption scheme has provable security. Ideally, one would like to establish the security of the scheme based on the sole assumption that the RSA function is one-way. Unfortunately, no encoding is currently known that allows such a proof.

Thus, the best one can hope for is a proof carried in a non-standard computational model, as proposed by Bellare and Rogaway [4], following an earlier suggestion by Fiat and Shamir [10]. In this model, called the random oracle model, concrete objects such that hash functions are treated as random objects. This allows to carry through the usual reduction arguments to the context of relativized computations, where the hash function is treated as an oracle returning a random answer for each new query. A reduction still uses an adversary as a subroutine of a program that contradicts a mathematical assumption, such as the assumption that RSA is one-way. However, probabilities are taken not only over coin tosses but also over the random oracle.

Of course, the significance of proofs carried in the random oracle is debatable. Firstly, hash functions are deterministic and therefore do not return random answers. Along those lines, Canetti *et al.* [8] gave an example of a signature scheme which is secure in the random oracle model, but insecure under any instantiation of the random oracle. Secondly, proofs in the random oracle model cannot easily receive a quantitative interpretation. One would like to derive concrete estimates - in terms of key sizes - from the proof: if a reduction is efficient, the security "loss" is small and the existence of an efficient adversary leads to an algorithm for solving the underlying mathematical problem, which is almost as efficient. Thus, key sizes that outreach the performances of the known algorithms to break the underlying problem can be used for the scheme as well.

Despite these restrictions, the random oracle model has proved extremely useful to analyze many encryption and signature schemes. It clearly provides an overall guarantee that a scheme is not flawed, based on the intuition that an attacker would be forced to use the hash function in a non generic way.

2.5 Plaintext-awareness

A further notion that has been defined in the literature and has been the source of potential misconceptions is *plaintext-awareness*. It was introduced by Bellare and Rogaway [5] to formally state the impossibility of creating a valid ciphertext without "knowing" the corresponding plaintext. This goes through the definition of a *plaintext-extractor* $\mathcal{PE}$. Such a definition only makes sense in the random oracle model: in this model, one can store the query/answer list $\mathcal{H}$ that an adversary $\mathcal{A}$ obtains while interacting with the oracle H. Basically, the plaintext-extractor $\mathcal{PE}$ is able to correctly

simulate the decryption algorithm, without the private key, when it receives a candidate ciphertext y produced by any adversary $\mathcal{A}$, together with the list $\mathcal{H}$ produced during the execution of $\mathcal{A}$. In other words, given y and $\mathcal{H}$, the plaintext-extractor $\mathcal{PE}$ outputs the plaintext (or the "Reject" answer), with overwhelming success probability, where probabilities are taken over the random coins of $\mathcal{A}$ and $\mathcal{PE}$. In symbols:

$$\mathsf{Succ}^{\mathsf{wpa}}(\mathcal{PE}) = \Pr\left[(\mathsf{pk},\mathsf{sk}) \leftarrow \mathcal{K}(1^k), (y,\mathcal{H}) \leftarrow \mathsf{Exec}^{\mathcal{A}}(\mathsf{pk}) : \mathcal{PE}(y,\mathcal{H}) = \mathcal{D}_{\mathsf{sk}}(y)\right].$$

The wpa superscript in the above relates to the name *weak plaintext-awareness* (WPA or PA94), that the notion has later received. Actually, it is not an appropriate definition for practical applications, since, in many scenarios, the adversary may have access to additional valid ciphertexts that it has not manufactured - say by eavesdropping.

Accordingly, the definition was modified in [3], to give the adversary $\mathcal{A}$ access to an encryption oracle outputting valid ciphertexts. We denote by C the list of ciphertexts obtained by the adversary from the encryption oracle. Since the adversary is given access to additional resources, the new notion is stronger: the adversary outputs a fresh ciphertext y (not in C), this ciphertext is given to the plaintext-extractor, together with the lists $\mathcal{H}$ and C. Based on these data, $\mathcal{PE}$ outputs the plaintext (or the "Reject" answer) with overwhelming success probability $\mathsf{Succ}^{\mathsf{pa}}(\mathcal{PE})$, where:

$$\mathsf{Succ}^{\mathsf{pa}}(\mathcal{PE}) = \Pr\left[(\mathsf{pk},\mathsf{sk}) \leftarrow \mathcal{K}(1^k), (y,C,\mathcal{H}) \leftarrow \mathsf{Exec}^{\mathcal{A}^{\mathcal{E}_{\mathsf{pk}}}}(\mathsf{pk}) : \mathcal{PE}(y,C,\mathcal{H}) = \mathcal{D}_{\mathsf{sk}}(y)\right].$$

It is of course important to note that $y \notin C$. In other words, y has been duly manufactured by the attacker and not obtained from the encryption oracle.

The new definition of *plaintext-awareness* (PA or PA98) allows to reach the strongest security level, IND-CCA2. Indeed, it is easily seen that the combination of IND-CPA and PA yields IND-CCA2, whereas the combination of IND-CPA and WPA only yields IND-CCA1. As we will see later, this does not even imply NM-CPA.

3 Optimal Asymmetric Encryption Padding

We now turn to OAEP (Optimal Asymmetric Encryption Padding). We first review, in a non technical manner, the various research contributions that this scheme has fostered since it was first published in [5]. Next, we give a mathematical description of OAEP and provide a complete proof of its security. This proof is different in spirit from what is published in [11] and is therefore of independent interest. Next, we discuss the version of OAEP that appears in PKCS#1 v2.0 [15] and discuss the meaning of the security proof in this setting. Finally, we discuss potential implementation errors and how they have been addressed by RSA Security Inc.

3.1 The history of OAEP

When Bleichenbacher published his attack on RSA–PKCS #1 v1.5 [6], it became clear that a scheme secure against chosen-ciphertext attacks was needed to securely practice RSA. At that time, the only choice was to use the Optimal Asymmetric Encryption Padding, proposed by Bellare and Rogaway [5]. In their paper, Bellare and Rogaway proved that, by applying any trapdoor one-way permutation to the output of OAEP encoding, one could obtain a encryption scheme which was both semantically secure and plaintext-aware. Of course, they were using the *weak* version of plaintext awareness, where the plaintext-extractor is not given additional valid ciphertexts received by the adversary. This precisely excludes the adaptive chosen-ciphertext attack scenario, where the adversary has access to the decryption oracle, even after receiving the challenge ciphertext. This challenge is a valid ciphertext, that should be handled to the plaintext extractor to perform the decryption simulation. In other words, what follows from the combination of semantic security and *weak plaintext-awareness* in [5], is only semantic security against indifferent chosen-ciphertext attacks (IND-CCA1).

After Bellare *et al.* [3] later published the modified definition of plaintext-awareness (PA), no research paper claimed that one could design a plaintext-extractor for OAEP, according to the new definition. Still, the original paper was commonly viewed as providing a strong security level. This view was definitely superficial: it had never been proven that OAEP was turning a trapdoor one-way permutation into an IND-CCA2 encryption scheme. In fact, Victor Shoup [27] very ingeniously demonstrated that it was very unlikely that such a proof could exist at all. Shortly after the announcement of Shoup's results, Fujisaki, Okamoto, Pointcheval and Stern [11], proved that RSA–OAEP was indeed IND-CCA2 secure. Quite logically, since the general result does not hold, they had to use specific properties of the RSA primitive: details will be given below.

3.2 Description of OAEP

Let f be a k-bit to k-bit trapdoor permutation, whose inverse is denoted by g. Let k_0, k_1 be two parameters such that $k_0 + k_1 < k$. We define $n = k - k_0 - k_1$ and fix two hash functions, G and H, such that:

$$G : \{0,1\}^{k_0} \longrightarrow \{0,1\}^{k-k_0} \text{ and } H : \{0,1\}^{k-k_0} \longrightarrow \{0,1\}^{k_0}.$$

Given the above, we can define an OAEP-encryption scheme $(\mathcal{K}, \mathcal{E}, \mathcal{D})$ as follows:

- $\mathcal{K}(1^k)$: specifies an instance of the function f, and of its inverse g. The public key pk is f and the private key sk is g.
- $\mathcal{E}_{\mathsf{pk}}(m; r)$: given a message $m \in \{0,1\}^n$, and a random value $r \in_R \{0,1\}^{k_0}$, the

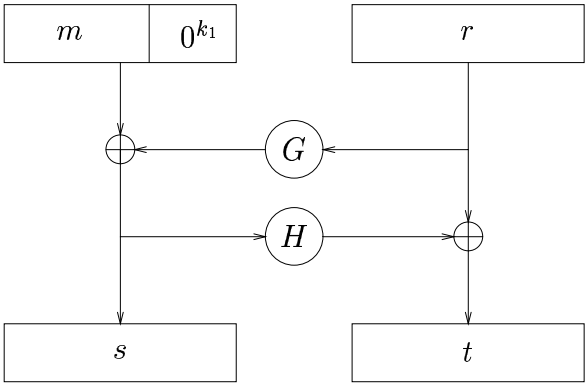

Figure 3: Optimal Asymmetric Encryption Padding

encryption algorithm $\mathcal{E}_{\mathsf{pk}}$ computes

$$s = (m\|0^{k_1}) \oplus G(r) \text{ and } t = r \oplus H(s),$$

and outputs the ciphertext $c = f(s\|t)$.

- $\mathcal{D}_{\mathsf{sk}}(c)$: using the private key $\mathsf{sk} = g = f^{-1}$, the decryption algorithm $\mathcal{D}_{\mathsf{sk}}$ extracts

$$(s\|t) = g(c), \text{ and next } r = t \oplus H(s) \text{ and } M = s \oplus G(r).$$

If $[M]_{k_1} = 0^{k_1}$, the algorithm returns $[M]^n$, otherwise it returns "Reject".

In the above, $[M]_{k_1}$ denotes the k_1 trailing bits of M, while $[M]^n$ denotes the n leading bits of M. The scheme is depicted on figure 3 .

As repeatedly stated, paper [5] includes a proof that, provided f is a one-way trapdoor permutation, the resulting OAEP encryption scheme is both semantically secure and weakly plaintext-aware. This implies the semantic security against indifferent chosen-ciphertext attacks (IND-CCA1), also called security against lunchtime attacks. We will not review the proofs from [5], since a full proof of the security of RSA-OAEP appears further in this report. We briefly comment on the intuition behind (weak) plaintext-awareness. When, the plaintext-extractor receives a ciphertext c, then:

- either s has been queried to H and r has been queried to G, in which case the extractor finds the cleartext by inspecting the two query lists $\mathcal{G}$ and $\mathcal{H}$,
- or else the decryption of (s, t) remains highly random and there is little chance to meet the redundancy 0^{k_1}: the plaintext extractor can safely declare the ciphertext invalid.

The argument collapses when the plaintext-extractor receives additional valid ciphertexts y, since this puts additional implicit constraints on G and H. These constraints cannot be seen by inspecting the query lists.

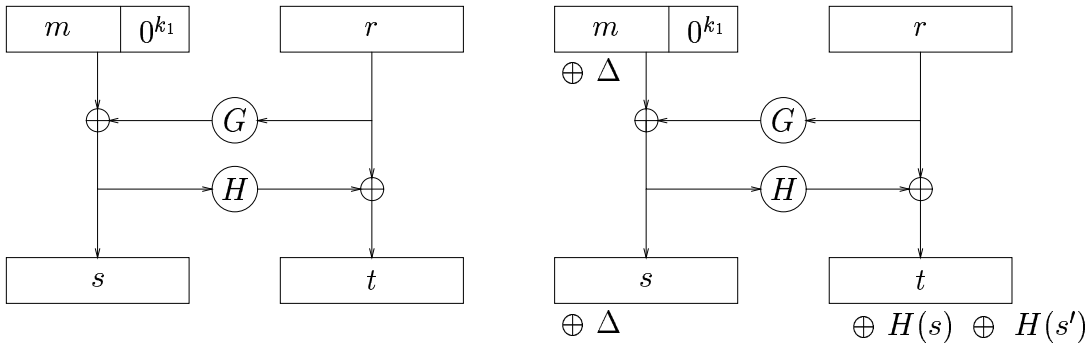

Figure 4: Shoup's Attack

3.3 Shoup's counter-example

In his paper [27], Victor Shoup showed that it was quite unlikely to extend the results of [5] so as to obtain adaptive chosen-ciphertext security, under the sole one-wayness of the permutation. His counter-example made use of the *ad hoc* notion of a *XOR-malleable* trapdoor one-way permutation: for such permutation f_0, one can compute $f_0(x \oplus a)$ from $f_0(x)$ and a, with non-negligible probability.

Let f_0 be such a XOR-malleable permutation. Defines f by $f(s\|t) = s\|f_0(t)$. Clearly, f is also a trapdoor one-way permutation. However it leads to a malleable encryption scheme as we now show. Start with a challenge ciphertext $y = g(s\|t) = s\|u$, where $s\|t$ is the output of the OAEP transformation on the redundant message $m\|0^{k_1}$ and the random string r (see figure 4)

$$s = G(r) \oplus (m\|0^{k_1}), t = H(s) \oplus r \text{ and } u = f_0(t).$$

Since f is the identity on its leftmost part, we know s, and can define $\Delta = \delta\|0^{k_1}$, for any random string δ, and $s' = s \oplus \Delta$. We then set $t' = H(s') \oplus r = t \oplus (H(s) \oplus H(s'))$. The XOR-malleability of f_0 allows to obtain $u' = f_0(t')$ from $u = f_0(t)$ and $H(s) \oplus H(s')$, with significant probability. Finally, $y' = s'\|u'$ is a valid ciphertext of $m' = m \oplus \delta$, built from $r' = r$, since:

$$t' = f_0^{-1}(u') = t \oplus (H(s) \oplus H(s')) = H(s') \oplus r \text{ and } r' = H(s') \oplus t' = r.$$

and

$$s' \oplus G(r') = \Delta \oplus s \oplus G(r) = \Delta \oplus (m\|0^{k_1}) = (m \oplus \delta)\|0^{k_1}.$$

Note that the above definitely contradicts adaptive chosen-ciphertext security: asking the decryption of y' after having received the ciphertext y, an adversary obtains m' and easily recovers the actual cleartext m from m' and δ. Also note that Shoup's counter-example exactly stems from where the intuition developped at the end of the

previous section failed: a valid ciphertext y' was created without quering the oracle at the corresponding random seed r', using in place the implicit constraint on G coming from the received valid ciphertext y.

Using methods from relativized complexity theory, Shoup [27] built a non-standard model of computation, where there exists a XOR-malleable trapdoor one-way permutation. As a consequence, it is very unlikely that one can prove the IND-CCA2 security of the OAEP construction, under the sole one-wayness of the underlying permutation. Indeed, all methods of proof currently known still apply in relativized models of computation.

3.4 The security of RSA–OAEP: outline

3.4.1 The intuition

Referring to our description of the intuition behind the original OAEP proof of security, given in section 3.2, we can carry a more subtle analysis by distinguishing the case where s has not been queried from oracle H from the case where r has not been queried from G. If s is not queried, then $H(s)$ is random and uniformly distributed and r is necessarily defined as $t \oplus H(s)$. This holds even if s matches with the string $s^\star$ coming from the valid ciphertext $y^\star$. There is a minute probability that $t \oplus H(s)$ is queried from G or equals $r^\star$. Thus, $G(r)$ is random: there is little chance that the redundancy 0^{k_1} is met and the extractor can safely reject.

We claim that r cannot match with $r^\star$, unless $s^\star$ is queried from H. This is because $r^\star = t^\star \oplus H(s^\star)$ equals $r = t \oplus H(s)$ with minute probability. Thus, if r is not queried, then $G(r)$ is random and we similarly infer that the extractor can safely reject. The argument only fails only if $s^\star$ is queried.

Thus rejecting when it cannot combine elements of the lists $\mathcal{G}$ and $\mathcal{H}$ so as to build a preimage of y, the plaintext extractor is only wrong with minute probability, unless $s^\star$ has been queried by the adversary. This seems to show that OAEP leads to an IND-CCA2 encryption scheme if it is difficult to "partially" invert f, which means: given $y = f(s\|t)$, find s.

3.4.2 The strategy

Based on the intuition just described, Fujisaki, Okamoto, Pointcheval and Stern [11] formally proved that applying OAEP encoding to a trapdoor permutation which is difficult to partially invert, leads to an IND-CCA2 encryption scheme. They used the term *partial-domain one-wayness* to express the fact that the above partial inversion problem was difficult. Precise definitions will be given in the next section. As the original proof from [5], their proof has two steps: it is first shown that the OAEP scheme is IND-CPA relative to another notion termed *set partial-domain one-wayness*.

Next, chosen-ciphertext security is addressed, by turning the intuition explained above, into a formal argument, involving a restricted variant of plaintext awareness. The authors of [11] also proved that the partial-domain one-wayness of the RSA function is equivalent to its one-wayness. Altogether, the expected security result for RSA–OAEP is finally proved.

3.5 The security of RSA–OAEP: the formal proof

3.5.1 Partial-domain one-wayness

Let f be, as above, a permutation $f : \{0,1\}^k \longrightarrow \{0,1\}^k$, which can also be written as

$$f : \{0,1\}^{n+k_1} \times \{0,1\}^{k_0} \longrightarrow \{0,1\}^{n+k_1} \times \{0,1\}^{k_0},$$

with $k = n + k_0 + k_1$. In the following, we consider, besides one-wayness, two related properties, namely partial-domain one-wayness and set partial-domain one-wayness. We provide "exact" definitions but the reader can easily supply the asymptotic counterparts.

- Permutation f is (τ, ε)-one-way if any adversary $\mathcal{A}$, whose running time is bounded by τ, has success probability $\mathsf{Succ}^{\mathsf{ow}}(\mathcal{A})$ upper-bounded by ε, where

$$\mathsf{Succ}^{\mathsf{ow}}(\mathcal{A}) = \Pr_{s,t}[\mathcal{A}(f(s,t)) = (s,t)];$$

- Permutation f is (τ, ε)-partial-domain one-way if any adversary $\mathcal{A}$, whose running time is bounded by τ, has success probability $\mathsf{Succ}^{\mathsf{pd-ow}}(\mathcal{A})$ upper-bounded by ε, where

$$\mathsf{Succ}^{\mathsf{pd-ow}}(\mathcal{A}) = \Pr_{s,t}[\mathcal{A}(f(s,t)) = s];$$

- Permutation f is $(\ell, \tau, \varepsilon)$-set partial-domain one-way if any adversary $\mathcal{A}$, outputting a set of ℓ elements within time bound τ, has success probability $\mathsf{Succ}^{\mathsf{s-pd-ow}}(\mathcal{A})$ upper-bounded by ε, where

$$\mathsf{Succ}^{\mathsf{s-pd-ow}}(\mathcal{A}) = \Pr_{s,t}[s \in \mathcal{A}(f(s,t))].$$

We denote by $\mathsf{Succ}^{\mathsf{ow}}(\tau)$, (resp. $\mathsf{Succ}^{\mathsf{pd-ow}}(\tau)$ and $\mathsf{Succ}^{\mathsf{s-pd-ow}}(\ell, \tau)$) the maximum success probability $\mathsf{Succ}^{\mathsf{ow}}(\mathcal{A})$ (resp. $\mathsf{Succ}^{\mathsf{pd-ow}}(\mathcal{A})$ and $\mathsf{Succ}^{\mathsf{s-pd-ow}}(\mathcal{A})$). The maximum ranges over all adversaries whose running time is bounded by τ. In the third case, there is an obvious additional restriction on this range stemming from the fact that $\mathcal{A}$ outputs sets with ℓ elements.

3.5.2 Semantic security

In the following, we prove that OAEP is IND-CCA2 in the random oracle model [4], relative to the *set partial-domain* one-wayness of f.

Our method of proof is inspired by Shoup [27] and differs from [11]: we define a sequence Game_1, Game_2, etc of modified attack games starting from the actual game Game_0. Each of the games operates on the same underlying probability space: the public and private key of the cryptosystem, the coin tosses of the adversary $\mathcal{A}$, the random oracles G and H and the hidden bit b for the challenge. Only the rules defining how the view is computed differ from game to game. To go from one game to another, we repeatedly use the following lemma from [27]:

Lemma 1 *Let* E_1, E_2 *and* F *be events defined on a probability space*

$$\Pr[\mathsf{E}_1 \wedge \neg\mathsf{F}] = \Pr[\mathsf{E}_2 \wedge \neg\mathsf{F}] \implies |\Pr[\mathsf{E}_1] - \Pr[\mathsf{E}_2]| \leq \Pr[\mathsf{F}].$$

Proof. The proof follows from easy computations:

$$\begin{aligned}
|\Pr[\mathsf{E}_1] - \Pr[\mathsf{E}_2]| &= |\Pr[\mathsf{E}_1 \wedge \neg\mathsf{F}] + \Pr[\mathsf{E}_1 \wedge \mathsf{F}] - \Pr[\mathsf{E}_2 \wedge \neg\mathsf{F}] - \Pr[\mathsf{E}_2 \wedge \mathsf{F}]| \\
&= |\Pr[\mathsf{E}_1 \wedge \mathsf{F}] - \Pr[\mathsf{E}_2 \wedge \mathsf{F}]| = |\Pr[\mathsf{E}_1 \mid \mathsf{F}] \cdot \Pr[\mathsf{F}] - \Pr[\mathsf{E}_2 \mid \mathsf{F}] \cdot \Pr[\mathsf{F}]| \\
&\leq |\Pr[\mathsf{E}_1 \mid \mathsf{F}] - \Pr[\mathsf{E}_2 \mid \mathsf{F}]| \cdot \Pr[\mathsf{F}] \leq \Pr[\mathsf{F}]
\end{aligned}$$

□

Lemma 2 *Let* $\mathcal{A}$ *be a* CPA*–adversary against the semantic security of the OAEP encryption scheme* $(\mathcal{K}, \mathcal{E}, \mathcal{D})$*. Assume that* $\mathcal{A}$ *has advantage* ε *and running time* τ *and makes* q_G *and* q_H *queries respectively to the hash functions* G *and* H*. Then,*

$$\mathsf{Succ}^{\mathsf{s-pd-ow}}(q_H, \tau) \geq \frac{\varepsilon}{2} - \frac{2q_G}{2^{k_0}}.$$

Proof. As explained, we start with the game coming from the actual attack, and modify it step by step in order to finally obtain a game where the adversary has no advantage.

Game_0**:** A pair of keys $(\mathsf{pk}, \mathsf{sk})$ is generated using $\mathcal{K}(1^k)$. Adversary A_1 is fed with pk, the description of f, and outputs a pair of messages (m_0, m_1). Next a challenge ciphertext is produced by flipping a coin b and producing a ciphertext $y^\star$ of m_b. This implicitly defines a random $r^\star \in_R \{0,1\}^{k_0}$ and a string $x^\star$ such that $y^\star = f(x^\star)$. We set $x^\star = s^\star \| t^\star$, where $s^\star = (m_b \| 0^{k_1}) \oplus G(r^\star)$ and $t^\star = r^\star \oplus H(s^\star)$. On input $y^\star$, A_2 outputs bit b'. We denote by S_0 the event $b' = b$ and use a similar notation S_i in any Game_i below. By definition, we have $\Pr[S_0] = 1/2 + \varepsilon/2$.

Game_1**:** We modify the above game, by immediately stopping the game if $s^\star$ is queried from H, and returning b, thus enforcing $b' = b$. We denote by AskH_1 the event that $s^\star$ is queried from H. We will use an identical notation AskH_i for any Game_i below. We have

$$|\Pr[S_0] - \Pr[S_1]| \leq \Pr[\mathsf{AskH}_1].$$

Game_2**:** This game is modified again, by making the value of the random seed $r^\star$ explicit and moving its generation upfront. In other words, one randomly chooses ahead of time, $r^+ \in_R \{0,1\}^{k_0}$ and $g^+ \in_R \{0,1\}^{k-k_0}$, and uses r^+ instead of $r^\star$, as well as g^+ instead of $G(r^\star)$. The game uses the following two rules:

Rule 1 $r^\star = r^+$ and $s^\star = (m_b \| 0^{k_1}) \oplus g^+$, from which it follows that $t^\star = r^\star \oplus H(s^\star)$, $x^\star = s^\star \| t^\star$ and $y^\star = f(x^\star)$;

Rule 2 whenever the random oracle G is queried at r^+, we answer with g^+.

Since we replace a pair of elements, $(r^\star, G(r^\star))$, by another, (r^+, g^+), with exactly the same distribution (by definition of the random oracle G):

$$\Pr[S_1] = \Pr[S_2] \text{ and } \Pr[\mathsf{AskH}_1] = \Pr[\mathsf{AskH}_2].$$

Game_3**:** In this game, we drop the second rule above and restore (potentially inconsistent) calls to G. Therefore, g^+ is just used in $x^\star$ but does not appear in the computation. The input to $\mathcal{A}_2$ follows a distribution that does not depend on b since m_b is masked with g^+. Accordingly,

$$\Pr[S_3] = \frac{1}{2}.$$

However, one may note that Game_2 and Game_3 may differ if $r^\star$ is queried from G, while Game_2 is played. Returning to Game_1, we see that this happens if $r^\star$ is queried from G before $s^\star$ is queried from H. Let RbS denote the event that, in the original game, $r^\star$ is asked to G before $s^\star$ is asked to H:

$$|\Pr[S_2] - \Pr[S_3]| \leq \Pr[\mathsf{RbS}] \text{ and } |\Pr[\mathsf{AskH}_2] - \Pr[\mathsf{AskH}_3]| \leq \Pr[\mathsf{RbS}].$$

Observe that, before $s^\star$ is queried from H, the value of $H(s^\star)$ is uniformly distributed, and so is $r^\star$. Therefore, the probability of querying $r^\star$ from G is less than $q_G/2^{k_0}$, which writes:

$$\Pr[\mathsf{RbS}] \leq \frac{q_G}{2^{k_0}}.$$

Game$_4$: In order to evaluate AskH_3, we again modify the previous game. When manufacturing the challenge ciphertext, we randomly chooses $y^+ \in_R \{0,1\}^k$, and simply set $y^\star = y^+$, ignoring the encryption algorithm altogether. Once again, the distribution of $y^\star$ remains the same: due to the fact that f is a permutation, the previous method defining $y^\star = f(s^\star \| t^\star)$, with $s^\star = (m_b \| 0^{k_1}) \oplus g^+$ and $t^\star = H(s^\star) \oplus r^+$ was already generating a uniform distribution over the k-bit elements. Thus, we have:

$$\Pr[\mathsf{AskH}_4] = \Pr[\mathsf{AskH}_3].$$

Simply outputting the list of queries to H during this game, one gets

$$\Pr[\mathsf{AskH}_4] \leq \mathsf{Succ}^{\mathsf{s-pd-ow}}(q_H, t).$$

Finally,

$$\begin{aligned} \frac{\varepsilon}{2} = |\Pr[S_0] - \Pr[S_3]| &\leq \Pr[\mathsf{AskH}_1] + \Pr[\mathsf{RbS}] \\ &\leq \Pr[\mathsf{AskH}_4] + 2\Pr[\mathsf{RbS}] \leq \mathsf{Succ}^{\mathsf{s-pd-ow}}(q_H, t) + \frac{2q_G}{2^{k_0}}. \end{aligned}$$

□

3.5.3 Simulating the decryption oracle

In order to prove the security against adaptive chosen-ciphertext attacks, it is necessary to simulate calls to a decryption oracle. As usual, this goes through the design of a plaintext-extractor. This has the flavour of (strong) plaintext-awareness (PA). However, to keep things simple, we will not formally address plaintext-awareness. In any case, as will be seen in the sequel, the situation is more intricate than in the original paper [5]: in particular, the success probability of the extractor cannot be estimated unconditionally but only relatively to some computational assumption.

Definition of the plaintext-extractor $\mathcal{PE}$: The plaintext-extractor receives as part of its input two lists of query-answer pairs corresponding to calls to the random oracles G and H, which we respectively denote by G-List and H-List. It also receives a valid ciphertext $y^\star$. Given these inputs, the extractor should decrypt a candidate ciphertext $y \neq y^\star$.

On query $y = f(s\|t)$, $\mathcal{PE}$ inspects each query/answer pair $(\gamma, G_\gamma) \in$ G-List and $(\delta, H_\delta) \in$ H-List. For each combination of elements, one from each list, it defines

$$\sigma = \delta, \theta = \gamma \oplus H_\delta, \mu = G_\gamma \oplus \delta,$$

and checks whether

$$y = f(\sigma\|\theta) \text{ and } [\mu]_{k_1} = 0^{k_1}.$$

If both equalities hold, $\mathcal{PE}$ outputs $[\mu]^n$ and stops. If no such pair is found, the extractor returns a "Reject" message.

Notations. In the following, $y^\star$ is the challenge ciphertext, obtained from the encryption oracle. Since we have in mind "plugging" the extractor into an adversary $\mathcal{A}$ trying to contradict semantic security, we assume that $y^\star$ is a ciphertext of m_b and denote by $r^\star$ its random seed. We have:

$$r^\star = H(s^\star) \oplus t^\star \text{ and } G(r^\star) = s^\star \oplus (m_b \| 0^{k_1}).$$

The call to H at $s^\star$ is not necessarily reported in H-List. We will explicitly assume that $s^\star$ does not appear H-List. Then, $H(s^\star)$ follows a uniformly distribution and $r^\star$ can be seen as a random variable.

In the sequel, all unstarred variables refer to the decryption query y to be decrypted by the plaintext-extractor.

Lemma 3 *Assume $s^\star$ does not appear in H-List. Then, the plaintext-extractor $\mathcal{PE}$ correctly produces the decryption output on query $y \neq y^\star$, within time bound t' and with probability greater than ε', where*

$$\varepsilon' \geq 1 - \left(\frac{1}{2^{k_1}} + \frac{q_G + 1}{2^{k_0 - 1}} \right) \ , \ t' \leq q_G \cdot q_H \cdot (T_f + \mathcal{O}(1)) \, ,$$

and T_f denotes the time complexity for evaluating f.

Proof. We recall that the plaintext-extractor, $\mathcal{PE}(y, y^\star, \text{G-List}, \text{H-List})$, is given the ciphertext y to be decrypted, the challenge ciphertext $y^\star$ obtained from an encryption oracle, and the G-List and H-List resulting from the execution of an adversary $\mathcal{A}$ interacting with the random oracles G and H.

We first check that the output of $\mathcal{PE}$ is uniquely defined, regardless of the ordering of the lists. To see this, observe that since f is a permutation, the value of $\sigma = s$ is uniquely defined and so is δ. Keep in mind that the G-List and H-List correspond to input-output pairs for the functions G and H, and at most one output is related to a given input. This makes H_δ uniquely defined as well. Similarly, $\theta = t$ is uniquely defined, and thus γ and G_γ: at most one μ may be selected, which is output depending on whether $[\mu]_{k_1} = 0^{k_1}$ or not.

Playing games as before, we denote by Game'_0 the actual game, with the above plaintext-extractor $\mathcal{PE}$, restricted to executions where $s^\star$ has not been queried from H. As already observed, this has the consequence that $H(s^\star)$ follows a uniform distribution. We denote by Fail_0 the event that the extractor's output is not correct. We use a similar notation Fail_i in the successive games.

Game'_1**:** In this game, we focus on the case RBad, defined by $r = r^\star$. Since f is a permutation, it follows that $s \neq s^\star$. Otherwise, we have $t = t^\star$ and thus $y = y^\star$, which is prohibited. Equality $r = r^\star$ writes $H(s) \oplus t = H(s^\star) \oplus t^\star$. Since $H(s^\star)$

is uniformly distributed, $H(s^\star) = H(s) \oplus t \oplus t^\star$ happens with probability at most 2^{-k_0}. If the RBad case happens, one extends the G-List setting

$$\mathsf{G\text{-}List}' = \mathsf{G\text{-}List} \cup \{(r^\star, G(r^\star))\},$$

else one keeps $\mathsf{G\text{-}List}' = \mathsf{G\text{-}List}$. Next, one runs $\mathcal{PE}(y, y^\star, \mathsf{G\text{-}List}', \mathsf{H\text{-}List})$. It follows from the above that:

$$|\Pr[\mathsf{Fail}_1] - \Pr[\mathsf{Fail}_0]| \leq \Pr[\mathsf{RBad}] \leq \frac{1}{2^{k_0}}.$$

Game'_2: We now focus on the event SBad, defined by $s = s^\star$. If the SBad case happens, one extends the H-List, setting

$$\mathsf{H\text{-}List}' = \mathsf{H\text{-}List} \cup \{(s^\star, H(s^\star))\},$$

else one keeps $\mathsf{H\text{-}List}' = \mathsf{H\text{-}List}$. Then, one runs $\mathcal{PE}(y, y^\star, \mathsf{G\text{-}List}', \mathsf{H\text{-}List}')$. Observe that, in the SBad case, $\mathsf{G\text{-}List}'$ is still the original G-List, because RBad and SBad cannot hold together (otherwise $y = y^\star$). Therefore, in the SBad case, the extended $\mathsf{H\text{-}List}'$ involves a change only if r is in $\mathsf{G\text{-}List}' = \mathsf{G\text{-}List}$, in which case Game'_2 might output a result not found by Game'_1. We are thus led to analyze the event AskR that $r = H(s) \oplus t = H(s^\star) \oplus t$ has been queried from G. Since $H(s^\star)$ follows a uniform distribution, the probability of AskR is at most $q_G \cdot 2^{-k_0}$.

$$|\Pr[\mathsf{Fail}_2] - \Pr[\mathsf{Fail}_1]| \leq \Pr[\mathsf{AskR}] \leq \frac{q_G}{2^{k_0}}.$$

Game'_3: In this game, we run the decryption algorithm, when r is in the $\mathsf{G\text{-}List}'$, but s is not in the $\mathsf{H\text{-}List}'$. This involves calls to the oracles. Observe however that $H(s)$ is uniformly distributed, so that the probability for $r = t \oplus H(s)$ to be in $\mathsf{G\text{-}List}'$ is less than $(q_G + 1) \cdot 2^{-k_0}$.

$$|\Pr[\mathsf{Fail}_3] - \Pr[\mathsf{Fail}_2]| \leq \frac{q_G + 1}{2^{k_0}}.$$

Game'_4: Finally, we allow running the decryption algorithm instead of the plaintext-extractor, if r is not in $\mathsf{G\text{-}List}'$. Since $G(r)$ is uniformly distributed, the probability that $[s \oplus G(r)]_{k_1} = 0^{k_1}$ is less than 2^{-k_1}.

$$|\Pr[\mathsf{Fail}_4] - \Pr[\mathsf{Fail}_3]| \leq \frac{1}{2^{k_1}}.$$

If both r and s are in $\mathsf{G\text{-}List}'$ and $\mathsf{H\text{-}List}'$ respectively, the plaintext-extractor never fails. In the other cases, in Game'_4, we use the actual decryption algorithm, thus

$$\Pr[\mathsf{Fail}_4] = 0.$$

We finally obtain the requested bound:

$$\Pr[\mathsf{Fail}_0] = |\Pr[\mathsf{Fail}_4] - \Pr[\mathsf{Fail}_0]| \leq \frac{1}{2^{k_1}} + \frac{q_G + 1}{2^{k_0}} + \frac{q_G}{2^{k_0}} + \frac{1}{2^{k_0}} \leq \frac{1}{2^{k_1}} + \frac{q_G + 1}{2^{k_0 - 1}}.$$

It remains to estimate the running time of the plaintext-extractor. It amounts to the computation of $f(\sigma, \theta)$ for all possible pairs obtained from both lists and is therefore bounded by $q_G \cdot q_H \cdot (T_f + \mathcal{O}(1))$. □

3.5.4 Semantic security against adaptive chosen-ciphertext attacks

We now complete the proof by inserting games Game'_i into games Game_j:

GAME_0: This game is played as Game_0 but the adversary is given additional access to a decryption oracle $\mathcal{D}_{\mathsf{sk}}$ during both steps of the attack. The only requirement is that the challenge ciphertext cannot be queried from the decryption oracle. By definition, we have $\Pr[S_0] = 1/2 + \varepsilon/2$.

GAME_1: In this game, one aborts whenever $s^\star$ is queried from H, and returns b thus enforcing $b' = b$.

$$|\Pr[S_0] - \Pr[S_1]| \leq \Pr[\mathsf{AskH}_1].$$

Observe that calls to the decryption oracle can be perfectly simulated by Game'_4, since $s^\star$ is not queried to H.

GAME'_1: One replaces the decryption oracle by the plaintext-extractor. This is equivalent to replacing each instance of Game'_4 by the corresponding instance of Game'_0, and thus

$$|\Pr[S'_1] - \Pr[S_1]| \leq q_D \times \left(\frac{1}{2^{k_1}} + \frac{q_G + 1}{2^{k_0 - 1}} \right),$$

where q_D is the number of queries to $\mathcal{D}_{\mathsf{sk}}$.

We pursue our modified games exactly as was done in section 3.5.2. We obtain:

$$\frac{\varepsilon}{2} = |\Pr[S_3] - \Pr[S_0]| \leq \Pr[\mathsf{RbS}] + 0 + q_D \times \left(\frac{1}{2^{k_1}} + \frac{q_G + 1}{2^{k_0 - 1}} \right) + \Pr[\mathsf{AskH}_1],$$

while

$$\Pr[\mathsf{AskH}_1] \leq \mathsf{Succ}^{\mathsf{s-pd-ow}}(q_H, t) + \Pr[\mathsf{RbS}] \text{ and } \Pr[\mathsf{RbS}] \leq \frac{q_G}{2^{k_0}}.$$

Therefore,

$$\frac{\varepsilon}{2} \leq \mathsf{Succ}^{\mathsf{s-pd-ow}}(q_H, t) + \frac{2q_G}{2^{k_0}} + q_D \times \left(\frac{1}{2^{k_1}} + \frac{q_G + 1}{2^{k_0 - 1}} \right).$$

This leads to the following theorem.

Theorem 1 *Let $\mathcal{A}$ be a CCA2–adversary against the semantic security of the OAEP encryption scheme $(\mathcal{K}, \mathcal{E}, \mathcal{D})$. Assume that $\mathcal{A}$ has advantage ε and running time τ and makes q_D, q_G and q_H queries to the decryption oracle, and the hash functions G and H respectively. Then,*

$$\begin{aligned} \mathsf{Succ}^{\mathsf{s-pd-ow}}(q_H, \tau') &\geq \frac{\varepsilon}{2} - 2 \times \frac{q_D q_G + q_D + q_G}{2^{k_0}} - \frac{q_D}{2^{k_1}}, \\ \text{with } \tau' &\leq \tau + q_G \cdot q_H \cdot (T_f + \mathcal{O}(1)), \end{aligned}$$

where T_f denotes the time complexity for evaluating f.

Proof. The proof is a straightforward consequence of the above computations and the probability estimates are clear. We comment on the running time. Although the plaintext-extractor is called q_D times, there is no q_D multiplicative factor in the bound for τ'. This comes from a simple bookkeeping argument. Instead of only storing the lists G-List and H-List, one stores an additional structure consisting of tuples $(\gamma, G_\gamma, \delta, H_\delta, y)$. A tuple is included only for $(\gamma, G_\gamma) \in$ G-List and $(\delta, H_\delta) \in$ H-List. For such a pair, one defines

$$\sigma = \delta, \theta = \gamma \oplus H_\delta, \mu = G_\gamma \oplus \delta,$$

and computes $y = f(\sigma, \theta)$. If $[\mu]_{k_1} = 0^{k_1}$, one stores the tuple $(\gamma, G_\gamma, \delta, H_\delta, y)$. The cumulative cost of maintaining the additional structure is $q_G \cdot q_H \cdot (T_f + \mathcal{O}(1))$ but, handling it to the plaintext-extractor allows to output the expected decryption of y, by table lookup, in constant time. □

3.6 The security of RSA–OAEP: partial-domain one-wayness of RSA

In this section, we prove that RSA is set partial-domain one-way, granted that it is one-way. The proof follows [11] and uses two-dimensional lattices. We make the assumption that the seed-length k_0 is not too large $k_0 < k/2$. It is possible to weaken this hypothesis by using higher dimensional lattices. Throughout this section we change the notation for the RSA modulus to N, since variable n is used for another purpose. We still denote by k the bitlength of N. We first prove a lemma on the uniqueness of small solutions to a linear equation modulo N.

Lemma 4 *Let (E) be a linear equation $t + \alpha u = c \bmod N$, with unknowns t and s. Assume that (E) has a solution with t and u smaller than 2^{k_0}. For all values of $\alpha < N$,*

except a fraction $2^{2k_0+6}/N$ of them, (t,u) is unique and can be computed within time bound $\mathcal{O}((\log N)^3)$.

Proof. Consider the lattice

$$L(\alpha) = \{(x,y) \in \mathbb{Z}^2 \,|\, x - \alpha y = 0 \bmod N\}.$$

We say that $L(\alpha)$ is an ℓ-good lattice and that α is an ℓ-good value, if $L(\alpha)$ does not have a non-zero vector of euclidean length at most ℓ. Otherwise, we use the wording ℓ-bad lattices and ℓ-bad values respectively. It is clear that there are approximately less than $\pi\ell^2$ such ℓ-bad lattices, which we bound by $4\ell^2$. Indeed, each bad value for α corresponds to a point with integer coordinates in the disk of radius ℓ. Thus, the proportion of bad values for α is less than $4\ell^2/N$.

Given an ℓ-good lattice, one applies the Gaussian reduction algorithm. One gets within time $\mathcal{O}((\log N)^3)$ a basis of $L(\alpha)$ consisting of two non-zero vectors U and V such that

$$\|U\| \leq \|V\| \text{ and } |(U,V)| \leq \|U\|^2/2.$$

where (U,V) denotes the usual inner product.

Let $T = (t,u)$, where (t,u) is a solution of (E), such that t and u are less than 2^{k_0}. Write:

$$T = \lambda U + \mu V, \text{ for some real } \lambda, \mu.$$

We get:

$$\begin{aligned} \|T\|^2 &= \lambda^2\|U\|^2 + \mu^2\|V\|^2 + 2\lambda\mu(U,V) \geq (\lambda^2 + \mu^2 - \lambda\mu) \times \|U\|^2 \\ &\geq \big((\lambda - \mu/2)^2 + 3\mu^2/4\big) \times \|U\|^2 \geq 3\mu^2/4 \times \|U\|^2 \geq 3\mu^2\ell^2/4. \end{aligned}$$

Since furthermore we have $\|T\|^2 \leq 2 \times 2^{2k_0}$, we obtain:

$$|\mu| \leq \frac{2\sqrt{2}\cdot 2^{k_0}}{\sqrt{3}\cdot\ell}, \text{ and } |\lambda| \leq \frac{2\sqrt{2}\cdot 2^{k_0}}{\sqrt{3}\cdot\ell} \text{ by symmetry.}$$

Assuming that we have set from the beginning $\ell = 2^{k_0+2} > 2^{k_0+2}\sqrt{2/3}$, then

$$-\frac{1}{2} < \lambda, \mu < \frac{1}{2}.$$

Choose any integer solution $T_0 = (t_0, u_0)$ of equation (E) by simply picking a random integer u_0 and setting $t_0 = c - \alpha u_0 \bmod N$. Write T in basis (U,V): $T_0 = \rho U + \sigma V$ where real numbers ρ and σ are easily computed. Now $T - T_0$ is a solution to the homogeneous equation, and thus an element of the lattice $L(\alpha)$: $T - T_0 = aU + bV$, with unknown integers a and b. But,

$$T = T_0 + aU + bV = (a+\rho)U + (b+\sigma)V = \lambda U + \mu V,$$

with $-1/2 \leq \lambda, \mu \leq 1/2$. As a conclusion, a and b are the closest integers to $-\rho$ and $-\sigma$ respectively. With a, b, ρ and σ, one can easily recover λ and μ and thus t and u, which are necessarily unique. □

Lemma 5 *Let $\mathcal{A}$ be an algorithm which outputs, with probability ε, a q-set containing the $k - k_0$ most significant bits of the e-th root modulo N of its input. Assume that $\mathcal{A}$ operates within time bound t. There exists an algorithm $\mathcal{B}$ which inverts the RSA function defined by (N, e) with success probability ε', within time bound t', where*

$$\begin{aligned} \varepsilon' &\geq \varepsilon \times (\varepsilon - 2^{2k_0 - k + 6}), \\ t' &\leq 2t + q^2 \times \mathcal{O}(k^3). \end{aligned}$$

Proof. We show how to compute the e-th root of X. We use the homomorphic property of RSA and define, for a randomly chosen α:

$$\begin{aligned} X &= (x \cdot 2^{k_0} + r)^e \bmod N, \\ Y &= X\alpha^e = (y \cdot 2^{k_0} + s)^e \bmod N, \end{aligned}$$

with r and s smaller than 2^{k_0}. If we can obtain x and y, we have:

$$\begin{aligned} (y \cdot 2^{k_0} + s) &= \alpha \times (x \cdot 2^{k_0} + r) \bmod N \\ \alpha r - s &= (y - x\alpha) \times 2^{k_0} \bmod N \end{aligned}$$

which is a linear modular equation with small solutions. Using lemma 4, we can obtain these solutions.

Algorithm $\mathcal{B}$ runs $\mathcal{A}$ twice, on inputs X and $X\alpha^e$, and next runs the Gaussian reduction algorithm on the lattice $L(\alpha)$. Then, applying the method of lemma 4, it solves the q^2 linear equations coming from pairs of elements (x, y) respectively taken from the two output sets. A solution (r, s) can be tested by raising $x \cdot 2^{k_0} + r$ to the power e modulo N. When both partial pre-images x and y are in the output sets, the correct value of x and r will be found, unless the random α is bad. The bounds for ε' and τ' are straightforward. □

3.7 The security of RSA–OAEP: final result

Putting together the results of the previous sections, we obtain the following result.

Theorem 2 *Let $\mathcal{A}$ be a CCA2–adversary against the semantic security of the RSA–OAEP encryption scheme $(\mathcal{K}, \mathcal{E}, \mathcal{D})$. Assume that $\mathcal{A}$ has advantage ε and running time τ and makes q_D, q_G and q_H queries to the decryption oracle, and the hash functions G and H respectively. Assume further that the parameters for OAEP are such that*

$k_0 < k/2$, where k is the bitlength of the RSA modulus. Then, the problem of RSA-inversion can be solved with probability ε' greater than

$$\frac{\varepsilon^2}{4} - \varepsilon \cdot \left(2 \times \frac{q_D q_G + q_D + q_G}{2^{k_0}} + \frac{q_D}{2^{k_1}} + \frac{32}{2^{k-2k_0}} \right)$$

within time bound $t' \leq 2t + q_H \cdot (q_H + 2q_G) \times \mathcal{O}(k^3)$.

Proof. Theorem 1 states that

$$\mathsf{Succ}^{\mathsf{s-pd-ow}}(q_H, t') \geq \frac{\varepsilon}{2} - 2 \times \frac{q_D q_G + q_D + q_G}{2^{k_0}} - \frac{q_D}{2^{k_1}},$$

with $t' \leq t + q_G \cdot q_H \cdot (T_f + \mathcal{O}(1))$, and $T_f = \mathcal{O}(k^3)$. We easily conclude, using lemma 5. □

Remark: There is a slight inconsistency in piecing together the results from section 3.5.4 and 3.6, coming from the fact that RSA is not a permutation over k-bit strings. Research papers usually ignore the problem. Of course, standards have to cope with it, and we will later discuss how PKCS #1 addresses the matter. Observe that one may decide to only encode message of $n - 1$ bits, where n is $k - k_0 - k_1$ as before. The additional redundancy leading bit can be treated the same way as the 0^{k_1} redundancy, especially with respect to decryption. However, this is not enough since $G(r)$ might still carry the string $(s\|t)$ outside the domain of the RSA encryption function. An easy way out is to start with another random seed if this happens. On average, two trials will be enough. We do not pursue here since formatting will be the central topic in the next section.

4 The PKCS#1 v2.0 version of RSA–OAEP

RSA–OAEP appears in many standards such as SET, TLS, PKCS #1 v2.0 [15], the IEEE standard [14] and an ANSI draft standard [1]. In this report, we only discuss the version from PKCS #1 v2.0. It turns out that the proof described in the previous section cannot be invoked in a direct manner to claim the security of this version. On one hand, the standard has its own specific way to cope with the fact that RSA is not a permutation over k-bit strings, a difficulty that was pointed out above in section 3.7. On the other hand, in the original RSA–OAEP [5] and in further research papers such as [11], the random seed is located at the least significant bits and the message and redundancy at the most significant bits, whereas in PKCS #1 v2.0 [15], the opposite option is taken.

4.1 Description of the encoding method EME–OAEP

In the PKCS #1 standard [15, 24], description of RSAES–OAEP, one uses the OAEP padding scheme with $k = 8(\mathsf{nLen} - 1)$, where nLen is the byte length of the modulus

N:

$$\mathsf{nLen} = \left\lceil \frac{|N|}{8} \right\rceil .$$

This is a simple way to only work with strings which are binary representations of numbers not exceeding the modulus. It should be noted that the various documents where RSA-OAEP is described slightly differ from a notational point of view and also at implementation level, although they all propose the same mathematical transformation. In this section, we follow the standard [15, 24]. We will later discuss the changes brought in the CRYPTRECH submission [25].

The encoding function takes the message M as an argument, together with a byte string P, which encodes some parameters (the default is the empty string), and the byte length emLen of the encoding: $k = 8 \times \mathsf{emLen}$. We denote by mLen the byte length of the message M: $n = 8 \times \mathsf{mLen}$.

The scheme specifies a hash function which outputs digests of hLen bytes. Presently, only SHA1 is allowed. From the hash function a mask generation function MGF is defined, that takes as input an octet string of variable length together with a desired output lentgh, and outputs an octet string of the desired length. Two instantiations of the mask generation function are used in place of G and H, $\mathsf{MGF}(., \mathsf{emLen} - \mathsf{hLen})$ and $\mathsf{MGF}(., \mathsf{hLen})$.

To perform the encoding, one chooses a hLen-byte random *seed* and then builds a Data Block (DB):

$$DB = pHash \,\|\, PS \,\|\, M$$

which is the concatenation of $pHash = h(P)$, a Padding String consisting of a possibly empty sequence of zero bytes ending with byte $\mathsf{01}$, and the message M. To link these notations with the original OAEP description, and thus with the notations of the previous sections, we provide the following dictionary:

OAEP $\longleftrightarrow$	EME-OAEP
m $\longleftrightarrow$	$PS \,\|\, M$
n $\longleftrightarrow$	$8 \times \mathsf{mLen}$
0^{k_1} $\longleftrightarrow$	$pHash$
k_0 $\longleftrightarrow$	$8 \times \mathsf{hLen}$
k_1 $\longleftrightarrow$	$8 \times \mathsf{hLen}$
$G(.)$ $\longleftrightarrow$	$\mathsf{MGF}(., \mathsf{emLen} - \mathsf{hLen})$
$H(.)$ $\longleftrightarrow$	$\mathsf{MGF}(., \mathsf{hLen})$
r $\longleftrightarrow$	$seed$
k $\longleftrightarrow$	$8 \times \mathsf{emLen}$
s $\longleftrightarrow$	$MaskedDB$
t $\longleftrightarrow$	$MaskedSeed$

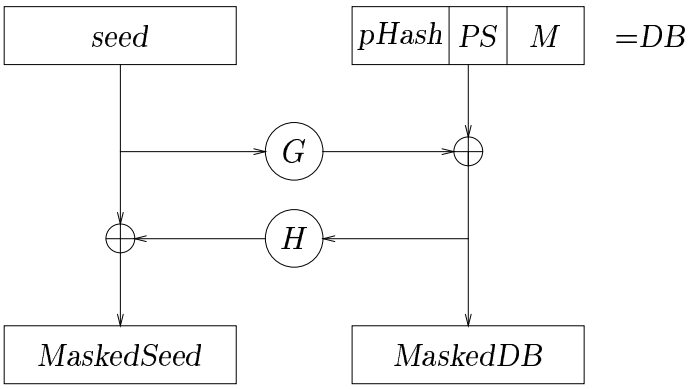

Figure 5: EME–OAEP–Encode

Note that the byte length of PS is $\mathsf{emLen} - \mathsf{mLen} - 2\mathsf{hLen}$, and at least one (the byte $\mathsf{01}$), hence the restriction:

$$\mathsf{mLen} \leq \mathsf{emLen} - 2\mathsf{hLen} - 1 = \mathsf{nLen} - 2\mathsf{hLen} - 2.$$

Once the seed has been chosen and the data block DB has been computed, encoding is performed as follows, where we use notations G and H for clarity (see figure 5):

$$\mathit{MaskedDB} = \mathit{DB} \oplus G(\mathit{seed}) \text{ and } \mathit{MaskedSeed} = \mathit{seed} \oplus H(\mathit{MaskedDB}),$$

The output is $\mathit{MaskedSeed} \,\|\, \mathit{MaskedDB}$. Remark that, as already observed, the output corresponds to $t\|s$, using the original OAEP notation, rather than $s\|t$.

4.2 Security of RSAES–OAEP

In documents [15, 24], the standard scheme RSAES–OAEP encrypts data by applying the RSA primitive to an EME–OAEP encoding of the message M. Recall from the previous section that the output of the encoding is an emLen-byte string, $\mathit{MaskedSeed} \,\|\, \mathit{MaskedDB}$, while the modulus is a nLen-byte number, with $\mathsf{nLen} = \mathsf{emLen} + 1$. In other words the integer handled to the RSA function, after the proper conversion routine has been called is $< 2^{8\mathsf{emLen}}$. For this reason, the security proof has to be double-checked.

Going through the proof, we see that the difference only matters in the proof of lemma 2. More accurately, the claim that games Game_3 and Game_4 were identical is no longer true: picking $y^\star$ at random may result into a value of $x^\star$ exceeding $2^{8\mathsf{emLen}}$. We are thus led to consider the event XBad that $x^\star = \alpha \,\|\, \mathit{MaskedSeed} \,\|\, \mathit{MaskedDB}$, where α is a non-zero byte. we have:

$$\Pr[\mathsf{AskH}_4 \mid \neg\mathsf{XBad}] = \Pr[\mathsf{AskH}_3] \text{ and } \Pr[\neg\mathsf{XBad}] \geq 2^{-8}.$$

Accordingly, a multiplicative factor 2^8 appears in the bound for $\Pr[\mathsf{AskH}_3]$

$$\begin{aligned}\Pr[\mathsf{AskH}_3] &= \Pr[\mathsf{AskH}_4 \mid \neg\mathsf{XBad}] = \Pr[\mathsf{AskH}_4 \wedge \neg\mathsf{XBad}]/\Pr[\neg\mathsf{XBad}] \\ &\leq \Pr[\mathsf{AskH}_4]/\Pr[\neg\mathsf{XBad}] \leq 2^8 \times \mathsf{Succ}^{\mathsf{s-pd-ow}}(q_H, t).\end{aligned}$$

The rest of the proof of theorem 1 goes through, *mutatis mutandis*, carrying this extra multiplicative factor 2^8 throughout the estimates.

Another difference that we have to deal with, is the swapping of s and t. Here, it is easily seen that lemma 5 still holds if one replaces the $k - k_0$ most significant bits by the $k - k_0$ least significant bits. The proof of the variant only requires minor notational changes. At this point, we can check that the assumption $k_0 < k/2$ that we made in section 3.6 is duly satisfied, since documents [15, 24] require $\mathsf{emLen} \geq 2\mathsf{hLen} + 1$. Note that this requirement has no consequence in practical terms: PKCS #1 V.2.0 [15] mandates the use SHA-1 [18] and suggests moduli of at least 1024 bits. The draft of PKCS #1 V.2.1 [24] allows hash functions with a larger value of hLen such as SHA-256, SHA-384 or SHA-512 [19]. The latter entails the use of larger moduli.

Summing up our remarks, we can state the following security result:

Theorem 3 *Let $\mathcal{A}$ be a CCA2–adversary against the semantic security of the RSAES–OAEP encryption scheme. Assume that $\mathcal{A}$ has advantage ε and running time τ and makes q_D, q_G and q_H queries to the decryption oracle, and the hash functions G and H respectively. Then*

$$\begin{aligned}\mathsf{Succ}^{\mathsf{s-pd-ow}}(q_H, \tau') &\geq \frac{\varepsilon}{2^9} - \frac{2q_Dq_G + 3q_D + 2q_G}{2^{\mathsf{hLen}+8}}, \\ \text{with } \tau' &\leq t + q_G \cdot q_H \cdot \mathcal{O}(\mathsf{nLen}^3).\end{aligned}$$

Furthermore, the problem of RSA-inversion can be solved with probability ε'' and within time bound τ'', where

$$\begin{aligned}\varepsilon'' &\geq \frac{1}{2^{16}}\left(\frac{\varepsilon^2}{4} - \varepsilon\left(\frac{2q_Dq_G + 3q_D + 2q_G}{2^{\mathsf{hLen}}} + \frac{2^{15}}{2^{\mathsf{nLen}-2\mathsf{hLen}}}\right)\right), \\ \tau'' &\leq 2\tau + (2q_G + q_H) \cdot q_H \cdot \mathcal{O}(\mathsf{nLen}^3).\end{aligned}$$

Assume $\mathsf{nLen} \geq 3\mathsf{hLen}$. The above shows that, given an CCA2–adversary against the semantic security of the RSAES–OAEP encryption scheme $(\mathcal{K}, \mathcal{E}, \mathcal{D})$, with advantage ε and running time τ, making q_D, q_G and q_H queries to the decryption oracle, and the hash functions G and H respectively, one can invert RSA with success probability greater than ε' and within a time bound τ', where

$$\begin{aligned}\varepsilon' &= \frac{1}{2^{16}}\left(\frac{\varepsilon^2}{4} - \varepsilon \times \frac{2q_Dq_G + 3q_D + 2q_G + 2^{15}}{2^{\mathsf{hLen}}}\right) \\ \text{with } \tau' &= 2\tau + (2q_G + q_H) \cdot q_H \cdot \mathcal{O}(\mathsf{nLen}^3).\end{aligned}$$

bit-size of the modulus	complexity of NFS in $\log_2$
512	63
1024	85
4096	155
6144	182
8192	206

Figure 6: Complexity of factoring

4.3 Implications in terms of key sizes

While RSA–OAEP is indeed chosen-ciphertext secure, the reduction to inverting RSA is not tight and the estimates that were provided in the previous section cannot be used to derive key sizes. To see this, observe that we can take $\varepsilon' \approx \varepsilon^2/2^{18}$ and $\tau' \approx 2q^2\mathcal{O}(\mathsf{nLen}^3)$, where q is an estimate of the number of hash queries that the attacker makes. This yields an estimate of the average time $T' = \tau'/\varepsilon'$ for inverting RSA of the form $\approx 2^{19} \times \frac{q^2}{\varepsilon^2}$. We now allow the adversary an average number of hash queries $\frac{q}{\varepsilon} = 2^\ell$, where ℓ is a security parameter. This allows inverting RSA with complexity $\approx 2^{19} \times 2^{2\ell}$. Taking logarithms yields $\approx 19 + 2\ell$.

We now use a table of time estimates for the best known factoring method NFS (-see figure 6). We derive an estimate of the key size corresponding to a given value of the security parameter by searching for a modulus for which the righthand side is $19 + 2\ell$. As a typical case, one should select a 6000-bit modulus to provide a security level 2^{80}. In the reverse direction, we see that the table indicates that a 1024-bit modulus provides a quite low security level 2^{33}.

We can thus conclude that the security proof only gives assurance that the overall design of RSA–OAEP is not flawed. It does not appear possible to obtain any meaningful indication on key sizes from the estimates derived from the proof. As noted in section 2.4, it is anyhow debatable whether or not security proofs in the random oracle model can bring more that a qualititive level of assurance. This is due to their intrinsic heuristic nature.

4.4 Implementation issues

In a recent paper, Manger [16] showed that the implementation of RSA–OAEP had to be done with extreme care: slight implementation errors may have devastating consequences. This was no surprise to the research community. However, it is interesting to review Manger's findings.

In any implementation of RSA–OAEP, there are many exception errors. Some are

actually related to the exact format of the EME–OAEP encoding. Given any ciphertext c, RSA decryption returns $x = c^d \bmod N$, where d is the private RSA exponent, or an error if the ciphertext exceeds the modulus. Next, a standard conversion routine turns x into a byte string, which can be parsed as $\alpha \,\|\, \mathit{MaskedSeed} \,\|\, \mathit{MaskedDB}$ where $\mathit{MaskedDB}$ consists of the $\mathsf{emLen} - \mathsf{hLen}$ least significant bytes, $\mathit{MaskedSeed}$ the hLen following ones, and α the most significant byte. From these data, one can recover the $\mathit{seed} = \mathit{MaskedSeed} \oplus H(\mathit{MaskedDB})$, and the data block $DB = \mathit{MaskedDB} \oplus G(\mathit{seed})$. The data block itself is parsed into $DB = \mathit{pHash} \,\|\, \beta \,\|\, \gamma \,\|\, M$, where pHash consists of the hLen most significant bytes, β the largest sequence consisting of consecutive zero-bytes following pHash and γ a non-zero byte. The trailing part is the candidate message M. Before outputting M, one should have checked the following:

- $\alpha = \mathsf{00}_{byte}$;
- $\gamma = \mathsf{01}_{byte}$;
- $\mathit{pHash} = h(P)$.

The PKCS #1 standard very clearly states that the error messages returned in case any one of the above is unsuccessful should be the same. Version 2.0 [15] writes:

> It is important that the error messages output be the same ...

while version 2.1 states

> It is important that the errors are indistinguishable ...

Both documents refer to Bleichenbacher's attack [6] against the earlier version 1.5. Manger focused on implementation errors that might allow an adversary to obtain the answer to the first check, $\alpha \stackrel{?}{=} 0$, which we will henceforth call the *first-byte test*. This immediately opens a way to the simplified version of Bleichenbacher's attack that we described in section 2.3. Suitably optimizing this attack, Manger [16] showed that approximately $8\mathsf{nLen}$ queries were enough to disclose the plaintext corresponding to a given ciphertext.

It is therefore important to check whether or not an adversary has access to the type of information of which Manger builds. In his paper, several potential sources of leakage are investigated:

- Error messages: misspelling one word in an error message might allow the adversary to spot occurrences of the first-byte test. Error logs documenting the error reason have a similar effect, if the adversary has access to the logs.
- Timings: the first-byte test might be performed at an earlier stage. Negative answers would abort further processing and this might be detected by measuring the delay before rejection happens.

- Side effects on function calls: this is a more subtle variant of the previous case. The adversary might try to submit a decryption request with an unsupported hash function. If the first-byte test is performed earlier, a negative answer might abort further processing while a positive answer triggers the detection of the unsupported object, through an error message when the hash function is called.

Some of the above scenarios may seem unlikely: for example, timing measurements are presumably unable to detect the additional computation of a hash function. However, it is good practice to help the programmer avoiding mistakes as much as possible. Accordingly, Manger's observations should be addressed. Note that, in documents [15, 24], the first-byte check is definitely performed *before* the EME–OAEP decoding function is called. Thus, these documents are not completely satisfactory, in view of Manger's results. Also note that it might be tempting to simply drop the first-byte test and only rely on the other two tests, as was suggested by Manger's himself. However, this would be an extremely bad idea in terms of provable security, since the proof included in the present report would fail. To understand why, refer to section 3.5.3 and consider the case where the plaintext-extractor receives a candidate ciphertext $y \neq y^\star$, such that $s = s^\star$ and $r = r^\star$. This cannot be formally excluded since discarding the first-byte check allows to flip one or more bits in the most significant byte of the integer handled to the RSA function to obtain $y^\star$. Our plaintext extractor, would erroneously reject such y.

Thus, the appropriate solution seems to perform the three checks in a single step. This is precisely what is suggested in the CRYPTRECH submission [25]. The approach taken there slightly change notations and makes the extended message of length $\mathsf{emLen} = \mathsf{nLen}$, rather than $\mathsf{nLen} - 1$. EME–OAEP encoding outputs a string $00 \| \mathit{MaskedSeed} \| \mathit{MaskedDB}$ and EME–OAEP decoding performs the three tests in a single step, which can be found on page 4 of the update to [25]. One should definitely refer to the update, since the corresponding test in the original document does not seem correct. In conclusion, it is fair to say that document [25] appropriately addresses Manger's observations.

5 Conclusions

Based on our analysis, we believe that the RSA–OAEP cryptosystem is semantically secure against chosen-ciphertext attacks, based on the hardness of inverting RSA. We have indeed provided a security proof in Shoup's style [27], different from the proof earlier published by Fujisaki, Okamoto, Pointcheval and Stern [11]. Both proofs use the random oracle model. The estimates that follow from our proof are essentially similar to those from [11] and they do not provide any conclusive evidence in terms of practical parameter sizes. Although schemes with tighter reductions might yield slightly more confidence, we believe anayway that it is debatable whether or not security proofs in

the random oracle model can bring more than a qualititive level of assurance, due to their intrinsic heuristic nature. This qualitative assurance level has been reached. We have also reviewed the potential implementation errors that might degrade the security of RSA–OAEP and found that document [25], notably the update, appropriately addressed these potential errors. We therefore recommend the scheme.

References

[1] ANSI X9.44. Key Establishment Using Factoring-Based Public Key Cryptography for the Financial Services Industry. Working draft, ANSI X9F1 Working Group, June 2000.

[2] M. Bellare. Practice-Oriented Provable Security. In *ISW '97*, LNCS 1396, Springer-Verlag, Berlin, 1997.

[3] M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway. Relations among Notions of Security for Public-Key Encryption Schemes. In *Crypto '98*, LNCS 1462, pages 26–45, Springer-Verlag, Berlin, 1998.

[4] M. Bellare and P. Rogaway. Random Oracles Are Practical: a Paradigm for Designing Efficient Protocols. In *Proc. of the 1st CCS*, pages 62–73, ACM Press, New York, 1993.

[5] M. Bellare and P. Rogaway. Optimal Asymmetric Encryption – How to Encrypt with RSA. In *Eurocrypt '94*, LNCS 950, pages 92–111, Springer-Verlag, Berlin, 1995.

[6] D. Bleichenbacher. A Chosen Ciphertext Attack against Protocols based on the RSA Encryption Standard PKCS #1. In *Crypto '98*, LNCS 1462, pages 1–12, Springer-Verlag, Berlin, 1998.

[7] D. Boneh and R. Venkatesan. Breaking RSA may not be equivalent to factoring. In *Eurocrypt '98*, LNCS 1402, pages 59–71, Springer-Verlag, Berlin, 1998.

[8] R. Canetti, O. Goldreich, and S. Halevi. The Random Oracles Methodology, Revisited. In *Proc. of the 30th STOC*, pages 209–218, ACM Press, New York, 1998.

[9] D. Dolev, C. Dwork, and M. Naor. Non-Malleable Cryptography. *SIAM Journal on Computing*, 30(2):391–437, 2000.

[10] A. Fiat and A. Shamir. How to Prove Yourself: Practical Solutions of Identification and Signature Problems. In *Crypto '86*, LNCS 263, pages 186–194, Springer-Verlag, Berlin, 1987.

[11] E. Fujisaki, T. Okamoto, D. Pointcheval, and J. Stern. RSA–OAEP is Secure under the RSA Assumption. In *Crypto '2001*, LNCS 2139, pages 260–274, Springer-Verlag, Berlin, 2001. Also appeared in the Cryptology ePrint Archive 2000/061. November 2000. Available from `http://eprint.iacr.org/`.

[12] S. Goldwasser and S. Micali. Probabilistic Encryption. *Journal of Computer and System Sciences*, 28:270–299, 1984.

[13] C. Hall, I. Goldberg, and B. Schneier. Reaction Attacks Against Several Public-Key Cryptosystems. In *Proc. of ICICS'99*, LNCS, pages 2–12, Springer-Verlag, 1999.

[14] IEEE Standard 1363–2000. Standard Specifications for Public Key Cryptography. IEEE. Available from `http://grouper.ieee.org/groups/1363`, August 2000.

[15] B. Kaliski and J. Staddon. IETF RFC 2437: PKCS #1: RSA Cryptography Specifications Version 2.0. October 1998.

[16] J. Manger. A Chosen Ciphertext Attack on RSA Optimal Asymmetric Encryption Padding (OAEP) as Standardized in PKCS #1. In *Crypto '2001*, LNCS 2139, pages 230–238, Springer-Verlag, Berlin, 2001.

[17] M. Naor and M. Yung. Public-Key Cryptosystems Provably Secure against Chosen Ciphertext Attacks. In *Proc. of the 22nd STOC*, pages 427–437, ACM Press, New York, 1990.

[18] NIST. Secure Hash Standard (SHS). Federal Information Processing Standards PUBlication 180–1, April 1995.

[19] NIST. Descriptions of SHA–256, SHA–384, and SHA–512. October 2000. Available from `http://www.nist.gov/sha/`.

[20] T. Okamoto and D. Pointcheval. REACT: Rapid Enhanced-security Asymmetric Cryptosystem Transform. In *CT – RSA '2001*, LNCS 2020, pages 159–175, Springer-Verlag, Berlin, 2001.

[21] C. Rackoff and D. R. Simon. Non-Interactive Zero-Knowledge Proof of Knowledge and Chosen Ciphertext Attack. In *Crypto '91*, LNCS 576, pages 433–444, Springer-Verlag, Berlin, 1992.

[22] R. Rivest, A. Shamir, and L. Adleman. A Method for Obtaining Digital Signatures and Public Key Cryptosystems. *Communications of the ACM*, 21(2):120–126, February 1978.

[23] RSA Laboratories. PKCS # 1 Version 1.5: RSA Cryptography Standard. November 1993, `http://www.rsa.com/rsalabs/pubs/PKCS/`.

[24] RSA Laboratories. PKCS # 1 Version 2.1: RSA Cryptography Standard. Draft, January 2001, `http://www.rsa.com/rsalabs/pubs/PKCS/`.

[25] RSA Laboratories. The RSA-OAEP Encryption Scheme. Cryptographic Technique Specification (with update), submission to CRYPTRECH, September 2001.

[26] RSA Laboratories. The RSA-OAEP Encryption Scheme. Self Evaluation Report (with update), submission to CRYPTRECH, September 2001.

[27] V. Shoup. OAEP Reconsidered. In *Crypto '2001*, LNCS 2139, pages 239–259. Springer-Verlag, Berlin, 2001. Also appeared in the Cryptology ePrint Archive 2000/060. November 2000. Available from `http://eprint.iacr.org/`.