Network Working Group
Internet Draft
Intended Status: Standard Track
Obsoletes: RFC 5208 (once approved)
Expires: 30 April 2009

Sean Turner, IECA
30 October 2008

Asymmetric Key Packages
draft-turner-asymmetrickeyformat-01.txt

Status of this Memo

Copyright Notice

Abstract

This document defines the syntax for private key information and a content type for it. Private-key information includes a private key for some public-key algorithm and a set of attributes. The document also describes a syntax for encrypted private keys. The Cryptographic Message Syntax, as defined in RFC 3852, can be used to

digitally sign, digest, authenticate, or encrypt the asymmetric key format content type. This document obsoletes RFC 5208.

Table of Contents

1. Introduction

This document defines the syntax for private key information and a content type for it. Private-key information includes a private key for some public-key algorithm and a set of attributes. The document also describes a syntax for encrypted private keys. The Cryptographic Message Syntax [RFC3852] can be used to digitally sign, digest, authenticate, or encrypt the asymmetric key format content type. This document obsoletes PKCS#8 v1.2 [RFC5208].

1.1. Requirements Terminology

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

1.2. ASN.1 Syntax Notation

The key package is defined using ASN.1 [X.680], [X.681], [X.682], and [X.683].

1.3. Changes since RFC 5208

The following are the changes since [RFC5208]:

- Defined Asymmetric Key Package CMS content type.

- Removed IMPLICIT from aKeyAttrs to align text with module.

- Added public key to OneAsymmetricKey and added new version number.

- Added that PKCS#9 attributes MAY be supported.

- Added Other Considerations section.

2. Asymmetric Key Package CMS Content Type

The asymmetric key package CMS content type is used to transfer one or more plaintext asymmetric keys from one party to another. An asymmetric key package MAY be encapsulated in one or more CMS protecting content types (see Section 4). This content type MUST be DER encoded [X.690].

The asymmetric key package content type has the following syntax:

```
PKCS7-CONTENT-TYPE ::= TYPE-IDENTIFIER

asymmetric-key-package PKCS7-CONTENT-TYPE ::=
  { AsymmetricKeyPackage IDENTIFIED BY id-ct-KP-aKeyPackage }

id-ct-KP-aKeyPackage OBJECT IDENTIFIER ::= |
  { TBD }

AsymmetricKeyPackage ::= SEQUENCE SIZE (1..MAX) OF OneAsymmetricKey

OneAsymmetricKey ::= SEQUENCE {
  version              Version,
  privateKeyAlgorithm  PrivateKeyAlgorithmIdentifier,
  privateKey           PrivateKey, -- DER encoded
  attributes       [0] Attributes OPTIONAL,
  publicKey        [1] PublicKey OPTIONAL }

PrivateKeyInfo ::= OneAsymmetricKey -- Used in [P12]

Version ::= INTEGER { v1(0), v2(1) } (v1, v2,...)

PrivateKeyAlgorithmIdentifier ::= AlgorithmIdentifier
                                  { { PrivateKeyAlgorithms } }

PrivateKey ::= OCTET STRING
                   -- Content varies based on type of key. The
                   -- algorithm identifier dictates the format of
                   -- the key. DSA's is an INTEGER ECDSA's is an
                   -- INTEGER, and RSA is as per [RFC3447].
```

```
PublicKey ::= OCTET STRING
                  -- Content varies based on type of key. The
                  -- algorithm identifier dictates the format of
                  -- the key. DSA is an INTEGER, ECDSA is an OCTET
                  -- STRING, and RSA is a sequence of two INTEGERs
                  -- [PKI-ALG].

Attributes ::= Set of Attribute
```

The AsymmetricKeyPackage contains one or more OneAsymmetricKey elements. The syntax of OneAsymmetricKey accommodates a version number, an indication of the algorithm to be used with the private key, a private key, and optionally keying material attributes (e.g., certificates) and a public key. In general, either the public key or the certificate will be present. In very rare cases will both the public key and the certificate be present as this includes two copies of the public key. The fields in OneAsymmetricKey are used as follows:

- version identifies version of the asymmetric key package content structure. For this version of the specification, version MUST be v1 if the publicKey field is absent and it MUST be set to v2 if the publicKey field is present.

- privateKeyAlgorithm identifies the private key algorithm and optionally contains parameters associated with the asymmetric key. The algorithm is identified by an OID and the parameters format depends on the OID. The value placed in privateKeyAlgorithmIdentifier is the value an originator would apply to indicate which algorithm was used.

- privateKey is an OCTET STRING whose contents is the DER encoded private key. The interpretation of the contents is defined in the registration of the private-key algorithm.

- attributes is optional. It contains information corresponding to the public key (e.g., certificates). The attributes field uses the class ATTRIBUTE which is restricted by the SupportedAttributes parameterized type. SupportedAttributes is an open ended set in this document. Others documents can constrain these values. Attributes from [RFC2985] MAY be supported.

- publicKey is optional. When present, it contains the public key encoded as an OCTET STRING. The structure within the octet string, if any, depends on the privateKeyAlgorithm.

3. Encrypted Private Key Info

This section gives the syntax for encrypted private-key information, which is used with [P12].

Encrypted private-key information shall have ASN.1 type EncryptedPrivateKeyInfo:

```
EncryptedPrivateKeyInfo ::= SEQUENCE {
  encryptionAlgorithm  EncryptionAlgorithmIdentifier,
  encryptedData        EncryptedData }

EncryptionAlgorithmIdentifier ::= AlgorithmIdentifier
                                  { { KeyEncryptionAlgorithms } }

EncryptedData ::= OCTET STRING
```

The EncAsymmetricKeyPackage contains one or more EncryptedPrivateKeyInfo elements. The fields in EncryptedPrivateKeyInfo are used as follows:

- encryptionAlgorithm identifies the algorithm under which the private-key information is encrypted. Implementations MUST support the TBD algorithm.

- encryptedData is the result of encrypting the private-key information (i.e., the PrivateKeyInfo).

The encryption process involves the following two steps:

1. The private-key information is BER encoded, yielding an octet string.

2. The result of step 1 is encrypted with the secret key to give an octet string, the result of the encryption process.

4. Protecting the AsymmetricKeyPackage

CMS [RFC3852] and [RFC5083] protecting content types can be used to provide security to the AsymmetricKeyPackage:

- SignedData can be used to apply a digital signature to the AsymmetricKeyPackage.

- EncryptedData can be used to encrypt the AsymmetricKeyPackage to provide confidentiality but does not distribute the content encryption keys.

- EnvelopedData can be used to encrypt the AsymmetricKeyPackage with simple symmetric encryption, where the sender and the receiver already share the necessary encryption key.

- AuthenticatedData can be used to protect the AsymmetricKeyPackage with message authentication codes, where key management information is handled in a manner similar to EnvelopedData.

- AuthEnvelopedData can be used to protect the AsymmetricKeypackage with algorithms that support authenticated encryption, where key management information is handled in a manner similar to EnvelopedData.

5. Other Considerations

This document defines the syntax and the semantics for content types that exchange asymmetric keys. There are two other standards for transporting asymmetric private keys:

- Personal Information Exchange (PFX) or more commonly referred to as P12 [P12], is a transfer syntax for personal identity information, including private keys, certificates, miscellaneous secrets, and extensions. Both PrivateKeyInfo and EncryptedPrivateKeyInfo can be carried in a P12 message.

- Microsoft's Exchange Security format, which is a proprietary format.

When locally storing private keys, the file format is either a DER encoded file with the file extension .p12 or a PEM encoded file with the file extension .pem.

When the private key is a character string, the OCTET STRING contains an embedded UTF8String.

6. Security Considerations

Protection of the private-key information is vital to public-key cryptography. Disclosure of the private-key material to another entity can lead to masquerades. The encryption algorithm used in the encryption process must be as 'strong' as the key it is protecting.

The asymmetric key package contents are not protected. This content type can be combined with a security protocol to protect the contents of the package.

The encrypted asymmetric key package contents are protected; as noted above the encryption algorithm must be as 'strong' as the key it is protecting.

7. IANA Considerations

None: All identifiers are already registered. Please remove this section prior to publication as an RFC.

8. References

8.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC3852] Housley, R., "Cryptographic Message Syntax (CMS)", RFC3852, July 2004.

[X.680] ITU-T Recommendation X.680 (2002) | ISO/IEC 8824-1:2002. Information Technology - Abstract Syntax Notation One.

[X.681] ITU-T Recommendation X.681 (2002) | ISO/IEC 8824-2:2002. Information Technology - Abstract Syntax Notation One: Information Object Specification.

[X.682] ITU-T Recommendation X.682 (2002) | ISO/IEC 8824-3:2002. Information Technology - Abstract Syntax Notation One: Constraint Specification.

[X.683] ITU-T Recommendation X.683 (2002) | ISO/IEC 8824-4:2002. Information Technology - Abstract Syntax Notation One: Parameterization of ASN.1 Specifications.

[X.690] ITU-T Recommendation X.690 (2002) | ISO/IEC 8825-1:2002. Information Technology - ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER).

8.2. Non-Normative References

[P12] RSA Laboratories, "PKCS #12 v1.0: Personal Information Exchange Syntax", June 1999.

[RFC2985] Nystrom, M., and B. Kaliski, "PKCS #9: Selected Object Classes and Attribute Types Version 2.0", RFC 2985, November 2000.

[RFC3447] Jonsson, J., and B. Kaliski, " Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1", RFC 3447, February 2003.

[RFC5208] Kaliski, B., "PKCS #8: Private Key Information Syntax Standard Version 1.2", RFC 5208, May 2008.

[RFC5083] Housley, R., "Cryptographic Message Syntax (CMS) Authenticated-Enveloped-Data Content Type", RFC 5083, November 2007.

[PKI-ALG] Turner, S., Brown, D., Yiu, K., Housley, R., and T. Polk, "Elliptic Curve Cryptography Subject Public Key Information", draft-ietf-pkix-ecc-subpubkeyinfo, work-in-progress.

APPENDIX A: ASN.1 Module

This annex provides the normative ASN.1 definitions for the structures described in this specification using ASN.1 as defined in [X.680] through [X.683].

```
AsymmetricKeyPackageModulev1  { tbd }

DEFINITIONS IMPLICIT TAGS ::=

BEGIN

-- EXPORTS ALL

IMPORTS NOTHING

Attribute{}, ATTRIBUTE, AlgorithmIdentifier{}
  FROM PKIX-CommonTypes
    { iso(1) identified-organization(3) dod(6) internet(1)
      security(5) mechanisms(5) pkix(7) id-mod(0)
      id-mod-pkixCommon(43) }

  id-aes128-wrap, id-aes192-wrap, id-aes1256-wrap
   FROM CMSAesRsaesOaep
     { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1)
       pkcs-9(9) smime(16) modules(0) id-mod-cms-aes(19) }

;

PKCS7-CONTENT-TYPE ::= TYPE-IDENTIFIER

KeyPackageContentTypes PKCS7-CONTENT-TYPE ::= {
  asymmetric-key-package |
  ... -- Expect additional content types --
}

asymmetric-key-package PKCS7-CONTENT-TYPE ::=
  { AsymmetricKeyPackage IDENTIFIED BY id-ct-KP-aKeyPackage }

id-ct-KP-aKeyPackage OBJECT IDENTIFIER ::=
  { TBD }

AsymmetricKeyPackage ::= SEQUENCE SIZE (1..MAX) OF OneAsymmetricKey
```

```
   OneAsymmetricKey ::= SEQUENCE {
     version               Version,
     privateKeyAlgorithm   PrivateKeyAlgorithmIdentifier,
     privateKey            PrivateKey, -- DER encoded
     attributes       [0] Attributes OPTIONAL,
     publicKey        [1] PublicKey OPTIONAL }

   PrivateKeyInfo ::= OneAsymmetricKey

   Version ::= INTEGER {v1(0), v2(1)} (v1, v2,...)

   PrivateKeyAlgorithmIdentifier ::= AlgorithmIdentifier
                                     { { PrivateKeyAlgorithms } }

   PrivateKey ::= OCTET STRING -- Content varies based on type of key
                               -- DSA is INTEGER, ECDSA is ECPublicKey

   PublicKey ::= OCTET STRING

   Attributes ::= Set of Attribute { { SupportAttributes } }

   SupportedAttributes ATTRIBUTE :: {
     ... -- For local profiles
   }

   EncryptedPrivateKeyInfo ::= SEQUENCE {
     encryptionAlgorithm  EncryptionAlgorithmIdentifier,
     encryptedData        EncryptedData }

   EncryptionAlgorithmIdentifier ::= AlgorithmIdentifier
                                     { { KeyEncryptionAlgorithms } }

   EncryptedData ::= OCTET STRING -- Encrypted PrivateKeyInfo

   PrivateKeyAlgorithms ALGORITHM-IDENTIFIER ::= {
     ... -- Extensible
   }

   KeyEncryptionAlgorithms ALGORITHM-IDENTIFIER ::= {
     id-aes128-wrap |
     id-aes192-wrap |
     id-aes256-wrap,
     ... -- Extensible
   }

   END
```

Acknowledgements

Many thanks go out to the Burt Kaliski and Jim Randall at RSA.
Without the prior version of the document, this one wouldn't exist.

We'd also like to thank Pasi Eronen and Russ Housley.

Author's Address

Sean Turner

IECA, Inc.
3057 Nutley Street, Suite 106
Fairfax, VA 22031
USA

Email: turners@ieca.com

Full Copyright Statement

Intellectual Property

Acknowledgment

Funding for the RFC Editor function is provided by the IETF Administrative Support Activity (IASA).