Network Working Group
Request for Comments: 3278
Category: Informational

S. Blake-Wilson
D. Brown
Certicom Corp
P. Lambert
Cosine Communications

April 2002

Use of Elliptic Curve Cryptography (ECC) Algorithms in Cryptographic Message Syntax (CMS)

Status of this Memo

This memo provides information for the Internet community. It does not specify an Internet standard of any kind. Distribution of this memo is unlimited.

Copyright Notice

Abstract

This document describes how to use Elliptic Curve Cryptography (ECC) public-key algorithms in the Cryptographic Message Syntax (CMS). The ECC algorithms support the creation of digital signatures and the exchange of keys to encrypt or authenticate content. The definition of the algorithm processing is based on the ANSI X9.62 standard, developed by the ANSI X9F1 working group, the IEEE 1363 standard, and the SEC 1 standard.

The readers attention is called to the Intellectual Property Rights section at the end of this document.

Table of Contents

1 Introduction

The Cryptographic Message Syntax (CMS) is cryptographic algorithm independent. This specification defines a profile for the use of Elliptic Curve Cryptography (ECC) public key algorithms in the CMS. The ECC algorithms are incorporated into the following CMS content types:

- 'SignedData' to support ECC-based digital signature methods (ECDSA) to sign content

- 'EnvelopedData' to support ECC-based public-key agreement methods (ECDH and ECMQV) to generate pairwise key-encryption keys to encrypt content-encryption keys used for content encryption

- 'AuthenticatedData' to support ECC-based public-key agreement methods (ECMQV) to generate pairwise key-encryption keys to encrypt MAC keys used for content authentication and integrity

Certification of EC public keys is also described to provide public-key distribution in support of the specified techniques.

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 BCP 14, RFC 2119 [MUST].

2 SignedData using ECC

This section describes how to use ECC algorithms with the CMS SignedData format to sign data.

2.1 SignedData using ECDSA

This section describes how to use the Elliptic Curve Digital Signature Algorithm (ECDSA) with SignedData. ECDSA is specified in [X9.62]. The method is the elliptic curve analog of the Digital Signature Algorithm (DSA) [FIPS 186-2].

In an implementation that uses ECDSA with CMS SignedData, the following techniques and formats MUST be used.

2.1.1 Fields of the SignedData

When using ECDSA with SignedData, the fields of SignerInfo are as in [CMS], but with the following restrictions:

digestAlgorithm MUST contain the algorithm identifier sha-1 (see Section 8.1) which identifies the SHA-1 hash algorithm.

signatureAlgorithm contains the algorithm identifier ecdsa-with-SHA1 (see Section 8.1) which identifies the ECDSA signature algorithm.

signature MUST contain the DER encoding (as an octet string) of a value of the ASN.1 type ECDSA-Sig-Value (see Section 8.2).

When using ECDSA, the SignedData certificates field MAY include the certificate(s) for the EC public key(s) used in the generation of the ECDSA signatures in SignedData. ECC certificates are discussed in Section 6.

2.1.2 Actions of the sending agent

When using ECDSA with SignedData, the sending agent uses the message digest calculation process and signature generation process for SignedData that are specified in [CMS]. To sign data, the sending agent uses the signature method specified in [X9.62, Section 5.3] with the following exceptions:

- In [X9.62, Section 5.3.1], the integer "e" is instead determined by converting the message digest generated according to [CMS, Section 5.4] to an integer using the data conversion method in [X9.62, Section 4.3.2].

The sending agent encodes the resulting signature using the ECDSA-Sig-Value syntax (see Section 8.2) and places it in the SignerInfo signature field.

2.1.3 Actions of the receiving agent

When using ECDSA with SignedData, the receiving agent uses the message digest calculation process and signature verification process for SignedData that are specified in [CMS]. To verify SignedData, the receiving agent uses the signature verification method specified in [X9.62, Section 5.4] with the following exceptions:

- In [X9.62, Section 5.4.1] the integer "e'" is instead determined by converting the message digest generated according to [CMS, Section 5.4] to an integer using the data conversion method in [X9.62, Section 4.3.2].

In order to verify the signature, the receiving agent retrieves the integers r and s from the SignerInfo signature field of the received message.

3 EnvelopedData using ECC Algorithms

This section describes how to use ECC algorithms with the CMS EnvelopedData format.

3.1 EnvelopedData using (ephemeral-static) ECDH

This section describes how to use the ephemeral-static Elliptic Curve Diffie-Hellman (ECDH) key agreement algorithm with EnvelopedData. Ephemeral-static ECDH is specified in [SEC1] and [IEEE1363]. Ephemeral-static ECDH is the the elliptic curve analog of the ephemeral-static Diffie-Hellman key agreement algorithm specified jointly in the documents [CMS, Section 12.3.1.1] and [CMS-DH].

In an implementation that uses ECDH with CMS EnvelopedData with key agreement, the following techniques and formats MUST be used.

3.1.1 Fields of KeyAgreeRecipientInfo

When using ephemeral-static ECDH with EnvelopedData, the fields of KeyAgreeRecipientInfo are as in [CMS], but with the following restrictions:

originator MUST be the alternative originatorKey. The originatorKey algorithm field MUST contain the id-ecPublicKey object identifier (see Section 8.1) with NULL parameters. The originatorKey publicKey field MUST contain the DER-encoding of a value of the ASN.1 type ECPoint (see Section 8.2), which represents the sending agent’s ephemeral EC public key.

keyEncryptionAlgorithm MUST contain the dhSinglePass-stdDH-sha1kdf-scheme object identifier (see Section 8.1) if standard ECDH primitive is used, or the dhSinglePass-cofactorDH-sha1kdf-scheme object identifier (see Section 8.1) if the cofactor ECDH primitive is used. The parameters field contains KeyWrapAlgorithm. The KeyWrapAlgorithm is the algorithm identifier that indicates the symmetric encryption algorithm used to encrypt the content-encryption key (CEK) with the key-encryption key (KEK).

3.1.2 Actions of the sending agent

When using ephemeral-static ECDH with EnvelopedData, the sending agent first obtains the recipient’s EC public key and domain parameters (e.g. from the recipient’s certificate). The sending agent then determines an integer "keydatalen", which is the KeyWrapAlgorithm symmetric key-size in bits, and also a bit string "SharedInfo", which is the DER encoding of ECC-CMS-SharedInfo (see Section 8.2). The sending agent then performs the key deployment and the key agreement operation of the Elliptic Curve Diffie-Hellman Scheme specified in [SEC1, Section 6.1]. As a result the sending agent obtains:

- an ephemeral public key, which is represented as a value of the type ECPoint (see Section 8.2), encapsulated in a bit string and placed in the KeyAgreeRecipientInfo originator field, and

- a shared secret bit string "K", which is used as the pairwise key-encryption key for that recipient, as specified in [CMS].

3.1.3 Actions of the receiving agent

When using ephemeral-static ECDH with EnvelopedData, the receiving agent determines the bit string "SharedInfo", which is the DER encoding of ECC-CMS-SharedInfo (see Section 8.2), and the integer "keydatalen" from the key-size, in bits, of the KeyWrapAlgorithm. The receiving agent retrieves the ephemeral EC public key from the bit string KeyAgreeRecipientInfo originator, with a value of the type ECPoint (see Section 8.2) encapsulated as a bit string. The receiving agent performs the key agreement operation of the Elliptic Curve Diffie-Hellman Scheme specified in [SEC1, Section 6.1]. As a result, the receiving agent obtains a shared secret bit string "K", which is used as the pairwise key-encryption key to unwrap the CEK.

3.2 EnvelopedData using 1-Pass ECMQV

This section describes how to use the 1-Pass elliptic curve MQV (ECMQV) key agreement algorithm with EnvelopedData. ECMQV is specified in [SEC1] and [IEEE1363]. Like the KEA algorithm [CMS-KEA], 1-Pass ECMQV uses three key pairs: an ephemeral key pair, a static key pair of the sending agent, and a static key pair of the receiving agent. An advantage of using 1-Pass ECMQV is that it can be used with both EnvelopedData and AuthenticatedData.

In an implementation that uses 1-Pass ECMQV with CMS EnvelopedData with key agreement, the following techniques and formats MUST be used.

3.2.1 Fields of KeyAgreeRecipientInfo

When using 1-Pass ECMQV with EnvelopedData, the fields of KeyAgreeRecipientInfo are:

originator identifies the static EC public key of the sender. It SHOULD be one of the alternatives, issuerAndSerialNumber or subjectKeyIdentifier, and point to one of the sending agent’s certificates.

ukm MUST be present. The ukm field MUST contain an octet string which is the DER encoding of the type MQVuserKeyingMaterial (see Section 8.2). The MQVuserKeyingMaterial ephemeralPublicKey

algorithm field MUST contain the id-ecPublicKey object identifier (see Section 8.1) with NULL parameters field. The MQVuserKeyingMaterial ephemeralPublicKey publicKey field MUST contain the DER-encoding of the ASN.1 type ECPoint (see Section 8.2) representing sending agent's ephemeral EC public key. The MQVuserKeyingMaterial addedukm field, if present, SHOULD contain an octet string of additional user keying material of the sending agent.

keyEncryptionAlgorithm MUST be the mqvSinglePass-sha1kdf-scheme algorithm identifier (see Section 8.1), with the parameters field KeyWrapAlgorithm. The KeyWrapAlgorithm indicates the symmetric encryption algorithm used to encrypt the CEK with the KEK generated using the 1-Pass ECMQV algorithm.

3.2.2 Actions of the sending agent

When using 1-Pass ECMQV with EnvelopedData, the sending agent first obtains the recipient's EC public key and domain parameters, (e.g. from the recipient's certificate) and checks that the domain parameters are the same. The sending agent then determines an integer "keydatalen", which is the KeyWrapAlgorithm symmetric key-size in bits, and also a bit string "SharedInfo", which is the DER encoding of ECC-CMS-SharedInfo (see Section 8.2). The sending agent then performs the key deployment and key agreement operations of the Elliptic Curve MQV Scheme specified in [SEC1, Section 6.2]. As a result, the sending agent obtains:

- an ephemeral public key, which is represented as a value of type ECPoint (see Section 8.2), encapsulated in a bit string, placed in an MQVuserKeyingMaterial ephemeralPublicKey publicKey field (see Section 8.2), and

- a shared secret bit string "K", which is used as the pairwise key-encryption key for that recipient, as specified in [CMS].

The ephemeral public key can be re-used with an AuthenticatedData for greater efficiency.

3.2.3 Actions of the receiving agent

When using 1-Pass ECMQV with EnvelopedData, the receiving agent determines the bit string "SharedInfo", which is the DER encoding of ECC-CMS-SharedInfo (see Section 8.2), and the integer "keydatalen" from the key-size, in bits, of the KeyWrapAlgorithm. The receiving agent then retrieves the static and ephemeral EC public keys of the originator, from the originator and ukm fields as described in Section 3.2.1, and its static EC public key identified in the rid

field and checks that the domain parameters are the same. The receiving agent then performs the key agreement operation of the Elliptic Curve MQV Scheme [SEC1, Section 6.2]. As a result, the receiving agent obtains a shared secret bit string "K" which is used as the pairwise key-encryption key to unwrap the CEK.

4 AuthenticatedData using ECC

This section describes how to use ECC algorithms with the CMS AuthenticatedData format. AuthenticatedData lacks non-repudiation, and so in some instances is preferable to SignedData. (For example, the sending agent might not want the message to be authenticated when forwarded.)

4.1 AuthenticatedData using 1-pass ECMQV

This section describes how to use the 1-Pass elliptic curve MQV (ECMQV) key agreement algorithm with AuthenticatedData. ECMQV is specified in [SEC1]. An advantage of using 1-Pass ECMQV is that it can be used with both EnvelopedData and AuthenticatedData.

4.1.1 Fields of the KeyAgreeRecipientInfo

The AuthenticatedData KeyAgreeRecipientInfo fields are used in the same manner as the fields for the corresponding EnvelopedData KeyAgreeRecipientInfo fields of Section 3.2.1 of this document.

4.1.2 Actions of the sending agent

The sending agent uses the same actions as for EnvelopedData with 1-Pass ECMQV, as specified in Section 3.2.2 of this document.

The ephemeral public key can be re-used with an EnvelopedData for greater efficiency.

Note: if there are multiple recipients, an attack is possible where one recipient modifies the content without other recipients noticing [BON]. A sending agent who is concerned with such an attack SHOULD use a separate AuthenticatedData for each recipient.

4.1.3 Actions of the receiving agent

The receiving agent uses the same actions as for EnvelopedData with 1-Pass ECMQV, as specified in Section 3.2.3 of this document.

Note: see Note in Section 4.1.2.

5 Recommended Algorithms and Elliptic Curves

Implementations of this specification MUST implement either SignedData with ECDSA or EnvelopedData with ephemeral-static ECDH. Implementations of this specification SHOULD implement both SignedData with ECDSA and EnvelopedData with ephemeral-static ECDH. Implementations MAY implement the other techniques specified, such as AuthenticatedData and 1-Pass ECMQV.

Furthermore, in order to encourage interoperability, implementations SHOULD use the elliptic curve domain parameters specified by ANSI [X9.62], NIST [FIPS-186-2] and SECG [SEC2].

6 Certificates using ECC

Internet X.509 certificates [PKI] can be used in conjunction with this specification to distribute agents' public keys. The use of ECC algorithms and keys within X.509 certificates is specified in [PKI-ALG].

7 SMIMECapabilities Attribute and ECC

A sending agent MAY announce to receiving agents that it supports one or more of the ECC algorithms in this document by using the SMIMECapabilities signed attribute [MSG, Section 2.5.2].

The SMIMECapability value to indicate support for the ECDSA signature algorithm is the SEQUENCE with the capabilityID field containing the object identifier ecdsa-with-SHA1 with NULL parameters. The DER encoding is:

```
30 0b 06 07  2a 86 48 ce   3d 04 01 05  00
```

The SMIMECapability capabilityID object identifiers for the supported key agreement algorithms in this document are dhSinglePass-stdDH-sha1kdf-scheme, dhSinglePass-cofactorDH-sha1kdf-scheme, and mqvSinglePass-sha1kdf-scheme. For each of these SMIMECapability SEQUENCEs, the parameters field is present and indicates the supported key-encryption algorithm with the KeyWrapAlgorithm algorithm identifier. The DER encodings that indicate capability of the three key agreement algorithms with CMS Triple-DES key wrap are:

```
30 1c 06 09  2b 81 05 10   86 48 3f 00  02 30 0f 06
0b 2a 86 48  86 f7 0d 01   09 10 03 06  05 00
```

for ephemeral-static ECDH,

```
30 1c 06 09  2b 81 05 10   86 48 3f 00  03 30 0f 06
0b 2a 86 48  86 f7 0d 01   09 10 03 06  05 00
```

for ephemeral-static ECDH with cofactor method, and

```
30 1c 06 09  2b 81 05 10   86 48 3f 00  10 30 0f 06
0b 2a 86 48  86 f7 0d 01   09 10 03 06  05 00
```

for ECMQV.

8 ASN.1 Syntax

The ASN.1 syntax used in this document is gathered in this section for reference purposes.

8.1 Algorithm identifiers

The algorithm identifiers used in this document are taken from [X9.62], [SEC1] and [SEC2].

The following object identifier indicates the hash algorithm used in this document:

```
sha-1 OBJECT IDENTIFIER ::= { iso(1) identified-organization(3)
   oiw(14) secsig(3) algorithm(2) 26 }
```

The following object identifier is used in this document to indicate an elliptic curve public key:

```
id-ecPublicKey OBJECT IDENTIFIER ::= { ansi-x9-62 keyType(2) 1 }
```

where

```
ansi-x9-62 OBJECT IDENTIFIER ::= { iso(1) member-body(2) us(840)
  10045 }
```

When the object identifier id-ecPublicKey is used here with an algorithm identifier, the associated parameters contain NULL.

The following object identifier indicates the digital signature algorithm used in this document:

```
ecdsa-with-SHA1 OBJECT IDENTIFIER ::= { ansi-x9-62 signatures(4)
   1 }
```

When the object identifier ecdsa-with-SHA1 is used within an algorithm identifier, the associated parameters field contains NULL.

The following object identifiers indicate the key agreement algorithms used in this document:

```
dhSinglePass-stdDH-sha1kdf-scheme OBJECT IDENTIFIER ::= {
   x9-63-scheme 2}

dhSinglePass-cofactorDH-sha1kdf-scheme OBJECT IDENTIFIER ::= {
   x9-63-scheme 3}

mqvSinglePass-sha1kdf-scheme OBJECT IDENTIFIER ::= {
   x9-63-scheme 16}
```

where

```
x9-63-scheme OBJECT IDENTIFIER ::= { iso(1)
   identified-organization(3) tc68(133) country(16) x9(840)
   x9-63(63) schemes(0) }
```

When the object identifiers are used here within an algorithm identifier, the associated parameters field contains the CMS KeyWrapAlgorithm algorithm identifier.

8.2 Other syntax

The following additional syntax is used here.

When using ECDSA with SignedData, ECDSA signatures are encoded using the type:

```
ECDSA-Sig-Value ::= SEQUENCE {
   r INTEGER,
   s INTEGER }
```

ECDSA-Sig-Value is specified in [X9.62]. Within CMS, ECDSA-Sig-Value is DER-encoded and placed within a signature field of SignedData.

When using ECDH and ECMQV with EnvelopedData and AuthenticatedData, ephemeral and static public keys are encoded using the type ECPoint.

```
ECPoint ::= OCTET STRING
```

When using ECMQV with EnvelopedData and AuthenticatedData, the sending agent’s ephemeral public key and additional keying material are encoded using the type:

```
MQVuserKeyingMaterial ::= SEQUENCE {
   ephemeralPublicKey OriginatorPublicKey,
   addedukm [0] EXPLICIT UserKeyingMaterial OPTIONAL  }
```

The ECPoint syntax in used to represent the ephemeral public key and placed in the ephemeralPublicKey field. The additional user keying material is placed in the addedukm field. Then the MQVuserKeyingMaterial value is DER-encoded and placed within a ukm field of EnvelopedData or AuthenticatedData.

When using ECDH or ECMQV with EnvelopedData or AuthenticatedData, the key-encryption keys are derived by using the type:

```
ECC-CMS-SharedInfo ::= SEQUENCE {
   keyInfo AlgorithmIdentifier,
   entityUInfo [0] EXPLICIT OCTET STRING OPTIONAL,
   suppPubInfo [2] EXPLICIT OCTET STRING   }
```

The fields of ECC-CMS-SharedInfo are as follows:

keyInfo contains the object identifier of the key-encryption algorithm (used to wrap the CEK) and NULL parameters.

entityUInfo optionally contains additional keying material supplied by the sending agent. When used with ECDH and CMS, the entityUInfo field contains the octet string ukm. When used with ECMQV and CMS, the entityUInfo contains the octet string addedukm (encoded in MQVuserKeyingMaterial).

suppPubInfo contains the length of the generated KEK, in bits, represented as a 32 bit number, as in [CMS-DH]. (E.g. for 3DES it would be 00 00 00 c0.)

Within CMS, ECC-CMS-SharedInfo is DER-encoded and used as input to the key derivation function, as specified in [SEC1, Section 3.6.1]. Note that ECC-CMS-SharedInfo differs from the OtherInfo specified in [CMS-DH]. Here, a counter value is not included in the keyInfo field because the key derivation function specified in [SEC1, Section 3.6.1] ensures that sufficient keying data is provided.

9 Summary

This document specifies how to use ECC algorithms with the S/MIME CMS. Use of ECC algorithm within CMS can result in reduced processing requirements for S/MIME agents, and reduced bandwidth for CMS messages.

References

[X9.62] ANSI X9.62-1998, "Public Key Cryptography For The Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA)", American National Standards Institute, 1999.

[PKI-ALG] Bassham, L., Housley R. and W. Polk, "Algorithms and Identifiers for the Internet X.509 Public Key Infrastructure Certificate and CRL Profile", RFC 3279, April 2002.

[BON] D. Boneh, "The Security of Multicast MAC", Presentation at Selected Areas of Cryptography 2000, Center for Applied Cryptographic Research, University of Waterloo, 2000. Paper version available from http://crypto.stanford.edu/~dabo/papers/mmac.ps

[MUST] Bradner, S., "Key Words for Use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[FIPS-180] FIPS 180-1, "Secure Hash Standard", National Institute of Standards and Technology, April 17, 1995.

[FIPS-186-2] FIPS 186-2, "Digital Signature Standard", National Institute of Standards and Technology, 15 February 2000.

[PKI] Housley, R., Polk, W., Ford, W. and D. Solo, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 3280, April 2002.

[CMS] Housley, R., "Cryptographic Message Syntax", RFC 2630, June 1999.

[IEEE1363] IEEE P1363, "Standard Specifications for Public Key Cryptography", Institute of Electrical and Electronics Engineers, 2000.

[K] B. Kaliski, "MQV Vulnerabilty", Posting to ANSI X9F1 and IEEE P1363 newsgroups, 1998.

[LMQSV] L. Law, A. Menezes, M. Qu, J. Solinas and S. Vanstone, "An efficient protocol for authenticated key agreement", Technical report CORR 98-05, University of Waterloo, 1998.

[CMS-KEA] Pawling, J., "CMS KEA and SKIPJACK Conventions", RFC 2876, July 2000.

[MSG] Ramsdell, B., "S/MIME Version 3 Message Specification", RFC 2633, June 1999.

[CMS-DH] Rescorla, E., "Diffie-Hellman Key Agreement Method", RFC 2631, June 1999.

[SEC1] SECG, "Elliptic Curve Cryptography", Standards for Efficient Cryptography Group, 2000. Available from www.secg.org/collateral/sec1.pdf.

[SEC2] SECG, "Recommended Elliptic Curve Domain Parameters", Standards for Efficient Cryptography Group, 2000. Available from www.secg.org/collateral/sec2.pdf.

Security Considerations

This specification is based on [CMS], [X9.62] and [SEC1] and the appropriate security considerations of those documents apply.

In addition, implementors of AuthenticatedData should be aware of the concerns expressed in [BON] when using AuthenticatedData to send messages to more than one recipient. Also, users of MQV should be aware of the vulnerability in [K].

When 256, 384, and 512 bit hash functions succeed SHA-1 in future revisions of [FIPS], [FIPS-186-2], [X9.62] and [SEC1], then they can similarly succeed SHA-1 in a future revision of this document.

Intellectual Property Rights

obtain a general license or permission for the use of such proprietary rights by implementors or users of this specification can be obtained from the IETF Secretariat.

Acknowledgments

The methods described in this document are based on work done by the ANSI X9F1 working group. The authors wish to extend their thanks to ANSI X9F1 for their assistance. The authors also wish to thank Peter de Rooij for his patient assistance. The technical comments of Francois Rousseau were valuable contributions.

Authors' Addresses

Simon Blake-Wilson
Certicom Corp
5520 Explorer Drive #400
Mississauga, ON L4W 5L1

EMail: sblakewi@certicom.com

Daniel R. L. Brown
pCerticom Corp
5520 Explorer Drive #400
Mississauga, ON L4W 5L1

EMail: dbrown@certicom.com

Paul Lambert

EMail: plambert@sprintmail.com

Full Copyright Statement

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.