Universal Padding Schemes for RSA with Optimal Bandwidth of Message Recovery

Wenbo Mao[1⋆] and John Malone-Lee[2]

[1] Hewlett-Packard Laboratories, Filton Road, Stoke Gifford,
Bristol, BS34 8QZ, UK.
wenbo.mao@hp.com
[2] University of Bristol, Department of Computer Science,
Merchant Venturers Building, Woodland Road,
Bristol, BS8 1UB, UK.
malone@cs.bris.ac.uk

Abstract. We prove that three OAEP-inspired randomised padding schemes (i.e., OAEP, OAEP+ and SAEP), when used with the RSA function in the trapdoor direction, form provably secure signature schemes with message recovery. Two of our three reductionist proofs are tight and hence provide exact security. Because of the exact security and OAEP's optimally high bandwidth for message recovery, our results form a desirable improvement from a previous universal RSA padding scheme good for both encryption and signature.

1 Introduction

Coron et al [5, 6] show that a minor variation of an RSA randomised padding scheme named "Probabilistic Signature Scheme with message Recovery" (RSA-PSS-R) of Bellare and Rogaway [3], in addition to being a provably secure signature scheme against adaptive chosen-message attack, can also be a provably IND-CCA2 secure encryption scheme if the padding scheme is used with the RSA function in the one-way direction. This result is very useful; it means that one key can be securely used for both signature and encryption thus reducing the key management burden. Because of the usability for both encryption and signature, Coron et al name their minor variation of the PSS-R *Universal Padding Scheme (UPS)*.

However, UPS has a limitation. Let us describe it now.

1.1 Low Bandwidth for Message Recovery

The original PSS-R padding scheme of Bellare and Rogaway and the UPS variation of Coron et al are illustrated in Figure 1. In these padding algorithms, G and H are two hash functions, M is a message either being encrypted or being

⋆ This author's research is partially funded by the EU Fifth Framework Project IST-2001-324467 "CASENET".

signed, r is a random nonce for making the padding scheme a probabilistic algorithm and the concatenation of w and s is the output of the padding scheme to which the RSA function is applied.

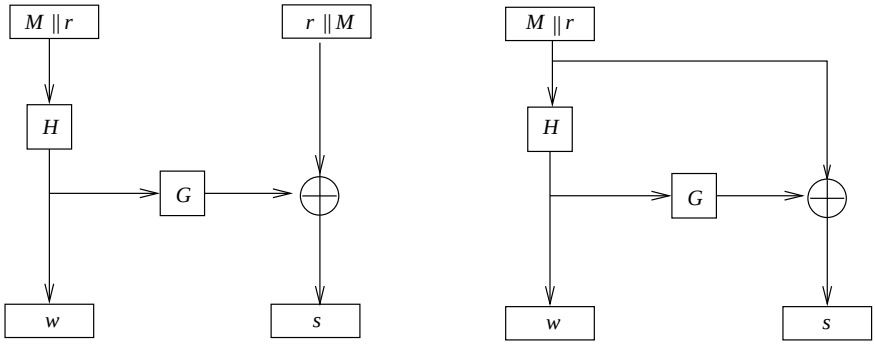

Fig. 1. The PSS-R Padding

The security proofs for the UPS encryption and signature schemes provided by Coron et al [5] are in the random oracle model (ROM) [1]. In the ROM, the hash functions G and H are are assumed to be perfect random functions or "random oracles" (ROs). The proofs are in a "reduction to contradiction" manner. In the case of UPS, a successful attacker, upon correctly submitting an educated guess (encryption case) or forgery (signature case) with a non-negligible probability Adv, must have made a certain RO query to H with another non-negligible probability Adv'. Using this piece of information (in the position of w in Figure 1), the reduction algorithm can achieve an inversion of the RSA function at a random point. If the inversion performed by the reduction is significantly faster, or has a far better advantage, than any known algorithm, we reach a contradiction.

Since a successful attack only causes a partial inversion (in the position of w in Figure 1), one run of the attacker does not lead to any evident contradiction. By a clever shift method discovered by Fujisaki et al [8], repeated running of the attacker will eventually allow the reduction algorithm to fully invert the RSA function. In order to produce a meaningful contradiction (namely solving the RSA problem far faster than is believed possible), the size of w (i.e., the input size of the RO G) must not be too small. Otherwise the reduction algorithm has to run the attacker many times. In fact, the minimum number of times for the reduction algorithm to run the attacker is 2 when $|w| > |N|/2$. Even in this minimal case, the reduction does not produce a meaningful contradiction for the currently widely adopted RSA key size of $|N| = 1024$. Consider that an

attacker with advantage $Adv = 2^{-40}$ being a valid one (i.e., consider the attacker being a dedicated one who is affordable to perform 2^{40} modulo exponentiations). The need to run the attacker at least twice makes $Adv' \approx Adv^2$. Thus, the reduction solves the RSA problem in time 2^{80} for a modulus of 1024 bits. The contradiction from this reduction can hardly be a meaningful one given the state of the factorisation art! A larger size modulus, e.g., $|N| = 2048$, is necessary to lead to a meaningful contradiction.

Now consider the case $|w| > |N|/2$ (see Figure 1), we have consequently, $|s| < |N|/2$, that is, $|\, M \parallel r \,| = |s| < |N|/2$. Further consider that the random source r must not be too small, so that sufficient entropy is obtained for the probabilistic encryption. All this leads to a very poor bandwidth for message recovery in UPS Encryption: the message M is significantly smaller than half the size of the modulus. Consider this situation when we have $|N| = 2048$, $|w| = 1025$ and $|r| = 160$, we reach $|M| = 1023 - 160 = 863$, that is, $|M|$ is only 42% of $|N|$.

This low bandwidth for message recovery is too wasteful. It has a bad consequence on a useful application of UPS which we shall describe in §1.2.

We should mention that our critique on the low bandwidth of UPS is on the encryption usage only. For the signature usage, due to a tight security proof originally obtained by Bellare and Rogaway [3], message recovery can have a desirably high bandwidth. Moreover, Coron has shown that the PSS-R signature scheme can achieve a *very* high bandwidth for message recovery [7] while security proof remaining tight.

1.2 Our Contributions

We are motivated to look for alternative padding schemes which have a desirably high bandwidth for message recovery *and* are provably secure for encryption *as well as* signatures, ideally with a tight security proof for the latter (tight security proof for the former is a non-trivial open question).

Such alternatives have a very useful application in RSA-based signcryption with provable security. In such an application, a signcrypt-text is created by "double wrappings": signing with message recovery as a "inner wrapping" using a sender's RSA trapdoor function, followed by an "outer wrapping" for encryption using a receiver's RSA one-way function. In the standard setting of one-size moduli for all users, the "double wrappings" is always possible because the receiver's verification procedure permits to fix back one possible missing bit in the case of the "inner wrapping" modulus being larger than that for the "outer wrapping". Malone-Lee and Mao achieve such a signcryption scheme with probable security by applying UPS [10].

From now on, when we say that an RSA padding scheme is good, we mean that it not only is provably secure for *both* encryption and signature uses, but also has a high bandwidth for message recovery; when we say that an RSA padding scheme is optimal, we mean that it is good, but also has a tight security proof for the signature case.

For this purpose, we have found that almost all provably secure (encryption case) RSA-OAEP like padding schemes are good, and two of them are optimal. We will provide ROM-based reductionist proofs for these schemes.

In Section 2 we show that RSA-OAEP of Bellare and Rogaway [2] is optimal. In Section 3 we show that Shoup's RSA-OAEP+ [11] is optimal. In Section 4 we show that a simplified OAEP named SAEP of Boneh [4] is good.

2 OAEP is Optimal

Optimal Asymmetric Encryption Padding (OAEP) is a randomised padding scheme proposed by Bellare and Rogaway [2] for encryption. When the padding scheme is used with the RSA function (RSA-OAEP), the encryption scheme has been proven by Fujisaki et al [8] to be IND-CCA2 secure. Figure 2 describes the padding scheme. The RSA-OAEP Signature scheme is very similar to that for encryption. We specify it in Figure 3.

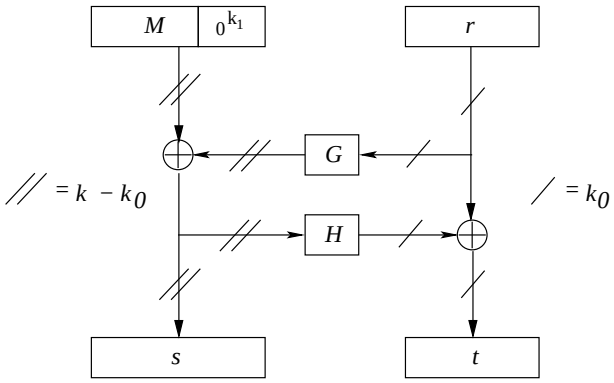

Fig. 2. The OAEP Padding

First, using Figure 2 we can argue that the OAEP padding scheme for encryption can have a bandwidth for message recovery far better than that of the PSS-R scheme used for encryption. From the picture we see $|M| = k - k_0 - k_1$ where $k = |N|$. Thus, for the required setting of $|N| = 2048$ (in order to reach a meaningful contradiction for security proof for encryption), we can use $k_0 = k_1 = 160$. In this reasonable setting, we achieve $|M| = 2048 - 320 = 1728$, that is, $|M|$ is 84% of $|N|$. That is why OAEP has been widely regarded as an efficient encryption scheme (though, as in the case of PSS-R for encryption, the reduction proof of security for encryption is not ideally tight).

Having shown the desirable bandwidth for message recovery, we now turn to investigate the issue of applying OAEP to create a secure signature scheme.

Key Parameters
Let $(N, e, d, G, H, n, k_0, k_1) \leftarrow GEN(1^k)$ satisfy the following: (N, e, d) are RSA key material; $k = |N| = n + k_0 + k_1$ with 2^{-k_0} and 2^{-k_1} being negligible quantities; G, H are hash functions satisfying

$$G : \{0,1\}^{k_0} \mapsto \{0,1\}^{k-k_0}, \quad H : \{0,1\}^{k-k0} \mapsto \{0,1\}^{k_0};$$

n is the length for the plaintext message.
Let (N, e) be Alice's RSA public and $d = e^{-1} \pmod{\phi(N)}$ be her private exponent.

Signature Generation
To sign a message $M \in \{0,1\}^n$, Alice performs the following steps:

1. $r \leftarrow_U \{0,1\}^{k_0}$; $s \leftarrow (M \parallel 0^{k_1}) \oplus G(r)$; $t \leftarrow r \oplus H(s)$;
2. if ($s \parallel t \geq N$) go to 1; ($*$ Probability for looping i times is 2^{-i} $*$)
3. $\sigma \leftarrow (s \parallel t)^d \pmod{N}$.

The signature is σ.

Signature Verification
To verify a signature σ, Bob performs the following steps:

1. $s \parallel t \leftarrow \sigma^e \pmod{N}$ satisfying $|s| = n + k_1 = k - k_0$, $|t| = k_0$;
2. $u \leftarrow t \oplus H(s)$; $v \leftarrow s \oplus G(u)$;
3. output $\begin{cases} M \parallel \text{“True”} & \text{if } v = M \parallel 0^{k_1} \\ \text{“False”} & \text{otherwise} \end{cases}$
   ($*$ when “True” is in the trailing part of the output, the prefix string is the recovered message. $*$)

Fig. 3. The RSA-OAEP Signature Scheme

2.1 Unforgeability

Adaptive chosen-message attack on digital signature schemes is defined in [9]. We consider that a successful forgery of a message-signature pair for a signature scheme under this attacking mode as *breaking* the signature scheme.

Theorem 1. *Let $GEN(1^k)$ be an RSA-OAEP Signature Scheme specified in Figure 3. If an adaptive chosen-message forger can break the scheme in time t with advantage Adv, then the RSA problem under (N, e) can be solved in time t' with advantage Adv' where*

$$t' \leq t + (q_s + q_G + q_H) \cdot k^3,$$

$$Adv' \geq Adv - 2^{-n-k_1},$$

where q_s, q_G, q_H are the numbers of signing, G and H oracle queries, respectively.

Proof Our approach follows essentially that of [3].

Under an adaptive chosen-message attack, a forger $\mathcal{F}$ is allowed to make q_s signing queries and adaptively use them in its forgery. However, in the ROM,

the hash functions are ROs owned by a simulator $\mathcal{S}$, and therefore whenever $\mathcal{F}$ wants to evaluate G and H, it must make RO queries to $\mathcal{S}$. It can make q_G and q_H RO queries to G and H respectively. Since $\mathcal{F}$ is bounded, we assume $q_s + q_G + g_H$ is bounded by a polynomial in $k = |N|$. We assume that after these queries, $\mathcal{F}$ will be able to output a valid forgery which is a new message signature pair, with a significant probability Adv.

$\mathcal{S}$ will run $\mathcal{F}$ by inputting the latter with the public key $(N, e, G, H, n, k_0, k_1)$ and by answering all the queries of the latter. $\mathcal{S}$ will pick a random point $\eta \in \mathbb{Z}_N^*$ and use $\mathcal{F}$ to find the e-th root of η, with probability Adv'.

Because $\mathcal{S}$ does not have the RSA private exponent $d = e^{-1} \pmod{\phi(N)}$, it can only simulate the signing algorithm. It will also simulate the ROs. These simulations are carried out as follows. Before simulation, $\mathcal{S}$ initialises three lists: a G-list for storing G-query results, an H-list for H query results and an R-list for storing possible RSA roots. Initially, these lists are empty.

Simulation of signing query For M queried by $\mathcal{F}$, $\mathcal{S}$ picks at random $r \in \{0,1\}^{k_0}$ and $x \in \{0,1\}^k$ with $x < N$, and responds to $\mathcal{F}$ with x. Let $y = x^e \pmod{N}$ where y is parsed as

$$y = s \parallel t$$

with $|s| = k - k_0$ and $|t| = k_0$. $\mathcal{S}$ further sets

$$G(r) = (M \parallel 0^{k_1}) \oplus s$$

and

$$H(s) = r \oplus t.$$

$\mathcal{S}$ stores $(r, G(r))$ in G-list, and $(s, H(s))$ in H-list.

Simulation of G query Let r_i be an RO query to G. If r_i has already been queried (i.e., it is in G-list) then $\mathcal{S}$ answers with $G(r_i)$ picked from G-list (stored together with r_i). Otherwise, $\mathcal{S}$ performs the following steps.

$\mathcal{S}$ picks at random $x_i \in \{0,1\}^k$ with $x_i < N$. $\mathcal{S}$ sets $y_i = \eta x_i^e \pmod{N}$ where y_i is parsed as

$$y_i = s_i \parallel t_i$$

with $|s_i| = k - k_0$ and $|t_i| = k_0$. $\mathcal{S}$ further sets at random $G(r_i) \in \{0,1\}^{k-k_0}$, and

$$H(s_i) = r_i \oplus t_i.$$

Now $\mathcal{S}$ answers $\mathcal{F}$ with $G(r_i)$. It also stores $(r_i, G(r_i), x_i)$ in G-list, $(s_i, H(s_i), x_i)$ in H-list, and x_i in R-list. The storage in these lists is indexed by integer i.

Simulation of H query Let s_j be an RO query to H. If s_j is already in H-list, then $\mathcal{S}$ answers with $H(s_i)$ picked from H-list. Otherwise, $\mathcal{S}$ performs the following steps.

$\mathcal{S}$ picks at random a string $H(s_j) \in \{0,1\}^{k_0}$ and answers $\mathcal{F}$ with $H(s_j)$. $\mathcal{S}$ also stores $(s_j, H(s_j))$ in the H list. The storage is indexed by integer j.

In the ROM, all these simulations are from perfectly correct distributions, except when an event named AskH occurs. This event is when an RO query is made to H where the query is related to a successful forgery (see *Remark 2*).

The precision of the simulations is very important since $\mathcal{F}$ can release its full capacity for forging only after having been correctly trained.

Analysis After $\mathcal{F}$ has completed the training course (i.e, after $q_s + q_G + q_H$ queries having been answered), it outputs a valid message-signature pair $(\hat{M}, z)$ with probability Adv. We denote this event by FWin.

The event AskH mentioned earlier is as follows: for

$$z^e \pmod N = \hat{s} \parallel \hat{t} \tag{1}$$

with $|\hat{s}| = k - k_0$ and $|\hat{t}| = k_0$, $\hat{s}$ is in H-list. When this event occurs, an imperfection in the simulation is exposed. However, it is too late for $\mathcal{F}$ to discover the imperfection now (reason see *Remark 2*). We have

$$\begin{aligned} Adv &= \Pr[\mathsf{FWin}] \\ &= \Pr[\mathsf{FWin} \mid \mathsf{AskH}] \Pr[\mathsf{AskH}] + \Pr[\mathsf{FWin} \mid \neg\mathsf{AskH}] \Pr[\neg\mathsf{AskH}]. \end{aligned} \tag{2}$$

In the ROM, the information inside the "wrapping" of the RSA trapdoor function in the event $\neg\mathsf{AskH}$ is uniformly random. Therefore, the second term in the right-hand side of (2) is a negligible quantity $neg < 2^{-n-k_1}$. So we have

$$\Pr[\mathsf{AskH}] = (Adv - neg)/\Pr[\mathsf{FWin} \mid \mathsf{AskH}] > Adv - 2^{-n-k_1}. \tag{3}$$

That is, in the event FWin, $\hat{s}$ defined in (1) is in $\mathcal{S}$'s H-list with probability at least $Adv - 2^{-n-k_1}$. Thus, $\mathcal{S}$ can search through H-list and find the correct index ℓ, allowing it to find x_ℓ from R-list. From the simulations, we know

$$z^e = \eta x_\ell^e \pmod N.$$

So $\eta^d = z/x_\ell \pmod N$ is discovered as desired.

Clearly, $t' \leq t + (q_s + q_G + q_H) \cdot k^3$ and $Adv' = \Pr[\mathsf{AskH}]$ which relates to Adv as in (3). □

Remark 1. Our security proof for the RSA-OAEP Signature scheme achieves a tight reduction thanks to the fact that a successful forgery z enables $\mathcal{S}$ to obtain $z^e \pmod N$ by simply applying the RSA one-way function. The security proof for the RSA-OAEP Encryption [8] is not so fortunate, because there $\mathcal{S}$ does not have the RSA trapdoor function to apply; instead, a shift technique is employed and the attacker is re-run in order to derive the full inversion from partial inversions, and the resulting reduction is not tight. □

Remark 2. Recall that we have mentioned that occurrence of the event AskH reveals an imperfection of the simulations. Observe that the successful forgery z satisfying (1) and the H-query $\hat{s}$ permit $\mathcal{F}$ to derive $\hat{r} = \hat{t} \oplus H(\hat{s})$ and so it can

query $\hat{r}$ to G to obtain a random $G(\hat{r})$. Then with an overwhelming probability $G(\hat{r}) \oplus \hat{s} \neq \hat{M} \parallel 0^{k_1}$ due to the randomness of $G(\hat{r})$. However, it is now too late for $\mathcal{F}$ to find this imperfection since it has already helped $\mathcal{S}$ to invert the RSA function. Of course, one may argue that if $\mathcal{F}$ is a conscious forger, e.g., a human being, then it should withhold the forgery without sending it $\mathcal{S}$. However, we should not consider $\mathcal{F}$ being conscious. It is a probabilistic algorithm and should forge successfully whenever it is properly trained, i.e., after having been fed with perfect simulations. We notice that this "imperfection discovered too late" by an attacker applies to all ROM security proofs for RSA-padding encryption or signature schemes. □

3 OAEP+ is Optimal

Shoup proposes the RSA-OAEP+ Encryption scheme which is a slight modification of RSA-OAEP [11] and provides an ROM based proof of the IND-CCA2 security. His proof achieves a tighter reduction than that for the RSA-OAEP Encryption scheme because Adv' is linearly related to Adv, although the running time of the reduction is still a quadratic function on the number of RO queries, and hence the reduction remains inefficient (see [11]). For a picture of the OAEP+ padding, see Figure 4.

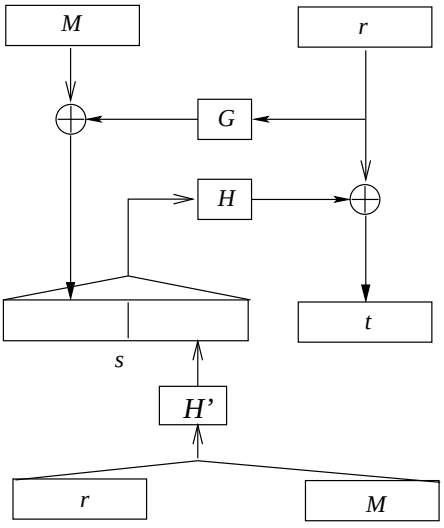

Fig. 4. The OAEP+ Padding

Key Parameters
Let $(N, e, d, G, H, n, k_0, k_1) \leftarrow GEN(1^k)$ satisfy the following: (N, e, d) are RSA key material; $k = |N| = n + k_0 + k_1$ with 2^{-k_0} and 2^{-k_1} being negligible quantities; G, H', H are hash functions satisfying

$$G : \{0,1\}^{k_0} \mapsto \{0,1\}^n, H' : \{0,1\}^{n+k_0} \mapsto \{0,1\}^{k_1}, H : \{0,1\}^{n+k_1} \mapsto \{0,1\}^{k_0}.$$

n is the length for the plaintext message.
Let (N, e) be Alice's RSA public and $d = e^{-1} \pmod{\phi(N)}$ be her private exponent.

Signature Generation
To sign a message $M \in \{0,1\}^n$, Alice performs the following steps:

1. $r \leftarrow_U \{0,1\}^{k_0}$; $s \leftarrow (G(r) \oplus M) \parallel H'(r \parallel M)$; $t \leftarrow H(s) \oplus r$;
2. if ($s \parallel t \geq N$) go to 1; ($*$ Probability for looping i times is 2^{-i} $*$)
3. $\sigma \leftarrow (s \parallel t)^d \pmod{N}$.

The signature is σ.

Signature Verification
To verify a signature σ, Bob performs the following steps:

1. $u \parallel v \parallel t \leftarrow \sigma^e \pmod{N}$ satisfying $|u| = n$, $|v| = k_1$, $|t| = k_0$;
2. $r \leftarrow H(u \parallel v) \oplus t$; $M \leftarrow G(r) \oplus u$;
3. output $\begin{cases} M \parallel \text{"True"} & \text{if } v = H'(r \parallel M) \\ \text{"False"} & \text{otherwise} \end{cases}$ ($*$ when "True" is in the trailing part of the output, the prefix string is the recovered message. $*$)

Fig. 5. The RSA-OAEP+ Signature Scheme

The RSA-OAEP+ Signature scheme is specified in Figure 5. From the scheme we can see that the bandwidth for message recovery is the same as that of the RSA-OAEP Signature. Hence, OAEP+ is also a very efficient padding scheme.

Theorem 2. *Let $GEN(1^k)$ be an RSA-OAEP+ Signature Scheme specified in Figure 5. If an adaptive chosen-message forger can break the scheme in time t with advantage Adv, then the RSA problem under (N, e) can be solved in time t' with advantage Adv' where*

$$t' \leq t + (q_s + q_G + g_H + q_{H'}) \cdot k^3,$$

$$Adv' \geq Adv - 2^{-n-k_1},$$

where $q_s, q_G, q_H, q_{H'}$ are the numbers of signing, G, H and H' oracle queries, respectively.

The technique for proving this theorem is similar to that for proving Theorem 1 and is given in Appendix A.

4 SAEP is Good

Simple-OAEP (SAEP) is a randomised padding scheme proposed by Boneh [4] with a proof that it creates a IND-CCA2 secure encryption scheme when used with the RSA or Rabin function. The padding scheme is illustrated in Figure 6. From the Figure we can see that the bandwidth for message recovery is the same as that for the OAEP.

We specify the SAEP Signature in Figure 7 and analyse its security.

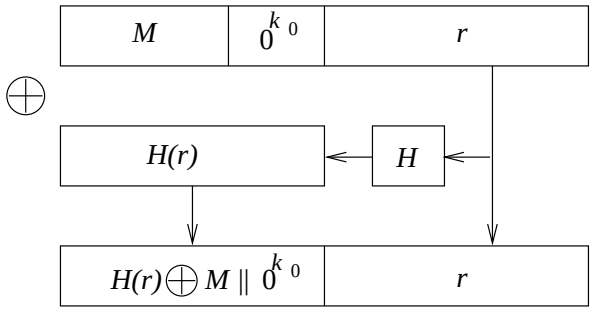

Fig. 6. The SAEP Padding

4.1 Unforgeability

Theorem 3. *Let $GEN(1^k)$ be an RSA-SAEP Signature Scheme specified in Figure 7. If an adaptive chosen-message forger can break the scheme in time t with advantage Adv, then the RSA problem under (N, e) can be solved in time t' with advantage Adv' where*

$$t' \leq t + (q_s + q_H) \cdot k^3,$$

$$Adv' \geq \frac{1}{q_s}(Adv - 2^{-n-k_0} - (q_H + q_s)^2 \cdot 2^{-n-k_0}),$$

where q_s, g_H are the numbers of signing, and H oracle queries, respectively.

From the expression on Adv' relating to Adv we can already see that our reduction proof for the SAEP signature scheme is no longer tight. That's why SAEP is not optimal.

Our proof technique is a variation to that of Coron et al for proving their UPS for the signature case [5]. In our reduction, we essentially simulate the RSA function into a random oracle.

Key Parameters
Let $(N, e, d, H, n, k_0, k_1) \leftarrow GEN(1^k)$ satisfy the following: (N, e, d) are RSA key material; $k = |N| = n + k_0 + k_1$ with 2^{-k_0} and 2^{-k_1} being negligible quantities; H is a hash function satisfying

$$H : \{0,1\}^{k_1} \mapsto \{0,1\}^{n+k_0}$$

n is the length for the plaintext message.
Let (N, e) be Alice's RSA public and $d = e^{-1} \pmod{\phi(N)}$ be her private exponent.

Signature Generation
To sign a message $M \in \{0,1\}^n$, Alice performs the following steps:

1. $r \leftarrow_U \{0,1\}^{k_1}$; $s \leftarrow H(r) \oplus (M \parallel 0^{k_0})$;
2. if ($s \parallel r \geq N$) go to 1; ($*$ Probability for looping i times is 2^{-i} $*$)
3. $\sigma \leftarrow (s \parallel r)^d \pmod{N}$.

The signature is σ.

Signature Verification
To verify a signature σ, Bob performs the following steps:

1. $s \parallel r \leftarrow \sigma^e \pmod{N}$ satisfying $|s| = n + k_0$, $|r| = k_1$;
2. output $\begin{cases} M \parallel \text{"True"} & \text{if } s \oplus H(r) = M \parallel 0^{k_0} \\ \text{"False"} & \text{otherwise} \end{cases}$
   ($*$ when "True" is in the trailing part of the output, the prefix string is the recovered message. $*$)

Fig. 7. The RSA-SAEP Signature Scheme

Proof Again, we assume that after $q_s + q_H$ queries, $\mathcal{F}$ will be able to output a valid forgery which is a new message signature pair, with a significant probability *Adv*.

Let $SAEP(M, r)$ denote a signature of message M output from the SAEP Signature scheme.

Top-level Description of the Reduction

1. A simulator $\mathcal{S}$ will run a forger $\mathcal{F}$ by inputting the latter with the public key (N, e, H, n, k_0, k_1) and by answering all the queries from the latter. It simulates the signing algorithm and RO H as to be described below.
2. $\mathcal{S}$ is given a random $\eta \in \mathbb{Z}_N^*$ to find η^d.
3. $\mathcal{S}$ selects uniformly at random an integer $j \in [1, q_s]$.
4. In the simulation of the signing algorithm, $\mathcal{S}$ maintains a counter i ($1 \leq i \leq q_s$) for the ith signing query M_i. The simulation is perfect when $i \neq j$. The simulation is maintained in such a way that when $i = j$, it holds $\eta = SAEP(M_i, r)^e \pmod{N}$ for some random number r which is *not* known to $\mathcal{S}$. Therefore, for this case of $i = j$, $\mathcal{S}$ will respond with a random value

$x < N$ as a valid signature, since there exists an unknown random nonce r satisfying $x = SAEP(M_j, r)$.

5. In the simulation of RO H (which is simulated perfectly all the time), $\mathcal{S}$ will maintain an H-list of query-response pairs to H. This list is initially empty.
6. Finally, $\mathcal{S}$ receives from $\mathcal{F}$ a forgery message-signature pair $(\hat{M}, z)$. If $\hat{M} = M_j$ then according to the ways of both simulations, $\eta = SAEP(\hat{M}, r)^e \pmod N$ and $\mathcal{S}$ outputs z.

Simulation of signing query For M_i queried by $\mathcal{F}$, there are two cases: (i) $i \neq j$, (ii) $i = j$. For (i), $\mathcal{S}$ generates random $x_i < N$ and sets $y_i = x^e$. For (ii), $\mathcal{S}$ sets $y_i = \eta$. In both cases $\mathcal{S}$ parses $y_i = s_i \parallel r_i$, computes $H(r_i) = (M_i \parallel 0^{k_0}) \oplus s_i$, record $(r_i, H(r_i))$ in H-list. Now $\mathcal{S}$ answers $\mathcal{F}$ with x_i for case (i), and answers a random $x < N$ for case (ii).

Simulation of H query Let r_j be an RO query to H. If r_j is already in H-list, then $\mathcal{S}$ answers with $H(r_j)$ picked from H-list. Otherwise, $\mathcal{S}$ performs the following steps.

$\mathcal{S}$ picks at random a string $H(r_j) \in \{0,1\}^{n+k_0}$ and answers $\mathcal{F}$ with $H(r_j)$. $\mathcal{S}$ also stores $(r_j, H(r_j))$ in H-list. The storage is indexed by integer j.

Analysis Let $(\hat{M}, z)$ be the forgery sent by $\mathcal{F}$. Let $z^e \pmod N = \hat{s} \parallel \hat{r}$ with $|\hat{s}| = n + k_0$ and $|\hat{r}| = k_1$. If $\hat{M}$ was never queried for signature, then probability for $\hat{M} \parallel 0^{k_0} = H(r) \oplus \hat{s}$ to hold for each r in H-list is 2^{-n-k_0}. Otherwise, let $\hat{M} = M_i$ such that M_i was a signing query. If $i = j$, then $\eta = SAEP(M_j, \hat{r})^e \pmod N$, and $\mathcal{S}$ succeeds in rooting η.

Now let us examine the probability for $\mathcal{S}$ to succeed. We have seen that for any message which was not signing queried, the probability for $\mathcal{F}$ to output a valid forgery using any random nonce (whether H-queried or not) is 2^{-n-k_0}. Thus, in order to have a non-negligible advantage to forge a signature, $\mathcal{F}$ must forge a signature for one of $M_1, M_2, \ldots, M_{q_s}$ (i.e., for one of the messages which was signing-queried). We notice that for these old messages, as long as $\mathcal{F}$ can output a new signature value σ which has never been output from any previous run of the signing algorithm (simulated or real), the (M_i, σ) will be regarded as a valid forgery.

Now we condition on $i = j$ (i.e., M_j is the only message for $\mathcal{F}$ to forge a signature). In order for the forgery z to be new, $\mathcal{F}$ is not allowed to use $\mathcal{S}$'s random answer x. Because the both simulations are perfect until $\mathcal{F}$ succeeds its forgery task (see *Remark 2*), unless some s appears twice. The probability for some s appearing twice is less than $(q_H + q_s)^2 \cdot 2^{-n-k_0}$. Consequently we have

$$Adv'(i = j) \geq Adv - 2^{-n-k_0} - (q_H + q_s)^2 \cdot 2^{-n-k_0}.$$

Finally, removing the condition $i = j$, we have

$$Adv' \geq \frac{1}{q_s}(Adv - 2^{-n-k_0} - (q_H + q_s)^2 \cdot 2^{-n-k_0}).$$

The time complexity is obvious from our analysis. □

Discussions

- We notice that in this security proof for the SAEP signature scheme we have to use the following more general notion of forgery: $\sigma = SAEP(M, r)$ is regarded as a valid forgery as long as σ is new even though M can be an old message. This notion of forgery is a reasonable one for a probabilistic signature algorithm, although in our security proofs for our other two signature schemes we do not need to use this notion of forgery.
- Same as our discussion in *Remark 2*, $\mathcal{F}$ can only detect an imperfection of the signature simulation, i.e., detect that the simulated signature reply on $\hat{M} = M_j$ (which was a random value $x < N$) was a not a correct one, when it succeeds its forgery z (again, too late!). Before $\mathcal{F}$ succeeds, x can indeed be an *existentially* valid signature on M_j for some unknown random nonce r.
- In order to root η with an advantage similar to Adv, $\mathcal{S}$ has to run $\mathcal{F}$ q_s times. So the overall time for rooting η with advantage Adv will be

$$q_s t' \leq q_s t + (q_s + q_H) q_s \cdot k^3 \approx q_s^2 \cdot k^3.$$

  Consider $q_s \approx 2^{40}$, i.e., consider that $2^{40} \cdot k^3$ is reasonably affordable by a dedicated forger. Then $\mathcal{S}$ can only solve the RSA problem in time $2^{80} \cdot k^3$ for a k-bit modulus. This does not form a nice contradiction for the case of $k = 1024$. That is why we say the this reduction is not an efficient one.

5 Conclusion

We have shown that OAEP and two of its inspired encryption padding schemes can be used to create RSA signature schemes with desirably high bandwidth for message recovery. We have seen that OAEP is actually a very good padding scheme. All three schemes have the same and desirably high bandwidth for message recovery: message bits are 84% of the modulus bits for the usual size of key setting, with OAEP and OAEP+ being optimal, meaning that they have tight security proofs for the signature case. SAEP is good and simple, however, we have not been able to provide a tight security proof for it. We have not been successful to provide a security proof for SAEP+ in the signature usage.

Our work shows that UPS schemes from the OAEP-family padding schemes are considerably better than that from the PSS-R padding scheme.

References

1. M. Bellare and P. Rogaway. Random oracles are practical: a paradigm for designing efficient protocols. In *First ACM Conference on Computer and Communications Security*, pages 62–73, New York, 1993. ACM Press.

2. M. Bellare and P. Rogaway. Optimal asymmetric encryption. In A. de Santis, editor, *Advances in Cryptology — Proceedings of EUROCRYPT'94, Lecture Notes in Computer Science 950*, pages 92–111. Springer-Verlag, 1995.
3. M. Bellare and P. Rogaway. The exact security of digital signatures – How to sign with RSA and Rabin. In U. Maurer, editor, *Advances in Cryptology — Proceedings of EUROCRYPT'96, Lecture Notes in Computer Science 1070*, pages 399–416. Springer-Verlag, 1996.
4. D. Boneh. Simplified OAEP for the RSA and Rabin functions. In J. Killian, editor, *Advances in Cryptology — Proceedings of CRYPTO'01, Lecture Notes in Computer Science 2139*, pages 275–291. Springer-Verlag, 2001.
5. J.-S. Coron, M. Joye, D. Naccache, and P. Paillier. Universal padding schemes for RSA. In M. Yung, editor, *Advances in Cryptology — Proceedings of CRYPTO'02, Lecture Notes in Computer Science 2442*, pages 226–241. Springer-Verlag, 2002.
6. J.-S. Coron and M. Joye and D. Naccache and P. Paillier. Universal Padding Schemes for RSA. Full version from http://eprint.iacr.org/2002/115/. 2002.
7. J.-S. Coron. Optimal security proofs for PSS and other signature schemes. In L.R. Knudsen, editor, *Advances in Cryptology — Proceedings of EUROCRYPT'02, Lecture Notes in Computer Science 2332*, pages 272–287. Springer-Verlag, 2002. In Advances in Cryptology - CRYPTO 2002, volume 2442 of Lecture Notes in Computer Science, pages 226-241. Springer-Verlag, 2002.
8. E. Fujisaki, T. Okamoto, D. Pointcheval, and J. Stern. RSA-OAEP Is secure under the RSA assumption. In J. Killian, editor, *Advances in Cryptology — Proceedings of CRYPTO'01, Lecture Notes in Computer Science 2139*, pages 260–274. Springer-Verlag, 2001.
9. S. Goldwasser, S. Micali, and R.L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal of Computing*, 17(2):281–308, 1988.
10. J. Malone-Lee and W. Mao. Two birds one stone: Signcryption using RSA. *Proceedings of the RSA Conference 2003*, Springer-Verlag, April 2003.
11. V. Shoup. OAEP Reconsidered. In J. Killian, editor, *Advances in Cryptology — Proceedings of CRYPTO'01, Lecture Notes in Computer Science 2139*, pages 239–259. Springer-Verlag, 2001.

A Proof of Theorem 2

The proof is very similar to that given in §2.1.

Again, we assume that after $q_s + q_G + q_H + q_{H'}$ queries, $\mathcal{F}$ will be able to output a valid forgery which is a new message signature pair, with a significant probability Adv.

The simulator $\mathcal{S}$ will run $\mathcal{F}$ by inputting the latter with the public key (N, e, G, H, H') and by answering all the queries from the latter. $\mathcal{S}$ will pick a random point $\eta \in \mathbb{Z}_N^*$ and use $\mathcal{F}$ to find the e-th root of η, also with a significant probability Adv'. The value of Adv' will be given later. Similar to the case of simulations in §2.1, $\mathcal{S}$ will maintain respective RO lists and an R-list.

Remark 3. We follow Shoup's reasonable stipulation by assuming without loss of generality that whenever $\mathcal{F}$ makes a query of the form $H'(r \parallel M)$ for any $r \in \{0,1\}^{k_0}$, $M \in \{0,1\}^n$, then $\mathcal{F}$ has previously made the query $G(r)$.

Simulation of signing query For M queried by $\mathcal{F}$ for a signature, $\mathcal{S}$ picks at random $r \in \{0,1\}^{k_0}$ and $x \in \{0,1\}^k$ with $x < N$. It responds to $\mathcal{F}$ with x. Let $y = x^e \pmod N$ where y is parsed as

$$y = u \parallel v \parallel t$$

with $|u| = n$, $|v| = k_1$ and $|t| = k_0$. $\mathcal{S}$ further sets

$$G(r) = M \oplus u,$$

$$H(u \parallel v) = r \oplus t,$$

$$H'(r \parallel M) = v.$$

$\mathcal{S}$ stores $(r, G(r))$ in G-list, $(u \parallel v, H(u \parallel v))$ in H-list, and $(r \parallel M, H'(r \parallel M))$ in H'-list.

Simulation of H' query Let $r_i \parallel M_i$ be an RO query to H'. From *Remark 3* we know that r_i must have been RO queried to G. So $\mathcal{S}$ has in its possession r_i.

If $r_i \parallel M_i$ has already been H'-queried (i.e., the concatenated string is in H'-list) then $\mathcal{S}$ answers with $H(r_i \parallel M_i)$ picked from H'-list. Otherwise, $\mathcal{S}$ performs the following steps.

$\mathcal{S}$ picks at random $x_i \in \{0,1\}^k$ with $x_i < N$. $\mathcal{S}$ sets $y_i = \eta x_i^e \pmod N$ where y_i is parsed as

$$y_i = s_i \parallel t_i = u_i \parallel v_i \parallel t_i$$

with $|u_i| = n$, $|v_i| = k_1$ and $|t_i| = k_0$. $\mathcal{S}$ further sets

$$H'(r_i \parallel M_i) = r_i \oplus t_i,$$

$$H(s_i) = r_i \oplus t_i.$$

Now $\mathcal{S}$ answers $\mathcal{F}$ with $H'(r_i \parallel M_i)$. It stores the pair $(r_i \parallel M_i, H'(r_i \parallel M_i))$ in H'-list, the pair $(s_i, H(s_i))$ in H-list, and x_i in R-list. The storage in these lists is indexed by integer i.

Simulation of G query Let r_j be a RO query to G. If r_j has already been queried (i.e., it is in the G-list) then $\mathcal{S}$ answers with $G(r_j)$ picked from G-list. Otherwise, $\mathcal{S}$ performs the following steps.

$\mathcal{S}$ picks at random a string $G(r_j) \in \{0,1\}^n$ and answers $\mathcal{F}$ with $G(r_j)$. $\mathcal{S}$ also stores $(r_j, G(r_j))$ in the G list. The storage is indexed by the integer j.

Simulation of H query Let s_k be an RO query to H. If s_k is already in H-list, then $\mathcal{S}$ answers with $H(s_k)$ picked from H-list. Otherwise, $\mathcal{S}$ performs the following steps.

$\mathcal{S}$ picks at random a string $H(s_k) \in \{0,1\}^{k_0}$ and answers $\mathcal{F}$ with $H(s_k)$. $\mathcal{S}$ also stores $(s_k, H(s_k))$ in the H list. The storage is indexed by integer k.

One may verify that in the ROM, all the above simulations are perfect, except when an event $\mathsf{AskH'}$ occurs. This event is when an RO query is made to H' where the query is related to a successful forgery. We shall see more details about this event in a moment.

Analysis

After $\mathcal{F}$ has completed its training (i.e, after $q_s + q_G + q_H + q_{H'}$ queries), it outputs a valid message-signature pair $(\hat{M}, z)$ with probability Adv. We denote by FWin this event.

The event $\mathsf{AskH'}$ mentioned earlier is as follows: for

$$z^e \pmod N = \hat{s} \parallel \hat{t} = \hat{u} \parallel \hat{v} \parallel \hat{t} \tag{4}$$

with $|\hat{u}| = n$, $|\hat{v}| = k_1$, and $|\hat{t}| = k_0$, $\hat{v}$ is in H'-list. When this event occurs, an imperfection in the simulation is exposed. However, as we have discussed in *Remark 2*, now it's too late for $\mathcal{F}$ to discover the imperfection. We have

$$\begin{aligned} Adv &= \Pr[\mathsf{FWin}] \\ &= \Pr[\mathsf{FWin} \mid \mathsf{AskH'}] \Pr[\mathsf{AskH'}] + \Pr[\mathsf{FWin} \mid \neg\mathsf{AskH'}] \Pr[\neg\mathsf{AskH'}]. \end{aligned} \tag{5}$$

In the ROM, the information inside the "wrapping" of the RSA trapdoor function in the event $\neg\mathsf{AskH'}$ is random. Therefore, the second term in the right-hand side of (5) is a negligible quantity $neg < 2^{-n-k_1}$. So we have

$$\Pr[\mathsf{AskH'}] = (Adv - neg)/\Pr[\mathsf{FWin} \mid \mathsf{AskH'}] > Adv - 2^{-n-k_1}. \tag{6}$$

That is, in the event FWin, $\hat{v}$ defined in (4) is in $\mathcal{S}$'s H'-list with probability at least $Adv - 2^{-n-k_1}$. Thus, $\mathcal{F}$ can search through H'-list and find the correct index ℓ, and then find x_ℓ from R-list. From the simulations, we know

$$z^e = \eta x_\ell^e \pmod N.$$

So $\eta^d = z/x_\ell \pmod N$ is discovered as desired.

Clearly, we have $t' \leq t + (q_s + q_G + q_H + q_{H'}) \cdot k^3$ and $Adv' = \Pr[\mathsf{AskH'}]$ shown in (6). □