Internet Engineering Task Force (IETF)
Request for Comments: 6211
Category: Standards Track
ISSN: 2070-1721
J. Schaad
Soaring Hawk Consulting
April 2011

Cryptographic Message Syntax (CMS) Algorithm Identifier Protection Attribute

Abstract

The Cryptographic Message Syntax (CMS), unlike X.509/PKIX certificates, is vulnerable to algorithm substitution attacks. In an algorithm substitution attack, the attacker changes either the algorithm being used or the parameters of the algorithm in order to change the result of a signature verification process. In X.509 certificates, the signature algorithm is protected because it is duplicated in the TBSCertificate.signature field with the proviso that the validator is to compare both fields as part of the signature validation process. This document defines a new attribute that contains a copy of the relevant algorithm identifiers so that they are protected by the signature or authentication process.

Status of This Memo

This is an Internet Standards Track document.

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). Further information on Internet Standards is available in 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/rfc6211.

Copyright Notice

Table of Contents

1. Introduction

The Cryptographic Message Syntax [CMS], unlike X.509/PKIX certificates [RFC5280], is vulnerable to algorithm substitution attacks. In an algorithm substitution attack, the attacker changes either the algorithm being used or the parameters of the algorithm in order to change the result of a signature verification process. In X.509 certificates, the signature algorithm is protected because it is duplicated in the TBSCertificate.signature field with the proviso that the validator is to compare both fields as part of the signature validation process. This document defines a new attribute that contains a copy of the relevant algorithm identifiers so that they are protected by the signature or authentication process.

In an algorithm substitution attack, the attacker looks for a different algorithm that produces the same result as the algorithm used by the signer. As an example, if the creator of the message used SHA-1 as the digest algorithm to hash the message content, then the attacker looks for a different hash algorithm that produces a result that is of the same length, but with which it is easier to find collisions. Examples of other algorithms that produce a hash value of the same length would be SHA-0 or RIPEMD-160. Similar attacks can be mounted against parameterized algorithm identifiers. When looking at some of the proposed randomized hashing functions, such as that in [RANDOM-HASH], the associated security proofs assume that the parameters are solely under the control of the originator and not subject to selection by the attacker.

Some algorithms have been internally designed to be more resistant to this type of attack. Thus, an RSA PKCS #1 v.15 signature [RFC3447] cannot have the associated hash algorithm changed because it is encoded as part of the signature. The Digital Signature Algorithm (DSA) was originally defined so that it would only work with SHA-1 as a hash algorithm; thus, by knowing the public key from the certificate, a validator can be assured that the hash algorithm cannot be changed. There is a convention, undocumented as far as I can tell, that the same hash algorithm should be used for both the content digest and the signature digest. There are cases, such as third-party signers that are only given a content digest, where such a convention cannot be enforced.

As with all attacks, the attack is going to be desirable on items that are both long term and high value. One would expect that these attacks are more likely to be made on older documents, as the algorithms being used when the message was signed would be more likely to have degraded over time. Countersigning, the classic method of protecting a signature, does not provide any additional protection against an algorithm substitution attack because countersignatures sign just the signature, but the algorithm substitution attacks leave the signature value alone while changing the algorithms being used.

Using the SignerInfo structure from CMS, let's take a more detailed look at each of the fields in the structure and discuss what fields are and are not protected by the signature. I have included a copy of the ASN.1 here for convenience. A similar analysis of the AuthenticatedData structure is left to the reader, but it can be done in much the same way.

```
SignerInfo ::= SEQUENCE {
  version CMSVersion,
  sid SignerIdentifier,
  digestAlgorithm DigestAlgorithmIdentifier,
  signedAttrs [0] IMPLICIT SignedAttributes OPTIONAL,
  signatureAlgorithm SignatureAlgorithmIdentifier,
  signature SignatureValue,
  unsignedAttrs [1] IMPLICIT UnsignedAttributes OPTIONAL }
```

version is not protected by the signature. As many implementations of CMS today ignore the value of this field, that is not a problem. If the value is increased, then no changes in the processing are expected. If the value is decreased, implementations that respect the structure would fail to decode, but an erroneous signature validation would not be completed successfully.

sid can be protected using either version of the signing certificate authenticated attribute. SigningCertificateV2 is defined in [RFC5035]. SigningCertificate is defined in [ESS-BASE]. In addition to allowing for the protection of the signer identifier, the specific certificate is protected by including a hash of the certificate to be used for validation.

digestAlgorithm has been implicitly protected by the fact that CMS has only defined one digest algorithm for each hash value length. (The algorithm RIPEMD-160 was never standardized.) There is also an unwritten convention that the same digest algorithm should be used both here and for the signature algorithm. If newer digest algorithms are defined so that there are multiple algorithms for a given hash length (it is expected that the SHA-3 project will do so), or that parameters are defined for a specific algorithm, much of the implicit protection will be lost.

signedAttributes are directly protected by the signature when they are present. The Distinguished Encoding Rules (DER) encoding of this value is what is hashed for the signature computation.

signatureAlgorithm has been protected by implication in the past. The use of an RSA public key implied that the RSA v1.5 signature algorithm was being used. The hash algorithm and this fact could be checked by the internal padding defined. This is no longer true with the addition of the RSA-PSS signature algorithms. The use of a DSA public key implied the SHA-1 hash algorithm as that was the only possible hash algorithm and the DSA was the public signature algorithm. This is still somewhat true as there is an implicit tie between the length of the DSA public key and the length of the hash algorithm to be used, but this is known by convention and there is no explicit enforcement for this.

signature is not directly protected by any other value unless a counter signature is present. However, this represents the cryptographically computed value that protects the rest of the signature information.

unsignedAttrs is not protected by the signature value. The SignedData structure was explicitly designed that unsignedAttrs are not protected by the signature value.

As can be seen above, the digestAlgorithm and signatureAlgorithm fields have been indirectly rather than explicitly protected in the past. With new algorithms that have been or are being defined, this will no longer be the case. This document defines and describes a new attribute that will explicitly protect these fields along with the macAlgorithm field of the AuthenticatedData structure.

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. Attribute Structure

The following defines the algorithm protection attribute:

The algorithm protection attribute has the ASN.1 type CMSAlgorithmProtection:

```
aa-cmsAlgorithmProtection ATTRIBUTE ::= {
    TYPE CMSAlgorithmProtection
    IDENTIFIED BY { id-aa-CMSAlgorithmProtection }
}
```

The following object identifier identifies the algorithm protection attribute:

```
 id-aa-CMSAlgorithmProtection OBJECT IDENTIFIER ::= { iso(1)
      member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs9(9) 52 }
```

The algorithm protection attribute uses the following ASN.1 type:

```
CMSAlgorithmProtection ::= SEQUENCE {
    digestAlgorithm         DigestAlgorithmIdentifier,
    signatureAlgorithm  [1] SignatureAlgorithmIdentifier OPTIONAL,
    macAlgorithm        [2] MessageAuthenticationCodeAlgorithm
                                     OPTIONAL
}
(WITH COMPONENTS { signatureAlgorithm PRESENT,
                   macAlgorithm ABSENT } |
 WITH COMPONENTS { signatureAlgorithm ABSENT,
                   macAlgorithm PRESENT })
```

The fields are defined as follows:

digestAlgorithm contains a copy of the SignerInfo.digestAlgorithm field or the AuthenticatedData.digestAlgorithm field including any parameters associated with it.

signatureAlgorithm contains a copy of the signature algorithm identifier and any parameters associated with it (SignerInfo.signatureAlgorithm). This field is populated only if the attribute is placed in a SignerInfo.signedAttrs sequence.

macAlgorithm contains a copy of the message authentication code algorithm identifier and any parameters associated with it (AuthenticatedData.macAlgorithm). This field is populated only if the attribute is placed in an AuthenticatedData.authAttrs sequence.

Exactly one of signatureAlgorithm or macAlgorithm SHALL be present.

An algorithm protection attribute MUST have a single attribute value, even though the syntax is defined as a SET OF AttributeValue. There MUST NOT be zero or multiple instances of AttributeValue present.

The algorithm protection attribute MUST be a signed attribute or an authenticated attribute; it MUST NOT be an unsigned attribute, an unauthenticated attribute, or an unprotected attribute.

The SignedAttributes and AuthAttributes syntax are each defined as a SET of Attributes. The SignedAttributes in a signerInfo MUST include only one instance of the algorithm protection attribute. Similarly, the AuthAttributes in an AuthenticatedData MUST include only one instance of the algorithm protection attribute.

3. Verification Process

While the exact verification steps depend on the structure that is being validated, there are some common rules to be followed when comparing the two algorithm structures:

- A field with a default value MUST compare as identical, independently of whether the value is defaulted or is explicitly provided. This implies that a binary compare of the encoded bytes is insufficient.

- For some algorithms, such as SHA-1, the parameter value of NULL can be included in the ASN.1 encoding by some implementations and be omitted by other implementations. It is left to the implementer of this attribute to decide the comparison for equality is satisfied in this case. As a general rule, the same implementation is expected to produce both encoded values thus making it unlikely that this corner case should exist. This is an issue because some implementations will omit a NULL element, while others will encode a NULL element for some digest algorithms such as SHA-1 (see the comments in Section 2.1 of [RFC3370]). The issue is even worse because the NULL is absent in some cases (e.g., [RFC3370]), but is required in other cases (e.g., [RFC4056]).

3.1. Signed Data Verification Changes

If a CMS validator supports this attribute, the following additional verification steps MUST be performed:

1. The SignerInfo.digestAlgorithm field MUST be compared to the digestAlgorithm field in the attribute. If the fields are not the same (modulo encoding), then signature validation MUST fail.

2. The SignerInfo.signatureAlgorithm field MUST be compared to the signatureAlgorithm field in the attribute. If the fields are not the same (modulo encoding), then the signature validation MUST fail.

3.2. Authenticated Data Verification Changes

If a CMS validator supports this attribute, the following additional verification steps MUST be performed:

1. The AuthenticatedData.digestAlgorithm field MUST be compared to the digestAlgorithm field in the attribute. If the fields are not same (modulo encoding), then authentication MUST fail.

2. The AuthenticatedData.macAlgorithm field MUST be compared to the macAlgorithm field in the attribute. If the fields are not the same (modulo encoding), then the authentication MUST fail.

4. IANA Considerations

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

5. Security Considerations

This document is designed to address the security issue of algorithm substitutions of the algorithms used by the validator. At this time, there is no known method to exploit this type of attack. If the attack could be successful, then either a weaker algorithm could be substituted for a stronger algorithm or the parameters could be modified by an attacker to change the behavior of the hashing algorithm used. (One example would be changing the initial parameter value for [RFC6210].)

The attribute defined in this document is to be placed in a location that is protected by the signature or message authentication code. This attribute does not provide any additional security if placed in an unsigned or unauthenticated location.

6. References

6.1. 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.

[ESS-BASE] Hoffman, P., "Enhanced Security Services for S/MIME", RFC 2634, June 1999.

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

[RFC5035] Schaad, J., "Enhanced Security Services (ESS) Update: Adding CertID Algorithm Agility", RFC 5035, August 2007.

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

6.2. Informative References

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

[RFC3370] Housley, R., "Cryptographic Message Syntax (CMS) Algorithms", RFC 3370, August 2002.

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

[RFC4056] Schaad, J., "Use of the RSASSA-PSS Signature Algorithm in Cryptographic Message Syntax (CMS)", RFC 4056, June 2005.

[RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, May 2008.

[RFC6210] Schaad, J., "Experiment: Hash Functions with Parameters in the Cryptographic Message Syntax (CMS) and S/MIME", RFC 6210, April 2011.

Appendix A. 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 ATTRIBUTE class defined in [RFC5912].

```
CMSAlgorithmProtectionAttribute
  { iso(1) member-body(2) us(840) rsadsi(113549)
    pkcs(1) pkcs-9(9) smime(16) modules(0)
    id-mod-cms-algorithmProtect(52) }
DEFINITIONS IMPLICIT TAGS ::=
BEGIN
 IMPORTS

   -- Cryptographic Message Syntax (CMS) [CMS]

   DigestAlgorithmIdentifier, MessageAuthenticationCodeAlgorithm,
   SignatureAlgorithmIdentifier
   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)};

   --
   --  The CMS Algorithm Protection attribute is a Signed Attribute or
   --  an Authenticated Attribute.
   --
   --  Add this attribute to SignedAttributesSet in [CMS]
   --  Add this attribute to AuthAttributeSet in [CMS]
   --

   aa-cmsAlgorithmProtection ATTRIBUTE ::= {
      TYPE CMSAlgorithmProtection
      IDENTIFIED BY { id-aa-cmsAlgorithmProtect }
   }

   id-aa-cmsAlgorithmProtect OBJECT IDENTIFIER ::= {
      iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1)
      pkcs9(9) 52 }
```

```
     CMSAlgorithmProtection ::= SEQUENCE {
        digestAlgorithm         DigestAlgorithmIdentifier,
        signatureAlgorithm  [1] SignatureAlgorithmIdentifier OPTIONAL,
        macAlgorithm        [2] MessageAuthenticationCodeAlgorithm
                                          OPTIONAL
     }
     (WITH COMPONENTS { signatureAlgorithm PRESENT,
                        macAlgorithm ABSENT } |
      WITH COMPONENTS { signatureAlgorithm ABSENT,
                        macAlgorithm PRESENT })

  END
```

Author's Address

Jim Schaad
Soaring Hawk Consulting

EMail: ietf@augustcellars.com