Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: August 8, 2014

Y. Nir
Check Point

February 4, 2014

ChaCha20 and Poly1305 and their use in IPsec
draft-nir-ipsecme-chacha20-poly1305-01

Abstract

This document describes the use of the ChaCha20 stream cipher in IPsec, as well as the use of the Poly1305 authenticator, both as stand-alone algorithms, and as a combined mode AEAD algorithm.

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 August 8, 2014.

Table of Contents

1. Introduction

The Advanced Encryption Standard (AES - [FIPS-197]) has become the gold standard in encryption. Its efficient design, wide implementation, and hardware support allow for high performance in many areas, including IPsec VPNs. On most modern platforms, AES is anywhere from 4x to 10x as fast as the previous most-used cipher, 3-key Data Encryption Standard (3DES - [FIPS-46]), which makes it not only the best choice, but the only choice.

The problem is that if future advances in cryptanalysis reveal a weakness in AES, VPN users will be in an unenviable position. With the only other widely supported cipher being the much slower 3DES, it is not feasible to re-configure IPsec implementations to use 3DES. [standby-cipher] describes this issue and the need for a standby cipher in greater detail.

This document proposes the ChaCha20 stream cipher as such a standby cipher with or without the Poly1305 authenticator. These algorithms are described in a separate document ([chacha_poly]). This document only describes the IPsec-specific things.

1.1. Conventions Used in This Document

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].

2. Algorithms for ESP & AH

This document defines three algorithms for use with the Encapsulated Security Protocol (ESP - [RFC4303]) and Authentication Header (AH - [RFC4302]):

- ChaCha20 for use as an encryption algorithms for ESP.
- Poly1305-MAC for use as a message authentication algorithm for ESP

and AH.

- ChaCha20-Poly1305-ESP as an AEAD algorithm for ESP.

2.1. ENCR_ChaCha20 for ESP

The algorithm for ChaCha20 is described in section 2.4 of [chacha_poly]. In ESP the following parameters are used:

- The IV is 64-bit. Since this is used for the nonce in the ChaCha20 function, this IV MUST NOT repeat. The most natural way to implement this is with a counter, but anything that guarantees uniqueness can be used, such as a linear feedback shift register (LFSR). Note that the encrypter can use any IV generation method

that meets the uniqueness requirement, without coordinating with the decrypter.

- The Internet Key Exchange protocol (IKE - [RFC5996]) generates a bitstring called KEYMAT that is generated from a PRF. That KEYMAT is divided into keys for encryption, message authentication and whatever else is needed. For the ChaCha20 algorithm, 256 bits are used for the key. TBD: do we want an extra 32 bits as salt for the nonce like in GCM?
- The ChaCha20 encryption algorithm is called with the 256-bit key, one (1) for the initial counter. The packet IV is prepended by a 32-bit sender ID value to form the 96-bit nonce. For regular IPsec, the Sender ID is set to zero. For multi-sender SAs, such as described in [RFC6054], there sender ID can be set to a different value for each sender. The reason that one (1) is used for the initial counter rather than zero is that one is used for encryption in the AEAD algorithm (because zero is reserved for generating the one-time Poly1305 key), so one was used here for consistency.
- As ChaCha20 is not a block cipher, no padding should be necessary. However, in keeping with the specification in RFC 4303, the ESP does have padding, so as to align the buffer to an integral multiple of 4 octets.

The encryption algorithm transform ID for negotiating this algorithm in IKE is TBA by IANA.

2.2. AUTH_Poly1305 for ESP and AH

You cannot use a part of the keying material directly, because

Poly1305 requires an unpredictable and non-repeating key for each authenticated message. So the Poly1305 key for each message is generated as in section 2.6 of [chacha_poly]:

- The key for AUTH_Poly1305 is 256-bits.
- To determine the per-packet Poly1305 key, the ChaCha20 block function is called with the following parameters:
  - The AUTH_Poly1305 256-bit key is used as the key.
  - The nonce is set from the sequence number (in the order as it appears in the packet). 32 or 64 zero bits are prepended depending on whether ESN is enabled or not, forming a 96-bit nonce.
  - Zero is used for the block counter.
- The 512-bit result is then truncated. The top 256 bits are used as the one-time key for Poly1305 to calculate the message authentication code (MAC).
- The 128-bit output serves as the MAC for the packet. All 16 bytes are included in the packet.

The integrity algorithm transform ID for negotiating this algorithm

in IKE is TBA by IANA.

2.2.1. Example One-Time Key Derivation

Suppose our 256-bit MAC key and sequence number are as follows:

```
AUTH_Poly1305 Key:
000  7b ac 2b 25 2d b4 47 af 09 b6 7a 55 a4 e9 55 84|{.+%-.G...zU..U.
016  0a e1 d6 73 10 75 d9 eb 2a 93 75 78 3e d5 53 ff|...s.u..*.ux>.S.

ESN: off
Sequence Number: 40 (represented on the packet as 00:00:00:28)
```

So the ChaCha20 block function (section 2.3 of [chacha_poly]) is called with the following parameters:

- The AUTH_Poly1305 key.
- The 96-bit nonce 00:00:00:00:00:00:00:00:00:00:00:28.
- The block count parameter set to zero (0).

Note in the following ChaCha20 block, that the sequence number (bottom right) is reversed. This is because the sequence number is big-endian in the ESP packet, but is treated as a sequence of 32-bit

little-endian numbers in ChaCha20.

Set up ChaCha20 block:

```
    61707865  3320646e  79622d32  6b206574
    252bac7b  af47b42d  557ab609  8455e9a4
    73d6e10a  ebd97510  7875932a  ff53d53e
    00000000  00000000  00000000  28000000
```

After running the ChaCha20 block operation:

```
    0c15e292  6e2fe7cc  dcd9bd92  f60a30b3
    4094c018  b0ba041d  88ed8acf  09f9bba5
    a29dbf3d  e1a6acdc  011e3fe8  d953dff5
    32989c29  12be0248  0c267749  e55f2037
```

The Poly1305 one-time key (256 bits):

```
000  92 e2 15 0c cc e7 2f 6e 92 bd d9 dc b3 30 0a f6  ....../n.....0..
016  18 c0 94 40 1d 04 ba b0 cf 8a ed 88 a5 bb f9 09  ...@............
```

2.3. ESP_ChaCha20-Poly1305 for ESP

ESP_ChaCha20-Poly1305 is a combined mode algorithm, or AEAD. The construction follows the AEAD construction in section 2.7 of [chacha_poly]:

- As in Section 2.1, the IV is 64-bit, and is used as part of the nonce.
- Also as in Section 2.1, a 32-bit sender ID (zero for regular IPsec) is prepended to the 64-bit IV to form the nonce.
- Also as in Section 2.1, the encryption key is 256-bit.
- As in Section 2.2, the nonce, along with a block counter of zero is passed to the ChaCha20 block function, and the top part of the result used as the Poly1305 key. However, unlike AUTH_Poly1305, the nonce passed to the block function here does not depend on the ESP sequence number, but is the same nonce that is used in ChaCha20, including the 32-bit Sender ID bits, and the key passed is the same as the encryption key.
- The ChaCha20 encryption function is then called with the nonce, the key, and an initial counter of zero.
- Finally, the Poly1305 function is run on the data to be

authenticated, which is, as specified in section 2.7 of [chacha_poly] a concatenation of the following:

* The Authenticated Additional Data (AAD) - see Section 2.3.1.
* The AAD length in bytes as a 32-bit network order quantity.
* The ciphertext
* The length of the ciphertext as a 32-bit network order quantity.

o The 128-bit output of Poly1305 is used as the tag. All 16 bytes are included in the packet.

The encryption algorithm transform ID for negotiating this algorithm in IKE is TBA by IANA.

2.3.1. AAD Construction

The construction of the Additional Authenticated Data (AAD) is similar to the one in [RFC4106]. For security associations (SAs) with 32-bit sequence numbers the AAD is 8 bytes: 4-byte SPI followed by 4-byte sequence number ordered exactly as it is in the packet. For SAs with ESN the AAD is 12 bytes: 4-byte SPI followed by an 8-byte sequence number as a 64-bit network order integer.

3. Security Considerations

The ChaCha20 cipher is designed to provide 256-bit security.

The Poly1305 authenticator is designed to ensure that forged messages are rejected with a probability of 1-(n/(2^102)) for a 16n-byte message, even after sending 2^64 legitimate messages, so it is SUF-CMA in the terminology of [AE].

The most important security consideration in implementing this draft

is the uniqueness of the nonce used in ChaCha20. This is trivial in AUTH_Poly1305, because the packet sequence number is used as a nonce, but is required for ChaCha20 as well. As said in Section 2.1, the nonce should be selected uniquely for a particular key. counters and LFSRs are both acceptable ways of generating unique nonces, as is encrypting a counter using a 64-bit cipher such as DES. Note that it is not acceptable to use a truncation of a counter encrypted with a 128-bit or 256-bit cipher, because such a truncation may repeat after

a short time.

4. IANA Considerations

IANA is requested to assign two values from the IKEv2 "Transform Type 1 - Encryption Algorithm Transform IDs" registry, as follows:

- ENCR_ChaCha20
- ESP_ChaCha20-Poly1305

IANA is also requested to assign one value from the IKEv2 "Transform Type 3 - Integrity Algorithm Transform IDs" registry with name "AUTH_Poly1305".

5. Acknowledgements

All of the algorithms in this document were designed by D. J. Bernstein. The AEAD construction was designed by Adam Langley. The author would also like to thank Adam for helpful comments, as well as Yaron Sheffer for telling me to write the algorithms draft.

6. References

6.1. Normative References

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

[RFC4302] Kent, S., "IP Authentication Header", RFC 4302, December 2005.

[RFC4303] Kent, S., "IP Encapsulating Security Payload (ESP)", RFC 4303, December 2005.

[RFC5996] Kaufman, C., Hoffman, P., Nir, Y., and P. Eronen, "Internet Key Exchange Protocol Version 2 (IKEv2)", RFC 5996, September 2010.

[RFC6054] McGrew, D. and B. Weis, "Using Counter Modes with

Encapsulating Security Payload (ESP) and Authentication Header (AH) to Protect Group Traffic", RFC 6054, November 2010.

[chacha_poly] Langley, A. and Y. Nir, "ChaCha20 and Poly1305 for IETF protocols", draft-nir-cfrg-chacha20-poly1305-01 (work in progress), January 2014.

6.2. Informative References

[AE] Bellare, M. and C. Namprempre, "Authenticated Encryption: Relations among notions and analysis of the generic composition paradigm", <http://cseweb.ucsd.edu/~mihir/papers/oem.html>.

[FIPS-197] National Institute of Standards and Technology, "Advanced Encryption Standard (AES)", FIPS PUB 197, November 2001, <http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf>.

[FIPS-46] National Institute of Standards and Technology, "Data Encryption Standard", FIPS PUB 46-2, December 1993, <http://www.itl.nist.gov/fipspubs/fip46-2.htm>.

[RFC4106] Viega, J. and D. McGrew, "The Use of Galois/Counter Mode (GCM) in IPsec Encapsulating Security Payload (ESP)", RFC 4106, June 2005.

[standby-cipher] McGrew, D., Grieco, A., and Y. Sheffer, "Selection of Future Cryptographic Standards", draft-mcgrew-standby-cipher (work in progress).

Author's Address

Yoav Nir
Check Point Software Technologies Ltd.
5 Hasolelim st.
Tel Aviv 6789735
Israel

Email: ynir@checkpoint.com