

Network Working Group
Internet-Draft
Expires: May 5, 2014

A. Langley
W. Chang
Google Inc
Nov 2013

ChaCha20 and Poly1305 based Cipher Suites for TLS
draft-agl-tls-chacha20poly1305-03

Abstract

This memo describes the use of the ChaCha20 cipher with a Poly1305 authenticator in Transport Layer Security (TLS).

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 May 5, 2014.

Copyright Notice

Copyright (c) 2013 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to [BCP 78](#) and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Internet-Draft

ChaCha20Poly1305 for TLS

Nov 2013

Table of Contents

1.	Introduction	3
2.	Requirements Notation	4
3.	ChaCha20	5
4.	Poly1305	7
5.	AEAD construction	8
6.	Cipher suites	10
7.	Test vectors	11
8.	Security Considerations	14
9.	IANA Considerations	15
10.	References	16
10.1.	Normative References	16
10.2.	Informative References	16
	Authors' Addresses	17

1. Introduction

Existing TLS [[RFC5246](#)] cipher suites either suffer from cryptographic weaknesses (RC4), major implementation pitfalls (CBC mode block ciphers) or are difficult to efficiently and securely implement in software (AES-GCM). In order to improve the state of software TLS implementations, this memo specifies cipher suites that can be fast and secure when implemented in software without sacrificing key agility.

2. Requirements Notation

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](#) [[RFC2119](#)].

[3.](#) ChaCha20

ChaCha20 [[chacha](#)] is a stream cipher developed by D. J. Bernstein. It is a refinement of Salsa20 and was used as the core of the SHA-3 finalist, BLAKE.

ChaCha20 maps 16, 32-bit input words to 16, 32-bit output words. By convention, 8 of the input words consist of a 256-bit key, 4 are constants and the remaining four are a nonce and block counter. The output words are converted to bytes and XORed with the plaintext to produce ciphertext. In order to generate sufficient output bytes to XOR with the whole plaintext, the block counter is incremented and ChaCha20 is run again, as many times as needed, for up to 2^{70} bytes of output.

ChaCha20 operates on a state of 16, 32-bit words which are initialised from the input words. The first four input words are constants: (0x61707865, 0x3320646e, 0x79622d32, 0x6b206574). Input words 4 through 11 are taken from the 256-bit key by reading the bytes in little-endian order, in 4-byte chunks. Input words 12 and 13 are a block counter, with word 12 overflowing into word 13. Lastly, words 14 and 15 are taken from an 8-byte nonce, again by

reading the bytes in little-endian order, in 4-byte chunks. The block counter words are initially zero.

ChaCha20 consists of 20 rounds, alternating between "column" rounds and "diagonal" rounds. Each round applies the following "quarter-round" function four times, to a different set of words each time. The quarter-round function updates 4, 32-bit words (a, b, c, d) as follows, where <<< is a bitwise, left rotation:

```
a += b; d ^= a; d <<<= 16;
c += d; b ^= c; b <<<= 12;
a += b; d ^= a; d <<<= 8;
c += d; b ^= c; b <<<= 7;
```

The 16 words are conceptually arranged in a four by four grid with the first word in the top-left position and the fourth word in the top-right position. The "column" rounds then apply the quarter-round function to the four columns, from left to right. The "diagonal" rounds apply the quarter-round to the top-left, bottom-right diagonal, followed by the pattern shifted one place to the right, for three more quarter-rounds.

Specifically, a column round applies the quarter-round function to the following indexes: (0, 4, 8, 12), (1, 5, 9, 13), (2, 6, 10, 14), (3, 7, 11, 15). A diagonal round applies it to these indexes: (0, 5, 10, 15), (1, 6, 11, 12), (2, 7, 8, 13), (3, 4, 9, 14).

After 20 rounds of the above processing, the original 16 input words are added to the 16 words to form the 16 output words.

The 64 output bytes are generated from the 16 output words by serialising them in little-endian order and concatenating the results.

4. Poly1305

Poly1305 [[poly1305](#)] is a Wegman-Carter, one-time authenticator designed by D. J. Bernstein. Poly1305 takes a 32-byte, one-time key and a message and 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.

The first 16 bytes of the one-time key form an integer, `_r_`, as follows: the top four bits of the bytes at indexes 3, 7, 11 and 15 are cleared, the bottom 2 bits of the bytes at indexes 4, 8 and 12 are cleared and the 16 bytes are taken as a little-endian value.

An accumulator is set to zero. For each chunk of 16 bytes from the input message, a byte with value 1 is appended and the 17 bytes are treated as a little-endian number. If the last chunk has less than 16 bytes then zero bytes are appended after the 1 byte is appended until there are 17 bytes. The value is added to the accumulator and then the accumulator is multiplied by `_r_`, all mod $2^{130} - 5$.

Finally the last 16 bytes of the one-time key are treated as a little-endian number and added to the accumulator, mod 2^{128} . The result is serialised as a little-endian number, producing the 16 byte tag. (The original specification of Poly1305 used AES to generate the constant term of the polynomial from a counter nonce. For a more recent treatment that avoids the use of a block cipher in this fashion, as is done here, see [section 9](#) of the NaCl specification [[naclcrypto](#)].)

The ChaCha20 and Poly1305 primitives are built into an AEAD algorithm [[RFC5116](#)], AEAD_CHACHA20_POLY1305, that takes a 32 byte key and 8 byte nonce as follows:

ChaCha20 is run with the given key and nonce and with the two counter words set to zero. The first 32 bytes of the 64 byte output are saved to become the one-time key for Poly1305. The remainder of the output is discarded. The first counter input word is set to one and the plaintext is encrypted by XORing it with the output of invocations of the ChaCha20 function as needed, incrementing the first counter word after each block and overflowing into the second. (In the case of the TLS, limits on the plaintext size mean that the first counter word will never overflow in practice.)

The reason for generating the Poly1305 key like this rather than using key material from the handshake is that handshake key material is per-session, but for a polynomial MAC, a unique, secret key is needed per-record.

The Poly1305 key is used to calculate a tag for the following input: the concatenation of the additional data, the number of bytes of additional data, the ciphertext and the number of bytes of ciphertext. Numbers are represented as 8-byte, little-endian values. The resulting tag is appended to the ciphertext, resulting in the output of the AEAD operation.

Authenticated decryption is largely the reverse of the encryption process: generate one block of ChaCha20 keystream and use the first 32 bytes as a Poly1305 key. Feed Poly1305 the additional data and ciphertext, with the length suffixing as described above. Verify, in constant time, that the calculated Poly1305 authenticator matches the final 16 bytes of the input. If not, the input can be rejected immediately. Otherwise, run ChaCha20, starting with a counter value of one, to decrypt the ciphertext.

When used in TLS, the "record_iv_length" is zero and the nonce is the sequence number for the record, as an 8-byte, big-endian number. The additional data is seq_num + TLSCompressed.type + TLSCompressed.version + TLSCompressed.length, where "+" denotes concatenation.

(In DTLS, the sequence number is only 48 bits. Thus, when used in DTLS, AEAD_CHACHA20_POLY1305 based cipher suites use the concatenation of the 16-bit epoch with the 48-bit sequence number as a replacement for TLS's 64-bit sequence number.)

In accordance with [section 4 of RFC 5116](#) [RFC5116], the constants for this AEAD algorithm are as follows: K_LEN is 32 bytes, N_MIN and N_MAX are 8 bytes, P_MAX and A_MAX are 2^{64} , C_MAX is $2^{64}+16$. An AEAD_CHACHA20_POLY1305 ciphertext is exactly 16 octets longer than its corresponding plaintext.

6. Cipher suites

The following cipher suites are defined which use the AEAD_CHACHA20_POLY1305 algorithm:

TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256 = {0xcc, 0x13}

TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256 = {0xcc, 0x14}

TLS_DHE_RSA_WITH_CHACHA20_POLY1305_SHA256 = {0xcc, 0x15}

These cipher suites use the TLS PRF [[RFC5246](#)] with SHA-256 as the hash function.

The following blocks contain test vectors for ChaCha20. The first line contains the 256-bit key, the second the 64-bit nonce and the last line contains a prefix of the resulting ChaCha20 key-stream.

[illegible]

INPUT: 48656c6c6f20776f726c6421
KEY: 746869732069732033322d62797465206b657920666f7220506f6c793133
3035
TAG: a6f745008f81c916a20dcc74eef2b2f0

The following block contains a test vector for the AEAD_CHACHA20_POLY1305 algorithm. The first four lines consist of the standard inputs to an AEAD algorithm and the last line contains the encrypted and authenticated result.

KEY: 4290bcb154173531f314af57f3be3b5006da371ece272afa1b5dbdd110
0a1007
INPUT: 86d09974840bde2a5ca
NONCE: cd7cf67be39c794a
AD: 87e229d4500845a079c0
OUTPUT: e3e446f7ede9a19b62a4677dabf4e3d24b876bb284753896e1d6

To aid implementations, the next block contains some intermediate values in the AEAD_CHACHA20_POLY1305 algorithm. The first line contains the Poly1305 key that is derived and the second contains the raw bytes that are authenticated by Poly1305.

KEY: 9052a6335505b6d507341169783dccac0e26f84ea84906b1558c05bf4815
0fbe
INPUT: 87e229d4500845a079c00a0000000000000000e3e446f7ede9a19b62a40a00
000000000000

[8.](#) Security Considerations

ChaCha20 is designed to provide a 256-bit security level. Poly1305 is designed to ensure that forged messages are rejected with a probability of $1-(n/2^{102})$ for a $16*n$ byte message, even after sending 2^{64} legitimate messages.

The AEAD_CHACHA20_POLY1305 algorithm is designed to meet the standard notions of privacy and authenticity. For formal definitions see Authenticated Encryption [\[AE\]](#).

These cipher suites require that a nonce never be repeated for the same key. This is achieved by simply using the TLS sequence number.

Only forward secure cipher suites are defined as it's incongruous to define a high-security cipher suite without forward security.

[9.](#) IANA Considerations

IANA is requested to assign the values for the cipher suites defined in this document from the TLS registry.

IANA is requested to assign a value for AEAD_CHACHA20_POLY1305 in the registry of AEAD algorithms.

[10.](#) References

[10.1.](#) Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", [BCP 14](#), [RFC 2119](#), March 1997.
- [RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", [RFC 5246](#), August 2008.
- [RFC5116] McGrew, D., "An Interface and Algorithms for Authenticated Encryption", [RFC 5116](#), January 2008.
- [chacha] Bernstein, D., "ChaCha, a variant of Salsa20.", Jan 2008, <<http://cr.yp.to/chacha/chacha-20080128.pdf>>.
- [poly1305] Bernstein, D., "The Poly1305-AES message-authentication code.", March 2005, <<http://cr.yp.to/mac/poly1305-20050329.pdf>>.

[10.2.](#) 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>>.
- [naclcrypto] Bernstein, D., "http://cr.yp.to/highspeed/naclcrypto-20090310.pdf", March 2009, <<http://cr.yp.to/highspeed/naclcrypto-20090310.pdf>>.

Internet-Draft

ChaCha20Poly1305 for TLS

Nov 2013

Authors' Addresses

Adam Langley
Google Inc

Email: agl@google.com

Wan-Teh Chang
Google Inc

Email: wtc@google.com

