Optimal Asymmetric Encryption and Signature Paddings

[Published in John Ioannidis, Angelos Keromytis, Moti Yung, Eds., *Applied Cryptography and Network Security 2005 – ACNS 2005*, vol. 3531 of *Lecture Notes in Computer Science*, pp. 254–268, Springer-Verlag, 2005.]

Benoît Chevallier-Mames[1,2], Duong Hieu Phan[2], and David Pointcheval[2]

[1] Gemplus, France – `benoit.chevallier-mames@gemplus.com`
[2] ENS, Paris, France – {`david.pointcheval,duong.hieu.phan`}`@ens.fr`

Abstract. Strong security notions often introduce strong constraints on the construction of cryptographic schemes: semantic security implies probabilistic encryption, while the resistance to existential forgeries requires redundancy in signature schemes. Some paddings have thus been designed in order to provide these minimal requirements to each of them, in order to achieve secure primitives.
A few years ago, Coron et al. suggested the design of a common construction, a *universal padding*, which one could apply for both encryption and signature. As a consequence, such a padding has to introduce both randomness and redundancy, which does not lead to an optimal encryption nor an optimal signature.
In this paper, we refine this notion of universal padding, in which a part can be either a random string in order to introduce randomness or a zero-constant string in order to introduce some redundancy. This helps us to build, with a unique padding, optimal encryption and optimal signature: first, in the random-permutation model, and then in the random-oracle model. In both cases, we study the concrete sizes of the parameters, for a specific security level: The former achieves an optimal bandwidth.

1 Introduction

When one deals with public-key encryption, chosen-ciphertext security [22] is by now the basic required security notion. Similarly, for signatures, resistance to existential forgeries against adaptive chosen-message attacks [10] is also the minimal requirement. But strong security is not enough, it has to be achieved in an efficient way, according to various criteria: time, bandwidth, but also size of the code.

The first two above criteria are the most usual goals, and improvements are continuously proposed. When dealing with public-key cryptography, one can indeed note that fast paddings have been proposed for encryption [3, 19] and signature [4]. About the bandwidth, Phan and Pointcheval recently addressed this problem for encryption [20, 21], and proposed an optimal padding, w.r.t. this criteria, by avoiding redundancy. Most signatures with message-recovery [18, 16, 4] improve the bandwidth, but these solutions are not optimal, since redundancy and randomization are always added. The notable exception is the recent idea of Katz and Wang, that achieves tight security by using FDH, but also PSS-R, constructions [4] with only one additional bit, that is not random but dependent on the message [13].

The last criteria has been more recently considered, by Coron, Joye, Naccache and Paillier [6], with the so-called notion of *universal paddings*: the code size is reduced by using a common padding for both encryption and signature. For such a goal, they used a variant of PSS, called PSS-ES. Other solutions have thereafter been proposed, including those of Komano and Ohta [14]. But in all these constructions, the resulting encryption contains redundancy, and the signature is probabilistic.

1.1 Contribution

In this paper, we address this problem of efficiency, trying to optimize the three above criteria at the same time: for a time-efficient construction, we consider simple paddings; for a good bandwidth, we extend the work of [20, 21], by avoiding not only redundancy in encryption, but also randomization in signatures; additionally, we use the idea of the Katz-Wang construction [13] in order to achieve tight security in signature. Finally, about the size of the code, we optimize the common parts in the two paddings (for signature and encryption), by giving a relaxed version of *universal padding*. Furthermore, we analyze the security of these paddings, to be used for both encryption and signature, but in the extreme case where the same primitive (trapdoor one-way permutation which might optionally be assumed claw-free) is used for encryption and signature, at the same time, as already suggested in [12]: the same public/private key pair is used for encryption and signature.

More precisely, we study two paddings with the above *universal* property. The first one is based on the Full-Domain Permutation construction, studied in [11] for signature and in [20], for encryption, which can be proved optimal with the three above criteria in the random-permutation model. Hence the name of *Optimal Permutation-based Padding* (OPbP). Then, we also review the OAEP 3-rounds construction [20, 21] (OAEP3r), in the random-oracle model [2].

1.2 Redundancy and Randomness

A basic requirement for encryption, to achieve semantic security, is a probabilistic mechanism which is necessary to make distributions of ciphertexts indistinguishable. But until recently, chosen-ciphertext security was thought to furthermore imply redundancy in the ciphertext (for a kind of proof of knowledge/awareness of the plaintext [3, 1, 7].) However, this was not mandatory [20, 21], at least in the random-oracle model and in the ideal-cipher model. Existence of such schemes in the standard model is still an open problem.

Similarly, for signature, to prevent forgeries, some redundancy in the message-signature pair (or unique string in case of message-recovery feature) is required, which should be hard to satisfy without the signing key. But most of the signature schemes are probabilistic [23, 17, 4, 8], while it is not necessary (e.g. the FDH-signature, but with *loose* security). Recently, Katz and Wang proved that it was possible to achieve *tight* security with a deterministic construction very close to FDH-signature or PSS-R, by adding a single bit that is not random but dependent on the message [13]. More precisely, this additional bit should be not predictable by anyone else than the signer, and so Katz and Wang proposed that it results from a PRF computation.

1.3 Universal Paddings

The goal of universal padding is to design a padding which can not only be applied for signature and for encryption independently, but for both at the same time, with the same user's keys: the public key is used for both encryption and verification, while the private key is used for both decryption and signature.

In the security model, the adversaries (against either semantic security or existential unforgeability) are given access to both the signing and decryption oracles, which is not the security scenario considered when one deals with encryption and signature, independently. The decryption oracle may indeed help to forge signatures, and vice-versa.

2 Security Model

2.1 Signature Schemes

Digital signature schemes are the electronic version of handwritten signatures for digital documents: a user's signature on a message m is a string which depends on m, on public and secret data specific to the user and —possibly— on randomly chosen data, in such a way that anyone can check the validity of the signature by using public data only. In this section, we briefly review the main security notions [10].

Definitions. A signature scheme $\mathsf{S} = (\mathcal{K}, \mathcal{S}, \mathcal{V})$ is defined by the three following algorithms:

– The *key generation algorithm* $\mathcal{K}$. On input 1^k, which is a formal notation for a machine with running time polynomial in k (1^k is indeed k in basis 1), the algorithm $\mathcal{K}$ produces a pair $(\mathsf{pk}, \mathsf{sk})$ of matching public and private keys. Algorithm $\mathcal{K}$ is probabilistic. The input k is called the security parameter. The sizes of the keys, or of any problem involved in the cryptographic scheme, will depend on it, in order to achieve an appropriate security level (the expected minimal time complexity of any attack).

– The *signing algorithm* $\mathcal{S}$. Given a message m and a pair of matching public and private keys $(\mathsf{pk}, \mathsf{sk})$ $\mathcal{S}$ produces a signature σ. The signing algorithm might be probabilistic.
– The *verification algorithm* $\mathcal{V}$. Given a signature σ, a message m, or just a part (possibly empty), and a public key pk, $\mathcal{V}$ possibly extracts the full message m and tests whether σ is a valid signature of m with respect to pk. In general, the verification algorithm need not be probabilistic.

Forgeries and Attacks. The simpler goal for an adversary is to build a new acceptable message-signature pair. This is called *existential forgery*. The corresponding security level is called *existential unforgeability* (EUF). On the other hand, the strongest scenario one usually considers is the so-called *adaptive chosen-message attack* (CMA), where the attacker can ask the signer to sign any message of its choice, in an adaptive way: it can adapt its queries according to previous answers. When signature generation is not deterministic, there may be several signatures corresponding to a given message. And then the notion of existential forgery may be ambiguous [24]: the original definition [10] says the adversary wins if it manages to forge a signature for a new message. Non-malleability [24] says the adversary wins if it manages to forge a new signature.

Thereafter, the security notion one wants to achieve is (at least) the resistance to existential forgeries under adaptive chosen-message attacks ($\mathsf{EUF/CMA}$): one wants that the success probability of any adversary $\mathcal{A}$ with a reasonable time is small, where

$$\mathsf{Succ}_{\mathsf{S}}^{\mathsf{euf/cma}}(\mathcal{A}) = \Pr\left[\,(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathcal{K}(1^k), (m, \sigma) \leftarrow \mathcal{A}^{\mathcal{S}_{\mathsf{sk}}}(\mathsf{pk}) : \mathcal{V}(\mathsf{pk}, m, \sigma) = 1\,\right].$$

2.2 Public-Key Encryption

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

Definitions. A public-key encryption scheme $\mathsf{S} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ is defined by the three following algorithms:

– The *key generation algorithm* $\mathcal{K}$. On input 1^k where k is the security parameter, the algorithm $\mathcal{K}$ produces a pair $(\mathsf{pk}, \mathsf{sk})$ of matching public and private keys. Algorithm $\mathcal{K}$ is probabilistic.
– The *encryption algorithm* $\mathcal{E}$. Given a message m and a public key pk, $\mathcal{E}$ produces a ciphertext c of m. This algorithm may be probabilistic. In the latter case, we write $\mathcal{E}_{\mathsf{pk}}(m; r)$ where r is the random input to $\mathcal{E}$.
– The *decryption algorithm* $\mathcal{D}$. Given a ciphertext c and the private key sk, $\mathcal{D}_{\mathsf{sk}}(c)$ gives back the plaintext m. This algorithm is necessarily deterministic.

Security Notions. The most widely admitted goal of an adversary is the distinction of ciphertexts (IND). One thus wants to make it unable to distinguish between two messages, chosen by the adversary, which one has been encrypted, with a probability significantly better than one half. On the other hand, an attacker can play many kinds of attacks. The strongest scenario consists in giving a full access to the decryption oracle, which on any ciphertext answers the corresponding plaintext. There is of course the natural restriction not to ask the challenge ciphertext to that oracle. This scenario which allows adaptively chosen ciphertexts as queries to the decryption oracle is named the *chosen-ciphertext attack* (CCA). Therefore, for any adversary $\mathcal{A}$, seen as a 2-stage attacker $(\mathcal{A}_1, \mathcal{A}_2)$, its advantage $\mathsf{Adv}_{\mathsf{S}}^{\mathsf{ind/cca}}(\mathcal{A})$ should be negligible, where

$$\mathsf{Adv}_{\mathsf{S}}^{\mathsf{ind/cca}}(\mathcal{A}) = 2 \times \Pr_{b,r}\left[\begin{array}{l}(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathcal{K}(1^k), (m_0, m_1, s) \leftarrow \mathcal{A}_1^{\mathcal{D}_{\mathsf{sk}}}(\mathsf{pk}),\\ c = \mathcal{E}_{\mathsf{pk}}(m_b; r) : \mathcal{A}_2^{\mathcal{D}_{\mathsf{sk}}}(m_0, m_1, s, c) = b\end{array}\right] - 1,$$

2.3 Signature and Encryption

As already noticed, our motivation is to design a unified padding which one could use for both encryption and signature at the same time, and furthermore with the same asymmetric primitive. The goals of an adversary are thus the same as above: build an existential forgery (EUF) against the signature scheme, or distinguish ciphertexts (IND) against the encryption scheme. However, the means are the combination of the above attacks: it has access to both the signing oracle and the decryption oracle in a fully adaptive way, hence the $\mathsf{CMA} + \mathsf{CCA}$ notation.

2.4 Claw-Free Permutations

In [13], Katz and Wang has shown that, by using trapdoor permutations induced by claw-free permutations, one can obtain a variant of FDH (just adding one more bit) with tight reduction. We can also use this technique for our construction. The existence of claw-free permutations seems be reasonable. In fact, any random self-reducible permutation can be seen as a trapdoor permutations induced by claw-free permutations [9] and almost all known examples of trapdoor permutations are self-reducible.

Definition 1 (Claw-Free Permutations). *A family of claw-free permutations is a tuple of algorithms* $\{\mathsf{Gen}; f_i; g_i | i \in I\}$ *for an index set* I *such that:*

- Gen *outputs a random index* i *and a trapdoor* td.
- f_i, g_i *are both permutations over the same domain* D_i.
- *there is an efficient sampling algorithm which, on index* i, *outputs a random* $x \in D_i$.
- f_i^{-1} *(the inverse of* f_i*) and* g_i^{-1} *(the inverse of* g_i*) are both efficiently computable given the trapdoor* td.

A claw is a pair (x_0, x_1) such that $f(x_0) = g(x_1)$. Probabilistic algorithm $\mathcal{A}$ is said to (t, ϵ)-break a family of claw-free permutations if $\mathcal{A}$ runs in time at most t and outputs a claw with probability greater than ϵ:

$$\Pr\left[(i, \mathsf{td}) \leftarrow \mathsf{Gen}(1^k), (x_0, x_1) \leftarrow \mathcal{A}(i) : f_i(x_0) = g_i(x_1)\right] \geq \epsilon$$

A family of claw-free permutations is (t, ϵ)-secure if no algorithm can (t, ϵ)-break it.

3 Optimal Permutation-based Padding

3.1 Our Optimal Proposal

In the following, we propose a universal padding, based on the construction from [20], in the random-permutation model. It is optimal both for signing and encrypting, *i.e.,* that uses only 82 bits of randomness for encrypting and only 82 bits of redundancy for signing. After the description, we show it is indeed secure, in the random-permutation model. In the next section, we provide another construction, based on the OAEP-3 rounds construction from the same paper [20], which is secure in the random-oracle model, but just near optimal (161 bits of overhead instead of 82).

The encryption and signature schemes use a permutation $\mathcal{P}$, that we assume to behave like a truly random permutation. Let k be a security parameter. Let $\varphi_{\mathsf{pk}} : \{0,1\}^n \rightarrow \{0,1\}^n$ be a trapdoor one-way permutation (whose inverse is called ψ_{sk}). Messages to sign or to encrypt with our padding function will be of size $\ell = n - k - 1$. The symbol "$\|$" denotes the bit-string concatenation and identifies $\{0,1\}^k \times \{0,1\}^\ell \times \{0,1\}$ to $\{0,1\}^n$. Finally, in the following, $\mathsf{PRF}_\varrho()$ designs a PRF that uses a secret key ϱ.

The Padding. The padding is quite simple, since it takes as input a single bit γ, the message m and an additional data r, and $\mathsf{OPbP}(\gamma, m, r) = \mathcal{P}(\gamma \| m \| r) = t \| u$. Thereafter, the reverse operation is natural: $\mathsf{OPbP}^{-1}(t, u) = \mathcal{P}^{-1}(t \| u) = \gamma \| m \| r$.

Encryption Algorithm. The space of the plaintexts is $\mathcal{M} = \{0,1\}^\ell$, the encryption algorithm uses a random coin from the set $r \in \mathcal{R} = \{0,1\}^k$, a random bit γ, and outputs a ciphertext c into $\{0,1\}^n$: on a plaintext $m \in \mathcal{M}$, one computes $t \| u = \mathsf{OPbP}(\gamma, m, r)$ and $c = \varphi_{\mathsf{pk}}(t \| u)$.

Decryption Algorithm. On a ciphertext c, one first computes $t \| u = \psi_{\mathsf{sk}}(c)$, where $t \in \{0,1\}^k$ and $u \in \{0,1\}^{\ell+1}$, and then $\gamma \| m \| r = \mathsf{OPbP}^{-1}(t, u)$. The answer is m.

Signature Algorithm. The space of the messages is $\mathcal{M} = \{0,1\}^\ell$, the signature algorithm outputs a signature σ into $\{0,1\}^n$: on a message $m \in \mathcal{M}$, one computes $\gamma = \mathsf{PRF}_\varrho(m)$, and then $t \| u = \mathsf{OPbP}(\gamma, m, 0^k)$ and $\sigma = \psi_{\mathsf{sk}}(t \| u)$.

Verification Algorithm. On a signature σ, one first computes $t\|u = \varphi_{\mathsf{pk}}(\sigma)$, where $t \in \{0,1\}^k$ and $u \in \{0,1\}^{\ell+1}$, and then $\gamma\|m\|r = \mathsf{OPbP}^{-1}(t,u)$. If $r = 0^k$, the verification outputs "Correct" and recovers m, otherwise outputs "Incorrect".

3.2 Security Analysis

A variant of this padding has already been proved to lead to an IND/CCA secure encryption scheme [20], and to a EUF/CMA signature scheme [11], in the random-permutation model. However, there was not the additional bit of Katz and Wang, that just makes more randomness in the encryption. Here, we extend these results to IND/CMA + CCA and EUF/CMA + CCA:

Theorem 2. *Let $\mathcal{A}$ and $\mathcal{B}$ be both chosen-ciphertext (to the decryption oracle) and chosen-message (to the signing oracle) adversaries, against the encryption scheme (IND) and the signature scheme (EUF) respectively. Let us assume that $\mathcal{A}$ can break the semantic security with an advantage ε_E, or $\mathcal{B}$ can produce an existential forgery with success probability ε_S (within a time bound t, after q_p, q_s, q_d queries to the permutation oracles, signing oracle and decryption oracle respectively.) Then the permutation φ_{pk} can be inverted with probability ε' within time t' where either:*

$$\varepsilon' \geq \varepsilon_E - \frac{(q_p+q_d+q_s+1)^2}{2^{k+\ell+1}} - \frac{(q_d+1)^2}{2^\ell} - \frac{2q_p+q_d+q_s+2}{2^k}, \text{ or}$$
$$\varepsilon' \geq \frac{1}{q_p+q_s+1} \cdot \left(\varepsilon_S - \frac{(q_p+q_d+q_s+1)^2}{2^{k+\ell+1}} - \frac{(q_d+1)^2}{2^\ell} - \frac{2q_p+q_d+q_s+2}{2^k}\right).$$

Particularly, if the function φ_{pk} is induced by a (t',ε')-secure claw-free permutation, the latter can be rewritten by:

$$\varepsilon' \geq \frac{1}{2}\left(\varepsilon_S - \frac{(q_p+q_d+q_s+1)^2}{2^{k+\ell+1}} - \frac{(q_d+1)^2}{2^\ell} - \frac{2q_p+q_d+q_s+2}{2^k}\right)$$

where $t' \leq t + (q_p+q_d+q_s+1)T_f$, and T_f is the time for an evaluation of φ_{pk}.

Proof. We provide now the proof of this theorem, with incremental games, to reduce the inversion of the permutation φ_{pk} on a random instance y (*i.e.,* find x such that $y = \varphi_{\mathsf{pk}}(x)$) to an attack against either the encryption or the signature. We show that either $\mathcal{A}$ or $\mathcal{B}$ can help us to invert φ_{pk}.

Some parts of this proof are similar to [20]. We anyway provide the proof without the similar parts. The full proof can be found in the full version [5].

Game $\mathbf{G}_0$: This is the attack game, in the random-permutation model. Several oracles are thus available to the adversary: two random permutation oracles ($\mathcal{P}$ and $\mathcal{P}^{-1}$), the signing oracle $\mathcal{S}_{\mathsf{sk}}$, and the decryption oracle $\mathcal{D}_{\mathsf{sk}}$.

To break the encryption, the adversary $\mathcal{A} = (A_1, A_2)$ runs its attack in two steps. First, A_1 is given the public key pk, and outputs a pair of messages

(m_0, m_1). Next a challenge ciphertext is produced by the challenger, which flips a coin b and computes a ciphertext $c^\star$ of $m^\star = m_b$. This ciphertext comes from a random $r^\star \overset{R}{\leftarrow} \{0,1\}^k$, a bit $\gamma^\star$ and $c^\star = \mathcal{E}(\gamma^\star, m_b, r^\star) = \varphi_{\mathsf{pk}}(\mathcal{P}(\gamma^\star, m_b, r^\star))$. In the second step, on input $c^\star$, A_2 outputs a bit b'. We denote by Dist_0 the event $b' = b$ and use the same notation Dist_n in any game $\mathbf{G}_n$.

To break the signature, the adversary $\mathcal{B}$ outputs its forgery, one checks whether it is actually valid or not. We denote by Forge_0 the event this forged signature is valid and use the same notation Forge_n in any game $\mathbf{G}_n$.

Note that the adversary is given access to the signing oracle $\mathcal{S}_{\mathsf{sk}}$ and the decryption oracle $\mathcal{D}_{\mathsf{sk}}$ at any time during the attack. Note also that if the adversary asks q_d queries to the decryption oracle, q_s queries to the signing oracle and q_p queries to the permutation oracles, at most $q_d + q_s + q_p + 1$ queries are asked to the permutation oracles during this game, since each decryption query or signing query may make such a new query, and the last verification step or the challenger step does too. By definition,

$$\varepsilon_E = \mathsf{Adv}^{\mathsf{ind/cma+cca}}_{\mathsf{OPbP}}(\mathcal{A}) = \Pr[\mathsf{Dist}_0] - 1/2$$
$$\varepsilon_S = \mathsf{Succ}^{\mathsf{euf/cma+cca}}_{\mathsf{OPbP}}(\mathcal{B}) = \Pr[\mathsf{Forge}_0].$$

<u>Game $\mathbf{G}_1$:</u> We skip the easy steps, similar to [20] for the encryption part, and to [4] for the signature. Details can be found in the full version [5], which leads to the simulation presented in Figures 1 and 2, which is statistically indistinguishable from the initial one since the distance is bounded by:

$$\Delta_G \leq \frac{(q_p + q_d + q_s + 1)^2}{2^{k+\ell+1}} + \frac{(q_d + 1)^2}{2^\ell} + \frac{2q_p + q_d + q_s + 2}{2^k}.$$

Challenger	For two messages (m_0, m_1), flip coins $\gamma^\star$ and b, set $m^\star = m_b$, and randomly choose $r^\star$. ►**Rule** $\mathsf{Chal}^{(1)}$ | $p^\star = \mathcal{P}(\gamma^\star, m^\star, r^\star)$; $c^\star = \varphi_{\mathsf{pk}}(p^\star)$. ►**Rule** $\mathsf{ChalAdd}^{(1)}$ | Add $(\gamma^\star, m^\star, r^\star, \perp, \perp, c^\star)$ in $\mathsf{P\text{-}List}$. Answer $c^\star$
$\mathcal{V}$-Oracle	The game ends with the verification of the output (σ) from the adversary. One first computes $t\|u = \varphi_{\mathsf{pk}}(\sigma)$, then asks for $(\gamma, m, r) = \mathcal{P}^{-1}(t\|u)$. Then he checks whether $r = 0^k$, in which case the signature is a valid signature of m.

Fig. 1. Simulation in the Game $\mathbf{G}_1$

$\mathcal{P}$-Oracle

A query $\mathcal{P}(\gamma, m, r)$ is answered by p, where

▶**Rule** $\mathsf{EvalP}^{(1)}$

```
– Look for $(\gamma, m, r, \alpha, \beta, c)$ in $\mathsf{P\text{-}List}$:
    • if the record is found,
        ∗ if $\alpha \neq \perp$, $p = \alpha$;
        ∗ otherwise, $\mathsf{Stop}$.
    • otherwise, choose a random element $s \in \{0,1\}^n$ and computes $p = \varphi_{\mathsf{pk}}(s)$. The record $(\gamma, m, r, p, s, \varphi_{\mathsf{pk}}(p))$ is added to $\mathsf{P\text{-}List}$.
```

Furthermore, if (γ, m, r) is a direct query from the adversary to $\mathcal{P}$, store the record $(\gamma, m, r, p, \perp, \varphi_{\mathsf{pk}}(p))$ in $\mathsf{P\text{-}List}$.

$\mathcal{P}^{-1}$-Oracle

A query $\mathcal{P}^{-1}(p)$ is answered by (γ, m, r), where

▶**Rule** $\mathsf{InvP}^{(1)}$

```
Compute $c = \varphi_{\mathsf{pk}}(p)$ and look for $(\gamma, m, r, \alpha, \beta, c)$ in $\mathsf{P\text{-}List}$:
  – if the record is found, $(\gamma, m, r)$ is defined,
  – otherwise we randomly choose $(\gamma, m, r)$ in $\{0,1\}^n$. If $r = 0^k$, $\mathsf{Stop}$.
```

Furthermore, if p a direct query from the adversary to $\mathcal{P}^{-1}$, store the record $(\gamma, m, r, p, \perp, \varphi_{\mathsf{pk}}(p))$ in $\mathsf{P\text{-}List}$.

$\mathcal{D}$-Oracle

A query $\mathcal{D}_{\mathsf{sk}}(c)$ is answered by m, where

▶**Rule** $\mathcal{D}^{(1)}$

```
Look for $(\gamma, m, r, \alpha, \beta, c)$ in $\mathsf{P\text{-}List}$:
  1. if the record is found, $(\gamma, m, r)$ is defined,
  2. otherwise we randomly choose $(\gamma, m, r)$ in $\{0,1\}^n$.
```

Store $(\gamma, m, r, \perp, \perp, c)$ in $\mathsf{P\text{-}List}$.

$\mathcal{S}$-Oracle

For a sign-query $\mathcal{S}_{\mathsf{sk}}(m)$, one first computes $\gamma = \mathsf{PRF}_{\varrho}(m)$, then asks for $p = \mathcal{P}(\gamma, m, 0^k)$ to the EvalP-oracle. The signature σ is then defined according to the following rule:

▶**Rule** $\mathcal{S}^{(1)}$

```
Look for $(\gamma, m, 0^k, p, s, c)$ in $\mathsf{P\text{-}List}$, and set $\sigma = s$.
```

Fig. 2. Simulation in the Game $\mathbf{G}_1$

In the following, depending on the goal of the adversary, namely against encryption or against signature, we complete the reduction to the inversion of the function ψ_{sk} on the given instance y.

Encryption Attack.

Game $\mathbf{G}_{1.1}$: We suppress the element $(\gamma^\star, m^\star, r^\star, \perp, \perp, c^\star)$ from P-List during the generation of the challenge.

▶**Rule** ChalAdd$^{(1.1)}$

```
| Do nothing.
```

The two games $\mathbf{G}_{1.1}$ and $\mathbf{G}_1$ are perfectly indistinguishable unless $(\gamma^\star, m^\star, r^\star)$ is asked for $\mathcal{P}$ (which event is included in event $\mathsf{BadP}_{1.1}$, already excluded) or $p^\star = \psi_{\mathsf{sk}}(c^\star)$ is asked to $\mathcal{P}^{-1}$. We define the latter event $\mathsf{AskInvP}_{1.1}$. We have: $\varDelta_{1.1} \leq \Pr[\mathsf{AskInvP}_{1.1}]$. Since $(\gamma^\star, m^\star, r^\star, \perp, \perp, c^\star)$ does not appear in P-List, the adversary receives answers which are perfectly independent of the latter, and therefore, it has no advantage for guessing b: $\Pr[\mathsf{Dist}_{1.1}] = \frac{1}{2}$.

Game $\mathbf{G}_{1.2}$: Instead of choosing $c^\star = \varphi_{\mathsf{pk}}(p^\star)$, we choose $c^\star = y$, uniformly at random.

▶**Rule** Chal$^{(1.2)}$

```
| $c^\star = y$.
```

So, one implicitly defines $p^\star = \psi_{\mathsf{sk}}(y)$. Since the tuple $(\gamma^\star, m^\star, r^\star, \perp, \perp, c^\star)$ is not used anywhere in the simulation, the two games $\mathbf{G}_{1.2}$ and $\mathbf{G}_{1.1}$ are perfectly indistinguishable: $\varDelta_{1.2} = 0$.

Finally, it is clear that when the event $\mathsf{AskInvP}_{1.2}$ happens, one can easily compute ψ_{sk} on y: with a look up into P-List (which contains at most $q_p+q_d+q_s+1$ elements), one can extract p such that $y = \varphi_{\mathsf{pk}}(p)$. Therefore, $\Pr[\mathsf{AskInvP}_{1.2}] \leq \mathsf{Succ}_{\varphi}^{\mathsf{ow}}(t')$, where T_φ is the time for evaluating φ_{pk}, and $t' \leq t + (q_p + q_d + q_s + 1) \times T_\varphi$ is the running time of the simulation in the current game. This completes the first part of the proof.

Signature Attack (The General Case).

Game $\mathbf{G}_{1.1}$: In the following, we number calls to the permutation oracle, but only those which are of the form $(\gamma, \star, 0^k)$, which are those that are used for signature. We define a variable ν which is initialized to 0.

▶**Rule** EvalP$^{(1.1)}$

```
| Look for $(\gamma, m, r, \alpha, \beta, c)$ in P-List:
|   – if the record is found,
|       • if $\alpha \neq \perp$, $p = \alpha$;
|       • otherwise, Stop.
|   – otherwise,
|       • if $r = 0^k$, increment $\nu$
|       • choose a random element $s \in \{0,1\}^n$ and computes $p =$
|         $\varphi_{\mathsf{pk}}(s)$. The record $(\gamma, m, r, p, s, \varphi_{\mathsf{pk}}(p))$ is added to P-List.
```

Clearly, this leaves the game indistinguishable from the game $\mathbf{G}_1$: $\Delta_{1.1} = 0$.

GAME $\mathbf{G}_{1.2}$: Since the verification process is included in the attack game, the output message is necessarily asked to the permutation oracle EvalP. Let us guess the index ν_0 of this (first) query. If the guess failed, we abort the game. Therefore, only a correct guess (event $\mathsf{GoodGuess}$) may lead to a success.

$$\Pr[\mathsf{Forge}_{1.2}] \geq \Pr[\mathsf{Forge}_{1.1}]/(q_p + q_s + 1).$$

GAME $\mathbf{G}_{1.3}$: We now incorporate the challenge y to the simulation of the permutation oracle. By this, we could extract the pre-image x. Our idea is to return y as the value of the guessed ν-th query:

```
▶Rule EvalP$^{(1.3)}$
    Look for $(\gamma, m, r, \alpha, \beta, c)$ in P-List:
      – if the record is found,
          • if $\alpha \neq \perp$, $p = \alpha$;
          • otherwise, Stop.
      – otherwise,
          • if $r = 0^k$, increment $\nu$
          • if $\nu \neq \nu_0$ or if $r \neq 0^k$, choose a random element $s \in \{0,1\}^n$
            and computes $p = \varphi_{\mathsf{pk}}(s)$.
          • if $\nu = \nu_0$ and $r = 0^k$, sets $p = y$.
          • The record $(\gamma, m, r, y, s, \varphi_{\mathsf{pk}}(p))$ is added to P-List.
```

Because of the random choice for the challenge y, this rule leaves the game indistinguishable from the previous one: $\Delta_{1.3} = 0$. It follows that the forgery leads to the pre-image of y: $\Pr[\mathsf{Forge}_{1.3}] = \mathsf{Succ}^{\mathsf{ow}}_{\varphi}(t + (q_p + q_d + q_s + 1)T_{\varphi})$. This concludes the second part of the proof.

Signature Attack (With (t', ε')-Secure Claw-Free Permutations). We assume that $(\varphi_{\mathsf{pk}}, \lambda_{\mathsf{pk}})$ are from a (t', ε')-secure claw-free permutations family.

GAME $\mathbf{G}_{1.1}$: We now exploit the bit γ to the simulation of the permutation oracle, as it was proposed firstly by Katz and Wang [13]. The idea is to use φ_{pk} in the OPbP output, for one and only one value of bit γ, and otherwise use λ_{pk}. As this value of γ is not predictable by the attacker, its forgery will, with a probability $\frac{1}{2}$, produce a claw.

```
▶Rule EvalP$^{(1.1)}$
    Look for $(\gamma, m, r, \alpha, \beta, c)$ in P-List:
      – if the record is found,
          • if $\alpha \neq \perp$, $p = \alpha$;
          • otherwise, Stop.
      – otherwise,
          • if $r \neq 0^k$ or $\gamma = \mathsf{PRF}_{\varrho}(m)$, choose a random element $s \in$
            $\{0,1\}^n$ and compute $p = \varphi_{\mathsf{pk}}(s)$.
          • if $r = 0^k$ or $\gamma \neq \mathsf{PRF}_{\varrho}(m)$, choose a random element $s \in$
            $\{0,1\}^n$ and compute $p = \lambda_{\mathsf{pk}}(s)$.
          • The record $(m, r, p, s, \varphi_{\mathsf{pk}}(p))$ is added to P-List.
```

Because of the random choice of s and so $\lambda_{\mathsf{pk}}(s)$, this rule leaves the game indistinguishable from the previous one: $\Delta_{1.1} = 0$.

Using arguments as in [13], one can easily see that the forgery leads to a claw with probability $\frac{1}{2}$. In fact, let us assume that the adversary can forge a signature $(\tilde{m}, \tilde{\sigma})$, where $(\tilde{m}, 0^k)$ has been asked to the permutation oracle $\mathcal{P}$ either in a permutation query or in the verification step. Since the bit $b_{\tilde{m}} = \mathsf{PRF}_{\varrho}(\tilde{m})$ is an unknown random bit in the view of the adversary, with probability of $\frac{1}{2}$, there exists an element $(\tilde{m}, \tilde{r}, \tilde{p} = \lambda_{\mathsf{pk}}(\tilde{s}), \tilde{s}, \varphi_{\mathsf{pk}}(\tilde{p}))$ in the P-List. In that case, the simulator can output a claw $\varphi_{\mathsf{pk}}(\tilde{\sigma}) = \lambda_{\mathsf{pk}}(\tilde{s})$.

□

3.3 Proposed Sizes for the Parameters

We say that a scheme achieves a security level of 2^κ, if the ratio between the running time t of the adversary, and its success probability ε, is at least 2^κ: this is an approximation of the expected time of success. Or similarly, we want $t/\varepsilon \leq 2^{-\kappa}$, with a usual security bound set with $\kappa = 80$.

First, we can simplify the above security result. Indeed, for practical purpose, where ℓ is the bit-size of the message, and k is the bit-size of the random/redundancy, the former is expected to be much larger than the latter: the quantity $Q/2^\ell$, or even $Q^2/2^\ell$, can be ignored in front of $Q/2^k$ (since Q, the global number of queries is bounded by 2^{80}). Therefore, the above reduction cost provides that

$$\begin{aligned}
\frac{\varepsilon_E}{t} &\leq \frac{\varepsilon'}{t} + \frac{2}{2^k} && \text{and} \\
\frac{\varepsilon_S}{t} &\leq \frac{Q\varepsilon'}{t} + \frac{2}{2^k} && \text{in the general case} \\
&\leq \frac{2\varepsilon'}{t} + \frac{2}{2^k} && \text{if the function } \varphi_{\mathsf{pk}} \text{ is induced by a claw-free permutation}
\end{aligned}$$

In the latter case (the most interesting case, where one uses RSA) we can assume the message length sufficiently large (and thus the RSA modulus) so that ε'/t is lower than 2^{-82}. Due to the Lenstra-Verheul's estimation [15], for the case of RSA, we can use a 1024-bit modulus.

In the general case, we have to consider that the security parameter (and thus message length ℓ) large enough such that the ration between ε'/t is lower than 2^{-161}. But then the overhead $k = 82$ is enough too.

As a conclusion, for the general case, we can choose $k = 82$ if the security level of the function φ is about 2^{161}. For the particular case of RSA, we can use a 1024-bit modulus. We remark then that, with only 82 bits of redundancy, we obtain the same level of security than RSA-PSS [3], which, compared to our scheme, uses a lowest bandwidth. For the encryption security, we find again the result from [20]: 82 bits of randomness are enough to achieve semantic security, even under chosen-ciphertext and chosen-message attacks.

4 The OAEP-3 Rounds Construction

4.1 Description

In order to work in the more usual random-oracle model [2], we now consider the OAEP-3 rounds construction proposed in [20, 21]. As above, the security of this padding has already been studied for encryption, but without giving access to the signing oracle to the adversary. We thus extend the security model to deal with the two oracles access.

The encryption and signature schemes 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 $n = k + \ell + 1$:

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

The encryption and signature schemes use any permutation family $(\varphi_{\mathsf{pk}})_{\mathsf{pk}}$ on the space $\{0,1\}^n$, 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 $\{0,1\}^k \times \{0,1\}^\ell \times \{0,1\}$ to $\{0,1\}^n$.

Padding $\mathsf{OAEP3r}$ and Unpadding $\mathsf{OAEP3r}^{-1}$

$$\mathsf{OAEP3r}(\gamma, m, r) : s = (\gamma\|m) \oplus \mathcal{F}(r) \quad t = r \oplus \mathcal{G}(s) \quad u = s \oplus \mathcal{H}(t)$$
$$\mathsf{OAEP3r}(\gamma, m, r) = t\|u$$
$$\mathsf{OAEP3r}^{-1}(t, u) : \ s = u \oplus \mathcal{H}(t) \qquad r = t \oplus \mathcal{G}(s) \quad \gamma\|m = s \oplus \mathcal{F}(r)$$
$$\mathsf{OAEP3r}^{-1}(t, u) = \gamma\|m\|r$$

Encryption Algorithm. The space of the plaintexts is $\mathcal{M} = \{0,1\}^\ell$, the encryption algorithm uses a random coin from the set $r \in \mathcal{R} = \{0,1\}^k$, a random bit γ and outputs a ciphertext c into $\{0,1\}^n$: on a plaintext $m \in \mathcal{M}$, one computes $t\|u = \mathsf{OAEP3r}(\gamma, m, r)$ and $c = \varphi_{\mathsf{pk}}(t\|u)$.

Decryption Algorithm. On a ciphertext c, one first computes $t\|u = \psi_{\mathsf{sk}}(c)$, where $t \in \{0,1\}^k$ and $u \in \{0,1\}^{\ell+1}$, and then $\gamma\|m\|r = \mathsf{OAEP3r}^{-1}(t, u)$. The answer is m.

Signature Algorithm. The space of the plaintexts is $\mathcal{M} = \{0,1\}^\ell$, the signature algorithm outputs a signature σ into $\{0,1\}^n$: on a plaintext $m \in \mathcal{M}$, one computes $\gamma = \mathsf{PRF}_\varrho(m)$, then computes $t\|u = \mathsf{OAEP3r}(\gamma, m, 0^k)$ and $\sigma = \psi_{\mathsf{sk}}(t\|u)$.

Verification Algorithm. On a signature σ, one first computes $t\|u = \varphi_{\mathsf{pk}}(\sigma)$, where $t \in \{0,1\}^k$ and $u \in \{0,1\}^{\ell+1}$, and then $\gamma\|m\|r = \mathsf{OAEP3r}^{-1}(t, u)$. If $r = 0^k$, the verification outputs "Correct" then recovers m, otherwise outputs "Incorrect"

4.2 Security Result

We extend the security result from [21] by the following theorem:

Theorem 3. *Let $\mathcal{A}$ and $\mathcal{B}$ be both chosen-ciphertext (to the decryption oracle) and chosen-message (to the signing oracle) adversaries, against the encryption scheme (*IND*) and the signature scheme (*EUF*) respectively. Let us assume that $\mathcal{A}$ can break the semantic security with the advantage ε_E, or $\mathcal{B}$ can produce an existential forgery with success probability ε_S (within a time bound t, after q_f, q_g, q_h, q_s, q_d queries to the oracles $\mathcal{F}$, $\mathcal{G}$, $\mathcal{H}$, signing oracle and decryption oracle respectively.) Then the permutation φ_{pk} can be inverted with probability ε' within time t' where either:*

$$\varepsilon' \geq \varepsilon_E - \left(q_d^2 \times \left(\frac{1}{2^{\ell+1}} + \frac{6}{2^k}\right) + \frac{4q_dq_g + q_g}{2^{\ell+1}} + \frac{5q_dq_f + q_gq_h + q_f + q_d}{2^k}\right) \quad \textit{or}$$
$$\varepsilon' \geq \frac{1}{q_g + q_s + 1} \times \left(\varepsilon_S - \left(q_d^2 \times \left(\frac{1}{2^{\ell+1}} + \frac{6}{2^k}\right) + \frac{4q_dq_g + q_g}{2^{\ell+1}} + \frac{5q_dq_f + q_gq_h + q_f + q_d}{2^k}\right)\right)$$

Particularly, if the function φ_{pk} is induced by a (t', ε')-secure claw-free permutation, the latter can be rewritten by:

$$\varepsilon' \geq \frac{1}{2} \times \left(\varepsilon_S - \left(q_d^2 \times \left(\frac{1}{2^{\ell+1}} + \frac{6}{2^k}\right) + \frac{4q_dq_g + q_g}{2^{\ell+1}} + \frac{5q_dq_f + q_gq_h + q_f + q_d}{2^k}\right)\right)$$

with $t' \leq t + (q_f + q_g + q_h + q_d)T_{lu} + q_d^2 T_{lu} + (q_d + 1)q_gq_h(T_\varphi + T_{lu})$, 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.

Proof. The full proof can be found in the full version [5]. The simulation of the oracles as well as the simulation of the decryption are similar to the ones in [21]. The simulation of the signature (after all the oracles are well simulated) is quite the same as in the random-permutation model case. □

4.3 Proposed Sizes for the Parameters

Using similar arguments as in the previous construction, one can simplify the constraints on the security parameters:

– For encryption, one has:
$$\frac{\varepsilon_E}{t} \leq \frac{\varepsilon'}{t} + \frac{Q}{2^k}.$$
Then, $k = 161$ is enough if the security parameters are large enough (*i.e.*, as soon as $\varepsilon'/t < 2^{-81}$).

– For signature, in the general case:

$$\frac{\varepsilon_S}{t} \leq \frac{Q\varepsilon'}{t} + \frac{Q}{2^k}.$$

In the general case, $k = 161$ is also valid, as soon as $\varepsilon'/t < 2^{-161}$.

– For signature, in case the function φ_{pk} is induced by a claw-free permutation:

$$\frac{\varepsilon_S}{t} \leq \frac{2\varepsilon'}{t} + \frac{Q}{2^k}.$$

We have a similar expression as in the above encryption case (the term ε'/t is replaced by $2\varepsilon'/t$, which allows shorter security parameters. Anyway, $k = 161$ is required, as soon as $\varepsilon'/t < 2^{-82}$.

To sum up, for the interesting case of the RSA, one can choose $k = 161$, with a security parameter chosen so that the security level of the function φ is about 2^{82}, that is 1024-bit modulus.

Acknowledgement

The work described in this paper has been supported in part by the European Commission through the IST Programme under Contract IST-2002-507932 ECRYPT.

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. M. Bellare and P. Rogaway. The Exact Security of Digital Signatures – How to Sign with RSA and Rabin. In *Eurocrypt '96*, LNCS 1070, pages 399–416. Springer-Verlag, Berlin, 1996.
5. B. Chevallier-Mames and D.H Phan and D. Pointcheval Optimal Asymmetric Encryption and Signature Paddings. In *Proc. of the ACNS*, LNCS 3531. Springer-Verlag, Berlin, 2005. Full version available from `http://www.di.ens.fr/users/pointche/`.
6. J.-S. Coron, M. Joye, D. Naccache, and P. Paillier. Universal Padding Schemes For RSA. In M. Yung, editor, *Advances in Cryptology – CRYPTO '02*, volume 2442 of *Lecture Notes in Computer Science*, pages 226–241. Springer-Verlag, Berlin, 2002.
7. R. Cramer and V. Shoup. A Practical Public Key Cryptosystem Provably Secure against Adaptive Chosen Ciphertext Attack. In *Crypto '98*, LNCS 1462, pages 13–25. Springer-Verlag, Berlin, 1998.

8. R. Cramer and V. Shoup. Signature Scheme based on the Strong RSA Assumption. In *Proc. of the 6th CCS*, pages 46–51. ACM Press, New York, 1999.
9. Y. Dodis and L. Reyzin. On the power of claw-free permutation. In *Security in Communication Networks*, 2002.
10. S. Goldwasser, S. Micali, and R. Rivest. A Digital Signature Scheme Secure Against Adaptative Chosen-Message Attacks. *SIAM Journal of Computing*, 17(2):281–308, April 1988.
11. L. Granboulan. Short Signatures in the Random Oracle Model. In *Asiacrypt '02*, LNCS 2501, pages 364–378. Springer-Verlag, Berlin, 2002.
12. S. Haber and B. Pinkas. Combining Public Key Cryptosystems. In *Proc. of the 8th ACM CSS*, pages 215–224. ACM Press, New York, 2001.
13. J. Katz and N. Wang. Efficiency improvements for signature schemes with tight security reductions. In *Proc. of the 10th CCS*, pages 155–164. ACM Press, Washington, 2003.
14. Y. Komano and K. Ohta. Efficient Universal Padding Schemes for Multiplicative Trapdoor One-Way Permutation. In D. Boneh, editor, *Advances in Cryptology – CRYPTO '03*, volume 2729 of *Lecture Notes in Computer Science*, pages 366–382. Springer-Verlag, Berlin, 2003.
15. A. Lenstra and E. Verheul. Selecting Cryptographic Key Sizes. In *PKC '00*, LNCS 1751, pages 446–465. Springer-Verlag, Berlin, 2000.
16. D. Naccache and J. Stern. Signing on a Postcard. In *Financial Cryptography '00*, LNCS 1962. Springer-Verlag, Berlin, 2001.
17. NIST. Digital Signature Standard (DSS). Federal Information Processing Standards PUBlication 186, November 1994.
18. K. Nyberg and R. A. Rueppel. Message Recovery for Signature Schemes Based on the Discrete Logarithm Problem. In *Eurocrypt '94*, LNCS 950, pages 182–193. Springer-Verlag, Berlin, 1995.
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. D. H. Phan and D. Pointcheval. Chosen-Ciphertext Security without Redundancy. In *Asiacrypt '03*, LNCS 2894, pages 1–18. Springer-Verlag, Berlin, 2003.
21. D. H. Phan and D. Pointcheval. OAEP 3-Round: A Generic and Secure Asymmetric Encryption Padding. In *Asiacrypt '04*, LNCS 3329, pages 63–77 Springer-Verlag, Berlin, 2004.
22. 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.
23. C. P. Schnorr. Efficient Signature Generation by Smart Cards. *Journal of Cryptology*, 4(3):161–174, 1991.
24. J. Stern, D. Pointcheval, J. Malone-Lee, and N. Smart. Flaws in Applying Proof Methodologies to Signature Schemes. In *Crypto '02*, LNCS 2442, pages 93–110. Springer-Verlag, Berlin, 2002.