Workgroup:
LAMPS - Limited Additional Mechanisms for PKIX
and SMIME
Internet-Draft: draft-gazdag-x509-hash-sigs-03
Published: 15 February 2024
Intended Status: Informational
Expires: 18 August 2024
Authors: K. Bashiri S. Fluhrer S. Gazdag
BSI Cisco Systems genua GmbH
D. Van Geest S. Kousidis
BSI

Internet X.509 Public Key Infrastructure: Algorithm Identifiers for Hash-based Signatures

Abstract

This document specifies algorithm identifiers and ASN.1 encoding formats for the Hash-Based Signature (HBS) schemes Hierarchical Signature System (HSS), eXtended Merkle Signature Scheme (XMSS), and XMSS^MT, a multi-tree variant of XMSS, as well as SLH-DSA (formerly SPHINCS+), the latter being the only stateless scheme. This specification applies to the Internet X.509 Public Key infrastructure (PKI) when those digital signatures are used in Internet X.509 certificates and certificate revocation lists.

About This Document

This note is to be removed before publishing as an RFC.

Status information for this document may be found at https://datatracker.ietf.org/doc/draft-gazdag-x509-hash-sigs/.

Discussion of this document takes place on the LAMPS Working Group mailing list (mailto:spasm@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/spasm/. Subscribe at https://www.ietf.org/mailman/listinfo/spasm/.

Source for this draft and an issue tracker can be found at https://github.com/x509-hbs/draft-gazdag-x509-hash-sigs.

Status of This Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

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."

This Internet-Draft will expire on 18 August 2024.

Copyright Notice

Table of Contents

1. Introduction

Hash-Based Signature (HBS) Schemes combine Merkle trees with One/Few Time Signatures (OTS/FTS) in order to provide digital signature schemes that remain secure even when quantum computers become available. Their theoretic security is well understood and depends only on the security of the underlying hash function. As such they can serve as an important building block for quantum computer resistant information and communication technology.

The private key of HSS, XMSS and XMSS^MT is a finite collection of OTS keys, hence only a limited number of messages can be signed and the private key's state must be updated and persisted after signing to prevent reuse of OTS keys. 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. Due to the statefulness of the private key of HSS, XMSS and XMSS^MT and the limited number of signatures that can be created, these signature algorithms might not be appropriate for use in interactive protocols. However, in some use case scenarios the deployment of these signature algorithms may be appropriate. Such use cases are described and discussed later in Section 3.

The private key of SLH-DSA is a finite but very large collection of FTS keys and hence stateless. This typically comes at the cost of larger signatures compared to the stateful HBS variants. Thus SLH-DSA is suitable for more use cases if the signature sizes fit the requirements.

2. Conventions and Definitions

The key words "**MUST**", "**MUST NOT**", "**REQUIRED**", "**SHALL**", "**SHALL NOT**", "**SHOULD**", "**SHOULD NOT**", "**RECOMMENDED**", "**NOT RECOMMENDED**", "**MAY**", and "**OPTIONAL**" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

The parameter 'n' is the security parameter, given in bytes. In practice this is typically aligned to the standard output length of the hash function in use, i.e. either 16, 24, 32 or 64 bytes. The height of a single tree is typically given by the parameter 'h'. The number of levels of trees is either called 'L' (HSS) or 'd' (XMSS, XMSS^MT, SLH-DSA).

3. Use Cases of HBS in X.509

As many cryptographic algorithms that are considered to be quantum-resistant, HBS have several pros and cons regarding their practical usage. On the positive side they are considered to be secure against

a classical as well as a quantum adversary, and a secure instantiation of HBS may always be built as long as a cryptographically secure hash function exists. Moreover, HBS offer small public key sizes, and, in comparison to other post-quantum signature schemes, the stateful HBS can offer relatively small signature sizes (for certain parameter sets). While key generation and signature generation may take longer than classical alternatives, fast and minimal verification routines can be built. The major negative aspect is the statefulness of several HBS. Private keys always have to be handled in a secure manner, but stateful HBS necessitate a special treatment of the private key in order to avoid security incidents like signature forgery [MCGREW], [NIST-SP-800-208]. Therefore, for stateful HBS, a secure environment **MUST** be used for key generation and key management.

Note that, in general, root CAs offer such a secure environment and the number of issued signatures (including signed certificates and CRLs) is often moderate due to the fact that many root CAs delegate OCSP services or the signing of end-entity certificates to other entities (such as subordinate CAs) that use stateless signature schemes. Therefore, many root CAs should be able to handle the required state management, and stateful HBS offer a viable solution.

As the above reasoning for root CAs usually does not apply for subordinate CAs, it is **NOT RECOMMENDED** for subordinate CAs to use stateful HBS for issuing end-entity certificates. Moreover, stateful HBS **MUST NOT** be used for end-entity certificates.

However, stateful HBS **MAY** be used for code signing certificates, since they are suitable and recommended in such non-interactive contexts. For example, see the recommendations for software and firmware signing in [CNSA2.0]. Some manufactures use common and well-established key formats like X.509 for their code signing and update mechanisms. Also there are multi-party IoT ecosystems where publicly trusted code signing certificates are useful.

4. 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 [RFC8708] and repeated here for convenience. For all of the OIDs, the parameters **MUST** be absent.

4.1. HSS Public Keys

The object identifier and public key algorithm identifier for HSS is defined in [RFC8708]. 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 [RFC8554].

The HSS public key identifier is 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 } }
```

The HSS public key is defined as follows:

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

See [NIST-SP-800-208] and [RFC8554] for more information on the contents and format of an HSS public key. Note that the single-tree signature scheme LMS is instantiated as HSS with level L=1.

4.2. XMSS Public Keys

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

```
id-alg-xmss-hashsig  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 identifier is as follows:

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

The XMSS public key is defined as follows:

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

See [NIST-SP-800-208] and [RFC8391] for more information on the contents and format of an XMSS public key.

4.3. XMSS^MT Public Keys

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

```
id-alg-xmssmt-hashsig  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 identifier is as follows:

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

The XMSS^MT public key is defined as follows:

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

See [NIST-SP-800-208] and [RFC8391] for more information on the contents and format of an XMSS^MT public key.

4.4. SLH-DSA Public Keys

The object and public key algorithm identifiers for SLH-DSA are defined in [I-D.ietf-lamps-cms-sphincs-plus]. The definitions are repeated here for reference.

```
id-alg-sphincs-plus-128 OBJECT IDENTIFIER ::= {
   TBD }

id-alg-sphincs-plus-192 OBJECT IDENTIFIER ::= {
   TBD }

id-alg-sphincs-plus-256 OBJECT IDENTIFIER ::= {
   TBD }
```

The SLH-DSA public key identifier is as follows:

```
pk-sphincs-plus-128 PUBLIC-KEY ::= {
   IDENTIFIER id-alg-sphincs-plus-128
   KEY SPHINCS-Plus-PublicKey
   PARAMS ARE absent
   CERT-KEY-USAGE
     { digitalSignature, nonRepudiation, keyCertSign, cRLSign } }

pk-sphincs-plus-192 PUBLIC-KEY ::= {
   IDENTIFIER id-alg-sphincs-plus-192
   KEY SPHINCS-Plus-PublicKey
   PARAMS ARE absent
   CERT-KEY-USAGE
     { digitalSignature, nonRepudiation, keyCertSign, cRLSign } }

pk-sphincs-plus-256 PUBLIC-KEY ::= {
   IDENTIFIER id-alg-sphincs-plus-256
   KEY SPHINCS-Plus-PublicKey
   PARAMS ARE absent
   CERT-KEY-USAGE
      { digitalSignature, nonRepudiation, keyCertSign, cRLSign } }
```

The SLH-DSA public key is defined as follows:

```
SPHINCS-Plus-PublicKey ::= OCTET STRING
```

See [NIST-FIPS-205] for more information on the contents and format of a SLH-DSA public key.

5. Key Usage Bits

The intended application for the key is indicated in the keyUsage certificate extension [RFC5280].

If the keyUsage extension is present in an end-entity certificate that indicates id-alg-sphincs-plus-128, id-alg-sphincs-plus-192, or id-alg-sphincs-plus-256, then it **MUST** contain at least one of the following values:

nonRepudiation; or
digitalSignature.

If the keyUsage extension is present in a code signing certificate that indicates id-alg-hss-lms-hashsig, id-alg-xmss-hashsig, id-alg-xmssmt-hashsig, id-alg-sphincs-plus-128, id-alg-sphincs-plus-192, or id-alg-sphincs-plus-256, then it **MUST** contain at least one of the following values:

nonRepudiation; or
digitalSignature.

If the keyUsage extension is present in a certification authority certificate that indicates id-alg-hss-lms-hashsig, id-alg-xmss-hashsig, id-alg-xmssmt-hashsig, id-alg-sphincs-plus-128, id-alg-sphincs-plus-192, or id-alg-sphincs-plus-256, then it **MUST** contain at least one of the following values:

nonRepudiation; or
digitalSignature; or
keyCertSign; or
cRLSign.

Note that for certificates that indicate id-alg-hss-lms-hashsig the above definitions are more restrictive than the requirement defined in Section 4 of [RFC8708].

6. Signature Algorithms

This section identifies OIDs for signing using HSS, XMSS, XMSS^MT, and SLH-DSA. 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 in the following subsections.

The data to be signed is prepared for signing. For the algorithms used in this document, the data is signed directly by the signature algorithm, the data is not hashed before processing. Then, a private key operation is performed to generate the signature value. For HSS, the signature value is described in section 6.4 of [RFC8554]. For XMSS and XMSS^MT the signature values are described in sections B.2 and C.2 of [RFC8391], respectively. For SLH-DSA the signature values are described in 9.2 of [NIST-FIPS-205]. The octet string representing the signature is encoded directly in the OCTET STRING without adding any additional ASN.1 wrapping. For the Certificate and CertificateList structures, the signature value is wrapped in the "signatureValue" OCTET STRING field.

6.1. 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 HSS signature is defined as follows:

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

See [NIST-SP-800-208] and [RFC8554] for more information on the contents and format of an HSS signature.

6.2. 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-hashsig  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 signature is defined as follows:

```
XMSS-HashSig-Signature ::= OCTET STRING
```

See [NIST-SP-800-208] and [RFC8391] for more information on the contents and format of an XMSS signature.

The signature generation **MUST** be performed according to 7.2 of [NIST-SP-800-208].

6.3. 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-hashsig  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 signature is defined as follows:

```
XMSSMT-HashSig-Signature ::= OCTET STRING
```

See [NIST-SP-800-208] and [RFC8391] for more information on the contents and format of an XMSS^MT signature.

The signature generation **MUST** be performed according to 7.2 of [NIST-SP-800-208].

6.4. SLH-DSA Signature Algorithm

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

```
id-alg-sphincs-plus-128 OBJECT IDENTIFIER ::= {
   TBD }

id-alg-sphincs-plus-192 OBJECT IDENTIFIER ::= {
   TBD }

id-alg-sphincs-plus-256 OBJECT IDENTIFIER ::= {
   TBD }
```

The SLH-DSA signature is defined as follows:

```
SPHINCS-Plus-Signature ::= OCTET STRING
```

See [NIST-FIPS-205] for more information on the contents and format of a SLH-DSA signature.

7. Key Generation

The key generation for XMSS and XMSS^MT **MUST** be performed according to 7.2 of [NIST-SP-800-208]

8. ASN.1 Module

For reference purposes, the ASN.1 syntax is presented as an ASN.1 module here. This ASN.1 Module builds upon the conventions established in [RFC5911].

```
--
-- ASN.1 Module
--

<CODE STARTS>

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

DEFINITIONS IMPLICIT TAGS ::= BEGIN

EXPORTS ALL;

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)}

--
-- Object Identifiers
--

-- id-alg-hss-lms-hashsig is defined in [RFC8708]

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-xmss-hashsig  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-xmssmt-hashsig  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-sphincs-plus-128 OBJECT IDENTIFIER ::= {
   TBD }

id-alg-sphincs-plus-192 OBJECT IDENTIFIER ::= {
   TBD }

id-alg-sphincs-plus-256 OBJECT IDENTIFIER ::= {
   TBD }

--
-- Signature Algorithms and Public Keys
--
```

```
-- sa-HSS-LMS-HashSig is defined in [RFC8708]

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

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

-- sa-sphincs-plus-128 is defined in [I-D.ietf-lamps-cms-sphincs-plus]

sa-sphincs-plus-128 SIGNATURE-ALGORITHM ::= {
   IDENTIFIER id-alg-sphincs-plus-128
   PARAMS ARE absent
   PUBLIC-KEYS { pk-sphincs-plus-128 }
   SMIME-CAPS { IDENTIFIED BY id-alg-sphincs-plus-128 } }

-- sa-sphincs-plus-192 is defined in [I-D.ietf-lamps-cms-sphincs-plus]

sa-sphincs-plus-192 SIGNATURE-ALGORITHM ::= {
   IDENTIFIER id-alg-sphincs-plus-192
   PARAMS ARE absent
   PUBLIC-KEYS { pk-sphincs-plus-192 }
   SMIME-CAPS { IDENTIFIED BY id-alg-sphincs-plus-192 } }

-- sa-sphincs-plus-256 is defined in [I-D.ietf-lamps-cms-sphincs-plus]

sa-sphincs-plus-256 SIGNATURE-ALGORITHM ::= {
   IDENTIFIER id-alg-sphincs-plus-256
   PARAMS ARE absent
   PUBLIC-KEYS { pk-sphincs-plus-256 }
   SMIME-CAPS { IDENTIFIED BY id-alg-sphincs-plus-256 } }

-- pk-HSS-LMS-HashSig is defined in [RFC8708]

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

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

XMSS-HashSig-PublicKey ::= OCTET STRING

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

XMSSMT-HashSig-PublicKey ::= OCTET STRING

-- pk-sphincs-plus-128 is defined in [I-D.ietf-lamps-cms-sphincs-plus]

pk-sphincs-plus-128 PUBLIC-KEY ::= {
   IDENTIFIER id-alg-sphincs-plus-128
   KEY SPHINCS-Plus-PublicKey
   PARAMS ARE absent
   CERT-KEY-USAGE
      { digitalSignature, nonRepudiation, keyCertSign, cRLSign } }

-- pk-sphincs-plus-192 is defined in [I-D.ietf-lamps-cms-sphincs-plus]

pk-sphincs-plus-192 PUBLIC-KEY ::= {
   IDENTIFIER id-alg-sphincs-plus-192
   KEY SPHINCS-Plus-PublicKey
   PARAMS ARE absent
   CERT-KEY-USAGE
      { digitalSignature, nonRepudiation, keyCertSign, cRLSign } }

-- pk-sphincs-plus-256 is defined in [I-D.ietf-lamps-cms-sphincs-plus]

pk-sphincs-plus-256 PUBLIC-KEY ::= {
   IDENTIFIER id-alg-sphincs-plus-256
   KEY SPHINCS-Plus-PublicKey
   PARAMS ARE absent
   CERT-KEY-USAGE
      { digitalSignature, nonRepudiation, keyCertSign, cRLSign } }
```

```
SPHINCS-Plus-PublicKey ::= OCTET STRING

END

<CODE ENDS>
```

9. Security Considerations

The security requirements of [NIST-SP-800-208] and [NIST-FIPS-205] **MUST** be taken into account.

For the stateful HBS (HSS, XMSS, XMSS^MT) it is crucial to stress the importance of a correct state management. If an attacker were able to obtain signatures for two different messages created using the same OTS key, then it would become computationally feasible for that attacker to create forgeries [BH16]. As noted in [MCGREW] and [ETSI-TR-103-692], extreme care needs to be taken in order to avoid the risk that an OTS key will be reused accidentally. This is a new requirement that most developers will not be familiar with and requires careful handling.

Various strategies for a correct state management can be applied:

* Implement a track record of all signatures generated by a key pair associated to a stateful HBS instance. This track record may be stored outside the device which is used to generate the signature. Check the track record to prevent OTS key reuse before a new signature is released. Drop the new signature and hit your PANIC button if you spot OTS key reuse.

* Use a stateful HBS instance only for a moderate number of signatures such that it is always practical to keep a consistent track record and be able to unambiguously trace back all generated signatures.

* Apply the state reservation strategy described in Section 5 of [MCGREW], where upcoming states are reserved in advance by the signer. In this way the number of state synchronisations between nonvolatile and volatile memory is reduced.

10. Backup and Restore Management

Certificate Authorities have high demands in order to ensure the availability of signature generation throughout the validity period of signing key pairs.

Usual backup and restore strategies when using a stateless signature scheme (e.g. SLH-DSA) are to duplicate private keying material and to operate redundant signing devices or to store and safeguard a copy of the private keying material such that it can be used to set up a new signing device in case of technical difficulties.

For stateful HBS such straightforward backup and restore strategies will lead to OTS reuse with high probability as a correct state management is not guaranteed. Strategies for maintaining

availability and keeping a correct state are described in Section 7 of [NIST-SP-800-208].

11. 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 6.

12. References

12.1. Normative References

[I-D.ietf-lamps-cms-sphincs-plus] Housley, R., Fluhrer, S., Kampanakis, P., and B. Westerbaan, "Use of the SLH-DSA Signature Algorithm in the Cryptographic Message Syntax (CMS)", Work in Progress, Internet-Draft, draft-ietf-lamps-cms-sphincs-plus-03, 14 November 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-lamps-cms-sphincs-plus-03>.

[NIST-FIPS-205] National Institute of Standards and Technology (NIST), "Stateless Hash-Based Digital Signature Standard", 24 August 2023, <https://doi.org/10.6028/NIST.FIPS.205.ipd>.

[NIST-SP-800-208] National Institute of Standards and Technology (NIST), "Recommendation for Stateful Hash-Based Signature Schemes", 29 October 2020, <https://doi.org/10.6028/NIST.SP.800-208>.

[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/rfc/rfc2119>.

[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/rfc/rfc5280>.

[RFC5911] 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, <https://www.rfc-editor.org/rfc/rfc5911>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/rfc/rfc8174>.

[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/rfc/rfc8391>.

[RFC8554] McGrew, D., Curcio, M., and S. Fluhrer, "Leighton-Micali Hash-Based Signatures", RFC 8554, DOI 10.17487/RFC8554, April 2019, <https://www.rfc-editor.org/rfc/rfc8554>.

[RFC8708] Housley, R., "Use of the HSS/LMS Hash-Based Signature Algorithm in the Cryptographic Message Syntax (CMS)", RFC 8708, DOI 10.17487/RFC8708, February 2020, <https://www.rfc-editor.org/rfc/rfc8708>.

12.2. Informative References

[BH16] Bruinderink, L. and S. Hülsing, "Oops, I did it again – Security of One-Time Signatures under Two-Message Attacks.", 2016, <https://eprint.iacr.org/2016/1042.pdf>.

[CNSA2.0] National Security Agency (NSA), "Commercial National Security Algorithm Suite 2.0 (CNSA 2.0) Cybersecurity Advisory (CSA)", 7 September 2022, <https://media.defense.gov/2022/Sep/07/2003071834/-1/-1/0/CSA_CNSA_2.0_ALGORITHMS_.PDF>.

[ETSI-TR-103-692] European Telecommunications Standards Institute (ETSI), "State management for stateful authentication mechanisms", November 2021, <https://www.etsi.org/deliver/etsi_tr/103600_103699/103692/01.01.01_60/tr_103692v010101p.pdf>.

[MCGREW] McGrew, D., Kampanakis, P., Fluhrer, S., Gazdag, S., Butin, D., and J. Buchmann, "State Management for Hash-Based Signatures", 2 November 2016, <https://tubiblio.ulb.tu-darmstadt.de/id/eprint/101633>.

[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/rfc/rfc3279>.

[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/rfc/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/rfc/rfc8411>.

Acknowledgments

Thanks for Russ Housley and Panos Kampanakis for helpful suggestions.

This document uses a lot of text from similar documents [NIST-SP-800-208], ([RFC3279] and [RFC8410]) as well as [RFC8708]. Thanks go to the authors of those documents. "Copying always makes things easier and less error prone" - [RFC8411].

Authors' Addresses

Kaveh Bashiri
BSI

Email: kaveh.bashiri.ietf@gmail.com

Scott Fluhrer
Cisco Systems

Email: sfluhrer@cisco.com

Stefan Gazdag
genua GmbH

Email: ietf@gazdag.de

Daniel Van Geest

Email: daniel.vangeest.ietf@gmail.com

Stavros Kousidis
BSI

Email: kousidis.ietf@gmail.com