Full version of the extended abstract, which appeared in
Advances in Cryptology – Proceedings of ASIACRYPT '2004 (5 – 9 december 2004, Jeju Island, South Korea)
P. J. Lee Ed. Springer-Verlag, LNCS ???, pages ???–???.

OAEP 3-Round

A Generic and Secure Asymmetric Encryption Padding

Duong Hieu Phan and David Pointcheval

École normale supérieure – Dépt d'informatique
45 rue d'Ulm, 75230 Paris Cedex 05, France.
{duong.hieu.phan,david.pointcheval}@ens.fr

Abstract. The OAEP construction is already 10 years old and well-established in many practical applications. But after some doubts about its actual security level, four years ago, the first efficient and provably IND-CCA1 secure encryption padding was formally and fully proven to achieve the expected IND-CCA2 security level, when used with any trapdoor permutation. Even if it requires the partial-domain one-wayness of the permutation, for the main application (with the RSA permutation family) this intractability assumption is equivalent to the classical (full-domain) one-wayness, but at the cost of an extra quadratic-time reduction. The security proof which was already not very tight to the RSA problem is thus much worse.
However, the practical optimality of the OAEP construction is two-fold, hence its attractivity: from the efficiency point of view because of two extra hashings only, and from the length point of view since the ciphertext has a minimal bit-length (the encoding of an image by the permutation.) But the bandwidth (or the ratio ciphertext/plaintext) is not optimal because of the randomness (required by the semantic security) and the redundancy (required by the plaintext-awareness, the sole way known to provide efficient CCA2 schemes.)
At last Asiacrypt '03, the latter intuition had been broken by exhibiting the first IND-CCA2 secure encryption schemes without redundancy, and namely without achieving plaintext-awareness, while in the random-oracle model: the OAEP 3-round construction. But this result achieved only similar practical properties as the original OAEP construction: the security relies on the partial-domain one-wayness, and needs a trapdoor permutation, which limits the application to RSA, with still a quite bad reduction.
This paper improves this result: first we show the OAEP 3-round actually relies on the (full-domain) one-wayness of the permutation (which improves the reduction), then we extend the application to a larger class of encryption primitives (including ElGamal, Paillier, etc.) The extended security result is still in the random-oracle model, and in a relaxed CCA2 model (which lies between the original one and the replayable CCA scenario.)

Keywords: OAEP, Asymmetric encryption, semantic security, chosen-ciphertext security, random oracle model.

1 Introduction

The OAEP construction [4, 12, 13] is now well-known and widely used, since it is an efficient and secure padding. However, the latter property had been recently called into question: indeed, contrarily to the widely admitted result, the security cannot be based on the sole one-wayness of the permutation [27], but the partial-domain one-wayness [12, 13]. For an application to RSA, the main trapdoor one-way permutation, the two problems are equivalent, but the security reduction is much worse than believed, because of a quadratic reduction between the two above problems.

There is also a second drawback of the OAEP construction, since its use is limited to permutations. It can definitely not apply to any function, as tried and failed on the NTRU primitive [15].

Finally, the optimality, as claimed in the name of the construction, is ambiguous and not clear: from the efficiency point of view, the extra cost for encryption and decryption is just two more hashings which is indeed quite good. But the most important optimality was certainly from the length point of view: the ciphertext is just an image by the permutation, and thus the shortest as possible. However, another important parameter is the bandwidth, or the ratio ciphertext/plaintext, which is not optimal: the construction requires a randomness over $2k$ bits for a semantic security in 2^{-k}, and redundancy over k bits for preventing chosen-ciphertext attacks (plaintext-awareness): the ciphertext is thus at least $3k$ bits as large as the plaintext.

1.1 Related Work

Right after the Shoup's remark about the security of OAEP [27], several alternatives to OAEP have been proposed: OAEP+ (by Shoup himself) and SAEP, SAEP+ (by Boneh [6]) but either the bandwidth, or the reduction cost remain pretty bad. Furthermore, their use was still limited to permutations.

About generic paddings applicable to more general encryption primitives, one had to wait five years after the OAEP proposal to see the first efficient suggestions: Fujisaki–Okamoto [10, 11] proposed the first constructions, then Pointcheval [22] suggested one, and eventually Okamoto–Pointcheval [18] introduced the most efficient construction, called REACT. However, all these proposals are far to be optimal for the ciphertext size. They indeed apply, in the random-oracle model, the general approach of symmetric and asymmetric components integration [26]: an ephemeral key is first encrypted using key-encapsulation, then this key is used on the plaintext with a symmetric encryption scheme (which is either already secure against chosen-ciphertext attacks, or made so by appending a MAC – or a tag with a random oracle, for achieving plaintext-awareness.)

Plaintext-awareness [4, 3] was indeed the essential ingredient to achieve IND-CCA2 security in the random-oracle model: it makes the simulation of the decryption oracle quite easy, by rejecting almost all the decryption queries, unless the plaintext is clearly known. But this property reduces the bandwidth since "unnecessary" redundancy is introduced. Randomness is required for the semantic security, but this is the sole mandatory extra data for constructing a secure ciphertext. At last Asiacrypt [21], the first encryption schemes with just such a randomness, but no redundancy, has been proposed: plaintext-awareness is no longer achieved, since any ciphertext is valid and corresponds to a plaintext. But this does not exclude the IND-CCA2 security level. In that paper [21], we indeed proved that an extension of OAEP, with 3 rounds but without redundancy, provides an IND-CCA2 secure encryption scheme, with any trapdoor permutation, but again under the partial-domain one-wayness. Hence a bad security reduction.

Note 1. The classical OAEP [4] construction can be seen as a 2-round Feistel network, while our proposal [21] was a 3-round network, hence the name *OAEP 3-round*. By the way, one should notice that SAEP [6] can be seen as a 1-round Feistel network.

1.2 Achievements

In this paper, we address the two above problems: the bad security reduction of the OAEP constructions, because of the need of the intractability of the partial-domain one-wayness; and the restriction to permutations.

First, we show that, contrarily to the OAEP (2-round) construction which cannot rely on the (full-domain) one-wayness, the OAEP 3-round simply requires the (full-domain) one-wayness: because of the third round, the adversary looses any control on the r value. It is not able to make ciphertexts with the same r, without querying it.

Then, we extend the application of OAEP 3-round to a larger class of encryption primitives: it applies to any efficiently computable probabilistic injection $f : \mathrm{E} \times \mathrm{R} \to \mathrm{F}$, which maps any $x \in \mathrm{E}$ into F in a probabilistic way according to the random string $\rho \in \mathrm{R}$. We need this function to be one-way: given $y \in \mathrm{F}$, it must be hard to recover $x \in \mathrm{E}$ (we do not mind about the random string ρ); this probabilistic function also needs to satisfy uniformity properties which are implied by a simple requirement: f is a bijection from $\mathrm{E} \times \mathrm{R}$ onto F. Some additional restrictions will appear in the security proof:

- we cannot really consider the CCA2 scenario, but a relaxed one denoted RCCA, which is between the usual one and the replayable CCA2 introduced last year [7] and considered enough in many applications.
- the simulation will need a decisional oracle which checks whether two elements in F have the same pre-images in E. The security result will thus be related to the well-known gap-problems [19, 18].

This extension allows *almost* optimal bandwidths for many very efficient asymmetric encryption schemes, with an IND-RCCA security level related to gap-problems (e.g. an ElGamal variant related to the Gap Diffie-Hellman problem.) Note that the application to trapdoor one-way permutations like RSA results in a much more efficient security result, and provides an IND-CCA2 encryption scheme under the sole one-wayness intractability assumption.

This paper is then organized as follows: in the next section, we review the classical security model for asymmetric encryption, and present our new CCA-variant. In section 3, we present the OAEP 3-round construction for any probabilistic injection, with some concrete applications. The security result is presented and proven in section 4.

2 Security Model

In this section, we review the security model widely admitted for asymmetric encryption. Then, we consider some relaxed CCA-variants. First, let us briefly remind that a public-key encryption scheme $\mathcal{S}$ is defined by three algorithms: the key generation algorithm $\mathcal{K}(1^k)$, which produces a pair of matching public and private keys $(\mathsf{pk}, \mathsf{sk})$; the encryption algorithm $\mathcal{E}_{\mathsf{pk}}(m; r)$ which outputs a ciphertext c corresponding to the plaintext $m \in \mathcal{M}$, using random coins $r \in \mathcal{R}$; and the decryption algorithm $\mathcal{D}_{\mathsf{sk}}(c)$ which outputs the plaintext m associated to the ciphertext c.

2.1 Classical Security Notions

Beyond *one-wayness*, which is the basic security level for an encryption scheme, it is now well-admitted to require *semantic security* (*a.k.a. polynomial security* or *indistinguishability of encryptions* [14], denoted IND): if the attacker has some *a priori* information about the plaintext, it should not learn more with the view of the ciphertext. More formally, this security notion requires the computational indistinguishability between two messages, chosen by the adversary, one of which has been encrypted, which one has been actually encrypted with a probability significantly better than one half:

the advantage $\mathsf{Adv}^{\mathsf{ind}}_{\mathcal{S}}(\mathcal{A})$, where the adversary $\mathcal{A}$ is seen as a 2-stage Turing machine (A_1, A_2), should be negligible, where $\mathsf{Adv}^{\mathsf{ind}}_{\mathcal{S}}(\mathcal{A})$ is formally defined as

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

Stronger security notions have also been defined thereafter (namely the *non-malleability* [8]), but we won't deal with it since it is similar to the semantic security in several scenarios [3, 5].

On the other hand, an attacker can use many kinds of attacks, according to the available information: since we are considering asymmetric encryption, the adversary can encrypt any plaintext of its choice with the public key, hence the basic *chosen-plaintext attack*. But the strongest attack is definitely when the adversary has an unlimited access to the decryption oracle itself, *adaptive chosen-ciphertext attacks* [24], denoted CCA or CCA2 (by opposition to the earlier *lunchtime attacks* [17], denoted CCA1, where this oracle access is limited until the challenge is known.) From now, we simply use CCA instead of CCA2 since we focus on adaptive adversaries.

The strongest security notion that we now widely consider is the *semantic security against adaptive chosen-ciphertext attacks* denoted IND-CCA —where the adversary just wants to distinguish which plaintext, between two messages of its choice, had been encrypted; it can ask any query to a decryption oracle (except the challenge ciphertext).

2.2 Relaxed CCA-Security

First, at Eurocrypt '02, An *et al* [1] proposed a "generalized CCA" security notion, where the adversary is restricted not to ask, to the decryption oracle, ciphertexts which are in *relation* with the challenge ciphertext. This relation must be an equivalence relation, publicly and efficiently computable, and decryption-respecting: if two ciphertexts are in relation, they necessarily encrypt identical plaintexts. This relaxation was needed in that paper, so that extra bits in the ciphertext, which can be easily added or suppressed, should not make the scheme theoretical insecure, while its security is clearly the same from a practical point of view.

More recently, another relaxation (an extra one beyond the above one) has been proposed by Canetti *et al* [7]: informally, it extends the above relation to the (possibly non-computable) equality of plaintexts. More precisely, if the adversary asks for a ciphertext c to the decryption oracle, c is first decrypted into m. Then, if m is one of the two plaintexts output in the first stage by the adversary, the decryption oracle returns test, otherwise the actual plaintext m is returned. They called this variant the "replayable CCA" security. They explain that this security level, while clearly weaker than the usual CCA one, is enough in most of the practical applications. The classical CCA security level is indeed very strong, *too strong* for the same reasons as explained above for the first relaxation.

In this paper, we could work with the latter relaxation, the "replayable CCA" scenario. But for a simpler security proof, as well as a more precise security result (with nice corollaries for particular cases, such as the RSA one) we restrict it a little bit into the "relaxed CCA" scenario, denoted RCCA. A scheme which is secure in this scenario is trivially secure in the "replayable CCA" one, but not necessarily in the "generalized CCA" or the usual CCA scenario. The actual relations between these scenarios depend on the way the random string is split. In the formal notation of

the encryption algorithm, we indeed split the randomness in two parts r and ρ: $c = \mathcal{E}_{\mathsf{pk}}(m; r, \rho)$. The encryption algorithm is thus a function from $\mathcal{M} \times \mathcal{R} \times \mathrm{R}$ into the ciphertext set. We know that for being an encryption scheme, this function must be an injection with respect to $\mathcal{M}$ (several elements in $\mathcal{M} \times \mathcal{R} \times \mathrm{R}$ can map to the same ciphertext, but all these elements must project uniquely on $\mathcal{M}$: the plaintext.) In our new relaxation, we split the randomness in $\mathcal{R} \times \mathrm{R}$ so that this function is also an injection with respect to $\mathcal{M} \times \mathcal{R}$.

Let us assume that the challenge ciphertext is $c^\star = \mathcal{E}_{\mathsf{pk}}(m^\star; r^\star, \rho^\star)$. Let us consider the ciphertext $c = \mathcal{E}_{\mathsf{pk}}(m; r, \rho)$. According to the above comment, $(m^\star, r^\star)$ and (m, r) are uniquely defined from $c^\star$ and c respectively, while $\rho^\star$ and ρ may not be unique. Upon receiving c, the *relaxed decryption oracle* first checks whether $(m^\star, r^\star) = (m, r)$ in which case it outputs test. Otherwise, it outputs m.

Definition 2 (Relaxed CCA). In the "relaxed CCA" scenario, an adversary has an unlimited access to the relaxed decryption oracle.

Property 3. Security in the "relaxed CCA" scenario implies security in the "replayable CCA" one.

Proof. As already noticed, this is a trivial relation, since the decryption oracle in the latter scenario can be easily simulated by the relaxed decryption oracle: if its output is test, this value is forwarded, else the returned plaintext m is compared to the output of the adversary at the end of the first stage. According to the result of the comparison, either a test-answer is also given (if $m \in \{m_0, m_1\}$), or m.

This property was just to make clear that we do not relax more the CCA security, but still keep it beyond what is clearly acceptable for practical use. Namely, note that if R is the empty set, then the RCCA scenario is exactly the usual CCA one: if f is a permutation from E onto F (the RSA case.)

3 OAEP 3-Round: A General and Efficient Padding

3.1 The Basic Primitive

Our goal is to prove that OAEP 3-round can be used with a large class of one-way functions. More precisely, we need an injective *probabilistic* trapdoor one-way function family $(\varphi_{\mathsf{pk}})_{\mathsf{pk}}$ from a set E_{pk} to a set F_{pk}, respectively to the index pk: almost any encryption primitive, where the plaintext set is denoted E_{pk} and the ciphertext set is denoted F_{pk}, is fine: for any parameter pk (the public key), there exists the inverse function ψ_{sk} (where sk is the private key) which returns the pre-image in E_{pk}. An injective *probabilistic* trapdoor one-way function f from E to F is actually a function $f : \mathrm{E} \times \mathrm{R} \to \mathrm{F}$, which takes as input a pair (x, ρ) and outputs $y \in \mathrm{F}$. The element x lies in E and is the important input, ρ is the random string in R which makes the function to be probabilistic. Injectivity means that for any y there is at most one x (but maybe several ρ) such that $y = f(x, \rho)$. The function g which on input y outputs x is the inverse of the probabilistic function f. Clearly, we need the function f to be efficiently computable, but the one-wayness means that computing the unique x (if it exists) such that $y = f(x, \rho)$ is intractable (unless one knows the trapdoor g.) These are the basic requirement for an asymmetric encryption primitive. But for our construction to work, we need two additional properties:

– the function $f : \mathrm{E} \times \mathrm{R} \to \mathrm{F}$ is a bijection;

– without knowing the trapdoor, it is intractable to invert f in E, even for an adversary which has access to the decisional oracle $\mathsf{Same}_f(y, y')$ which answers whether $g(y) = g(y')$.

The latter property is exactly the "gap problem" notion, which is defined by the following success probability $\mathsf{Succ}_f^{\mathsf{gap}}(t, q)$, for any adversary $\mathcal{A}$ whose running time is limited by t, and the number of queries to the decisional oracle Same_f is upper-bounded by q:

$$\mathsf{Succ}_f^{\mathsf{gap}}(t, q) = \max_{\mathcal{A}}\{x \stackrel{R}{\leftarrow} \mathrm{E}, \rho \stackrel{R}{\leftarrow} \mathrm{R}, y = f(x, \rho) : \mathcal{A}^{\mathsf{Same}_f}(y) = x\}.$$

For a family of functions, this success probability includes the random choice of the keys in the probability space, and assumes the inputs randomly drawn from the appropriate sets, hence the notation $\mathsf{Succ}_{\varphi}^{\mathsf{gap}}(t, q)$ for a family $(\varphi_{\mathsf{pk}})_{\mathsf{pk}}$.

3.2 Examples

Let us see whether the two above additional properties are restrictive or not in practice:

– The first example is clearly the RSA permutation [25], where for a given public key $\mathsf{pk} = (n, e)$, the sets are $\mathrm{E} = \mathrm{F} = \mathbb{Z}_n^\star$ and R is the empty set. Then, this is clearly an injective (but deterministic) function, which is furthermore a bijection. Because of the determinism, the decisional oracle $\mathsf{Same}(y, y')$ simply checks whether $y = y'$: the gap problem is thus the classical RSA problem.
– The goal of our extension of OAEP is to apply it to the famous ElGamal encryption [9] in a cyclic group $\mathbb{G}$ of order q, generated by g. Given a public key $\mathsf{pk} = y \in \mathbb{G}$, the sets are $\mathrm{E} = \mathbb{G}$, $\mathrm{R} = \mathbb{Z}_q$ and $\mathrm{F} = \mathbb{G} \times \mathbb{G}$: $\varphi_y(x, \rho) = (g^\rho, x \times y^\rho)$, which is a probabilistic injection from E onto F, and a bijection from $\mathrm{E} \times \mathrm{R}$ onto F. About the decisional oracle, it should check, on inputs $(a = g^\rho, b = x \times y^\rho)$ and $(a' = g^{\rho'}, b' = x' \times y^{\rho'})$, whether $x = x'$, which is equivalent to decide whether $(g, y, a'/a = g^{\rho'-\rho}, b'/b = (x'/x) \times y^{\rho'-\rho})$ is a Diffie-Hellman quadruple: the gap problem is thus the well-known Gap Diffie-Hellman problem [18, 19].
– One can easily see that the Paillier's encryption [20] also fits this formalism.

3.3 Description of OAEP 3-Round

Notations and Common Parameters. For a simpler presentation, and an easy to read analysis, we focus on the case where $\mathrm{E} = \{0,1\}^n$ (is a binary set). A similar analysis as in [21] could be performed to deal with more general sets. On the other hand, any function can be mapped into this formalism at some low cost [2].

The encryption and decryption algorithms use three hash functions: $\mathcal{F}$, $\mathcal{G}$, $\mathcal{H}$ (assumed to behave like random oracles in the security analysis) where the security parameters satisfy $n = k + \ell$:

$$\mathcal{F} : \{0,1\}^k \to \{0,1\}^\ell \qquad \mathcal{G} : \{0,1\}^\ell \to \{0,1\}^k \qquad \mathcal{H} : \{0,1\}^k \to \{0,1\}^\ell.$$

The encryption scheme uses any probabilistic injection family $(\varphi_{\mathsf{pk}})_{\mathsf{pk}}$, whose inverses are respectively denoted ψ_{sk}, where sk is the private key associated to the public key pk. The symbol "$\|$" denotes the bit-string concatenation and identifies $\{0,1\}^k \times \{0,1\}^\ell$ to $\{0,1\}^n$.

Encryption Algorithm. The space of the plaintexts is $\mathcal{M} = \{0,1\}^\ell$, the encryption algorithm uses random coins, from two distinct sets $r \in \mathcal{R} = \{0,1\}^k$ and $\rho \in \mathrm{R}$, and outputs a ciphertext c into F: on a plaintext $m \in \mathcal{M}$, one computes

$$s = m \oplus \mathcal{F}(r) \qquad t = r \oplus \mathcal{G}(s) \qquad u = s \oplus \mathcal{H}(t) \qquad c = \varphi_{\mathsf{pk}}(t\|u, \rho).$$

Decryption Algorithm. On a ciphertext c, one first computes $t\|u = \psi_{\mathsf{sk}}(c)$, where $t \in \{0,1\}^k$ and $u \in \{0,1\}^\ell$, and then

$$s = u \oplus \mathcal{H}(t) \qquad r = t \oplus \mathcal{G}(s) \qquad m = s \oplus \mathcal{F}(r).$$

4 Security Result

In this section, we state and prove the security of this construction. A sketch is provided in the body of the paper, the full proof can be found in the appendix.

Theorem 4. *Let $\mathcal{A}$ be an* IND-RCCA *adversary against the OAEP 3-round construction with any trapdoor one-way probabilistic function family $(\varphi_{\mathsf{pk}})_{\mathsf{pk}}$, within time τ. Let us assume that after q_f, q_g, q_h and q_d queries to the random oracles $\mathcal{F}$, $\mathcal{G}$ and $\mathcal{H}$, and the decryption oracle respectively, its advantage $\mathsf{Adv}^{\mathsf{ind\text{-}rcca}}_{\mathsf{oaep\text{-}3}}(\tau)$ is greater than ε. Then, $\mathsf{Succ}^{\mathsf{gap}}_{\varphi}(\tau', q_d(q_g q_h + q_d))$ is upper-bounded by*

$$\frac{\varepsilon}{2} - q_d^2 \times \left(\frac{1}{2^\ell} + \frac{6}{2^k}\right) - (4q_d + 1) \times \left(\frac{q_g}{2^\ell} + \frac{q_f}{2^k}\right) - q_d \times \frac{q_f + 1}{2^k},$$

with $\tau' \leq \tau + (q_f + q_g + q_h + q_d)T_{lu} + q_d^2 T_{\mathsf{Same}} + (q_d + 1)q_g q_h (T_\varphi + T_{\mathsf{Same}})$, where T_φ is the time complexity for evaluating any function φ_{pk}, T_{Same} is the time for the decisional oracle $\mathsf{Same}_{\varphi_{\mathsf{pk}}}$ to give its answer, and T_{lu} is the time complexity for a look up in a list.

4.1 Trapdoor Permutations

Before proving this general result, let us consider the particular case where φ_{pk} is a permutation from E onto F (*i.e.*, a deterministic function.) The general result has indeed several drawbacks:

- the reduction cost introduces a cubic factor $q_d q_g q_h$ which implies larger keys for achieving a similar security level as for some other constructions;
- the security relies on a gap problem, which is a strong assumption in many cases;
- and we cannot achieve the usual IND-CCA security level.

These drawbacks are acceptable as the price of generality: this becomes one of the best padding for ElGamal or Paillier strongly secure variants. However, for trapdoor permutations, such as RSA, several OAEP variants achieve much better efficiency.

But one should interpret the above result in this particular case: first, the gap-problem becomes the classical one-wayness, since the decisional oracle is simply the equality test; Furthermore, the RCCA scenario becomes the classical CCA one; Finally, because of the determinism of the permutation, with proper bookkeeping, one can avoid the cubic factor, and fall back to the usual quadratic factor $q_g q_h$, as for any OAEP-like constructions (OAEP+, SAEP and SAEP+). Then, one can claim a much better security result:

Theorem 5. *Let $\mathcal{A}$ be an* IND-CCA *adversary against the OAEP 3-round construction with a trapdoor one-way permutation family $(\varphi_{\mathsf{pk}})_{\mathsf{pk}}$, within time τ. Let us assume that after q_f, q_g, q_h and q_d queries to the random oracles $\mathcal{F}$, $\mathcal{G}$ and $\mathcal{H}$, and the decryption oracle respectively, its advantage $\mathsf{Adv}^{\mathsf{ind\text{-}cca}}_{\mathsf{oaep\text{-}3}}(\tau)$ is greater than ε. Then, $\mathsf{Succ}^{\mathsf{ow}}_{\varphi}(\tau')$ is upper-bounded by*

$$\frac{\varepsilon}{2} - q_d^2 \times \left(\frac{1}{2^\ell} + \frac{6}{2^k}\right) - (4q_d + 1) \times \left(\frac{q_g}{2^\ell} + \frac{q_f}{2^k}\right) - q_d \times \frac{q_f + 1}{2^k},$$

with $\tau' \leq \tau + (q_f + q_g + q_h + q_d)T_{lu} + q_g q_h T_\varphi$, where T_φ is the time complexity for evaluating any function φ_{pk} and T_{lu} is the time complexity for a look up in a list.

4.2 Sketch of the Proof

The proof is very similar to the one in [21], but the larger class (injective probabilistic functions), and the improved security result (relative to the one-wayness) make some points more intricate: for a permutation f, each value x maps to a unique image $y = f(x)$; whereas for a function f, each value x maps to several images $y = f(x, \rho)$, according to the random string ρ. Consequently, when used as an asymmetric encryption primitive, the adversary may have the ability to build another y' whose pre-image is identical to the one of y: $x = g(y) = g(y')$. Such a query to the decryption oracle is not excluded in the CCA scenario, while we may not be able to either detect or answer. Hence the relaxed version of chosen-ciphertext security, and the decisional oracle Same_f: the latter helps to detect ciphertexts with identical pre-images, the relaxed scenario gives the ability to answer test in this case. Granted the decisional oracle Same_f, we can also detect whether a decryption query c has the same pre-image as a previous decryption query c' in which case we output the same plaintext. If it is a really new ciphertext, by using again the decisional oracle Same_f, we can check whether s and t have both been asked to $\mathcal{G}$ and $\mathcal{H}$, respectively, which immediately leads to the plaintext m. In the negative case, a random plaintext can be safely returned.

4.3 More Details

The full proof can be found in the appendix, but here are the main steps, since the proof goes by successive games in order to show that the above decryption simulation is almost indistinguishable for the adversary. Then, a successful IND-RCCA adversary can be easily used for inverting the one-way function.

$\mathbf{G}_0$: We first start from the real IND-RCCA attack game.

$\mathbf{G}_1$–$\mathbf{G}_2$: We then simulate the view of the adversary, first, as usual with lists for the random oracles and the decryption oracle (see figures 1 and 2.)
We then modify the generation of the challenge ciphertext, using a random mask $f^\star$, totally independent of the view of the adversary: the advantage of any adversary is then clearly zero. The plaintext is indeed unconditionally hidden.

The only way for any adversary to detect this simulation is to ask $\mathcal{F}(r^\star)$ and then detect that the answer differs from any possible $f^\star$. We are thus interested in this event, termed AskF, which denotes the event that $r^\star$ is asked to $\mathcal{F}$.

The main difference with the OAEP 2-round construction, as shown by Shoup with his counter-example [27], is that here an adversary cannot make another ciphertext with the same r as $r^\star$, in the challenge ciphertext, but either by chance, or if it had asked for *both* $\mathcal{G}(s^\star)$ and $\mathcal{H}(t^\star)$. We now try to show this fact.

$\mathcal{F}$, $\mathcal{G}$ and $\mathcal{H}$ Oracles

Query $\mathcal{F}(r)$: if a record (r, f) appears in $\mathcal{F}$-List, the answer is f.
Otherwise the answer f is chosen randomly in $\{0,1\}^k$ and the record (r, f) is added in $\mathcal{F}$-List.

Query $\mathcal{G}(s)$: if a record (s, g) appears in $\mathcal{G}$-List, the answer is g.
Otherwise the answer g is chosen randomly in $\{0,1\}^\ell$ and the record (s, g) is added in $\mathcal{G}$-List.

►**Rule** EvalGAdd$^{(1)}$

| Do nothing `% To be defined later`

Query $\mathcal{H}(t)$: if a record (t, h) appears in $\mathcal{H}$-List, the answer is h.
Otherwise the answer h is chosen randomly in $\{0,1\}^k$ and the record (t, h) is added in $\mathcal{H}$-List.

$\mathcal{D}$ Oracle

Query $\mathcal{D}_{\mathsf{sk}}(c)$: first, if we are in the second stage (the challenge $c^\star$ as been defined), ask for $(c, c^\star)$ to the decisional oracle $\mathsf{Same}_{\varphi_{\mathsf{pk}}}$. In case of positive decision, the answer is test.
Else, for each (m', c') in $\mathcal{D}$-List, ask for (c, c') to the decisional oracle $\mathsf{Same}_{\varphi_{\mathsf{pk}}}$. In case of a positive decision, the answer is the corresponding$'m'$.
Otherwise, the answer m is defined according to the following rules:

►**Rule** Decrypt-Init$^{(1)}$

| Compute $t \| u = \psi_{\mathsf{sk}}(c)$;

Look up for $(t, h) \in \mathcal{H}$-List:

- if the record is found, compute $s = u \oplus h$.
  Look up for $(s, g) \in \mathcal{G}$-List:
  - ∗ if the record is found, compute $r = t \oplus g$.
    Look up for $(r, f) \in \mathcal{F}$-List:
    - · if the record is found
      ►**Rule** Decrypt-TSR$^{(1)}$
      | $h = \mathcal{H}(t)$,
      | $s = u \oplus h, \quad g = \mathcal{G}(s)$,
      | $r = t \oplus g, \quad f = \mathcal{F}(r)$,
      | $m = s \oplus f$.
    - · else
      ►**Rule** Decrypt-TSnoR$^{(1)}$
      | same as rule Decrypt-TSR$^{(1)}$.
  - ∗ else
    ►**Rule** Decrypt-TnoS$^{(1)}$
    | same as rule Decrypt-TSR$^{(1)}$.
- else
  ►**Rule** Decrypt-noT$^{(1)}$
  | same as rule Decrypt-TSR$^{(1)}$.

Answer m and add (m, c) to $\mathcal{D}$-List.

Fig. 1. Formal Simulation of the IND-RCCA Game: Oracles

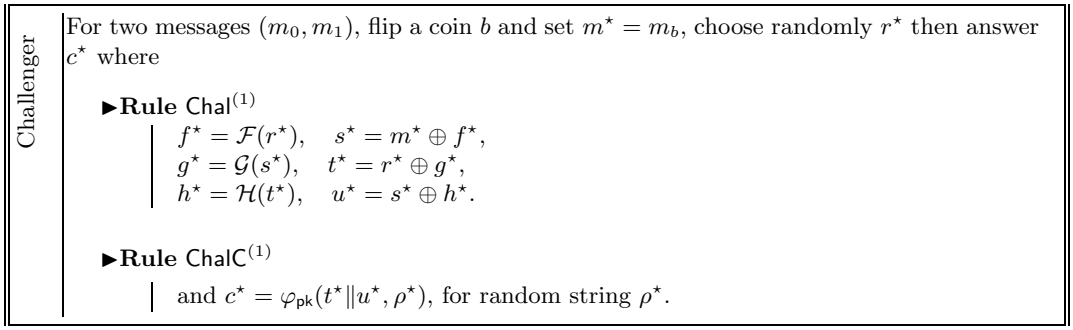

Fig. 2. Formal Simulation of the IND-RCCA Game: Challenger

$\mathbf{G}_3$–$\mathbf{G}_8$: We thus modify the decryption process so that it makes no new query to $\mathcal{G}$ and $\mathcal{H}$. The sequence of games leads to the following new rules:

▶**Rule** Decrypt-noT$^{(8)}$

> Choose $m \xleftarrow{R} \{0,1\}^\ell$.

▶**Rule** Decrypt-TnoS$^{(8)}$

> Choose $m \xleftarrow{R} \{0,1\}^\ell$.

▶**Rule** Decrypt-TSnoR$^{(8)}$

> If $s = s^\star$ but $s^\star$ has not been directly asked by the adversary yet: $m \xleftarrow{R} \{0,1\}^\ell$.
> Else, one chooses $m \xleftarrow{R} \{0,1\}^\ell$, computes $f = m \oplus s$ and adds (r, f) in $\mathcal{F}$-List.

▶**Rule** EvalGAdd$^{(8)}$

> For each $(t, h) \in \mathcal{H}$-List and each $(m, c) \in \mathcal{D}$-List, choose an arbitrary random $\rho \in \mathrm{R}$ and ask for $(c, c' = \varphi_{\mathsf{pk}}(t \| h \oplus s, \rho))$ to the decisional oracle $\mathsf{Same}_{\varphi_{\mathsf{pk}}}$. If the record is found (the decisional oracle $\mathsf{Same}_{\varphi_{\mathsf{pk}}}$ answers "yes"), we compute $r = t \oplus g$ and $f = m \oplus s$, and finally add (r, f) in $\mathcal{F}$-List.

Some bad cases may appear, which make our simulation to fail. But they are very unlikely, we thus can safely cancel executions, applying the following rule

▶**Rule** Abort$^{(8)}$

> Abort and output a random bit:
>
> - If $s^\star$ has been asked to $\mathcal{G}$ by the adversary, while the latter did not ask for $\mathcal{H}(t^\star)$.
> - If a Decrypt-TSR/Decrypt-TSnoR rule has been applied with $t = t^\star$, while $\mathcal{H}(t^\star)$ had not been asked by the adversary yet.
> - If a Decrypt-TSR rule has been applied with $s = s^\star$, while $\mathcal{G}(s^\star)$ had not been asked by the adversary yet.

The remaining bad case (termed AskGHA) is if both $s^\star$ and $t^\star$ have been asked to $\mathcal{G}$ and $\mathcal{H}$ by the adversary. Such a case helps the adversary to distinguish our simulation. On the other hand, this case helps to invert φ_{pk}.

$\mathbf{G}_9$: With these new rules for decryption, the simulation of the decryption oracle does not use at all the queries previously asked to $\mathcal{G}$ and $\mathcal{H}$ by the generation of the challenge, but just the queries directly asked by the adversary, which are available

to the simulator (we remind that we are in the random-oracle model.) One can thus make $g^\star$ and $h^\star$ to be values independent to the view of the adversary:

►**Rule** $\mathsf{Chal}^{(9)}$

> The two values $r^+ \stackrel{R}{\leftarrow} \{0,1\}^k$ and $f^+ \stackrel{R}{\leftarrow} \{0,1\}^\ell$ are given, as well as $g^+ \stackrel{R}{\leftarrow} \{0,1\}^k$ and $h^+ \stackrel{R}{\leftarrow} \{0,1\}^\ell$ then $r^\star = r^+$, $f^\star = f^+$, $s^\star = m^\star \oplus f^+$, $g^\star = g^+$, $t^\star = r^+ \oplus g^\star$, $h^\star = h^+$ and $u^\star = s^\star \oplus h^\star$.

And then the decryption oracle can be simply replaced by the classical plaintext-extractor which looks up in the lists $\mathcal{G}$-List and $\mathcal{H}$-List (which only contain the queries directly asked by the adversary) to obtain the values (s, g) and (t, h) which match with $c = \varphi_{\mathsf{pk}}(t\|s \oplus h, \rho)$, using the decisional oracle $\mathsf{Same}_{\varphi_{\mathsf{pk}}}$, but without using anymore ψ_{sk}. In case of failure, one answers a random plaintext m.

We simply conclude, since our reduction does not use any oracle, but can answer any query of the adversary, in an indistinguishable way, unless the bad case AskGHA happens: in which case we have inverted φ_{sk}.

The time complexity of one simulation is thus upper-bounded by $q_g q_h \times (T_\varphi + T_{\mathsf{Same}})$, where T_φ is the time to evaluate one function in the φ family, and T_{Same} the time for the decisional oracle, plus the initial look up in the $\mathcal{D}$-List: $T_{lu} + q_d T_{\mathsf{Same}}$. Thus the global running time is bounded by (including all the list look up):

$$\tau' \leq \tau + q_d q_g q_h \times (T_\varphi + T_{\mathsf{Same}}) + q_d^2 \times T_{\mathsf{Same}} + (q_f + q_g + q_h + q_d) \times T_{lu}.$$

In the particular case where φ_{pk} is a permutation from E onto F (a deterministic one), one can improve it, using an extra list of size $q_g q_h$, which stores all the tuples $(s, g = \mathcal{G}(s), t, h = \mathcal{H}(t), c' = \varphi_{\mathsf{pk}}(t\|s \oplus h))$. The time complexity then falls down to $\tau + q_g q_h \times T_\varphi + (q_f + q_g + q_h + q_d) \times T_{lu}$.

5 Conclusion

All the OAEP variants [27, 6] applied to RSA, with general exponents (*i.e.*, not Rabin nor $e = 3$) admit, in the best cases, reductions to the RSA problem with a quadratic loss in time complexity [23] – the original OAEP is even worst because of the reduction to the partial-domain case, which requires a more time consuming reduction to the full-domain RSA problem. Furthermore, for a security level in 2^{-k}, a randomness of $2k$ bits is required, plus a redundancy of k bits.

In this paper, we show that the variant of OAEP with 3 rounds admits a reduction as efficient as the best OAEP variants (to the full-domain RSA, when applied to the RSA family) without having to add redundancy: one can thus earn k bits. But this is not the main advantage.

Considering any criteria, OAEP with 3 rounds is at least as good as all the other OAEP variants, but from a more practical point of view

- since no redundancy is required, implementation becomes easier, namely for the decryption process [16];
- it applies to more general families than just (partial-domain) one-way trapdoor permutations, but to any probabilistic trapdoor one-way function. It is thus safer to use it with a new primitive [15].

As a conclusion, OAEP with 3 round is definitely the most generic and the simplest padding to use with almost all the encryption primitives.

References

1. J. H. An, Y. Dodis, and T. Rabin. On the Security of Joint Signatures and Encryption. In *Eurocrypt '02*, LNCS 2332, pages 83–107. Springer-Verlag, Berlin, 2002.
2. M. Bellare, A. Boldyreva, A. Desai, and D. Pointcheval. Key-Privacy in Public-Key Encryption. In *Asiacrypt '01*, LNCS 2248, pages 566–582. Springer-Verlag, Berlin, 2001.
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. Optimal Asymmetric Encryption – How to Encrypt with RSA. In *Eurocrypt '94*, LNCS 950, pages 92–111. Springer-Verlag, Berlin, 1995.
5. M. Bellare and A. Sahai. Non-Malleable Encryption: Equivalence between Two Notions, and an Indistinguishability-Based Characterization. In *Crypto '99*, LNCS 1666, pages 519–536. Springer-Verlag, Berlin, 1999.
6. D. Boneh. Simplified OAEP for the RSA and Rabin Functions. In *Crypto '01*, LNCS 2139, pages 275–291. Springer-Verlag, Berlin, 2001.
7. R. Canetti, H. Krawczyk, and J. B. Nielsen. Relaxing Chosen-Ciphertext Security. In *Crypto '03*, LNCS 2729, pages 565–582. Springer-Verlag, Berlin, 2003.
8. D. Dolev, C. Dwork, and M. Naor. Non-Malleable Cryptography. *SIAM Journal on Computing*, 30(2):391–437, 2000.
9. T. El Gamal. A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. *IEEE Transactions on Information Theory*, IT–31(4):469–472, July 1985.
10. E. Fujisaki and T. Okamoto. Secure Integration of Asymmetric and Symmetric Encryption Schemes. In *Crypto '99*, LNCS 1666, pages 537–554. Springer-Verlag, Berlin, 1999.
11. E. Fujisaki and T. Okamoto. How to Enhance the Security of Public-Key Encryption at Minimum Cost. *IEICE Transaction of Fundamentals of Electronic Communications and Computer Science*, E83-A(1):24–32, January 2000.
12. E. Fujisaki, T. Okamoto, D. Pointcheval, and J. Stern. RSA–OAEP is Secure under the RSA Assumption. In *Crypto '01*, LNCS 2139, pages 260–274. Springer-Verlag, Berlin, 2001.
13. E. Fujisaki, T. Okamoto, D. Pointcheval, and J. Stern. RSA–OAEP is Secure under the RSA Assumption. *Journal of Cryptology*, 17(2):81–104, 2004.
14. S. Goldwasser and S. Micali. Probabilistic Encryption. *Journal of Computer and System Sciences*, 28:270–299, 1984.
15. E. Jaulmes and A. Joux. A Chosen Ciphertext Attack on NTRU. In *Crypto '00*, LNCS 1880, pages 20–35. Springer-Verlag, Berlin, 2000.
16. J. Manger. A Chosen Ciphertext Attack on RSA Optimal Asymmetric Encryption Padding (OAEP) as Standardized in PKCS #1. In *Crypto '01*, 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. T. Okamoto and D. Pointcheval. REACT: Rapid Enhanced-security Asymmetric Cryptosystem Transform. In *CT – RSA '01*, LNCS 2020, pages 159–175. Springer-Verlag, Berlin, 2001.
19. T. Okamoto and D. Pointcheval. The Gap-Problems: a New Class of Problems for the Security of Cryptographic Schemes. In *PKC '01*, LNCS 1992. Springer-Verlag, Berlin, 2001.
20. P. Paillier. Public-Key Cryptosystems Based on Discrete Logarithms Residues. In *Eurocrypt '99*, LNCS 1592, pages 223–238. Springer-Verlag, Berlin, 1999.
21. D. H. Phan and D. Pointcheval. Chosen-Ciphertext Security without Redundancy. In *Asiacrypt '03*, LNCS 2894, pages 1–18. Springer-Verlag, Berlin, 2003. Full version available from `http://www.di.ens.fr/users/pointche/`.
22. D. Pointcheval. Chosen-Ciphertext Security for any One-Way Cryptosystem. In *PKC '00*, LNCS 1751, pages 129–146. Springer-Verlag, Berlin, 2000.
23. D. Pointcheval. How to Encrypt Properly with RSA. *CryptoBytes*, 5(1):10–19, winter/spring 2002.
24. 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.
25. 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.
26. V. Shoup. A Proposal for an ISO Standard for Public-Key Encryption, december 2001. ISO/IEC JTC 1/SC27.
27. V. Shoup. OAEP Reconsidered. In *Crypto '01*, LNCS 2139, pages 239–259. Springer-Verlag, Berlin, 2001.

A Proof of the Theorem 4

GAME $\mathbf{G}_0$: This is the real IND-RCCA attack game, in which the adversary is fed with the public key pk, 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 $c^\star$ of $m^\star = m_b$. This ciphertext $c^\star$ comes from the two random strings $r^\star \xleftarrow{R} \{0,1\}^k$ and $\rho^\star \xleftarrow{R} \mathrm{R}$: $c^\star = \mathcal{E}_{\mathsf{pk}}(m_b, r^\star, \rho^\star) = \varphi_{\mathsf{pk}}(t^\star \| u^\star, \rho^\star)$. On input $c^\star$, A_2 outputs bit b' after an overall running time bounded by τ. We denote by S_0 the event $b' = b$ and use the same notation S_n in any game $\mathbf{G}_n$ below. Note that the adversary is given access to the decryption oracle $\mathcal{D}_{\mathsf{sk}}$ during both steps of the attack. The adversary can also ask the random oracles $\mathcal{F}$, $\mathcal{G}$ and $\mathcal{H}$.

$$\Pr[\mathsf{S}_0] = \frac{1}{2} \times \left(\mathsf{Adv}^{\mathsf{ind\text{-}rcca}}_{\mathsf{oaep\text{-}3}}(\tau) + 1 \right).$$

Advantage Zero. The goal of the first two games is to build a game in which b is perfectly indistinguishable to any adversary.

GAME $\mathbf{G}_1$: The simulation in this game is presented in figures 1 and 2. We simulate the random oracles $\mathcal{F}$, $\mathcal{G}$ and $\mathcal{H}$, as well as the decryption oracle $\mathcal{D}_{\mathsf{sk}}$, by maintaining lists: $\mathcal{F}$-List, $\mathcal{G}$-List and $\mathcal{H}$-List as usual for the random oracles, and $\mathcal{D}$-List to deal with similar decryption queries. This perfect simulation does not modify any probability.

GAME $\mathbf{G}_2$: In order to make the advantage of any (even powerful) adversary exactly zero, we define the mask $f^\star$ so that it is totally independent of the view of the adversary:

▶**Rule** $\mathsf{Chal}^{(2)}$

> The two values $r^+ \xleftarrow{R} \{0,1\}^k$ and $f^+ \xleftarrow{R} \{0,1\}^\ell$ are given, then $r^\star = r^+, f^\star = f^+$ and $s^\star = m^\star \oplus f^+, g^\star = \mathcal{G}(s^\star), t^\star = r^+ \oplus g^\star, h^\star = \mathcal{H}(t^\star), u^\star = s^\star \oplus h^\star$.

The two games $\mathbf{G}_2$ and $\mathbf{G}_1$ are perfectly indistinguishable unless $r^\star$ has been asked for $\mathcal{F}$ (by the adversary or the decryption oracle). We thus define this event AskF_2, and we have:

$$|\Pr[\mathsf{S}_2] - \Pr[\mathsf{S}_1]| \leq \Pr[\mathsf{AskF}_2].$$

As hoped, in this game, f^+ is used in the generation of the challenge, but does not appear anywhere else since $\mathcal{F}(r^+)$ is not defined to be equal to f^+. Thus, the output of $\mathcal{A}_2$ follows a distribution that does not depend on b. Accordingly, $\Pr[\mathsf{S}_2] = 1/2$. We thus obtain a first conclusion:

$$\mathsf{Adv}^{\mathsf{ind\text{-}rcca}}_{\mathsf{oaep\text{-}3}}(t) \leq 2 \times \Pr[\mathsf{AskF}_2]. \tag{1}$$

The main difference with the OAEP 2-round construction, as shown by Shoup with his counter-example, is that here, an adversary cannot make another ciphertext with the same r as $r^\star$, in the challenge ciphertext, but either by chance, or if its had asked for both $\mathcal{G}(s^\star)$ and $\mathcal{H}(t^\star)$. We now try to show this fact. We are thus interested in the event AskF.

No $\mathcal{G}$ and $\mathcal{H}$ Queries in the Decryption Simulation. We modify the decryption process so that it makes no new query to $\mathcal{G}$ and $\mathcal{H}$.

GAME $\mathbf{G}_3$: We begin to simulate the decryption oracle. First, we modify the rules Decrypt-noT, Decrypt-TnoS and Decrypt-TSnoR by outputting a random message, and choosing at random the $\mathcal{F}$, $\mathcal{G}$ and $\mathcal{H}$ oracles outputs, without looking first in $\mathcal{F}$-List and $\mathcal{G}$-List:

▶**Rule** Decrypt-noT$^{(3)}$

> Choose $m \xleftarrow{R} \{0,1\}^\ell$, $h \xleftarrow{R} \{0,1\}^\ell$ and $g \xleftarrow{R} \{0,1\}^k$
> Set $s = u \oplus h$, $r = t \oplus g$ and compute $f = m \oplus s$.
> Add (r, f) in $\mathcal{F}$-List, (s, g) in $\mathcal{G}$-List, (t, h) in $\mathcal{H}$-List.

▶**Rule** Decrypt-TnoS$^{(3)}$

> Choose $m \xleftarrow{R} \{0,1\}^\ell$ and $g \xleftarrow{R} \{0,1\}^k$
> Set $r = t \oplus g$ and compute $f = m \oplus s$.
> Add (r, f) in $\mathcal{F}$-List, (s, g) in $\mathcal{G}$-List.

▶**Rule** Decrypt-TSnoR$^{(3)}$

> Choose $m \xleftarrow{R} \{0,1\}^\ell$.
> Compute $f = m \oplus s$.
> Add (r, f) in $\mathcal{F}$-List.

The above rules are almost similar as before, except that inconsistencies may appear if some elements were already in the lists. But inputs are random, and thus the collisions are unlikely. More details are given in appendix B.1, to prove the following relation:

$$|\Pr[\mathsf{AskF}_3] - \Pr[\mathsf{AskF}_2]| \leq q_d \times \left(\frac{q_g + q_d}{2^\ell} + 2 \times \frac{q_f + q_d}{2^k}\right). \tag{2}$$

GAME $\mathbf{G}_4$: In the previous rules for the simulation of the decryption simulation, the random oracles were almost perfectly simulated, adding new relations to the corresponding lists. We now make more technical modifications, which differ a lot from previous proofs. Granted that, we can achieve a stronger result. We thus modify the above rules by not storing anymore the new relations (s, g) in $\mathcal{G}$-List, defined during these simulations. Therefore, when g is no longer explicitly defined, we cannot compute r and thus we do not store (r, f) in $\mathcal{F}$-List either. However, as soon as $\mathcal{G}(s)$ is known, we must define $\mathcal{F}(r)$ accordingly, and update the lists:

▶**Rule** Decrypt-TnoS$^{(4)}$

> Choose $m \xleftarrow{R} \{0,1\}^\ell$.

▶**Rule** Decrypt-noT$^{(4)}$

> Choose $m \xleftarrow{R} \{0,1\}^\ell$ and $h \xleftarrow{R} \{0,1\}^\ell$.
> Add (t, h) in $\mathcal{H}$-List.

▶**Rule** EvalGAdd$^{(4)}$

> For each $(t, h) \in \mathcal{H}$-List and each $(m, c) \in \mathcal{D}$-List, choose an arbitrary random $\rho \in \mathrm{R}$ and ask for $(c, c' = \varphi_{\mathsf{pk}}(t \| h \oplus s, \rho))$ to the decisional oracle $\mathsf{Same}_{\varphi_{\mathsf{pk}}}$. If the record is found (the decisional oracle $\mathsf{Same}_{\varphi_{\mathsf{pk}}}$ answers "yes"), we compute $r = t \oplus g$ and $f = m \oplus s$, and finally add (r, f) in $\mathcal{F}$-List.

With the above new rules, the answers in the two games $\mathbf{G}_4$ and $\mathbf{G}_3$ are perfectly indistinguishable unless r is asked to $\mathcal{F}$ before s is asked to $\mathcal{G}$, which event is denoted by AskRbS_4. In fact, if r is asked after s, at the moment that s is asked, by the above simulation of $\mathcal{G}$ (and the extra rule $\mathsf{EvalGAdd}$), we will find out (t, h) and therefore (r, f) is computed in a consistent way, exactly as it would have been in the game $\mathbf{G}_3$, and added in $\mathcal{F}$-List.

Note that for each ciphertext c, the value t is unique, and thus h, and consequently s: this rule is thus applied at most once for each ciphertext asked to the decryption oracle.

However, until s is asked, g is a uniformly distributed random variable, and r is so too. Therefore, the probability that r has been asked to $\mathcal{F}$ is $q_f/2^k$:

$$\Pr[\mathsf{AskRbS}_4] \leq q_d \times \frac{q_f + q_d}{2^k}.$$

A more important gap may appear because of the removal of some elements (s, g) from $\mathcal{G}$-List, and (r, f) from $\mathcal{F}$-List, which may have some impact on the simulation of later decryption queries, but also on the event AskF itself:

- if we remove (r, f) for $r = r^\star$, then the event AskF happened in the previous game but does not occur in the new game. Fortunately, since $r = t \oplus g$ where g is randomly chosen, the probability of this event is $1/2^k$.
- in the simulation of a later decryption query $c' = \varphi_{\mathsf{pk}}(t' \| u', \rho')$, the element $s' = s$ might have been found in the previous game, while it is no longer in the list in the current game. A rule $\mathsf{Decrypt\text{-}TSR}/\mathsf{Decrypt\text{-}TSnoR}$ is thus replaced by the rule $\mathsf{Decrypt\text{-}TnoS}$, which means that g was just defined during the first decryption, in the previous game, but never revealed later. Therefore, the probability that $r' = t' \oplus g' = t' \oplus g$ is in the $\mathcal{F}$-List is $(q_f + q_d)/2^k$ (modification of $\mathsf{Decrypt\text{-}TSR}$ into $\mathsf{Decrypt\text{-}TnoS}$). In the case that r' was not in the $\mathcal{F}$-List, m' was and is still random: modification of $\mathsf{Decrypt\text{-}TSnoR}$ into $\mathsf{Decrypt\text{-}TnoS}$.

$$|\Pr[\mathsf{AskF}_4] - \Pr[\mathsf{AskF}_3]| \leq \Pr[\mathsf{AskRbS}_4] + q_d \times \frac{q_f + q_d}{2^k} + q_d \times \frac{1}{2^k} \leq 2q_d \times \frac{q_f + q_d}{2^k} + q_d \times \frac{1}{2^k}.$$

GAME $\mathbf{G}_5$: We follow in simplifying the simulation of the decryption, by not storing the new relations (t, h) in $\mathcal{H}$-List either:

▶**Rule** $\mathsf{Decrypt\text{-}noT}^{(5)}$

Choose $m \xleftarrow{R} \{0,1\}^\ell$.

In the two games, the answers of the decryption simulations are identical, since they are random values. Nevertheless, the $\mathcal{H}$-List has been changed, which may impact several other things:

- an $\mathcal{F}$-answer can be changed. Indeed, if s is asked to $\mathcal{G}$ before t is asked to $\mathcal{H}$, which event is denoted by AskSbT_5, the rule $\mathsf{EvalGAdd}$ will not apply. Otherwise, when the event AskSbT_5 does not happen, the $\mathcal{F}$-List and the $\mathcal{F}$ simulation are unchanged, after the $\mathsf{EvalGAdd}$ rule. Fortunately, until t is asked to $\mathcal{H}$, h is a uniformly distributed random variable, and $s = u \oplus h$ is so too. Therefore, the probability that s has been asked to $\mathcal{G}$ is $q_g/2^\ell$ (since no new $\mathcal{G}$ relation is added by the decryption simulation):

$$\Pr[\mathsf{AskSbT}_5] \leq q_d \times \frac{q_g}{2^\ell}.$$

– the removal of (t, h) from $\mathcal{H}$-List, may have some impact on the simulation of a later decryption query $c' = \varphi_{\mathsf{pk}}(t'\|u', \rho')$:
  - if s' is in the $\mathcal{G}$-List and $t' = t$, it was found in the previous game, but it is no longer in the list. A rule Decrypt-TSR/Decrypt-TSnoR is thus replaced by the rule Decrypt-noT. This event means that $h' = h$ was just defined during the first decryption in the previous game, but never revealed later. The probability for $s' = t' \oplus h$ to be in the $\mathcal{G}$-List was less than $q_g/2^\ell$, which is an upper-bound of this case to appear.
  - if s' is not in the $\mathcal{G}$-List but $t' = t$ was found in the previous game, it may not be in the list any longer. A rule Decrypt-TnoS is thus replaced by the rule Decrypt-noT. In this case, the decryption is the same (it gives always a random plaintext and adds no element in the lists).

Summing up for all decryption queries, we get:

$$|\Pr[\mathsf{AskF}_5] - \Pr[\mathsf{AskF}_4]| \leq \Pr[\mathsf{AskSbT}_5] + q_d \times \frac{q_g}{2^\ell} \leq 2q_d \times \frac{q_g}{2^\ell}.$$

Remark that the $\mathcal{G}$-List and $\mathcal{H}$-List contain now only the queries asked by the adversaries and by the generation of the challenge. The simulation of the decryption queries does not make/simulate any new query to $\mathcal{G}$ or $\mathcal{H}$, but to $\mathcal{F}$ only.

We denote by AskGA_5 and AskHA_5 the events that $s^\star$ and $t^\star$ (the values involved in the challenge), respectively, are asked by the adversary. The event AskGHA_5 is also set to true when both AskGA_5 and AskHA_5 happen: $\mathsf{AskGHA} = \mathsf{AskGA} \wedge \mathsf{AskHA}$. Note that these two queries, $s^\star$ and $t^\star$, are also asked for the generation of the challenge $c^\star$, but we do not consider them for the events AskGA_5 and AskHA_5.

The Classical Plaintext Extractor. We now complete the modifications of the decryption process so that it behaves exactly as the classical plaintext extractor, briefly described in the sketch of the proof.

GAME $\mathbf{G}_6$: Before going on in some other modifications, we exclude some executions, with then a random output: the rule Abort is always checked. If it is true, we stop the game with a random output b'.

►**Rule** $\mathsf{Abort}^{(6)}$
Abort and output a random bit: If $\mathsf{AskGA}_6 \wedge \neg\mathsf{AskHA}_6$, at the end.

When $\neg\mathsf{AskHA}_6$, $\mathcal{H}(t^\star) = u^\star \oplus m_b \oplus f^+$ is never revealed, while f^+ is a random value independent to the adversary's view. Therefore, $\mathcal{H}(t^\star)$ is a uniformly distributed random variable: $s^\star = u^\star \oplus H(t^\star)$ is so too. Consequently, the probability that $s^\star$ is queried is $q_g/2^\ell$. Consequently, the probability that this rule is applied is $q_g/2^\ell$:

$$|\Pr[\mathsf{AskF}_6] - \Pr[\mathsf{AskF}_5]| \leq \frac{q_g}{2^\ell}.$$

Furthermore, $\Pr[\mathsf{AskF}_6]$ can easily be upper-bounded with the following relation, which proof is detailed in the appendix B.2

$$\Pr[\mathsf{AskF}_6] \leq \frac{q_f}{2^k} + \Pr[\mathsf{AskGHA}_6]. \tag{3}$$

We can thus make another intermediate conclusion, which explains our interest in $\Pr[\mathsf{AskGHA}_6]$:

$$\Pr[\mathsf{AskF}_2] \leq q_d^2 \times \left(\frac{4}{2^k} + \frac{1}{2^\ell}\right) + (3q_d + 1) \times \left(\frac{q_f}{2^k} + \frac{q_g}{2^\ell}\right) + q_d \times \frac{q_f + 1}{2^k} + \Pr[\mathsf{AskGHA}_6]. \tag{4}$$

GAME $\mathbf{G}_7$: We furthermore abort some games during the execution, if one of the following situations is met:

►**Rule** $\mathsf{Abort}^{(7)}$

Abort and output a random bit:

- If $\mathsf{AskGA}_7 \wedge \neg\mathsf{AskHA}_7$, at the end.
- If a Decrypt-TSR/Decrypt-TSnoR rule has been applied with $t = t^\star$, while $\mathcal{H}(t^\star)$ had not been asked by the adversary yet.
- If a Decrypt-TSR rule has been applied with $s = s^\star$, while $\mathcal{G}(s^\star)$ had not been asked by the adversary yet.

The gap between the two games is detailed in the appendix B.3 with the proof of the following relation:

$$|\Pr[\mathsf{AskGHA}_7] - \Pr[\mathsf{AskGHA}_6]| \leq q_d \times \left(\frac{q_f + q_d}{2^k} + \frac{q_g}{2^\ell}\right). \tag{5}$$

GAME $\mathbf{G}_8$: In this game, we complete the simulation of the decryption oracle, so that it does not depend on the queries that the generation of the challenge makes. The decryption oracle does not use anymore the element $(s^\star, g^\star)$ if the adversary did not ask for $s^\star$.

►**Rule** $\mathsf{Decrypt\text{-}TSnoR}^{(8)}$

If $s = s^\star$ but $s^\star$ has not been directly asked by the adversary yet: $m \xleftarrow{R} \{0,1\}^\ell$.
Else, one chooses $m \xleftarrow{R} \{0,1\}^\ell$, computes $f = m \oplus s$ and adds (r, f) in $\mathcal{F}$-List.

If $s = s^\star$ but $s^\star$ has not been asked to $\mathcal{G}$ by the adversary during a Decrypt-TSnoR rule, $f = \mathcal{F}(r)$ is a uniformly distributed random variable, therefore, we can give a random answer m. However, in this case, we do not store anymore (r, f) and this could make some problems: if a latter decryption query c' involves $r' = r$. In this case $g^\star \oplus t = g' \oplus t'$. Therefore, if $s' = s$ then $t' = t$ and the answer m' should be equal to m, which is checked out at the beginning of the simulation by using the decisional oracle $\mathsf{Same}_{\varphi_{\mathsf{pk}}}$ on (c, c'). In the case $s' \neq s$, $g = \mathcal{G}(s) = g^\star$ is independent to g'. Moreover, $s^\star$ is not queried and thus $r = g^\star \oplus t$ is a uniformly distributed random variable: the probability that a later decryption query c' satisfies that is $1/2^k$:

$$|\Pr[\mathsf{AskGHA}_8] - \Pr[\mathsf{AskGHA}_7]| \leq \frac{q_d^2}{2^k}.$$

GAME $\mathbf{G}_9$: In the above game, one can remark that the simulation of the decryption does not use at all the queries asked to $\mathcal{G}$ and $\mathcal{H}$ by the generation of the challenge:

- Decrypt-TSR
  - with $r = r^\star$: is not possible if the query $r^\star$ has not been queried directly by the adversary, since it has not been queried during the generation of the challenge;
  - with $s = s^\star$: excluded in the game $\mathbf{G}_7$;
  - with $t = t^\star$: excluded in the game $\mathbf{G}_7$;
- Decrypt-TSnoR

  - with $s = s^\star$: similar to Decrypt-TnoS since the game $\mathbf{G}_8$;
  - with $t = t^\star$: excluded in the game $\mathbf{G}_7$;
- Decrypt-TnoS
  - with $t = t^\star$: similar to Decrypt-noT since the game $\mathbf{G}_5$;

We can thus modify the simulation of the challenge, without querying $\mathcal{G}$ or $\mathcal{H}$:

▶**Rule** Chal$^{(9)}$

> The two values $r^+ \xleftarrow{R} \{0,1\}^k$ and $f^+ \xleftarrow{R} \{0,1\}^\ell$ are given, as well as $g^+ \xleftarrow{R} \{0,1\}^k$ and $h^+ \xleftarrow{R} \{0,1\}^\ell$ then $r^\star = r^+$, $f^\star = f^+$, $s^\star = m^\star \oplus f^+$, $g^\star = g^+$, $t^\star = r^+ \oplus g^\star$, $h^\star = h^+$ and $u^\star = s^\star \oplus h^\star$.

As seen above, this does not impact at all the simulation of the decryption, and the rule EvalGAdd either since the modification had already been considered in the game $\mathbf{G}_5$. The probability distributions are thus unchanged.

In this game, the simulation of the decryption of c is the simple plaintext extractor [4, 12, 21] which looks up in the lists $\mathcal{G}$-List and $\mathcal{H}$-List (which only contain the queries directly asked by the adversary) to obtain the values (s, g) and (t, h) which match with $c = \varphi_{\mathsf{pk}}(t\|s \oplus h, \rho)$, using the decisional oracle $\mathsf{Same}_{\varphi_{\mathsf{pk}}}$, but without using anymore ψ_{sk}:

▶**Rule** Decrypt-Init$^{(9)}$

> Look up for $(t, h) \in \mathcal{H}$-List and $(s, g) \in \mathcal{G}$-List, we choose arbitrarily $\rho \in \mathrm{R}$ and compute $c' = \varphi_{\mathsf{pk}}(t\|s \oplus h, \rho)$ and ask for (c, c') to the decisional oracle $\mathsf{Same}_{\varphi_{\mathsf{pk}}}$.
>
> - if the record is found (a positive answer from the decisional oracle $\mathsf{Same}_{\varphi_{\mathsf{pk}}}$), we found out the corresponding s and t, and we furthermore define $u = s \oplus h$.
> - otherwise, we take $t = \perp$ and $u = \perp$.

Note that the definitions $t = \perp$ and $u = \perp$ are just done to make the answer m to be random in the following of the simulation. The time complexity of one simulation is thus upper-bounded by $q_g q_h \times (T_\varphi + T_{\mathsf{Same}})$, where T_φ is the time to evaluate one function in the φ family, and T_{Same} the time for the decisional oracle, plus the initial look up in the $\mathcal{D}$-List: $T_{lu} + q_d T_{\mathsf{Same}}$. Thus the global running time is bounded by (including all the list look up):

$$\tau' \leq \tau + q_d q_g q_h \times (T_\varphi + T_{\mathsf{Same}}) + q_d^2 \times T_{\mathsf{Same}} + (q_f + q_g + q_h + q_d) \times T_{lu}.$$

In the particular case where φ_{pk} is a permutation from E onto F (a deterministic one), one can improve it, using an extra list of size $q_g q_h$, which stores all the tuples $(s, g = \mathcal{G}(s), t, h = \mathcal{H}(t), c' = \varphi_{\mathsf{pk}}(t\|s \oplus h))$. The time complexity then falls down to $\tau + q_g q_h \times T_\varphi + (q_f + q_g + q_h + q_d) \times T_{lu}$.

Conclusion. The proof is almost finished, granted the permutation property of φ_{pk} from $\mathrm{E} \times \mathrm{R}$ onto F. Indeed using a classical argument, one easily gets the relation, formally proven in appendix B.4:

$$\Pr[\mathsf{AskGHA}_9] \leq \mathsf{Succ}^{\mathsf{gap}}_{\varphi}(\tau' + q_g q_h (T_\varphi + T_{\mathsf{Same}}), q_d q_g q_h + q_d^2), \tag{6}$$

where τ' is the above running time of the simulation, which concludes the proof of the Theorem, since

$$\Pr[\mathsf{AskGHA}_6] \leq \frac{2q_d^2}{2^k} + q_d \times \left(\frac{q_f}{2^k} + \frac{q_g}{2^\ell}\right) + \mathsf{Succ}^{\mathsf{gap}}_{\varphi}(\tau' + q_g q_h (T_\varphi + T_{\mathsf{Same}}), q_d q_g q_h + q_d^2). \tag{7}$$

B Relations for the Proof of the Theorem 4

B.1 Proof of the Relation (2)

To explain the relation (2), we exhibit a sub-sequence of games, starting from the game $\mathbf{G}_2$.

GAME $\mathbf{G}_{2.1}$: First, we do not query anymore the oracles $\mathcal{G}$ and $\mathcal{H}$. Let us remind that we focus here on queries $c = \varphi_{\mathsf{pk}}(t\|u, \rho)$ where $\mathcal{H}(t)$ has never been queried.

►**Rule** Decrypt-noT$^{(2.1)}$

> Choose $h \xleftarrow{R} \{0,1\}^\ell$ and set $s = u \oplus h$.
> Choose $g \xleftarrow{R} \{0,1\}^k$ and set $r = t \oplus g$.
> Compute $f = \mathcal{F}(r)$ and set $m = s \oplus f$.
> Add (s, g) in $\mathcal{G}$-List, (t, h) in $\mathcal{H}$-List.

The two games $\mathbf{G}_{2.1}$ and $\mathbf{G}_2$ are perfectly indistinguishable unless s is already in $\mathcal{G}$-List. Since h is randomly chosen, $s = u \oplus h$ is uniformly distributed. So, the probability that s has already been queried to $\mathcal{G}$ is $(q_g + q_d)/2^\ell$:

$$|\Pr[\mathsf{AskF}_{2.1}] - \Pr[\mathsf{AskF}_2]| \leq q_d(q_g + q_d)/2^\ell.$$

GAME $\mathbf{G}_{2.2}$: In this game, we do not query the oracle $\mathcal{F}$ either:

►**Rule** Decrypt-noT$^{(2.2)}$

> Choose $h \xleftarrow{R} \{0,1\}^\ell$ and set $s = u \oplus h$.
> Choose $g \xleftarrow{R} \{0,1\}^k$ and set $r = t \oplus g$.
> Choose $f \xleftarrow{R} \{0,1\}^\ell$ and set $m = s \oplus f$.
> Add (r, f) in $\mathcal{F}$-List, (s, g) in $\mathcal{G}$-List, (t, h) in $\mathcal{H}$-List.

The two games $\mathbf{G}_{2.2}$ and $\mathbf{G}_{2.1}$ are perfectly indistinguishable unless r is already in $\mathcal{F}$-List. Since g is randomly chosen, $r = t \oplus g$ is uniformly distributed. So, the probability that r has already been queried to $\mathcal{F}$ is less than $(q_f + q_d)/2^k$:

$$|\Pr[\mathsf{AskF}_{2.2}] - \Pr[\mathsf{AskF}_{2.1}]| \leq q_d(q_f + q_d)/2^k.$$

GAME $\mathbf{G}_{2.3}$: We now want to clearly show that m is uniformly distributed, thus we make a formal change, which does not modify anything:

►**Rule** Decrypt-noT$^{(2.3)}$

> Choose $m \xleftarrow{R} \{0,1\}^\ell$.
> Choose $h \xleftarrow{R} \{0,1\}^\ell$ and set $s = u \oplus h$.
> Choose $g \xleftarrow{R} \{0,1\}^k$ and set $r = t \oplus g$.
> Compute $f = m \oplus s$.
> Add (r, f) in $\mathcal{F}$-List, (s, g) in $\mathcal{G}$-List, (t, h) in $\mathcal{H}$-List.

The two games $\mathbf{G}_{2.3}$ and $\mathbf{G}_{2.2}$ are perfectly indistinguishable, since we just define f from a random m instead of defining m from a random f:

$$\Pr[\mathsf{AskF}_{2.3}] = \Pr[\mathsf{AskF}_{2.2}].$$

GAME $\mathbf{G}_{2.4}$: We now modify the rule Decrypt-TnoS by not calling anymore the oracles $\mathcal{F}$ and $\mathcal{G}$. Note that here, we apply the rule on a ciphertext $c = \varphi_{\mathsf{pk}}(t\|u, \rho)$ such that $h = \mathcal{H}(t)$ is known, but $s = u \oplus h$ has never been queried to $\mathcal{G}$.

►**Rule** $\mathsf{Decrypt\text{-}TnoS}^{(2.4)}$

> Choose $g \xleftarrow{R} \{0,1\}^k$ and set $r = t \oplus g$.
> Choose $f \xleftarrow{R} \{0,1\}^\ell$ and set $m = s \oplus f$.
> Add (r, f) in $\mathcal{F}$-List, (s, g) in $\mathcal{G}$-List.

The two games $\mathbf{G}_{2.4}$ and $\mathbf{G}_{2.3}$ are perfectly indistinguishable unless r is already in $\mathcal{F}$-List. Since g is randomly chosen, $r = t \oplus g$ as uniformly distributed. So, the probability that r is queried to $\mathcal{F}$ is less than $(q_f + q_d)/2^k$:

$$|\Pr[\mathsf{AskF}_{2.4}] - \Pr[\mathsf{AskF}_{2.3}]| \leq q_d(q_f + q_d)/2^k.$$

GAME $\mathbf{G}_{2.5}$: As above, we now make clear that m is uniformly distributed, thus we make a formal change, which does not modify anything:

►**Rule** $\mathsf{Decrypt\text{-}TnoS}^{(2.5)}$

> Choose $m \xleftarrow{R} \{0,1\}^\ell$.
> Choose $g \xleftarrow{R} \{0,1\}^k$ and set $r = t \oplus g$.
> Compute $f = m \oplus s$.
> Add (r, f) in $\mathcal{F}$-List, (s, g) in $\mathcal{G}$-List.

The two games $\mathbf{G}_{2.5}$ and $\mathbf{G}_{2.4}$ are perfectly indistinguishable:

$$\Pr[\mathsf{AskF}_{2.5}] = \Pr[\mathsf{AskF}_{2.4}].$$

GAME $\mathbf{G}_{2.6}$: We eventually do the same for the rule Decrypt-TSnoR, but note that s and t were already in $\mathcal{G}$-List and $\mathcal{H}$-List respectively: no more query is asked, but $\mathcal{F}(r)$, which thus gives a random value. As above, we just need to make clear that m is uniformly distributed, thus we make a formal change, which does not modify anything:

►**Rule** $\mathsf{Decrypt\text{-}TSnoR}^{(2.6)}$

> Choose $m \xleftarrow{R} \{0,1\}^\ell$.
> Compute $f = m \oplus s$.
> Add (r, f) in $\mathcal{F}$-List.

The two games $\mathbf{G}_{2.6}$ and $\mathbf{G}_{2.5}$ are perfectly indistinguishable:

$$\Pr[\mathsf{AskF}_{2.6}] = \Pr[\mathsf{AskF}_{2.5}].$$

This is exactly the game $\mathbf{G}_3$.

B.2 Proof of the Relation (3)

To explain the relation (3), we exhibit a sub-sequence of games, starting from the game $\mathbf{G}_6$.
GAME $\mathbf{G}_{6.1}$:

►**Rule** $\mathsf{Abort}^{(6.1)}$

> If $\mathsf{AskGA}_{6.1}$.

We have:

$$\Pr[\mathsf{AskF}_{6.1}] = \Pr[\mathsf{AskF}_6 \wedge \neg\mathsf{AskGA}_6].$$

GAME $\mathbf{G}_{6.2}$: In this game, we modify the simulation of the challenge, so that $s^\star$ is really random.

►**Rule** $\mathsf{Chal}^{(6.2)}$

The values $r^+ \xleftarrow{R} \{0,1\}^k$, $s^+ \xleftarrow{R} \{0,1\}^\ell$ and $g^+ \xleftarrow{R} \{0,1\}^k$ are given, then

$$r^\star = r^+ \quad s^\star = s^+ \quad g^\star = g^+ \quad f^\star = s^+ \oplus m^\star$$

$$t^\star = r^+ \oplus g^+ \quad h^\star = \mathcal{H}(t^\star) \quad u^\star = s^+ \oplus h^\star.$$

This game and the previous game are perfectly indistinguishable unless $s^\star$ is asked for $\mathcal{G}$ by the adversary. But we know that this event has already been rejected in the previous game, and the decryption simulation does not make any $\mathcal{G}$-query. We have thus:

$$\Pr[\mathsf{AskF}_{6.2}] = \Pr[\mathsf{AskF}_{6.1}].$$

In this game, $r^+ = t^\star \oplus g^+$ is uniformly distributed, and independent of the adversary's view since g^+ is never revealed:

$$\Pr[\mathsf{AskF}_{6.2}] = \frac{q_f}{2^k}.$$

We deduce that:

$$\begin{aligned}\Pr[\mathsf{AskF}_6] &= \Pr[\mathsf{AskF}_6 \wedge \neg\mathsf{AskGA}_6] + \Pr[\mathsf{AskF}_6 \wedge \mathsf{AskGA}_6 \wedge \neg\mathsf{AskHA}_6] \\ &\quad + \Pr[\mathsf{AskF}_6 \wedge \mathsf{AskGHA}_6] \\ &\leq \Pr[\mathsf{AskF}_{6.1}] + \Pr[\mathsf{AskGA}_6 \wedge \neg\mathsf{AskHA}_6] + \Pr[\mathsf{AskGHA}_6] \\ &\leq \Pr[\mathsf{AskF}_{6.2}] + 0 + \Pr[\mathsf{AskGHA}_6] \leq \frac{q_f}{2^k} + \Pr[\mathsf{AskGHA}_6].\end{aligned}$$

B.3 Proof of the Relation (5)

To explain the relation (5), we exhibit a sub-sequence of games. Note that we forget the sub-sequence of games presented in the previous appendix B.2, but we indeed start again from the game $\mathbf{G}_6$.

GAME $\mathbf{G}_{6.1}$: We modify the game so that the simulation of the decryption oracle does not depend on the queries made by the generation of the challenge. First, it does not use anymore the element $(t^\star, h^\star)$ if the adversary did not asked for $t^\star$ to $\mathcal{H}$:

►**Rule** $\mathsf{Abort}^{(6.1)}$

- If $\mathsf{AskGA}_{6.1} \wedge \neg\mathsf{AskHA}_{6.1}$, at the end.
- If a Decrypt-TSR/Decrypt-TSnoR rule has been applied with $t = t^\star$, while $\mathcal{H}(t^\star)$ had not been asked by the adversary yet.

Until $\mathcal{H}(t^\star)$ is asked by the adversary, $\mathcal{H}(t) = \mathcal{H}(t^\star) = u^\star \oplus m_b \oplus f^+$ has never been revealed, while f^+ is a random value, independent of the adversary's view. Therefore, $\mathcal{H}(t)$ is also a random value to $\mathcal{A}$: $s = u \oplus H(t^\star)$ is so too. Moreover, because of the permutation property (on $\mathrm{E} \times \mathrm{R} \to \mathrm{F}$) of the function φ_{pk}, knowing that $t = t^\star$ and that we've already dealt with queries which have a similar pre-image as the challenge $c^\star$ in the preliminary step of the decryption, s must be different to $s^\star$. Consequently, the probability that s is in $\mathcal{G}$-List (directly asked by the adversary) is less than $q_g/2^\ell$:

$$|\Pr[\mathsf{AskGHA}_{6.1}] - \Pr[\mathsf{AskGHA}_{6.6}]| \leq \frac{q_g}{2^\ell} \times q_d.$$

GAME $\mathbf{G}_{6.2}$: In this game, we continue to simulate the decryption oracle so that it does not depend on the queries that the generation of the challenge makes. We show

that the adversary could not make a query in which r is in the list and s is implicitly equal to $s^\star$ while it did not make the query on $s^\star$.

►**Rule** $\mathsf{Abort}^{(6.2)}$

- If $\mathsf{AskGA}_{6.2} \wedge \neg\mathsf{AskHA}_{6.2}$, at the end.
- If a Decrypt-TSR/Decrypt-TSnoR rule has been applied with $t = t^\star$, while $\mathcal{H}(t^\star)$ had not been asked by the adversary yet.
- If a Decrypt-TSR rule has been applied with $s = s^\star$, while $\mathcal{G}(s^\star)$ had not been asked by the adversary yet.

As $s = s^\star$ is not asked by the adversary, the value $\mathcal{G}(s^\star) = r^+ \oplus t^\star$ is a truly random value (because of r^+). Therefore, the probability that $r = t \oplus \mathcal{G}(s^\star)$ has been asked by the adversary is $(q_f + q_d)/2^k$:

$$|\Pr[\mathsf{AskGHA}_{6.2}] - \Pr[\mathsf{AskGHA}_{6.1}]| \leq \frac{q_f + q_d}{2^k} \times q_d.$$

This is exactly the game $\mathbf{G}_7$.

B.4 Proof of the Relation (6)

To explain the relation (6), one just needs one more game:
GAME $\mathbf{G}_{9.1}$: We now easily complete the proof, by using the challenge instance y in F, on which one wants to get the pre-image x, as the challenge ciphertext $c^\star$. This challenge y is uniformly distributed in the space F.

►**Rule** $\mathsf{ChalC}^{(9.1)}$

Given the challenge $y \in \mathrm{F}$, one defines $c^\star = y$.

The distribution of $c^\star$ is clearly unchanged, because of the permutation property of φ_{pk}. The values $t^\star$ and $u^\star$ were not used anymore in the simulation, thus everything remains perfectly indistinguishable:

$$\Pr[\mathsf{AskGHA}_{9.1}] = \Pr[\mathsf{AskGHA}_9].$$

Finally, when $\mathsf{AskGHA}_{9.1}$ happens, a look up in $\mathcal{G}$-List and $\mathcal{H}$-List leads to (s, g) and (t, h) such that the decisional oracle $\mathsf{Same}_{\varphi_{\mathsf{pk}}}$ agrees with the pair $(c^\star, \varphi_{\mathsf{pk}}(t\|s \oplus h, \rho))$ (for an arbitrary random ρ in R). We can thus invert the function φ_{pk} on $c^\star = y$:

$$\Pr[\mathsf{AskGHA}_{9.1}] \leq \mathsf{Succ}^{\mathsf{gap}}_{\varphi}(\tau' + q_g q_h (T_\varphi + T_{\mathsf{Same}}), q_d q_g q_h + q_d^2),$$

where τ' is the above running time of the simulation.