        Use of Edwards-Curve Digital Signature Algorithm (EdDSA) Signatures
                  in the Cryptographic Message Syntax (CMS)

Abstract

   This document specifies the conventions for using the Edwards-curve
   Digital Signature Algorithm (EdDSA) for curve25519 and curve448 in
   the Cryptographic Message Syntax (CMS). For each curve, EdDSA
   defines the PureEdDSA and HashEdDSA modes. However, the HashEdDSA
   mode is not used with the CMS. In addition, no context string is
   used with the CMS.

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 7841.

   Information about the current status of this document, any errata,
   and how to provide feedback on it may be obtained at
   https://www.rfc-editor.org/info/rfc8419.

Table of Contents

1. Introduction

   This document specifies the conventions for using the Edwards-curve
   Digital Signature Algorithm (EdDSA) [RFC8032] for curve25519
   [CURVE25519] and curve448 [CURVE448] with the Cryptographic Message
   Syntax (CMS) [RFC5652] signed-data content type. For each curve,
   [RFC8032] defines the PureEdDSA and HashEdDSA modes; however, the
   HashEdDSA mode is not used with the CMS. In addition, no context
   string is used with CMS. EdDSA with curve25519 is referred to as
   "Ed25519", and EdDSA with curve448 is referred to as "Ed448". The
   CMS conventions for PureEdDSA with Ed25519 and Ed448 are described in
   this document.

1.1. Terminology

   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.

1.2. ASN.1

   CMS values are generated using ASN.1 [X680], which uses the Basic
   Encoding Rules (BER) and the Distinguished Encoding Rules (DER)
   [X690].

2. EdDSA Signature Algorithm

   The Edwards-curve Digital Signature Algorithm (EdDSA) [RFC8032] is a
   variant of Schnorr's signature system with (possibly twisted) Edwards
   curves. Ed25519 is intended to operate at around the 128-bit
   security level; Ed448 is intended to operate at around the 224-bit
   security level.

   One of the parameters of the EdDSA algorithm is the "prehash"
   function. This may be the identity function, resulting in an
   algorithm called "PureEdDSA", or a collision-resistant hash function,
   resulting in an algorithm called "HashEdDSA". In most situations,
   the CMS SignedData includes signed attributes, including the message
   digest of the content. Since HashEdDSA offers no benefit when signed
   attributes are present, only PureEdDSA is used with the CMS.

2.1. Algorithm Identifiers

   Each algorithm is identified by an object identifier, and the
   algorithm identifier may contain parameters if needed.

   The ALGORITHM definition is repeated here for convenience:

      ALGORITHM ::= CLASS {
          &id   OBJECT IDENTIFIER UNIQUE,
          &Type OPTIONAL }
        WITH SYNTAX {
          OID &id [PARMS &Type] }

2.2. EdDSA Algorithm Identifiers

   The EdDSA signature algorithm is defined in [RFC8032], and the
   conventions for encoding the public key are defined in [RFC8410].

   The id-Ed25519 and id-Ed448 object identifiers are used to identify
   EdDSA public keys in certificates. The object identifiers are
   specified in [RFC8410], and they are repeated here for convenience:

      sigAlg-Ed25519 ALGORITHM ::= { OID id-Ed25519 }

      sigAlg-Ed448   ALGORITHM ::= { OID id-Ed448 }

      id-Ed25519 OBJECT IDENTIFIER ::= { 1 3 101 112 }

      id-Ed448   OBJECT IDENTIFIER ::= { 1 3 101 113 }

2.3. Message Digest Algorithm Identifiers

   When the signer includes signed attributes, a message digest
   algorithm is used to compute the message digest on the eContent
   value. When signing with Ed25519, the message digest algorithm MUST
   be SHA-512 [FIPS180]. Additional information on SHA-512 is available
   in [RFC6234]. When signing with Ed448, the message digest algorithm
   MUST be SHAKE256 [FIPS202] with a 512-bit output value.

   Signing with Ed25519 uses SHA-512 as part of the signing operation,
   and signing with Ed448 uses SHAKE256 as part of the signing
   operation.

   For convenience, the object identifiers and parameter syntax for
   these algorithms are repeated here:

      hashAlg-SHA-512 ALGORITHM ::= { OID id-sha512 }

      hashAlg-SHAKE256 ALGORITHM ::= { OID id-shake256 }
      hashAlg-SHAKE256-LEN ALGORITHM ::= { OID id-shake256-len
                          PARMS ShakeOutputLen }

      hashalgs OBJECT IDENTIFIER ::= { joint-iso-itu-t(2)
                          country(16) us(840) organization(1)
                          gov(101) csor(3) nistalgorithm(4) 2 }

      id-sha512 OBJECT IDENTIFIER ::= { hashAlgs 3 }

      id-shake256 OBJECT IDENTIFIER ::= { hashAlgs 12 }

      id-shake256-len OBJECT IDENTIFIER ::= { hashAlgs 18 }

      ShakeOutputLen ::= INTEGER -- Output length in bits

   When using the id-sha512 or id-shake256 algorithm identifier, the
   parameters MUST be absent.

   When using the id-shake256-len algorithm identifier, the parameters
   MUST be present, and the parameter MUST contain 512, encoded as a
   positive integer value.

2.4. EdDSA Signatures

   The id-Ed25519 and id-Ed448 object identifiers are also used for
   signature values. When used to identify signature algorithms, the
   AlgorithmIdentifier parameters field MUST be absent.

The data to be signed is processed using PureEdDSA, and then a
private key operation generates the signature value. As described in
Section 3.3 of [RFC8032], the signature value is the opaque value
ENC(R) || ENC(S), where || represents concatenation. As described in
Section 5.3 of [RFC5652], the signature value is ASN.1 encoded as an
OCTET STRING and included in the signature field of SignerInfo.

3. Signed-data Conventions

   The processing depends on whether the signer includes signed
   attributes.

   The inclusion of signed attributes is preferred, but the conventions
   for signed-data without signed attributes are provided for
   completeness.

3.1. Signed-data Conventions with Signed Attributes

   The SignedData digestAlgorithms field includes the identifiers of the
   message digest algorithms used by one or more signer. There MAY be
   any number of elements in the collection, including zero. When
   signing with Ed25519, the digestAlgorithm SHOULD include id-sha512,
   and if present, the algorithm parameters field MUST be absent. When
   signing with Ed448, the digestAlgorithm SHOULD include
   id-shake256-len, and if present, the algorithm parameters field MUST
   also be present, and the parameter MUST contain 512, encoded as a
   positive integer value.

   The SignerInfo digestAlgorithm field includes the identifier of the
   message digest algorithms used by the signer. When signing with
   Ed25519, the digestAlgorithm MUST be id-sha512, and the algorithm
   parameters field MUST be absent. When signing with Ed448, the
   digestAlgorithm MUST be id-shake256-len, the algorithm parameters
   field MUST be present, and the parameter MUST contain 512, encoded as
   a positive integer value.

   The SignerInfo signedAttributes MUST include the message-digest
   attribute as specified in Section 11.2 of [RFC5652]. When signing
   with Ed25519, the message-digest attribute MUST contain the message
   digest computed over the eContent value using SHA-512. When signing
   with Ed448, the message-digest attribute MUST contain the message
   digest computed over the eContent value using SHAKE256 with an output
   length of 512 bits.

   The SignerInfo signatureAlgorithm field MUST contain either
   id-Ed25519 or id-Ed448, depending on the elliptic curve that was used
   by the signer. The algorithm parameters field MUST be absent.

   The SignerInfo signature field contains the octet string resulting
   from the EdDSA private key signing operation.

3.2. Signed-data Conventions without Signed Attributes

   The SignedData digestAlgorithms field includes the identifiers of the
   message digest algorithms used by one or more signer. There MAY be
   any number of elements in the collection, including zero. When
   signing with Ed25519, the list of identifiers MAY include id-sha512,
   and if present, the algorithm parameters field MUST be absent. When
   signing with Ed448, the list of identifiers MAY include id-shake256,
   and if present, the algorithm parameters field MUST be absent.

   The SignerInfo digestAlgorithm field includes the identifier of the
   message digest algorithms used by the signer. When signing with
   Ed25519, the digestAlgorithm MUST be id-sha512, and the algorithm
   parameters field MUST be absent. When signing with Ed448, the
   digestAlgorithm MUST be id-shake256, and the algorithm parameters
   field MUST be absent.

      NOTE: Either id-sha512 or id-shake256 is used as part to the
      private key signing operation. However, the private key signing
      operation does not take a message digest computed with one of
      these algorithms as an input.

   The SignerInfo signatureAlgorithm field MUST contain either
   id-Ed25519 or id-Ed448, depending on the elliptic curve that was used
   by the signer. The algorithm parameters field MUST be absent.

   The SignerInfo signature field contains the octet string resulting
   from the EdDSA private key signing operation.

4. Implementation Considerations

   The EdDSA specification [RFC8032] includes the following warning. It
   deserves highlighting, especially when signed-data is used without
   signed attributes and the content to be signed might be quite large:

      PureEdDSA requires two passes over the input. Many existing APIs,
      protocols, and environments assume digital signature algorithms
      only need one pass over the input and may have API or bandwidth
      concerns supporting anything else.

5. Security Considerations

   Implementations must protect the EdDSA private key. Compromise of
   the EdDSA private key may result in the ability to forge signatures.

The generation of EdDSA private key relies on random numbers. The
use of inadequate pseudo-random number generators (PRNGs) to generate
these values can result in little or no security. An attacker may
find it much easier to reproduce the PRNG environment that produced
the keys, searching the resulting small set of possibilities, rather
than brute-force searching the whole key space. The generation of
quality random numbers is difficult. RFC 4086 [RANDOM] offers
important guidance in this area.

Unlike DSA and Elliptic Curve Digital Signature Algorithm (ECDSA),
EdDSA does not require the generation of a random value for each
signature operation.

Using the same private key with different algorithms has the
potential to leak extra information about the private key to an
attacker. For this reason, the same private key SHOULD NOT be used
with more than one set of EdDSA parameters, although it appears that
there are no security concerns when using the same private key with
PureEdDSA and HashEdDSA [RFC8032].

When computing signatures, the same hash function SHOULD be used for
all operations. This reduces the number of failure points in the
signature process.

6. IANA Considerations

This document has no IANA actions.

7. References

7.1. Normative References

   [FIPS180]   National Institute of Standards and Technology, "Secure
                 Hash Standard (SHS)", FIPS PUB 180-4,
                 DOI 10.6028/NIST.FIPS.180-4, August 2015.

   [FIPS202]   National Institute of Standards and Technology, "SHA-3
                 Standard: Permutation-Based Hash and Extendable-Output
                 Functions", FIPS PUB 202, DOI 10.6028/NIST.FIPS.202,
                 August 2015.

   [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>.

   [RFC5652]   Housley, R., "Cryptographic Message Syntax (CMS)",
               STD 70, RFC 5652, DOI 10.17487/RFC5652, September 2009,
               <https://www.rfc-editor.org/info/rfc5652>.

   [RFC8032]   Josefsson, S. and I. Liusvaara, "Edwards-Curve Digital
               Signature Algorithm (EdDSA)", RFC 8032,
               DOI 10.17487/RFC8032, January 2017,
               <https://www.rfc-editor.org/info/rfc8032>.

   [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/info/rfc8174>.

   [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>.

   [X680]   ITU-T, "Information technology -- Abstract Syntax
               Notation One (ASN.1): Specification of basic notation",
               ITU-T Recommendation X.680, ISO/IEC 8824-1, August 2015,
               <https://www.itu.int/rec/T-REC-X.680/en>.

   [X690]   ITU-T, "Information technology -- ASN.1 encoding rules:
               Specification of Basic Encoding Rules (BER), Canonical
               Encoding Rules (CER) and Distinguished Encoding Rules
               (DER)", ITU-T Recommendation X.690, ISO/IEC 8825-1,
               August 2015, <https://www.itu.int/rec/T-REC-X.690/en>.

7.2. Informative References

   [CURVE25519] Bernstein, D., "Curve25519: new Diffie-Hellman speed
               records", DOI 10.1007/11745853_14, February 2006,
               <http://cr.yp.to/ecdh.html>.

   [CURVE448] Hamburg, M., "Ed448-Goldilocks, a new elliptic curve",
               June 2015, <http://eprint.iacr.org/2015/625>.

   [RANDOM]   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>.

   [RFC6234]   Eastlake 3rd, D. and T. Hansen, "US Secure Hash
               Algorithms (SHA and SHA-based HMAC and HKDF)", RFC 6234,
               DOI 10.17487/RFC6234, May 2011,
               <https://www.rfc-editor.org/info/rfc6234>.

Acknowledgements

Author's Address

   Russ Housley
   918 Spring Knoll Drive
   Herndon, VA 20170
   United States of America

   Email: housley@vigilsec.com