Evaluation of Security Level of Cryptography: RSA-OAEP, RSA-PSS, RSA Signature

Alfred Menezes
University of Waterloo
Contact: ajmeneze@uwaterloo.ca

December 14, 2001

Contents

1 Executive Summary

The report is an evaluation of three RSA-based cryptographic schemes:

1. The RSA-OAEP public-key encryption scheme as specified in PKCS#1 v2.1 [46].
2. The RSA-PKCS1-v1.5 signature scheme as specified in PKCS# v2.1 [45].
3. The RSA-PSS signature scheme as specified in PKCS#1 v2.1 [46].

2 Introduction

The RSA cryptosystem was invented by R. Rivest, A. Shamir and L. Adleman in 1977 [42], and is the most widely used public-key cryptographic system. It supports both confidentiality and digital signatures, and its security is based on the difficulty of factoring integers.

Even though the RSA system has been widely studied for 25 years, it is only now that cryptographers are beginning to understand how to securely implement RSA encryption and signature schemes. Numerous schemes for RSA encryption and signatures are being proposed in the cryptographic literature, and numerous proposals are being submitted to standards bodies. RSA encryption and signatures have already been standardized by some standards organizations, including the following.

1. American National Standards Institute (ANSI X9.31 [1], ANSI X9.44 [2]).
2. Institute of Electrical and Electronics Engineers (IEEE 1363-2000 [26]).
3. International Standards Organization (ISO 9796-1 [27], ISO 9796-2 [28]).
4. Public-Key Cryptography Standards (PKCS#1 v2.1 [46]).
5. U.S. government's National Institute of Standards and Technology (FIPS 186-2 [38]).

This report provides an evaluation of the RSA-OAEP encryption scheme, the RSA-PKCS1-v1.5 signature scheme, and the RSA-PSS signature scheme as specified in PKCS#1 v2.1 [46]. In particular, we comment on what has been *proven* about the security of these schemes.

The remainder of this report is organized as follows. In Section 3, we review the basic RSA encryption and signature schemes and summarize the results known about the security of these basic schemes. In Section 4, we analyze the RSA-OAEP encryption scheme. The RSA-PKCS1-v1.5 and RSA-PSS signature schemes are analyzed in Section 5 and Section 6. Our conclusions are stated in Section 7.

REMARK. A large portion of the analysis presented in this report also applies to the Rabin-Williams encryption and signature schemes [40, 53] where an even encryption exponent is used. However, this report will not make any future reference to the Rabin-Williams schemes.

3 General Remarks on the Security of RSA

3.1 RSA Key Generation

The RSA key generation procedure is common to all RSA-based cryptographic schemes. To generate an RSA key pair, each entity A does the following:

1. Randomly select two large distinct primes p and q of the same bit length.
2. Compute $n = pq$ and $\phi(n) = (p-1)(q-1)$.
3. Select an arbitrary integer e, $1 < e < \phi(n)$, such that $\gcd(e, \phi(n)) = 1$.
4. Compute d, $1 < d < \phi(n)$, such that $ed \equiv 1 \pmod{\phi(n)}$.
5. A's public key is (n, e); A's private key is d.

NOTES.

1. n is called the *RSA modulus*.
2. e is called the *encryption exponent*.
3. d is called the *decryption exponent*.
4. If n is a k-bit integer, then the RSA scheme is referred to as *k-bit RSA*.
5. RSA public-key operations (encryption and signature verification) involve modular exponentiations by the number e—computations of the form $M^e \bmod n$. To speed up public-key operations, a widely-employed strategy is to select a small encryption exponent e, such as $e = 3$ or $e = 2^{16} + 1$. With such a choice, RSA public-key operations are much faster than RSA private-key operations (decryption and signature generation). There are, however, some security concerns with selecting small encryption exponents (see §3.4.3). Note also that RSA private-key operations cannot be sped up by selecting a small decryption exponent d without incurring a significant security risk (see §3.4.4).

It is known that the problem of finding the private key d from a public key (n, e) is polynomial time equivalent to the problem of factoring n. Thus, assuming that factoring n is intractable, the private key d cannot be computed from the public key (n, e).

3.2 Basic RSA Encryption Scheme

This subsection describes the basic RSA public-key encryption scheme and explains why the basic scheme is not considered to be secure.

3.2.1 Description

Let (n,e) be A's RSA public key, and let d be A's corresponding private key.

Basic RSA Encryption Scheme. (B sends a message m to A)

1. **Encryption**. B does the following:
   - 1.1. Obtain an authentic copy of A's public key (n,e).
   - 1.2. Represent the message as an integer m in the interval $[0,n-1]$; if the message is too long, break it into blocks of the appropriate size.
   - 1.3. Compute $c = m^e \bmod n$.
   - 1.4. Send c to A.
2. **Decryption**. A does the following:
   - 2.1. Use the private key d to recover $m = c^d \bmod n$.

3.2.2 Security

Ideally, a public-key encryption scheme should be *semantically secure against adaptive chosen-ciphertext attacks* [23, 41]. Informally, this means that an adversary can learn nothing about the plaintext corresponding to a given ciphertext c, even when the adversary is allowed to obtain the plaintexts corresponding to ciphertexts (not equal to c) of its choice. We will henceforth denote this security notion as IND-CCA ("indistinguishability against adaptive chosen-ciphertext attacks").

ATTACKS ON BASIC RSA ENCRYPTION. Basic RSA encryption, as described above, is not used in practice because of various security weaknesses. For example, if an adversary knows that the plaintext m belongs to a small or predictable message space, then the adversary can decrypt ciphertext c by simply encrypting all possible plaintext messages until c is obtained. Other potential weaknesses arise from the so-called *homomorphic property* of basic RSA: if c_1 and c_2 are the RSA encryptions of plaintext messages m_1 and m_2, respectively (so $c_1 = m_1^e \bmod n$ and $c_2 = m_2^e \bmod n$), then $c_1c_2 \equiv (m_1m_2)^e \pmod n$, from which it follows that $c = c_1c_2 \bmod n$ is the encryption of $m = m_1m_2 \bmod n$. (See also [12] for further discussion on the weakness of basic RSA encryption.)

BLEICHENBACHER'S ATTACK ON PKCS#1 V1.5 RSA ENCRYPTION. The above attacks, and other related attacks, can be circumvented in practice by imposing some structural constraints on plaintext messages. This is typically done by padding the message in some specified way prior to encryption. However, the following attack by Bleichenbacher [8] on PKCS#1 v1.5 RSA encryption [44] shows that the padding technique employed must be carefully chosen.

PKCS#1 v1.5 [44] is a widely-deployed standard for RSA encryption. It defines an encoding method for RSA encryption, where a data value D (which is often a symmetric key) is converted to an integer prior to encryption with an RSA public key. If k is the length in bytes of the recipient's RSA modulus n, the encoding method produces a k-byte string from D of the form EB=00 || 02 || PS || 00 || D, where 00 and 02 are bytes with value 0 and 2, and PS is a padding string consisting of $k-|D|-3$ pseudorandomly generated nonzero bytes. The byte string EB is then converted to an integer m which is encrypted with basic RSA by the usual exponentiation: $c = m^e \bmod n$. The recipient decrypts the ciphertext c to obtain the integer m, converts m to EB, checks that the result has the expected form, and if so, recovers the data value D. In 1998, Bleichenbacher [8] devised a chosen-ciphertext attack on the PKCS#1 encoding method which succeeds in the case where the recipient is willing to reveal whether or not a decryption operation has succeeded (i.e., the result EB has the expected form) or failed. For a 1024-bit modulus, the total number of chosen ciphertexts required is about 2^{20}.

Bleichenbacher's attack is especially significant against interactive key establishment protocols such as SSL (Secure Sockets Layer), where a server is willing to process many messages and may reveal the success or failure of an operation. The attack necessitated a patch to numerous SSL implementations, and caused RSA Laboratories to change PKCS#1 to support Optimal Asymmetric Encryption Padding (OAEP). RSA-OAEP will be presented and analyzed in Section 4.

3.3 Basic RSA Signature Scheme

This subsection describes the basic RSA signature scheme and explains why the basic scheme is not considered to be secure.

3.3.1 Description

Let (n,e) be A's RSA public key, and let d be A's corresponding private key.

Basic RSA Signature Scheme. (A signs a message m for B)

1. **Signature Generation**. A does the following:
    - 1.1. Compute $M = H(m)$ where H is a cryptographic hash function such as SHA-1.
    - 1.2. Use the private key d to compute $s = M^d \bmod n$.
    - 1.3. Send the signature s and the message m to B.
2. **Signature Verification**. B does the following:
    - 2.1. Obtain an authentic copy of A's public key (n,e).
    - 2.2. Compute $M = H(m)$.
    - 2.3. Compute $M' = s^e \bmod n$.
    - 2.4. Accept A's signature provided that $M = M'$.

3.3.2 Security

Ideally, a signature scheme should be *existentially unforgeable against chosen-message attacks* [24]. Informally, this means that an adversary who is able to obtain entity A's signatures for any messages of its choice is unable to successfully forge A's signature on another message.

ATTACKS ON BASIC RSA SIGNATURES. Even if the hash function H is a cryptographically secure hash function (e.g., H is one-way and collision resistant, or if H is a random function), the basic RSA signature scheme may still be insecure. For example, if the hash values $H(m)$ are 160-bit strings, then if an active adversary could somehow obtain the signatures of messages whose hash values are smooth integers (i.e., have only small prime factors), then the adversary could use these signatures to forge the signature of a new message. This vulnerability was first observed by Desmedt and Odlyzko [20]; see also [37] for further discussion on the weakness of basic RSA signatures.

Such attacks can be circumvented in practice by either using a hash function whose range is the entire set of integers $[0, n-1]$ (this variant of RSA is called Full Domain Hash (FDH) and is analyzed in [7]), or formatting the hash value prior to applying the RSA function. Two ways of formatting the hash values are RSA-PKCS1-v1.5 and RSA-PSS which will be presented and analyzed in Sections 5 and 6.

3.4 Other Attacks on Basic RSA

As mentioned in Section 3.1, any RSA-based cryptographic scheme can be totally broken if the private key d can be efficiently recovered from the public key (n, e). Every method for this task is known to be equivalent in difficulty to the problem of factoring n. However, it must be noted there are other mathematical ways in which an RSA-based cryptographic scheme can be attacked. The various approaches can be categorized as follows:

1. Factoring the modulus n (Section 3.4.1).
2. Finding e^{th} roots modulo n (Section 3.4.2).
3. Attacks when small encryption exponents are used (Section 3.4.3).
4. Attacks when small decryption exponents are used (Section 3.4.4).

3.4.1 Integer Factorization

This section provides a brief summary of the state-of-the-art in algorithms for the integer factorization problem by which the RSA private key can be recovered from a corresponding RSA public key, thus totally breaking any RSA-based scheme.

PROBLEM DEFINITION. The *integer factorization problem (IFP)* is the following: given an integer n which is known to be the product of two distinct primes p and q, determine p and q.

KNOWN ATTACKS. While the integer factorization problem has received attention over the centuries from well-known mathematicians such as Fermat and Gauss, extensive practical study commenced only in the late 1970's following the invention of the RSA cryptosystem.

There are two general types of factoring algorithms: *special-purpose* and *general-purpose*. Special-purpose factoring algorithms attempt to exploit special features of the number n being factored. In contrast, the running times of general-purpose factoring algorithms depend only on the size of n.

SPECIAL-PURPOSE FACTORING ALGORITHMS. Special-purpose factoring algorithms include *Pollard's* $p-1$ *algorithm* which is only efficient if either $p-1$ or $q-1$ is smooth, and *Pollard's* $p+1$ *algorithm* which is only efficient if either $p+1$ or $q+1$ is smooth.

One of the most powerful special-purpose factoring algorithms is the *elliptic curve factoring method (ECM)* which was invented by H. Lenstra in 1985 [35]. Its running time depends on the size of the prime factors of n, so the algorithm tends to find small factors first. The largest prime factor found thus far by ECM is a 55-decimal digit (187-bit) prime factor of a 112-decimal digit (371-bit) number reported by Izumi Miyamoto in October 2001. If both the prime factors of an RSA modulus n are at least 256 bits in length, as is the case with 512-bit RSA, then it is extremely unlikely that the ECM will be able to factor n. Thus, the ECM does not have a significant impact on the security of RSA.

Another powerful special-purpose factoring algorithm is the Special Number Field Sieve (SNFS). The SNFS requires a polynomial of a special form that currently can only be efficiently found for integers having certain properties, e.g., integers of the form $2^a \pm b$ where a and b are small. For this reason, the SNFS is currently only applicable to factoring integers of a special form, and is not applicable to factoring RSA moduli. When SNFS is applicable, however, it is by far the most efficient factoring method currently known. In April 1999, the SNFS was used to factor the 210-decimal (698-bit) number $(10^{211}-1)/9$. The expected running time of the SNFS for factoring an integer n is

$$O(\exp((1.526+o(1))(\log n)^{1/3}(\log\log n)^{2/3})).$$

Here $o(1)$ is an (undetermined) expression that becomes negligible as n becomes larger.

GENERAL-PURPOSE FACTORING ALGORITHMS. Since the invention of RSA, there have been two significant developments in algorithms for factoring numbers of the type used in RSA: the discovery of the quadratic sieve (QS) factoring algorithm in 1982 and the discovery of the number field sieve (NFS) factoring algorithm in 1990. While the number field sieve was known to be faster than the quadratic sieve in theory, it was only in 1996 that it was accepted as being the superior method in practice and superseded the quadratic sieve method as the champion of factoring algorithms.

The NFS has two main parts: the sieving stage and the matrix stage. The sieving stage, where most of the work is done, can be efficiently parallelized on a network of workstations. In contrast, existing techniques to solve the matrix stage seem to work well only when done on a single large parallel computer. The expected running time of the NFS for factoring an integer n is

$$O(\exp((1.923 + o(1))(\log n)^{1/3}(\log\log n)^{2/3})).$$

Note that the running time of the SNFS differs from ths running time of the NFS in that the constant 1.526 replaces the constant 1.923. For this reason, the SNFS is much faster than the NFS. Like the SNFS, the NFS also requires one to find a suitable polynomial with appropriate properties.

EXPERIMENTAL RESULTS. Table 1 summarizes the progress in integer factorization in the 1990's as measured by the size of numbers factored from the RSA Challenge list [43]. The results provide conclusive evidence for the superiority of the NFS algorithm over the QS algorithm. It is also important to note that the dramatic advances in the sizes of numbers factored with the NFS in 1999 were due not to general increases in computer speed, but rather to some significant algorithmic improvements in the NFS by Peter Montgomery and Brian Murphy.

Number Factored	Size in Bits	Date Factored	Algorithm Used
$10^{71}-1$	236	1984	QS
RSA-100	330	April 1991	QS
RSA-110	364	April 1992	QS
RSA-120	397	June 1993	QS
RSA-129	425	April 1994	QS
RSA-130	430	April 1996	NFS
RSA-140	463	February 1999	NFS
RSA-155	512	August 1999	NFS

Table 1: Recent progress in factoring.

THE FACTORIZATION OF RSA-155. On August 22 1999, a team of scientists from six different countries led by Herman te Riele of CWI (Amsterdam) found the factors of RSA-155, a 155-decimal (512-bit) from the RSA Factoring Challenge [15].

RSA-155 was factored using the NFS. The sieving was done on about 300 SGI, SUN, Pentium II, and Digital workstations running at 175-500 MHz. The sieving stage was completed in 3.5 months. The matrix stage took about 40 days of which 9 days were spent on a Cray C916 supercomputer.

The total amount of computer time spent was estimated to be the equivalent of 8000 MY. (1 MY is the computational power obtained by running for 1 year a machine that performs 1 million instructions per second.) While this was a substantial computational task, it was still only slightly more than the 5000 MY that were expended in the factorization of RSA-129 in 1994 using the QS algorithm. Since the RSA-129 number is in fact 87 bits smaller than the RSA-155 number, this provides compelling evidence for the superiority of the NFS over the QS.

THE FUTURE OF FACTORING. Because of the sophisticated nature of the mathematics employed in the NFS and the complicated nature of the algorithm itself, it will be a few years before experts fully understand the capabilities of the NFS for factoring large integers. Indeed, one can reasonably expect that current implementations of NFS can be used to factor numbers of bitsize 600 (or more) within a few months. And this does not take into account improvements in the algorithm itself that are likely to be made in the next year or two.

The general conclusion is that 512-bit RSA is already very insecure, and that it will be possible for covert efforts (involving only a few people at a single institution, and thus not easily monitored) to crack 768-bit RSA moduli in a few years.

SELECTING PARAMETERS FOR LONG-TERM SECURITY. Lenstra and Verheul [34] performed an extensive and careful study of the key sizes for symmetric-key encryption schemes, RSA, discrete logarithm systems, and elliptic curve systems. Their study incorporates both software and hardware attacks, and takes into account the continual improvements in hardware, and hardware costs. Assuming that the NFS remains the best algorithm for integer factorization, they estimate that 1024-bit RSA will provide the same level of security in the year 2002 as the Data Encryption Standard (DES) provided in the year 1982. (DES was considered very secure in 1982 for banking applications.) Similarly, 2048-bit RSA will provide the same level of security in the year 2023 as DES provided in the year 1982, and 3072-bit RSA will provide the same level of security in the year 2038 as DES provided in the year 1982.

In a recent study, Lenstra [33] provides estimates of RSA key lengths required to provide equivalent levels of security as the common symmetric-key encryption schemes DES, 2K3DES (two key Triple-DES), 3K3DES (three key Triple-DES), AES-128 (128-bit Advanced Encryption Standard), AES-192 and AES-256 [39]. Lenstra's estimates are reproduced in Table 2. The entries of the table are to be interpreted as follows. The entry of 3296 for AES-128 in the year 2020 means that one should used a 3296-bit RSA modulus to protect a 128-bit symmetric key if in the year 2020 one desires the same amount of resistance to the NFS algorithm for factoring RSA moduli as exhaustive key search on AES-128.

Year	DES	2K3DES	3K3DES	AES-128	AES-192	AES-256
2001	416	1333	1941	2644	6897	13840
2010	518	1532	2189	2942	7426	14645
2020	647	1773	2487	3296	8042	15574
2030	793	2035	2807	3675	8689	16538

Table 2: Matching RSA modulus sizes.

3.4.2 Finding e^{th} Roots Modulo n

All RSA encryption and signatures schemes can also be totally broken if it is easy to find e^{th} roots modulo n. This is sometimes known as the *RSA problem*: given n, e, and an integer c, find an integer m such that $m^e \equiv c \pmod{n}$.

Clearly, if n can be factored, then an adversary can compute the private key d and thus efficiently solve the RSA problem. Hence, the RSA problem is no harder than the integer factorization problem. What is more desirable with respect to the security of RSA is that the RSA problem itself be difficult, preferably as difficult as the integer factorization problem. However, no proof of this is known.

In fact, Boneh and Venkatesan [13] provided evidence that if a small encryption exponent (for example, $e = 3$) is used, then the RSA problem cannot be equivalent to the integer factorization problem, i.e., the RSA problem is *easier*. Boneh and Venkatesan proved that *any* polynomial time reduction of the integer factorization problem to the RSA problem with $e = 3$ that uses only algebraic operations can be efficiently converted to a polynomial time algorithm for integer factorization. Thus, if one believes that integer factorization cannot be solved in polynomial time, then one can conclude that there is no polynomial time reduction of the integer factorization problem to the RSA problem with $e = 3$.

We emphasize that the results of Boneh and Venkatesan do not expose any specific weaknesses in RSA (with or without a small encryption exponent); however the results suggest that the RSA problem *may* be easier than the integer factorization problem, especially for small encryption exponents.

We therefore recommend that RSA encryption and signature schemes should not use $e = 3$. Encryption exponents $e = 2^{16} + 1$ can be used, but even larger exponents are to be preferred.

3.4.3 Small Encryption Exponents

An attractive option for speeding up RSA public key operations (encryption and signature verification) is to select a small encryption exponent e. A variety of attacks have been discovered in recent years against the use of RSA with small encryption exponents. These attacks are summarized in this subsection.

BROADCAST ATTACKS. An old attack of Håstad [25] works as follows. Suppose that entity A wishes to send the same message m to three parties B_1, B_2 and B_3, whose RSA public keys are $(n_1, e_1 = 3)$, $(n_2, e_2 = 3)$ and $(n_3, e_3 = 3)$, respectively. We assume that $m < n_1$, $m < n_2$ and $m < n_3$. A computes $c_1 = m^3 \bmod n_1$, $c_2 = m^3 \bmod n_2$, $c_3 = m^3 \bmod n_3$, and transmits these ciphertexts. Since the moduli n_1, n_2, n_3 are most likely pairwise relatively prime, there exists a unique integer x, $0 \le x < n_1 n_2 n_3$, such that $x \equiv c_1 \pmod{n_1}$, $x \equiv c_2 \pmod{n_2}$, and $x \equiv c_3 \pmod{n_3}$; an adversary can efficiently find x. Since $0 \le m^3 < n_1 n_2 n_3$, it is the case that $x = m^3$. Thus, the adversary can efficiently recover m by computing the integer cube root of x.

The attack has also been extended to situations (see Håstad [25] and Boneh [9]) where certain forms of padding are applied to the message prior to encryption, for example by appending i to the message before broadcasting to party i.

Such broadcast attacks can be circumvented by selecting an encryption exponent e which is larger than the maximum number of parties to which a message might be broadcast, or by using appropriate random padding. More generally, such broadcast attacks can be circumvented by using an RSA variant, such as RSA-OAEP, that is semantically secure against adaptive chosen-ciphertext attacks. It then follows from the work of Bellare, Boldyreva and Micali [3] that the encryption scheme is secure in a distributed environment and thus resists all Håstad-like attacks.

RELATED-MESSAGE ATTACKS. Suppose that two or more plaintext messages which have a (known) polynomial relationship (e.g. m_1 and m_2 might be *linearly related*: $m_1 = am_2 + b$) are encrypted with the same small encryption exponent (e.g., $e = 3$ or $e = 2^{16} + 1$) and the same modulus n. Coppersmith et al. [17] presented a new class of attacks on RSA which enable a passive adversary to recover such plaintext from the corresponding ciphertext. This attack is of practical significance because various cryptographic protocols have been proposed which require the encryption of polynomially related messages. Note that these attacks are different from the broadcast attacks considered above where the same plaintext is encrypted under different public keys.

Coppersmith [16] presented an efficient algorithm for finding a root of a polynomial of degree k over the integers modulo n, where n is an RSA-like modulus, provided that there is a root smaller than $n^{1/k}$. The algorithm yielded the following two attacks on RSA with small encryption exponents. If $e = 3$ and if an adversary knows a ciphertext c and more than $2/3$ of the plaintext m corresponding to c, then the adversary can efficiently recover the rest of m. Suppose now that messages are padded with random bitstrings and encrypted with exponent $e = 3$. If an adversary knows two ciphertexts c_1 and c_2 which correspond to two encryptions (under the same public key) of the same message m (with different padding), then the adversary can efficiently recovery m, provided that the padding is less than $1/9$ of the length of n. The latter attack suggests that caution must be exercised when using random padding in conjunction with a small encryption exponent.

3.4.4 Small Decryption Exponents

An attractive option for speeding up RSA private key operations (decryption and signature generation) is to select a relatively small decryption exponent d.

Suppose that the RSA modulus n is a k-bit integer. Wiener [52] showed that if the private key d of bitlength less than $0.25k$ is selected, then an adversary can efficiently recover d using only his knowledge of the public key (n, e). In 1998, Boneh and Durfee [11] improved Wiener's result to show that d can also be efficiently recovered if its bitlength is less than $0.292k$. It is believed [11] that these results can be extended to the situation where the bitlength of d is less than $0.5k$, but this has not yet been accomplished.

It is strongly recommended that the decryption exponent d not be chosen to have any special properties, e.g., small bitsize.

4 Security of the RSA-OAEP Encryption Scheme

The Optimal Asymmetric Encryption Padding (OAEP) scheme for RSA was proposed by Bellare and Rogaway in 1994 [6]. RSA-OAEP is intended to be both *efficient* and *provably secure*. RSA-OAEP is designed to encrypt only short messages—typically secret keys for symmetric encryption or MAC algorithms.

4.1 Description

The following notation is used in the description of the encryption scheme:

- A's RSA public key is (n,e), and d is A's corresponding private key. The integer n is k octets in length. (For example, if n is a 1024-bit modulus, then $k = 128$.)
- H is a hash function which has outputs that are l octets in length. (For example, H may be SHA-1, in which case $l = 20$.)
- G is the following *mask generating function*: if the inputs to G are an octet string s and a positive integer $t < 2^{32}l$, then the output of G is an octet string of length t obtained by concatenating successive hash values $H(s \| i)$, $0 \leq i \leq \lceil t/l \rceil - 1$, and deleting any extra rightmost octets if necessary (i is a 32-bit counter).
- P consists of some (optional) encoding parameters.
- A padding string consists of a string of 00 octets (possible empty) followed by the octet 01.

RSA-OAEP Encryption Scheme. (B sends message m of length at most $k-2-2l$ octets to A)

1. **Encryption**. B does the following:

   1.1. Obtain an authentic copy of A's public key (n,e).
   1.2. Select a random seed s of length l octets.
   1.3. Apply the OAEP encoding operation (depicted in Figure 1) with inputs s, P and m to obtain an octet string M of length k octets.
   1.4. Compute $c = M^e \bmod n$.
   1.5. Send c to A.

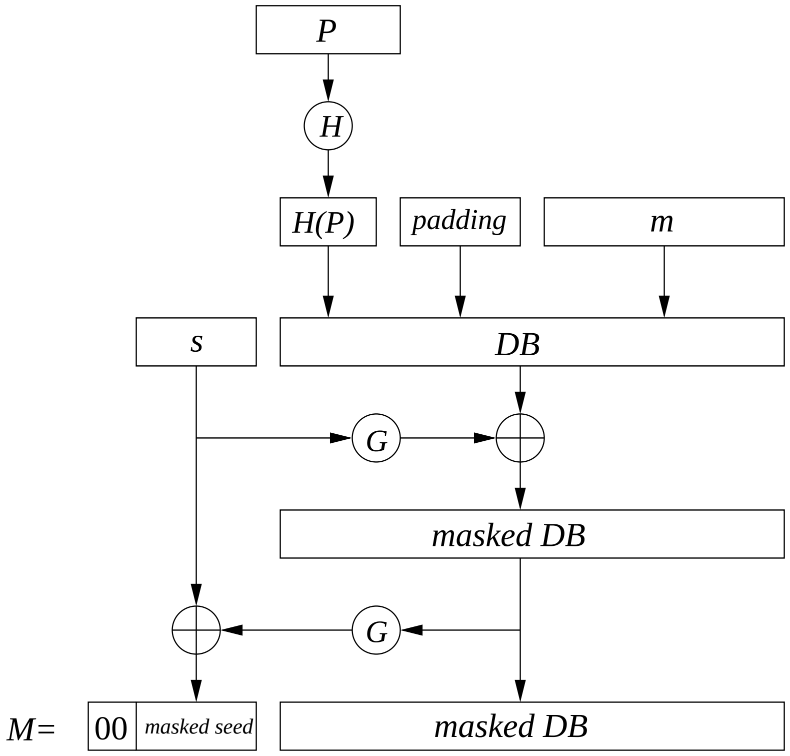

Figure 1: OAEP encoding function.

2. **Decryption**. A does the following:
   - 2.1. Check that $c \in [0, n-1]$; if not then output "error" and stop.
   - 2.2. Use the private key d to compute $M = c^d \bmod n$.
   - 2.3. Apply the OAEP decoding operation (depicted in Figure 2) with input M to obtain octet strings X, Q, T, M; here X has octet length 1, Q consists of the first l octets of DB, T is the first non-zero octet following Q, and m consists of the octets to the right of T.
   - 2.4. Compute $Q' = H(P)$.
   - 2.5. If $Q' \neq Q$, or if $X \neq 00$, or if $T \neq 01$, then output "error" and stop.
   - 2.6. Output M.

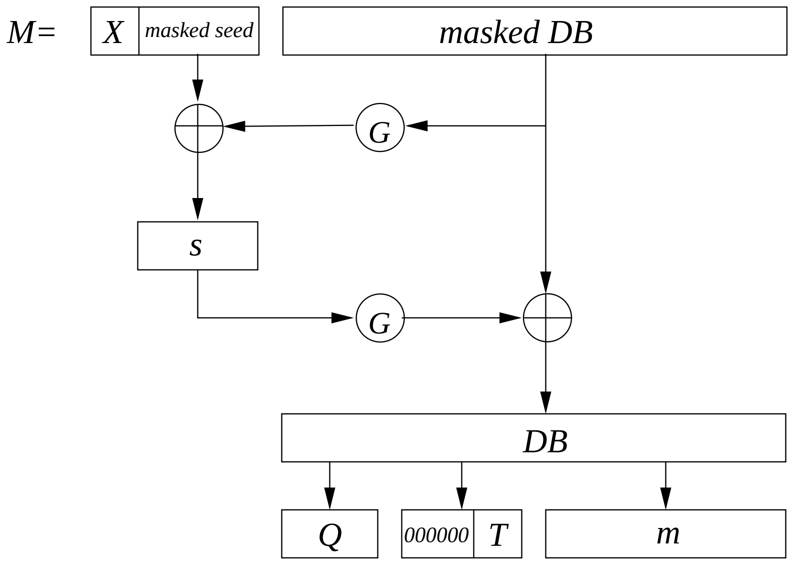

Figure 2: OAEP decoding function.

REMARKS.

1. Step 1.1 of the OAEP decoding operation (Section 3.2) of [47] is incomplete and should include the checks that $T = 01$ and $X = 00$.
2. The decoding operation in [47] should output "decoding error" if all the octets to the right of Q are zero (in which case T is not defined).

4.2 Security Proofs

BELLARE-ROGAWAY SECURITY PROOF. In their 1994 paper, Bellare and Rogaway [6] proved that RSA-OAEP is plaintext aware in the random oracle model under the assumption that the

RSA function $m \mapsto m^e \bmod n$ is a one-way function. (Roughly speaking, an encryption scheme is *plaintext aware* if it is infeasible for an adversary to construct a ciphertext c whose plaintext is valid—that is, was correctly formatted using the OAEP encoding function—without knowing m. The *random oracle model*, popularized by Bellare and Rogaway [5], is a computational model in which all entities, including the adversary, have access to a public random function. In the proof of security, the hash function H is modeled as a random function.) The assumption that the RSA function is one-way is called the *RSA assumption*. Bellare and Rogaway claimed that plaintext awareness in the random oracle model implied semantic security against adaptive chosen-ciphertext attacks. Thus, it was believed that RSA-OAEP provided the strongest notion of security for public-key encryption schemes.

FLAWS IN THE BELLARE-ROGAWAY SECURITY PROOF AND ITS CORRECTION. It came as a big surprise when Victor Shoup [50] announced in 2000 that the Bellare-Rogaway security proof was in fact incorrect. In [4], it is proven that plaintext awareness in the random oracle model implies IND-CCA. However, the Bellare-Rogaway security proof used a different notion of plaintext awareness which is not known (and not believed) to imply IND-CCA. In fact, Shoup provides a strong argument that OAEP when used with a general trapdoor permutation f *cannot* in general be proven secure in the random oracle model under the assumption that f is one-way. (However, Shoup did prove that RSA-OAEP is IND-CCA for the case $e = 3$.)

Shoup's work cast a serious doubt about the security of RSA-OAEP.

A short time after Shoup's paper was circulated, Fujisaki, Okamoto, Pointcheval and Stern (FOPS) [21] were able to provide a proof that RSA-OAEP is IND-CCA in the random oracle model under the RSA assumption. They did this by first proving security under the assumption that the RSA function is partial-domain one-way—given $c = m^e \bmod n$, it is infeasible to find the l most significant bits of m for some fixed l—and then noting that the partial-domain one-way RSA assumption is equivalent to the standard RSA assumption. The proof is elegant and correct. Thus, we now have a proof that RSA-OAEP is IND-CCA in the random oracle model under the RSA assumption.

However, the proof has two shortcomings which are discussed next.

RANDOM ORACLE MODEL. The first objection to the security proof of RSA-OAEP is that it takes place in the random oracle model, and therefore does not imply security in the real world where the hash function H is no longer a random function. This objection was given some credence by Canetti, Goldreich and Halevi [14] who provided examples of signature and encryption schemes which they proved secure in the random oracle model, but which are insecure with *any* reasonable instantiation of the random function. However, the encryption and signature schemes in [14] are rather contrived, and a security proof in the random oracle model is now widely accepted as providing valuable *heuristic* evidence for security. Additionally, the security proof of RSA-OAEP implies that any attack on RSA-OAEP must either invert the RSA function, or exploit some properties of the hash function H.

We conclude that the objection that the RSA-OAEP security proof is in the random oracle model is not a very serious one.

TIGHTNESS OF THE REDUCTION. The second objection which has do to with the tightness of the reduction in the FOPS proof [21] is a more serious one.

Security proofs typically work by providing a *reduction* from the solution of a problem P to the breaking of a cryptographic scheme S. That is, one shows how P can be solved if one is given an oracle for breaking S. In the case of the RSA-OAEP proof, S is RSA-OAEP, while P is the RSA problem. The success of the reduction is measured by the tightness of the reduction. More precisely, a reduction is *tight* if when an adversary can break S in time t with probability ε, then the reduction algorithm that solves P runs in time t' with success probability ε', where $t' \approx t$ and $\varepsilon' \approx \varepsilon$. A tight reduction is important because one then has the assurance that the security level of S is very closely related to the security level of P. Thus one can select security parameters in S to be of the same size as the parameters that make P intractable against all known attacks. For example, if the reduction in a security proof of RSA-OAEP is tight, then one can use 1024-bit moduli n and be assured that breaking RSA-OAEP will require roughly the same amount of work as it takes to solve the RSA problem for 1024-bit moduli.

The original security proof of Bellare and Rogaway did provide a tight reduction. However, as noted above, this proof is incorrect.

Unfortunately, the reduction in the FOPS security proof is *not* tight. The success probability ε' is related to ε by $\varepsilon' \approx \varepsilon/(2 \cdot Q_1)$, while the time t' is repeated to t by $t' \approx t + Q_1 \cdot Q_2$ (see Theorem 1 of [21] for a more precise statement). Here Q_1 denotes the number of queries made to the mask generating function G (modeled as a random oracle in the proof) which masks the string "masked DB" (see Figure 1) by a hypothetical adversary that breaks RSA-OAEP in time t with success probability ε. Similarly, Q_2 denotes the number of queries made to the mask generating function G which masks the seed s. Since Q_1 and Q_2 can in principle be large, e.g., $Q_1 \approx 2^{40}$, the reduction is not tight.

Therefore, in order to be assured from the FOPS proof that RSA-OAEP provides a certain acceptable level of security, one has to use an RSA modulus that is far bigger than one would actually use in practice. Thus, it is not at all clear what assurances, if any, are provided by the security proof for RSA-OAEP.

Shoup [50] proposed a small change to RSA-OAEP which he called RSA-OAEP+. The change is that the padding string in the OAEP encoding function is replaced by the string $H'(m \| s)$, where H' is a hash function (see Figure 1). Shoup provides a security proof for RSA-OAEP+ where the reduction is tighter than the FOPS reduction. However, Shoup notes [50, page 24]:

> ....the reductions for OAEP+ are not tight enough to actually imply that an algorithm that breaks, say, 1024-bit RSA-OAEP+ in a "reasonable" amount of time implies an algorithm that solves the RSA problem in time faster than the best known factoring algorithms.

The self-evaluation report submitted to the CRYPTREC project [48] states (page 6):

> The main conclusion is that earlier claims about the security need to be revised; the security reduction is not as tight as was earlier claimed. In particular, the reduction does not apply to typical parameters such as 1024-bit key size. Still, the reduction does provide a heuristic argument that the OAEP construction is sound when combined with RSA and that a successful attack against RSA-OAEP in the random oracle model relates to the problem of inverting the RSA primitive, if not as efficiently as was earlier believed. Shoup's variant RSA-OAEP+ has a stronger proof, but the proof is still not applicable in practice, which means that we would need to rely on heuristic arguments anyway.

So, it appears that the real-world security assurances provided by the RSA-OAEP proof of security are not well understood.

4.3 Resistance to Manger's Attack

At the Crypto 2001 conference, Manger [36] described an ingenious attack that would be successful on some implementations[1] of RSA-OAEP.

Manger's attack is similar in spirit to Bleichenbacher's Attack on PKCS#1 v1.5 RSA encryption that was outlined in Section 3.2.2. Given a target ciphertext c, an adversary modifies c to obtain ciphertext c' and provides c' to a decrytion oracle. Now, c' may not be a valid ciphertext, i.e., the corresponding plaintext M' may not be a valid output of the OAEP encoding function. For example, the first octet X of M' may not be 00, or the first non-zero octet T of the padding string may not be 01 (see Figure 2). Manger showed that if the decryption oracle returns different error messages for each of these two possible errors, then the adversary can recover the plaintext corresponding to c.

In response to this attack, the description of RSA-OAEP in [47] was modified so that the checks $X = 00$ and $T = 01$ are done in the same step as the check $Q' = H(P)$ (see step 2.5 of the RSA-OAEP decryption algorithm in Section 4.1). Moreover, the decryption algorithm returns the *same* error message if any of the checks fail. Note that these modifications do not change the mathematical description of RSA-OAEP—the modifications are only recommendations for implementors that operations be performed in a certain order. The modifications also help prevent timing attacks which may yield information on which check failed during a decryption operation.

With these modifications, it appears that RSA-OAEP is resistant to Manger's attack.

[1]There are many implementation attacks that can be launched on cryptographic schemes. Examples include timing attacks (Kocher [31]), differential fault analysis attacks (Boneh, DeMillo and Lipton [10]), differential power analysis (Kocher, Jaffe and Jun [32]), and attacks which exploit weak random or pseudorandom number generators (Kelsey et al. [30]). These kinds of attacks are not considered in this report.

5 Security of the RSA-PKCS#1 v1.5 Signature Scheme

This section presents the RSA-PKCS1-v1.5 signature scheme as described in [45]. This scheme is a refinement of the basic RSA signature scheme (Section 3.3). It specifies a method for padding the hashed message in order to defeat known attacks on basic RSA such as the ones listed in Section 3.3.

5.1 Description

Let (n,e) be A's RSA public key, and let d be A's corresponding private key. The integer n is k octets in length. (For example, if n is a 1024-bit modulus, then $k = 128$.) H is a hash function such as SHA-1 that is identified by some octet string HID.

RSA-PKCS1-v1.5 Signature Scheme. (A signs a message m for B)

1. **Signature Generation**. A does the following:
   - 1.1. Compute $h = H(m)$.
   - 1.2. Form the following octet string M of length k octets:

     $$00 \,\|\, 01 \,\|\, \mathrm{PS} \,\|\, \mathrm{HID} \,\|\, h,$$

     where PS denotes a string of FF octets.
   - 1.3. Use the private key d to compute $s = M^d \bmod n$.
   - 1.4. Send the signature s and the message m to B.
2. **Signature Verification**. B does the following:
   - 2.1. Obtain an authentic copy of A's public key (n,e).
   - 2.2. Compute $M' = s^e \bmod n$, and convert M' to an octet string of length k octets.
   - 2.3. Compute $h = H(m)$.
   - 2.4. Form the following octet string M of length k octets:

     $$00 \,\|\, 01 \,\|\, \mathrm{PS} \,\|\, \mathrm{HID} \,\|\, h,$$

     where PS denotes a string of FF octets.
   - 2.5. Compare M and M'. If $M = M'$ then accept the signature; otherwise reject the signature.

5.2 Security Proofs

There are no attacks known on the RSA-PKCS1-v1.5 signature scheme. In particular, the attack of Coron, Naccache and Stern [18] is ineffective on RSA-PKCS1-v1.5. However, there is also no known proof that RSA-PKCS1-v1.5 is existentially unforgeable against adaptive chosen-message attacks.

Since the RSA-PSS signature scheme is as efficient as RSA-PKCS1-v1.5, and has a tight security proof (see Section 6), there is no reason to use RSA-PKCS1-v1.5 instead of RSA-PSS. Thus, it is recommended that all standards for RSA signatures adopt RSA-PSS instead of RSA-PKCS1-v1.5.

6 Security of the RSA-PSS Signature Scheme

The Probabilistic Signature Scheme (PSS) variant of RSA was proposed by Bellare and Rogaway in 1996 [7]. RSA-PSS is intended to be both *efficient* and *provably secure*. The RSA-PSS scheme described here from [46] is significantly different from the original scheme proposed by Bellare and Rogaway.

6.1 Description

The following notation is used in the description of RSA-PSS:

- A's RSA public key is (n,e), and d is A's corresponding private key. The integer n is k octets in length, and y bits in length (For example, if n is a 1028-bit modulus, then $k = 129$ and $y = 1028$.)
- $x = \lceil (y-1)/8 \rceil$.
- $z = 8x - (y-1)$.
- H is a hash function which has outputs that are l octets in length. (For example, H may be SHA-1, in which case $l = 20$.)
- G is the following *mask generating function*: if the inputs to G are an octet string s and a positive integer $t < 2^{32}l$, then the output of G is an octet string of length t obtained by concatenating successive hash values $H(s\|i)$, $0 \le i \le \lceil t/l \rceil - 1$, and deleting any extra rightmost octets if necessary (i is a 32-bit counter).
- TF (trailer field) either consists of a single octet BC, or the concatenation of two octets HID and CC, where HID is the single octet identifier of the hash function H as specified in the ISO/IEC 10118 standard.

- *salt* is a randomly generated octet string.

RSA-PSS Signature Scheme. (*A* signs a message for *B*.)

1. **Signature Generation**. *A* does the following:

   1.1. Apply the PSS encoding function (depicted in Figure 3) to m to obtain the octet string EM. The string EM has length x octets, and its integer representation has bitlength at most $y-1$. The z leftmost bits of EM are 0.

   1.2. Use the private key d to compute $s = \mathrm{EM}^d \bmod n$.

   1.3. Send the signature s and the message m to B.

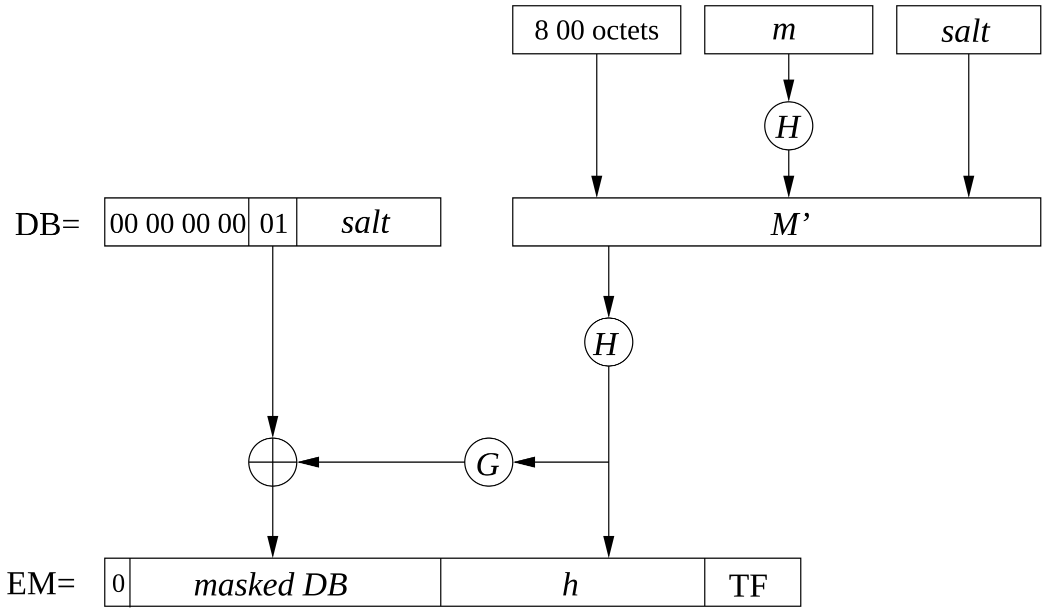

Figure 3: PSS encoding function.

2. **Signature Verification**. *B* does the following:

   2.1. Obtain an authentic copy of *A*'s public key (n, e).

   2.2. Check that s is an integer in the interval $[0, n-1]$; if not then output "error" and stop.

   2.3. Compute $\mathrm{EM} = s^e \bmod n$.

   2.4. If the octet length of EM is greater than x, then output "error" and stop.

   2.5. If the z most significant bits of EM are not 0, then output "error" and stop.

   2.6. Apply the PSS decoding function (depicted in Figure 4). During this decoding process, check that octet string TF is either consists of a single octet BC, or is the concatenation of two octets HID and CC, where HID is the single octet identifier of the hash function H as specified in the ISO/IEC 10118 standard. Also, check that the string DB begins with a string of 00 octets of the appropriate length, followed by the octet 01.

2.7. If $h' = H(m)$, then accept the signature; otherwise reject.

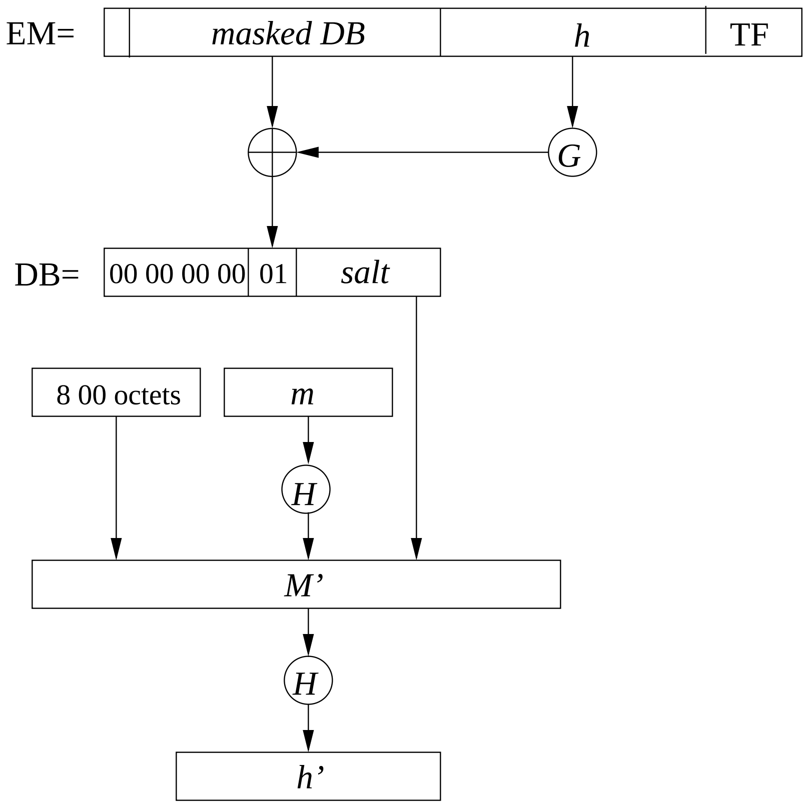

Figure 4: PSS decoding function.

SALT LENGTH. The specification of RSA-PSS [49] does not give any guidance on what the length of the salt should be. PKCS#1 v2.1 [46] states that typical lengths of salt are 0 and l, and these values are analyzed in [29]. However, we recommend that explicit and clear guidance on the salt length be provided in the RSA-PSS specification.

6.2 Security Proofs

Since the RSA-PSS specification differs significantly from Bellare and Rogaway's original scheme, it is important to provide a clear and complete proof of security of the modified scheme in order to be convinced that the security assurances are still present. This has been done in the accompanying paper by Jonsson [29].

Jonsson proves that RSA-PSS is existentially unforgeable against adaptive chosen-message attacks in the random oracle model under the assumption that the RSA problem is intractable.

TIGHTNESS OF THE REDUCTION. The reduction in Jonsson's proof of security is very tight (see Section 4.2 for background on tightness of a reduction). Thus we are assured that breaking RSA-PSS will require roughly the same amount of work as it takes to solve the RSA problem.

RANDOM ORACLE MODEL. As with the security proof of RSA-OAEP (Section 4.2), the security proof of RSA-PSS takes place in the random oracle model. Thus the security proof only provides *heuristic* evidence for security in the real world.

REMOVING THE RANDOM ORACLE ASSUMPTION. We remark that there are RSA-based signatures schemes that are provably secure in a standard model—one which does not make the unrealistic assumption that hash functions employed are random functions. For an example of such a scheme, see Cramer and Shoup [19] and Gennaro, Halevi and Rabin [22]. However, the security of these schemes is based on a very strong assumption called the *strong RSA assumption* (given an RSA modulus n and $c \in_R [0, n-1]$, it is difficult to find $t > 1$ and m such that $c \equiv m^t \pmod{n}$), so it does not appear that the security assurances by such schemes are any greater than the security assurance provided by RSA-PSS.

HASH FUNCTION IDENTIFIER. Inclusion of the hash function identifier HID in the PSS encoding operation is optional. Including the hash function identifier HID may be useful in preventing attacks by an adversary who may substitute the utilized hash function by a different hash function (e.g., a hash function that has recently been shown to be weak). However, since the mask generation function is defined using the same hash function as that used to hash the message, it does not seem possible that such an adversary can succeed.

7 Conclusions

We provide a concise overview of our analysis of the RSA-OAEP, RSA-PKCS1-v1.5 and RSA-PSS schemes.

1. LENGTH OF RSA MODULUS n. The modulus n should be at least 1024 bits in length. Recommendations should be provided on the moduli lengths for protecting keys of commonly used symmetric encryption schemes such as Triple DES (112 bits), AES-128, AES-192 and AES-256.

2. USE OF SMALL ENCRYPTION EXPONENT e. Using an encryption exponent $e = 3$ is risky, as there is some evidence that solving the RSA problem for $e = 3$ may be easier than factoring n. We recommend that the encryption exponent e be at least $2^{16} + 1$.

3. RSA-OAEP ENCRYPTION SCHEME. RSA-OAEP has been proven to be semantically secure against adaptive chosen-ciphertext attacks in the random oracle model under the

RSA assumption. However, the reduction is not tight, and thus it is not clear what security assurances the proof provides. We recommend that RSA-OAEP be modified to RSA-OAEP+ which has a tighter security reduction, and furthermore can be easily modified to allow encryption of arbitrarily-long messages (see [50]). Furthermore, the RSA-KEM encryption scheme of Shoup [51] which has a tight reduction should be considered as a replacement for RSA-OAEP.

4. RSA-PKCS1-V1.5 SIGNATURE SCHEME. There is no formal proof of security of RSA-PKCS1-v1.5. Since RSA-PSS is as efficient as RSA-PKCS1-v1.5 and has a security proof, there is no reason to use RSA-PKCS1-v1.5.

5. RSA-PSS SIGNATURE SCHEME. RSA-PSS has been proven to be existentially unforgeable against adaptive chosen-message attacks in the random oracle model under the RSA assumption. The reduction is tight, and so the assurances provided by the security proof are strong. More guidance should be given on the desired length of the salt used in the RSA-PSS encoding operation.

References

[1] ANSI X9.31, *Digital Signatures Using Reversible Public Key Cryptography for the Financial Services Industry (rDSA)*, 1998.

[2] ANSI X9.44, *Key Establishment Using Factoring-Based Public Key Cryptography for the Financial Services Industry*, working draft, 2000.

[3] M. Bellare, A. Boldyreva and S. Micali, "Public-key encryption in a multi-user setting: security proofs and improvements", *Advances in Cryptology–Eurocrypt 2000*, Lecture Notes in Computer Science, **1807** (2000), Springer-Verlag, 259-274.

[4] M. Bellare, A. Desai, D. Pointcheval and P. Rogaway, "Relations among notions of security for public-key encryption schemes", *Advances in Cryptology–Crypto '98*, Lecture Notes in Computer Science, **1462** (1998), Springer-Verlag, 26-45.

[5] M. Bellare and P. Rogaway, "Random oracles are practical: a paradigm for designing efficient protocols", *1st ACM Conference on Computer and Communications Security*, 1993, 62-73. Full version available at http://www.cs.ucdavis.edu/~rogaway/papers/index.html

[6] M. Bellare and P. Rogaway, "Optimal asymmetric encryption", *Advances in Cryptology–Eurocrypt '94*, Lecture Notes in Computer Science, **950** (1995), Springer-Verlag, 92-111. Full version available at http://www.cs.ucdavis.edu/~rogaway/papers/index.html

[7] M. Bellare and P. Rogaway, "The exact security of digital signatures—How to sign with RSA and Rabin", *Advances in Cryptology–Eurocrypt '96*, Lecture Notes in Computer Science, **1070** (1996), Springer-Verlag, 399-416. Full version available at http://www.cs.ucdavis.edu/~rogaway/papers/index.html

[8] D. Bleichenbacher, "Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS#1", *Advances in Cryptology–Crypto 1998*, Lecture Notes in Computer Science, **1462** (1998), Springer-Verlag, 1-12.

[9] D. Boneh, "Twenty years of attacks on the RSA cryptosystem", *Notices of the American Mathematical Society*, **46** (1999), 203-213.

[10] D. Boneh, R. DeMillo and R. Lipton, "On the importance of checking cryptographic protocols for faults", *Advances in Cryptology–Eurocrypt '97*, Lecture Notes in Computer Science, **1233** (1997), Springer-Verlag, 37-51.

[11] D. Boneh and G. Durfee, "Cryptanalysis of RSA with private key d less than $N^{0.292}$", *IEEE Transactions on Information Theory*, **46** (2000), 1339-1349.

[12] D. Boneh, A. Joux and P. Nguyen, “Why textbook ElGamal and RSA encryption are insecure”, *Advances in Cryptology–Asiacrypt 2000*, Lecture Notes in Computer Science, **1976** (2000), Springer-Verlag, 30-43.

[13] D. Boneh and R. Venkatesan, “Breaking RSA mey not be equivalent to factoring”, *Advances in Cryptology–Eurocrypt ’98*, Lecture Notes in Computer Science, **1403** (1998), Springer-Verlag, 59-71.

[14] R. Canetti, O. Goldreich and S. Halevi, “The random oracle methodology, revisited”, *Proceedings of the 30th Annual ACM Symposium on the Theory of Computing*, 1998, 209-218.

[15] S. Cavallar et al., “Factorization of a 512-bit RSA modulus”, *Advances in Cryptology–Eurocrypt 2000*, Lecture Notes in Computer Science, **1807** (2000), Springer-Verlag, 1-18.

[16] D. Coppersmith, “Small solutions to polynomial equations, and low exponent RSA vulnerabilities”, *Journal of Cryptology*, **10** (1997), 233-260.

[17] D. Coppersmith, M. Franklin, J. Patarin and M. Reiter, “Low-exponent RSA with related messages”, *Advances in Cryptology–Eurocrypt ’96*, Lecture Notes in Computer Science, **1070** (1996), Springer-Verlag, 1-9.

[18] J. Coron, D. Naccache and J. Stern, “On the security of RSA padding”, *Advances in Cryptology–Crypto ’99*, Lecture Notes in Computer Science, **1666** (1999), Springer-Verlag, 1-18.

[19] R. Cramer and V. Shoup, “Signature schemes based on the strong RSA assumption”, *ACM Transactions on Information and System Security*, **3** (2000), 161-185.

[20] Y. Desmedt and A. Odlyzko, “A chosen text attack on the RSA cryptosystem and some discrete logarithm schemes”, *Advances in Cryptology–Crypto ’85*, Lecture Notes in Computer Science, **218** (1986), Springer-Verlag, 516-522.

[21] E. Fujisaki, T. Okamoto, D. Pointcheval and J. Stern, “RSA-OAEP is secure under the RSA assumption”, *Advances in Cryptology–Crypto 2001*, Lecture Notes in Computer Science, **2139** (2001), Springer-Verlag, 260-274.

[22] R. Gennaro, S. Halevi and T. Rabin, “Secure hash-and-sign signatures without the random oracle”, *Advances in Cryptology–Eurocrypt ’99*, Lecture Notes in Computer Science, **1592** (1999), Springer-Verlag, 123-139.

[23] S. Goldwasser and S. Micali, “Probabilistic encryption”, *Journal of Computer and System Sciences*, **29** (1984), 270-299.

[24] S. Goldwasser, S. Micali and R. Rivest, “A digital signature scheme secure against adaptive chosen-message attacks”, *SIAM J. Computing*, **17** (1988), 281-308.

[25] J. Håstad, “Solving simultaneous modular equations of low degree”, *SIAM Journal on Computing*, **17** (1988), 336-341.

[26] IEEE 1363-2000, *Standard Specifications for Public-Key Cryptography*, 2000. http://grouper.ieee.org/groups/1363/index.html

[27] ISO/IEC 9796-1, *Information Technology–Security Techniques-Digital Signature Scheme Giving Message Recovery, Part 1: Mechanisms Using Redundancy*, 1999.

[28] ISO/IEC 9796-2, *Information Technology–Security Techniques-Digital Signature Scheme Giving Message Recovery, Part 1: Mechanisms Using a Hash Function*, 1997.

[29] J. Jonsson, “Security proofs for the RSA-PSS signature scheme and its variants”, Draft 1.1, June 27 2000.

[30] J. Kelsey, B. Schneier, D. Wagner and C. Hall, “Cryptanalytic attacks on pseudorandom number generators”, *Fast Software Encryption–FSE ’98*, Lecture Notes in Computer Science, **1372** (1998), Springer-Verlag, 168-188.

[31] P. Kocher, “Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems”, *Advances in Cryptology–Crypto ’96*, Lecture Notes in Computer Science, **1109** (1996), Springer-Verlag, 104-113.

[32] P. Kocher, J. Jaffe and B. Jun, “Differential power analysis”, *Advances in Cryptology–Crypto ’99*, Lecture Notes in Computer Science, **1666** (1999), Springer-Verlag, 388-397.

[33] A. Lenstra, “Unbelievable security: Matching AES security using public key systems”, *Advances in Cryptology–Asiacrypt 2001*, Lecture Notes in Computer Science, **2248** (2001), Springer-Verlag, 67-86

[34] A. Lenstra and E. Verheul, “Selecting cryptographic key sizes”, *Public Key Cryptography–Proceedings of PKC 2000*, Lecture Notes in Computer Science, **1751** (2000), Springer-Verlag, 446-465. Full version to appear in *Journal of Cryptology*.

[35] H. Lenstra, “Factoring integers with elliptic curves”, *Annals of Mathematics*, **126** (1987), 649-673.

[36] J. Manger, “A chosen ciphertext attack on RSA optimal asymmetric encryption padding (OAEP) as standardized in PKCS#1 v2.0”, *Advances in Cryptology–Crypto 2001*, Lecture Notes in Computer Science, **2139** (2001), Springer-Verlag, 230-238.

[37] J. Misarsky, “How (not) to design RSA signature schemes”, *Public Key Cryptography–Proceedings of PKC ’98*, Lecture Notes in Computer Science, **1431** (1998), Springer-Verlag, 14-28.

[38] National Institute of Standards and Technology, *Digital Signature Standard*, FIPS Publication 186-2, 2000.

[39] National Institute of Standards and Technology, *Advanced Encryption Standard (AES)* FIPS Publication 197, 2001.

[40] M. Rabin, ‘Digitalized signatures and public-key functions as intractable as factorization”, MIT Lab. for Computer Science, Technical Report LCS/TR-212, 1979

[41] C. Rackoff and D. Simon, “Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack”, *Advances in Cryptology–Crypto ’91*, Lecture Notes in Computer Science, **576** (1992), 433-444.

[42] R. Rivest, A. Shamir and L. Adleman, “A method for obtaining digital signatures and public-key cryptosystems”, *Communications of the ACM*, **21** (1978), 120-126.

[43] RSA Data Security, *RSA Factoring Challenge*, http://www.rsasecurity.com/rsalabs/challenges/factoring/index.html

[44] RSA Laboratories, *PKCS#1 v1.5: RSA Encryption Standard*, November 1993.

[45] RSA Laboratories, *PKCS#1 v2.0: RSA Cryptography Standard*, September 1998. Available at http://www.rsa.com/rsalabs/pkcs/pkcs-1/

[46] RSA Laboratories, *PKCS#1 v2.1: RSA Cryptography Standard* (Draft 2), January 5 2001. Available at http://www.rsa.com/rsalabs/pkcs/pkcs-1/

[47] RSA Laboratories, *The RSA-OAEP Encryption Scheme*, Cryptographic Technique Specification (01espec), September 26 2001.

[48] RSA Laboratories, *The RSA-OAEP Encryption Scheme*, Self Evaluation Report (01eeval), September 26 2001.

[49] RSA Laboratories, *RSA-PSS Signature Scheme With Appendix*, Cryptographic Technique Specification (call-3e), September 25 2001.

[50] V. Shoup, “OAEP reconsidered”, *Advances in Cryptology–Crypto 2001*, Lecture Notes in Computer Science, **2139** (2001), Springer-Verlag, 239-259. Full version available at http://shoup.net/papers/

[51] V. Shoup, “A proposal for an ISO standard for public key encryption (version 2.0)”, September 17 2001. Available at http://shoup.net/papers/

[52] M. Wiener, “Cryptanalysis of short RSA secret exponents”, *IEEE Transactions on Information Theory*, **36** (1990), 553-558.

[53] H. Williams, “A modification of the RSA public-key encryption procedure”, *IEEE Transactions on Information Theory*, **26** (1980), 726-729.