Internet Engineering Task Force (IETF)
Request for Comments: 7905
Updates: 5246, 6347
Category: Standards Track
ISSN: 2070-1721

A. Langley
W. Chang
Google, Inc.
N. Mavrogiannopoulos
Red Hat
J. Strombergson
Secworks Sweden AB
S. Josefsson
SJD AB

June 2016

ChaCha20-Poly1305 Cipher Suites for Transport Layer Security (TLS)

Abstract

This document describes the use of the ChaCha stream cipher and Poly1305 authenticator in the Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS) protocols.

This document updates RFCs 5246 and 6347.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

This document describes the use of the ChaCha stream cipher and Poly1305 authenticator in version 1.2 or later of the Transport Layer Security (TLS) protocol [RFC5246] as well as version 1.2 or later of the Datagram Transport Layer Security (DTLS) protocol [RFC6347].

ChaCha [CHACHA] is a stream cipher developed by D. J. Bernstein in 2008. It is a refinement of Salsa20, which is one of the selected ciphers in the eSTREAM portfolio [ESTREAM], and it was used as the core of the SHA-3 finalist, BLAKE.

The variant of ChaCha used in this document has 20 rounds, a 96-bit nonce, and a 256-bit key; it is referred to as "ChaCha20". This is the conservative variant (with respect to security) of the ChaCha family and is described in [RFC7539].

Poly1305 [POLY1305] is a Wegman-Carter, one-time authenticator designed by D. J. Bernstein. Poly1305 takes a 256-bit, one-time key and a message, and it produces a 16-byte tag that authenticates the message such that an attacker has a negligible chance of producing a valid tag for an inauthentic message. It is described in [RFC7539].

ChaCha and Poly1305 have both been designed for high performance in software implementations. They typically admit a compact implementation that uses few resources and inexpensive operations, which makes them suitable on a wide range of architectures. They have also been designed to minimize leakage of information through side-channels.

Recent attacks [CBC-ATTACK] have indicated problems with the CBC-mode cipher suites in TLS and DTLS, as well as issues with the only supported stream cipher (RC4) [RC4-ATTACK]. While the existing Authenticated Encryption with Associated Data (AEAD) cipher suites (based on AES-GCM) address some of these issues, there are concerns about their performance and ease of software implementation.

Therefore, a new stream cipher to replace RC4 and address all the previous issues is needed. It is the purpose of this document to describe a secure stream cipher for both TLS and DTLS that is comparable to RC4 in speed on a wide range of platforms and can be implemented easily without being vulnerable to software side-channel attacks.

2. ChaCha20 Cipher Suites

The ChaCha20 and Poly1305 primitives are built into an AEAD algorithm [RFC5116], AEAD_CHACHA20_POLY1305, as described in [RFC7539]. This AEAD is incorporated into TLS and DTLS as specified in Section 6.2.3.3 of [RFC5246].

AEAD_CHACHA20_POLY1305 requires a 96-bit nonce, which is formed as follows:

1. The 64-bit record sequence number is serialized as an 8-byte, big-endian value and padded on the left with four 0x00 bytes.

2. The padded sequence number is XORed with the client_write_IV (when the client is sending) or server_write_IV (when the server is sending).

In DTLS, the 64-bit seq_num is the 16-bit epoch concatenated with the 48-bit sequence_number.

This nonce construction is different from the one used with AES-GCM in TLS 1.2 but matches the scheme expected to be used in TLS 1.3. The nonce is constructed from the record sequence number and the shared secret, both of which are known to the recipient. The advantage is that no per-record, explicit nonce need be transmitted, which saves eight bytes per record and prevents implementations from mistakenly using a random nonce. Thus, in the terms of [RFC5246], SecurityParameters.fixed_iv_length is twelve bytes and SecurityParameters.record_iv_length is zero bytes.

The following cipher suites are defined:

```
TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256   = {0xCC, 0xA8}
TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256 = {0xCC, 0xA9}
TLS_DHE_RSA_WITH_CHACHA20_POLY1305_SHA256     = {0xCC, 0xAA}

TLS_PSK_WITH_CHACHA20_POLY1305_SHA256         = {0xCC, 0xAB}
TLS_ECDHE_PSK_WITH_CHACHA20_POLY1305_SHA256   = {0xCC, 0xAC}
TLS_DHE_PSK_WITH_CHACHA20_POLY1305_SHA256     = {0xCC, 0xAD}
TLS_RSA_PSK_WITH_CHACHA20_POLY1305_SHA256     = {0xCC, 0xAE}
```

The DHE_RSA, ECDHE_RSA, ECDHE_ECDSA, PSK, ECDHE_PSK, DHE_PSK, and RSA_PSK key exchanges for these cipher suites are unaltered; thus, they are performed as defined in [RFC5246], [RFC4492], and [RFC5489].

The pseudorandom function (PRF) for all the cipher suites defined in this document is the TLS PRF with SHA-256 [FIPS180-4] as the hash function.

3. IANA Considerations

IANA has added the following entries in the TLS Cipher Suite Registry:

```
TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256   = {0xCC, 0xA8}
TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256 = {0xCC, 0xA9}
TLS_DHE_RSA_WITH_CHACHA20_POLY1305_SHA256     = {0xCC, 0xAA}

TLS_PSK_WITH_CHACHA20_POLY1305_SHA256         = {0xCC, 0xAB}
TLS_ECDHE_PSK_WITH_CHACHA20_POLY1305_SHA256   = {0xCC, 0xAC}
TLS_DHE_PSK_WITH_CHACHA20_POLY1305_SHA256     = {0xCC, 0xAD}
TLS_RSA_PSK_WITH_CHACHA20_POLY1305_SHA256     = {0xCC, 0xAE}
```

4. Security Considerations

ChaCha20 follows the same basic principle as Salsa20 [SALSA20SPEC], a cipher with significant security review [SALSA20-SECURITY] [ESTREAM]. At the time of writing this document, there are no known significant security problems with either cipher, and ChaCha20 is shown to be more resistant in certain attacks than Salsa20 [SALSA20-ATTACK]. Furthermore, ChaCha20 was used as the core of the BLAKE hash function, a SHA3 finalist, which has received considerable cryptanalytic attention [NIST-SHA3].

Poly1305 is designed to ensure that forged messages are rejected with a probability of 1-(n/2^107), where n is the maximum length of the input to Poly1305. In the case of (D)TLS, this means a maximum forgery probability of about 1 in 2^93.

The cipher suites described in this document require that a nonce never be repeated under the same key. The design presented ensures this by using the TLS sequence number, which is unique and does not wrap [RFC5246].

It should be noted that AEADs, such as ChaCha20-Poly1305, are not intended to hide the lengths of plaintexts. When this document speaks of side-channel attacks, it is not considering traffic analysis, but rather timing and cache side-channels. Traffic analysis, while a valid concern, is outside the scope of the AEAD and is being addressed elsewhere in future versions of TLS.

Otherwise, this document should not introduce any additional security considerations other than those that follow from the use of the AEAD_CHACHA20_POLY1305 construction, thus the reader is directed to the Security Considerations section of [RFC7539].

5. References

5.1. Normative References

[FIPS180-4]
National Institute of Standards and Technology, "Secure Hash Standard (SHS)", FIPS PUB 180-4, DOI 10.6028/NIST.FIPS180-4, August 2015, <http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf>.

[RFC4492] Blake-Wilson, S., Bolyard, N., Gupta, V., Hawk, C., and B. Moeller, "Elliptic Curve Cryptography (ECC) Cipher Suites for Transport Layer Security (TLS)", RFC 4492, DOI 10.17487/RFC4492, May 2006, <http://www.rfc-editor.org/info/rfc4492>.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <http://www.rfc-editor.org/info/rfc5246>.

[RFC5489] Badra, M. and I. Hajjeh, "ECDHE_PSK Cipher Suites for Transport Layer Security (TLS)", RFC 5489, DOI 10.17487/RFC5489, March 2009, <http://www.rfc-editor.org/info/rfc5489>.

[RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, DOI 10.17487/RFC6347, January 2012, <http://www.rfc-editor.org/info/rfc6347>.

[RFC7539] Nir, Y. and A. Langley, "ChaCha20 and Poly1305 for IETF Protocols", RFC 7539, DOI 10.17487/RFC7539, May 2015, <http://www.rfc-editor.org/info/rfc7539>.

5.2. Informative References

[CBC-ATTACK]
AlFardan, N. and K. Paterson, "Lucky Thirteen: Breaking the TLS and DTLS Record Protocols", IEEE Symposium on Security and Privacy, 2013, <http://www.ieee-security.org/TC/SP2013/papers/4977a526.pdf>.

[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>.

[NIST-SHA3] Chang, S., Perlner, R., Burr, W., Turan, M., Kelsey, J., Paul, S., and L. Bassham, "Third-Round Report of the SHA-3 Cryptographic Hash Algorithm Competition", DOI 10.6028/NIST.IR.7896, November 2012, <http://dx.doi.org/10.6028/NIST.IR.7896>.

[POLY1305] Bernstein, D., "The Poly1305-AES message-authentication code", FSE '05 Proceedings of the 12th international conference on Fast Software Encryption Pages 32-49, DOI 10.1007/11502760_3, February 2005, <http://cr.yp.to/mac/poly1305-20050329.pdf>.

[RC4-ATTACK] Isobe, T., Ohigashi, T., Watanabe, Y., and M. Morii, "Full Plaintext Recovery Attack on Broadcast RC4", International Workshop on Fast Software Encryption FSE, DOI 10.1007/978-3-662-43933-3_10, 2013, <http://www.iacr.org/archive/fse2013/84240167/84240167.pdf>.

[RFC5116] McGrew, D., "An Interface and Algorithms for Authenticated Encryption", RFC 5116, DOI 10.17487/RFC5116, January 2008, <http://www.rfc-editor.org/info/rfc5116>.

[SALSA20-ATTACK] Aumasson, J-P., Fischer, S., Khazaei, S., Meier, W., and C. Rechberger, "New Features of Latin Dances: Analysis of Salsa, ChaCha, and Rumba", DOI 10.1007/978-3-540-71039-4_30, 2007, <http://eprint.iacr.org/2007/472.pdf>.

[SALSA20-SECURITY] Bernstein, D., "Salsa20 security", April 2005, <http://cr.yp.to/snuffle/security.pdf>.

[SALSA20SPEC] Bernstein, D., "Salsa20 specification", April 2005, <http://cr.yp.to/snuffle/spec.pdf>.

Acknowledgements

The authors would like to thank Zooko Wilcox-O'Hearn, Samuel Neves, and Colm MacCarthaigh for their suggestions and guidance.

Authors' Addresses

Adam Langley
Google, Inc.

Email: agl@google.com

Wan-Teh Chang
Google, Inc.

Email: wtc@google.com

Nikos Mavrogiannopoulos
Red Hat

Email: nmav@redhat.com

Joachim Strombergson
Secworks Sweden AB

Email: joachim@secworks.se
URI: http://secworks.se/

Simon Josefsson
SJD AB

Email: simon@josefsson.org
URI: http://josefsson.org/