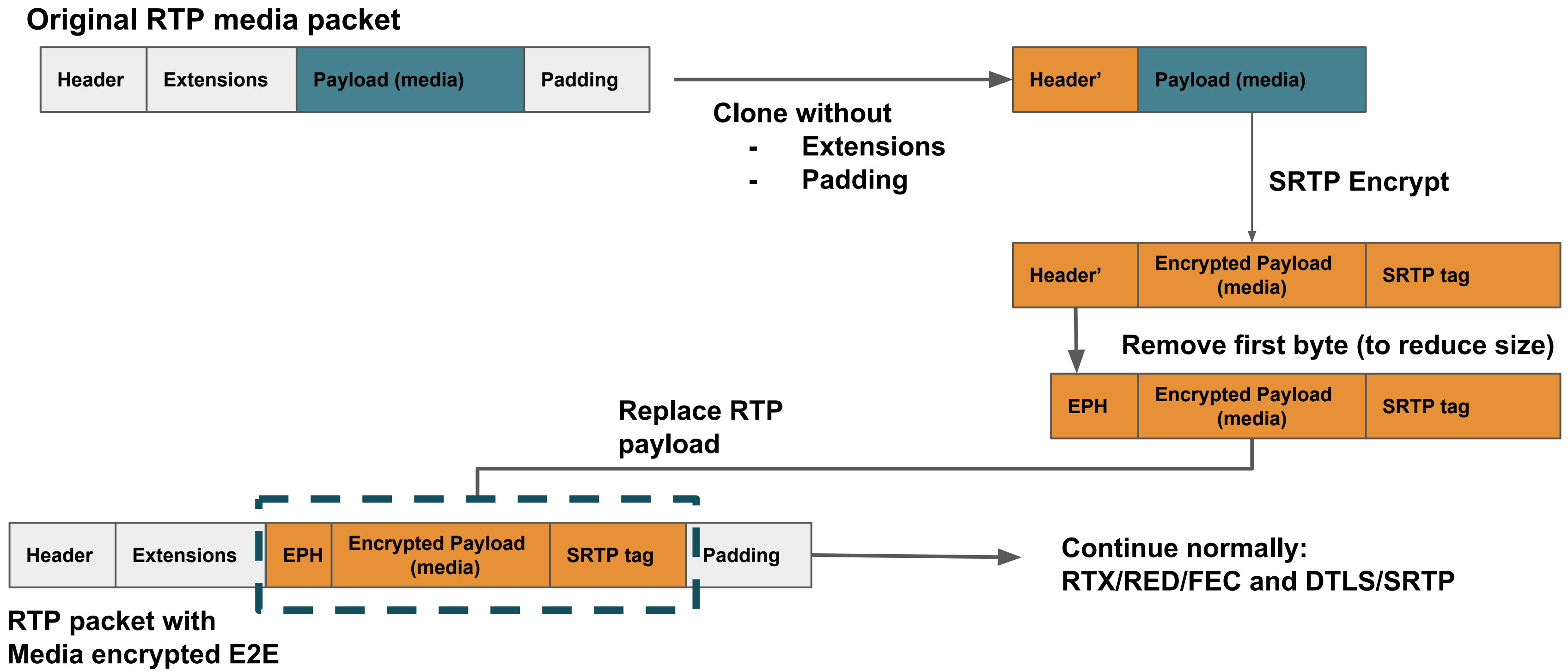
PERC LITE

End to End Media Encryption

Objectives & Key Points

- Minimum viable PERC implementation.
- Minimize impact on both endpoints and MD
- OHB is carried in the RTP payload (Encrypted Payload Header).
- No changes to the DTLS/SRTP code.
- No RTP E2E Header extensions.
- RTX/FEC/RED is supported HBH without any change.
- Key sharing is out of scope.

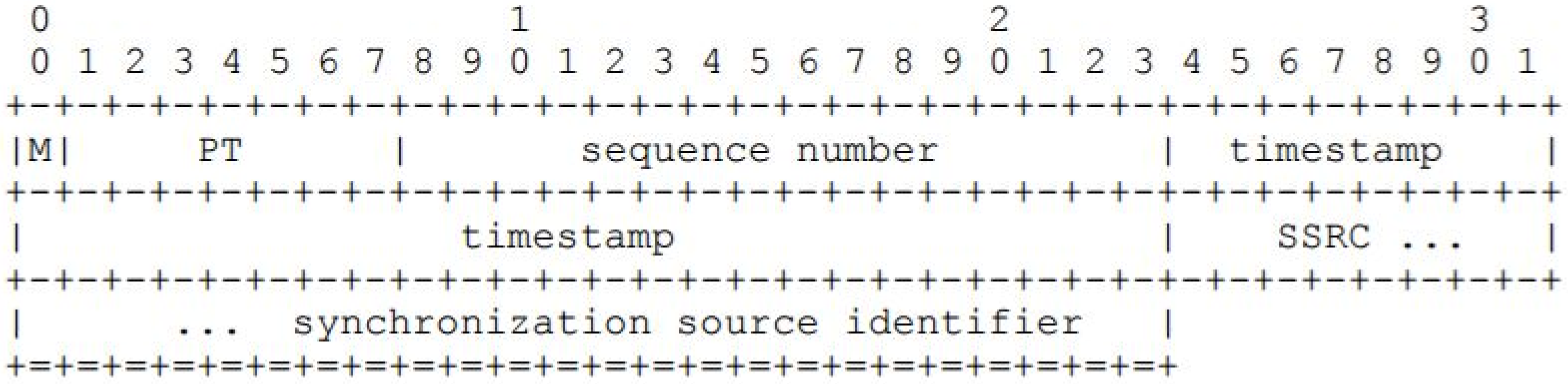
Procedures at the Media sender



Original RTP media packet
Header
Extensions
Payload (media)
Padding
Clone without
- Extensions
- Padding
Header'
Payload (media)
SRTP Encrypt
Header'
Encrypted Payload (media)
SRTP tag
Remove first byte (to reduce size)
EPH
Encrypted Payload (media)
SRTP tag
Replace RTP payload
Header
Extensions
EPH
Encrypted Payload (media)
SRTP tag
Padding
Continue normally: RTX/RED/FEC and DTLS/SRTP
RTP packet with Media encrypted E2E

Encrypted Payload Header

Almost the same than an RTP Header without version, padding and extension bits.

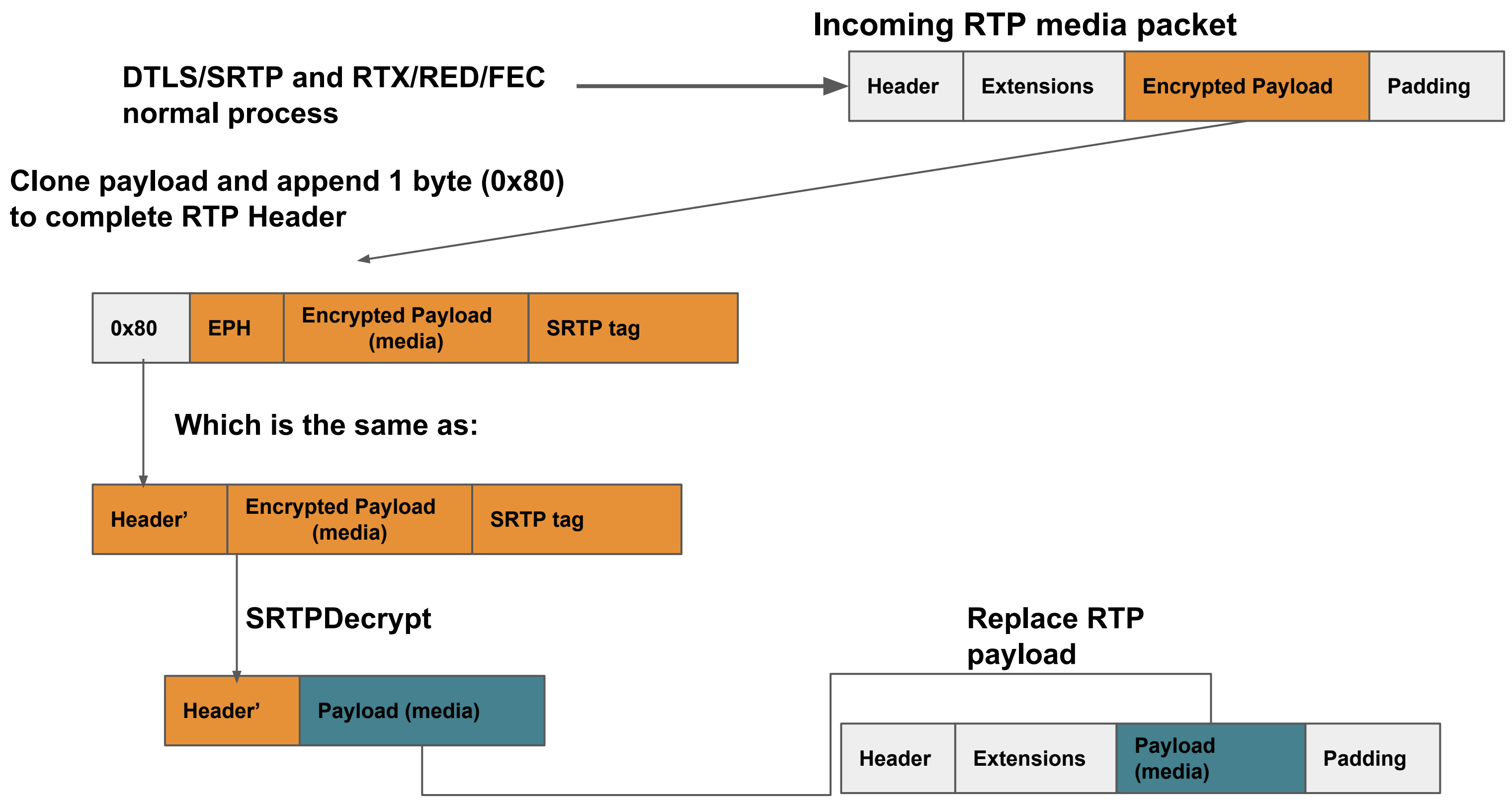


```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|M|     PT      |       sequence number         |  timestamp    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                  timestamp                    |    SSRC ...   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      ...  synchronization source identifier   |
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

Procedures at the Media Distributor

- **NO CHANGES REQUIRED IN RTP PROCESSING.**
- **NO INTEGRATION WITH KEY MANAGER REQUIRED.**
- Just support Frame Marking extension and use it to check for I frames, start & end frame marks and SVC layer indexes.
- MD can perform any transformation on the incoming and outgoing RTP.

Procedures at the Media Receiver



Status

- Draft is under construction:
  - https://github.com/murillo128/draft-perc-lite/blob/master/draft-perc-lite.xml

- Chrome 57 libwertc and chromium implementation already available:
  - https://github.com/agouaillard/perc-webrtc

- JITSI already supports End to End Media Encryption based on this proposal on its master branch (only frame marking was required).

WebRTC key setting (Informative Only)

- Until WebRTC identity is implemented, you will have to trust your javascript code anyway.
- Enable it as a peerconnection configuration parameter:

```
const pc = new RTCPeerConnection({
        mediaCryptoKey : 'VEhJUyBJUyBUSEUgMzIgS0VZIFdJVEggMTIgU0FMVCBET1VCTEUgUEVSQyE='
});
```

- Additionally: Enable it per RTPSender/RTPReceiver.