Workgroup: mmusic
Internet-Draft:
draft-davis-mmusic-srtp-assurance-03
Updates: 4568 (if approved)
Published: 10 January 2024
Intended Status: Standards Track
Expires: 13 July 2024
Authors: K. R. Davis, Cisco Systems
E. Valverde, Cisco Systems
G. Salgueiro, Cisco Systems

Signaling Additional SRTP Context information via SDP

Abstract

This document specifies additional cryptographic attributes for signaling additional Secure Real-time Transport Protocol (SRTP) cryptographic context information via the Session Description Protocol (SDP) in alongside those defined by RFC4568.

The SDP extension defined in this document address situations where the receiver needs to quickly and robustly synchronize with a given sender. The mechanism also enhances SRTP operation in cases where there is a risk of losing sender-receiver synchronization.

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 https://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 13 July 2024.

Table of Contents

1. Introduction

1.1. Discussion Venues

This section is to be removed before publishing as an RFC.

Source for this draft and an issue tracker can be found at https://github.com/kyzer-davis/srtp-assurance-rfc-draft.

1.2. Changelog

This section is to be removed before publishing as an RFC.

draft-03

*Consolidate references to late binding, fix nits #20

draft-02

*Better define that the tags must match #16
*Revise ANBF #15
*Handling and Signaling Unknown Values #18
*Cite default behavior of underlying RFCs if value is unknown/ omitted #17

draft-01

*Change contact name from IESG to IETF in IANA Considerations #2
*Discuss RFC4568 "Late Joiner" in problem statement: #3
*Split Serial forking scenario into its own section #4
*Add MIKEY considerations to Protocol Design section #6
*Change doc title #7
*Add SEQ abbreviation earlier #8
*Discuss why this can't be a RTP Header Extension #11
*Add Appendix further discussing why SDP Security Session Parameters extension not used #5
*Method to Convey Multiple SSRCs for a given stream #1
*Discuss why SEQ is signaled in the SDP #9

1.3. Problem Statement

While [RFC4568] provides most of the information required to instantiate an SRTP cryptographic context for RTP Packets, the state of a few crucial items in the SRTP cryptographic context are missing. One such item is the Rollover Counter (ROC) defined by Section 3.2.1 [RFC3711] which is not signaled in any packet across the wire and shared between applications.

The ROC is one item that is used to create the SRTP Packet Index along with the the [RFC3550] transmitted sequence numbers (SEQ) for a given synchronization sources (SSRC). The Packet index is integral to the encryption, decryption and authentication process of SRTP key streams. Failure to synchronize the value properly at any point in the SRTP media exchange leads to encryption or decryption failures, degraded user experience and at cross-vendor interoperability issues with many hours of engineering time spent debugging a value that is never negotiated on the wire (and oftentimes not even logged in application logs.)

The current method of ROC handling is to instantiate a new media stream's cryptographic context at 0 as per Section 3.3.1 of [RFC3711]. Then track the state ROC for a given cryptographic context as the time continues on and the stream progresses.

[RFC4568], states 'there is no concept of a "late joiner" in SRTP security descriptions' as the main reason for not conveying the ROC, SSRC, or SEQ via the key management protocol but as one will see below; this argument is not true in practice.

When joining ongoing streams, resuming held/transferred streams, or devices without embedded application logic for clustering/high availability where a given cryptographic context is resumed; without any explicit signaling about the ROC state, devices must make an educated guess as defined by Section 3.3.1 of [RFC3711]. The method specially estimates the received ROC by calculating ROC-1, ROC, ROC+1 to see which performs a successful decrypt. While this may work on paper, this process usually only done at the initial instantiation of a cryptographic context rather than at later points later during the session. Instead many applications take the easy route and set the value at 0 as if this is a new stream. While technically true from that receivers perspective, the sender of this stream may be encrypting packets with a ROC greater than 0. Further this does not cover scenarios where the ROC is greater than +1.

Where possible the ROC state (and the rest of the cryptographic context) is usually synced across clustered devices or high availability pairs via proprietary methods rather than open standards.

These problems detailed technically above lead to a few very common scenarios where the ROC may become out of sync. These are are briefly detailed below with the focus on the ROC Value.

Joining an ongoing session:

- When a receiver joins an ongoing session, such as a broadcast conference, there is no signaling method which can quickly allow the new participant to know the state of the ROC assuming the state of the stream is shared across all participants.

Hold/Resume, Transfer Scenarios:

- A session is created between sender A and receiver B. ROC is instantiated at 0 normally and continues as expected.
- At some point the receiver is put on hold while the sender is connected to some other location such as music on hold or another party altogether.
- At some future point the receiver is reconnected to the sender and the original session is resumed.
- The sender may re-assume the original cryptographic context rather rather than create one net new.
- Here if the sender starts the stream from the last observed sequence number the receiver observed the ROC will be in sync.

* However there are scenarios where the sender may have been transmitting packets on the previous cryptographic context and if a ROC increment occurred; the receiver would never know. This can lead to problems when the streams are reconnected as the ROC is now out of sync between both parties.
* Further, a sender may be transferred to some upstream device transparently to them. If the sender does not reset their cryptographic context that new receiver will now be out of sync with possible ROC values.

Serial Forking Case:

* [RFC4568] itself cites a problematic scenario in their own Appendix A, Scenario B, Problem 3 where a ROC out of sync scenario could occur.
* The proposed solution for problem 3 involves a method to convey the ROC however known the problem; the authors still did not include this in the base SDP Security specification.

Application Failover (without stateful syncs):

* In this scenario a cryptographic context was was created with Device A and B of a high availability pair.
* An SRTP stream was created and ROC of 0 was created and media streamed from the source towards Device A.
* Time continues and the sequence wraps from 65535 to 0 and the ROC is incremented to 1.
* Both the sender and device A are tracking this locally and the encrypt/decrypt process proceeds normally.
* Unfortunate network conditions arise and Device B must assume sessions of Device A transparently.
* Without any proprietary syncing logic between Device A and B which disclose the state of the ROC, Device B will likely instantiate the ROC at 0.
* Alternatively Device B may try to renegotiate the stream over the desired signaling protocol however this does not ensure the remote sender will change their cryptographic context and reset the ROC to 0.
* The transparent nature of the upstream failover means the local application will likely proceed using ROC 1 while upstream receiver has no method of knowing ROC 1 is the current value.

Secure SIPREC Recording:

* If a SIPREC recorder is brought into recording an ongoing session through some form of transfer or on-demand recording solution the ROC may have incremented.

*Without an SDP mechanism to share this information the SIPREC will be unaware of the full SRTP context required to ensure proper decrypt of media streams being monitored.

Improper SRTP context resets:

*As defined by Section 3.3.1 of [RFC3711] an SRTP re-key **MUST NOT** reset the ROC within SRTP Cryptographic context.
*However, some applications may incorrectly use the re-key event as a trigger to reset the ROC leading to out-of-sync encrypt/decrypt operations.

Out of Sync Sliding Windows / Sequence Numbers:

*There is corner case situation where two devices communicating via a Back to Back User Agent (B2BUA) which is performing RTP-SRTP inter-working.
*In this scenario the B2BUA is also a session border controller (SBC) tasked with topology abstraction. That is, the signaling itself is abstracted from both parties.
*In this scenario a hold/resume where a sequence rolls can not only cause problems with the ROC; but can also cause sliding window issues.
*To be more specific, assume that both parties did have access to the cryptographic context and resumed the old ROC value after the hold thus ROC is not out of sync.
*What should the sliding window and sequence be set to in this scenario?
*The post-hold call could in theory have a problem where the sequence number of received packets is lower than what was originally observed before the hold.
*Thus the sliding window would drop packets until the sequence number gets back to the last known sequence and the sliding window advances.
*Advertising the Sequence in some capacity to reinitialize the sliding window (along with advertising the ROC) can ensure a remote application can properly re-instantiate the cryptographic context in this scenario.

This is a problem that other SRTP Key Management protocols (MIKEY, DTLS-SRTP, EKT-SRTP) have solved but SDP Security has lagged behind in solution parity. For a quick comparison of all SRTP Key Management negotiations refer to [RFC7201] and [RFC5479].

1.4. Previous Solutions

As per RFC3711, "Receivers joining an on-going session **MUST** be given the current ROC value using out-of-band signaling such as key-

management signaling." [RFC4771] aimed to solve the problem however this solution has a few technical shortcomings detailed below.

First, this specifies the use of Multimedia Internet KEYing (MIKEY) defined by [RFC3830] as the out-of-band signaling method. A proper MIKEY implementation requires more overhead than is needed to convey and solve this problem. By selecting MIKEY as the out-of-band signaling method the authors may have inadvertently inhibited significant adoption by the industry.

Second, [RFC4771] also transforms the SRTP Packet to include the four byte value after the encrypted payload and before an optional authentication tag. This data about the SRTP context is unencrypted on the wire and not covered by newer SRTP encryption protocols such as [RFC6904] and [RFC9335]. Furthermore this makes the approach incompatible with AEAD SRTP Cipher Suites which state that trimming/ truncating the authentication tag weakens the security of the protocol in Section 13.2 of [RFC7714].

Third, this is not in line with the standard method of RTP Packet modifications. The proposal would have benefited greatly from being an RTP Header Extension rather than a value appended after payload. But even an RTP header extension proves problematic in where modern SRTP encryption such as Cryptex defined by [RFC9335] are applied. That is, the ROC is a required input to decrypt the RTP packet contents. It does not make sense to convey this data as an RTP Header Extension obfuscated by the very encryption it is required to decrypt.

Lastly, there is no defined method for applications defined for applications to advertise the usage of this protocol via any signaling methods.

[RFC5159] also defined some SDP attributes namely the "a=SRTPROCTxRate" attribute however this does not cover other important values in the SRTP Cryptographic context and has not seen widespread implementation.

[RFC8870] solves the problem for DTLS-SRTP [RFC5763]/[RFC5764] implementations.

2. Conventions and Definitions

The key words "**MUST**", "**MUST NOT**", "**REQUIRED**", "**SHALL**", "**SHALL NOT**", "**SHOULD**", "**SHOULD NOT**", "**RECOMMENDED**", "**NOT RECOMMENDED**", "**MAY**", and "**OPTIONAL**" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. Protocol Design

A few points of note are below about this specifications relationship to other SRTP Key Management protocols or SRTP protocols as to leave no ambiguity.

Session Description Protocol (SDP) Security Descriptions for Media Streams:

The authors have chosen to avoid modifying RFC4568 a=crypto offers as to avoid backwards compatibility issues with a non-versioned protocol. Instead this specification adds to what is defined in SDP Security Framework [RFC4568] by allowing applications to explicitly negotiate additional items from the cryptographic context such as the packet index ingredients: ROC, SSRC and Sequence Number via a new SDP Attribute. By coupling this information with the applicable "a=crypto" offers; a receiving application can properly instantiate an SRTP cryptographic context at the start of a session, later in a session, after session modification or when joining an ongoing session.

Key Management Extensions for Session Description Protocol (SDP) and Real Time Streaming Protocol (RTSP):

This specifications makes no attempt to be compatible with the Key Management Extension for SDP "a=key-mgmt" defined by [RFC4567]

ZRTP: Media Path Key Agreement for Unicast Secure RTP:

This specifications makes no attempt to be compatible with the Key Management via SDP for ZRTP "a=zrtp-hash" defined by [RFC6189].

MIKEY:

This specifications makes no attempt to be compatible with the SRTP Key Management via MIKEY [RFC3830].

DTLS-SRTP, EKT-SRTP, Privacy Enhanced Conferencing items (PERC):

All DTLS-SRTP items including Privacy Enhanced Conferencing items (PERC) ([RFC8723] and [RFC8871]) are out of scope for the purposes of this specification.

Secure Real Time Control Protocol (SRTCP):

This specification is not required by SRTCP since the packet index is carried within the SRTCP packet and does not need an out-of-band equivalent.

Source-Specific Media Attributes in the Session Description Protocol (SDP):

The authors of this specification vetted [RFC5576] SSRC Attribute "a=ssrc" but felt that it would require too much modification and

additions to the SSRC Attribute specification to allow unknown SSRC values and the other information which needs to be conveyed. Further, requiring implementation of the core SSRC Attribute RFC could pose as a barrier entry and separating the two into different SDP Attributes is the better option. An implementation **SHOULD NOT** send RFC5576 SSRC Attributes alongside SRTP Context SSRC Attributes. If both are present in SDP, a receiver **SHOULD** utilize prioritize the SRTP Context Attributes over SSRC Attributes since these attributes will provide better SRTP cryptographic context initialization.

Completely Encrypting RTP Header Extensions and Contributing Sources:

SRTP Context is compatible with [RFC9335] "a=cryptex" media and session level attribute.

3.1. Generic SRTP Context Syntax

This specification introduces a new SRTP Context Attribute defined as "a=srtpctx".

The SRTP Context syntax utilizes standard attribute key-value pairs to convey data. The implementation's goal is extendable allowing for additional vendor specific key-value pairs alongside the ones defined in this document or room for future specifications to add additional key-value pairs.

The SRTP context can convey one or more key-value pair lists as per the following rules:

* Multiple key-value pairs are separated by semicolons to create a single list.
* Individual key names **MUST** be unique within a given list.
* Two or more lists of separate key-value pair groupings can be conveyed by wrapping a list in parenthesis and separating them with a comma.
* This method of parenthesis grouping **MUST NOT** be used when there is a single list of key-value pairs (with unique key names.)
* Multiple of the same key name **MAY** exist within different key-value list groupings.
* Further key-value list groupings may contain more or less keys-value pairs than other groupings.
* A given Key's value does not need to be unique within a given list or across list groupings.
* The final list member of a given single key-value list or key-value list grouping **MUST NOT** feature a trailing semicolon or comma.

The first line of Figure 1 details a single list, without parenthesis, which conveys two unique key-value pairs. The second line of Figure 1 details a list of multiple key-value pair groupings where the key named "key1" exists in two lists, keys name "key2" and "key3" are unique to their list grouping and finally a grouping only contains a single key-value pair named "key4".

Note that long lines in this document have been broken into multiple lines using the "The Single Backslash Strategy" defined by [RFC8792].

```
a=srtpctx:1 key1=value1;key2=value2
a=srtpctx:1 (key1=value1;key2=value2),\
  (key1=value1;key3=value3),(key4=value4)
```

Figure 1: Base SRTP Context Syntax

The formal definition of the SRTP Context Attribute, including generic key-value pairs is provided by the following ABNF [RFC5234] found in Figure 2.

```
srtp-context    = srtp-attr
                  srtp-tag
                  srtp-fmt-param
                  CRLF

srtp-fmt-param  = srtp-param / srtp-param-list

srtp-param      = 1srtp-ext [*(";" srtp-ext)]
                  ; One or more key=value pairs
                  ; key=value pairs separated by semicolon
                  ; e.g: key=value;key2=value2

srtp-param-list = 1"("srtp-param")" 1*("," "("srtp-param")")
                  ; Two or more lists of key=value pairs
                  ; Lists wrapped in parenthesis and
                  ; separated by commas
                  ; (key=value;key2=value2), (key=value;key3=value3)

srtp-attr       = "a=srtpctx:"

srtp-tag        = 1*9DIGIT SP
                  ; Matches tag length ABNF from RFC 4568

srtp-ext        = param-key "=" param-value
                  ; key=value

param-key       = 1*(ALPHA / DIGIT / "_" / "-")
                  ; Alphanumeric key name
                  ; May include underscore or hyphen

param-value     = 1*BYTESTRING
                  ; Byte String key value

ALPHA           = %x41-5A / %x61-7A
                  ; A-Z / a-z

DIGIT           = %x30-39
                  ; 0-9

BYTESTRING      = %x01-09 / %x0B-0C / %x0E-27 /
                  %x2A-2B / %x2D-3A / %x3C-FF
                  ; Excluding
                  ; %x00 (NULL)
                  ; %x0A (LF)
                  ; %x0D (CR)
                  ; %x28-29 (Left and Right Parenthesis)
                  ; %x2C (Comma)
                  ; %x3B (Semicolon)
```

Figure 2: ABNF of Generic of the SRTP Context Attribute

Note that Figure 2 does not allow raw left or right parenthesis, comma or semicolons within a parameter value as to avoid parsing errors with those specific delimiters. If these specific values need to be conveyed, the value **MAY** be "percent encoded" as described by the logic in [RFC3986], Section 2.1.

3.2. SSRC, ROC, SEQ Syntax

This specification specifically defines SRTP Context Attribute key-value pairs of "ssrc", "roc", and "seq". The formal definition of the "ssrc", "roc", and "seq" key-value pairs which align to "srtp-ext" of Figure 2 are detailed in this specification are defined by the ABNF of Figure 3.

```
srtp-ext  = srtp-ssrc / srtp-roc / srtp-seq
srtp-ssrc = "ssrc=" "0x"1*8HEXDIG
            ; 32 bit SSRC
srtp-roc  = "roc=" "0x"1*8HEXDIG
            ; 32 bit ROC
srtp-seq  = "seq=" "0x"1*4HEXDIG
            ; 16 bit Sequence
HEXDIG    = %x30-39 / %x41-46
            ; 0-9 / A-F
```

Figure 3: ABNF of Specific Syntax

For "ssrc", "roc", and "seq", leading 0s may be omitted and the alphanumeric hex may be upper or lowercase. Thus as per Figure 4 these three lines are functionally identical.

```
a=srtpctx:1 ssrc=0x00845FED;roc=0x00000000;seq=0x005D
a=srtpctx:1 ssrc=0x845fed;roc=0x0;seq=0x05d
a=srtpctx:1 ssrc=0x845feD;roc=0x0;seq=0x5D
```

Figure 4: Comparison with and without Leading 0s

In Figure 5 the sender has shares the usual cryptographic information as per a=crypto but has included other information such as the 32 bit SSRC, 32 bit ROC, and 16 bit Last Known Sequence number as hex values within the a=srtpctx attribute. Together these attributes provide better insights as to the state of the SRTP cryptographic context from the senders perspective.

```
a=crypto:1 AEAD_AES_256_GCM \
  inline:3/sxOxrbg3CVDrxeaNs91Vle+wW1RvT/zJWTCUNP1i6L45S9qcstjBv+eo0=\
  |2^20|1:32
a=srtpctx:1 ssrc=0x00845FED;roc=0x0000;seq=0x0150
```

Figure 5: Example SRTP Context Attribute

3.3. Pairing SRTP Context Attributes to SDP Security Attributes

When SRTP context information needs to be conveyed about a given stream, the SRTP Context attribute (a=srtpctx) is coupled with the relevant SDP Security attribute (a=crypto) in the SDP. This coupling is done via the "tag" found in both SDP attributes. The tag used by SRTP Context Attributes is functionally the same as detailed in [RFC4568], Section 4.1. The tag advertised in the SRTP Context Attribute is used to identify the SDP Security parameter a given SRTP Context Attribute is meant to pair with. As such, within given media stream (m=), the tag of the SRTP Context Attribute **MUST** exactly match the SDP Security parameters tag as to create a pair of cryptographic attributes.

The example in shown in Figure 6, within the audio stream, the sender is advertising an explicit packet index mapping for a=crypto tag 2 (a=srtpctx:2) which matches the SDP security parameter with the same tag (a=crypto:2) Within the audio and video media stream tag 1 (a=crypto:1) does not feature any paired SRTP Context Attributes.

```
c=IN IP4 192.0.0.1
m=audio 49170 RTP/SAVP 0
a=crypto:1 AES_CM_128_HMAC_SHA1_80 \
  inline:d0RmdmcmVCspeEc3QGZiNWpVLFJhQX1cfHAwJSoj|2^20|1:32
a=crypto:2 AEAD_AES_256_GCM \
  inline:HGAPy4Cedy/qumbZvpuCZSVT7rNDk8vG4TdUXp5hkyWqJCqiLRGab0KJy1g=
a=srtpctx:2 ssrc=0xBFBDD;roc=0x0001;seq=0x3039
m=video 49172 RTP/SAVP 126
a=crypto:1 AEAD_AES_128_GCM \
  inline:bQJXGzEPXJPClrd78xwALdaZDs/dLttBLfLE5Q==
a=srtpctx:1 ssrc=0xDD147C14;roc=0x0001;seq=0x3039
```

Figure 6: Example crypto and SRTP Context tag mapping

It is unlikely a sender will send SRTP Context Attributes for every crypto attribute since many will be fully unknown (such as the start of a session.) However it is theoretically possible for every a=crypto tag to have a similar a=srtpctx attribute for additional details.

3.4. Handling Unknown Cryptographic Attributes

Applications **MUST NOT** include SRTP Context Attributes if all the values are unknown; such as the start of a session or later in a session where full SRTP context is lost by an application. These unknown SRTP Context Attributes **MAY** be signaled at any later time

but applications **SHOULD** ensure any offer/answer has the appropriate SRTP Context Attributes.

Further, if an SRTP Context Attribute key-value pair is advertised at one point during a session and then later removed during a session modification; the peer receiving application **SHOULD** fallback to default application logic, or locally derived/stored cryptographic context information, rather than failing/rejecting the session.

For "ssrc", "roc", and "seq" the following are quick pointers to the default application logic that can be used when locally derived/ stored cryptographic context information is not available and an SRTP Context Attribute was omitted or removed during session modification.

Rollover Counter (ROC):
If at the start of a session set the ROC to zero. If later in a session, solve for "v" as per [RFC3711], Section 3.3.1

Synchronization Source (SSRC)
Via "Late Binding" defined by [RFC4568], Section 6.4.1

Sequence (SEQ):
Via "Late Binding" defined by [RFC4568], Section 6.4.1

3.5. SRTP Multiplexing

For scenarios where SRTP Multiplexing are concerned, EKT-SRTP ([RFC8870]) **SHOULD** be used in lieu of SDP Security as per [RFC8872] Section 4.3.2. If SRTP Context Attributes are to be used, multiple SRTP Context Attribute key-value pairs can be grouped in a different lists using parenthesis as a delimiter with a comma to separate multiple key-value list groupings. The default syntax for key-value list groupings detailed further in Section 3.1.

The key-value list groupings for "ssrc, "roc" and "seq" can be observed in Figure 7 where three SSRC and the respective ROC/SEQ are provided as a key-value list groupings within the a=srtpctx attribute:

```
a=crypto:1 AES_CM_128_HMAC_SHA1_80 \
inline:d0RmdmcmVCspeEc3QGZiNWpVLFJhQX1cfHAwJSo
a=srtpctx:1 (ssrc=0x01;roc=0x0;seq=0x1234),\
  (ssrc=0x02;roc=0x1;seq=0xABCD),\
  (ssrc=0x845fed;roc=0x0000)
```

Figure 7: Example SRTP Context with Multiple SSRC

3.6. SDP Bundling

For scenarios where SDP Bundling are concerned, SRTP Context Attributes follow the same bundling guidelines defined by [RFC8859], section 5.7 for SDP Securities a=crypto attribute.

3.7. SDP Considerations

The presence of the "a=srtpctx" attribute in the SDP (in either an offer or an answer) indicates that the endpoint is signaling explicit cryptographic context information and this data **SHOULD** be used in place of derived values (see Section 3.4).

3.7.1. Sender Behavior

Senders utilizing SDP Security via "a=crypto" **MUST** make an attempt to signal any known packet index values to the peer receiver. The exception being when all values are unknown, such as at the very start of media stream negotiation.

For best results all sending parties of a given session stream **SHOULD** advertise known packet index values for all media streams. This should continue throughout the life of the session to ensure any errors or out of sync errors can be quickly corrected via new signaling methods. See Section 3.7.3 for update frequency recommendations.

3.7.2. Receiver Behavior

Receivers **SHOULD** utilize the signaled information in application logic to instantiate the SRTP cryptographic context.

In the even there is no SRTP Context Attributes present in SDP receivers **SHOULD** fall back to application defaults as outlined in Section 3.4.

See Section 3.4 for handling scenarios where a value was advertised and has been removed during session modification.

3.7.3. Update Frequency

Senders **SHOULD** provide SRTP Context SDP when SDP Crypto attributes are negotiated. There is no explicit time or total number of packets in which a new update is required from sender to receiver. This specification will not cause overcrowding on the session establishment protocol's signaling channel if natural session updates, session changes, and session liveliness checks are followed.

3.8. Future Extendability

As stated in Section 3.1, the SRTP Context SDP implementation's goal is extendability allowing for additional vendor specific key-value pairs alongside the ones defined in this document. This ensures that a=crypto SDP security may remain compatible with future algorithms that need to signal cryptographic context information outside of what is currently specified in [RFC4568].

A complying specification needs only to follow the general rules defined by Section 3.1 and the generic ABNF outlined in Figure 2.

To illustrate, imagine a new example SRTP algorithm and crypto suite is created named "FOO_CHACHA20_POLY1305_SHA256" and the application needs to signal "Foo, "Bar", and "Nonce" values to properly instantiate the SRTP context. Rather than modify a=crypto SDP security or create a new unique SDP attribute, one can simply utilize SRTP Context SDP's key-value pairs to convey the information. Implementations **MUST** define how to handle default scenarios where the value is not present, unknown, or is removed later in a session.

```
a=crypto:1 FOO_CHACHA20_POLY1305_SHA256 \
 inline:1ef9a49f1f68f75f95feca6898921db8c73bfa53e71e33726c4c983069dd7d44
a=srtpctx:1 foo=1;bar=abc123;nonce=8675309
```

With this extendable method, all that is now required in the fictional RFC defining "FOO_CHACHA20_POLY1305_SHA256" is to include a section which details the expected SRTP Context Attribute key-value pair syntax, offer/answer usage (including unknown values and later session modifications). Don't forget to detail other aspects of importance such as usage with SDP bundling, SRTP multiplexing and comparability with SRTP extensions defined in Section 3.

This approach is similar to how Media Format Parameter Capability ("a=fmtp") is utilized in modern SDP. An example is [RFC6184], Section 8.2.1 for H.264 video Media Format Parameters.

4. Security Considerations

When SDP carries SRTP Context Attributes additional insights are present about the SRTP cryptographic context. Due to this an intermediary **MAY** be able to analyze how long a session has been active by the ROC value.

Since the SRTP Context Attribute is carried in plain-text (alongside existing values like the SRTP Master Key for a given session) care **MUST** be taken as per the [RFC8866] that keying material must not be sent over unsecure channels unless the SDP can be both private (encrypted) and authenticated.

5. IANA Considerations

This document updates the "attribute-name (formerly "att-field")" sub-registry of the "Session Description Protocol (SDP) Parameters" registry (see Section 8.2.4 of [RFC8866]). Specifically, it adds the SDP "a=srtpctx" attribute for use at the media level.

Form	Value
Contact name	IETF
Contact email address	kydavis@cisco.com
Attribute name	srtpctx
Attribute value	srtpctx
Attribute syntax	Provided by ABNF found in Section 3.1
Attribute semantics	Provided by sub-sections of Section 3
Usage level	media
Charset dependent	No
Purpose	Provide additional insights about SRTP context information not conveyed required by a receiver to properly decrypt SRTP.
O/A procedures	SDP O/A procedures are described in Section 3.1, specifically sections Section 3.7.1 and Section 3.7.2 of this document.
Mux Category	TRANSPORT

Table 1: IANA SDP Registration Form

6. Acknowledgements

Thanks to Paul Jones for reviewing early draft material and providing valuable feedback.

7. References

7.1. 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, <https://www.rfc-editor.org/rfc/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, <https://www.rfc-editor.org/rfc/rfc3711>.

[RFC4568] Andreasen, F., Baugher, M., and D. Wing, "Session Description Protocol (SDP) Security Descriptions for Media Streams", RFC 4568, DOI 10.17487/RFC4568, July 2006, <https://www.rfc-editor.org/rfc/rfc4568>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/rfc/rfc8174>.

[RFC8859] Nandakumar, S., "A Framework for Session Description Protocol (SDP) Attributes When Multiplexing", RFC 8859, DOI 10.17487/RFC8859, January 2021, <https://www.rfc-editor.org/rfc/rfc8859>.

[RFC8866] Begen, A., Kyzivat, P., Perkins, C., and M. Handley, "SDP: Session Description Protocol", RFC 8866, DOI 10.17487/RFC8866, January 2021, <https://www.rfc-editor.org/rfc/rfc8866>.

7.2. Informative References

[RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, DOI 10.17487/RFC3550, July 2003, <https://www.rfc-editor.org/rfc/rfc3550>.

[RFC3830] Arkko, J., Carrara, E., Lindholm, F., Naslund, M., and K. Norrman, "MIKEY: Multimedia Internet KEYing", RFC 3830, DOI 10.17487/RFC3830, August 2004, <https://www.rfc-editor.org/rfc/rfc3830>.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <https://www.rfc-editor.org/rfc/rfc3986>.

[RFC4567] Arkko, J., Lindholm, F., Naslund, M., Norrman, K., and E. Carrara, "Key Management Extensions for Session Description Protocol (SDP) and Real Time Streaming Protocol (RTSP)", RFC 4567, DOI 10.17487/RFC4567, July 2006, <https://www.rfc-editor.org/rfc/rfc4567>.

[RFC4771] Lehtovirta, V., Naslund, M., and K. Norrman, "Integrity Transform Carrying Roll-Over Counter for the Secure Real-time Transport Protocol (SRTP)", RFC 4771, DOI 10.17487/RFC4771, January 2007, <https://www.rfc-editor.org/rfc/rfc4771>.

[RFC5159] Dondeti, L., Ed. and A. Jerichow, "Session Description Protocol (SDP) Attributes for Open Mobile Alliance (OMA) Broadcast (BCAST) Service and Content Protection", RFC 5159, DOI 10.17487/RFC5159, March 2008, <https://www.rfc-editor.org/rfc/rfc5159>.

[RFC5234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, DOI 10.17487/RFC5234, January 2008, <https://www.rfc-editor.org/rfc/rfc5234>.

[RFC5479] Wing, D., Ed., Fries, S., Tschofenig, H., and F. Audet, "Requirements and Analysis of Media Security Management Protocols", RFC 5479, DOI 10.17487/RFC5479, April 2009, <https://www.rfc-editor.org/rfc/rfc5479>.

[RFC5576] Lennox, J., Ott, J., and T. Schierl, "Source-Specific Media Attributes in the Session Description Protocol (SDP)", RFC 5576, DOI 10.17487/RFC5576, June 2009, <https://www.rfc-editor.org/rfc/rfc5576>.

[RFC5763] Fischl, J., Tschofenig, H., and E. Rescorla, "Framework for Establishing a Secure Real-time Transport Protocol (SRTP) Security Context Using Datagram Transport Layer Security (DTLS)", RFC 5763, DOI 10.17487/RFC5763, May 2010, <https://www.rfc-editor.org/rfc/rfc5763>.

[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, <https://www.rfc-editor.org/rfc/rfc5764>.

[RFC6184] Wang, Y.-K., Even, R., Kristensen, T., and R. Jesup, "RTP Payload Format for H.264 Video", RFC 6184, DOI 10.17487/RFC6184, May 2011, <https://www.rfc-editor.org/rfc/rfc6184>.

[RFC6189] Zimmermann, P., Johnston, A., Ed., and J. Callas, "ZRTP: Media Path Key Agreement for Unicast Secure RTP", RFC 6189, DOI 10.17487/RFC6189, April 2011, <https://www.rfc-editor.org/rfc/rfc6189>.

[RFC6904] Lennox, J., "Encryption of Header Extensions in the Secure Real-time Transport Protocol (SRTP)", RFC 6904,

DOI 10.17487/RFC6904, April 2013, <https://www.rfc-editor.org/rfc/rfc6904>.

[RFC7201] Westerlund, M. and C. Perkins, "Options for Securing RTP Sessions", RFC 7201, DOI 10.17487/RFC7201, April 2014, <https://www.rfc-editor.org/rfc/rfc7201>.

[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, <https://www.rfc-editor.org/rfc/rfc7714>.

[RFC8723] Jennings, C., Jones, P., Barnes, R., and A.B. Roach, "Double Encryption Procedures for the Secure Real-Time Transport Protocol (SRTP)", RFC 8723, DOI 10.17487/RFC8723, April 2020, <https://www.rfc-editor.org/rfc/rfc8723>.

[RFC8792] Watsen, K., Auerswald, E., Farrel, A., and Q. Wu, "Handling Long Lines in Content of Internet-Drafts and RFCs", RFC 8792, DOI 10.17487/RFC8792, June 2020, <https://www.rfc-editor.org/rfc/rfc8792>.

[RFC8870] Jennings, C., Mattsson, J., McGrew, D., Wing, D., and F. Andreasen, "Encrypted Key Transport for DTLS and Secure RTP", RFC 8870, DOI 10.17487/RFC8870, January 2021, <https://www.rfc-editor.org/rfc/rfc8870>.

[RFC8871] Jones, P., Benham, D., and C. Groves, "A Solution Framework for Private Media in Privacy-Enhanced RTP Conferencing (PERC)", RFC 8871, DOI 10.17487/RFC8871, January 2021, <https://www.rfc-editor.org/rfc/rfc8871>.

[RFC8872] Westerlund, M., Burman, B., Perkins, C., Alvestrand, H., and R. Even, "Guidelines for Using the Multiplexing Features of RTP to Support Multiple Media Streams", RFC 8872, DOI 10.17487/RFC8872, January 2021, <https://www.rfc-editor.org/rfc/rfc8872>.

[RFC9335] Uberti, J., Jennings, C., and S. Murillo, "Completely Encrypting RTP Header Extensions and Contributing Sources", RFC 9335, DOI 10.17487/RFC9335, January 2023, <https://www.rfc-editor.org/rfc/rfc9335>.

Appendix A. Protocol Design Overview

This appendix section is included to details some important itmes integral to the decision process of creating this specification. This section may be removed by the editors or left for future generations to understand why specific things were done as they are.

In general, the overall design for this protocol tends to follow the phrase found in RFC6709, Section 1. "Experience with many protocols has shown that protocols with few options tend towards ubiquity, whereas protocols with many options tend towards obscurity.

Each and every extension, regardless of its benefits, must be carefully scrutinized with respect to its implementation, deployment, and interoperability costs."

A.1. Why not an RTP Header Extension?

In order to be compatible with "a=cryptex", a protocol which extends the SRTP encryption over the RTP Extension Headers, the designed specification must ensure that information about the SRTP context is not within these RTP extension headers. Thus one has to provide this information in an out of band mechanism.

A.2. Why not an SDP Security Session Parameter?

While analyzing SDP Security's Session Parameter feature number of interesting details were found. That is sections 6.3.7, 7.1.1, 9.2, and 10.3.2.2 of [RFC4568] specifically.

A few illustrative examples below detail what this could look like are provided below, though these **MUST NOT** be used.

```
a=crypto:1 [..omitted..] SSRC=0x00845FED ROC=0x00000000 SEQ=0x005D

a=crypto:1 ..omitted.. -SSRC=0x00845FED -ROC=0x00000000 -SEQ=0x005D

a=crypto:1 AEAD_AES_256_GCM \
 inline:3/sxOxrbg3CVDrxeaNs91Vle+wW1RvT/zJWTCUNP1i6L45S9qcstjBv+eo0=\
 |2^20|1:32 SSRC=0x00845FED ROC=0x0000 SEQ=0x0150

a=crypto:1 AES_CM_128_HMAC_SHA1_80 \
  inline:QUJjZGVmMTIzNDU2Nzg5QUJDREUwMTIzNDU2Nzg5|2:18\
  ;inline:QUJjZGVmMTIzNDU2Nzg5QUJDREUwMTIzNDU2Nzg5|21|3:4 \
  KDR=23 FEC_ORDER=SRTP_FEC UNENCRYPTED_SRTP \
  SSRC=0xDD148F16 ROC=0x0 SEQ=0x5A53
a=crypto:2 AES_CM_128_HMAC_SHA1_32 \
  inline:QUJjZGVmMTIzNDU2Nzg5QUJDREUwMTIzNDU2Nzg5|2^20 \
  FEC_KEY=inline:QUJjZGVmMTIzNDU2Nzg5QUJDREUwMTIzNDU2Nzg5|2^20|2:4 \
  WSH=60 SSRC=0xD903 ROC=0x0002 SEQ=0xB043
a=crypto:3 AEAD_AES_256_GCM \
  inline:HGAPy4Cedy/qumbZvpuCZSVT7rNDk8vG4TdUXp5hkyWqJCqiLRGab0KJy1g= \
  UNAUTHENTICATED_SRTP SSRC=0x05 ROC=0x02 SEQ=unknown
a=crypto:4 AEAD_AES_128_GCM \
  inline:bQJXGzEPXJPClrd78xwALdaZDs/dLttBLfLE5Q== \
  UNENCRYPTED_SRTCP SSRC=0x6500
```

To analyze the faults of this method: First, a unknown and/or unsupported SDP Security Session Parameter is destructive. If one side where to advertise the ROC value as an SDP Security Session Parameter and the remote party does not understand that specific SDP Security Session Parameter, that entire crypto line is to be considered invalid. If this is the only a=crypto entry then the entire session may fail. The solution in this document allows for a graceful fallback to known methods to determine these value. Implementations could get around this by duplicating the a=crypto SDP attribute into two values: one with the postfix and one without to create to potential offers; but at this point we have a second SDP attribute. Instead this specification decided to cut to the chase and format the second attribute in a standardized way and avoid endless duplication (and potentially other harmful issues, see the final item in this document.)

Second, there is a method to advertise "optional" SDP Security Session Parameters. However, upon further scrutiny, the document contradicts itself in many sections. To be specific, Section 6.3.7 states that an SDP Security Session Parameter prefixed with a dash character "-" **MAY** be ignored. Subsequent sections (9.2 and 10.3.2.2) state that a dash character is illegal and **MUST NOT** be used. It is not very well defined as such pursuit of this method has been dropped.

Further, we know how applications will handle unknown SDP attributes; we do not know how applications will handle new mandatory (or optional) SDP Security Session Parameter values as none have ever been created. See IANA registry which only details those from the original RFC. (https://www.iana.org/assignments/sdp-security-descriptions/sdp-security-descriptions.xhtml#sdp-security-descriptions-4) Including these could cause larger application issues and are the reason modern protocols use logic like Generate Random Extensions And Sustain Extensibility (GREASE) to catch bad implementation behavior and correct it before it leads to problems like those described in this section.

In closing, this method has too many challenges but a lot has been learned. These items have influenced the protocol design and sections like Section 3.8 which aim to avoid making the same mistakes.

Authors' Addresses

Kyzer R. Davis
Cisco Systems

Email: kydavis@cisco.com

Esteban Valverde
Cisco Systems

Email: jovalver@cisco.com

Gonzalo Salgueiro
Cisco Systems

Email: gsalguei@cisco.com