**GRIFFIN *Consulting***

Date: 2006-06-10

**Key Commitment**

Using CMS in ECMQV Key Agreement

# Key Commitment Using Cryptographic Message Syntax (CMS) in Elliptic Curve Menezes-Qu-Vanstone (ECMQV) Key Agreement

## Copyright notice



Comments on this white paper should be directed to the author at the following address:

Phillip H. Griffin
GRIFFIN *Consulting*
1625 Glenwood Avenue
Raleigh, North Carolina 27608-2319 USA
Tel. + 1 919 291 0019
E-mail information@phillipgriffin.com
Web http://phillipgriffin.com

# Contents

Page

## Introduction

The document author is Phillip H. Griffin, Principal of GRIFFIN *Consulting*, an Information Security consulting company located in Raleigh, North Carolina, USA. The consultancy provides secure software design and development, national and international security standards committee representation, management and development of security standards, training, and related information security services.

This white paper provides important information for readers who want to advance their understanding of key agreement security techniques. Abstract Cryptographic Message Syntax (CMS) [7, 11] used to convey cryptographic key management information is described in this document, and concrete representations of abstract values are provided in compact binary and XML markup formats. All of the message examples were created using the XCMS Toolkit designed, developed and licensed by GRIFFIN *Consulting*.

## 1 Abstract

This white paper describes a key commitment protocol that provides key control in key agreement schemes used for key establishment. Motivation is provided for the use of a key commitment protocol to ensure that neither party gains advantage during the key distribution procedure phase of a key agreement scheme. Though particular attention is given to the full, 1-pass ECMQV key agreement scheme, the key commitment protocol described in this document is applicable to most key agreement schemes.

Abstract syntax used to define message components is described and example concrete syntax representations of these abstract values are provided using XML [12] markup and compact binary formats. A discussion on how to form and process key commitment protocol messages using Cryptographic Message Syntax (CMS) syntax is presented.

## 2 Terms and definitions

For the purposes of this document, the following terms and definitions apply.

**2.1**
**encryption**
(reversible) transformation of data by a cryptographic algorithm to produce ciphertext, i.e., to hide the information content of the data

**2.2**
**ephemeral**
a key or data specific to one execution of a cryptographic scheme

**2.3**
**key agreement scheme**
a key establishment scheme in which the keying data established is a function of contributions provided by both entities in such a way that neither party can predetermine the value of the keying data

**2.4**
**key derivation function**
**KDF**
a function that takes as input a shared secret value and outputs keying data suitable for later cryptographic use

**2.5**
**keying data**
data suitable for use as cryptographic keys

**2.6**
**message**
set of data elements used to exchange information

**2.7**
**pass**
a set of data sent from one entity to another or received by one entity from another at a particular stage in the operation of a protocol

**2.8**
**private key**
an asymmetric (public-key) system, that key of an entity's key pair that is known only by that entity

**2.9**
**public key**
in an asymmetric (public-key) system, that key of an entity's key pair that is publicly known

**2.10**
**shared secret value**
an intermediate value in a key establishment scheme from which keying data is derived

## 3 Symbols and abbreviated terms

| | |
|---|---|
| ANS | American National Standard |
| ASN.1 | Abstract Syntax Notation One |
| CMS | Cryptographic Message Syntax |
| DER | Distinguished Encoding Rules |
| EC | Elliptic Curve |
| IEC | International Electrotechnical Commission |
| IETF | Internet Engineering Task Force |
| ISO | International Organization for Standardization |
| ITU-T | International Telecommunication Union - Telecommunication Standardization Sector |
| KEK | Key Encryption Key |
| MQV | Menezes-Qu-Vanstone |
| OID | Information object identifier |
| RFC | Request For Comments |
| XCMS | XML Cryptographic Message Syntax |
| XER | XML Encoding Rules of ASN.1 |
| XML | Extensible Markup Language |

## 4 ECMQV key agreement

### 4.1 Overview

The Elliptic Curve Menezes-Qu-Vanstone (ECMQV) key agreement scheme is a mechanism used for key establishment. Using ECMQV, two participants (party *A* and party *B*) can construct a shared secret symmetric encryption key. Since both participants contribute public information to construct the shared secret, neither party need trust the other to select the symmetric encryption key.

ECMQV key agreement can be described in terms of:

1) a setup procedure,

2) a key deployment procedure,

3) an encryption operation, and;

4) a decryption operation.

The setup and key deployment procedures are performed before the actual operation of the key agreement scheme. When the key deployment procedure is performed properly, neither party should be able to predetermine any of the bits of the shared secret key.

The ECMQV setup procedure is used to decide which Elliptic Curve (EC) domain parameters will be used by the parties to perform the ECMQV key agreement scheme. The setup procedure also determines which (if any) key derivation function (KDF) will be used by the two parties. Cryptographic toolkits and service providers often provide a configuration capability with settings for a default elliptic curve and specified hash function to be used for key derivation. The setup values must be the same for both parties participating in the scheme.

The key deployment procedure is used by both participants to distribute one EC static public key and one EC ephemeral public key to the other party. Either of these public keys may be associated with an identity certificate [4]. The domain parameters associated with these keys must be the same, and must agree with those determined during the setup procedure.

Domain parameters may be specified as a "named" elliptic curve [13], whose name is an ASN.1 [1] information object identifier (OID) [5]. For example, the X9.63 [8] standard defines the domain parameters of a 192-bit elliptic curve with a prime characteristic finite field as the named curve **`ansip192r1`**, whose information object identifier is the value **`1.2.840.10045.3.1.1.`**

Once each participant has the two public keys of the other party, the encryption operation can begin. Either party can perform the encryption to initiate operation of the key agreement scheme. Party *A* can use two remote public keys of party *B*, along with their own static and ephemeral public-private key pairs, to generate a symmetric Key Encryption Key (KEK). And party *B* can use two remote public keys of party *A*, along with their own static and ephemeral public-private key pairs, to generate the same KEK. Operation of the key agreement scheme may be performed using the CMS **`EnvelopedData`** message defined in ISO 22895 [7].

## 4.2 Process

Each participant in a full, 1-Pass Elliptic Curve Menezes-Qu-Vanstone (ECMQV) key agreement scheme uses three pairs of keys:

1) a local static public-private key pair, and

2) a local ephemeral public-private key pair, and

3) a remote pair of ephemeral and static public keys.

Each participant contributes a static public key and an ephemeral public key to the key construction process. The remote pair of public keys must be provided by each party to the other party. If this distribution of public key information is done fairly, neither participant can predetermine any of the bits of the value of the shared secret key. These three key pairs allow both parties to derive a shared secret key by following the same six step process:

1) initialize the process with a local private key,

2) pass in the associated local public key,

3) pass in a second local private key,

4) pass in the associated local public key,

5) pass in a remote public key,

6) pass in a second remote public key.

# 5 Key control

## 5.1 Key distribution fairness

To perform the key distribution procedure for ECMQV key agreement without key control, party *A* sends its two public keys to party *B* and party *B* responds by sending its two public keys to party *A*. Where there is complete trust between the two parties, this process may be sufficient. But party *B* may have an unfair advantage over party *A*, since party *A* must disclose the keys it will use to operate the key agreement scheme before party *B* has revealed the keys that it will use.

During the time between receiving keys from party *A* and sending party *A* its keys, party *B* has the opportunity to choose the keys it will use to operate the key agreement scheme, based on the keys revealed by party *A*. Though in some key agreement applications, this time interval may be very small, even a one second interval allows party B to predetermine some of the bits in the shared secret key that will be derived during operation of the key agreement scheme.

Mitchell, Ward and Wilson [14] describe how party *B* may choose its public keys and predetermine some or all of the bits of the shared secret key. They point out that most key agreement schemes defined in ISO/IEC 11770-3 [6] are vulnerable to such attacks. Longer intervals of time between receiving and sending keys allows a greater number of bits of the shared secret key to be predetermined. A key commitment protocol can be used to eliminate this threat.

## 5.2 Key commitment

A key commitment protocol requires two messages. A first message commits the sender to use specific cryptographic keys in the subsequent operation of a key agreement scheme with the message recipient. A second message distributes the actual keys that the sender committed to in the first message.

The first message must contain a hash of the public cryptographic keys the sender is committing to use for key agreement, but not the actual keys. This message is illustrated in the example provided in A.2. This hash must be retained by the message recipient for use when the actual keys are received. The second message must contain the keys, and may contain a hash of the keys that can be verified by the message recipient. This message is illustrated in the example provided in A.3.

For 1-pass ECMQV, this message contains the hash of the two cryptographic keys, one ephemeral public key and one static public key. ISO 22895 defines an ASN.1 type named **`ECKeyCommitment`** for this purpose. CMS type **`DigestedData`** provides a means of exchanging message digest information and optionally, the cryptographic keys needed to implement a key commitment protocol.

# 6 CMS Schema elements

## 6.1 DigestedData

A digested data CMS message is a value of ASN.1 type **`DigestedData`** and contains a message content identifier and a cryptographic hash of the identified content. The message content may be absent or present in the message, and any type of content can be digested.

The ASN.1 schema for type **`DigestedData`** is defined as

```
DigestedData ::= SEQUENCE {
   version           CMSVersion,
   digestAlgorithm   DigestAlgorithmIdentifier,
   encapContentInfo  EncapsulatedContentInfo,
   digest            Digest
}
```

The values of the components of type **DigestedData** are as follows:

— the **version** component is a positive integer that indicates the version number of the schema.

— the **digestAlgorithm** component is a value of type **DigestAlgorithmIdentifier**, which contains two components named **algorithm** and **parameters**. The **algorithm** component identifies a message digest algorithm, and the optional **parameters** component contains any associated parameters.

— the **encapContentInfo** component is a value of type **EncapsulatedContentInfo**, which contains two components named **eContentType** and **eContent**. The **eContentType** component identifies the type of content being digested and the optional **eContent** contains this content when present.

In the examples presented in Annex A, the key agreement scheme is ECMQV and the digested content is a value of type **ECKeyCommitment**. For other key agreement schemes, a different ASN.1 type can be used in the key commitment protocol, since type **DigestedData** supports arbitrary content.

— the **digest** component is a value of type **Digest** and contains the result of applying the hash function indicated by the **digestAlgorithm** component to the message content. In the examples presented in Annex A, the hash function is applied to a value of type **ECKeyCommitment**.

A complete detailed definition of these types can be found in the ISO 22895 standard. Values of the **DigestedData** type are provided in A.2 and A.3 which describe the two messages needed to implement a key commitment protocol using CMS.

## 6.2 ECKeyCommitment

In the key commitment protocol described in this document, the message digest process is applied to a value of ASN.1 type **ECKeyCommitment**, which is defined as

```
ECKeyCommitment ::= SEQUENCE {
   staticPublicKey     ECPoint,
   ephemeralPublicKey  ECPoint  OPTIONAL
}

ECPoint ::= OCTET STRING
```

a sequence of two components named **staticPublicKey** and **ephemeralPublicKey**. Both components are EC public keys defined as values of type **ECPoint**. A value of this type is provided in A.1.

# Annex A

# Protocol message examples

## A.1 Overview

The example key commitment messages shown below in subclause A.2 and A.3 are values of CMS type **DigestedData**, defined in subclause 6.1 These values are represented as XML markup encoded using the XML Encoding Rules (XER) [3] of ASN.1 . The same abstract values can also be represented in a compact binary format encoded using the Distinguished Encoding Rules (DER) [2] to reduce the size of the encoded value by 105 octets. For the purpose of illustration, both formats are provided in the example messages.

The content to be digested is an XML encoding of a value of ISO 22895[1] type **ECKeyCommitment** as it would be used for key control in a key commitment protocol in the key deployment procedure of a key agreement scheme.

In both example encodings, the input to the message digest process is a sequence of two Elliptic Curve Digital Signature Algorithm (ECDSA) public keys represented as character data using XML markup as the value

```
<ECKeyCommitment>
   <staticPublicKey>
      4441443537383246434534324645344544364141
      4141394443413643393639303333453134424431
   </staticPublicKey>
   <ephemeralPublicKey>
      4530443438323632313331433944423841434639
      3745454141353733463531443141354137344630
   </ephemeralPublicKey>
</ECKeyCommitment>
```

This value contains a static ECDSA public key "**DAD5782FCE42FE4ED6AAAA9DCA6C969033E14BD1**" and an ephemeral ECDSA public key "**E0D48262131C9DB8ACF97EEAA573F51D1A5A74F0**" shown here using the hexadecimal representation needed for display of values type ECPoint. When input to the message digest process, the value may be encoded using canonical XER, which contains no white space or other formatting information as shown here.

## A.2 Commitment

The example key commitment message that follows shows the XML encoded **DigestedData** value displayed with the content to be digested, a value of type **ECKeyCommitment**, not present in the message. By omitting the content and only transferring the hash of the content, party *A* can send party *B* the hash of the public keys party *A* will use when it participates in the operation of a key agreement scheme, without revealing the keys.

Line 1 indicates the version number of the syntax, the value **3**.

Lines 2-6 describe the message digest algorithm and any associated digest algorithm parameter used to compute the message digest in line 13.

---

[1] The ISO 22895 Cryptographic syntax schema combines two US National standards, X9.73 CMS [9] X9.96 XCMS [10] which share a common ASN.1 schema and are aligned with the S/MIME CMS RFC [11] from IETF.

Line 4 is the information object identifier of the SHA-256 message digest algorithm. This algorithm has no associated parameters.

Line 9 indicates that the type of value digested is ordinary data, and not another CMS type. The associated content is "detached", and not present in this message.

Line 13 contains the result of computing a message digest on ordinary data using the SHA-256 message digest algorithm specified in line 4. The message digest size is 256 bits (32 octets) in length and provides 128 bits of security. Disregarding white space, the value is encoded in 295 octets.

```
000 <DigestedData>
001    <version> 3 </version>
002    <digestAlgorithm>
003       <algorithm>
004          2.16.840.1.101.3.4.2.1
005       </algorithm>
006    </digestAlgorithm>
007    <encapContentInfo>
008       <eContentType>
009          1.2.840.113549.1.7.1
010       </eContentType>
011    </encapContentInfo>
012    <digest>
013       1B1FA4C0B3CFE4F4FE0A299A329C53EF285EC6A392BABC5BC026AC4CFC332812
014    </digest>
015 </DigestedData>
```

The complete DER encoding of the same **DigestedData** message follows, displayed using hexadecimal notation. Using DER the message is encoded in 65 octets.

```
303F020103300B06096086480165030402 01300B
06092A864886F70D01070104201B1FA4C0B3CFE4
F4FE0A299A329C53EF285EC6A392BABC5BC026AC
4CFC332812
```

## A.3 Distribution

The following example key distribution message shows the XML encoded **DigestedData** value displayed with the content to be digested, a value of type **ECKeyCommitment**, present in the message. The content to be digested is a value of ASN.1 type **OCTET STRING**, an opaque value. Since the encoded value of **ECKeyCommitment** is encapsulated in an "octet hole", the details of its structure are obscured.

By including the content in the message along with the hash of the content, party *A* can distribute the public keys it previously committed to use in the operation of a key agreement scheme to party *B*. Party *B* can verify that the value in the digest component of type **DigestedData** is correct, then compare that value to the digest in a prior key commitment message received from party *A*. If these two digest values are identical, then party *A* and party *B* can both assume that the key agreement key distribution procedure was fair.

Disregarding white space, the value is encoded in 874 octets.

Line 1 indicates the version number of the syntax, the value **3**.

Lines 2-6 describe the message digest algorithm and any associated digest algorithm parameter used to compute the message digest in line 13.

Line 4 is the information object identifier of the SHA-256 message digest algorithm. This algorithm has no associated parameters.

```
000 <DigestedData>
001    <version> 3 </version>
002    <digestAlgorithm>
003       <algorithm>
004          2.16.840.1.101.3.4.2.1
005       </algorithm>
006    </digestAlgorithm>
007    <encapContentInfo>
008       <eContentType>
009          1.2.840.113549.1.7.1
010       </eContentType>
011       <eContent>
012          203C45434B6579436F6D6D69746D656E743E0D0A20203C737461746963507562
013          6C69634B65793E0D0A2020203434343134343335333733833332343634333435
014          333433323436343533343435343433363431343134313413433399343434333431
015          333634333339333633393330 0A20202033333333343533313334343234343331
016          0D0A20203C2F737461746963507562 6C69634B65793E0D0A20203C657068656D
017          6572616C5075626C69634B65793E0D0A2020203435333034343343338333233
018          363332333133333331343333393434343233383431343334363393337343534
019          353431343133353337333334363335333134340A2020203331343133353433133
020          373334343633300D0A20203C2F657068656D6572616C5075626C69634B65793E
021          0D0A203C2F45434B6579436F6D6D69746D656E743E0D0A
022       </eContent>
023    </encapContentInfo>
024    <digest>
025       1B1FA4C0B3CFE4F4FE0A299A329C53EF285EC6A392BABC5BC026AC4CFC332812
026    </digest>
027 </DigestedData>
```

The complete DER encoding of the same **DigestedData** message follows, displayed using hexadecimal notation. Using DER the message is encoded in 384 octets.

```
3082017C020103300B06096086480165030402013082014606092A864886F70D01070180820137203
C45434B6579436F6D6D69746D656E743E0D0A20203C737461746963507562 6C69634B65793E0D0A20
202034343431343433353337333833323436343334353334333234363435333434353434333634313
431343134313339343434333431333634333339333633393300A2020203333333334353331333434
32343433310D0A20203C2F737461746963507562 6C69634B65793E0D0A20203C657068656D6572616
C5075626C69634B65793E0D0A2020203435333034343334333833323336333233313333333134343333
393434343233383431343334363339333734353435343134313335333733333436333533313434 0A2
0202033313431333534313337333434363330 0D0A20203C2F657068656D6572616C5075626C69634B
65793E0D0A203C2F45434B6579436F6D6D69746D656E743E0D0A04201B1FA4C0B3CFE4F4FE0A299A3
29C53EF285EC6A392BABC5BC026AC4CFC332812
```

# Bibliography


[1] ISO/IEC 8824:2001 (All parts) | ITU-T Recommendation X.680-series (2000), *Information Technology - Abstract Syntax Notation One (ASN.1)*

[2] ISO/IEC 8825-1:2001 | ITU-T Recommendation X.690 (2000), *Information Technology - ASN.1 Encoding Rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)*

[3] ISO/IEC 8825-2:2001 | ITU-T Recommendation X.693 (2000), *Information Technology - ASN.1 Encoding Rules: Specification of XML Encoding Rules (XER)*

[4] ISO/IEC 9594-8 | ITU-T X.509 Information technology - Open Systems Interconnection - The Directory: Public-key and attribute certificate frameworks

[5] ISO/IEC 9834-8 | ITU-T Rec. X.667, Information technology – Open Systems Interconnection – Procedures for the operation of OSI Registration Authorities: Generation and Registration of Universally Unique Identifiers (UUIDs) and their Use as ASN.1 Object Identifier Components

[6] ISO/IEC 11770-3:1999 Information Technology – Security Techniques – Key Management: Part 3 - Mechanisms Using Asymmetric Techniques

[7] ISO CD 22895 Financial Services – Security – Cryptographic Syntax Schema

[8] ANS X9.63:2001, Public Key Cryptography for the Financial Services Industry: Key Agreement and Key Transport Using Elliptic Curve Cryptography

[9] ANS X9.73:2003, Cryptographic Message Syntax (CMS)

[10] ANS X9.96:2004, XML Cryptographic Message Syntax (XCMS)

[11] RFC 3852, Cryptographic Message Syntax (CMS), http://ietfreport.isoc.org/rfc/rfc3852.txt

[12] Extensible Markup Language (XML) 1.0 (Third Edition), Copyright © 1994-2003 World Wide Web Consortium (W3C®), (Massachusetts Institute of Technology, European Research Consortium for Informatics and Mathematics, Keio University), W3C Recommendation 04, http://www.w3.org/TR/2004/REC-xml-20040204/

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

[14] C.J. Mitchell, M. Ward and P. Wilson, *"Key control in key agreement protocols",* ELECTRONICS LETTERS *14th May 1998 Vol. 34 No. 10, pages 980-981,* http://www.iee.org/