Using ChaCha20-Poly1305 Authenticated Encryption
in the Cryptographic Message Syntax (CMS)

<draft-ietf-curdle-cms-chacha20-poly1305-04.txt>

Abstract

   This document describes the conventions for using ChaCha20-Poly1305
   Authenticated Encryption in the Cryptographic Message Syntax (CMS).
   ChaCha20-Poly1305 is an authenticated encryption algorithm
   constructed of the ChaCha stream cipher and Poly1305 authenticator.

Status of This Memo

Copyright Notice

Internet-Draft   1 December 2016

1. Introduction

   This document specifies the conventions for using the
   ChaCha20-Poly1305 Authenticated Encryption as the content-
   authenticated-encryption algorithm with the Cryptographic Message
   Syntax (CMS) [CMS] authenticated-enveloped-data content type
   [AUTHENV].

   ChaCha [CHACHA] is a stream cipher developed by D. J. Bernstein in
   2008. It is a refinement of Salsa20, which is one of the ciphers in
   the eSTREAM portfolio [ESTREAM].

   ChaCha20 is the 20-round variant of ChaCha; it requires a 256-bit key
   and a 96-bit nonce. ChaCha20 is described in [FORIETF].

   Poly1305 [POLY1305] is a Wegman-Carter, one-time authenticator
   designed by D. J. Bernstein. Poly1305 produces a 16-byte
   authentication tag; it requires a 256-bit, single-use key. Poly1305
   is also described in [FORIETF].

   ChaCha20 and Poly1305 have been designed for high performance
   software implementations. They can typically be implemented with few
   resources and inexpensive operations, making them suitable on a wide
   range of systems. They have also been designed to minimize leakage
   of information through side channels.

1.1. The ChaCha20 and Poly1305 AEAD Construction

   ChaCha20 and Poly1305 have been combined to create an Authenticated
   Encryption with Associated Data (AEAD) algorithm [AEAD]. This AEAD
   algorithm is often referred to as AEAD_CHACHA20_POLY1305, and it is
   described [FORIETF].

   AEAD_CHACHA20_POLY1305 accepts four inputs: a 256-bit key, a 96-bit
   nonce, an arbitrary length plaintext, and an arbitrary length
   additional authenticated data (AAD). As the name implies, a nonce
   value cannot be used securely more than once with the same key.

   AEAD_CHACHA20_POLY1305 produces two outputs: ciphertext of the same
   length as the plaintext and a 128-bit authentication tag.

   AEAD_CHACHA20_POLY1305 authenticated decryption processing is similar
   to the encryption processing. Of course, the roles of ciphertext and

plaintext are reversed, so the ChaCha20 encryption function is
applied to the ciphertext, producing the plaintext. The Poly1305
function is run over the AAD and the ciphertext, not the plaintext,
and the resulting authentication tag is bitwise compared to the
received authentication tag. The message is authenticated if and

only if the calculated and received authentication tags match.

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].

1.3. 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 RFC 2119 [STDWORDS].

2. Key Management

The reuse of an AEAD_CHACHA20_POLY1305 nonce value with the same key
destroys the security guarantees. It can be extremely difficult to
use a statically configured AEAD_CHACHA20_POLY1305 key and never
repeat a nonce value; however, the CMS authenticated-enveloped-data
content type supports four key management techniques that allow a
fresh AEAD_CHACHA20_POLY1305 key to be used as the content-
authenticated-encryption key for a single protected content:

   Key Transport: the fresh content-authenticated-encryption key
      is encrypted in the recipient's public key;

   Key Agreement: the recipient's public key and the sender's
      private key are used to generate a pairwise symmetric key-
      encryption key, then the fresh content-authenticated-encryption
      key is encrypted in the pairwise symmetric key;

   Symmetric Key-Encryption Keys: the fresh content-authenticated-
      encryption key is encrypted in a previously distributed
      symmetric key-encryption key; and

Passwords: the fresh content-authenticated-encryption key is
         encrypted in a key-encryption key that is derived from a
         password or other shared secret value.

   In addition to these four general key management techniques, CMS
   supports other key management techniques. See Section 6.2.5 of
   [CMS]. Since the properties of these key management techniques are
   unknown, no statement about their support of fresh content-
   authenticated-encryption keys can be made. Designers and
   implementers must perform their own analysis if one of these other
   key management techniques is supported.

3. Using the AEAD_CHACHA20_POLY1305 Algorithm with AuthEnvelopedData

   This section specifies the conventions employed by CMS
   implementations that support the authenticated-enveloped-data content
   type and the AEAD_CHACHA20_POLY1305 algorithm.

   The AEAD_CHACHA20_POLY1305 algorithm identifier is located in the
   AuthEnvelopedData EncryptedContentInfo contentEncryptionAlgorithm
   field.

   The AEAD_CHACHA20_POLY1305 algorithm is used to encipher the content
   located in the AuthEnvelopedData EncryptedContentInfo
   encryptedContent field and to provide the message authentication code
   (MAC) located in the AuthEnvelopedData mac field. The ciphertext and
   the MAC are the two outputs of the AEAD_CHACHA20_POLY1305 Algorithm.
   Note that the MAC, which is called the authentication tag in
   [FORIETF], provides integrity protection for both the
   AuthEnvelopedData authAttrs and the AuthEnvelopedData
   EncryptedContentInfo encryptedContent.

   Neither the plaintext content nor the optional AAD inputs need to be
   padded prior to invoking the AEAD_CHACHA20_POLY1305 algorithm.

   There is one algorithm identifier for the AEAD_CHACHA20_POLY1305
   algorithm:

      id-alg-AEADChaCha20Poly1305 OBJECT IDENTIFIER ::=
         { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1)
           pkcs9(9) smime(16) alg(3) TBD1 }

The AlgorithmIdentifier parameters field MUST be present, and the
parameters field must contain a AEADChaCha20Poly1305Nonce:

    AEADChaCha20Poly1305Nonce ::= OCTET STRING (SIZE(12))

The AEADChaCha20Poly1305Nonce contains a 12-octet nonce. With the
CMS, the content-authenticated-encryption key is normally used for a
single content. Within the scope of any content-authenticated-
encryption key, the nonce value MUST be unique. That is, the set of
nonce values used with any given key MUST NOT contain any duplicate
values.

4. S/MIME Capabilities Attribute

   Section 2.5.2 of RFC 5751 [MSG] defines the SMIMECapabilities
   attribute, which is used to specify a partial list of algorithms that
   the software announcing the SMIMECapabilities can support. When
   constructing a CMS signed-data content type, compliant software MAY

   include the SMIMECapabilities signed attribute to announce support
   for the AEAD_CHACHA20_POLY1305 algorithm.

   The SMIMECapability SEQUENCE representing the AEAD_CHACHA20_POLY1305
   algorithm MUST include the id-alg-AEADChaCha20Poly1305 object
   identifier in the capabilityID field and MUST omit the parameters
   field.

   The DER encoding of a SMIMECapability SEQUENCE is the same as the DER
   encoding of an AlgorithmIdentifier. The DER encoding for the
   AEAD_CHACHA20_POLY1305 algorithm in the SMIMECapability SEQUENCE (in
   hexadecimal) is:

       30 0c 06 0b 2a 86 48 86 f7 0d 01 09 10 03 ??

   {{{ Correct above after IANA assigns the object identifier. }}}

5. IANA Considerations

   IANA is requested to add the following entry in the SMI Security for
   S/MIME Algorithms (1.2.840.113549.1.9.16.3) registry:

TBD1   id-alg-AEADChaCha20Poly1305   [This Document]

    IANA is requested to add the following entry in the SMI Security for
    S/MIME Module Identifier (1.2.840.113549.1.9.16.0) registry:

           TBD2   id-mod-CMS-AEADChaCha20Poly1305   [This Document]

6. Security Considerations

    The CMS AuthEnvelopedData provides all of the tools needed to avoid
    reuse of the same nonce value under the same key. See the discussion
    in Section 2 of this document. RFC 7539 [FORIETF] describes the
    consequences of using a nonce value more than once:

        Consequences of repeating a nonce: If a nonce is repeated, then
        both the one-time Poly1305 key and the keystream are identical
        between the
         messages. This reveals the XOR of the plaintexts, because the
        XOR of the plaintexts is equal to the XOR of the ciphertexts.

    When using AEAD_CHACHA20_POLY1305, the resulting ciphertext is always
    the same size as the original plaintext.   Some other mechanism needs
    to be used in conjunction with AEAD_CHACHA20_POLY1305 if disclosure
    of the size of the plaintext is a concern.

    The amount of encrypted data possible in a single invocation of

    AEAD_CHACHA20_POLY1305 is $2^{32}-1$ blocks of 64 octets each, because of
    the size of the block counter field in the ChaCha20 block function.
    This gives a total of 247,877,906,880 octets, which likely to be
    sufficient to handle the size of any CMS content type. Note that
    ciphertext length field in the authentication buffer will accomodate
    2^{64} octets, which is much larger than necessary.

    The AEAD_CHACHA20_POLY1305 construction is a novel composition of
    ChaCha20 and Poly1305. A security analysis of this composition is
    given in [PROCTER].

    Implementations must randomly generate content-authenticated-
    encryption keys. The use of inadequate pseudo-random number
    generators (PRNGs) to generate cryptographic keys 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.

7. Acknowledgements

Thanks to Jim Schaad, Daniel Migault, and Stephen Farrell for their
review and insightful comments.

8. Normative References

[AUTHENV]   Housley, R., "Cryptographic Message Syntax (CMS)
            Authenticated-Enveloped-Data Content Type", RFC 5083,
            November 2007.

[CMS]   Housley, R., "Cryptographic Message Syntax (CMS)", RFC
            5652, September 2009.

[FORIETF]   Nir, Y. and A. Langley, "ChaCha20 and Poly1305 for IETF
            Protocols", RFC 7539, May 2015.

[MSG]   Ramsdell, B., and S. Turner, "Secure/Multipurpose Internet
            Mail Extensions (S/MIME) Version 3.2 Message
            Specification", RFC 5751, January 2010.

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

[X680]   ITU-T, "Information technology -- Abstract Syntax Notation
            One (ASN.1): Specification of basic notation", ITU-T
            Recommendation X.680, 2015.

[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, 2015.

9. Informative References

[AEAD]   McGrew, D., "An Interface and Algorithms for Authenticated

                  Encryption", RFC 5116, January 2008.

   [CHACHA]   Bernstein, D., "ChaCha, a variant of Salsa20", January
               2008,
               <http://cr.yp.to/chacha/chacha-20080128.pdf>.

   [ESTREAM]   Babbage, S., DeCanniere, C., Cantenaut, A., Cid, C.,
               Gilbert, H., Johansson, T., Parker, M., Preneel, B.,
               Rijmen, V., and M. Robshaw, "The eSTREAM Portfolio
               (rev. 1)", September 2008,
               <http://www.ecrypt.eu.org/stream/finallist.html>.

   [POLY1305] Bernstein, D., "The Poly1305-AES message-authentication
               code.", March 2005,
               <http://cr.yp.to/mac/poly1305-20050329.pdf>.

   [PROCTER]   Procter, G., "A Security Analysis of the Composition of
               ChaCha20 and Poly1305", August 2014,
               <http://eprint.iacr.org/2014/613.pdf>.

   [RANDOM]   Eastlake, D., Schiller, J., and S. Crocker, "Randomness
               Recommendations for Security", BCP 106, RFC 4086, June
               2005.

Appendix: ASN.1 Module

```
   CMS-AEADChaCha20Poly1305
      { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1)
        pkcs-9(9) smime(16) modules(0) TBD2 }

   DEFINITIONS IMPLICIT TAGS ::= BEGIN

   IMPORTS
      CONTENT-ENCRYPTION
      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) };

   -- EXPORTS All

   AEADContentEncryptionAlgs CONTENT-ENCRYPTION ::=
      { cea-AEADChaCha20Poly1305, ... }

   cea-AEADChaCha20Poly1305 CONTENT-ENCRYPTION ::= {
      IDENTIFIER id-alg-AEADChaCha20Poly1305
      PARAMS TYPE AEADChaCha20Poly1305Nonce ARE required
      SMIME-CAPS { IDENTIFIED BY id-alg-AEADChaCha20Poly1305 } }

   id-alg-AEADChaCha20Poly1305 OBJECT IDENTIFIER ::=
      { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1)
        pkcs9(9) smime(16) alg(3) TBD1 }

   AEADChaCha20Poly1305Nonce ::= OCTET STRING (SIZE(12))

   END
```

Author's Address

   Russell Housley
   Vigil Security, LLC
   918 Spring Knoll Drive
   Herndon, VA 20170
   USA

   EMail: housley@vigilsec.com