COSE Working Group
Internet-Draft
Intended status: Standards Track
Expires: October 6, 2016

M. Jones
Microsoft

April 4, 2016

Using RSA Algorithms with COSE Messages
draft-jones-cose-rsa-00

Abstract

The CBOR Object Signing and Encryption (COSE) specification defines cryptographic message encodings using Concise Binary Object Representation (CBOR). This specification defines algorithm encodings and representations enabling RSA algorithms to be used for COSE messages.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on October 6, 2016.

Copyright Notice

Table of Contents

1. Introduction

The CBOR Object Signing and Encryption (COSE) [I-D.ietf-cose-msg] specification defines cryptographic message encodings using Concise Binary Object Representation (CBOR) [RFC7049]. This specification defines algorithm encodings and representations enabling RSA algorithms to be used for COSE messages.

1.1. Requirements Notation and Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

2. Signature Algorithms

2.1. RSASSA-PSS

The RSASSA-PSS signature algorithm is defined in [RFC3447].

The RSASSA-PSS signature algorithm is parameterized with a hash function (h), a mask generation function (mgf) and a salt length (sLen). For this specification, the mask generation function is

fixed to be MGF1 as defined in [RFC3447]. It has been recommended that the same hash function be used for hashing the data as well as in the mask generation function, for this specification we following this recommendation. The salt length is the same length as the hash function output.

Implementations need to check that the key type is 'RSA' when creating or verifying a signature.

The algorithms defined in this document can be found in Table 1.

name	value	hash	salt length	description
PS256	-26	SHA-256	32	RSASSA-PSS w/ SHA-256
PS384	-27	SHA-384	48	RSASSA-PSS w/ SHA-384
PS512	-28	SHA-512	64	RSASSA-PSS w/ SHA-512

Table 1: RSASSA-PSS Algorithm Values

2.1.1. Security Considerations

In addition to needing to worry about keys that are too small to provide the required security, there are issues with keys that are too large. Denial of service attacks have been mounted with overly large keys. This has the potential to consume resources with potentially bad keys. There are two reasonable ways to address this attack. First, a key should not be used for a cryptographic operation until it has been matched back to an authorized user. This approach means that no cryptography would be done except for authorized users. Second, applications can impose maximum as well as minimum length requirements on keys. This limits the resources consumed even if the matching is not performed until the cryptography has been done.

There is a theoretical hash substitution attack that can be mounted against RSASSA-PSS. However, the requirement that the same hash function be used consistently for all operations is an effective mitigation against it. Unlike ECDSA, hash functions are not truncated so that the full hash value is always signed. The internal padding structure of RSASSA-PSS means that one needs to have multiple collisions between the two hash functions in order to be successful in producing a forgery based on changing the hash function. This is highly unlikely.

3. Recipient Algorithm Classes

3.1. Key Encryption

Key Encryption mode is also called key transport mode in some standards. Key Encryption mode differs from Key Wrap mode in that it uses an asymmetric encryption algorithm rather than a symmetric encryption algorithm to protect the key. This document defines one Key Encryption mode algorithm.

When using a key encryption algorithm, the COSE_encrypt structure for the recipient is organized as follows:

- The 'protected' field MUST be absent.
- The plain text to be encrypted is the key from next layer down (usually the content layer).
- At a minimum, the 'unprotected' field MUST contain the 'alg' parameter and SHOULD contain a parameter identifying the asymmetric key.

3.1.1. RSAES-OAEP

RSAES-OAEP is an asymmetric key encryption algorithm. The definition of RSAEA-OAEP can be find in Section 7.1 of [RFC3447]. The algorithm is parameterized using a masking generation function (mgf), a hash function (h) and encoding parameters (P). For the algorithm identifiers defined in this section:

- o mgf is always set to MFG1 from [RFC3447] and uses the same hash function as h.

- o P is always set to the empty octet string.

Table 2 summarizes the rest of the values.

name	value	hash	description
RSAES-OAEP w/SHA-256	-25	SHA-256	RSAES OAEP w/ SHA-256
RSAES-OAEP w/SHA-512	-26	SHA-512	RSAES OAEP w/ SHA-512

Table 2: RSAES-OAEP Algorithm Values

The key type MUST be 'RSA'.

3.1.1.1. Security Considerations for RSAES-OAEP

A key size of 2048 bits or larger MUST be used with these algorithms. This key size corresponds roughly to the same strength as provided by a 128-bit symmetric encryption algorithm.

It is highly recommended that checks on the key length be done before starting a decryption operation. One potential denial of service operation is to provide encrypted objects using either abnormally long or oddly sized RSA modulus values. Implementations SHOULD be able to encrypt and decrypt with modulus between 2048 and 16K bits in length. Applications can impose additional restrictions on the length of the modulus.

4. Keys

Key types are identified by the 'kty' member of the COSE_Key object. In this document we define one value for the member.

name	value	description

RSA	3	RSA Keys

Table 3: Key Type Values

4.1. RSA Keys

This document defines a key structure for both the public and private halves of RSA keys. Together, an RSA public key and an RSA private key form an RSA key pair. [[CREF1: Looking at the CBOR specification, the bstr that we are looking in our table below should most likely be specified as big numbers rather than as binary strings. This means that we would use the tag 6.2 instead. From my reading of the specification, there is no difference in the encoded size of the resulting output. The specification of bignum does explicitly allow for integers encoded with leading zeros. --JLS]]

The document also provides support for the so-called "multi-prime" RSA where the modulus may have more than two prime factors. The benefit of multi-prime RSA is lower computational cost for the decryption and signature primitives. For a discussion on how multi-prime affects the security of RSA crypto-systems, the reader is referred to [MultiPrimeRSA].

This document follows the naming convention of [RFC3447] for the naming of the fields of an RSA public or private key. The table

Table 4 provides a summary of the label values and the types associated with each of those labels. The requirements for fields for RSA keys are as follows:

- For all keys, 'kty' MUST be present and MUST have a value of 3.
- For public keys, the fields 'n' and 'e' MUST be present. All other fields defined in Table 4 MUST be absent.
- For private keys with two primes, the fields 'other', 'r_i', 'd_i' and 't_i' MUST be absent, all other fields MUST be present.
- For private keys with more than two primes, all fields MUST be present. For the third to nth primes, each of the primes is represented as a map containing the fields 'r_i', 'd_i' and 't_i'.

The field 'other' is an array of those maps.

name	key type	value	type	description
n	3	-1	bstr	Modulus Parameter
e	3	-2	int	Exponent Parameter
d	3	-3	bstr	Private Exponent Parameter
p	3	-4	bstr	First Prime Factor
q	3	-5	bstr	Second Prime Factor
dP	3	-6	bstr	First Factor CRT Exponent
dQ	3	-7	bstr	Second Factor CRT Exponent
qInv	3	-8	bstr	First CRT Coefficient
other	3	-9	array	Other Primes Info
r_i	3	-10	bstr	i-th factor, Prime Factor
d_i	3	-11	bstr	i-th factor, Factor CRT Exponent
t_i	3	-12	bstr	i-th factor, Factor CRT Coefficient

Table 4: RSA Key Parameters

5. IANA Considerations

5.1. COSE Algorithm Registry

This section registers values in the IANA "COSE Algorithm Registry" registry.

The values in Table 1 are to be added to the registry.

5.2. COSE Key Type Parameter Registry

This section registers values in the IANA "COSE Key Type Parameters" registry.

The values in Table 4 are to be added to the registry.

6. Security Considerations

See the per-algorithm security considerations described in Section 2.1.1 and Section 3.1.1.1.

7. References

7.1. Normative References

[I-D.ietf-cose-msg]
Schaad, J., "CBOR Encoded Message Syntax", draft-ietf-cose-msg-11 (work in progress), March 2016.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC3447] Jonsson, J. and B. Kaliski, "Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1", RFC 3447, DOI 10.17487/RFC3447, February 2003, <http://www.rfc-editor.org/info/rfc3447>.

[RFC7049] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", RFC 7049, DOI 10.17487/RFC7049, October 2013, <http://www.rfc-editor.org/info/rfc7049>.

7.2. Informative References

[MultiPrimeRSA]
Hinek, M. and D. Cheriton, "On the Security of Multi-prime RSA", June 2006.

Appendix A. Acknowledgements

The initial version of this specification incorporates text from draft-ietf-cose-msg-05 by Jim Schaad.

Appendix B. Document History

[[to be removed by the RFC Editor before publication as an RFC]]

-00

- This specification addresses COSE issue #21: Restore RSA-PSS and the "RSA" key type. The initial version of this specification incorporates text from draft-ietf-cose-msg-05 -- the last COSE message specification version before the RSA algorithms were removed.

Author's Address

Michael B. Jones
Microsoft

Email: mbj@microsoft.com
URI: http://self-issued.info/