Extension Encryption

Encrypting All The Bits

Justin Uberti, Google
Sergio Garcia Murillo, Cosmo
IETF 108

Problem Statement

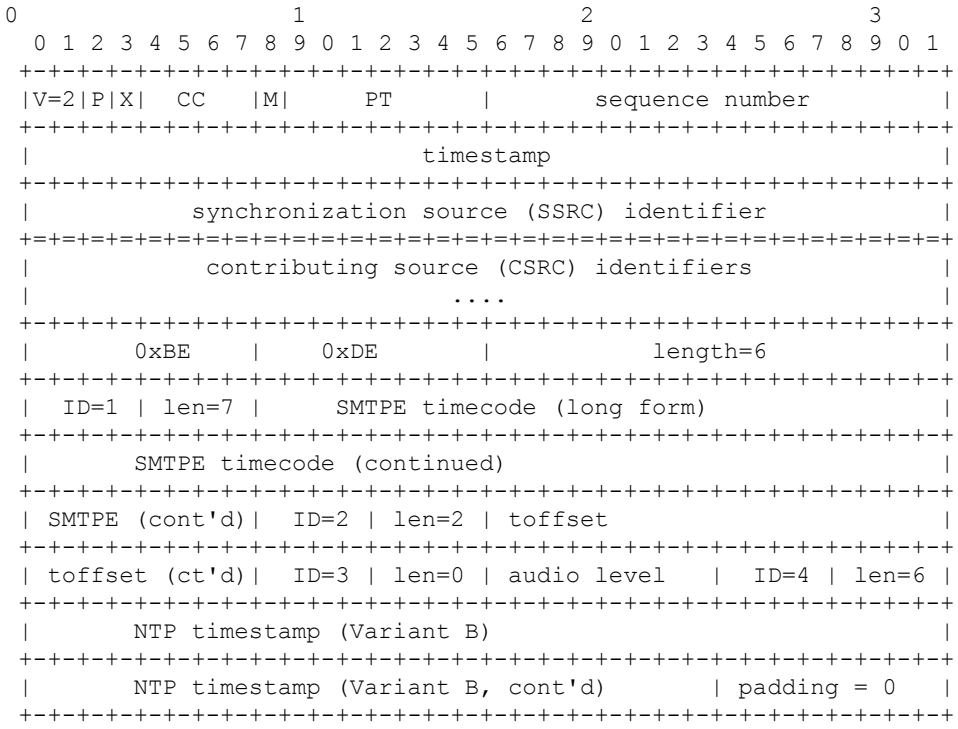
- WebRTC implementations use a lot of extensions (e.g., Safari)

```
a=extmap:1 urn:ietf:params:rtp-hdrext:ssrc-audio-level
a=extmap:14 urn:ietf:params:rtp-hdrext:toffset
a=extmap:2 http://www.webrtc.org/experiments/rtp-hdrext/abs-send-time
a=extmap:13 urn:3gpp:video-orientation
a=extmap:3 http://www.ietf.org/id/draft-holmer-rmcat-transport-wide-cc-extensions-01
a=extmap:12 http://www.webrtc.org/experiments/rtp-hdrext/playout-delay
a=extmap:11 http://www.webrtc.org/experiments/rtp-hdrext/video-content-type
a=extmap:7 http://www.webrtc.org/experiments/rtp-hdrext/video-timing
a=extmap:8 http://tools.ietf.org/html/draft-ietf-avtext-framemarking-07
a=extmap:9 http://www.webrtc.org/experiments/rtp-hdrext/color-space
a=extmap:4 urn:ietf:params:rtp-hdrext:sdes:mid
a=extmap:5 urn:ietf:params:rtp-hdrext:sdes:rtp-stream-id
a=extmap:6 urn:ietf:params:rtp-hdrext:sdes:repaired-rtp-stream-id
```

- Some of these are at least somewhat sensitive
  - e.g., `ssrc-audio-level, video-content-type`
- All of these leak some amount of metadata
  - e.g., application type, HDR support, HW/SW encoder

Current

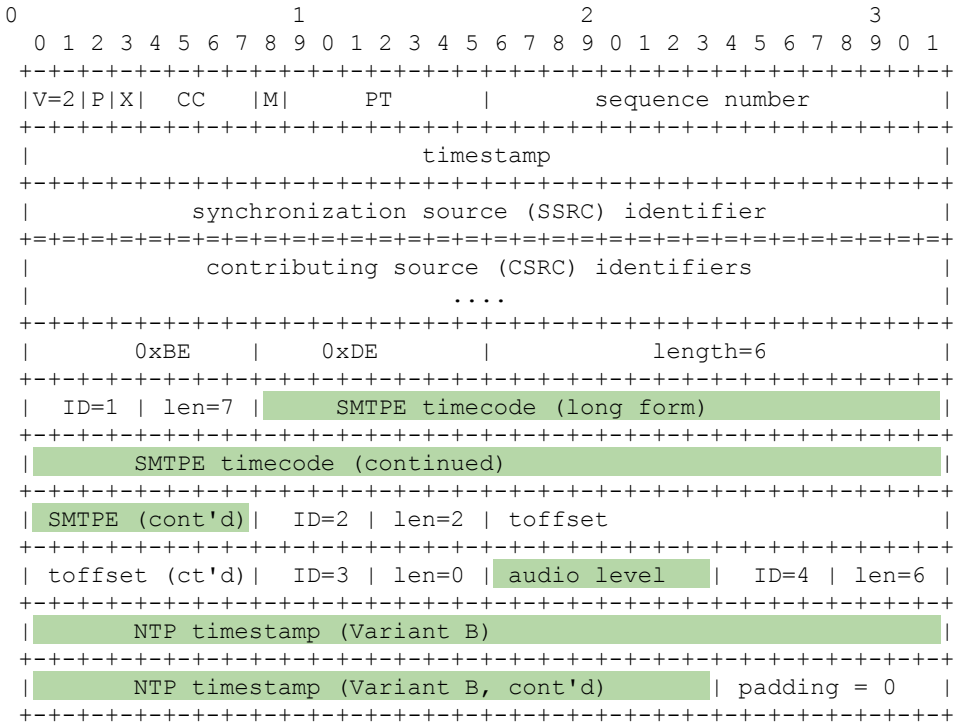
None of the RTP header is encrypted, including extensions



Solution

- To prevent metadata leaks, we should encrypt RTP header extensions, and make this MTI for WebRTC implementations

Option A: RFC 6904



Challenges

- RFC 6904 defines a mechanism to encrypt header extension values, but is complicated
  - need to offer extensions in both encrypted/unencrypted forms
  - need to negotiate encryption individually for each extension
  - SDP bloat (every extmap now requires 2 lines)
  - if you have 7+ extensions, requires RFC 8285's a=extmap-allow-mixed, an additional piece of negotiation

Challenges (2)

- This complexity has hindered adoption of RFC 6904
  - Client and SFU implementors have felt the perceived value doesn't justify the complexity
  - libsrtp had a serious bug that was not found for 5+ years
  - libwebrtc had a serious bug that was not found for multiple years
  - General desire for a mechanism that can be mostly handled by libsrtp with minimal negotiation

Challenges (3)

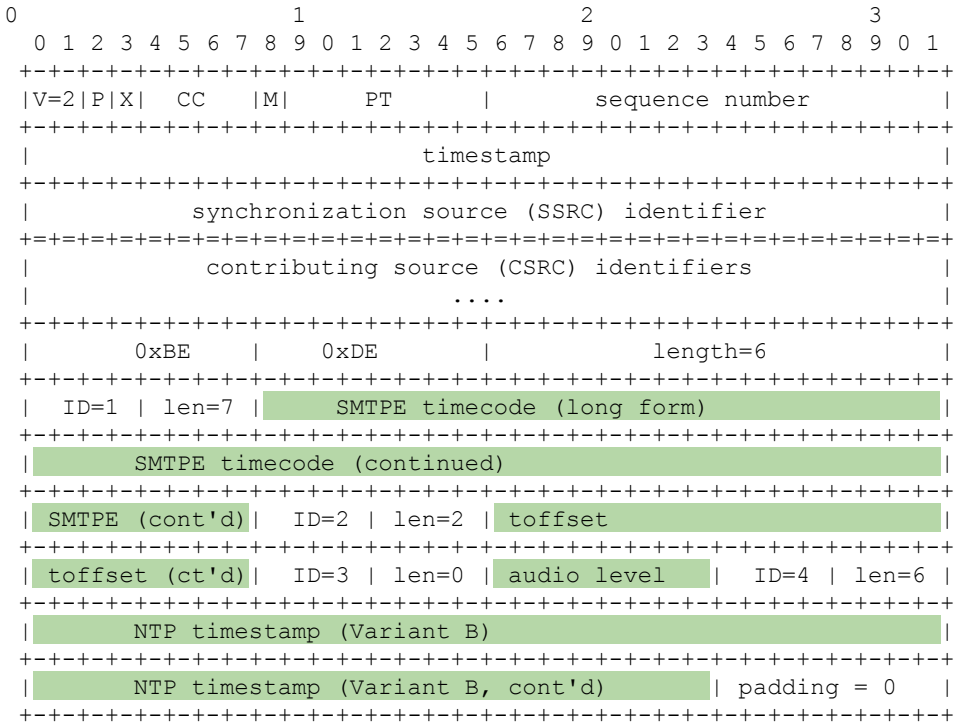
- RFC 6904 also has technical deficiencies
  - easy bid-down against encryption negotiation
  - header IDs still sent as cleartext (metadata leakage)
  - when using allow-mixed, encrypted extensions will need to use a 2-byte exthdr (since we need to use 1-byte exthdrs for the unencrypted forms, for backcompat), causing RTP packet bloat

Opportunity

- Is there any real need for encryption of a subset of headers?
- SFRAME assumes two encryption domains:
  - Client-Server: DTLS/SRTP
  - End-to-End: SFRAME
- In this world, middlebox can still see extensions, even if encrypted
- If we assume header encryption is all-or-none, what options do we have?

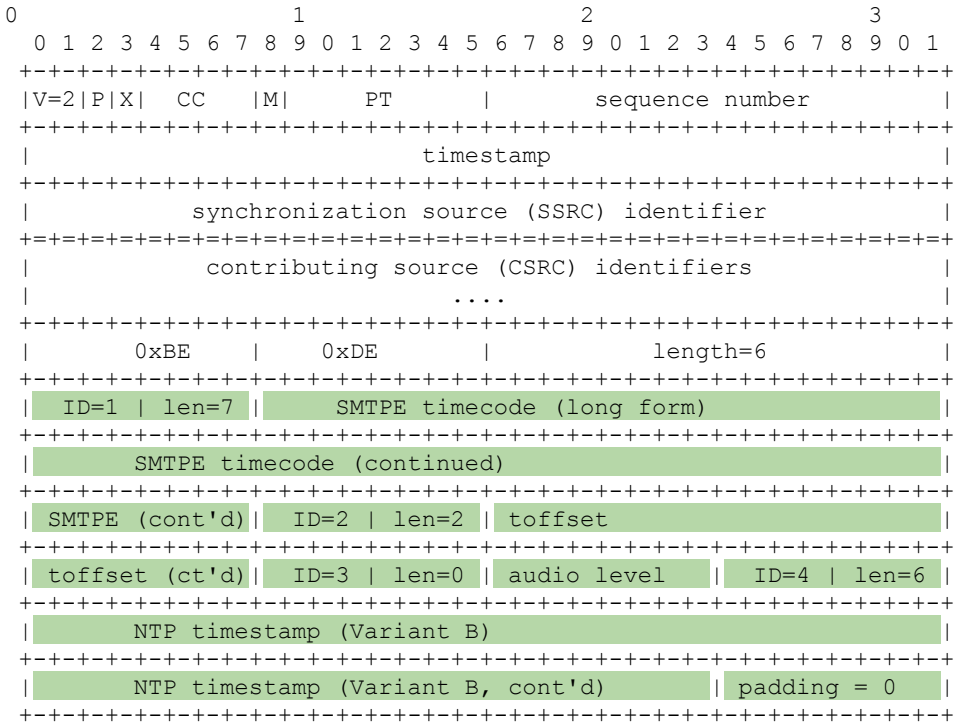
Option B1: Encrypt All Extension Values

Like 6904, but only negotiate a single bit



Option B2: Encrypt All Extensions

Encrypt entire extension block (including IDs)

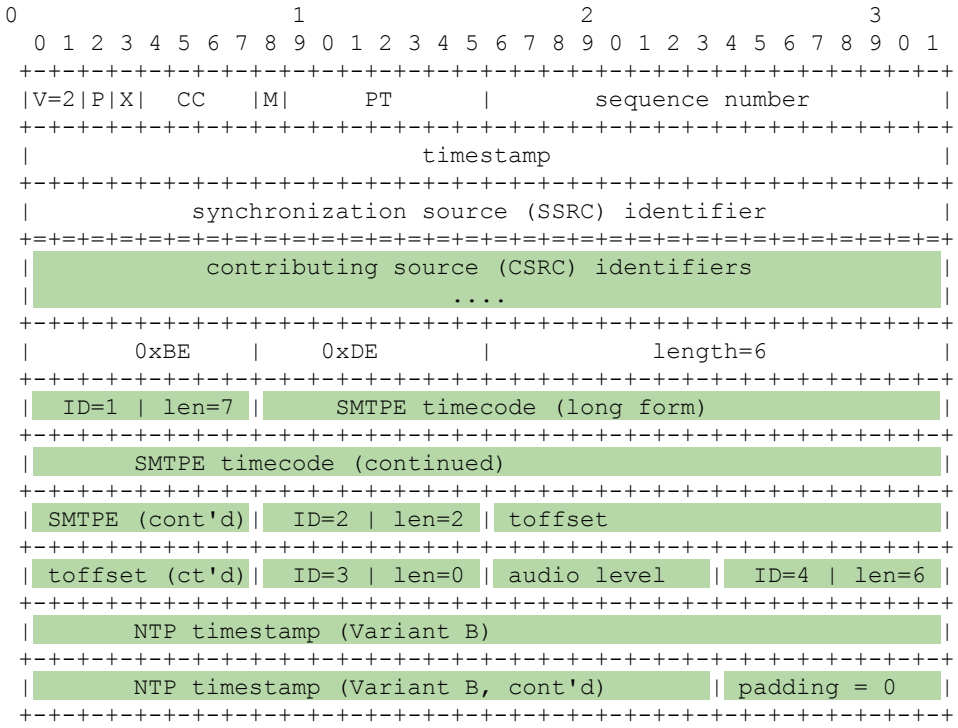


Opportunity (2)

- Even if we encrypt all extensions, there is still unencrypted metadata in the packet
  - CSRCs (who is speaking in a call)
  - Marker bit (when speech is occurring)
  - PT (what codecs are used)
- If we're just telling libsrtp to encrypt part of the header, what else could we encrypt?

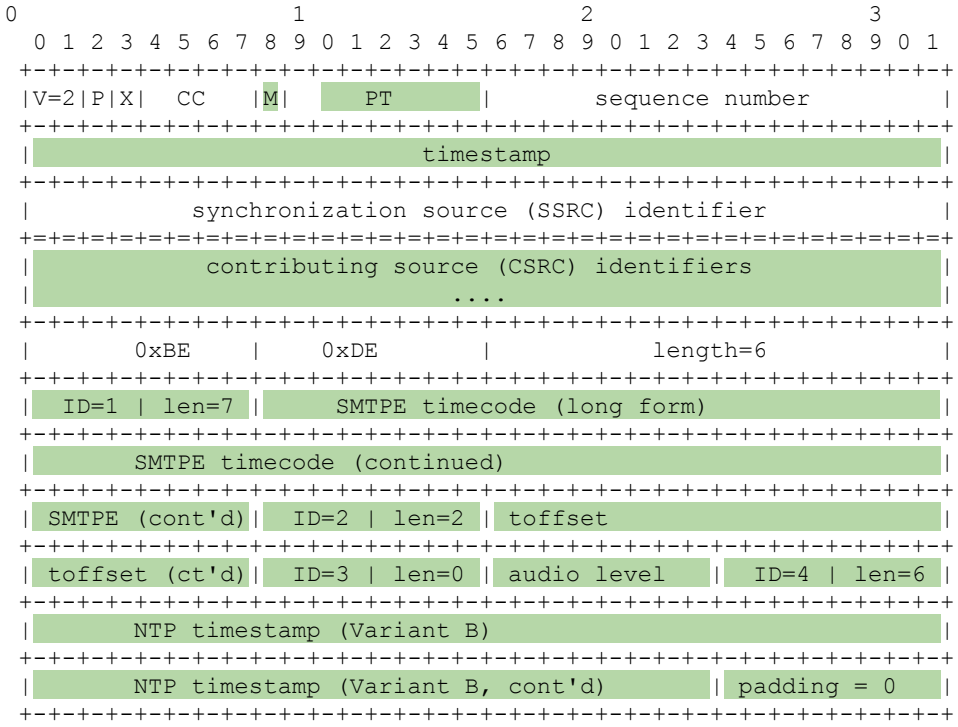
Option C1: Encrypt Extensions + CSRCs

Encrypt CSRCs + extension block



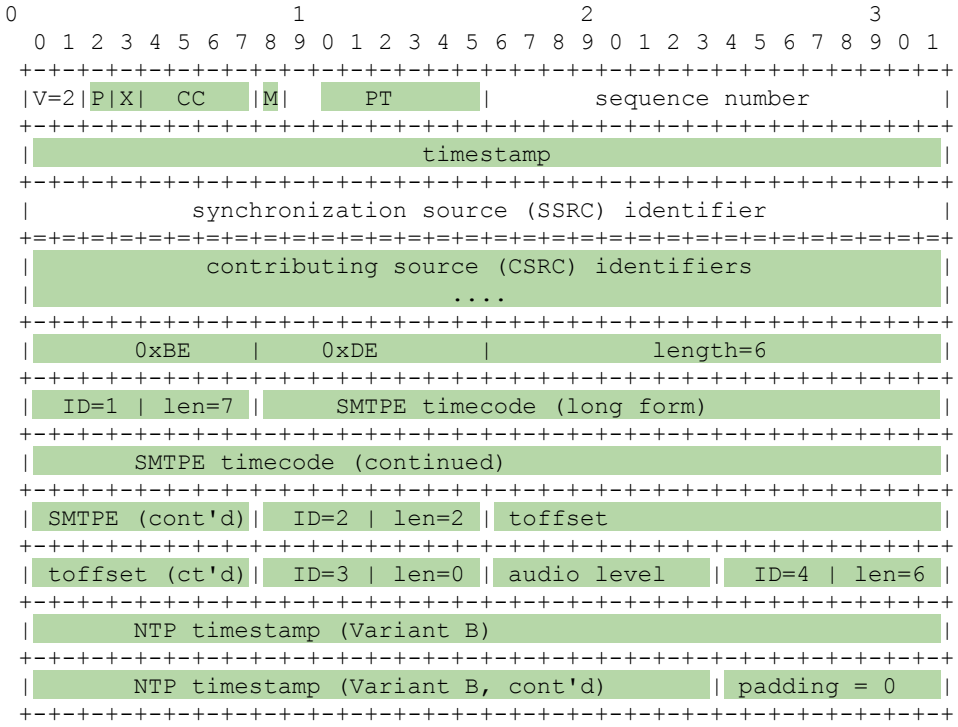
Option C2: Encrypt Everything Compatible

PT and timestamp not needed for keying/parsing (high bit needed for RTCP demux)



Option C3: Encrypt Everything Possible

Additional fields can be encrypted if non-encrypted parsing is not needed
V, PT high bit needed for demux, seqnum/SSRC for keying



Questions

- Is there interest in pursuing header encryption as MTI?
- Is there interest in a different mechanism than 6904?
- Is there interest in going beyond just header values?
- Negotiate capability via SDP or DTLS?
  - DTLS prevents bid-down, but more work
- What APIs do we need?
  - Suggest single RTCConfiguration API, 'negotiate' or 'require'

Thank You