

# PKCS #1 Encoding Method for IF Signatures

Burt Kaliski<sup>1</sup> and Yiqun Lisa Yin<sup>2</sup>  
RSA Laboratories

March 12, 1999

*Submission to IEEE P1363a*

**Abstract.** An additional encoding method for RSA signatures is proposed for inclusion in the IEEE P1363a document. Derived from the signature format in PKCS #1 v1.5, the encoding method has the advantages that it is relatively widely deployed and that it is more flexible in terms of hash function support compared to EMSA2, the method presently in IEEE P1363. The proposed encoding method shares many of the same security attributes as EMSA2, and is equally straightforward to implement.

## 1. Introduction

As defined in IEEE P1363 [1], a signature scheme with appendix combines signature and verification primitives with an *encoding method for signatures with appendix*, or EMSA. An EMSA consists of an encoding operation and a verification operation. The encoding operation encodes a message to produce a non-negative integer called a *message representative*. The verification operation verifies whether a message representative is a valid encoding of a message. In a signature scheme, a message representative is the input to the signature primitive; it may either be an input to or an output from the verification primitive. The encoding method connects the message representative to the actual message being signed.

IEEE P1363 defines two encoding methods for signatures with appendix, one for signature schemes in the discrete logarithm (DL) and elliptic curve (EC) families, and a second for signature schemes in the integer factorization (IF) family. This note proposes an additional encoding method for the IF family, EMSA-PKCS1-v1\_5, based on the signature format in PKCS #1 v1.5 [2] which is also supported in PKCS #1 v2.0 [3][4].

---

<sup>1</sup> RSA Laboratories, 20 Crosby Drive, Bedford, MA 01730 USA. E-mail: burt@rsa.com.

<sup>2</sup> RSA Laboratories, 2955 Campus Drive, Suite 400, San Mateo, CA 94403-2507 USA. E-mail: yiqun@rsa.com.

## 2. Description of the cryptographic technique

EMSA-PKCS1-v1\_5 is an encoding method for signatures with appendix based on a hash function with some additional formatting, as described in PKCS #1. It is recommended for use with IFSSA (Section 10.3)<sup>3</sup> for the RSA key type.

The function is parameterized by the following choice:

- a hash function *Hash*, which shall be SHA-1 (Section 14.1.1), or RIPEMD-160<sup>4</sup> (Section 14.1.2), or a technique designated for use with EMSA-PKCS1-v1\_5 in an addendum to the standard

NOTE—EMSA-PKCS1-v1\_5 cannot produce message representatives shorter than a certain length that depends on the hash function.

### 2.1 Encoding operation

#### Input:

- a message, which is an octet string *M* (depending on the hash function chosen, there may be a limitation on the length of *M*; for SHA-1 and RIPEMD-160, the maximum length is  $2^{61} - 1$  octets)
- the maximum bit length *l* of the output

**Output:** a message representative, which is an integer  $f \geq 0$  of bit length at most *l*; or “error”

The message representative *f* shall be computed by the following or an equivalent sequence of steps:

1. If the length of *M* is greater than the length limitation for the hash function ( $2^{61} - 1$  octets for SHA-1 or RIPEMD-160), output “error” and stop.
2. Compute *Hash* (*M*) with the selected hash function to produce an octet string *H*.
3. Let *D* be the DER encoding of an ASN.1 value of type `DigestInfo`:

---

<sup>3</sup> All Section references are to IEEE P1363 draft D8 (available from <http://grouper.ieee.org/groups/1363>), which are the same as in draft D9, the ballot version. (The differences between the two drafts are only in pagination.)

<sup>4</sup> Although the RIPEMD-160 hash function is not mentioned in PKCS #1 v2.0, the format is easily extended to support RIPEMD-160, as is done here.

```
DigestInfo ::= SEQUENCE {
    digestAlgorithm AlgorithmIdentifier,
    digest OCTET STRING }
```

where the `digestAlgorithm` field identifies the hash function *Hash*, and the `digest` field contains the octet string *H*. If  $\|D\| > \lfloor l / 8 \rfloor - 10$ , where  $\|D\|$  is the length of *D* in octets, output “error” and stop.

4. Let *P* be an octet string of length  $\lfloor l / 8 \rfloor - \|D\| - 2$  octets, each with hexadecimal value `ff`.
5. Let  $T = 01 \parallel P \parallel 00 \parallel D$ , where 01 and 00 are single octets represented in hexadecimal.
6. Convert *T* to an integer *f* using OS2IP.
7. Output *f* as the message representative.

## 2.2 Verification operation

### Input:

- a message, which is an octet string *M* (depending on the hash function chosen, there may be a limitation on the length of *M*; for SHA-1 and RIPEMD-160, the maximum length is  $2^{61} - 1$  octets)
- the maximum bit length *l* of the message representative
- the message representative, which is an integer  $f \geq 0$

**Output:** “valid” if *f* is a correct representative of *M*; “invalid”

The validity indicator shall be computed by the following or an equivalent sequence of steps:

1. If the length of *M* is greater than the length limitation ( $2^{61} - 1$  octets for SHA-1 or RIPEMD-160), output “invalid” and stop.
2. Compute *Hash* (*M*) with the selected hash function to produce an octet string *H*.
3. Let *D* be the DER encoding of an ASN.1 value of type `DigestInfo`:

```
DigestInfo ::= SEQUENCE {
    digestAlgorithm AlgorithmIdentifier,
    digest OCTET STRING }
```

where the `digestAlgorithm` field identifies the hash function *Hash*, and the `digest` field contains the octet string *H*. If  $\|D\| > \lfloor l / 8 \rfloor - 10$ , where  $\|D\|$  is the length of *D* in octets, output “error” and stop.

4. Let *P* be an octet string of length  $\lfloor l / 8 \rfloor - \|D\| - 2$  octets, each with hexadecimal value ff.
5. Let  $T = 01 \parallel P \parallel 00 \parallel D$ , where 01 and 00 are single octets represented in hexadecimal.
6. Convert *T* to an integer  $f'$  using OS2IP.
7. If  $f = f'$  output “valid.” Otherwise, output “invalid.”

### 3. Claimed attributes and advantages

The proposed method has the following advantages compared to EMSA2, the current encoding method for signatures with appendix in the IF family:

- **Wide deployment.** EMSA-PKCS1-v1\_5 is the encoding method for RSA signatures in many industry standards that have wide and emerging deployment, including SSL (and its successor, TLS), S/MIME, and PKIX. There is a large installed base of implementations in industry. EMSA2, although included in recent standards such as ANSI X9.31 (for which NIST has announced its support) and ISO/IEC 14888, does not yet have a substantial base of implementations. Thus, the proposed method has the advantage of reflecting current industry practice.
- **More flexibility in hash function support.** EMSA-PKCS1-v1\_5 identifies the hash function with an ASN.1 object identifier, so it is easy for developers to extend the method to support other hash functions than SHA-1 or RIPEMD-160, since any entity can register an object identifier. In contrast, EMSA2 identifies the underlying hash function by a single octet, subject to a special registration process.

The proposed method is comparable to EMSA2 in terms of implementation. Although EMSA-PKCS1-v1\_5 is defined in terms of ASN.1 syntax, ASN.1 syntax processing is not required. In particular, the proposed method can be implemented as a fixed format for each hash function. For instance, for SHA-1, the octet string *T* has the format

$$T = 01 \parallel P \parallel 00 \parallel D ,$$

where *P* is an octet string of length  $l-37$  octets each with hexadecimal value ff, the octet string *D* has the format

$$D = 30 \ 21 \ 30 \ 1f \ 06 \ 05 \ 2b \ 0e \ 03 \ 02 \ 1a \ 05 \ 00 \ 04 \ 14 \parallel H,$$

(corresponding to the DER encoding of the `DigestInfo` value) and  $H$  is the 20-octet hash value. The proposed encoding method is thus as straightforward to implement as EMSA2.

The proposed method can support signatures with either RSA and RW keys, including a one-bit savings in signature size.

For RSA without a one-bit savings, the RSA1 signature and verification primitives may be applied directly.

For RSA with a one-bit savings, the RSA2 signature primitive may be applied together with a modification of the RSA2 verification primitive. A message representative, extended to the modulus length, will have the form

$$00\ 01\ ||\ P\ ||\ 00\ ||\ D.$$

The most significant two octets of the negation of a message representative will be at least `00 fe`. The modified RSA2 verification operation distinguishes between the two based on the two most significant octets.

For RW, modifications of both the RW signature and verification primitives may be applied. Conversion to an element with Jacobi symbol 1 during the modified RW signature operation is done with doubling rather than halving. This produces a result whose two most significant octets are `00 03`. The modified RW verification operation again distinguishes doubles and negations from a message representative based on the two most significant octets.

#### 4. Security assessment and considerations

As stated in Section D.5.2.2.1, an encoding method for a signature scheme with appendix should be one-way and collision resistant, and it should have minimal mathematical structure that could interact with the selected signature primitive. In addition, it is desirable that the encoding method may identify the underlying hash function. The proposed method EMSA-PKCS1-v1\_5 is considered to have all these properties for IFSSA (Section 10.3).

The security of EMSA-PKCS1-v1\_5 depends on properties of the underlying hash function. (See Section D.5.2.2.1 for related discussions.) The inclusion of the hash function algorithm ID in the message representative serves as a *hash function firewall*. In particular, this can prevent an adversary from making it appear as though the signature was produced with a different hash function than the one the signer selected. This may also provide some protection against repudiation by the signer.

As observed by Rivest [5], a message representative constructed from the hash function output alone, or together with an algorithm ID, may not be enough to thwart certain variants of the small-message attack by Desmedt and Odlyzko [6]. A message representative so constructed is a relatively small integer (perhaps as short as 160 bits)

and thus has a reasonable chance of being useful for the small-message attack (that is, it is smooth). The fixed padding  $01 \parallel P \parallel 00$  in EMSA-PKCS1-v1\_5 prevents such types of attack by producing large message representatives. Moreover, the overall structure of the block also prevents various multiplicative attacks on IF signature schemes, such as “blinding” attacks where an opponent requests a signature on one message in an attempt to obtain a signature on a different message.

Even though EMSA-PKCS1-v1\_5 is not known to have “provable security” which relates the difficulty of forging signatures to some underlying hard problem, a recent result by Baudron and Stern [7] provides some evidence on the provable security of the method. They show that when the message representative is constructed from a truly random function whose output is at least half of the length of the modulus, instead of a hash function, existential forgery of an RSA signature is equivalent to the problem of inverting RSA. So this result gives some confidence in the security of the encoding method EMSA-PKCS1-v1\_5, since it produces an output whose length is close to the length of the modulus (although obviously it isn’t random).

For further discussions on the security of RSA signatures, see [8][9].

## 5. Known limitations and disadvantages

The proposed method, like EMSA2, is not known to enjoy any “provable security” properties relating the difficulty of forging signatures constructed with the encoding method to the difficulty of some underlying hard problem such as integer factorization. If provable security is desired, an encoding method such as PSS [10][11] may be preferable.

## 6. Intellectual property issues

RSA Laboratories makes no patent claims on the EMSA-PKCS1-v1\_5 method described in Section 2.

RSA Laboratories makes no representations regarding intellectual property claims by other parties.

## 7. References

- [1] IEEE. *IEEE P1363: Standard Specifications for Public Key Cryptography*. Draft Version 8, October 5, 1998.
- [2] RSA Laboratories. *PKCS #1: RSA Encryption Standard*. Version 1.5, November 1993.
- [3] RSA Laboratories. *PKCS #1: RSA Cryptography Standard*. Version 2.0, October 1, 1998.

- [4] B. KALISKI and J. STADDON. *PKCS #1: RSA Cryptography Specifications Version 2.0*. IETF RFC 2437, October 1998.
- [5] R. Rivest. Personal communication, 1991.
- [6] Y. Desmedt and A. Odlyzko. *A chosen text attack on the RSA cryptosystem and some discrete logarithm schemes*. In *Advances in Cryptology – Crypto '85*, pages 516–522, Springer-Verlag, 1986.
- [7] O. Baudron and J. Stern. *To pad or not to pad: Does formatting degrade security?* Presented at the 1999 RSA Data Security Conference, January 1999.
- [8] D. Boneh. *Twenty Years of Attacks on the RSA Cryptosystem*. In *Notices of the American Mathematical Society (AMS)*, vol. 46, no. 2, pages 203–213, 1999.
- [9] B. Kaliski and M. Robshaw. *The Secure Use of RSA*. In *CryptoBytes*, vol. 1, no. 3, pages 7–13, 1995.
- [10] M. Bellare and P. Rogaway. *The Exact Security of Digital Signatures-How to Sign with RSA and Rabin*. In *Advances in Cryptology - Eurocrypt '96*, pages 399–416, Springer-Verlag, 1996.
- [11] M. Bellare and P. Rogaway. *PSS: Provably Secure Encoding Method for Digital Signatures*. IEEE P1363a submission, August 1998.