Internet Engineering Task Force (IETF)
Request for Comments: 6210
Category: Experimental
ISSN: 2070-1721

J. Schaad
Soaring Hawk Consulting

April 2011

Experiment: Hash Functions with Parameters in the Cryptographic Message Syntax (CMS) and S/MIME

Abstract

New hash algorithms are being developed that may include parameters. Cryptographic Message Syntax (CMS) has not currently defined any hash algorithms with parameters, but anecdotal evidence suggests that defining one could cause major problems. This document defines just such an algorithm and describes how to use it so that experiments can be run to find out how bad including hash parameters will be.

Status of This Memo

This document is not an Internet Standards Track specification; it is published for examination, experimental implementation, and evaluation.

This document defines an Experimental Protocol for the Internet community. This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Not all documents approved by the IESG are a candidate for any level of Internet Standard; see Section 2 of RFC 5741.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at http://www.rfc-editor.org/info/rfc6210.

Table of Contents

1. Introduction

At the present time, all hash algorithms that are used in Cryptographic Message Syntax (CMS) implementations are defined as having no parameters. Anecdotal evidence suggests that if a hash algorithm is defined that does require the presence of parameters, there may be extensive problems. This document presents the details needed to run an experiment so that the community can find out just how bad the situation really is and, if needed, either make drastic changes in implementations or make sure that any hash algorithms chosen do not have parameters.

In CMS data structures, hash algorithms currently exist in the following locations:

- SignerInfo.digestAlgorithm - holds the digest algorithm used to compute the hash value over the content.
- DigestedData.digestAlgorithm - holds the digest algorithm used to compute the hash value over the content.
- AuthenticatedData.digestAlgorithm - holds the digest algorithm used to compute the hash value over the content.
- SignedData.digestAlgorithms - an optional location to hold the set of digest algorithms used in computing the hash value over the content.
- multipart/signed micalg - holds a textual indicator of the hash algorithm for multipart signed MIME messages.

The first three locations hold the identification of a single hash, and would hold the parameters for that hash. It’s mandatory to fill these fields.

The ASN.1 structures defined for the DigestedData and AuthenticatedData types place the digest algorithm field before the encapsulated data field. This means that the hash algorithm (including the parameters) is fully defined, and therfore can be instantiated, before the hash function would start hashing the encapsulated data.

In the ASN.1 defined for the SignedData type, the value of SignerInfo.digestAlgorithm is not seen until the content has been processed. This is the reason for the existence of the SignedData.digestAlgorithms field, so that the set of all digest algorithms used can be seen prior to the content being processed. It is not currently mandatory to fill in this field, and the signature

validation process is supposed to succeed even if this field is absent. (RFC 5652 says signature validation MAY fail if the digest algorithm is absent.)

For the case of detached content, the ASN.1 structures need to be processed before processing the detached content in order to obtain the parameters of the hash function. The MIME multipart/signature content type attempts to avoid this problem by defining a micalg field that contains the set of hash algorithms (with parameters) so that the hash functions can be set up prior to processing the content.

When processing multipart/signed messages, two paths exists:

1. Process the message content before the ASN.1. The steps involved are:

   * Get a set of hash functions by looking at the micalg parameter and potentially add a set of generic algorithms.

   * Create a hasher for each of those algorithms.

   * Hash the message content (the first part of the multipart).

   * Process the ASN.1 and have a potential failure point if a hash algorithm is required but was not computed.

2. Process the message content after the ASN.1. The steps involved are:

   * Save the message content for later processing.

   * Parse the ASN.1 and build a list of hash functions based on its content.

   * Create a hasher for each of those algorithms.

   * Hash the saved message content.

   * Perform the signature validation.

The first path allows for single-pass processing, but has the potential that a fallback path needs to be added in some cases. The second path does not need a fallback path, but does not allow for single-pass processing.

The fallback path above may also be needed for the encapsulated content case. Since it is optional to place hash algorithms in the SignedData.digestAlgorithms field, the content will be completely parsed before the set of hash algorithms used in the various SignerInfo structures are determined. It may be that an update to CMS is required to make population of the SignedData.digestAlgorithms field mandatory, in the event that a parameterized hash algorithm is adopted.

In this document, a new hash function is created that is based on the XOR operator and on MD5. MD5 was deliberately used as the basis of this digest algorithm since it is known to be insecure, and I do not want to make any statements that the hash algorithm designed here is in any way secure. This hash function MUST NOT be released as shipping code, it is designed only for use in experimentation. An example of a parameterized hash algorithm that might be standardized is a scheme developed by Shai Halevi and Hugo Krawczyk [RANDOM-HASH].

1.1. Notation

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].

2. XOR-MD5 Digest Algorithm

The XOR-MD5 digest algorithm has been designed to use two existing operators, XOR and the MD5 hash algorithm [MD5]. The hash algorithm works as follows:

1. A random XOR string consisting of exactly 64 bytes is created.

2. The input content is broken up into 64-byte blocks. The last block may be less that 64 bytes.

3. Each block is XOR-ed with the random string. The last block uses the same number of bits from the random string as it contains.

4. The resulting string is run through the MD5 hash function.

The length of the XOR string was designed to match the barrel size of the MD5 hash function.

3. ASN.1 Encoding

The following ASN.1 is used to define the algorithm:

```
mda-xor-md5-EXPERIMENT DIGEST-ALGORITHM ::= {
   IDENTIFIER id-alg-MD5-XOR-EXPERIMENT
   PARAMS TYPE MD5-XOR-EXPERIMENT ARE required
}

id-alg-MD5-XOR-EXPERIMENT OBJECT IDENTIFIER ::= {
   iso(1) member-body(2) us(840) rsadsi(113549)
   pkcs(1) pkcs-9(9) smime(16) id-alg(3) 13
}

MD5-XOR-EXPERIMENT ::= OCTET STRING (SIZE(64))
```

The octet string holds the value of the random XOR string.

4. CMS ASN.1 Handling

The algorithm is added to the DigestAlgorithmSet in [CMS].

When this algorithm is used in a signed message, it is REQUIRED that the algorithm be placed in the SignedData.digestAlgorithms sequence. The algorithm MUST appear in the sequence at least once for each unique set of parameters. The algorithm SHOULD NOT appear multiple times with the same set of parameters.

5. MIME Handling

This section defines the string that appears in the micalg parameter.

The algorithm is identified by the string xor-md5. The parameters for the algorithm are the hex-encoded Distinguished Encoding Rules (DER) ASN.1 encoding. The parameters and the identifier string are separated by a colon. One of the issues that needs to be addressed is the fact that this will generate very long data values for parameters. These will be too long for many systems to deal with. The issue of how to deal with this has been addressed in [RFC2231] by creating a method to fragment values. An example content-type string that has been fragmented is:

```
Content-Type: multipart/signed;
  protocol="application/pkcs7-signature";
  micalg*0="sha1, xor-md5:044001020304050607080090a0b0c0d0e0f0011";
  micalg*1="12131415161718191a1b1c1d1e1f1021222324252627282 92a2b";
  micalg*2="2c2d2e2f203132333435363738";
  micalg*3="393a3b3c3d3e3f30";  boundary=boundar42
```

Arguments could be made that the string should be base64 encoded rather than hex encoded. The advantage is that the resulting encoding is shorter. This could be significant if there are a substantial number of parameters and of a substantial size. Even with the above example, it was necessary to break the encoding across multiple lines. The downside would be the requirement that the micalg parameter always be quoted.

It may be reasonable to require that whitespace be inserted only on encoding boundaries, but it seems to be overly restrictive.

6. IANA Considerations

All identifiers are assigned out of the S/MIME OID arc.

7. Security Considerations

The algorithm XOR-MD5 is not designed for general-purpose use. The hash algorithm included here is designed for running this experiment and nothing more.

This document makes no representation that XOR-MD5 is a secure digest algorithm. I believe that the algorithm is no more secure than MD5, and I consider MD5 to be a broken hash algorithm for many purposes.

One known issue with the algorithm at present is the fact that the XOR pattern is always 64 bytes long, even if the data is shorter. This means that there is a section of the data than can be manipulated without changing the hash. In a real algorithm, this should either be truncated or forced to a known value.

8. References

8.1. Normative References

[ASN.1-2008] ITU-T, "ITU-T Recommendations X.680, X.681, X.682, and X.683", 2008.

[CMS] Housley, R., "Cryptographic Message Syntax (CMS)", RFC 5652, September 2009.

[MD5] Rivest, R., "The MD5 Message-Digest Algorithm", RFC 1321, April 1992.

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

[RFC2231] Freed, N. and K. Moore, "MIME Parameter Value and Encoded Word Extensions: Character Sets, Languages, and Continuations", RFC 2231, November 1997.

[SMIME-MSG] Ramsdell, B. and S. Turner, "Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.2 Message Specification", RFC 5751, January 2010.

8.2. Informative References

[CMS-ASN] Hoffman, P. and J. Schaad, "New ASN.1 Modules for Cryptographic Message Syntax (CMS) and S/MIME", RFC 5911, June 2010.

[RANDOM-HASH] Halevi, S. and H. Krawczyk, "Strengthening Digital Signatures via Random Hashing", January 2007, <http://webee.technion.ac.il/~hugo/rhash/rhash.pdf>.

[RFC5912] Hoffman, P. and J. Schaad, "New ASN.1 Modules for the Public Key Infrastructure Using X.509 (PKIX)", RFC 5912, June 2010.

[SMIME-EXAMPLES] Hoffman, P., "Examples of S/MIME Messages", RFC 4134, July 2005.

Appendix A. Examples

Provided here are a set of simple S/MIME messages [SMIME-MSG] that are for testing. The content used is the same as that found in Section 2.1 of [SMIME-EXAMPLES]. The certificates and key pairs found in [SMIME-EXAMPLES] are also used here.

The Perl script in Appendix A of [SMIME-EXAMPLES] can be used to extract the binary examples from this file. The MIME examples can be extracted with a standard text editor.

Note: The examples presented here have not been independently verified. I was unable to use the Microsoft APIs because of the new cryptographic hash algorithm. However, for the purposes of this experiment, I believe that the form of the messages, which can be verified visually as correct, is more important than the question of the message validating.

A.1. Encapsulated Signed Data Example

This section contains a detached signed data example. The content was hashed with the MD5-XOR algorithm defined in this document. The signature is performed using RSA with MD5. The signature is wrapped as an embedded signed mime message.

```
MIME-Version: 1.0
To: BobRSA@example.com
From: AliceDss@example.com
Subject: MD5-XOR example message
Message-Id: <34567809323489fd.esc@example.com>
Date: Wed, 16 Dec 2010 23:13:00 -0500
Content-Type: application/pkcs7-mime; smime-type=signed-data;
  name=smime.p7m;
  micalg*0="xor-md5: 04400102034050607080090a0b0c0d0e0f10";
  micalg*1="111213415161718191a1b1c1d1e1f202122234252627282 92a2b2c";
  micalg*2="2d2e2f303132333435363738393a3b3c3d3e3f40"
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename=smime.p7m

MIIEqAYJKoZIhvcNAQcCoIIEmTCCBJUCAQExUTBPBgsqhkiG9w0BCRADDQRAAQIDBAUGBw
gJCgsMDQ4PEBESEwQVFhcYGRobHB0eHyAhIiMEJSYnKCkqKywtLi8wMTIzBDU2Nzg5Ojs8
PT4/QDArBgkqhkiG9w0BBwGgHgQcVGhpcyBpcyBzb21lIHNhbXBsZSBjb250ZW50LqCCAi
swggInMIIBkKADAgECAhBGNGvHgABWvBHTbi7NXXHQMA0GCSqGSIb3DQEBBQUAMBIxEDAO
BgNVBAMTB0NhcmxSU0EwHhcNOTkwOTE5MDEwOTAyWhcNMzkxMjMxMjM1OTU5WjARMQ8wDQ
YDVQQDEwZCb2JSU0EwgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAKnhZ5g/OdVf8qCT
QV6meYmFyDVdmpFb+x0B2hlwJhcPvaUi0DWFbXqYZhRBXM+3twg7CcmRuBlpN235ZR572a
kzJKN/O7uvRgGGNjQyywcDWVL8hYsxBLjMGAgUSOZPHPtdYMTgXB9T039T2GkB8QX4enDR
voPGXzjPHCyqaqfrAgMBAAGjfzB9MAwGA1UdEwEB/wQCMAAwDgYDVR0PAQH/BAQDAgUgMB
```

```
 8GA1UdIwQYMBaAFOngkCeseCB6mtNM8kI3TiKunji7MB0GA1UdDgQWBBTo9Lhn2LOWpCrz
 Eaop05Vahha0JDAdBgNVHREEFjAUgRJCb2JSU0FAZXhhbXBsZS5jb20wDQYJKoZIhvcNAQ
 EFBQADgYEAe45mxfEQPxAgTIhxq3tAayEz+kqV3p0OW2uUIQXA8uF+Ks2ck4iH+4u3fn1B
 YeHk1m354gRVYUW8ZCdEwKG9WXnZHWQ8IdZFsF1oM5LqrPFX5YF9mOY1kaM53nf06Bw7Kd
 x/UQeX8zbwUArdm962XjgRK/tX6oltrcmI2I/PK9MxggHfMIIB2wIBATAmMBIxEDAOBgNV
 BAMTB0NhcmxSU0ECEEY0a8eAAFa8EdNuLs1dcdAwTwYLKoZIhvcNAQkQAw0EQAECAwQFBg
 cICQoLDA0ODxAREhMEFRYXGBkaGxwdHh8gISIjBCUmJygpKissLS4vMDEyMwQ1Njc4OTo7
 PD0+P0CggcowGAYJKoZIhvcNAQkDMQsGCSqGSIb3DQEHATAcBgkqhkiG9w0BCQUxDxcNMD
 kxMjEwMjMyNTAwWjAfBgkqhkiG9w0BCQQxEgQQlmmuYRtXnoPqECtrSd3A+TBvBgkqhkiG
 9w0BCTQxYjBgME8GCyqGSIb3DQEJEAMNBEABAgMEBQYHCAkKCwwNDg8QERITBBUWFxgZGh
 scHR4fICEiIwQlJicoKSorLC0uLzAxMjMENTY3ODk6Ozw9Pj9AoQ0GCSqGSIb3DQEBBAUA
 MA0GCSqGSIb3DQEBBAUABIGAClMpfG4IL1yAdRxWdvYKbtuFz1XKnFqo9ui7V5PndjlDut
 yib02knY7UtGNhg6oVEkiZHxYh/iLuoLOHSFA1P4ZacTYrEKChF4K18dsqvlFip1vn8BG/
 ysFUDfbx5VcTG2Md0/NHV+qj5ihqM+Pye6Urp+5jbqVgpZOXSLfP+pI=

 |>sd.bin
 |MIIEqAYJKoZIhvcNAQcCoIIEmTCCBJUCAQExUTBPBgsqhkiG9w0BCRADDQRAAQIDBAUGBw
 |gJCgsMDQ4PEBESEwQVFhcYGRobHB0eHyAhIiMEJSYnKCkqKywtLi8wMTIzBDU2Nzg5Ojs8
 |PT4/QDArBgkqhkiG9w0BBwGgHgQcVGhpcyBpcyBzb21lIHNhbXBsZSBjb250ZW50LqCCAi
 |swggInMIIBkKADAgECAhBGNGvHgABWvBHTbi7NXXHQMA0GCSqGSIb3DQEBBQUAMBIxEDAO
 |BgNVBAMTB0NhcmxSU0EwHhcNOTkwOTE5MDEwOTAyWhcNMzkxMjMxMjM1OTU5WjARMQ8wDQ
 |YDVQQDEwZCb2JSU0EwgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAKnhZ5g/OdVf8qCT
 |QV6meYmFyDVdmpFb+x0B2hlwJhcPvaUi0DWFbXqYZhRBXM+3twg7CcmRuBlpN235ZR572a
 |kzJKN/O7uvRgGGNjQyywcDWVL8hYsxBLjMGAgUSOZPHPtdYMTgXB9T039T2GkB8QX4enDR
 |voPGXzjPHCyqaqfrAgMBAAGjfzB9MAwGA1UdEwEB/wQCMAAwDgYDVR0PAQH/BAQDAgUgMB
 |8GA1UdIwQYMBaAFOngkCeseCB6mtNM8kI3TiKunji7MB0GA1UdDgQWBBTo9Lhn2LOWpCrz
 |Eaop05Vahha0JDAdBgNVHREEFjAUgRJCb2JSU0FAZXhhbXBsZS5jb20wDQYJKoZIhvcNAQ
 |EFBQADgYEAe45mxfEQPxAgTIhxq3tAayEz+kqV3p0OW2uUIQXA8uF+Ks2ck4iH+4u3fn1B
 |YeHk1m354gRVYUW8ZCdEwKG9WXnZHWQ8IdZFsF1oM5LqrPFX5YF9mOY1kaM53nf06Bw7Kd
 |x/UQeX8zbwUArdm962XjgRK/tX6oltrcmI2I/PK9MxggHfMIIB2wIBATAmMBIxEDAOBgNV
 |BAMTB0NhcmxSU0ECEEY0a8eAAFa8EdNuLs1dcdAwTwYLKoZIhvcNAQkQAw0EQAECAwQFBg
 |cICQoLDA0ODxAREhMEFRYXGBkaGxwdHh8gISIjBCUmJygpKissLS4vMDEyMwQ1Njc4OTo7
 |PD0+P0CggcowGAYJKoZIhvcNAQkDMQsGCSqGSIb3DQEHATAcBgkqhkiG9w0BCQUxDxcNMD
 |kxMjEwMjMyNTAwWjAfBgkqhkiG9w0BCQQxEgQQlmmuYRtXnoPqECtrSd3A+TBvBgkqhkiG
 |9w0BCTQxYjBgME8GCyqGSIb3DQEJEAMNBEABAgMEBQYHCAkKCwwNDg8QERITBBUWFxgZGh
 |scHR4fICEiIwQlJicoKSorLC0uLzAxMjMENTY3ODk6Ozw9Pj9AoQ0GCSqGSIb3DQEBBAUA
 |MA0GCSqGSIb3DQEBBAUABIGAClMpfG4IL1yAdRxWdvYKbtuFz1XKnFqo9ui7V5PndjlDut
 |yib02knY7UtGNhg6oVEkiZHxYh/iLuoLOHSFA1P4ZacTYrEKChF4K18dsqvlFip1vn8BG/
 |ysFUDfbx5VcTG2Md0/NHV+qj5ihqM+Pye6Urp+5jbqVgpZOXSLfP+pI=
 |<sd.bin
```

A.2. Multipart Signed Message

This section contains a detached signed data example. The content was hashed with the MD5-XOR algorithm defined in this document. The signature is performed using RSA with MD5. The signature is wrapped as a detached signed mime message.

```
MIME-Version: 1.0
To: User2@example.com
From: BobRSA@example.com
Subject: MD5-XOR signing example
Message-Id: <091218002550300.249@example.com>
Date: Fri, 18 Dec 2010 00:25:21 -0300
Content-Type: multipart/signed;
  micalg*0="xor-md5: 0440010203405060708090a0b0c0d0e0f10";
  micalg*1="111213415161718191a1b1c1d1e1f202122234252627282 92a2b2c2d2e";
  micalg*2="2f303132334353637383 93a3b3c3d3e3f40";
    boundary="----=_NextBoundry____Fri,_18_Dec_2009_00:25:21";
    protocol="application/pkcs7-signature"

This is a multi-part message in MIME format.

------=_NextBoundry____Fri,_18_Dec_2009_00:25:21

This is some sample content.
------=_NextBoundry____Fri,_18_Dec_2009_00:25:21
Content-Type: application/pkcs7-signature; name=smime.p7s
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename=smime.p7s

MIIEiAYJKoZIhvcNAQcCoIIEeTCCBHUCAQExUTBPBgsqhkiG9w0BCRADDQRAAQIDBAUGBw
gJCgsMDQ4PEBESEwQVFhcYGRobHB0eHyAhIiMEJSYnKCkqKywtLi8wMTIzBDU2Nzg5Ojs8
PT4/QDALBgkqhkiG9w0BBwGgggIrMIICJzCCAZCgAwIBAgIQRjRrx4AAVrwR024uzV1x0D
ANBgkqhkiG9w0BAQUFADASMRAwDgYDVQQDEwdDYXJsUlNBMB4XDTk5MDkxOTAxMDkwMloX
DTM5MTIzMTIzNTk1OVowETEPMA0GA1UEAxMGQm9iUlNBMIGfMA0GCSqGSIb3DQEBAQUAA4
GNADCBiQKBgQCp4WeYPznVX/Kgk0FepnmJhcg1XZqRW/sdAdoZcCYXD72lItA1hW16mGYU
QVzPt7cIOwnJkbgZaTdt+WUee9mpMySjfzu7r0YBhjY0MssHA1lS/IWLMQS4zBgIFEjmTx
z7XWDE4FwfU9N/U9hpAfEF+Hpw0b6Dxl84zxwsqmqn6wIDAQABo38wfTAMBgNVHRMBAf8E
AjAAMA4GA1UdDwEB/wQEAwIFIDAfBgNVHSMEGDAWgBTp4JAnrHggeprTTPJCN04irp44uz
AdBgNVHQ4EFgQU6PS4Z9izlqQq8xGqKdOVWoYWtCQwHQYDVR0RBBYwFIESQm9iUlNBQGV4
YW1wbGUuY29tMA0GCSqGSIb3DQEBBQUAA4GBAHuOZsXxED8QIEyIcat7QGshM/pKld6dDl
trlCEFwPLhfirNnJOIh/uLt359QWHh5NZt+eIEVWFFvGQnRMChvVl52R1kPCHWRbBdaDOS
6qzxV+WBfZjmNZGjOd539OgcOyncf1EHl/M28FAK3Zvetl44ESv7V+qJba3JiNiPzyvTMY
IB3zCCAdsCAQEwJjASMRAwDgYDVQQDEwdDYXJsUlNBAhBGNGvHgABWvBHTbi7NXXHQME8G
CyqGSIb3DQEJEAMNBEABAgMEBQYHCAkKCwwNDg8QERITBBUWFxgZGhscHR4fICEiIwQlJi
coKSorLC0uLzAxMjMENTY3ODk6Ozw9Pj9AoIHKMBgGCSqGSIb3DQEJAzELBgkqhkiG9w0B
BwEwHAYJKoZIhvcNAQkFMQ8XDTEwMTIxMDIzMjUwMFowHwYJKoZIhvcNAQkEMRIEEJZprm
EbV56D6hAra0ndwPkwbwYJKoZIhvcNAQk0MWIwYDBPBgsqhkiG9w0BCRADDQRAAQIDBAUG
BwgJCgsMDQ4PEBESEwQVFhcYGRobHB0eHyAhIiMEJSYnKCkqKywtLi8wMTIzBDU2Nzg5Oj
s8PT4/QKENBgkqhkiG9w0BAQQFADANBgkqhkiG9w0BAQQFAASBgEDMeyAkXMYqg/wW2B3P
i8HWwGnZVA/4muJJ7+dEPacv3bRqE7n4dP0vXIYR7TJ1eRJk9uB/wry2fRPcnG3Y/Rn0Jy
CqXsb+dXXfwOGK/rvLvJOloXUCy4+HxQk6eaYIBrjiVIUgZjpZXGJcZg2xq5yH1e4aw5Ov
fQlfQXPiKp1l

------=_NextBoundry____Fri,_18_Dec_2009_00:25:21--
```

A.3. Authenticated Data Example

This section contains an authenticated data example. The content was hashed with the MD5-XOR algorithm defined in this document. The authentication was done with the HMAC-SHA1 algorithm. The key is transported using RSA encryption to BobRSASignByCarl certificate.

```
MIME-Version: 1.0
To: BobRSA@example.com
From: AliceDss@example.com
Subject: MD5-XOR example message
Message-Id: <34567809323489fd.esc@example.com>
Date: Wed, 16 Dec 2010 23:13:00 -0500
Content-Type: application/pkcs7-mime; smime-type=authenticated-data;
  name=smime.p7m;
  micalg*0="xor-md5: 0440010203405060708090a0b0c0d0e0f10";
  micalg*1="111213415161718191a1b1c1d1e1f202122234252627282 92a2b2c2d2e";
  micalg*2="2f303132343536373839 3a3b3c3d3e3f40"
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename=smime.p7m

MIICRQYLKoZIhvcNAQkQAQKgggI0MIICMAIBADGBwDCBvQIBADAmMBIxEDAOBgNVBAMMB0
NhcmxSU0ECEEY0a8eAAFa8EdNuLs1dcdAwDQYJKoZIhvcNAQEBBQAEgYCH70EpEikY7deb
859YJRAWfFondQv1D4NFltw6C1ceheWnlAU0C2WEXr3LUBXZp1/PSte29FnJxu5bXCTn1g
elMm6zNlZNWNd0KadVBcaxi1n8L52tVM5sWFGJPO5cStOyAka2ucuZM6iAnCSkn1Ju7fgU
5j2g3bZ/IM8nHTcygjAKBggrBgEFBQgBAqFPBgsqhkiG9w0BCRADDQRAAQIDBAUGBwgJCg
sMDQ4PEBESEwQVFhcYGRobHB0eHyAhIiMEJSYnKCkqKywtLi8wMTIzBDU2Nzg5Ojs8PT4/
QDArBgkqhkiG9w0BBwGgHgQcVGhpcyBpcyBzb21lIHNhbXBsZSBjb250ZW50LqKBxzAYBg
kqhkiG9w0BCQMxCwYJKoZIhvcNAQcBMBwGCSqGSIb3DQEJBTEPFw0wOTEyMTAyMzI1MDBa
MB8GCSqGSIb3DQEJBDESBBCWaa5hG1eeg+oQK2tJ3cD5MGwGCSqGSIb3DQEJNDFfMF0wTw
YLKoZIhvcNAQkQAw0EQAECAwQFBgcICQoLDA0ODxAREhMEFRYXGBkaGxwdHh8gISIjBCUm
JygpKissLS4vMDEyMwQ1Njc4OTo7PD0+P0CiCgYIKwYBBQUIAQIEFLjUxQ9PJFzFnWraxb
EIbVbg2xql

|>ad.bin
|MIICRQYLKoZIhvcNAQkQAQKgggI0MIICMAIBADGBwDCBvQIBADAmMBIxEDAOBgNVBAMMB0
|NhcmxSU0ECEEY0a8eAAFa8EdNuLs1dcdAwDQYJKoZIhvcNAQEBBQAEgYCH70EpEikY7deb
|859YJRAWfFondQv1D4NFltw6C1ceheWnlAU0C2WEXr3LUBXZp1/PSte29FnJxu5bXCTn1g
|elMm6zNlZNWNd0KadVBcaxi1n8L52tVM5sWFGJPO5cStOyAka2ucuZM6iAnCSkn1Ju7fgU
|5j2g3bZ/IM8nHTcygjAKBggrBgEFBQgBAqFPBgsqhkiG9w0BCRADDQRAAQIDBAUGBwgJCg
|sMDQ4PEBESEwQVFhcYGRobHB0eHyAhIiMEJSYnKCkqKywtLi8wMTIzBDU2Nzg5Ojs8PT4/
|QDArBgkqhkiG9w0BBwGgHgQcVGhpcyBpcyBzb21lIHNhbXBsZSBjb250ZW50LqKBxzAYBg
|kqhkiG9w0BCQMxCwYJKoZIhvcNAQcBMBwGCSqGSIb3DQEJBTEPFw0wOTEyMTAyMzI1MDBa
|MB8GCSqGSIb3DQEJBDESBBCWaa5hG1eeg+oQK2tJ3cD5MGwGCSqGSIb3DQEJNDFfMF0wTw
|YLKoZIhvcNAQkQAw0EQAECAwQFBgcICQoLDA0ODxAREhMEFRYXGBkaGxwdHh8gISIjBCUm
|JygpKissLS4vMDEyMwQ1Njc4OTo7PD0+P0CiCgYIKwYBBQUIAQIEFLjUxQ9PJFzFnWraxb
|EIbVbg2xql
|<ad.bin
```

Appendix B. 2008 ASN.1 Module

The ASN.1 module defined uses the 2008 ASN.1 definitions found in [ASN.1-2008]. This module contains the ASN.1 module that contains the required definitions for the types and values defined in this document. The module uses the class defined in [CMS-ASN] and [RFC5912].

```
MD5-HASH-EXPERIMENT
  { iso(1) member-body(2) us(840) rsadsi(113549)
    pkcs(1) pkcs-9(9) smime(16) modules(0)
    id-mod-MD5-XOR-EXPERIMENT(999) }
DEFINITIONS IMPLICIT TAGS ::=
BEGIN
 IMPORTS

   -- Cryptographic Message Syntax (CMS) [CMS]

   DigestAlgorithmIdentifier, MessageAuthenticationCodeAlgorithm,
   SignatureAlgorithmIdentifier, DIGEST-ALGORITHM
   FROM  CryptographicMessageSyntax-2009
     { iso(1) member-body(2) us(840) rsadsi(113549)
       pkcs(1) pkcs-9(9) smime(16) modules(0) id-mod-cms-2004-02(41) }

   -- Common PKIX structures [RFC5912]

   ATTRIBUTE
   FROM PKIX-CommonTypes-2009
     { iso(1) identified-organization(3) dod(6) internet(1)
       security(5) mechanisms(5) pkix(7) id-mod(0)
       id-mod-pkixCommon-02(57)};

   mda-xor-md5-EXPERIMENT DIGEST-ALGORITHM ::= {
      IDENTIFIER id-alg-MD5-XOR-EXPERIMENT
      PARAMS TYPE MD5-XOR-EXPERIMENT ARE required
   }

   id-alg-MD5-XOR-EXPERIMENT OBJECT IDENTIFIER ::= {
      iso(1) member-body(2) us(840) rsadsi(113549)
      pkcs(1) pkcs-9(9) smime(16) id-alg(3) 13
   }

   MD5-XOR-EXPERIMENT ::= OCTET STRING (SIZE(64))

END
```

Author’s Address

Jim Schaad
Soaring Hawk Consulting

EMail: ietf@augustcellars.com