INTERNET-DRAFT
Intended Status: Proposed Standard
Expires: 18 June 2018

R. Housley
Vigil Security

18 December 2017

Use of the Hash-based Merkle Tree Signature (MTS) Algorithm in the Cryptographic Message Syntax (CMS)

<draft-housley-cms-mts-hash-sig-08>

Abstract

This document specifies the conventions for using the Merkle Tree Signatures (MTS) digital signature algorithm with the Cryptographic Message Syntax (CMS). The MTS algorithm is one form of hash-based digital signature.

Status of this Memo

This Internet-Draft is submitted to IETF in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/1id-abstracts.html

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html

Copyright and License Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction 3
  1.1. ASN.1 3
  1.2. Terminology 3
2. MTS Digital Signature Algorithm Overview 3
  2.1. Hierarchical Signature System (HSS) 3
  2.2. Leighton-Micali Signature (LMS) 4
  2.3. Leighton-Micali One-time Signature Algorithm (LM-OTS) . . 5
3. Algorithm Identifiers and Parameters 6
4. Signed-data Conventions 6
5. Security Considerations 7
  5.1. Implementation Security Considerations 7
  5.2. Algorithm Security Considerations 7
6. IANA Considerations 8
7. Normative References 8
8. Informative References 8
Appendix: ASN.1 Module 10
Author's Address 11

1. Introduction

This document specifies the conventions for using the Merkle Tree Signatures (MTS) digital signature algorithm with the Cryptographic Message Syntax (CMS) [CMS] signed-data content type. The MTS algorithm is one form of hash-based digital signature that can only be used for a fixed number of signatures. The MTS algorithm is described in [HASHSIG]. The MTS algorithm uses small private and public keys, and it has low computational cost; however, the signatures are quite large.

1.1. ASN.1

CMS values are generated using ASN.1 [ASN1-B], using the Basic Encoding Rules (BER) and the Distinguished Encoding Rules (DER) [ASN1-E].

1.2. 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 RFC 2119 [KEYWORDS].

2. MTS Digital Signature Algorithm Overview

Merkle Tree Signatures (MTS) are a method for signing a large but fixed number of messages. An MTS system depends on a one-time signature method and a collision-resistant hash function.

This specification makes use of the MTS algorithm specified in [HASHSIG], which is the Leighton and Micali adaptation [LM] of the original Lamport-Diffie-Winternitz-Merkle one-time signature system [M1979][M1987][M1989a][M1989b]. It makes use of the LM-OTS one-time signature scheme and the SHA-256 one-way hash function [SHS].

2.1. Hierarchical Signature System (HSS)

The MTS system specified in [HASHSIG] uses a hierarchy of trees. The Hierarchical N-time Signature System (HSS) allows subordinate trees to be generated when needed by the signer. Otherwise, generation of the entire tree might take weeks or longer.

An HSS signature as specified in specified in [HASHSIG] carries the number of signed public keys (Nspk), followed by that number of signed public keys, followed by the LMS signature as described in Section 2.2. Each signed public key is represented by the hash value at the root of the tree, and the signature over that public key is an LMS signature as described in Section 2.2.

The elements of the HSS signature value for a stand-alone tree can be summarized as:

```
u32str(0) ||
lms_signature_on_message
```

The elements of the HSS signature value for a tree with Nspk levels can be summarized as:

```
u32str(Nspk) ||
lms_signature_on_public_key[0] || public_key[1] ||
lms_signature_on_public_key[1] || public_key[2] ||
   ...
lms_signature_on_public_key[Nspk-2] || public_key[Nspk-1] ||
lms_signature_on_public_key[Nspk-1] || public_key[Nspk] ||
lms_signature_on_message
```

2.2. Leighton-Micali Signature (LMS)

Each tree in the system specified in [HASHSIG] uses the Leighton-Micali Signature (LMS) system. LMS systems have two parameters. The first parameter is the height of the tree, h, which is the number of levels in the tree minus one. The [HASHSIG] specification supports five values for this parameter: h=5; h=10; h=15; h=20; and h=25. Note that there are 2^h leaves in the tree. The second parameter is the number of bytes output by the hash function, m, which the amount of data associated with each node in the tree. The [HASHSIG] specification supports only the SHA-256 hash function [SHS], with m=32.

Five tree sizes are specified in [HASHSIG]:

```
LMS_SHA256_M32_H5;
LMS_SHA256_M32_H10;
LMS_SHA256_M32_H15;
LMS_SHA256_M32_H20; and
LMS_SHA256_M32_H25.
```

An LMS signature consists of four elements: a typecode indicating the particular LMS algorithm, the number of the leaf associated with the LM-OTS signature, an LM-OTS signature as described in Section 2.3, and an array of values that is associated with the path through the tree from the leaf associated with the LM-OTS signature to the root. The array of values contains the siblings of the nodes on the path from the leaf to the root but does not contain the nodes on the path itself. The array for a tree with height h will have h values. The first value is the sibling of the leaf, the next value is the sibling of the parent of the leaf, and so on up the path to the root.

The four elements of the LMS signature value can be summarized as:

```
u32str(q) ||
ots_signature ||
u32str(type) ||
path[0] || path[1] || ... || path[h-1]
```

2.3. Leighton-Micali One-time Signature Algorithm (LM-OTS)

Merkle Tree Signatures (MTS) depend on a one-time signature method. [HASHSIG] specifies the use of the LM-OTS. An LM-OTS has five parameters.

n - The number of bytes associated with the hash function. [HASHSIG] supports only SHA-256 [SHS], with n=32.

H - A preimage-resistant hash function that accepts byte strings of any length, and returns an n-byte string.

w - The width in bits of the Winternitz coefficients. [HASHSIG] supports four values for this parameter: w=1; w=2; w=4; and w=8.

p - The number of n-byte string elements that make up the LM-OTS signature.

ls - The number of left-shift bits used in the checksum function, which is defined in Section 4.5 of [HASHSIG].

The values of p and ls are dependent on the choices of the parameters n and w, as described in Appendix A of [HASHSIG].

Four LM-OTS variants are defined in [HASHSIG]:

```
LMOTS_SHA256_N32_W1;
LMOTS_SHA256_N32_W2;
LMOTS_SHA256_N32_W4; and
LMOTS_SHA256_N32_W8.
```

Signing involves the generation of C, an n-byte random value.

The LM-OTS signature value can be summarized as:

```
u32str(type) || C || y[0] || ... || y[p-1]
```

3. Algorithm Identifiers and Parameters

The algorithm identifier for an MTS signature is id-alg-mts-hashsig:

```
id-alg-mts-hashsig  OBJECT IDENTIFIER ::= { iso(1) member-body(2)
      us(840) rsadsi(113549) pkcs(1) pkcs9(9) smime(16) alg(3) 17 }
```

When the id-alg-mts-hashsig algorithm identifier is used for a signature, the AlgorithmIdentifier parameters field MUST be absent (that is, the parameters are not present; the parameters are not set to NULL).

The signature values is a large OCTET STRING. The signature format is designed for easy parsing. Each format includes a counter and type codes that indirectly providing all of the information that is needed to parse the value during signature validation. The first 4 octets of the signature value contains the number of signed public keys (Nspk) in the HSS. The first 4 octets of each LMS signature value contains type code, which tells how to parse the remaining parts of the signature value. The first 4 octets of each LM-OTS signature value contains type code, which tells how to parse the remaining parts of the signature value.

4. Signed-data Conventions

As specified in [CMS], the digital signature is produced from the message digest and the signer's private key. If signed attributes are absent, then the message digest is the hash of the content. If signed attributes are present, then the hash of the content is placed in the message-digest attribute, the set of signed attributes is DER encoded, and the message digest is the hash of the encoded attributes. In summary:

```
IF (signed attributes are absent)
THEN md = Hash(content)
ELSE message-digest attribute = Hash(content);
     md = Hash(DER(SignedAttributes))

Sign(md)
```

When using [HASHSIG], the fields in the SignerInfo are used as follows:

digestAlgorithms SHOULD contain the one-way hash function used to compute the message digest on the eContent value. Since the hash-based signature algorithms all depend on SHA-256, it is strongly RECOMMENDED that SHA-256 also be used to compute the message digest on the content.

Further, the same one-way hash function SHOULD be used to compute the message digest on both the eContent and the signedAttributes value if signedAttributes are present. Again, since the hash-based signature algorithms all depend on SHA-256, it is strongly RECOMMENDED that SHA-256 be used.

signatureAlgorithm MUST contain id-alg-mts-hashsig. The algorithm parameters field MUST be absent.

signature contains the single HSS signature value resulting from the signing operation as specified in [HASHSIG].

5. Security Considerations

5.1. Implementation Security Considerations

Implementations must protect the private keys. Compromise of the private keys may result in the ability to forge signatures. Along with the private key, the implementation must keep track of which leaf nodes in the tree have been used. Loss of integrity of this tracking data can cause an one-time key to be used more than once. As a result, when a private key and the tracking data are stored on non-volatile media or stored in a virtual machine environment, care must be taken to preserve confidentiality and integrity.

An implementation must ensure that a LM-OTS private key is used to generate a signature only one time, and ensure that it cannot be used for any other purpose.

The generation of private keys relies on random numbers. The use of inadequate pseudo-random number generators (PRNGs) to generate these values can result in little or no security. An attacker may find it much easier to reproduce the PRNG environment that produced the keys, searching the resulting small set of possibilities, rather than brute force searching the whole key space. The generation of quality random numbers is difficult. RFC 4086 [RANDOM] offers important guidance in this area.

When computing signatures, the same hash function SHOULD be used for all operations. In this specification, only SHA-256 is used. Using only SHA-256 reduces the number of possible failure points in the signature process.

5.2. Algorithm Security Considerations

At Black Hat USA 2013, some researchers gave a presentation on the current sate of public key cryptography. They said: "Current cryptosystems depend on discrete logarithm and factoring which has

seen some major new developments in the past 6 months" [BH2013]. They encouraged preparation for a day when RSA and DSA cannot be depended upon.

A post-quantum cryptosystem is a system that is secure against quantum computers that have more than a trivial number of quantum bits. It is open to conjecture when it will be feasible to build such a machine. RSA, DSA, and ECDSA are not post-quantum secure.

The LM-OTP one-time signature, LMS, and HSS do not depend on discrete logarithm or factoring, as a result these algorithms are considered to be post-quantum secure.

Today, RSA is often used to digitally sign software updates. This means that the distribution of software updates could be compromised if a significant advance is made in factoring or a quantum computer is invented. The use of MTS signatures to protect software update distribution, perhaps using the format described in [FWPROT], will allow the deployment of software that implements new cryptosystems.

6. IANA Considerations

This document has no actions for IANA.

7. Acknowledgements

Many thanks to Panos Kampanakis, Jim Schaad, and Sean Turner for their careful review and comments.

8. Normative References

[ASN1-B] ITU-T, "Information technology -- Abstract Syntax Notation One (ASN.1): Specification of basic notation", ITU-T Recommendation X.680, 2015.

[ASN1-E] ITU-T, "Information technology -- ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)", ITU-T Recommendation X.690, 2015.

[CMS] Housley, R., "Cryptographic Message Syntax (CMS)", STD 70, RFC 5652, DOI 10.17487/RFC5652, September 2009, <http://www.rfc-editor.org/info/rfc5652>.

[HASHSIG] McGrew, D., M. Curcio, and S. Fluhrer, "Hash-Based Signatures", Work in progress. <draft-mcgrew-hash-sigs-07>

[KEYWORDS] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[SHS] National Institute of Standards and Technology (NIST), FIPS Publication 180-3: Secure Hash Standard, October 2008.

9. Informative References

[BH2013] Ptacek, T., T. Ritter, J. Samuel, and A. Stamos, "The Factoring Dead: Preparing for the Cryptopocalypse", August 2013. <https://media.blackhat.com/us-13/us-13-Stamos-The-Factoring-Dead.pdf>

[CMSASN1] Hoffman, P. and J. Schaad, "New ASN.1 Modules for Cryptographic Message Syntax (CMS) and S/MIME", RFC 5911, DOI 10.17487/RFC5911, June 2010, <http://www.rfc-editor.org/info/rfc5911>.

[CMSASN1U] Schaad, J. and S. Turner, "Additional New ASN.1 Modules for the Cryptographic Message Syntax (CMS) and the Public Key Infrastructure Using X.509 (PKIX)", RFC 6268, DOI 10.17487/RFC6268, July 2011, <http://www.rfc-editor.org/info/rfc6268>.

[FWPROT] Housley, R., "Using Cryptographic Message Syntax (CMS) to Protect Firmware Packages", RFC 4108, DOI 10.17487/RFC4108, August 2005, <http://www.rfc-editor.org/info/rfc4108>.

[LM] Leighton, T. and S. Micali, "Large provably fast and secure digital signature schemes from secure hash functions", U.S. Patent 5,432,852, July 1995.

[M1979] Merkle, R., "Secrecy, Authentication, and Public Key Systems", Stanford University Information Systems Laboratory Technical Report 1979-1, 1979.

[M1987] Merkle, R., "A Digital Signature Based on a Conventional Encryption Function", Lecture Notes in Computer Science crypto87, 1988.

[M1989a] Merkle, R., "A Certified Digital Signature", Lecture Notes in Computer Science crypto89, 1990.

[M1989b] Merkle, R., "One Way Hash Functions and DES", Lecture Notes in Computer Science crypto89, 1990.

[PKIXASN1] Hoffman, P. and J. Schaad, "New ASN.1 Modules for the Public Key Infrastructure Using X.509 (PKIX)", RFC 5912, DOI 10.17487/RFC5912, June 2010, <http://www.rfc-editor.org/info/rfc5912>.

[PQC] Bernstein, D., "Introduction to post-quantum cryptography", 2009. <http://www.pqcrypto.org/www.springer.com/cda/content/document/cda_downloaddocument/9783540887010-c1.pdf>

[RANDOM] Eastlake 3rd, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, DOI 10.17487/RFC4086, June 2005, <http://www.rfc-editor.org/info/rfc4086>.

Appendix: ASN.1 Module

```
   MTS-HashSig-2013
     { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs9(9)
       id-smime(16) id-mod(0) id-mod-mts-hashsig-2013(64) }

   DEFINITIONS IMPLICIT TAGS ::= BEGIN

   EXPORTS ALL;

   IMPORTS
     PUBLIC-KEY, SIGNATURE-ALGORITHM, SMIME-CAPS
       FROM AlgorithmInformation-2009  -- RFC 5911 [CMSASN1]
         { iso(1) identified-organization(3) dod(6) internet(1)
           security(5) mechanisms(5) pkix(7) id-mod(0)
           id-mod-algorithmInformation-02(58) }
     mda-sha256
       FROM PKIX1-PSS-OAEP-Algorithms-2009  -- RFC 5912 [PKIXASN1]
         { iso(1) identified-organization(3) dod(6)
           internet(1) security(5) mechanisms(5) pkix(7) id-mod(0)
           id-mod-pkix1-rsa-pkalgs-02(54) } ;

   --
   -- Object Identifiers
   --

   id-alg-mts-hashsig  OBJECT IDENTIFIER ::= { iso(1) member-body(2)
         us(840) rsadsi(113549) pkcs(1) pkcs9(9) smime(16) alg(3) 17 }

   --
   -- Signature Algorithm and Public Key
   --

   sa-MTS-HashSig SIGNATURE-ALGORITHM ::= {
        IDENTIFIER id-alg-mts-hashsig
        PARAMS ARE absent
        HASHES { mda-sha256 }
        PUBLIC-KEYS { pk-MTS-HashSig }
        SMIME-CAPS { IDENTIFIED BY id-alg-mts-hashsig } }

   pk-MTS-HashSig PUBLIC-KEY ::= {
       IDENTIFIER id-alg-mts-hashsig
       KEY MTS-HashSig-PublicKey
       PARAMS ARE absent
       CERT-KEY-USAGE
           { digitalSignature, nonRepudiation, keyCertSign, cRLSign } }

   MTS-HashSig-PublicKey ::= OCTET STRING
```

```
   --
   -- Expand the signature algorithm set used by CMS [CMSASN1U]
   --

   SignatureAlgorithmSet SIGNATURE-ALGORITHM ::=
        { sa-MTS-HashSig, ... }

   --
   -- Expand the S/MIME capabilities set used by CMS [CMSASN1]
   --

   SMimeCaps SMIME-CAPS ::= { sa-MTS-HashSig.&smimeCaps, ... }

   END
```

Author's Address

Russ Housley
Vigil Security, LLC
918 Spring Knoll Drive
Herndon, VA 20170
USA

EMail: housley@vigilsec.com