Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 12, 2019

D. Van Geest
ISARA Corporation
S. Fluhrer
Cisco Systems

March 11, 2019

Algorithm Identifiers for HSS and XMSS for Use in the Internet X.509 Public Key Infrastructure

draft-vangeest-x509-hash-sigs-02

Abstract

This document specifies algorithm identifiers and ASN.1 encoding formats for the Hierarchical Signature System (HSS), eXtended Merkle Signature Scheme (XMSS), and XMSS^MT, a multi-tree variant of XMSS. This specification applies to the Internet X.509 Public Key infrastructure (PKI) when digital signatures are used to sign certificates and certificate revocation lists (CRLs).

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

The Hierarchical Signature System (HSS) is described in [I-D.mcgrew-hash-sigs].

The eXtended Merkle Signature Scheme (XMSS), and its multi-tree variant XMSS^MT, are described in [RFC8391].

These signature algorithms are based on well-studied Hash Based Signature (HBS) schemes, which can withstand known attacks using quantum computers. They combine Merkle Trees with One Time Signature (OTS) schemes in order to create signature systems which can sign a large but limited number of messages per private key. The private keys are stateful; a key's state must be updated and persisted after signing to prevent reuse of OTS keys. If an OTS key is reused, cryptographic security is not guaranteed for that key.

Due to the statefulness of the private key and the limited number of signatures that can be created, these signature algorithms might not be appropriate for use in interactive protocols. While the right selection of algorithm parameters would allow a private key to sign a

virtually unbounded number of messages (e.g. 2^60), this is at the cost of a larger signature size and longer signing time. Since these algorithms are already known to be secure against quantum attacks, and because roots of trust are generally long-lived and can take longer to be deployed than end-entity certificates, these signature algorithms are more appropriate to be used in root and subordinate CA certificates. They are also appropriate in non-interactive contexts such as code signing. In particular, there are multi-party IoT ecosystems where publicly trusted code signing certificates are useful.

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. Subject Public Key Algorithms

Certificates conforming to [RFC5280] can convey a public key for any public key algorithm. The certificate indicates the algorithm through an algorithm identifier. An algorithm identifier consists of an OID and optional parameters.

In this document, we define new OIDs for identifying the different hash-based signature algorithms. An additional OID is defined in [I-D.ietf-lamps-cms-hash-sig] and repeated here for convenience. For all of the OIDs, the parameters MUST be absent.

2.1. HSS Public Keys

The object identifier and public key algorithm identifier for HSS is defined in [I-D.ietf-lamps-cms-hash-sig]. The definitions are repeated here for reference.

The object identifier for an HSS public key is id-alg-hss-lms-hashsig:

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

Note that the id-alg-hss-lms-hashsig algorithm identifier is also referred to as id-alg-mts-hashsig. This synonym is based on the terminology used in an early draft of the document that became [I-D.ietf-lamps-cms-hash-sig].

The HSS public key's properties are defined as follows:


```
pk-HSS-LMS-HashSig PUBLIC-KEY ::= {
   IDENTIFIER id-alg-hss-lms-hashsig
   KEY HSS-LMS-HashSig-PublicKey
   PARAMS ARE absent
   CERT-KEY-USAGE
      { digitalSignature, nonRepudiation, keyCertSign, cRLSign } }

HSS-LMS-HashSig-PublicKey ::= OCTET STRING
```

[I-D.ietf-lamps-cms-hash-sig] contains more information on the contents and format of an HSS public key.

2.2. XMSS Public Keys

The object identifier for an XMSS public key is id-alg-xmss:

```
id-alg-xmss  OBJECT IDENTIFIER ::= { itu-t(0)
   identified-organization(4) etsi(0) reserved(127)
   etsi-identified-organization(0) isara(15) algorithms(1)
   asymmetric(1) xmss(13) 0 }
```

The XMSS public key's properties are defined as follows:

```
pk-XMSS PUBLIC-KEY ::= {
   IDENTIFIER id-alg-xmss
   KEY XMSS-PublicKey
   PARAMS ARE absent
   CERT-KEY-USAGE
      { digitalSignature, nonRepudiation, keyCertSign, cRLSign } }
```

```
XMSS-PublicKey ::= OCTET STRING
```

The format of an XMSS public key is is formally defined using XDR [RFC4506] and is defined in Appendix B.3 of [RFC8391]. In particular, the first 4 bytes represents the big-ending encoding of the XMSS algorithm type.

2.3. XMSS^MT Public Keys

The object identifier for an XMSS^MT public key is id-alg-xmssmt:

```
id-alg-xmssmt  OBJECT IDENTIFIER ::= { itu-t(0)
   identified-organization(4) etsi(0) reserved(127)
   etsi-identified-organization(0) isara(15) algorithms(1)
   asymmetric(1) xmssmt(14) 0 }
```

The XMSS^MT public key's properties are defined as follows:


```
pk-XMSSMT PUBLIC-KEY ::= {
   IDENTIFIER id-alg-xmssmt
   KEY XMSSMT-PublicKey
   PARAMS ARE absent
   CERT-KEY-USAGE
      { digitalSignature, nonRepudiation, keyCertSign, cRLSign } }

XMSSMT-PublicKey ::= OCTET STRING
```

The format of an XMSS^MT public key is is formally defined using XDR [RFC4506] and is defined in Appendix C.3 of [RFC8391]. In particular, the first 4 bytes represents the big-ending encoding of the XMSS^MT algorithm type.

3. Key Usage Bits

The intended application for the key is indicated in the keyUsage certificate extension.

If the keyUsage extension is present in an end-entity certificate that indicates id-alg-xmss or id-alg-xmssmt in SubjectPublicKeyInfo, then the keyUsage extension MUST contain one or both of the following

values:

   nonRepudiation; and
   digitalSignature.

If the keyUsage extension is present in a certification authority certificate that indicates id-alg-xmss or id-alg-xmssmt, then the keyUsage extension MUST contain one or more of the following values:

   nonRepudiation;
   digitalSignature;
   keyCertSign; and
   cRLSign.

[I-D.ietf-lamps-cms-hash-sig] defines the key usage for id-alg-hss-lms-hashsig, which is the same as for the keys above.

4. Signature Algorithms

This section identifies OIDs for signing using HSS, XMSS, and XMSS^MT. When these algorithm identifiers appear in the algorithm field as an AlgorithmIdentifier, the encoding MUST omit the parameters field. That is, the AlgorithmIdentifier SHALL be a SEQUENCE of one component, one of the OIDs defined below.

The data to be signed is prepared for signing. With the OIDs id-alg-hss-lms-hashsig, id-alg-xmss and id-alg-xmssmt the full data is signed directly. With the other OIDs defined in this document, an appropriate hash function is applied first and the resulting digest is signed. Then, a private key operation is performed to generate the signature value. For HSS, the signature value is described in section 3.3 of [I-D.mcgrew-hash-sigs]. For XMSS and XMSS^MT the signature values are described in sections B.2 and C.2 of [RFC8391] respectively. The octet string representing the signature is encoded directly in the BIT STRING without adding any additional ASN.1 wrapping. For the Certificate and CertificateList structures, the signature value is wrapped in the "signatureValue" BIT STRING field.

4.1. HSS Signature Algorithm

The HSS public key OID is also used to specify that an HSS signature was generated on the full message, i.e. the message was not hashed before being processed by the HSS signature algorithm.

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

The ASN.1 OIDs used to specify that an HSS signature was generated on a SHA-256, SHA-384 or SHA-512 hash of an object are, respectively:

```
id-alg-hss-with-SHA256  OBJECT IDENTIFIER ::= { itu-t(0)
   identified-organization(4) etsi(0) reserved(127)
   etsi-identified-organization(0) isara(15) algorithms(1)
   asymmetric(1) hss(12) 2 }

id-alg-hss-with-SHA384  OBJECT IDENTIFIER ::= { itu-t(0)
   identified-organization(4) etsi(0) reserved(127)
   etsi-identified-organization(0) isara(15) algorithms(1)
   asymmetric(1) hss(12) 3 }

id-alg-hss-with-SHA512  OBJECT IDENTIFIER ::= { itu-t(0)
   identified-organization(4) etsi(0) reserved(127)
   etsi-identified-organization(0) isara(15) algorithms(1)
   asymmetric(1) hss(12) 1 }
```

[I-D.ietf-lamps-cms-hash-sig] contains more information on the contents and format of an HSS signature.

4.2. XMSS Signature Algorithm

The XMSS public key OID is also used to specify that an XMSS signature was generated on the full message, i.e. the message was not hashed before being processed by the XMSS signature algorithm.

```
id-alg-xmss  OBJECT IDENTIFIER ::= { itu-t(0)
   identified-organization(4) etsi(0) reserved(127)
```

```
      etsi-identified-organization(0) isara(15) algorithms(1)
      asymmetric(1) xmss(13) 0 }
```

The ASN.1 OIDs used to specify that an XMSS signature was generated on a SHA-256, SHA-384 or SHA-512 hash of an object are, respectively:

```
   id-alg-xmss-with-SHA256  OBJECT IDENTIFIER ::= { itu-t(0)
      identified-organization(4) etsi(0) reserved(127)
      etsi-identified-organization(0) isara(15) algorithms(1)
      asymmetric(1) xmss(13) 2 }

   id-alg-xmss-with-SHA384  OBJECT IDENTIFIER ::= { itu-t(0)
      identified-organization(4) etsi(0) reserved(127)
      etsi-identified-organization(0) isara(15) algorithms(1)
      asymmetric(1) xmss(13) 3 }

   id-alg-xmss-with-SHA512  OBJECT IDENTIFIER ::= { itu-t(0)
      identified-organization(4) etsi(0) reserved(127)
      etsi-identified-organization(0) isara(15) algorithms(1)
      asymmetric(1) xmss(13) 1 }
```

The format of an XMSS signature is is formally defined using XDR [RFC4506] and is defined in Appendix B.2 of [RFC8391].

4.3. XMSS^MT Signature Algorithm

The XMSS^MT public key OID is also used to specify that an XMSS^MT signature was generated on the full message, i.e. the message was not hashed before being processed by the XMSS^MT signature algorithm.

```
   id-alg-xmssmt  OBJECT IDENTIFIER ::= { itu-t(0)
      identified-organization(4) etsi(0) reserved(127)
      etsi-identified-organization(0) isara(15) algorithms(1)
      asymmetric(1) xmssmt(14) 0 }
```

The ASN.1 OIDs used to specify that an XMSS^MT signature was generated on a SHA-256, SHA384 or SHA-512 hash of an object are, respectively:


```
   id-alg-xmssmt-with-SHA256  OBJECT IDENTIFIER ::= { itu-t(0)
```

```
      identified-organization(4) etsi(0) reserved(127)
      etsi-identified-organization(0) isara(15) algorithms(1)
      asymmetric(1) xmssmt(14) 2 }

   id-alg-xmssmt-with-SHA384  OBJECT IDENTIFIER ::= { itu-t(0)
      identified-organization(4) etsi(0) reserved(127)
      etsi-identified-organization(0) isara(15) algorithms(1)
      asymmetric(1) xmssmt(14) 3 }

   id-alg-xmssmt-with-SHA512  OBJECT IDENTIFIER ::= { itu-t(0)
      identified-organization(4) etsi(0) reserved(127)
      etsi-identified-organization(0) isara(15) algorithms(1)
      asymmetric(1) xmssmt(14) 1 }
```

The format of an XMSS^MT signature is is formally defined using XDR [RFC4506] and is defined in Appendix C.2 of [RFC8391].

5. ASN.1 Module

For reference purposes, the ASN.1 syntax is presented as an ASN.1 module here.

```
-- ASN.1 Module

Hashsigs-pkix-0 -- TBD - IANA assigned module OID

DEFINITIONS EXPLICIT TAGS ::=
BEGIN

IMPORTS
  PUBLIC-KEY, SIGNATURE-ALGORITHM
  FROM AlgorithmInformation-2009
    {iso(1) identified-organization(3) dod(6) internet(1) security(5)
    mechanisms(5) pkix(7) id-mod(0)
    id-mod-algorithmInformation-02(58)}
;
```

```
   --
   -- HSS Signatures
   --

   -- HSS Object Identifiers

   --
   -- id-alg-hss-lms-hashsig is defined in [ietf-lamps-cms-hash-sig]
   --
   -- id-alg-hss-lms-hashsig OBJECT IDENTIFIER ::= { iso(1)
   --     member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs9(9)
   --     smime(16) alg(3) 17 }

   id-alg-hss-with-SHA256  OBJECT IDENTIFIER ::= { itu-t(0)
      identified-organization(4) etsi(0) reserved(127)
      etsi-identified-organization(0) isara(15) algorithms(1)
      asymmetric(1) hss(12) 2 }

   id-alg-hss-with-SHA384  OBJECT IDENTIFIER ::= { itu-t(0)
      identified-organization(4) etsi(0) reserved(127)
      etsi-identified-organization(0) isara(15) algorithms(1)
      asymmetric(1) hss(12) 3 }

   id-alg-hss-with-SHA512  OBJECT IDENTIFIER ::= { itu-t(0)
      identified-organization(4) etsi(0) reserved(127)
      etsi-identified-organization(0) isara(15) algorithms(1)
      asymmetric(1) hss(12) 1 }

   -- HSS Signature Algorithms and Public Key

   --
   -- sa-HSS-LMS-HashSig is defined in [ietf-lamps-cms-hash-sig]
   --
   -- sa-HSS-LMS-HashSig SIGNATURE-ALGORITHM ::= {
   --     IDENTIFIER id-alg-hss-lms-hashsig
   --     PARAMS ARE absent
   --     PUBLIC-KEYS { pk-HSS-LMS-HashSig }
   --     SMIME-CAPS { IDENTIFIED BY id-alg-hss-lms-hashsig } }

   sa-HSS-with-SHA256 SIGNATURE-ALGORITHM ::= {
      IDENTIFIER id-alg-hss-with-SHA256
      PARAMS ARE absent
      HASHES { mda-sha256 }
      PUBLIC-KEYS { pk-HSS-LMS-HashSig }
      SMIME-CAPS { IDENTIFIED BY id-alg-hss-with-SHA256 } }
```

```
   sa-HSS-with-SHA384 SIGNATURE-ALGORITHM ::= {
      IDENTIFIER id-alg-hss-with-SHA384
      PARAMS ARE absent
      HASHES { mda-sha384 }
      PUBLIC-KEYS { pk-HSS-LMS-HashSig }
      SMIME-CAPS { IDENTIFIED BY id-alg-hss-with-SHA384 } }

   sa-HSS-with-SHA512 SIGNATURE-ALGORITHM ::= {
      IDENTIFIER id-alg-hss-with-SHA512
      PARAMS ARE absent
      HASHES { mda-sha512 }
      PUBLIC-KEYS { pk-HSS-LMS-HashSig }
      SMIME-CAPS { IDENTIFIED BY id-alg-hss-with-SHA512 } }

   --
   -- pk-HSS-LMS-HashSig is defined in [ietf-lamps-cms-hash-sig]
   --
   -- pk-HSS-LMS-HashSig PUBLIC-KEY ::= {
   --    IDENTIFIER id-alg-hss-lms-hashsig
   --    KEY HSS-LMS-HashSig-PublicKey
   --    PARAMS ARE absent
   --    CERT-KEY-USAGE
   --       { digitalSignature, nonRepudiation, keyCertSign, cRLSign } }
   --
   -- HSS-LMS-HashSig-PublicKey ::= OCTET STRING

   --
   -- XMSS Keys and Signatures
   --

   -- XMSS Object Identifiers

   id-alg-xmss  OBJECT IDENTIFIER ::= { itu-t(0)
      identified-organization(4) etsi(0) reserved(127)
      etsi-identified-organization(0) isara(15) algorithms(1)
      asymmetric(1) xmss(13) 0 }

   id-alg-xmss-with-SHA256  OBJECT IDENTIFIER ::= { itu-t(0)
      identified-organization(4) etsi(0) reserved(127)
```

```
      etsi-identified-organization(0) isara(15) algorithms(1)
      asymmetric(1) xmss(13) 2 }

   id-alg-xmss-with-SHA384  OBJECT IDENTIFIER ::= { itu-t(0)
      identified-organization(4) etsi(0) reserved(127)
      etsi-identified-organization(0) isara(15) algorithms(1)
      asymmetric(1) xmss(13) 3 }
```



```
   id-alg-xmss-with-SHA512  OBJECT IDENTIFIER ::= { itu-t(0)
      identified-organization(4) etsi(0) reserved(127)
      etsi-identified-organization(0) isara(15) algorithms(1)
      asymmetric(1) xmss(13) 1 }

   -- XMSS Signature Algorithms and Public Key

   sa-XMSS SIGNATURE-ALGORITHM ::= {
       IDENTIFIER id-alg-xmss
       PARAMS ARE absent
       PUBLIC-KEYS { pk-XMSS }
       SMIME-CAPS { IDENTIFIED BY id-alg-xmss } }

   sa-XMSS-with-SHA256 SIGNATURE-ALGORITHM ::= {
      IDENTIFIER id-alg-xmss-with-SHA256
      PARAMS ARE absent
      HASHES { mda-sha256 }
      PUBLIC-KEYS { pk-XMSS }
      SMIME-CAPS { IDENTIFIED BY id-alg-xmss-with-SHA256 } }

   sa-XMSS-with-SHA384 SIGNATURE-ALGORITHM ::= {
      IDENTIFIER id-alg-xmss-with-SHA384
      PARAMS ARE absent
      HASHES { mda-sha384 }
      PUBLIC-KEYS { pk-XMSS }
      SMIME-CAPS { IDENTIFIED BY id-alg-xmss-with-SHA384 } }

   sa-XMSS-with-SHA512 SIGNATURE-ALGORITHM ::= {
      IDENTIFIER id-alg-xmss-with-SHA512
      PARAMS ARE absent
      HASHES { mda-sha512 }
      PUBLIC-KEYS { pk-XMSS }
```

```
      SMIME-CAPS { IDENTIFIED BY id-alg-xmss-with-SHA512 } }

   pk-XMSS PUBLIC-KEY ::= {
      IDENTIFIER id-alg-xmss
      KEY XMSS-PublicKey
      PARAMS ARE absent
      CERT-KEY-USAGE
         { digitalSignature, nonRepudiation, keyCertSign, cRLSign } }

   XMSS-PublicKey ::= OCTET STRING
```



```
   --
   -- XMSS^MT Keys and Signatures
   --

   -- XMSS^MT Object Identifiers

   id-alg-xmssmt  OBJECT IDENTIFIER ::= { itu-t(0)
      identified-organization(4) etsi(0) reserved(127)
      etsi-identified-organization(0) isara(15) algorithms(1)
      asymmetric(1) xmssmt(14) 0 }

   id-alg-xmssmt-with-SHA256  OBJECT IDENTIFIER ::= { itu-t(0)
      identified-organization(4) etsi(0) reserved(127)
      etsi-identified-organization(0) isara(15) algorithms(1)
      asymmetric(1) xmssmt(14) 2 }

   id-alg-xmssmt-with-SHA384  OBJECT IDENTIFIER ::= { itu-t(0)
      identified-organization(4) etsi(0) reserved(127)
      etsi-identified-organization(0) isara(15) algorithms(1)
      asymmetric(1) xmssmt(14) 3 }

   id-alg-xmssmt-with-SHA512  OBJECT IDENTIFIER ::= { itu-t(0)
      identified-organization(4) etsi(0) reserved(127)
      etsi-identified-organization(0) isara(15) algorithms(1)
      asymmetric(1) xmssmt(14) 1 }
```

```
   -- XMSS^MT Signature Algorithms and Public Key

   sa-XMSSMT SIGNATURE-ALGORITHM ::= {
       IDENTIFIER id-alg-xmssmt
       PARAMS ARE absent
       PUBLIC-KEYS { pk-XMSSMT }
       SMIME-CAPS { IDENTIFIED BY id-alg-xmssmt } }

   sa-XMSSMT-with-SHA256 SIGNATURE-ALGORITHM ::= {
      IDENTIFIER id-alg-xmssmt-with-SHA256
      PARAMS ARE absent
      HASHES { mda-sha256 }
      PUBLIC-KEYS { pk-XMSSMT }
      SMIME-CAPS { IDENTIFIED BY id-alg-xmssmt-with-SHA256 } }

   sa-XMSSMT-with-SHA384 SIGNATURE-ALGORITHM ::= {
      IDENTIFIER id-alg-xmssmt-with-SHA384
      PARAMS ARE absent
      HASHES { mda-sha384 }
      PUBLIC-KEYS { pk-XMSSMT }
      SMIME-CAPS { IDENTIFIED BY id-alg-xmssmt-with-SHA384 } }
```



```
   sa-XMSSMT-with-SHA512 SIGNATURE-ALGORITHM ::= {
      IDENTIFIER id-alg-xmssmt-with-SHA512
      PARAMS ARE absent
      HASHES { mda-sha512 }
      PUBLIC-KEYS { pk-XMSSMT }
      SMIME-CAPS { IDENTIFIED BY id-alg-xmssmt-with-SHA512 } }

   pk-XMSSMT PUBLIC-KEY ::= {
      IDENTIFIER id-alg-xmssmt
      KEY XMSSMT-PublicKey
      PARAMS ARE absent
      CERT-KEY-USAGE
         { digitalSignature, nonRepudiation, keyCertSign, cRLSign } }

   XMSSMT-PublicKey ::= OCTET STRING

   END
```

6. Security Considerations

6.1. Algorithm Security Considerations

The cryptographic security of the signatures generated by the algorithms mentioned in this document depends only on the hash algorithms used within the signature algorithms and the pre-hash algorithm used to create an X.509 certificate's message digest. Grover's algorithm [Grover96] is a quantum search algorithm which gives a quadratic improvement in search time to brute-force pre-image attacks. The results of [BBBV97] show that this improvement is optimal, however [Fluhrer17] notes that Grover's algorithm doesn't parallelize well. Thus, given a bounded amount of time to perform the attack and using a conservative estimate of the performance of a real quantum computer, the pre-image quantum security of SHA-256 is closer to 190 bits. All parameter sets for the signature algorithms in this document currently use SHA-256 internally and thus have at least 128 bits of quantum pre-image resistance, or 190 bits using the security assumptions in [Fluhrer17].

[Zhandry15] shows that hash collisions can be found using an algorithm with a lower bound on the number of oracle queries on the order of 2^(n/3) on the number of bits, however [DJB09] demonstrates that the quantum memory requirements would be much greater. Therefore a pre-hash using SHA-256 would have at least 128 bits of quantum collision-resistance as well as the pre-image resistance mentioned in the previous paragraph.

Given the quantum collision and pre-image resistance of SHA-256 estimated above, the algorithm identifiers id-alg-hss-with-SHA256,

id-alg-xmss-with-SHA256 and id-alg-xmssmt-with-SHA256 defined in this document provide 128 bits or more of quantum security. This is believed to be secure enough to protect X.509 certificates for well beyond any reasonable certificate lifetime, although the SHA-384 and SHA-512 variants could be used if there are any doubts.

6.2. 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 a 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.

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. [RFC4086] offers important guidance in this area.

The generation of hash-based signatures also depends on random numbers. While the consequences of an inadequate pseudo-random number generator (PRNGs) to generate these values is much less severe than the generation of private keys, the guidance in [RFC4086] remains important.

7. Acknowledgements

Thanks for Russ Housley for the helpful suggestions.

This document uses a lot of text from similar documents ([RFC3279] and [RFC8410]) as well as [I-D.ietf-lamps-cms-hash-sig]. Thanks go to the authors of those documents. "Copying always makes things easier and less error prone" - [RFC8411].

8. IANA Considerations

IANA is requested to assign a module OID from the "SMI for PKIX Module Identifier" registry for the ASN.1 module in Section 5.

9. References

9.1. Normative References

[I-D.ietf-lamps-cms-hash-sig]
Housley, R., "Use of the HSS/LMS Hash-based Signature Algorithm in the Cryptographic Message Syntax (CMS)", draft-ietf-lamps-cms-hash-sig-07 (work in progress), March 2019.

[I-D.mcgrew-hash-sigs]
McGrew, D., Curcio, M., and S. Fluhrer, "Hash-Based Signatures", draft-mcgrew-hash-sigs-15 (work in progress), January 2019.

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

[RFC4506] Eisler, M., Ed., "XDR: External Data Representation Standard", STD 67, RFC 4506, DOI 10.17487/RFC4506, May 2006, <https://www.rfc-editor.org/info/rfc4506>.

[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, DOI 10.17487/RFC5280, May 2008, <https://www.rfc-editor.org/info/rfc5280>.

[RFC8391] Huelsing, A., Butin, D., Gazdag, S., Rijneveld, J., and A. Mohaisen, "XMSS: eXtended Merkle Signature Scheme", RFC 8391, DOI 10.17487/RFC8391, May 2018, <https://www.rfc-editor.org/info/rfc8391>.

9.2. Informative References

[BBBV97] Bennett, C., Bernstein, E., Brassard, G., and U. Vazirani, "Strengths and weaknesses of quantum computing", SIAM J. Comput. 26(5), 1510-1523, 1997.

[DJB09] Bernstein, D., "Cost analysis of hash collisions: Will quantum computers make SHARCS obsolete?", SHARCS 9, p. 105, 2009.

[Fluhrer17]
Fluhrer, S., "Reassessing Grover's Algorithm", Cryptology ePrint Archive Report 2017/811, August 2017, <https://eprint.iacr.org/2017/811.pdf>.

[Grover96]
Grover, L., "A fast quantum mechanical algorithm for database search", 28th ACM Symposium on the Theory of Computing p. 212, 1996.

[RFC3279] Bassham, L., Polk, W., and R. Housley, "Algorithms and Identifiers for the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 3279, DOI 10.17487/RFC3279, April 2002, <https://www.rfc-editor.org/info/rfc3279>.

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

[RFC8410] Josefsson, S. and J. Schaad, "Algorithm Identifiers for Ed25519, Ed448, X25519, and X448 for Use in the Internet X.509 Public Key Infrastructure", RFC 8410, DOI 10.17487/RFC8410, August 2018, <https://www.rfc-editor.org/info/rfc8410>.

[RFC8411] Schaad, J. and R. Andrews, "IANA Registration for the Cryptographic Algorithm Object Identifier Range", RFC 8411, DOI 10.17487/RFC8411, August 2018, <https://www.rfc-editor.org/info/rfc8411>.

[Zhandry15]
Zhandry, M., "A note on the quantum collision and set equality problems", Quantum Information & Computation 15, 7-8, 557-567, May 2015.

Authors' Addresses

Daniel Van Geest
ISARA Corporation
560 Westmount Rd N
Waterloo, Ontario N2L 0A9
Canada

Email: daniel.vangeest@isara.com

Scott Fluhrer
Cisco Systems
170 West Tasman Drive
San Jose, CA 95134
USA

Email: sfluhrer@cisco.com