Journal of Cryptology.
Sringer-Verlag, 2002, to appear

Journal of CRYPTOLOGY
The Journal of the International Association for Cryptologic Research

RSA–OAEP is Secure under the RSA Assumption*

Eiichiro Fujisaki[1], Tatsuaki Okamoto[1], David Pointcheval[2], and Jacques Stern[2]

[1] NTT Labs, 1-1 Hikarino-oka, Yokosuka-shi, 239-0847 Japan.
E-mail: {fujisaki,okamoto}@isl.ntt.co.jp.
[2] Dépt d'Informatique, ENS – CNRS, 45 rue d'Ulm, 75230 Paris Cedex 05, France.
E-mail: {David.Pointcheval,Jacques.Stern}@ens.fr
URL: http://www.di.ens.fr/users/{pointche,stern}.

Abstract. Recently Victor Shoup noted that there is a gap in the widely-believed security result of OAEP against adaptive chosen-ciphertext attacks. Moreover, he showed that, presumably, OAEP cannot be proven secure from the *one-wayness* of the underlying trapdoor permutation. This paper establishes another result on the security of OAEP. It proves that OAEP offers semantic security against adaptive chosen-ciphertext attacks, in the random oracle model, under the *partial-domain* one-wayness of the underlying permutation. Therefore, this uses a formally stronger assumption. Nevertheless, since partial-domain one-wayness of the RSA function is equivalent to its (full-domain) one-wayness, it follows that the security of RSA–OAEP can actually be proven under the sole RSA assumption, although the reduction is not tight.

1. Introduction

The OAEP conversion method [3] was introduced by Bellare and Rogaway in 1994 and was believed to provide semantic security against adaptive chosen-ciphertext attacks [8, 12], based on the one-wayness of a trapdoor permutation, using the (corrected) definition of plaintext-awareness [1].

Victor Shoup [15] recently showed that it is quite unlikely that such a security proof exists — at least for non-malleability — under the one-wayness of the permutation. He also proposed a slightly modified version of OAEP, called OAEP+, which can be proven secure, under the one-wayness of the permutation.

Does Shoup's result mean that OAEP is insecure or that it is impossible to prove the security of OAEP? This would be a misunderstanding of [15]: Shoup's result only states that it is highly unlikely to find any proof, under just the one-wayness assumption. In other words, it does not preclude the possibility of proving the security of OAEP from stronger assumptions.

This paper uses such a stronger assumption. More precisely, in our reduction, a new computational assumption is introduced to prove the existence of

* This is the full version of the paper presented at Crypto '01 [7]

a simulator of the decryption oracle. Based on this idea, we prove that OAEP is semantically secure against adaptive chosen-ciphertext attack in the random oracle model [3], under the *partial-domain* one-wayness of the underlying permutation, which is stronger than the original assumption.

Since partial-domain one-wayness of the RSA function [13] is equivalent to the (full-domain) one-wayness, the security of RSA-OAEP can actually be proven under the one-wayness of the RSA function.

The rest of this paper is organized as follows. Section 2. recalls the basic notions of asymmetric encryption and the various security notions. Section 3. reviews the OAEP conversion [3], with a thorough discussion of its proven security. Section 4. presents our new security result together with a formal proof for general OAEP applications, using the Shoup's formalism [15] which differs from our original paper [7]. In Section 5., we focus on the RSA application of OAEP, RSA-OAEP. Finally, Section 6. and appendix include a more precise, but more intricate proof, which provides a tighter security result.

2. Public-Key Encryption

The aim of public-key encryption is to allow anybody who knows the public key of Alice to send her a message that only she will be able to recover by means of her private key.

2.1. *Definitions*

A public-key encryption scheme over a message space $\mathcal{M}$ is defined by the three following algorithms:

– the *key generation algorithm* $\mathcal{K}(1^k)$, where k is the security parameter, produces a pair $(\mathsf{pk}, \mathsf{sk})$ of matching public and private keys. Algorithm $\mathcal{K}$ is probabilistic.
– 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 occasionally 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.

2.2. *Security Notions*

The first security notion that one would like for an encryption scheme is *one-wayness*: starting with just public data, an attacker cannot recover the complete plaintext of a given ciphertext. More formally, this means that, for any adversary $\mathcal{A}$, its success probability in inverting $\mathcal{E}$ without the private key should be negligible over the probability space $\mathcal{M} \times \Omega$, where $\mathcal{M}$ is the message space and Ω includes the random coins r used for the encryption scheme, and the internal random coins of the adversary. For the sake of consistency, the message space $\mathcal{M}$ is assumed to be quite large, whereas the random space Ω is of any size (it can even be empty, if one considers a deterministic encryption scheme). In symbols, the success probability reads

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

However, many applications require more from an encryption scheme, namely *semantic security* (*a.k.a. polynomial security* or *indistinguishability of encryptions* [8], denoted IND): if the attacker has some information about the plaintext, for example that it is either "yes" or "no" to a crucial query, no adversary should learn more with the view of the ciphertext. This is an extension of the above one-wayness, when the message space may be made quite small. This security notion requires computational impossibility to distinguish between two messages, chosen by the adversary, one of which has been encrypted, with a probability significantly better than one half: the advantage $\mathsf{Adv}^{\mathsf{ind}}(\mathcal{A})$, where the adversary $\mathcal{A}$ is seen as a 2-stage Turing machine (A_1, A_2), should be negligible, where $\mathsf{Adv}^{\mathsf{ind}}(\mathcal{A})$ is formally defined as.

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

Another notion was defined thereafter, the so-called *non-malleability* (NM) [6], in which the adversary tries to produce a new ciphertext such that the plaintexts are meaningfully related. This notion is stronger than the above one, but it is equivalent to semantic security in the most interesting scenario [1].

On the other hand, an attacker can use many kinds of attacks: since we are considering asymmetric encryption, the adversary can encrypt any plaintext of its choice with the public key, hence *chosen-plaintext attack*. It may, furthermore, have access to more information, modeled by restricted or unrestricted 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 rises to *plaintext-checking attack* [11]. A *validity-checking oracle* answers whether its input c is a valid ciphertext or not. This scenario has been termed *reaction attack* [9]. It has been successfully applied to break the famous PKCS #1 v1.5 encryption scheme [4]. Finally, a *decryption oracle* returns the decryption of any ciphertext, with the only restriction that it should be different from the challenge ciphertext. When the oracle access is only granted to the adversary before the view of the challenge ciphertext, the corresponding scenario is termed *indifferent chosen-ciphertext attack* (*a.k.a. non-adaptive chosen-ciphertext attack* or *lunchtime attack* [10]), denoted CCA1. When the adversary has also access to the decryption oracle in the second stage, we talk about *adaptive chosen-ciphertext attack* [12], denoted CCA2. This latter scenario is the strongest one. A general study of these security notions and attacks was given in [1]. The results are summarized in the diagram shown on figure 1.

Thus, in the latter scenario, semantic security and non-malleability are equivalent. This is the strongest security notion that we now consider: semantic security against adaptive chosen-ciphertext attacks (IND-CCA2) – where the adversary just wants to distinguish which plaintext, between two messages of its

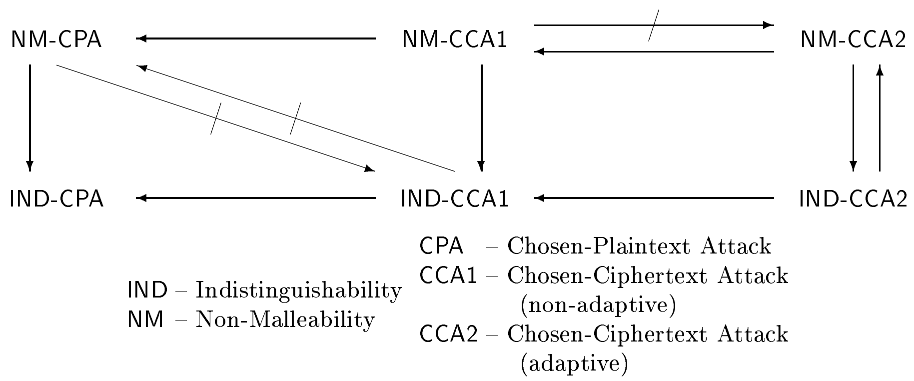

Fig. 1. Relations between security notions

choice, had been encrypted; it can ask any query to a decryption oracle (except the challenge ciphertext).

2.3. *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 [3] 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 $\mathcal{PE}$:

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

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 [1], 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:

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

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. This does not even imply NM-CPA.

3. Review of OAEP

3.1. *The OAEP Cryptosystem*

We briefly describe the OAEP cryptosystem $(\mathcal{K}, \mathcal{E}, \mathcal{D})$ obtained from a permutation f, whose inverse is denoted by g (see figure 2). We need two hash functions G and H:

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

Then,

- $\mathcal{K}(1^k)$: specifies an instance of the function f, and of its inverse g. The public key pk is therefore 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 $r \stackrel{R}{\leftarrow} \{0,1\}^{k_0}$, the 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 $c = f(s, t)$.
- $\mathcal{D}_{\mathsf{sk}}(c)$: thanks to the private key, 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 description, $[M]_{k_1}$ denotes the k_1 least significant bits of M, while $[M]^n$ denotes the n most significant bits of M.

3.2. *Previous Security Results*

As already mentioned, paper [3] 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, also called security against lunchtime attacks (IND-CCA1). We briefly comment on the intuition behind (weak) plaintext-awareness. When, the plaintext-extractor receives a ciphertext c, then:

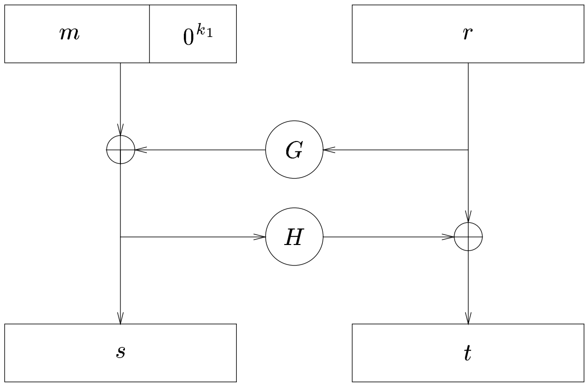

Fig. 2. Optimal Asymmetric Encryption Padding

– 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, since this puts additional implicit constraints on G and H. These constraints cannot be seen by inspecting the query lists.

3.3. *Shoup's Counter-Example*

In his paper [15], Victor Shoup showed that it was quite unlikely to extend the results of [3] 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.

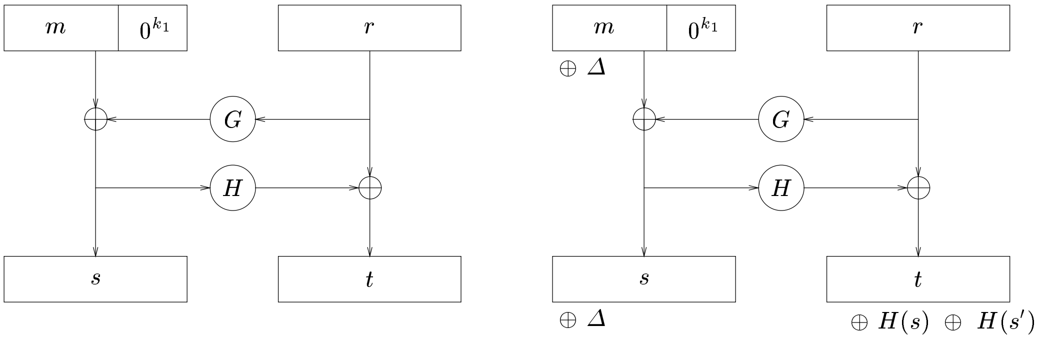

Fig. 3. Shoup's Attack

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 $y = f(s\|t) = s\|u$, 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 3)

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

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 developed at the end of the previous section failed: a valid ciphertext y' was created without querying 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 [15] 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.

4. The Security of OAEP

4.1. *Security Result*

Shoup [15] furthermore provided a specific proof for RSA with public exponent 3. But there is little hope to extend this proof for higher exponents.

In the following, we provide a general security analysis, but under a stronger assumption about the underlying permutation. Indeed, we prove that the scheme is IND-CCA2 in the random oracle model [2], relative to the *partial-domain* one-wayness of permutation f.

4.2. *Outline of the Proof*

In the following, we use starred letters ($r^\star$, $s^\star$, $t^\star$ and $y^\star$) to refer to the challenge ciphertext, whereas unstarred letters (r, s, t and y) will refer to the ciphertext asked to the decryption oracle.

4.2.1. *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 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 pre-image 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 $\mathsf{IND\text{-}CCA2}$ encryption scheme if it is difficult to "partially" invert f, which means: given $y = f(s\|t)$, find s.

4.2.2. *The Strategy.*

Based on the intuition just described, we can formally prove that applying OAEP encoding to a trapdoor permutation which is difficult to partially invert, leads to an $\mathsf{IND\text{-}CCA2}$ encryption scheme, hence the *partial-domain one-wayness*, which expresses the fact that the above partial inversion problem is difficult. Precise definitions will be given in the next paragraph.

As the original proof from [3], our proof has two steps: it is first shown that the OAEP scheme is $\mathsf{IND\text{-}CPA}$ 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 (where the list C of ciphertexts is limited to only one ciphertext, the challenge ciphertext $y^\star$.)

4.2.3. *Partial-Domain One-Wayness.*

Let f 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 original description of OAEP from [3], it is only required that f is a trapdoor one-way permutation. However, in the following, we consider two additional related problems, namely partial-domain one-wayness and set partial-domain one-wayness.

– 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 maximal 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 from the fact that $\mathcal{A}$ outputs sets with ℓ elements. It is clear that for any τ and $\ell \geq 1$,

$$\mathsf{Succ}^{\mathsf{s-pd-ow}}(\ell, \tau) \geq \mathsf{Succ}^{\mathsf{pd-ow}}(\tau) \geq \mathsf{Succ}^{\mathsf{ow}}(\tau).$$

Note that, by randomly selecting an element in the set returned by an adversary to the Set Partial-Domain One-Wayness, one breaks Partial-Domain One-Wayness with probability $\mathsf{Succ}^{\mathsf{s-pd-ow}}(\mathcal{A})/\ell$. This provides the following inequality $\mathsf{Succ}^{\mathsf{pd-ow}}(\tau) \geq \mathsf{Succ}^{\mathsf{s-pd-ow}}(\ell, \tau)/\ell$. However, for specific choices of f, more efficient reductions may exist. Also, in some cases, all three problems are polynomially equivalent. This is the case for the RSA permutation [13], hence the results in Section 5..

4.3. *The Formal Proof*

In the following, we prove that OAEP is $\mathsf{IND\text{-}CCA2}$, in the random oracle model [2], relative to the *set partial-domain* one-wayness of f. More precisely, the rest of the paper is devoted to proving 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,*

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

where T_f denotes the time complexity for evaluating f.

Our method of proof is inspired by Shoup [15]: 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 keys 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 [15]:

Lemma 1. *Let* E_1, E_2 *and* F_1, F_2 *be events defined on a probability space*

$$\Pr[\mathsf{E}_1 \wedge \neg\mathsf{F}_1] = \Pr[\mathsf{E}_2 \wedge \neg\mathsf{F}_2] \text{ and } \Pr[\mathsf{F}_1] = \Pr[\mathsf{F}_2] = \varepsilon \ \Rightarrow \ |\Pr[\mathsf{E}_1] - \Pr[\mathsf{E}_2]| \leq \varepsilon.$$

Proof. The proof follows from easy computations:

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

4.3.1. *Semantic Security.*

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,*

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

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 directly related to the ability of the adversary to partially invert permutation f. The IND-CPA security level of OAEP has already been proven by Bellare and Rogaway [3], relative to an even weaker assumption: the one-wayness of the permutation. In the following, we only consider partial-domain one-wayness, and accordingly, we provide a specific proof which is similar to the Bellare and Rogaway's original proof, but is based on this new algorithmic assumption. We will later extend our proof to deal with chosen-ciphertext attacks.

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 ciphertext comes from a random $r^\star \stackrel{R}{\leftarrow} \{0,1\}^{k_0}$ 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 making the value of the random seed $r^\star$ explicit and moving its generation upfront. In other words, one randomly chooses ahead of time, $r^+ \stackrel{R}{\leftarrow} \{0,1\}^{k_0}$ and $g^+ \stackrel{R}{\leftarrow} \{0,1\}^{k-k_0}$, and uses r^+ instead of $r^\star$, as well as g^+ instead of $G(r^\star)$. The game obeys the following two rules:

Rule 1. $r^\star = r^+$ and $s^\star = (m_b \| 0^{k_1}) \oplus g^+$, from which it follows that

$$t^\star = r^\star \oplus H(s^\star), x^\star = s^\star \| t^\star \text{ and } y^\star = f(x^\star);$$

Rule 2. whenever the random oracle G is queried at r^+, the answer is 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):

$$\Pr[S_1] = \Pr[S_0].$$

Game_2**:** 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. Thus, the input to A_2 follows a distribution that does not depend on b. Accordingly, $\Pr[S_2] = 1/2$.
One may note that Game_1 and Game_2 may differ if $r^\star$ is queried from G. Let AskG_2 denotes the event that, in Game_2, $r^\star$ is queried from G (except by the encryption oracle, for producing the challenge). We will use an identical notation AskG_i for any Game_i below. Then

$$|\Pr[S_2] - \Pr[S_1]| \leq \Pr[\mathsf{AskG}_2].$$

Game_3**:** We now define $s^\star$ independently of anything else, as well as $H(s^\star)$. In other words, one randomly chooses ahead of time, $s^+ \stackrel{R}{\leftarrow} \{0,1\}^{k-k_0}$ and $h^+ \stackrel{R}{\leftarrow} \{0,1\}^{k_0}$, and uses s^+ instead of $s^\star$, as well as h^+ instead of $H(s^\star)$. The only change is that $s^\star = s^+$ instead of $(m_b\|0^{k_1}) \oplus g^+$. The game uses the following two rules:
Rule 1'. $g^+ = (m_b\|0^{k_1}) \oplus s^+$ and $t^\star = r^\star \oplus h^+$;
Rule 2'. whenever the random oracle H is queried at s^+, the answer is h^+.
Since we replace the quadruple $(s^\star, H(s^\star), g^+, b)$ by another with exactly the same distribution (by definition of the random oracle H):

$$\Pr[\mathsf{AskG}_3] = \Pr[\mathsf{AskG}_2].$$

Game_4**:** In this game, we drop the second rule above and restore (potentially inconsistent) calls to H. Therefore, h^+ is just used in $x^\star$ but does not appear in the computation. One may note that Game_3 and Game_4 may differ if $s^\star$ is queried from H. Let AskH_4 denote the event that, in Game_4, $s^\star$ is queried from H (except by the encryption oracle, for producing the challenge). We will use an identical notation AskH_i for any Game_i below. Then

$$|\Pr[\mathsf{AskG}_4] - \Pr[\mathsf{AskG}_3]| \leq \Pr[\mathsf{AskH}_4].$$

Furthermore, $r^\star = t^\star \oplus h^+$ is uniformly distributed, and independent of the adversary's view, since h^+ is never revealed: $\Pr[\mathsf{AskG}_4] \leq q_G/2^{k_0}$, where q_G denotes the number of queries asked to G.

Game_5**:** In order to evaluate AskH_4, we again modify the previous game. When manufacturing the challenge ciphertext, we randomly choose $y^+ \stackrel{R}{\leftarrow} \{0,1\}^k$, 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 $s^\star = s^+$ and $t^\star = h^+ \oplus r^+$ was already generating a uniform distribution over the k-bit elements. Thus, we have:

$$\Pr[\mathsf{AskH}_5] = \Pr[\mathsf{AskH}_4].$$

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

$$\Pr[\mathsf{AskH}_5] \leq \mathsf{Succ}^{\mathsf{s-pd-ow}}(q_H, \tau).$$

Finally,

$$\frac{\varepsilon}{2} = |\Pr[S_0] - \Pr[S_2]| \leq \Pr[\mathsf{AskG}_2] \leq \Pr[\mathsf{AskG}_4] + \Pr[\mathsf{AskH}_4]$$
$$\leq \Pr[\mathsf{AskG}_4] + \Pr[\mathsf{AskH}_5] \leq \mathsf{Succ}^{\mathsf{s-pd-ow}}(q_H, \tau) + \frac{q_G}{2^{k_0}}.$$

4.3.2. *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. The situation is more intricate than in the original paper [3]: 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.

Comments. One can easily 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.

Furthermore, if both r and s have been queried by the adversary, the plaintext-extractor perfectly simulates the decryption oracle.

4.3.3. *Semantic Security against Adaptive Chosen-Ciphertext Attacks.*

In the following, $y^\star$ is the challenge ciphertext, obtained from the encryption oracle. Since we have in mind using the plaintext-extractor instead of the decryption oracle, 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:

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

In the sequel, all unstarred variables refer to the decryption queries.

We now present a complete proof, which is an easy extension of the previous one, but makes use of the decryption oracle. We sequentially discard all cases for which the above plaintext-extractor may fail.

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 randomly chooses $r^+ \stackrel{R}{\leftarrow} \{0,1\}^{k_0}$ and $g^+ \stackrel{R}{\leftarrow} \{0,1\}^{k-k_0}$, and uses r^+ instead of $r^\star$, as well as g^+ instead of $G(r^\star)$. The game obeys the same rules as Game_1:

$$\Pr[S_1] = \Pr[S_0].$$

GAME_2**:** In this game, we drop the second rule of GAME_1. Then, as was the case for Game_2: $\Pr[S_2] = 1/2$, and

$$|\Pr[S_2] - \Pr[S_1]| \leq \Pr[\mathsf{AskG}_2],$$

where AskG_2 denotes the event that, in GAME_2, $r^\star$ is queried from G (by the adversary, or by the decryption oracle).

GAME_3**:** We now define $s^\star$ independently of anything else, as well as $H(s^\star)$, by randomly choosing $s^+ \stackrel{R}{\leftarrow} \{0,1\}^{k-k_0}$ and $h^+ \stackrel{R}{\leftarrow} \{0,1\}^{k_0}$, and using s^+ instead of $s^\star$, as well as h^+ instead of $H(s^\star)$. The game obeys the same rules as Game_3:

$$\Pr[\mathsf{AskG}_3] = \Pr[\mathsf{AskG}_2].$$

GAME_4**:** In this game, we drop the second rule of GAME_3. Then, as was the case for Game_4,

$$|\Pr[\mathsf{AskG}_4] - \Pr[\mathsf{AskG}_3]| \leq \Pr[\mathsf{AskH}_4],$$

where AskH_4 denotes the event that, in GAME_4, $s^\star$ is queried from H (by the adversary, or by the decryption oracle).

Furthermore, $r^\star = t^\star \oplus h^+$ is uniformly distributed, and independent of the adversary's view: $\Pr[\mathsf{AskG}_4] \leq (q_G + q_D)/2^{k_0}$, where q_G and q_D denote the number of queries asked by the adversary to G, or to the decryption oracle, respectively.

GAME_5**:** We manufacture the challenge ciphertext as in Game_5. We randomly choose $y^+ \stackrel{R}{\leftarrow} \{0,1\}^k$, and simply set $y^\star = y^+$. As before, we have:

$$\Pr[\mathsf{AskH}_5] = \Pr[\mathsf{AskH}_4].$$

We now deal with the decryption oracle, which has remained perfect up to this game.

GAME_6: We make the decryption oracle reject all ciphertexts y such that the corresponding r value has not been previously queried from G by the adversary. This makes a difference only if y is a valid ciphertext, while $G(r)$ has not been asked. Since $G(r)$ is uniformly distributed, equality $[s \oplus G(r)]_{k_1} = 0^{k_1}$ happens with probability $1/2^{k_1}$. Summing up for all decryption queries, we get

$$|\Pr[\mathsf{AskH}_6] - \Pr[\mathsf{AskH}_5]| \leq \frac{q_D}{2^{k_1}}.$$

GAME_7: We now make the decryption oracle reject all ciphertexts y such that the corresponding s value has not been previously queried from H by the adversary. This makes a difference only if y is a valid ciphertext, and r has been queried from G, while $H(s)$ has not been asked. Since $r = H(s) \oplus t$ is uniformly distributed, it has been queried from G with probability less than $q_G/2^{k_0}$ (note that in the previous game, the decryption oracle makes no additional query to G.) Summing up for all decryption queries, we get

$$|\Pr[\mathsf{AskH}_7] - \Pr[\mathsf{AskH}_6]| \leq \frac{q_D q_G}{2^{k_0}}.$$

GAME_8: We finally replace the decryption oracle by the plaintext-extractor which perfectly simulates the decryption, since both r and s have been previously queried:

$$\Pr[\mathsf{AskH}_8] = \Pr[\mathsf{AskH}_7].$$

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

$$\Pr[\mathsf{AskH}_8] \leq \mathsf{Succ}^{\mathsf{s-pd-ow}}(q_H, \tau').$$

Therefore,

$$\begin{aligned}
\frac{\varepsilon}{2} &= |\Pr[S_0] - \Pr[S_2]| \leq \Pr[\mathsf{AskG}_2] \leq \Pr[\mathsf{AskG}_4] + \Pr[\mathsf{AskH}_4] \leq \frac{q_G + q_D}{2^{k_0}} + \Pr[\mathsf{AskH}_5] \\
&\leq \frac{q_G + q_D}{2^{k_0}} + \frac{q_D}{2^{k_1}} + \Pr[\mathsf{AskH}_6] \leq \frac{q_G + q_D}{2^{k_0}} + \frac{q_D}{2^{k_1}} + \frac{q_D q_G}{2^{k_0}} + \Pr[\mathsf{AskH}_7] \\
&\leq \frac{q_G + q_D + q_D q_G}{2^{k_0}} + \frac{q_D}{2^{k_1}} + \mathsf{Succ}^{\mathsf{s-pd-ow}}(q_H, \tau').
\end{aligned}$$

To conclude the proof of Theorem 1, one just has to comment on the running time τ'. 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. Of course, a time-space tradeoff is possible, giving up the additional table, but raising the computing time to $q_D \cdot q_G \cdot q_H \cdot (T_f + \mathcal{O}(1))$.

5. Application to RSA–OAEP

The main application of OAEP is certainly the famous RSA–OAEP, which has been used to update the PKCS #1 standard [14]. In his paper [15], Shoup was able to repair the security result for a small exponent, $e = 3$, using Coppersmith's algorithm from [5]. However, our result can be applied to repair RSA–OAEP, regardless of the exponent; thanks to the random self-reducibility of RSA, the partial-domain one-wayness of RSA is equivalent to that of the whole RSA problem, as soon as a constant fraction of the most significant bits (or the least significant bits) of the pre-image can be recovered.

We note that, in the original RSA–OAEP [3], the most significant bits are involved in the H function, but in PKCS #1 standards v2.0 and v2.1 [14] and RFC2437, the least significant bits are used: the value $\mathsf{maskedSeed}\|\mathsf{maskedDB}$ is the input to f, the RSA function, where $\mathsf{maskedSeed}$ plays the role of t, and $\mathsf{maskedDB}$ the role of s. But it is clear that the following result holds in both situations (and can be further extended).

One may also remark that the following argument can be applied to any random (multiplicatively) self-reducible problem, such as the Rabin function. Before presenting the final reduction, let us consider the problem of finding small solutions for a linear modular equation.

Lemma 3. *Consider an equation $t + \alpha u = c \bmod N$ which has solutions t and u smaller than 2^{k_0}. For all values of $\alpha \in \{0, \ldots, N-1\}$, 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

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

We say that $L(\alpha)$ is an ℓ-good lattice (and that α is an ℓ-good value) if there is no non-zero vector of length at most ℓ (with respect to the Euclidean norm). 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. Furthermore, the above lattices have pairwise intersection limited to the single point $(0, 0)$, if $\ell < p$, where p is the smallest factor of N. 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

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

Let T be the point (t, u), where (t, u) is a solution of the equation $t + \alpha u = c \bmod N$, with both t and u less than 2^{k_0}: $T = \lambda U + \mu V$, for some real λ, μ.

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

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

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

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 the equation simply by picking a random integer u_0 and setting $t_0 = c - \alpha u_0 \bmod N$. Write it in the basis (U, V): $T_0 = \rho U + \sigma V$ using real numbers ρ and σ. These coordinates can be found, so $T - T_0$ is a solution to the homogeneous equation, and thus indicate a lattice point: $T - T_0 = aU + bV$, 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 4. *Let $\mathcal{A}$ be an algorithm that outputs a q-set containing $k - k_0$ of the most significant bits of the e-th root of its input (partial-domain RSA, for any $2^{k-1} < N < 2^k$, with $k > 2k_0$), within time bound t, with probability ε. There exists an algorithm $\mathcal{B}$ that solves the RSA problem (N, e) with success probability ε', within time bound t' where*

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

Proof. Thanks to the random self-reducibility of RSA, with part of the bits of the e-th root of $X = (x \cdot 2^{k_0} + r)^e \bmod N$, and the e-th root of $Y = X\alpha^e = (y \cdot 2^{k_0} + s)^e \bmod N$, for a randomly chosen α, one gets both x and y. Thus,

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

which is a linear modular equation with two unknowns r and s which is known to have small solutions (smaller than 2^{k_0}). It can be solved using lemma 3.

Algorithm $\mathcal{B}$ just runs twice $\mathcal{A}$, on inputs X and $X\alpha^e$ and next runs the Gaussian reduction on all the q^2 pairs of elements coming from both sets. If the partial pre-images are in the sets, they will be found, unless the random α is bad (*cf.* the Gaussian reduction in lemma 3.) □

Remark 1. The above lemma can be extended to the case where a constant fraction Θ of the leading or trailing bits of the e-th root is found. The reduction runs $1/\Theta$ times the adversary $\mathcal{A}$, and the success probability decreases to approximately $\varepsilon^{1/\Theta}$. Extensions to any constant fraction of consecutive bits are also possible. Anyway, in PKCS #1 v2.0, k_0 is much smaller than $k/2$.

Theorem 2. *Let $\mathcal{A}$ be a CCA2–adversary against the "semantic security" of RSA–OAEP (where the modulus is k-bit long, $k > 2k_0$), with running time bounded by t and advantage ε, making q_D, q_G and q_H queries to the decryption oracle, and the hash functions G and H respectively. Then, the RSA problem can be solved with probability ε' greater than*

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

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

Proof. Theorem 1 states that

$$\mathsf{Succ}^{\mathsf{s-pd-ow}}(q_H, \tau) \geq \frac{\varepsilon}{2} - \frac{q_D q_G + q_D + q_G}{2^{k_0}} - \frac{q_D}{2^{k_1}},$$

with $\tau \leq t + q_G \cdot q_H \cdot (T_f + \mathcal{O}(1))$, and $T_f = \mathcal{O}(k^3)$. Using the previous results relating q_H-set partial-domain–RSA and RSA, we easily conclude. □

Remark 2. There is a slight inconsistency in piecing together the results from Sections 4. and 5., 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. Observe that one may decide to only encode message of $n - 8$ bits, where n is $k - k_0 - k_1$ as before, as it is done in PKCS #1 standard. 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, 256 trials will be enough.

6. Improved Security Result

We can improve a little bit the reduction cost in the above theorem. More precisely:

Theorem 3. *Let $\mathcal{A}$ be a CCA2–adversary against the "semantic security" of RSA–OAEP (where the modulus is k-bit long, $k > 2k_0$), with running time bounded by t and advantage ε, making q_D, q_G and q_H queries to the decryption oracle, and the hash functions G and H respectively. Then, the RSA problem can be solved with probability ε' greater than*

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

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

This theorem comes from the lemma stated below, which is proved in the appendix.

Lemma 5. *Let $\mathcal{A}$ be a CCA2–adversary against the "semantic security" of the OAEP conversion $(\mathcal{K}, \mathcal{E}, \mathcal{D})$, with advantage ε and running time t, making q_D, q_G and q_H queries to the decryption oracle, and the hash functions G and H respectively. Then,* $\mathsf{Succ}^{\mathsf{s-pd-ow}}(q_H, t')$ *is greater than*

$$\varepsilon - \frac{2q_D q_G + q_D + q_G}{2^{k_0}} - \frac{2q_D}{2^{k_1}},$$

where $t' \leq t + q_G \cdot q_H \cdot (T_f + \mathcal{O}(1))$, and T_f denotes the time complexity of function f.

7. Conclusion

Our conclusion is that one can still trust the security of RSA–OAEP, but the reduction is more costly than the original one. However, for other OAEP applications, more care is needed, since the security does not actually rely on the one-wayness of the permutation, only on its partial-domain one-wayness.

Acknowledgments

We thank Dan Boneh, Don Coppersmith, Victor Shoup and the anonymous referee for fruitful comments. We are especially grateful to the referee for clarifying our proof of theorem 1.

References

1. 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.
2. 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.
3. 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.
4. 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.
5. D. Coppersmith. Finding a Small Root of a Univariate Modular Equation. In *Eurocrypt '96*, LNCS 1070, pages 155–165. Springer-Verlag, Berlin, 1996.
6. D. Dolev, C. Dwork, and M. Naor. Non-Malleable Cryptography. *SIAM Journal on Computing*, 30(2):391–437, 2000.
7. 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.
8. S. Goldwasser and S. Micali. Probabilistic Encryption. *Journal of Computer and System Sciences*, 28:270–299, 1984.
9. 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.
10. 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.
11. 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.
12. 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.
13. 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.
14. RSA Data Security, Inc. Public Key Cryptography Standards – PKCS. Available from `http://www.rsa.com/rsalabs/pubs/PKCS/`.
15. V. Shoup. OAEP Reconsidered. In *Crypto '2001*, LNCS 2139, pages 239–259. Springer-Verlag, Berlin, 2001.

A Proof of Lemma 5

The next section is devoted to proving this lemma. Hereafter, we will repeatedly use the following simple result:

Lemma 6. *For any probability events* E, F *and* G

$$\Pr[\mathsf{E} \wedge \mathsf{F} \mid \mathsf{G}] \leq \begin{cases} \Pr[\mathsf{E} \mid \mathsf{F} \wedge \mathsf{G}] \\ \Pr[\mathsf{F} \mid \mathsf{G}]. \end{cases}$$

We prove lemma 5 in three stages. The first presents the reduction of an $\mathsf{IND\text{-}CCA2}$ adversary $\mathcal{A}$ to an algorithm $\mathcal{B}$ for breaking the partial-domain one-wayness of f. The second shows that there exists a plaintext-extractor which correctly simulates the decryption oracle, with overwhelming probability, under the partial-domain one-wayness of f. Finally, we analyze the success probability of our reduction in total, through the incorporation of the above-mentioned analysis of the plaintext-extractor.

A1. *Description of the Reduction*

In this first part, we recall how reduction operates. Let $\mathcal{A} = (A_1, A_2)$ be an adversary against the semantic security of $(\mathcal{K}, \mathcal{E}, \mathcal{D})$, under chosen-ciphertext attacks. Within time bound τ, $\mathcal{A}$ asks q_D, q_G and q_H queries to the decryption oracle and the random oracles G and H respectively, and distinguishes the right plaintext with an advantage greater than ε. Let us describe the reduction $\mathcal{B}$.

A1.1. *Top Level Description of the Reduction.*

1. $\mathcal{B}$ is given a function f (defined by the public key) and $y^\star \leftarrow f(s^\star, t^\star)$, for $(s^\star, t^\star) \stackrel{R}{\leftarrow} \{0,1\}^{k-k_0} \times \{0,1\}^{k_0}$. The aim of $\mathcal{B}$ is to recover the partial pre-image $s^\star$ of $y^\star$.
2. $\mathcal{B}$ runs A_1 on the public data, and gets a pair of messages $\{m_0, m_1\}$ as well as state information st. It chooses a random bit b, and then gives $y^\star$ to A_1, as the ciphertext of m_b. $\mathcal{B}$ simulates the answers to the queries of A_1 to the decryption oracle and random oracles G and H respectively. See the description of these simulations below.
3. $\mathcal{B}$ runs $A_2(y^\star, st)$ and finally gets answer b'. $\mathcal{B}$ simulates the answers to the queries of A_2 to the decryption oracle and random oracles G and H respectively. See the description of these simulations below. $\mathcal{B}$ then outputs the partial pre-image $s^\star$ of $y^\star$, if one has been found among the queries asked to H (see below), or the list of queries asked to H.

A1.2. *Simulation of Random Oracles G and H.*

The random oracle simulation has to simulate the random oracle answers, managing query/answer lists $\mathsf{G\text{-}List}$ and $\mathsf{H\text{-}List}$ for the oracles G and H respectively, both are initially set to empty lists:

- for a fresh query γ to G, one looks at the H-List, and for any query δ asked to H with answer H_δ, one builds $z = \gamma \oplus H_\delta$, and checks whether $y^\star = f(\delta, z)$. If for some δ, that relation holds, function f has been inverted, and we can still correctly simulate G, by answering $G_\gamma = \delta \oplus (m_b \| 0^{k_1})$. Note that G_γ is then a uniformly distributed value since $\delta = s^\star$, and the latter is uniformly distributed. Otherwise, one outputs a random value G_γ. In both cases, the pair (γ, G_γ) is concatenated to the G-List.
- for a fresh query δ to H, one outputs a random value H_δ, and the pair (δ, H_δ) is concatenated to the H-List. Note that, once again, for any $(\gamma, G_\gamma) \in$ G-List, one may build $z = \gamma \oplus H_\delta$, and check whether $y^\star = f(\delta, z)$. If for some γ that relation holds, we have inverted the function f.

A1.3. *Simulation of the Decryption Oracle.*

We refer the reader to section 4.3.2., since the simulation works exactly the same way.

A1.4. *Remarks.*

When we have found the pre-image of $y^\star$, and thus inverted f, we could output the expected result $s^\star$ and stop the reduction. But for this analysis, we assume the reduction goes on and that $\mathcal{B}$ only outputs it, or the list of queries asked to H, once A_2 has answered b' (or after a time limit).

Even if no answer is explicitly specified, except by a random value for new queries, some are implicitly defined. Indeed, $y^\star$ is defined to be a ciphertext of m_b with random tape $r^\star$, thus $r^\star \leftarrow H(s^\star) \oplus t^\star$ and $G(r^\star) \leftarrow s^\star \oplus (m_b \| 0^{k_1})$.

Since $H(s^\star)$ is randomly defined, $r^\star$ can be seen as a random variable. Let us denote by AskG the event that query $r^\star$ has been asked to G, and by AskH the event that query $s^\star$ has been asked to H. Let us furthermore denote by GBad the event that $r^\star$ has been asked to G, but the answer is something other than $s^\star \oplus (m_b \| 0^{k_1})$ (bit b is fixed in the reduction scenario). Note that the event GBad implies AskG. One may remark that GBad is the only event that makes the random oracle simulation imperfect, in the chosen-plaintext attack scenario. In the chosen-ciphertext attack scenario, we described a decryption simulator that may sometimes fail. Such an event of decryption failure will be denoted by DBad. We thus denote Bad = GBad $\vee$ DBad.

A2. *Notations*

In order to proceed with the analysis of the success probability of the above reduction, one needs to set up notations. First, we still denote with a star (*) all variables related to the challenge ciphertext $y^\star$, obtained from the encryption oracle. Indeed, this ciphertext, of either m_0 or m_1, implicitly defines hash values, but the corresponding pairs may not appear in the G or H lists. All other (unstarred) variables refer to the decryption query y, asked by the adversary to the decryption oracle, and thus to be decrypted by the simulator. We consider several further events about a ciphertext queried to the decryption oracle:

– CBad denotes the union of the bad events, $\mathsf{CBad} = \mathsf{RBad} \vee \mathsf{SBad}$, where
  - SBad denotes the event that $s = s^\star$;
  - RBad denotes the event that $r = r^\star$, and thus $H(s) \oplus t = H(s^\star) \oplus t^\star$;

– AskRS denotes the intersection of both events about the oracle queries, $\mathsf{AskRS} = \mathsf{AskR} \wedge \mathsf{AskS}$, which means that both r and s have been asked to G and H respectively, since
  - AskR denotes the event that r $(= H(s) \oplus t)$ has been asked to G;
  - AskS denotes the event that s has been asked to H;

– Fail denotes the event that the above decryption oracle simulator outputs a wrong decryption answer to query y. (More precisely, we let Fail_i denote the instantiation of Fail on the i-th query y_i $(i = 1, \ldots, q_D)$. For our analysis, however, we can evaluate probabilities regarding event Fail_i in a uniform manner for any i. Hence, we just employ notation Fail.) Therefore, in the global reduction, the event DBad will be set to true as soon as one decryption simulation fails.

Note that the Fail event is limited to the situation in which the plaintext-extractor rejects a ciphertext whereas it would be accepted by the actual decryption oracle. Indeed, as soon as it accepts, we see that the ciphertext is actually valid and corresponds to the output plaintext.

A3. *Analysis of the Decryption Oracle Simulation*

We analyze the success probability of decryption oracle simulator $\mathcal{PE}$.

A3.1. *Security Claim.*

We claim the following, which repairs the previous proof [3], based on the new computational assumption. More precisely, we show that additional cases to consider, due to the corrected definition of plaintext-awareness [1], are very unlikely under the partial-domain one-wayness of the permutation f:

Lemma 7. *When at most one ciphertext $y^\star = f(s^\star, t^\star)$ has been directly obtained from the encryption oracle, but $s^\star$ has not been asked to H, the plaintext extractor correctly produces the decryption oracle's output on query (ciphertext) y $(\neq y^\star)$ with probability greater than ε', within time bound τ', where*

$$\varepsilon' \geq 1 - \left(\frac{2}{2^{k_1}} + \frac{2q_G + 1}{2^{k_0}} \right) \text{ and } \tau' \leq q_G \cdot q_H \cdot (T_f + \mathcal{O}(1)).$$

We refer the reader to section 4.3.2. for a discussion about the plaintext-extractor. We just insist on the fact that if the ciphertext has been correctly built by the adversary (r has been asked to G and s to H), the simulation will output the correct answer. However, it will output "Reject" in any other situation, whereas the adversary may have built a valid ciphertext without asking both queries to the random oracles G and H.

A3.2. *Success Probability.*

Since our goal is to prove the security relative to the partial-domain one-wayness of f, we are only interested in the probability of the event Fail, while $\neg\mathsf{AskH}$ occurred, which may be split according to other events. Granted $\neg\mathsf{CBad} \wedge \mathsf{AskRS}$, the simulation is perfect, and cannot fail. Thus, we have to consider the complementary events:

$$\Pr[\mathsf{Fail} \mid \neg\mathsf{AskH}] = \Pr[\mathsf{Fail} \wedge \mathsf{CBad} \mid \neg\mathsf{AskH}] + \Pr[\mathsf{Fail} \wedge \neg\mathsf{CBad} \wedge \neg\mathsf{AskRS} \mid \neg\mathsf{AskH}].$$

Concerning the second contribution to the right hand side, we first note that both

$$\begin{aligned} \neg\mathsf{AskRS} &= \neg\mathsf{AskR} \vee \neg\mathsf{AskS} = (\neg\mathsf{AskR}) \vee (\neg\mathsf{AskS} \wedge \mathsf{AskR}) \\ \neg\mathsf{CBad} &= \neg\mathsf{RBad} \wedge \neg\mathsf{SBad}. \end{aligned}$$

Forgetting $\neg\mathsf{AskH}$ for a while, using lemma 6, one gets that the probability $\Pr[\mathsf{Fail} \wedge \neg\mathsf{CBad} \wedge \neg\mathsf{AskRS}]$ is less than

$$\begin{aligned} &\Pr[\mathsf{Fail} \wedge \neg\mathsf{RBad} \wedge \neg\mathsf{AskR}] + \Pr[\mathsf{Fail} \wedge \neg\mathsf{SBad} \wedge (\mathsf{AskR} \wedge \neg\mathsf{AskS})] \\ &\leq \Pr[\mathsf{Fail} \mid \neg\mathsf{AskR} \wedge \neg\mathsf{RBad}] + \Pr[\mathsf{AskR} \mid \neg\mathsf{AskS} \wedge \neg\mathsf{SBad}]. \end{aligned}$$

But without having asked r to G, taking into account the further event $\neg\mathsf{RBad}$, $G(r)$ is unpredictable, and thus the probability that $[s \oplus G(r)]_{k_1} = 0^{k_1}$ is less than 2^{-k_1}. On the other hand, the probability of having asked r to G, without any information about $H(s)$ and thus about r ($H(s)$ not asked, and $s \neq s^\star$, which both come from the conditioning $\neg\mathsf{AskS} \wedge \neg\mathsf{SBad}$), is less than $q_G \cdot 2^{-k_0}$. Furthermore, this event is independent of AskH, which yields

$$\Pr[\mathsf{Fail} \wedge \neg\mathsf{CBad} \wedge \neg\mathsf{AskRS} \mid \neg\mathsf{AskH}] \leq 2^{-k_1} + q_G \cdot 2^{-k_0}.$$

We now focus on the first contribution to the right hand side, $\mathsf{Fail} \wedge \mathsf{CBad}$, while $\neg\mathsf{AskH}$, which was missing in the original proof [3] based on a weaker notion of plaintext-awareness. It can be split according to the disjoint sub-cases of CBad, which are SBad and $\neg\mathsf{SBad} \wedge \mathsf{RBad}$. Then again using lemma 6,

$$\Pr[\mathsf{Fail} \wedge \mathsf{CBad} \mid \neg\mathsf{AskH}] \leq \Pr[\mathsf{Fail} \mid \mathsf{SBad} \wedge \neg\mathsf{AskH}] + \Pr[\mathsf{RBad} \mid \neg\mathsf{SBad} \wedge \neg\mathsf{AskH}].$$

The latter event means that RBad occurs provided $s \neq s^\star$ and the adversary has not queried $s^\star$ from H. When $s^\star$ has not been asked to H, and $s \neq s^\star$, $H(s^\star)$ is unpredictable and independent of $H(s)$, as well as t and $t^\star$. Then, event RBad, $H(s^\star) = H(s) \oplus t \oplus t^\star$, occurs with probability at most 2^{-k_0}.

The former event can be further split according to AskR, and, using once again lemma 6, it is upper-bounded by

$$\Pr[\mathsf{AskR} \mid \mathsf{SBad} \wedge \neg\mathsf{AskH}] + \Pr[\mathsf{Fail} \mid \neg\mathsf{AskR} \wedge \mathsf{SBad} \wedge \neg\mathsf{AskH}].$$

The former event means that r is asked to G whereas $s = s^\star$ and $H(s^\star)$ is unpredictable, thus $H(s)$ is unpredictable. Since r is unpredictable, the probability of this event is at most $q_G \cdot 2^{-k_0}$ (the probability of asking r to G). On the other hand, the latter event means that the simulator rejects the valid ciphertext y whereas $H(s)$ is unpredictable and r is not asked to G. From the one-to-one property of the Feistel network, it follows from $s = s^\star$ that $r \neq r^\star$, and thus $G(r)$ is unpredictable. Then the redundancy cannot hold with probability greater than 2^{-k_1}. To sum up, $\Pr[\mathsf{Fail} \mid \mathsf{SBad} \wedge \neg\mathsf{AskH}] \leq 2^{-k_1} + q_G \cdot 2^{-k_0}$, thus $\Pr[\mathsf{Fail} \wedge \mathsf{CBad} \mid \neg\mathsf{AskH}] \leq 2^{-k_1} + (q_G+1) \cdot 2^{-k_0}$.

As a consequence,

$$\Pr[\mathsf{Fail} \mid \neg\mathsf{AskH}] \leq \frac{2}{2^{k_1}} + \frac{2q_G+1}{2^{k_0}}.$$

The running time of this simulator includes just the computation of $f(\sigma, \theta)$ for all possible pairs and is thus bounded by $q_G \cdot q_H \cdot (T_f + \mathcal{O}(1))$.

A4. *Success Probability of the Reduction*

This subsection analyzes the success probability of our reduction with respect to the advantage of the IND-CCA2 adversary. The goal of the reduction is, given $y^\star = f(s^\star, t^\star)$, to obtain $s^\star$. Therefore, the success probability is obtained by the probability that event AskH occurs during the reduction (*i.e.*, $\Pr[\mathsf{AskH}] \leq \mathsf{Succ}^{\mathsf{s-pd-ow}}(q_H, t')$, where t' is the running time of the reduction).

We thus evaluate $\Pr[\mathsf{AskH}]$ by splitting event AskH according to event Bad.

$$\Pr[\mathsf{AskH}] = \Pr[\mathsf{AskH} \wedge \mathsf{Bad}] + \Pr[\mathsf{AskH} \wedge \neg\mathsf{Bad}].$$

Let us evaluate the first term, using lemma 6 and that GBad implies AskG.

$$\begin{aligned}
\Pr[\mathsf{AskH} \wedge \mathsf{Bad}] &= \Pr[\mathsf{Bad}] - \Pr[\neg\mathsf{AskH} \wedge \mathsf{Bad}] \\
&\geq \Pr[\mathsf{Bad}] - \Pr[\neg\mathsf{AskH} \wedge \mathsf{GBad}] - \Pr[\neg\mathsf{AskH} \wedge \mathsf{DBad}] \\
&\geq \Pr[\mathsf{Bad}] - \Pr[\mathsf{AskG} \mid \neg\mathsf{AskH}] - \Pr[\mathsf{DBad} \mid \neg\mathsf{AskH}] \\
&\geq \Pr[\mathsf{Bad}] - \frac{2q_D q_G + q_D + q_G}{2^{k_0}} - \frac{2q_D}{2^{k_1}}.
\end{aligned}$$

Here, $\Pr[\mathsf{DBad} \mid \neg\mathsf{AskH}] \leq q_D \left(2 \cdot 2^{-k_1} + (2q_G+1) \cdot 2^{-k_0}\right)$ is directly obtained from lemma 7. When $\neg$AskH occurs, $H(s^\star)$ is unpredictable, and $r^\star = t^\star \oplus H(s^\star)$ is also unpredictable. Hence $\Pr[\mathsf{AskG} \mid \neg\mathsf{AskH}] \leq q_G \cdot 2^{-k_0}$.

We then evaluate the second term.

$$\begin{aligned}
\Pr[\mathsf{AskH} \wedge \neg\mathsf{Bad}] &\geq \Pr[\mathcal{A} = b \wedge \mathsf{AskH} \wedge \neg\mathsf{Bad}] \\
&= \Pr[\mathcal{A} = b \wedge \neg\mathsf{Bad}] - \Pr[\mathcal{A} = b \wedge \neg\mathsf{AskH} \wedge \neg\mathsf{Bad}].
\end{aligned}$$

Here, when $\neg$AskH occurs, $H(s^\star)$ is unpredictable, thus $r^\star = t^\star \oplus H(s^\star)$ is unpredictable, and so is b as well. This fact is independent from event $\neg\mathsf{AskH} \wedge \neg\mathsf{Bad}$. In addition,

$$\Pr[\mathsf{Bad}] + (\Pr[\mathsf{AskH} \wedge \neg\mathsf{Bad}] + \Pr[\neg\mathsf{AskH} \wedge \neg\mathsf{Bad}]) = 1.$$

Let $P_A = \Pr[\mathsf{AskH} \wedge \neg\mathsf{Bad}]$, hence

$$\begin{aligned}
\Pr[\mathcal{A} = b \wedge \neg\mathsf{AskH} \wedge \neg\mathsf{Bad}] &= \Pr[\neg\mathsf{AskH} \wedge \neg\mathsf{Bad}] \cdot \Pr[\mathcal{A} = b \mid \neg\mathsf{AskH} \wedge \neg\mathsf{Bad}] \\
&= (1 - P_A - \Pr[\mathsf{Bad}]) \cdot \frac{1}{2}.
\end{aligned}$$

Furthermore,

$$\Pr[\mathcal{A} = b \wedge \neg\mathsf{Bad}] \geq \Pr[\mathcal{A} = b] - \Pr[\mathsf{Bad}] = \frac{\varepsilon}{2} + \frac{1}{2} - \Pr[\mathsf{Bad}].$$

Therefore,

$$\begin{aligned} P_A = \Pr[\mathsf{AskH} \wedge \neg\mathsf{Bad}] &\geq \frac{\varepsilon}{2} + \frac{1}{2} - \Pr[\mathsf{Bad}] - (1 - P_A - \Pr[\mathsf{Bad}]) \cdot \frac{1}{2} \\ &= \frac{\varepsilon + P_A - \Pr[\mathsf{Bad}]}{2}. \end{aligned}$$

That is, $P_A = \Pr[\mathsf{AskH} \wedge \neg\mathsf{Bad}] \geq \varepsilon - \Pr[\mathsf{Bad}]$.

Combining the evaluation for the first and second terms, one finally gets

$$\Pr[\mathsf{AskH}] \geq \varepsilon - \frac{2q_D q_G + q_D + q_G}{2^{k_0}} - \frac{2q_D}{2^{k_1}}.$$