Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 22, 2016

C. Jennings
P. Jones
Cisco Systems
A. Roach
Mozilla

March 21, 2016

SRTP Double Encryption Procedures
draft-jennings-perc-double-01

Abstract

In some conferencing scenarios, it is desirable for an intermediary to be able to manipulate some RTP parameters, while still providing strong end-to-end security guarantees. This document defines SRTP procedures that use two separate but related cryptographic contexts to provide "hop-by-hop" and "end-to-end" security guarantees. Both the end-to-end and hop-by-hop cryptographic transforms can utilize an authenticated encryption with associated data scheme or take advantage of future SRTP transforms with different properties.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Cloud conferencing systems that are based on switched conferencing have a central media distribution device (MDD) that receives media from endpoints and distributes it to other endpoints, but does not need to interpret or change the media content. For these systems, it is desirable to have one cryptographic context from the sending endpoint to the receiving endpoint that can encrypt and authenticate the media end-to-end while still allowing certain RTP header information to be changed by the MDD. At the same time, a separate cryptographic context provides integrity and optional confidentiality for the media flowing between the MDD and the endpoints. See the framework document that describes this concept in more detail in more detail in [I-D.jones-perc-private-media-framework].

This specification RECOMMENDS the SRTP AES-GCM transform [RFC7714] to encrypt an RTP packet for the end-to-end cryptographic context. The output of this is treated as an RTP packet and again encrypted with an SRTP transform used in the hop-by-hop cryptographic context between the endpoint and the MDD. The MDD decrypts and checks

integrity of the hop-by-hop security. The MDD MAY change some of the RTP header information that would impact the end-to-end integrity. The original value of any RTP header field that is changed is included in a new RTP header extension called the Original Header Block. The new RTP packet is encrypted with the hop-by-hop cryptographic transform before it is sent. The receiving endpoint decrypts and checks integrity using the hop-by-hop cryptographic transform and then replaces any parameters the MDD changed using the information in the Original Header Block before decrypting and checking the end-to-end integrity.

2. 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 [RFC2119].

Terms used throughout this document include:

- MDD: media distribution device that routes media from one endpoint to other endpoints

- E2E: end-to-end, meaning the link from one endpoint through one or more MDDs to the endpoint at the other end.

- HBH: hop-by-hop, meaning the link from the endpoint to or from the MDD.

- OHB: Original Header Block is an RTP header extension that contains the original values from the RTP header that might have been changed by an MDD.

3. Cryptographic Contexts

This specification uses two cryptographic contexts: an inner ("end-to-end") context that is used by endpoints that originate and consume media to ensure the integrity of media end-to-end, and an outer ("hop-by-hop") context that is used between endpoints and MDDs to ensure the integrity of media over a single hop and to enable an MDD to modify certain RTP header fields. RTCP is also encrypted using the hop-by-hop cryptographic context. The RECOMMENDED cipher for the hop-by-hop and end-to-end contexts is AES-GCM. Other combinations of SRTP ciphers that support the procedures in this document can be added to the IANA registry.

The keys and salt for these contexts are generated with the following steps:

- Generate key and salt values of the length required for the combined inner (end-to-end) and outer (hop-by-hop) transforms.

- Assign the key and salt values generated for the inner (end-to-end) transform.

- Assign the key and salt values for the outer (hop-by-hop) transform.

As a special case, the outer cryptographic transform MAY be the NULL cipher (see [RFC3711]) if a secure transport such as [RFC6347] is used over a hop (i.e., between an endpoint and MDD or between two MDDs). In that case, the key and salt values generated would be the length required only for the inner cryptographic transform.

Obviously, if the MDD is to be able to modify header fields but not decrypt the payload, then it must have cryptographic context for the outer transform, but not the inner transform. This document does not define how the MDD should be provisioned with this information. One possible way to provide keying material for the outer ("hop-by-hop") transform is to use [I-D.jones-perc-dtls-tunnel].

4. Original Header Block

Any SRTP packet processed following these procedures MAY contain an Original Header Block (OHB) RTP header extension.

The OHB contains the original values of any modified header fields and MUST be placed after any already-existing RTP header extensions. Placement of the OHB after any original header extensions is important so that the receiving endpoint can properly authenticate the original packet and any originally included RTP header extensions. The receiving endpoint will authenticate the original packet by restoring the modified RTP header field values and header extensions. It does this by copying the original values from the OHB and then removing the OHB extension and any other RTP header extensions that appear after the OHB extension.

The MDD is only permitted to modify the extension (X) bit, payload type (PT) field, and the RTP sequence number field.

The OHB extension is either one octet in length, two octets in length, or three octets in length. The length of the OHB indicates what data is contained in the extension.

If the OHB is one octet in length, it contains both the original X bit and PT field value. In this case, the OHB has this form:

```
 0
 0 1 2 3 4 5 6 7
+-+-+-+-+-+-+-+-+
|X|     PT      |
+-+-+-+-+-+-+-+-+
```

If the OHB is two octets in length, it contains the original RTP packet sequence number. In this case, the OHB has this form:

```
 0                   1
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        Sequence Number        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

If the OHB is three octets in length, it contains the original X bit, PT field value, and RTP packet sequence number. In this case, the OHB has this form:

```
 0                   1                   2
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|X|     PT      |        Sequence Number        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

If an MDD modifies an original RTP header value, the MDD MUST include the OHB extension to reflect the changed value. If another MDD along the media path makes additional changes to the RTP header and any original value is not already present in the OHB, the MDD must extend the OHB by adding the changed value to the OHB. To properly preserve original RTP header values, an MDD MUST NOT change a value already present in the OHB extension.

5. RTP Operations

5.1. Encrypting a Packet

To encrypt a packet, the endpoint encrypts the packet using the inner cryptographic context and then encrypts using the outer cryptographic context. The processes is as follows:

- o Form an RTP packet. If there are any header extensions, they MUST use [RFC5285].

- o Apply the inner cryptographic transform to the RTP packet. If encrypting RTP header extensions end-to-end, then [RFC6904] MUST be used when encrypting the RTP packet using the inner cryptographic context.

- If the endpoint wishes to insert header extensions that can be modified by an MDD, it MUST insert an OHB header extension at the end of any header extensions protected end-to-end, then add any MDD-modifiable header extensions. The OHB MUST replicate the information found in the RTP header following the application of the inner cryptographic transform. For example, if the packet had no header extensions when the inner cryptographic transform was applied, the X bit would be 0. If the endpoint introduces an OHB and then adds MDD-modifiable header extensions, the X bit in the OHB would be 0. After introducing the OHB and MDD-modifiable header extensions, of course, the X bit in the RTP header would be set to 1.

- Apply the outer cryptographic transform to the RTP packet. If encrypting RTP header extensions hop-by-hop, then [RFC6904] MUST be used when encrypting the RTP packet using the outer cryptographic context.

5.2. Modifying a Packet

The MDD does not have a notion of outer or inner cryptographic contexts. Rather, the MDD has a single cryptographic context. The cryptographic transform and key used to decrypt a packet and any encrypted RTP header extensions would be the same as those used in the endpoint's outer cryptographic context.

In order to modify a packet, the MDD decrypts the packet, modifies the packet, updates the OHB with any modifications not already present in the OHB, and re-encrypts the packet using the cryptographic context used for next hop.

- Apply the cryptographic transform to the packet. If decrypting RTP header extensions hop-by-hop, then [RFC6904] MUST be used.

- Change any required parameters

- If a changed RTP header field is not already in the OHB, add it with its original value to the OHB. An MDD MAY add information to the OHB, but MUST NOT change existing information in the OHB.

- If the MDD resets a parameter to its original value, it MAY drop it from the OHB as long as there are no other header extensions following the OHB. Note that this might result in a decrease in the size of the OHB.

- The MDD MUST NOT delete any header extensions before the OHB, but MAY add, delete, or modify any that follow the OHB.

  - If the MDD adds any header extensions, it must append them and it must maintain the order of the original header extensions in the [RFC5285] block.

  - If the MDD appends header extensions, then it MUST add the OHB header extension (if not present), even if the OHB merely replicates the original header field values, and append the new extensions following the OHB. The OHB serves as a demarcation point between original RTP header extensions introduced by the endpoint and those introduced by an MDD.

- The MDD MAY modify any header extension appearing after the OHB, but MUST NOT modify header extensions that are present before the OHB.

- Apply the cryptographic transform to the packet. If encrypting RTP header extensions hop-by-hop, then [RFC6904] MUST be used.

5.3. Decrypting a Packet

To decrypt a packet, the endpoint first decrypts and verifies using the outer cryptographic context, then uses the OHB to reconstruct the original packet, which it decrypts and verifies with the inner cryptographic context.

- Apply the outer cryptographic transform to the packet. If the integrity check does not pass, discard the packet. The result of this is referred to as the outer SRTP packet. If decrypting RTP header extensions hop-by-hop, then [RFC6904] MUST be used when decrypting the RTP packet using the outer cryptographic context.

- Form a new synthetic SRTP packet with:

  - Header = Received header, with header fields replaced with values from OHB (if present).

  - Insert all header extensions up to the OHB extension, but exclude the OHB and any header extensions that follow the OHB. If the original X bit is 1, then the remaining extensions MUST be padded to the first 32-bit boundary and the overall length of the header extensions adjusted accordingly. If the original X bit is 0, then the header extensions would be removed entirely.

  - Payload is the original encrypted payload.

- Apply the inner cryptographic transform to this synthetic SRTP packet. If the integrity check does not pass, discard the packet.

If decrypting RTP header extensions end-to-end, then [RFC6904] MUST be used when decrypting the RTP packet using the inner cryptographic context.

Once the packet has successfully decrypted, the application needs to be careful about which information it uses to get the correct behavior. The application MUST use only the information found in the synthetic SRTP packet and MUST NOT use the other data that was in the outer SRTP packet with the following exceptions:

- The PT from the outer SRTP packet is used for normal matching to SDP and codec selection.

- The sequence number from the outer SRTP packet is used for normal RTP ordering.

If any of the following RTP headers extensions are found in the outer SRTP packet, they MAY be used:

- TBD

6. RTCP Operations

Unlike RTP, which is encrypted both hop-by-hop and end-to-end using two separate cryptographic contexts, RTCP is encrypted using only the outer (HBH) cryptographic context. The procedures for RTCP encryption are specified in [RFC3711] and this document introduces no additional steps.

7. Recommended Inner and Outer Cryptographic Transforms

This specification recommends and defines AES-GCM as both the inner and outer cryptographic transforms, identified as DOUBLE_AEAD_AES_128_GCM_AEAD_AES_128_GCM and DOUBLE_AEAD_AES_256_GCM_AEAD_AES_256_GCM. These transforms provide for authenticated encryption and will consume additional processing time double-encrypting for HBH and E2E. However, the approach is secure and simple, and is thus viewed as an acceptable trade-off in processing efficiency.

Note that names for the cryptographic transforms are of the form DOUBLE_(inner transform)_(outer transform).

This specification also allows for the NULL cipher to be used as the outer cryptographic transform in cases where a secure transport is used over the hop, with those transforms identified as DOUBLE_AEAD_AES_128_GCM_NULL_NULL and DOUBLE_AEAD_AES_256_GCM_NULL_NULL.

Open Issue: It is not clear if the NULL ciphers are needed or not. The authors plan to remove them from the next version of the draft unless there is a reasonable support and reasons to keep them in.

While this document only defines a profile based on AES-GCM, it is possible for future documents to define further profiles with different inner and outer transforms in this same framework. For example, if a new SRTP transform was defined that encrypts some or all of the RTP header, it would be reasonable for systems to have the option of using that for the outer transform. Similarly, if a new transform was defined that provided only integrity, that would also be reasonable to use for the HBH as the payload data is already encrypted by the E2E.

The AES-GCM cryptographic transform introduces an additional 16 octets to the length of the packet. When using AES-GCM for both the inner and outer cryptographic transforms, the total additional length is 32 octets. If no other header extensions are present in the packet and the OHB is introduced, that will consume an additional 8 octets. If other extensions are already present, the OHB will consume up to 4 additional octets.

8. Security Considerations

It is obviously critical that the intermediary have only the outer transform parameters and not the inner transform parameters. We rely on an external key management protocol to assure this property.

Modifications by the intermediary result in the recipient getting two values for changed parameters (original and modified). The recipient will have to choose which to use; there is risk in using either that depends on the session setup.

The security properties for both the inner and outer key holders are the same as the security properties of classic SRTP.

The NULL cipher MUST be used in conjunction with an encrypted transport for both RTP and RTCP. Use of the NULL cipher for the outer cryptographic context without the use of an encrypted transport exposes the RTCP traffic to undetectable modification as it is transmitted over the network. Likewise, RTP traffic under the same conditions would be subject to modification that would not be detectable by the MDD. While the endpoint could detect modification of the end-to-end information, reliance on information like payload type value in the packet received from the MDD could present problems such as attempting to decode media with the wrong codec.

9. IANA Considerations

9.1. RTP Header Extension

This document defines a new extension URI in the RTP Compact Header Extensions part of the Real-Time Transport Protocol (RTP) Parameters registry, according to the following data:

Extension URI: urn:ietf:params:rtp-hdrext:ohb

Description: Original Header Block

Contact: Cullen Jennings <mailto:fluffy@iii.ca>

Reference: RFCXXXX

Note to RFC Editor: Replace RFCXXXX with the RFC number of this specification.

9.2. DTLS-SRTP

We request IANA to add the following values to defines a DTLS-SRTP "SRTP Protection Profile" defined in [RFC5764].

Value	Profile	Reference
{TBD, TBD}	DOUBLE_AEAD_AES_128_GCM_AEAD_AES_128_GCM	RFCXXXX
{TBD, TBD}	DOUBLE_AEAD_AES_256_GCM_AEAD_AES_256_GCM	RFCXXXX
{TBD, TBD}	DOUBLE_AEAD_AES_128_GCM_NULL_NULL	RFCXXXX
{TBD, TBD}	DOUBLE_AEAD_AES_256_GCM_NULL_NULL	RFCXXXX

Note to IANA: Please assign value RFCXXXX and update table to point at this RFC for these values.

The SRTP transform parameters for each of these protection are:

DOUBLE_AEAD_AES_128_GCM_AEAD_AES_128_GCM

cipher:	AES_128_GCM then AES_128_GCM
cipher_key_length:	256 bits
cipher_salt_length:	192 bits
aead_auth_tag_length:	32 octets
auth_function:	NULL
auth_key_length:	N/A
auth_tag_length:	N/A
maximum lifetime:	at most 2^31 SRTCP packets and at most 2^48 SRTP packets

DOUBLE_AEAD_AES_256_GCM_AEAD_AES_256_GCM

cipher:	AES_256_GCM then AES_256_GCM
cipher_key_length:	512 bits
cipher_salt_length:	192 bits
aead_auth_tag_length:	32 octets
auth_function:	NULL
auth_key_length:	N/A
auth_tag_length:	N/A
maximum lifetime:	at most 2^31 SRTCP packets and at most 2^48 SRTP packets

DOUBLE_AEAD_AES_128_GCM_NULL_NULL

cipher:	AES_128_GCM then identity transform
cipher_key_length:	128 bits
cipher_salt_length:	96 bits
aead_auth_tag_length:	16 octets
auth_function:	NULL
auth_key_length:	N/A
auth_tag_length:	N/A
maximum lifetime:	at most 2^31 SRTCP packets and at most 2^48 SRTP packets

DOUBLE_AEAD_AES_256_GCM_NULL_NULL

cipher:	AES_256_GCM then identity transform
cipher_key_length:	256 bits
cipher_salt_length:	96 bits
aead_auth_tag_length:	16 octets
auth_function:	NULL
auth_key_length:	N/A
auth_tag_length:	N/A
maximum lifetime:	at most 2^31 SRTCP packets and at most 2^48 SRTP packets

Except when the NULL cipher is used for the outer (HBH) transform, the first half of the key and salt is used for the inner (E2E) transform and the second half is used for the outer (HBH) transform.

For those that use the NULL cipher for the outer transform, the the key and salt is applied only to the inner transform.

10. Acknowledgments

Many thanks to review from GET YOUR NAME HERE. Please, send comments.

11. References

11.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC3711] Baugher, M., McGrew, D., Naslund, M., Carrara, E., and K. Norrman, "The Secure Real-time Transport Protocol (SRTP)", RFC 3711, DOI 10.17487/RFC3711, March 2004, <http://www.rfc-editor.org/info/rfc3711>.

[RFC5285] Singer, D. and H. Desineni, "A General Mechanism for RTP Header Extensions", RFC 5285, DOI 10.17487/RFC5285, July 2008, <http://www.rfc-editor.org/info/rfc5285>.

[RFC5764] McGrew, D. and E. Rescorla, "Datagram Transport Layer Security (DTLS) Extension to Establish Keys for the Secure Real-time Transport Protocol (SRTP)", RFC 5764, DOI 10.17487/RFC5764, May 2010, <http://www.rfc-editor.org/info/rfc5764>.

[RFC6904] Lennox, J., "Encryption of Header Extensions in the Secure Real-time Transport Protocol (SRTP)", RFC 6904, DOI 10.17487/RFC6904, April 2013, <http://www.rfc-editor.org/info/rfc6904>.

[RFC7714] McGrew, D. and K. Igoe, "AES-GCM Authenticated Encryption in the Secure Real-time Transport Protocol (SRTP)", RFC 7714, DOI 10.17487/RFC7714, December 2015, <http://www.rfc-editor.org/info/rfc7714>.

11.2. Informative References

[I-D.jones-perc-dtls-tunnel]
Jones, P., "DTLS Tunnel between Media Distribution Device and Key Management Function to Facilitate Key Exchange", draft-jones-perc-dtls-tunnel-02 (work in progress), March 2016.

[I-D.jones-perc-private-media-framework]
Jones, P., Ismail, N., and D. Benham, "A Solution Framework for Private Media in Privacy Enhanced RTP Conferencing", draft-jones-perc-private-media-framework-02 (work in progress), March 2016.

[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>.

Authors' Addresses

Cullen Jennings
Cisco Systems

Email: fluffy@iii.ca

Paul E. Jones
Cisco Systems

Email: paulej@packetizer.com

Adam Roach
Mozilla

Email: adam@nostrum.com