Network Working Group S. Josefsson
Internet-Draft SJD AB
Intended status: Informational J. Strombergson
Expires: October 11, 2013 Secworks Sweden AB
N. Mavrogiannopoulos
KU Leuven
April 9, 2013

The Salsa20 Stream Cipher for Transport Layer Security
draft-josefsson-salsa20-tls-02

Abstract

This document describe how the Salsa20 stream cipher can be used in the Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS) protocols.

Status of this Memo

Copyright Notice

Table of Contents

1. Introduction

This document describe how the Salsa20 stream cipher can be used in the Transport Layer Security (TLS) version 1.0 [RFC2246], TLS version 1.1 [RFC4346], and TLS version 1.2 [RFC5246] protocols, as well as in the Datagram Transport Layer Security (DTLS) versions 1.0 [RFC4347] and 1.2 [RFC6347]. It can also be used with Secure Sockets Layer (SSL) version 3.0 [RFC6101].

Salsa20 [SALSA20SPEC] is a stream cipher that has been designed for high performance in software implementations. The cipher has compact implementation and uses few resources and inexpensive operations that makes it suitable for implementation on a wide range of architectures. It has been designed to prevent leakage of information through side channel analysis, has a simple initialization sequence and provides good key agility and performance. Salsa20 is one of the ciphers selected as part of the eSTREAM portfolio of stream ciphers [ESTREAM].

Recent attacks [CBC-ATTACK] have indicated problems with CBC-mode cipher suites in TLS and DTLS as well as issues with the only supported stream cipher (RC4) [RC4-ATTACK]. While the existing AEAD ciphersuites address these issues, concerns about their performance, on general purpose CPUs, are sometimes raised [AEAD-PERFORMANCE].

Moreover, the RC4 cipher cannot be used in DTLS because it does not provide random access in the key stream. That allowed no choice of a fast stream cipher in the context of DTLS.

The purpose of this document is to provide an alternative stream cipher for both TLS and DTLS that is comparable to RC4 in speed on a wide range of platforms.

2. Salsa20 Cipher Suites

The following variants of Salsa20 are specified. The variants provide a range of performance and security that can be selected as appropriate.

ESTREAM_SALSA20: Salsa20 with 12 rounds and a 256 bit key. This cipher is the high performant eSTREAM Salsa20 with 256 bit key.

SALSA20: Salsa20 with 20 rounds and a 256 bit key. This is the original (conservative with respect to security) variant of Salsa20.

In the next sections different ciphersuites are defined that utilize the Salsa20 cipher combined with various MAC methods

In all cases, the pseudorandom function (PRF) for TLS 1.2 is the TLS PRF with SHA-256 as the hash function. When used with TLS versions prior to 1.2, the PRF is calculated as specified in the appropriate version of the TLS specification.

2.1. Salsa20 Cipher Suites with HMAC-SHA1

The following CipherSuites are defined:

```
  TLS_RSA_WITH_ESTREAM_SALSA20_SHA1          = {0xTBD, 0xTBD}
  TLS_RSA_WITH_SALSA20_SHA1                  = {0xTBD, 0xTBD}
  TLS_DHE_RSA_WITH_ESTREAM_SALSA20_SHA1      = {0xTBD, 0xTBD}
  TLS_DHE_RSA_WITH_SALSA20_SHA1              = {0xTBD, 0xTBD}
  TLS_ECDHE_RSA_WITH_ESTREAM_SALSA20_SHA1    = {0xTBD, 0xTBD}
  TLS_ECDHE_RSA_WITH_SALSA20_SHA1            = {0xTBD, 0xTBD}
  TLS_ECDHE_ECDSA_WITH_ESTREAM_SALSA20_SHA1 = {0xTBD, 0xTBD}
  TLS_ECDHE_ECDSA_WITH_SALSA20_SHA1          = {0xTBD, 0xTBD}

  TLS_PSK_WITH_ESTREAM_SALSA20_SHA1          = {0xTBD, 0xTBD}
  TLS_PSK_WITH_SALSA20_SHA1                  = {0xTBD, 0xTBD}
  TLS_DHE_PSK_WITH_ESTREAM_SALSA20_SHA1      = {0xTBD, 0xTBD}
  TLS_DHE_PSK_WITH_SALSA20_SHA1              = {0xTBD, 0xTBD}
  TLS_RSA_PSK_WITH_ESTREAM_SALSA20_SHA1      = {0xTBD, 0xTBD}
  TLS_RSA_PSK_WITH_SALSA20_SHA1              = {0xTBD, 0xTBD}
  TLS_ECDHE_PSK_WITH_ESTREAM_SALSA20_SHA1    = {0xTBD, 0xTBD}
  TLS_ECDHE_PSK_WITH_SALSA20_SHA1            = {0xTBD, 0xTBD}
```

Note that Salsa20 requires a 64-bit nonce. That nonce is updated on the encryption of every TLS record, and is set to be the 64-bit TLS record sequence number. In case of DTLS the 64-bit nonce is formed as the concatenation of the 16-bit epoch with the 48-bit sequence number.

The RSA, DHE_RSA, ECDHE_RSA, ECDHE_ECDSA, PSK, DHE_PSK, RSA_PSK, ECDHE_PSK key exchanges are performed as defined in [RFC5246], [RFC4492], and [RFC5489].

The MAC algorithm used in the ciphersuites above is HMAC-SHA1 [RFC6234].

2.2. Salsa20 Cipher Suites with UMAC-96

The following CipherSuites utilize Salsa20 in combination with UMAC-96 [RFC4418], a very fast MAC algorithm based on Universal Hashing.

```
TLS_RSA_WITH_ESTREAM_SALSA20_UMAC96         = {0xTBD, 0xTBD}
TLS_RSA_WITH_SALSA20_UMAC96                 = {0xTBD, 0xTBD}
TLS_DHE_RSA_WITH_ESTREAM_SALSA20_UMAC96     = {0xTBD, 0xTBD}
TLS_DHE_RSA_WITH_SALSA20_UMAC96             = {0xTBD, 0xTBD}
TLS_ECDHE_RSA_WITH_ESTREAM_SALSA20_UMAC96   = {0xTBD, 0xTBD}
TLS_ECDHE_RSA_WITH_SALSA20_UMAC96           = {0xTBD, 0xTBD}
TLS_ECDHE_ECDSA_WITH_ESTREAM_SALSA20_UMAC96 = {0xTBD, 0xTBD}
TLS_ECDHE_ECDSA_WITH_SALSA20_UMAC96         = {0xTBD, 0xTBD}

TLS_PSK_WITH_ESTREAM_SALSA20_UMAC96         = {0xTBD, 0xTBD}
TLS_PSK_WITH_SALSA20_UMAC96                 = {0xTBD, 0xTBD}
TLS_DHE_PSK_WITH_ESTREAM_SALSA20_UMAC96     = {0xTBD, 0xTBD}
TLS_DHE_PSK_WITH_SALSA20_UMAC96             = {0xTBD, 0xTBD}
TLS_RSA_PSK_WITH_ESTREAM_SALSA20_UMAC96     = {0xTBD, 0xTBD}
TLS_RSA_PSK_WITH_SALSA20_UMAC96             = {0xTBD, 0xTBD}
TLS_ECDHE_PSK_WITH_ESTREAM_SALSA20_UMAC96   = {0xTBD, 0xTBD}
TLS_ECDHE_PSK_WITH_SALSA20_UMAC96           = {0xTBD, 0xTBD}
```

Note that both Salsa20 and UMAC-96 are used with a 64-bit nonce. That nonce is set to be the 64-bit TLS record sequence number. In case of DTLS the 64-bit nonce is formed as the concatenation of the 16-bit epoch with the 48-bit sequence number.

The RSA, DHE_RSA, ECDHE_RSA, ECDHE_ECDSA, PSK, DHE_PSK, RSA_PSK, ECDHE_PSK key exchanges are performed as defined in [RFC5246], [RFC4492], and [RFC5489].

3. The TLS GenericStreamCipher

The ciphersuites defined in this document differ from the TLS RC4 ciphersuites that have been the basis for the definition of GenericStreamCipher. Unlike RC4, Salsa20 requires a nonce per record. This however, does not affect the description of the GenericStreamCipher if one assumes that a nonce is optional and depends on the cipher's characteristics (in that case RC4 uses a 0 byte nonce, and Salsa20 an 8-byte nonce).

Moreover, in order to accommodate MAC algorithms like UMAC that require a nonce as part of their operation, the document extends the MAC algorithm as specified in the TLS protocol. The extended MAC includes a nonce as a second parameter. MAC algorithms that do not require a nonce, such as HMAC, are assumed to ignore the nonce input value. The MAC in a GenericStreamCipher is then calculated as follows.

```
      MAC(MAC_write_key, nonce,
                            seq_num +
                            TLSCompressed.type +
                            TLSCompressed.version +
                            TLSCompressed.length +
                            TLSCompressed.fragment);
```

where "+" denotes concatenation.

nonce The nonce for this record. If the size of the nonce accepted by the MAC is 64-bits then nonce equals the sequence number (or the concatenation of the 16-bit epoch with the 48-bit sequence number in DTLS). Otherwise the MAC algorithm must specify how the nonce is formed.

seq_num The sequence number for this record.

MAC The MAC algorithm specified by SecurityParameters.mac_algorithm.

As specified in TLS [RFC5246] the MAC is computed before encryption and the stream cipher encrypts the entire block, including the MAC.

4. Acknowledgements

The authors would like to thank D. J. Bernstein, David McGrew, Wan-Teh Chang, and Adam Langley for discussion and suggestions.

5. IANA Considerations

IANA is requested to allocate the following numbers in the TLS Cipher Suite Registry:

TLS_RSA_WITH_ESTREAM_SALSA20_SHA1	= {0xTBD, 0xTBD}
TLS_RSA_WITH_SALSA20_SHA1	= {0xTBD, 0xTBD}
TLS_DHE_RSA_WITH_ESTREAM_SALSA20_SHA1	= {0xTBD, 0xTBD}
TLS_DHE_RSA_WITH_SALSA20_SHA1	= {0xTBD, 0xTBD}
TLS_ECDHE_RSA_WITH_ESTREAM_SALSA20_SHA1	= {0xTBD, 0xTBD}
TLS_ECDHE_RSA_WITH_SALSA20_SHA1	= {0xTBD, 0xTBD}
TLS_ECDHE_ECDSA_WITH_ESTREAM_SALSA20_SHA1	= {0xTBD, 0xTBD}
TLS_ECDHE_ECDSA_WITH_SALSA20_SHA1	= {0xTBD, 0xTBD}
TLS_PSK_WITH_ESTREAM_SALSA20_SHA1	= {0xTBD, 0xTBD}
TLS_PSK_WITH_SALSA20_SHA1	= {0xTBD, 0xTBD}
TLS_DHE_PSK_WITH_ESTREAM_SALSA20_SHA1	= {0xTBD, 0xTBD}
TLS_DHE_PSK_WITH_SALSA20_SHA1	= {0xTBD, 0xTBD}
TLS_RSA_PSK_WITH_ESTREAM_SALSA20_SHA1	= {0xTBD, 0xTBD}
TLS_RSA_PSK_WITH_SALSA20_SHA1	= {0xTBD, 0xTBD}
TLS_ECDHE_PSK_WITH_ESTREAM_SALSA20_SHA1	= {0xTBD, 0xTBD}
TLS_ECDHE_PSK_WITH_SALSA20_SHA1	= {0xTBD, 0xTBD}
TLS_RSA_WITH_ESTREAM_SALSA20_UMAC96	= {0xTBD, 0xTBD}
TLS_RSA_WITH_SALSA20_UMAC96	= {0xTBD, 0xTBD}
TLS_DHE_RSA_WITH_ESTREAM_SALSA20_UMAC96	= {0xTBD, 0xTBD}
TLS_DHE_RSA_WITH_SALSA20_UMAC96	= {0xTBD, 0xTBD}
TLS_ECDHE_RSA_WITH_ESTREAM_SALSA20_UMAC96	= {0xTBD, 0xTBD}
TLS_ECDHE_RSA_WITH_SALSA20_UMAC96	= {0xTBD, 0xTBD}
TLS_ECDHE_ECDSA_WITH_ESTREAM_SALSA20_UMAC96	= {0xTBD, 0xTBD}
TLS_ECDHE_ECDSA_WITH_SALSA20_UMAC96	= {0xTBD, 0xTBD}
TLS_PSK_WITH_ESTREAM_SALSA20_UMAC96	= {0xTBD, 0xTBD}
TLS_PSK_WITH_SALSA20_UMAC96	= {0xTBD, 0xTBD}
TLS_DHE_PSK_WITH_ESTREAM_SALSA20_UMAC96	= {0xTBD, 0xTBD}
TLS_DHE_PSK_WITH_SALSA20_UMAC96	= {0xTBD, 0xTBD}
TLS_RSA_PSK_WITH_ESTREAM_SALSA20_UMAC96	= {0xTBD, 0xTBD}
TLS_RSA_PSK_WITH_SALSA20_UMAC96	= {0xTBD, 0xTBD}
TLS_ECDHE_PSK_WITH_ESTREAM_SALSA20_UMAC96	= {0xTBD, 0xTBD}
TLS_ECDHE_PSK_WITH_SALSA20_UMAC96	= {0xTBD, 0xTBD}

6. Security Considerations

The security of Salsa20 is discussed in the Salsa20 security [SALSA20-SECURITY] paper. At the time of writing this document, there are no known significant security problems with the eSTREAM variant of Salsa20, nor with the original 20 round variant. As of early 2013, the best cryptanalysis breaks 8 out of 20 rounds to recover the 256-bit secret key in 2^251 operations, using 2^31 keystream pairs (see [SALSA20-ATTACK]). For more background, see the eSTREAM report [ESTREAM].

There are no ciphersuites defined in this document that utilize the variant of Salsa20 with 128-bit key material, because (due to the design of Salsa20) they provide no performance advantage over the 256-bit variant.

The ciphersuites that utilize UMAC-96 use a short MAC (96-bits), to be consistent with the MAC size used in the TLS Finished messages, which is also 96-bits, and thus allow room for more data in TLS records. The security considerations of [RFC4418] also apply.

This document should not introduce any other security considerations than those that directly follow from any use of the stream cipher Salsa20 and those that directly follow from introducing any set of stream cipher suites into TLS and DTLS.

7. References

7.1. Normative References

[RFC2246] Dierks, T. and C. Allen, "The TLS Protocol Version 1.0", RFC 2246, January 1999.

[RFC4346] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.1", RFC 4346, April 2006.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.

[RFC4347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security", RFC 4347, April 2006.

[RFC4418] Krovetz, T., "UMAC: Message Authentication Code using Universal Hashing", RFC 4418, March 2006.

[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, May 2006.

[RFC5489] Badra, M. and I. Hajjeh, "ECDHE_PSK Cipher Suites for Transport Layer Security (TLS)", RFC 5489, March 2009.

[RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, January 2012.

[RFC6234] Eastlake, D. and T. Hansen, "US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF)", RFC 6234, May 2011.

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

7.2. Informative References

[RFC6101] Freier, A., Karlton, P., and P. Kocher, "The Secure Sockets Layer (SSL) Protocol Version 3.0", RFC 6101, August 2011.

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

[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)", WWW http://www.ecrypt.eu.org/stream/finallist.html, September 2008.

[CBC-ATTACK]
AlFardan, N. and K. Paterson, "Lucky Thirteen: Breaking the TLS and DTLS Record Protocols", IEEE Symposium on Security and Privacy , 2013.

[RC4-ATTACK]
ISOBE, T., OHIGASHI, T., WATANABE, Y., and M. MORII, "Full Plaintext Recovery Attack on Broadcast RC4", International Workshop on Fast Software Encryption , 2013.

[AEAD-PERFORMANCE]
Krovetz, T. and P. Rogaway, "The Software Performance of Authenticated-Encryption Modes", International Workshop on Fast Software Encryption , 2011.

[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", WWW http://eprint.iacr.org/2007/472.pdf, 2007.

Authors' Addresses

Simon Josefsson
SJD AB

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

Joachim Strombergson
Secworks Sweden AB

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

Nikos Mavrogiannopoulos
KU Leuven

Email: nikos.mavrogiannopoulos@esat.kuleuven.be