PKCS #1 v2.0 Amendment 1: Multi-Prime RSA

RSA Laboratories

DRAFT 1 — May 20, 2000

Editor's note: This is the first draft of amendment 1 to PKCS #1 v2.0, which is available for a 30-day public review period. Please send comments and suggestions, both technical and editorial, to pkcs-editor@rsasecurity.com or pkcs-tng@rsasecurity.com.

Table of Contents

1. Introduction

This document amends PKCS #1 v2.0 [3] to support so-called "multi-prime" RSA where the modulus may have more than two prime factors. Only private-key operations and representations are affected. The encryption and signature schemes and the public-key operations and representation for multi-prime RSA are the same as in PKCS #1 v.2.0.

The benefit of multi-prime RSA is lower computational cost for the decryption and signature primitives, provided that the CRT (Chinese Remainder Theorem) is used. Better performance can be achieved on single processor platforms, but to a greater extent on multiprocessor platforms, where the modular exponentiations involved can be done in parallel.

The reader is referred to [5] for a discussion on how multi-prime affects the security of the RSA cryptosystem.

This amendment is written as revisions to PKCS #1 v2.0. Only the affected sections are included.

2. Changes to Section 2, "Notation"

[Update the notation as follows:]

n	modulus, $n = r_1 \cdot r_2 \cdot \ldots \cdot r_k$, $k \geq 2$
p, q	first two prime factors of the modulus
LCM (. , … , .)	least common multiple of a list of nonnegative integers
$\lambda(n)$	LCM ($r_1 - 1$, $r_2 - 1$, …, $r_k - 1$)

[Add the following new notation:]

d_i — additional factor r_i's exponent, a positive integer such that:

$$e \cdot d_i \equiv 1 \pmod{(r_i - 1)},\ i = 3, \ldots, k$$

k — number of prime factors of the modulus, $k \geq 2$

r_i — prime factors of the modulus, including $r_1 = p$, $r_2 = q$, and additional factors if any

t_i — additional factor r_i's CRT coefficient, a positive integer less than r_i such that

$$r_1 \cdot r_2 \cdot \ldots \cdot r_{(i-1)} \cdot t_i \equiv 1 \pmod{r_i},\ i = 3, \ldots, k$$

Note. The CRT can be applied in a non-recursive as well as a recursive way. In this document we use a recursive approach that follows Garner's algorithm [1]. See also the note in Section 3.2.

3. Changes to Section 3, "Key types"

3.1 Changes to Section 3.1, "RSA public key"

[Replace the second paragraph of this section with the following:]

In a *valid RSA public key*, the modulus n is a product of k distinct odd primes r_i, $i = 1, 2, \ldots, k$, where $k \geq 2$ and the public exponent e is an integer between 3 and $n-1$ satisfying GCD $(e, \lambda(n)) = 1$, where $\lambda(n) = \text{LCM}\ (r_1 - 1, \ldots, r_k - 1)$. By convention, the first two primes r_1 and r_2 may also be denoted p and q respectively.

3.2 Changes to Section 3.2, "RSA private key"

[Replace the second item in the list with the following:]

2. The second representation consists of a quintuple (p, q, dP, dQ, $qInv$) and a (possibly empty) sequence of triplets (r_i, d_i, t_i), $i = 3, \ldots, k$, one for each prime not in the quintuple, where the components have the following meanings:

— p, the first factor, a nonnegative integer

— q, the second factor, a nonnegative integer

— dP, the first factor's exponent, a nonnegative integer

— dQ, the second factor's exponent, a nonnegative integer

— $qInv$, the (first) CRT coefficient, a nonnegative integer

— r_i, the i^{th} factor, a nonnegative integer

— d_i, the i^{th} factor's exponent, a nonnegative integer

— t_i, the i^{th} factor's CRT coefficient, a nonnegative integer

[Replace the second paragraph after the list with the following:]

In a valid RSA private key with the second representation, the two factors p and q are the *first two* prime factors of the modulus n (i.e., r_1 and r_2), the exponents dP and dQ are positive integers less than p and q respectively satisfying

$$e \cdot dP \equiv 1 \pmod{(p-1)}$$
$$e \cdot dQ \equiv 1 \pmod{(q-1)},$$

and the CRT coefficient $qInv$ is a positive integer less than p satisfying

$$q \cdot qInv \equiv 1 \pmod{p}.$$

If $k > 2$, the representation will include one or more triplets (r_i, d_i, t_i), $i = 3, \ldots, k$. The factors r_i, are the additional prime factors of the modulus n. Each exponent d_i satisfies

$$e \cdot d_i \equiv 1 \pmod{(r_i - 1)}, i = 3, \ldots, k.$$

Each CRT coefficient t_i, $i = 3, \ldots, k$, is a positive integer less than r_i satisfying

$$R_i \cdot t_i \equiv 1 \bmod r_i,$$

where $R_i = r_1 \cdot r_2 \cdot \ldots \cdot r_{(i-1)}$.

Note. The definition of the CRT coefficients here and the formulas that use them in the primitives in Section 5 generally follows Garner's algorithm [1] (see also Algorithm 14.71 in [2]). However, for compatibility with the representations of RSA private keys in PKCS #1 v2.0 and previous versions, the roles of p and q are reversed compared to the rest of the primes, so the first CRT coefficient, *qInv*, is defined as the inverse of q mod p, rather than as the inverse of R_1 mod r_2, i.e., of p mod q. The benefit of applying the Chinese Remainder Theorem to RSA operations was observed by Quisquater and Couvreur [3].

4. Changes to Section 5, "Cryptographic primitives"

4.1 Changes to Section 5.1.2, "RSADP"

[Replace this decryption primitive with the following:]

RSADP (K, c)

Input: K RSA private key, where K has one of the following forms:

- — a pair (n, d)
- — a quintuple (p, q, dP, dQ, $qInv$) and a (possibly empty) sequence of triplets (r_i, d_i, t_i), $i = 3, \ldots, k$

c ciphertext representative, an integer between 0 and n–1

Output: m message representative, an integer between 0 and n–1

Errors: "ciphertext representative out of range"

Assumptions: private key K is valid

Steps:

1. If the ciphertext representative c is not between 0 and n–1, output "ciphertext representative out of range" and stop.

2. If the first form (n, d) of K is used:

   2.1 Let $m = c^d \bmod n$.

   Else, if the second form (p, q, dP, dQ, $qInv$) and (r_i, d_i, t_i) of K is used:

   2.2 Let $m_1 = c^{dP} \bmod p$.

   2.3 Let $m_2 = c^{dQ} \bmod q$.

2.4 If $k > 2$, then let $m_i = c^{d_i} \bmod r_i$, $i = 3, \ldots, k$.

2.5 Let $h = (m_1 - m_2) \cdot qInv \bmod p$.

2.6 Let $m = m_2 + q \cdot h$.

2.7 If $k > 2$, then let $R = r_1$ and for $i = 3$ to k do

2.7.1 Let $R = R \cdot r_{(i-1)}$.

2.7.2 Let $h = (m_i - m) \cdot t_i \pmod{r_i}$.

2.7.3 Let $m = m + R \cdot h$.

3. Output m.

Note. Steps 2.2–2.7 can be rewritten as a single loop, provided that one reverses the order of p and q. For consistency with PKCS #1 v2.0, however, the first two primes p and q are treated separately from the additional primes.

4.2 Changes to Section 5.2.1, "RSASP1"

[Replace this signature primitive with the following:]

RSASP1 (K, m)

Input: K RSA private key, where K has one of the following forms:

— a pair (n, d)

— a quintuple $(p, q, dP, dQ, qInv)$ and a (possibly empty) sequence of triplets (r_i, d_i, t_i), $i = 3, \ldots, k$

m message representative, an integer between 0 and n–1

Output: s signature representative, an integer between 0 and n–1

Errors: "message representative out of range"

Assumptions: private key K is valid

Steps:

1. If the message representative m is not between 0 and n–1, output "message representative out of range" and stop.

2. If the first form (n, d) of K is used:

2.1 Let $s = m^d \bmod n$.

Else, if the second form (p, q, dP, dQ, $qInv$) and (r_i, d_i, t_i) of K is used:

2.2 Let $m_1 = c^{dP} \bmod p$.

2.3 Let $m_2 = c^{dQ} \bmod q$.

2.4 If $k > 2$, then let $s_i = m^{d_i} \bmod r_i$, $i = 3, \ldots, k$.

2.5 Let $h = (s_1 - s_2) \cdot qInv \bmod p$.

2.6 Let $s = s_2 + q \cdot h$.

2.7 If $k > 2$, then let $R = r_1$ and for $i = 3$ to k do

2.7.1 Let $R = R \cdot r_{(i-1)}$.

2.7.2 Let $h = (s_i - s) \cdot t_i \pmod{r_i}$.

2.7.3 Let $s = s + R \cdot h$.

3. Output s.

5. Changes to Section 11, "ASN.1"

5.1 Changes to Section 11.1.2, "Private-key syntax"

[Replace this section with:]

An RSA private key should be represented with ASN.1 type `RSAPrivateKey`:

```
RSAPrivateKey ::= SEQUENCE {
  version Version,
  modulus INTEGER, -- n
  publicExponent INTEGER, -- e
  privateExponent INTEGER, -- d
  prime1 INTEGER, -- p
  prime2 INTEGER, -- q
  exponent1 INTEGER, -- d mod (p-1)
  exponent2 INTEGER, -- d mod (q-1)
  coefficient INTEGER -- (inverse of q) mod p
  otherPrimeInfos OtherPrimeInfos OPTIONAL }

Version ::= INTEGER

OtherPrimeInfos ::= SEQUENCE OF OtherPrimeInfo
```

```
OtherPrimeInfo ::= SEQUENCE {
  prime INTEGER,  -- ri
  exponent INTEGER, -- di
  coefficient INTEGER -- ti }
```

The fields of type `RSAPrivateKey` have the following meanings:

- `version` is the version number, for compatibility with future revisions of this document. It shall be 0 if there are only two prime factors and 1 for this version of the document if there are three or more prime factors.
- `modulus` is the modulus n.
- `publicExponent` is the public exponent e.
- `privateExponent` is the private exponent d.
- `prime1` is the prime factor p of n.
- `prime2` is the prime factor q of n.
- `exponent1` is d mod $(p–1)$.
- `exponent2` is d mod $(q–1)$.
- `coefficient` is the Chinese Remainder Theorem coefficient q^{-1} mod p.
- `otherPrimeInfos` contains the information for the additional primes r_3, …, r_k, in order. It shall be omitted if `version` is 0 and shall contain at least one instance of `OtherPrimeInfo` if `version` is 1.

The fields of type `OtherPrimeInfo` have the following meanings:

- `prime` is a prime factor r_i of n, where $i \geq 3$.
- `exponent` is $d_i = d \bmod (r_i - 1)$.
- `coefficient` is the Chinese Remainder Theorem coefficient $t_i = (r_1 \cdot r_2 \cdot \ldots \cdot r_{(i-1)})^{-1} \bmod r_i$.

Note. It is important to protect the private key against both disclosure and modification. Techniques for such protection are outside the scope of this document. Method for protecting for private keys and other cryptographic data are described in PKCS #12 and #15.

6. Changes to Section 13, "References"

[Merge in these new references and renumber:]

[1] H. Garner. The residue number system. *IRE Transactions on Electronic Computers*, EC-8 (6), pp. 140-147, June 1959.

[2] A. Menezes, P. van Oorschot and S. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996.

[3] J.-J. Quisquater and C. Couvreur. Fast decipherment algorithm for RSA public-key cryptosystem. *Electronics Letters*, 18(21), pp. 905–907, October 14, 1982.

[4] RSA Laboratories. *PKCS #1 v2.0: RSA Cryptography Standard*. October 1998.

[5] Robert D. Silverman. *A Cost-Based Security Analysis of Symmetric and Asymmetric Key Lengths*. RSA Laboratories Bulletin No. 13, April 2000. Available at `http://www.rsasecurity.com.rsalabs/bulletins/`.

A. Intellectual property considerations

The RSA public-key cryptosystem is protected by U.S. Patent 4,405,829. RSA Security Inc. makes no other patent claims on the constructions described in this document, although specific underlying techniques may be covered.

Multi-prime RSA is claimed in U.S. Patent 5,848,159.

License to copy this document is granted provided that it is identified as “RSA Security Inc. Public-Key Cryptography Standards (PKCS)” in all material mentioning or referencing this document.

RSA Security Inc. makes no other representations regarding intellectual property claims by other parties. Such determination is the responsibility of the user.

B. About PKCS

The *Public-Key Cryptography Standards* are specifications produced by RSA Laboratories in cooperation with secure systems developers worldwide for the purpose of accelerating the deployment of public-key cryptography. First published in 1991 as a result of meetings with a small group of early adopters of public-key technology, the PKCS documents have become widely referenced and implemented. Contributions from the PKCS series have become part of many formal and *de facto* standards, including ANSI X9 documents, PKIX, SET, S/MIME, and SSL.

Further development of PKCS occurs through mailing list discussions and occasional workshops, and suggestions for improvement are welcome. For more information, contact:

PKCS Editor
RSA Laboratories
20 Crosby Drive
Bedford, MA 01730 USA
pkcs-editor@rsasecurity.com
http://www.rsasecurity.com/rsalabs/pkcs