A Solution Framework for Private Media in a Switched Conferencing Environment

(draft-jones-avtcore-private-media-framework-01)

IETF 92 / March 2015

Paul E. Jones • Nermeen Ismail • David Benham

Cisco

Security Objectives – Hop-by-Hop

- Use SRTP procedures and key shared between only two adjacent entities to perform all hop-by-hop operations:
  - Authentication of the RTP and RTCP packets
  - Optional hop-by-hop encryption of RTP header extensions
  - Optional hop-by-hop encryption of RTCP packets

Security Objectives – End-to-End

- Use an EKT key known to all conference participants to facilitate end-to-end authenticated encryption of media content (i.e., audio & video)
- Why EKT?
  - We want a single shared key known to all conference participants
  - We want to ensure quick cryptographic context synchronization (avoid ROC-guessing) when a media flow is switched to a receiver and not seen by the receiver for a long time (e.g., a participant was silent) or due to a late-joining participant and where rekeying is not performed
  - EKT allows us to avoid re-keying the conference when an SSRC collision occurs, thus addressing the "two time pad" issue described in Section 9.1/RFC 3711. With EKT, every sender will have a distinct SRTP master key for end-to-end encryption

What Entity Has Access to What Key Material?

	Endpoint A	Switch A	Switch B	Endpoint B
End-to-End EKT Key and Salt	Yes	No	No	Yes
Hop-by-Hop Key (A⇔ Switch A)	Yes	Yes	No	No
Hop-by-Hop Key (Switch A ⇔ Switch B)	No	Yes	Yes	No
Hop-by-Hop Key (B⇔ Switch B)	No	No	Yes	Yes

An "endpoint" might be an end-user terminal device, gateway, or other entity that is trusted

A View into the Private Media RTP Packet

- The actual media content ("payload") is encrypted using an authenticated encryption mechanism like AES-GCM
- A modified version of EKT is used to convey the SRTP master key used for authenticated encryption
- The key used to generate the authentication tag, encrypt the RTP header extension, and RTCP-related security operations is derived from a separate SRTP master key unrelated to the EKT key

```
   0                   1                   2                   3
   0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+<+
  |V=2|P|X|  CC   |M|     PT      |       sequence number         | |
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |
  |                           timestamp                           | |
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |
  |           synchronization source (SSRC) identifier            | |
  +=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+ |
  |            contributing source (CSRC) identifiers             | |
  |                               ....                            | |
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |
  |                   RTP extension (OPTIONAL*)                   | |
+>+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |
| |                          payload  ...                         | |
| |                               +-------------------------------+ |
| |                               | RTP padding   | RTP pad count | |
+>+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+<+
| |                           SRTP ROC                            | |
| +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |
| |                         EKT ciphertext ...                    | |
| |                               +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |
| |                               |  Security Parameter Index   |1| |
| +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+<+
| :                     authentication tag (MANDATORY)            : |
| +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |
|                                                                   |
+-- Authenticated Encryption                Authenticated Portion --+
    End-to-End                              using Hop-by-Hop Key

    * Header extensions are optionally Encrypted Hop-by-Hop
```

Cryptographic Context & RTP Header Considerations

- Framework document adds an end-to-end key, salt, and AEAD cipher suite to the SRTP cryptographic context at transmitter, all must be conveyed to receiver
- As the SRTP cryptographic context makes use of SSRC, sequence number, and ROC, those must be known by the receiver in order to decrypt

Alternatives to convey cryptographic context to receiver:

a) Forward with those values in header unchanged, per current draft
b) If re-writing those values in header, copy the original values to elsewhere in the packet (e.g., header extension)
c) Replace SSRC, sequence number, and ROC with functionally equivalent values that are carried in the SRTP payload

… form design team to discuss

Discussion

- The draft focuses only on media security, thus does not address signaling security that might be necessary to allow the KMF to trust the endpoint (e.g., exchange of public key, certificate, or fingerprint).
- We are proposing several changes to EKT
    - ROC is transmitted as plaintext in the EKT Tag
    - We need a mechanism to negotiate SRTP Protection Profiles for the end-to-end encryption/authentication (DTLS-SRTP is proposed for hop-by-hop, but we need something for end-to-end and adding something to the EKT message exchanges might be an approach.)
    - How much flexibility do we have here?

Discussion (continued)

- Anything explicitly in or out of scope?
  - New RTP topologies
  - EKT and modifications to it
- Congestion control
  - A design team is exploring this area; the intent is to refer to that work