INTERNET-DRAFT R. Housley
Intended Status: Proposed Standard Vigil Security
Expires: 21 September 2016 21 March 2016

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

<draft-housley-cms-mts-hash-sig-04>

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

Table of Contents

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.

CMS values are generated using ASN.1 [ASN1-02], using the Basic Encoding Rules (BER) and the Distinguished Encoding Rules (DER).

1.1. MTS Digital Signature Algorithm

Merkle Tree Signatures (MTS) are a method for signing a large but fixed number of messages. An MTS system is an N-time signature system, meaning that the private key can be used to generate at most N signatures.

An MTS system uses two cryptographic components: a one-time signature method and a collision-resistant hash function. Each MTS public/private key pair is associated with a k-way tree. Each leaf of the tree can be used to generate a one-time signature (OTS), which can be used to securely sign exactly one message, but cannot securely sign more than one.

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 [SHS] one-way hash function.

An LMS system has two parameters. The height of the tree, h, which is the number of levels in the tree minus one. The [HASHSIG] specification supports three values for this parameter: h=20; h=10; and h=5. The number of bytes associated with each node in the tree, n, is defined by the hash function. The [HASHSIG] specification supports two hash functions: SHA-256 [SHS], with n=32; and SHA-256-16, which is the same as SHA-256, except that the hash result is truncated to 16 bytes, with n=16. Note that there are 2^h leaves in the tree.

Six tree sizes are specified in [HASHSIG]:
lms_sha256_n32_h20;
lms_sha256_n32_h10;
lms_sha256_n32_h5;
lms_sha256_n16_h20;
lms_sha256_n16_h10; and
lms_sha256_n16_h5.

An LMS signature consists of three things: a typecode indicating the particular LMS algorithm, an LM-OTS signature, 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.

1.2. LM-OTS One-time Signature Algorithm

Merkle Tree Signatures (MTS) depend on a LM-OTS one-time signature method. An LM-OTS has four parameters.

n - The number of bytes associated with the hash function, which is the same as the LMS parameter. The [HASHSIG] specification supports two hash functions: SHA-256 [SHS], with n=32; and SHA-256-16, with n=16.

w - The the Winternitz parameter. The [HASHSIG] specification 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.

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

Eight LM-OTS variants are defined in [HASHSIG]:
LMOTS_SHA256_N32_W1;
LMOTS_SHA256_N32_W2;
LMOTS_SHA256_N32_W4;
LMOTS_SHA256_N32_W8;
LMOTS_SHA256_N16_W1;
LMOTS_SHA256_N16_W2;
LMOTS_SHA256_N16_W4; and

LMOTS_SHA256_N16_W8.

1.3. 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. Algorithm Identifiers and Parameters

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

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

id-alg  OBJECT IDENTIFIER ::= { id-smime  3 }

id-alg-mts-hashsig  OBJECT IDENTIFIER ::= { id-alg 17 }
```

When the id-alg-mts-hashsig algorithm identifier is used for a signature, the AlgorithmIdentifier parameters field MUST be absent.

The first 4 bytes of the signature value contains the mls_algorithm_type as defined in Section 5.5 of [HASHSIG]. This type tells how to parse the remaining parts of the signature value, which is composed of an LM-OTS signature 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 first 4 bytes of the LM-OTS signature value contains the ots_algorithm_type as defined in Section 4.10 of [HASHSIG]. This type is followed by n*p bytes of signature value.

The signature format is designed for easy parsing. Each format starts with a 4-byte enumeration value that indicates all of the details of the signature algorithm, indirectly providing all of the information that is needed to parse the value during signature validation.

3. Signed-data Conventions

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 exist. 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 value resulting from the signing operation as specified in [HASHSIG].

4. Security Considerations

4.1. 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 maintain a counter value that indicates which leaf nodes in the tree have been used. Loss of integrity of this counter can cause an one-time key to be used more than once. As a result, when a private key and an associated counter value are stored on non-volatile media or stored in a virtual machine environment, care must be taken to preserve these properties.

An implementation must ensure that a LDWM private key is used only one time, and ensure that the LDWM private key 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. This reduces the number of failure points in the signature process.

4.2. 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 whether it is feasible to build such a machine. RSA, DSA, and ECDSA are not post-quantum secure.

The LM-OTP one-time signature and LMS do not depend on discrete logarithm or factoring, and 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.

5. IANA Considerations

{{ RFC Editor: Please remove this section prior to publication. }}

This document has no actions for IANA.

6. Normative References

[ASN1-02] ITU-T, "ITU-T Recommendation X.680, X.681, X.682, and X.683", ITU-T X.680, X.681, X.682, and X.683, 2002.

[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., and M. Curcio, "Hash-Based Signatures", Work in progress. <draft-mcgrew-hash-sigs-03>

[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.

7. 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>.

[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 EXPLICIT TAGS ::= BEGIN

   EXPORTS ALL;

   IMPORTS
     SIGNATURE-ALGORITHM PUBLIC-KEY
       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-smime  OBJECT IDENTIFIER ::= { iso(1) member-body(2)
               us(840) rsadsi(113549) pkcs(1) pkcs9(9) 16 }

   id-alg  OBJECT IDENTIFIER ::= { id-smime  3 }

   id-alg-mts-hashsig  OBJECT IDENTIFIER ::= { id-alg 17 }


   --
   -- Signature Algorithm and Public Key
   --

   sa-MTS-HashSig SIGNATURE-ALGORITHM ::= {
        IDENTIFIER id-alg-mts-hashsig
        HASHES { mda-sha256, ... }
        PUBLIC-KEYS { pk-MTS-HashSig } }

   pk-MTS-HashSig PUBLIC-KEY ::= {
       IDENTIFIER id-alg-mts-hashsig
       KEY MTS-HashSig-PublicKey }
```

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

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

   END
```

Author's Address

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

EMail: housley@vigilsec.com