IP Security Maintenance and Extensions (ipsecme)
Internet-Draft
Updates: RFC 5996 (if approved)
Intended status: Standards Track
Expires: September 29, 2014

T. Kivinen
INSIDE Secure

March 28, 2014

Signature Authentication in IKEv2
draft-kivinen-ipsecme-signature-auth-05.txt

Abstract

The Internet Key Exchange Version 2 (IKEv2) protocol has limited support for the Elliptic Curve Digital Signature Algorithm (ECDSA). The current version only includes support for three Elliptic Curve groups, and there is fixed hash algorithm tied to each curve. This document generalizes the IKEv2 signature support so it can support any signature method supported by the PKIX and also adds signature hash algorithm negotiation. This is generic mechanism, and is not limited to ECDSA, but can also be used with other signature algorithms.

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 http://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 September 29, 2014.

Table of Contents

1. Introduction

This document adds new IKEv2 ([RFC5996]) authentication method to support all kinds of signature methods. The current signature based authentication methods in the IKEv2 are per algorithm, i.e. there is one for RSA Digital signatures, one for DSS Digital Signatures (using SHA-1) and three for different ECDSA curves each tied to exactly one hash algorithm. This design starts to be cumbersome when more signature algorithms, hash algorithms and elliptic curves are to be supported:

- The RSA Digital Signatures format in the IKEv2 is specified to use RSASSA-PKCS1-v1_5 padding, but Additional RSA Algorithms and Identifiers for X.509 document recommends the use of the newer RSASSA_PSS (See section 5 of [RFC4055]) instead.
- With ECDSA and DSS there is no way to extract the hash algorithm from the signature, thus, for each new hash function to be supported with ECDSA or DSA new authentication methods would be needed. Support for new hash functions is particularly needed for DSS because the current restriction to SHA-1 limits its security, meaning there is no point of using long keys with it.
- The tying of ECDSA authentication methods to particular elliptic curve groups requires definition of additional methods for each new group. By combination of new ECDSA groups with various hash functions the number of required authentication methods may grow unmanageable. Furthermore, the restriction of ECDSA authentication to a specific group is inconsistent with the approach taken with DSS.

With the selection of SHA-3, it is seen that it might be possible that in the future the signature methods are used with SHA-3 also, not only SHA-2. This means new mechanism for negotiating the hash algorithm for the signature algorithms is needed.

This documents specifies two things, one is one new authentication method, which includes enough information inside the Authentication payload data that the signature hash algorithm can be extracted from there (see Section 3). The another thing is to add indication of supported signature hash algorithms by the peer (see Section 4). This allows peer to know which hash algorithms are supported by the other end and use one of them (provided one is allowed by policy). There is no need to actually negotiate one common hash algorithm, as different hash algorithms can be used in different directions if needed.

The new digital signature method needs to be flexible enough to include all current signature methods (RSA, DSA, ECDSA, RSASSA-PSS, etc), and also allow adding new things in the future (ECGDSA, ElGamal

etc). For this the signature algorithm is specified in the same way as the PKIX ([RFC5280]) specifies the signature of the Certificate, i.e. there is simple ASN.1 object before the actual signature data. This ASN.1 object contains the OID specifying the algorithm, and associated parameters to it. In normal case the IKEv2 implementations supports fixed amount of signature methods, with commonly used parameters, so it is acceptable for the implementation to just treat this ASN.1 object as binary blob which is compared against the known values, or the implementation can parse the ASN.1 and extract information from there.

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 [RFC2119].

3. Authentication Payload

This document specifies new "Digital Signature" authentication method. This method can be used with any types of signatures. As the authentication methods are not negotiated in the IKEv2, the peer is only allowed to use this authentication method if the SIGNATURE_HASH_ALGORITHMS Notify Payload has been sent and received.

In this newly defined authentication method, the Authentication Data field inside the Authentication Payload does not include only the signature value, but instead the signature value is prefixed with the ASN.1 object containing the algorithm used to generate the signature. The ASN.1 object contains the algorithm identification OID, and this OID identifies both the signature algorithm and the hash used when calculating the signature. In addition to the OID there is optional parameters which might be needed for algorithms like RSASSA-PSS.

To make implementations easier, the ASN.1 object is prefixed by the 8-bit length field. This length field allows simple implementations to be able to know the length of the ASN.1 without the need to parse it, so they can use it as binary blob which is compared against the known signature algorithm ASN.1 objects, i.e. they do not need to be able to parse or generate ASN.1 objects. See Appendix A for commonly used ASN.1 objects.

The ASN.1 used here is the same ASN.1 which is used in the AlgorithmIdentifier of the PKIX (Section 4.1.1.2 of [RFC5280]) encoded using distinguished encoding rules (DER) [CCITT.X690.2002]. The algorithm OID inside the ASN.1 specifies the signature algorithm

and the hash function, which are needed for signature verification. The EC curve is always known by the peer because it needs to have the certificate or the public key of the other end before it can do signature verification and public key specifies the curve.

Currently only the RSASSA-PSS uses the parameters, for all others the parameters is either NULL or missing. Note, that for some algorithms there is two possible ASN.1 encoding possible, one with parameters being NULL and others where the whole parameters is omitted. This is because some of those algorithms are specified that way. When encoding the ASN.1 implementations should use the preferred way, i.e. if the algorithm specification says "preferredPresent" then parameter object needs to be there (i.e. it will be NULL if no parameters is specified), and if it says "preferredAbsent", then the whole parameters object is missing.

The Authentication payload is defined in IKEv2 as follows:

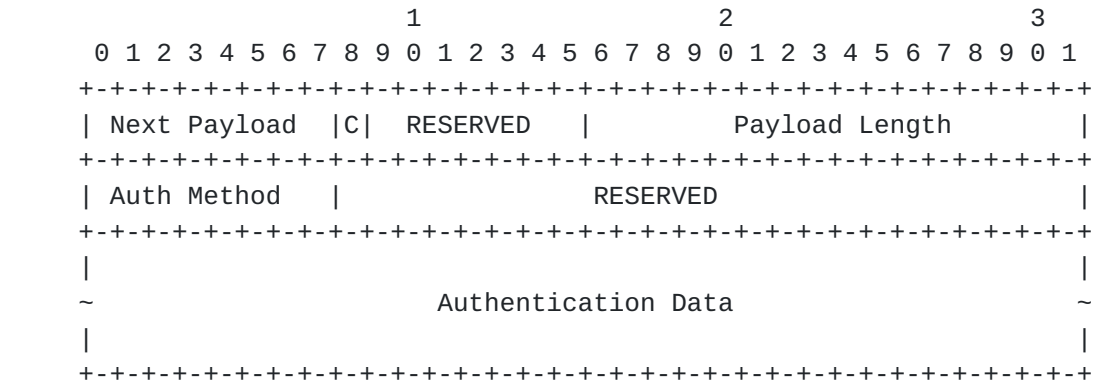

Figure 1: Authentication Payload Format.

o Auth Method (1 octet) - Specifies the method of authentication used.

Mechanism	Value
Digital Signature	<TBD>

Computed as specified in Section 2.15 of RFC5996 using a private key associated with the public key sent in certificate payload, and using one of the hash algorithms sent by the other end in the SIGNATURE_HASH_ALGORITHMS notify payload. If both ends send and receive SIGNATURE_HASH_ALGORITHMS and signature authentication is to be used, then this method MUST be used. The Authentication Data field has bit different format than in other Authentication methods (see below).

o Authentication Data (variable length) - see Section 2.15 of RFC5996. For "Digital Signature" format the Authentication data contains special format as follows:

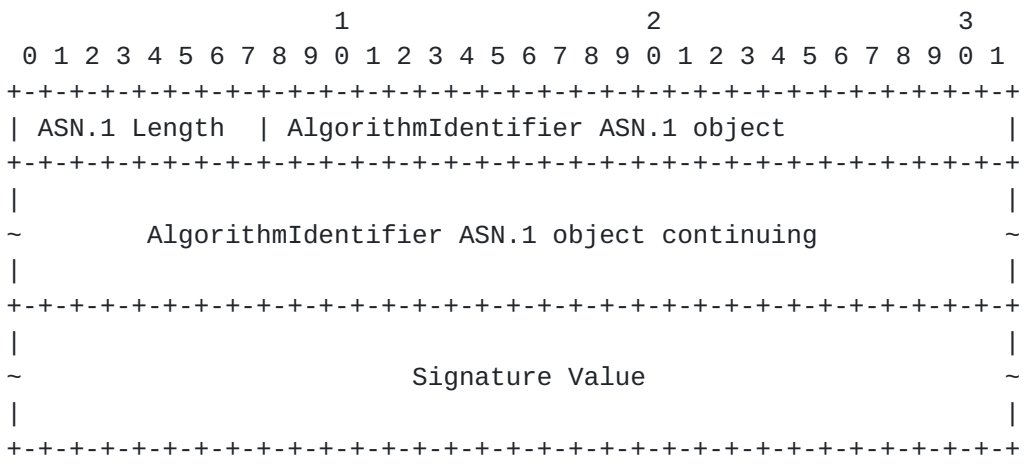

Figure 2: Authentication Data Format.

Where the ASN.1 Length is the length of the ASN.1 encoded AlgorithmIdentifier object, and after that is the actual AlgorithmIdentifier ASN.1 object, followed by the actual signature value. There is no padding between ASN.1 object and signature value. For the hash truncation the method of X9.62 ([X9.62]) MUST be used.

4. Hash Algorithm Notification

The supported hash algorithms that can be used for the signature algorithms are now indicated with new SIGNATURE_HASH_ALGORITHMS Notification Payload sent inside the IKE_SA_INIT exchange. This notification also indicates the support of the new signature algorithm method, i.e. sending this notification tells that new "Digital Signature" authentication method is supported and that following hash functions are supported by sending peer. Both ends sends their list of supported hash-algorithms and when calculating signature a peer MUST pick one algorithm sent by the other peer. Note, that different algorithms can be used in different directions. The algorithm OID matching selected hash algorithm (and signature algorithm) used when calculating the signature is sent inside the Authentication Data field of the Authentication Payload.

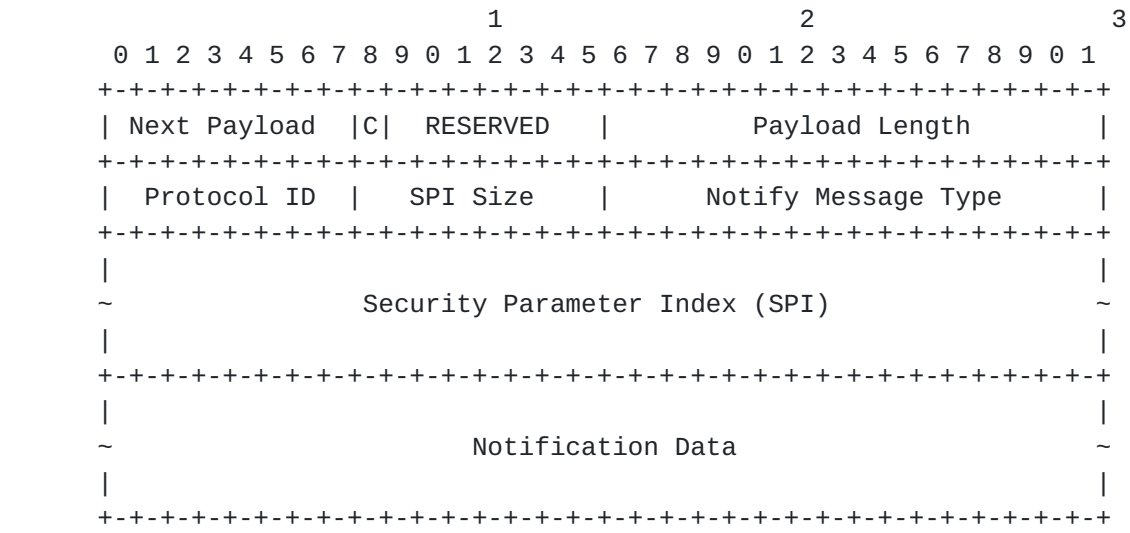

Figure 3: Notify Payload Format.

Protocol ID is 0, SPI Size 0, and Notify Message Type <TBD from status types>. The Notification Data value contains list of 16-bit hash algorithm identifiers from the newly created Hash Algorithm Identifiers for the IKEv2 IANA registry.

5. Selecting Public Key Algorithm

This specification does not provide a way for the peers to indicate the public / private key pair types they have. I.e. how can the responder select public / private key pair type that the initiator supports. There is already several ways this information can be found in common cases.

One of the ways to find out which key the initiator wants the responder to use is to indicate that in the IDr payload of the IKE_AUTH request of the initiator. I.e initiator indicates that it wants the responder to use certain public / private key pair by sending IDr which indicates that information. This means the responder needs to have different identities configured and each of those identities needs to be tied up to certain public / private key (or key type).

Another way to get this information is from the Certificate Request payload sent by the initiator. For example if the initiator indicates in his Certificate Request payload that it trust CA which is signed by the ECDSA key, that will also indicate it can be process ECDSA signatures, thus responder can safely use ECDSA keys when authenticating himself.

Responder can also check the key type used by the initiator, and use same key type than the initiator used. This does not work in case the initiator is using shared secret or EAP authentication, as in that case it is not using public key. If initiator is using public key authentication himself this is most likely the best way for the responder to find the type the initiator supports.

In case the initiator uses a public key type that the responder will not support, the responder will reply with AUTHENTICATION_FAILED error. If initiator has multiple different keys it can try different key (and perhaps different key type) until it finds key that the other end accepts. Initiator can also use the Certificate Request payload sent by the responder to help deciding which public key should be tried. In normal case if initiator has multiple public keys, there is configuration that will select one of those for each connection, so the proper key is know by configuration.

6. Security Considerations

The "Recommendations for Key Management" ([NIST800-57]) table 2 combined with table 3 gives recommendations for how to select suitable hash functions for the signature.

This new digital signature method does not tie the EC curve to the specific hash function, which was done in the old IKEv2 ECDSA methods. This means it is possible to use 512-bit EC curve with SHA1, i.e. this allows mixing different security levels. This means that the security of the authentication method is the security of the weakest of components (signature algorithm, hash algorithm, curve). This might make the security analysis of the system bit more complex. Note, that this kind of mixing of the security can be disallowed by the policy.

The hash algorithm registry does not include MD5 as supported hash algorithm, as it is not considered safe enough for signature use ([WY05]).

The current IKEv2 uses RSASSA-PKCS1-v1_5, which do have some problems ([KA08], [ME01]) and does not allow using newer padding methods like RSASSA-PSS. This new method allows using other padding methods.

The current IKEv2 only allows using normal DSA with SHA-1, which means the security of the regular DSA is limited to the security of SHA-1. This new methods allows using longer keys and longer hashes with DSA.

7. IANA Considerations

This document creates new IANA registry for IKEv2 Hash Algorithms. Changes and additions to this registry is by expert review.

The initial values of this registry is:

Hash Algorithm	Value
RESERVED	0
SHA1	1
SHA2-256	2
SHA2-384	3
SHA2-512	4

MD5 is not included to the hash algorithm list as it is not considered safe enough for signature hash uses.

Values 5-1023 are reserved to IANA. Values 1024-65535 are for private use among mutually consenting parties.

This specification also allocates one new IKEv2 Notify Message Types - Status Types value for the SIGNATURE_HASH_ALGORITHMS, and adds new value "Digital Signature" to the IKEv2 Authentication Method registry.

8. Acknowledgements

Most of this work was based on the work done in the IPsecME design team for the ECDSA. The design team members were: Dan Harking, Johannes Merkle, Tero Kivinen, David McGrew, and Yoav Nir.

9. References

9.1. Normative References

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

[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.

[RFC5996] Kaufman, C., Hoffman, P., Nir, Y., and P. Eronen, "Internet Key Exchange Protocol Version 2 (IKEv2)",

RFC 5996, September 2010.

9.2. Informative References

[CCITT.X690.2002]
International Telephone and Telegraph Consultative Committee, "ASN.1 encoding rules: Specification of basic encoding Rules (BER), Canonical encoding rules (CER) and Distinguished encoding rules (DER)", CCITT Recommendation X.690, July 2002.

[KA08] Kuehn, U., Pyshkin, A., Tews, E., and R. Weinmann, "Variants of Bleichenbacher's Low-Exponent Attack on PKCS#1 RSA Signatures", Proc. Sicherheit 2008 pp.97-109.

[ME01] Menezes, A., "Evaluation of Security Level of Cryptography: RSA-OAEP, RSA-PSS, RSA Signature", December 2001.

[NIST800-57]
Barker, E., Barker, W., Burr, W., Polk, W., and M. Smid, "Recommendations for Key Management", NIST SP 800-57, March 2007.

[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, April 2002.

[RFC4055] Schaad, J., Kaliski, B., and R. Housley, "Additional Algorithms and Identifiers for RSA Cryptography for use in the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 4055, June 2005.

[RFC5480] Turner, S., Brown, D., Yiu, K., Housley, R., and T. Polk, "Elliptic Curve Cryptography Subject Public Key Information", RFC 5480, March 2009.

[RFC5758] Dang, Q., Santesson, S., Moriarty, K., Brown, D., and T. Polk, "Internet X.509 Public Key Infrastructure: Additional Algorithms and Identifiers for DSA and ECDSA", RFC 5758, January 2010.

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

[WY05] Wang, X. and H. Yu, "How to break MD5 and other hash functions", Proceedings of EuroCrypt 2005, Lecture Notes in Computer Science Vol. 3494, 2005.

[X9.62] American National Standards Institute, "Public Key Cryptography for the Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA)", ANSI X9.62, November 2005.

Appendix A. Commonly used ASN.1 objects

This section lists commonly used ASN.1 objects in binary form. This section is not-normative, and these values should only be used as examples, i.e. if this and the actual specification of the algorithm ASN.1 object is different the actual format specified in the actual specification needs to be used. These values are taken from the New ASN.1 Modules for the Public Key Infrastructure Using X.509 ([RFC5912]).

A.1. PKCS#1 1.5 RSA Encryption

These algorithm identifiers here include several different ASN.1 objects with different hash algorithms. In this document we only include the commonly used ones i.e. the one using SHA-1, or SHA-2 as hash function. Some of those other algorithms (MD2, MD5) specified for this are not safe enough to be used as signature hash algorithm, and some are omitted as there is no hash algorithm specified in the our IANA registry for them. Note, that there is no parameters in any of these, but all specified here needs to have NULL parameters present in the ASN.1.

See Algorithms and Identifiers for PKIX Profile ([RFC3279]) and Additional Algorithms and Identifiers for RSA Cryptography for PKIX Profile ([RFC4055]) for more information.

A.1.1. sha1WithRSAEncryption

```
sha1WithRSAEncryption OBJECT IDENTIFIER ::= { iso(1) member-body(2)
us(840) rsadsi(113549) pkcs(1) pkcs-1(1) 5 }
```

Parameters are required, and they must be NULL.

```
Name = sha1WithRSAEncryption, oid = 1.2.840.113549.1.1.5
Length = 15
0000: 300d 0609 2a86 4886 f70d 0101 0505 00
```

A.1.2. sha256WithRSAEncryption

sha256WithRSAEncryption OBJECT IDENTIFIER ::= { pkcs-1 11 }

Parameters are required, and they must be NULL.

Name = sha256WithRSAEncryption, oid = 1.2.840.113549.1.1.11
Length = 15
0000: 300d 0609 2a86 4886 f70d 0101 0b05 00

A.1.3. sha384WithRSAEncryption

sha384WithRSAEncryption OBJECT IDENTIFIER ::= { pkcs-1 12 }

Parameters are required, and they must be NULL.

Name = sha384WithRSAEncryption, oid = 1.2.840.113549.1.1.12
Length = 15
0000: 300d 0609 2a86 4886 f70d 0101 0c05 00

A.1.4. sha512WithRSAEncryption

sha512WithRSAEncryption OBJECT IDENTIFIER ::= { pkcs-1 13 }

Parameters are required, and they must be NULL.

Name = sha512WithRSAEncryption, oid = 1.2.840.113549.1.1.13
Length = 15
0000: 300d 0609 2a86 4886 f70d 0101 0d05 00

A.2. DSA

With different DSA algorithms the parameters are always omitted. Again we omit dsa-with-sha224 as there is no hash algorithm in our IANA registry for it.

See Algorithms and Identifiers for PKIX Profile ([RFC3279]) and PKIX Additional Algorithms and Identifiers for DSA and ECDSA ([RFC5758] for more information.

A.2.1. dsa-with-sha1

dsa-with-sha1 OBJECT IDENTIFIER ::= { iso(1) member-body(2) us(840) x9-57(10040) x9algorithm(4) 3 }

Parameters are absent.

```
Name = dsa-with-sha1, oid = 1.2.840.10040.4.3
Length = 11
0000: 3009 0607 2a86 48ce 3804 03
```

A.2.2. dsa-with-sha256

```
dsa-with-sha256 OBJECT IDENTIFIER ::= { joint-iso-ccitt(2)
country(16) us(840) organization(1) gov(101) csor(3) algorithms(4)
id-dsa-with-sha2(3) 2 }

Parameters are absent.

Name = dsa-with-sha256, oid = 2.16.840.1.101.3.4.3.2
Length = 13
0000: 300b 0609 6086 4801 6503 0403 02
```

A.3. ECDSA

With different ECDSA algorithms the parameters are always omitted. Again we omit ecdsa-with-sha224 as there is no hash algorithm in our IANA registry for it.

See Elliptic Curve Cryptography Subject Public Key Information ([RFC5480]), Algorithms and Identifiers for PKIX Profile ([RFC3279]) and PKIX Additional Algorithms and Identifiers for DSA and ECDSA ([RFC5758] for more information.

A.3.1. ecdsa-with-sha1

```
ecdsa-with-SHA1 OBJECT IDENTIFIER ::= { iso(1) member-body(2) us(840)
ansi-X9-62(10045) signatures(4) 1 }

Parameters are absent.

Name = ecdsa-with-sha1, oid = 1.2.840.10045.4.1
Length = 11
0000: 3009 0607 2a86 48ce 3d04 01
```

A.3.2. ecdsa-with-sha256

```
ecdsa-with-SHA256 OBJECT IDENTIFIER ::= { iso(1) member-body(2)
us(840) ansi-X9-62(10045) signatures(4) ecdsa-with-SHA2(3) 2 }

Parameters are absent.

Name = ecdsa-with-sha256, oid = 1.2.840.10045.4.3.2
Length = 12
0000: 300a 0608 2a86 48ce 3d04 0302
```

A.3.3. ecdsa-with-sha384

ecdsa-with-SHA384 OBJECT IDENTIFIER ::= { iso(1) member-body(2) us(840) ansi-X9-62(10045) signatures(4) ecdsa-with-SHA2(3) 3 }

Parameters are absent.

Name = ecdsa-with-sha384, oid = 1.2.840.10045.4.3.3
Length = 12
0000: 300a 0608 2a86 48ce 3d04 0303

A.3.4. ecdsa-with-sha512

ecdsa-with-SHA512 OBJECT IDENTIFIER ::= { iso(1) member-body(2) us(840) ansi-X9-62(10045) signatures(4) ecdsa-with-SHA2(3) 4 }

Parameters are absent.

Name = ecdsa-with-sha512, oid = 1.2.840.10045.4.3.4
Length = 12
0000: 300a 0608 2a86 48ce 3d04 0304

A.4. RSASSA-PSS

With the RSASSA-PSS the algorithm object identifier is always id-RSASSA-PSS, but the hash function is taken from the parameters, and it is required. See [RFC4055] for more information.

A.4.1. RSASSA-PSS with empty parameters

id-RSASSA-PSS OBJECT IDENTIFIER ::= { pkcs-1 10 }

Parameters are empty, but the ASN.1 part of the sequence must be there. This means default parameters are used (same as the next example).

0000 : SEQUENCE
0002 : OBJECT IDENTIFIER RSASSA-PSS (1.2.840.113549.1.1.10)
000d : SEQUENCE

Length = 15
0000: 300d 0609 2a86 4886 f70d 0101 0a30 00

A.4.2. RSASSA-PSS with default parameters

```
id-RSASSA-PSS OBJECT IDENTIFIER ::= { pkcs-1 10 }

Here the parameters are present, and contains the default parameters,
i.e.  SHA-1, mgf1SHA1, saltlength of 20, trailerfield of 1.

0000 : SEQUENCE
0002 :   OBJECT IDENTIFIER  RSASSA-PSS (1.2.840.113549.1.1.10)
000d :   SEQUENCE
000f :     CONTEXT 0
0011 :       SEQUENCE
0013 :         OBJECT IDENTIFIER  Sha-1 (1.3.14.3.2.26)
001a :         NULL
001c :     CONTEXT 1
001e :       SEQUENCE
0020 :         OBJECT IDENTIFIER  1.2.840.113549.1.1.8
002b :         SEQUENCE
002d :           OBJECT IDENTIFIER  Sha-1 (1.3.14.3.2.26)
0034 :           NULL
0036 :     CONTEXT 2
0038 :       INTEGER   0x14 (5 bits)
003b :     CONTEXT 3
003d :       INTEGER   0x1 (1 bits)


Name = RSASSA-PSS with default parameters,
       oid = 1.2.840.113549.1.1.10
Length = 64
0000: 303e 0609 2a86 4886 f70d 0101 0a30 31a0
0010: 0b30 0906 052b 0e03 021a 0500 a118 3016
0020: 0609 2a86 4886 f70d 0101 0830 0906 052b
0030: 0e03 021a 0500 a203 0201 14a3 0302 0101
```

A.4.3. RSASSA-PSS with SHA-256

```
id-RSASSA-PSS OBJECT IDENTIFIER ::= { pkcs-1 10 }

Here the parameters are present, and contains the SHA-256 for both
the hash and mgf, saltlength of 32, and trailerfield of 1.
```

```
0000 : SEQUENCE
0002 :   OBJECT IDENTIFIER  RSASSA-PSS (1.2.840.113549.1.1.10)
000d :   SEQUENCE
000f :     CONTEXT 0
0011 :       SEQUENCE
0013 :         OBJECT IDENTIFIER  Sha-256 (2.16.840.1.101.3.4.2.1)
001e :         NULL
0020 :     CONTEXT 1
0022 :       SEQUENCE
0024 :         OBJECT IDENTIFIER  1.2.840.113549.1.1.8
002f :         SEQUENCE
0031 :           OBJECT IDENTIFIER  Sha-256 (2.16.840.1.101.3.4.2.1)
003c :           NULL
003e :     CONTEXT 2
0040 :       INTEGER   0x20 (6 bits)
0043 :     CONTEXT 3
0045 :       INTEGER   0x1 (1 bits)
```

```
Name = RSASSA-PSS with sha-256, oid = 1.2.840.113549.1.1.10
Length = 72
0000: 3046 0609 2a86 4886 f70d 0101 0a30 39a0
0010: 0f30 0d06 0960 8648 0165 0304 0201 0500
0020: a11c 301a 0609 2a86 4886 f70d 0101 0830
0030: 0d06 0960 8648 0165 0304 0201 0500 a203
0040: 0201 20a3 0302 0101
```

Appendix B. IKEv2 Payload Example

B.1. sha1WithRSAEncryption

The IKEv2 AUTH payload would start like this:

```
00000000: NN00 00LL XX00 0000 0f30 0d06 092a 8648
00000010: 86f7 0d01 0105 0500 ....
```

Where the NN will be the next payload type (i.e. that value depends on what is the next payload after this Authentication payload), the LL will be the length of this payload, and after the sha1WithRSAEncryption ASN.1 block (15 bytes) there will be the actual signature, which is omitted here.

Note to the RFC editor / IANA, replace the XX above with the newly allocated authentication method type for Digital Signature, and remove this note.

Author's Address

Tero Kivinen
INSIDE Secure
Eerikinkatu 28
HELSINKI FI-00180
FI

Email: kivinen@iki.fi