         Signature Authentication in the Internet Key Exchange Version 2 (IKEv2)

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 a fixed hash algorithm tied to each group. This
   document generalizes IKEv2 signature support to allow any signature
   method supported by PKIX and also adds signature hash algorithm
   negotiation. This is a generic mechanism and is not limited to
   ECDSA; it can also be used with other signature algorithms.

Status of This Memo

   This is an Internet Standards Track document.

   This document is a product of the Internet Engineering Task Force
   (IETF). It represents the consensus of the IETF community. It has
   received public review and has been approved for publication by the
   Internet Engineering Steering Group (IESG). Further information on
   Internet Standards is available in Section 2 of RFC 5741.

   Information about the current status of this document, any errata,
   and how to provide feedback on it may be obtained at
   http://www.rfc-editor.org/info/rfc7427.

Table of Contents

1. Introduction

   This document adds a new IKEv2 [RFC7296] authentication method to
   support signature methods in a more general way. The current
   signature-based authentication methods in 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 is cumbersome when
   more signature algorithms, hash algorithms, and elliptic curves need
   to be supported:

o In IKEv2, authentication using RSA digital signatures calls for
   padding based on RSASSA-PKCS1-v1_5, although the newer RSASSA-PSS
   padding method is now recommended. (See Section 5 of "Additional
   Algorithms and Identifiers for RSA Cryptography for use in PKIX
   Profile" [RFC4055].)

o With ECDSA and the Digital Signature Standard (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 SHA-1.

o The tying of ECDSA authentication methods to particular elliptic
   curve groups requires definition of additional methods for each
   new group. The combination of new ECDSA groups and hash functions
   will cause the number of required authentication methods to become
   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 might be possible that a signature
method can be used with either SHA-3 or SHA-2. This means that a new
mechanism for negotiating the hash algorithm for a signature
algorithm is needed.

This document specifies two things:

1. A new authentication method that includes enough information
    inside the Authentication payload data so the signature hash
    algorithm can be extracted (see Section 3).

2. A method to indicate supported signature hash algorithms (see
    Section 4). This allows the 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 requirement to actually
    negotiate one common hash algorithm, as different hash algorithms
    can be used in different directions if needed.

The new digital signature method is flexible enough to include all
current signature methods (RSA, DSA, ECDSA, RSASSA-PSS, etc.) and add
new methods (ECGDSA, ElGamal, etc.) in the future. To support this
flexibility, the signature algorithm is specified in the same way
that PKIX [RFC5280] specifies the signature of the Digital
Certificate, by placing a simple ASN.1 object before the actual
signature data. This ASN.1 object contains an OID specifying the
algorithm and associated parameters. When an IKEv2 implementation

supports a fixed set of signature methods with commonly used
parameters, it is acceptable for the implementation to treat the
ASN.1 object as a binary blob that can be compared against the fixed
set of known values. IKEv2 implementations can also parse the ASN.1
and extract the signature algorithm and associated parameters.

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 a new "Digital Signature" authentication
method. This method can be used with any type of signature. As the
authentication methods are not negotiated in IKEv2, the peer is only
allowed to use this authentication method if the Notify payload of
type SIGNATURE_HASH_ALGORITHMS has been sent and received by each
peer.

In this authentication method, the Authentication Data field inside
the Authentication payload does not just include the signature value,
as do other existing IKEv2 Authentication payloads. Instead, the
signature value is prefixed with an ASN.1 object indicating the
algorithm used to generate the signature. The ASN.1 object contains
the algorithm identification OID, which identifies both the signature
algorithm and the hash used when calculating the signature. In
addition to the OID, the ASN.1 object can contain optional parameters
that might be needed for algorithms such as RSASSA-PSS (see
Section 8.1 of [RFC3447]).

To make implementations easier, the ASN.1 object is prefixed by the
8-bit length field. This length field allows simple implementations
to know the length of the ASN.1 object without the need to parse it,
so they can use it as a binary blob to be compared against known
signature algorithm ASN.1 objects. Thus, simple implementations may
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 used in the AlgorithmIdentifier
of PKIX (see 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, both of which are needed for signature verification.

Currently, only the RSASSA-PSS signature algorithm uses the optional
parameters. For other signature algorithms, the parameters are

either NULL or missing. Note that for some algorithms there are two
possible ASN.1 encodings, one with optional parameters included but
set to NULL and the other where the optional parameters are omitted.
These dual encodings exist because of the way those algorithms are
specified. When encoding the ASN.1, implementations SHOULD use the
preferred format called for by the algorithm specification. If the
algorithm specification says "preferredPresent", then the parameters
object needs to be present, although it will be NULL if no parameters
are specified. If the algorithm specification says
"preferredAbsent", then the entire optional parameters object is
missing.

The Authentication payload is defined in IKEv2 as follows:

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Next Payload  |C|  RESERVED   |         Payload Length        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Auth Method   |                RESERVED                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
~                     Authentication Data                       ~
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
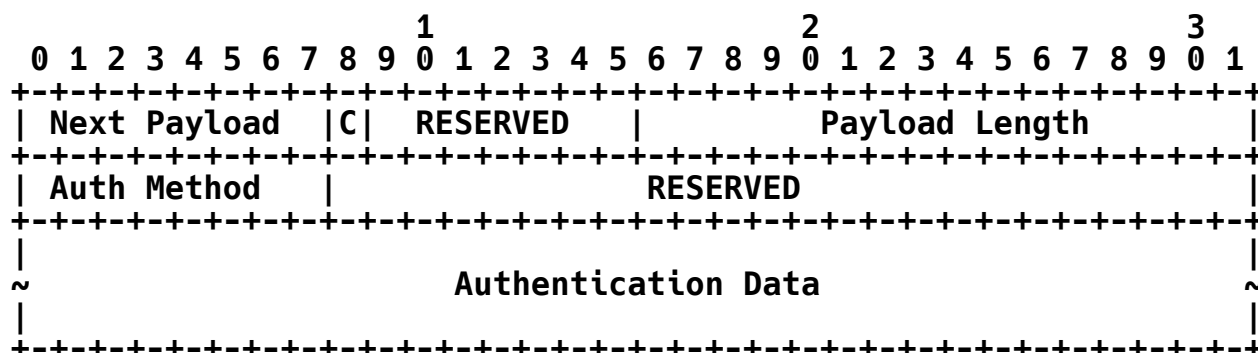
Figure 1: Authentication Payload Format

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

```
   Mechanism                                  Value
   ------------------------------------------------------------------
   Digital Signature                          14
```

   Computed as specified in Section 2.15 of [RFC7296] using a private
   key associated with the public key sent in the Certificate payload
   and using one of the hash algorithms sent by the other end in the
   Notify payload of type SIGNATURE_HASH_ALGORITHMS. If both ends
   send and receive SIGNATURE_HASH_ALGORITHMS Notify payloads, and
   signature authentication is to be used, then the authentication
   method specified in this Authentication payload MUST be used. The
   format of the Authentication Data field is different from other
   Authentication methods and is specified below.

o Authentication Data (variable length) - See Section 2.15 of
   [RFC7296]. For "Digital Signature" format, the Authentication
   Data is formatted as follows:

```
                        1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   | ASN.1 Length  | AlgorithmIdentifier ASN.1 object              |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   ~           AlgorithmIdentifier ASN.1 object continuing         ~
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   ~                       Signature Value                         ~
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
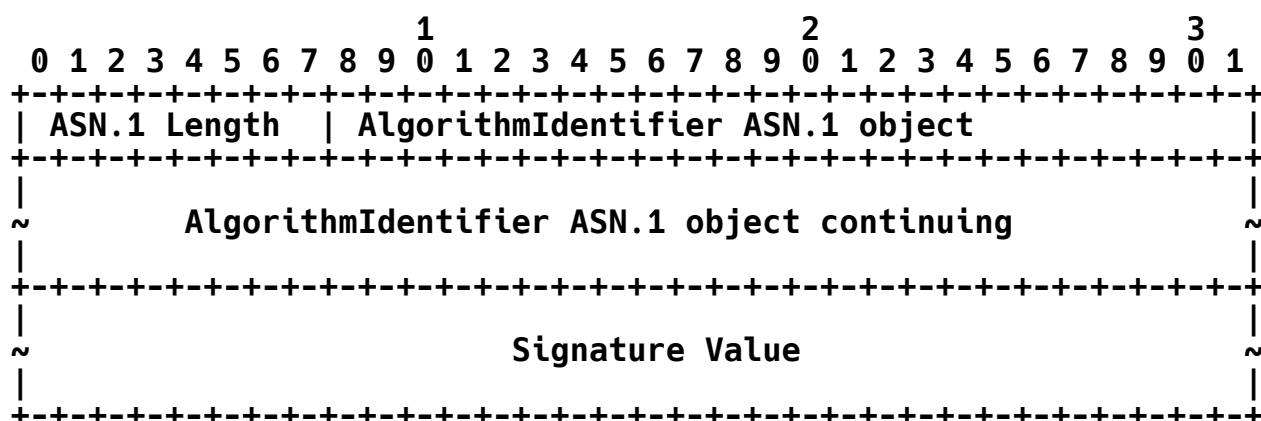
                 Figure 2: Authentication Data Format

   * ASN.1 Length (1 octet) - This field contains the length of the
      ASN.1-encoded AlgorithmIdentifier object.

   * Algorithm Identifier (variable length) - This field contains
      the AlgorithmIdentifier ASN.1 object.

   * Signature Value (variable length) - This field contains the
      actual signature value.

   There is no padding between the ASN.1 object and the signature
   value. For hash truncation, the method specified in ANSI
   X9.62:2005 [X9.62] MUST be used.

4. Hash Algorithm Notification

   The supported hash algorithms that can be used for the signature
   algorithms are indicated with a Notify payload of type
   SIGNATURE_HASH_ALGORITHMS sent inside the IKE_SA_INIT exchange.

   This notification also implicitly indicates support of the new
   "Digital Signature" algorithm method, as well as the list of hash
   functions supported by the sending peer.

   Both ends send their list of supported hash algorithms. When
   calculating the digital 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 indicating the selected
   hash algorithm (and signature algorithm) used when calculating the
   signature is sent inside the Authentication Data field of the
   Authentication payload (with Auth Method of "Digital Signature" as
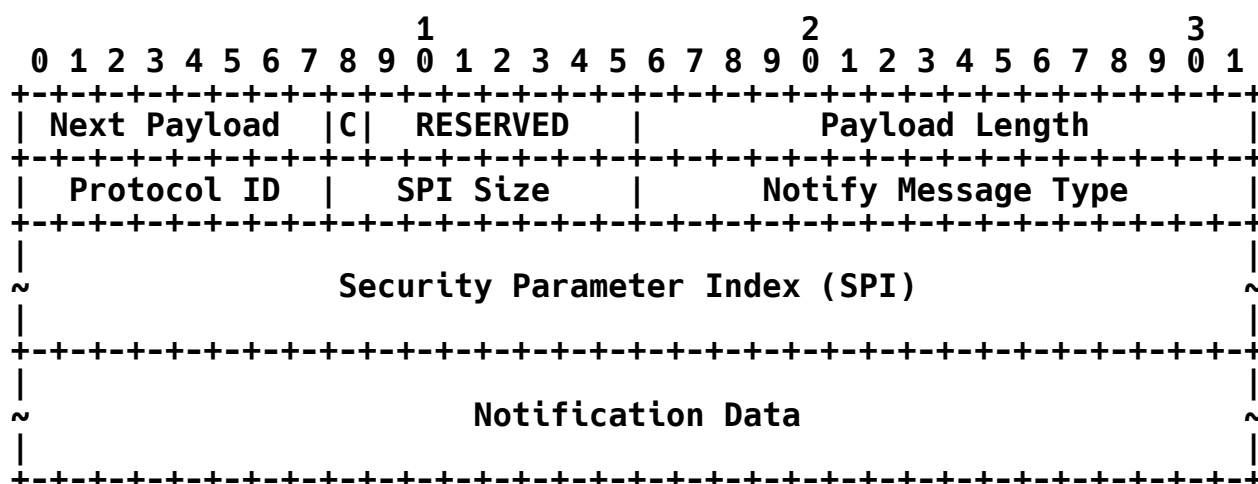   defined above).

```
                            1                   2                   3
        0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       | Next Payload  |C|  RESERVED   |         Payload Length        |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |  Protocol ID  |   SPI Size    |      Notify Message Type      |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |                                                               |
       ~              Security Parameter Index (SPI)                   ~
       |                                                               |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |                                                               |
       ~                     Notification Data                         ~
       |                                                               |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 3: Notify Payload Format

The Notify payload format is defined in Section 3.10 of [RFC7296].
When a Notify payload of type SIGNATURE_HASH_ALGORITHMS is sent, the
Protocol ID field is set to 0, the SPI Size is set to 0, and the
Notify Message Type is set to 16431.

The Notification Data field contains the list of 16-bit hash
algorithm identifiers from the Hash Algorithm Identifiers of IANA's
"Internet Key Exchange Version 2 (IKEv2) Parameters" registry. There
is no padding between the hash algorithm identifiers.

5. Selecting the Public Key Algorithm

   This specification does not provide a way for the peers to indicate
   the public/private key pair types they have. This raises the
   question of how the responder selects a public/private key pair type
   that the initiator supports. This information can be found by
   several methods.

   One method to signal the key the initiator wants the responder to use
   is to indicate that in the IDr (Identification - Responder) payload
   of the IKE_AUTH request sent by the initiator. In this case, the
   initiator indicates that it wants the responder to use a particular
   public/private key pair by sending an IDr payload that indicates that
   information. In this case, the responder has different identities
   configured, with each of those identities associated to a public/
   private key or key type.

   Another method to ascertain the key the initiator wants the responder
   to use is through a Certificate Request payload sent by the
   initiator. For example, the initiator could indicate in the

Certificate Request payload that it trusts a certificate authority
certificate signed by an ECDSA key. This indication implies that the
initiator can process ECDSA signatures, which means that the
responder can safely use ECDSA keys when authenticating.

A third method is for the responder to check the key type used by the
initiator and use the same key type that the initiator used. This
method does not work if the initiator is using shared secret or
Extensible Authentication Protocol (EAP) authentication (i.e., is not
using public keys). If the initiator is using public key
authentication, this method is the best way for the responder to
ascertain the type of key the initiator supports.

If the initiator uses a public key type that the responder does not
support, the responder replies with a Notify message with error type
AUTHENTICATION_FAILED. If the initiator has multiple different keys,
it may try a different key (and perhaps a different key type) until
it finds a key that the other end accepts. The initiator can also
use the Certificate Request payload sent by the responder to help
decide which public key should be tried. In normal cases, when the
initiator has multiple public keys, out-of-band configuration is used
to select a public key for each connection.

6. Security Considerations

Tables 2 and 3 of the "Recommendations for Key Management"
[NIST800-57] give recommendations for how to select suitable hash
functions for the signature.

This new digital signature method does not tie the Elliptic Curve to
a specific hash function, which was done in the old IKEv2 ECDSA
methods. This means it is possible to mix different security levels.
For example, it is possible to use a 512-bit Elliptic Curve with
SHA1. This means that the security of the authentication method is
the security of the weakest component (signature algorithm, hash
algorithm, or curve). This complicates the security analysis of the
system.

IKEv2 peers have a series of policy databases (see Section 4.4 of
[RFC4301]) that define which security algorithms and methods should
be used during establishment of security associations. To help end
users select the desired security levels for communications protected
by IPsec, implementers may wish to provide a mechanism in the IKE
policy databases to limit the mixing of security levels or to
restrict combinations of protocols.

Security downgrade attacks, where more secure methods are deleted or
modified from a payload by a man-in-the-middle to force lower levels

of security, are not a significant concern in IKEv2 Authentication payloads, as discussed in this RFC. This is because a modified AUTH payload will be detected when the peer computes a signature over the IKE messages.

One specific class of downgrade attacks requires selection of catastrophically weak ciphers. In this type of attack, the man-in-the-middle attacker is able to "break" the cryptography in real time. This type of downgrade attack should be blocked by policy regarding cipher algorithm selection, as discussed above.

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

The current IKEv2 protocol uses RSASSA-PKCS1-v1_5, which has known security vulnerabilities [KA08] [ME01] and does not allow using newer padding methods such as RSASSA-PSS. The new method described in this RFC allows the use of other padding methods.

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

7. IANA Considerations

This document creates a new IANA registry for IKEv2 Hash Algorithms. Changes and additions to this registry are by Expert Review [RFC5226].

The initial values of this registry are:

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

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

Values 5-1023 are Unassigned. Values 1024-65535 are reserved for Private Use among mutually consenting parties.

This specification also adds a new value for
SIGNATURE_HASH_ALGORITHMS (16431) to the "IKEv2 Notify Message Types
- Status Types" registry and adds a new value for Digital Signature
(14) to the "IKEv2 Authentication Method" registry.

8. References

8.1. Normative References

[RFC2119]   Bradner, S., "Key words for use in RFCs to Indicate
            Requirement Levels", BCP 14, RFC 2119, March 1997,
            <http://www.rfc-editor.org/info/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, May 2008,
            <http://www.rfc-editor.org/info/rfc5280>.

[RFC7296]   Kaufman, C., Hoffman, P., Nir, Y., Eronen, P., and T.
            Kivinen, "Internet Key Exchange Protocol Version 2
            (IKEv2)", RFC 7296, October 2014,
            <http://www.rfc-editor.org/info/rfc7296>.

8.2. 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", Proceedings of Sicherheit 2008,
            pp.97-109, 2008.

[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,
            "Recommendation for Key Management - Part 1: General
            (Revised)", NIST Special Publication 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,
              <http://www.rfc-editor.org/info/rfc3279>.

   [RFC3447] Jonsson, J. and B. Kaliski, "Public-Key Cryptography
              Standards (PKCS) #1: RSA Cryptography Specifications
              Version 2.1", RFC 3447, February 2003,
              <http://www.rfc-editor.org/info/rfc3447>.

   [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, <http://www.rfc-editor.org/info/rfc4055>.

   [RFC4301] Kent, S. and K. Seo, "Security Architecture for the
              Internet Protocol", RFC 4301, December 2005,
              <http://www.rfc-editor.org/info/rfc4301>.

   [RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an
              IANA Considerations Section in RFCs)", BCP 26, RFC 5226,
              May 2008, <http://www.rfc-editor.org/info/rfc5226>.

   [RFC5480] Turner, S., Brown, D., Yiu, K., Housley, R., and T. Polk,
              "Elliptic Curve Cryptography Subject Public Key
              Information", RFC 5480, March 2009,
              <http://www.rfc-editor.org/info/rfc5480>.

   [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,
              <http://www.rfc-editor.org/info/rfc5758>.

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

   [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. If the ASN.1 object listed in Appendix A and the ASN.1
   object specified by the algorithm differ, then the algorithm
   specification must be used. These values are taken from "New ASN.1
   Modules for the Public Key Infrastructure Using X.509 (PKIX)"
   [RFC5912].

A.1. PKCS#1 1.5 RSA Encryption

   The algorithm identifiers here include several different ASN.1
   objects with different hash algorithms. This document only includes
   the commonly used ones, i.e., the ones using SHA-1 or SHA-2 as the
   hash function. Some other algorithms (such as MD2 and MD5) are not
   safe enough to be used as signature hash algorithms and are omitted.
   The IANA registry does not have code points for these other
   algorithms with RSA Encryption. Note that there are no optional
   parameters in any of these algorithm identifiers, but all included
   here need NULL optional parameters present in the ASN.1.

   See "Algorithms and Identifiers for PKIX Profile" [RFC3279] and
   "Additional Algorithms and Identifiers for RSA Cryptography for use
   in the Internet X.509 Public Key Infrastructure Certificate and
   Certificate Revocation List (CRL) 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 DSA algorithms, optional parameters are always omitted. Only
   algorithm combinations for DSA that are listed in the IANA registry
   are included.

   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 ECDSA algorithms, the optional parameters are always omitted.
   Only algorithm combinations for the ECDSA listed in the IANA registry
   are included.

   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 RSASSA-PSS, the algorithm object identifier must always be
   id-RSASSA-PSS, and the hash function and padding parameters are
   conveyed in the parameters (which are not optional in this case).
   See Additional RSA Algorithms and Identifiers [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
   present. This means default parameters are used.

   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 contain the default parameters,
   i.e., hashAlgorithm of SHA-1, maskGenAlgorithm of mgf1SHA1,
   saltLength of 20, 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 id-sha1 (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 id-sha1 (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 contain hashAlgorithm of SHA-256,
   maskGenAlgorithm of SHA-256, 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 id-sha256 (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 id-sha256 (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 0e00 0000 0f30 0d06 092a 8648
   00000010: 86f7 0d01 0105 0500 ....

   Where the NN will be the next payload type (i.e., the value depends
   on 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.

Authors' Addresses

   Tero Kivinen
   INSIDE Secure
   Eerikinkatu 28
   Helsinki FI-00180
   Finland

   EMail: kivinen@iki.fi

   Joel Snyder
   Opus One
   1404 East Lind Road
   Tucson, AZ 85719

   Phone: +1 520 324 0494
   EMail: jms@opus1.com