PERC Virtual Interim

29 February 2016

WG Roadmap

Signaling
Key management ← you are here
SRTP/SRTCP transforms ← (last few calls)

Today:

Requirements / architecture discussion for key management

PERC is creating an entity with intermediate privilege

Normal SRTP/SRTCP divides the world into two classes:

- In the session: Can encrypt / decrypt payload, MAC/verify headers + payload
- Not in the session: Can observe header fields, encrypted payload

PERC is about creating an entity **intermediate** between these two

- Not in the session, but gets some capabilities that things in the session have
- MDD = Network Attacker + (minimum privilege to do conferencing)

PERC key management assigns these roles

Basic requirement: Establish and distribute two crypto contexts

1. HbH context shared among participants AND MDD
2. E2E context shared among participants AND NOT MDD

E2E and HbH contexts => normal SRTP participant, full access

only HbH context => intermediate participant, partial access (i.e., the MDD)

What does the KMF do?

- Distributes E2E and HbH contexts to participants
- Distributes the HbH context to the MDD
- Authenticates to participants (at the DTLS layer)
- Participants authenticate to it (also DTLS)
- [[Might be further identity / authentication]]

… over to Adam …

KMF Location and Mobility

- Do we need to support having KMF co-located with endpoint?
  - Probably so, for non-enterprise cases
- Is the KMF static, or can it change during a conference?
  - If co-located with a participant, we can either have a single owner, and the conference dies when they depart, *or* we can have the ability to migrate among participants
- If the KMF moves, who needs to know about it?
  - MDD?
  - Web Service?
  - Endpoints?

E2E Key Lifetime

- Do we need to support E2E key rotation on user join?
  - Should there be some hysteresis here to prevent rapid rekeying as people join (e.g., at the start of a conference)?
- Do we need to support E2E key rotation on user departure?
  - If so, how quickly do we need to rotate? E.g., if several users leave in rapid succession, do we need to change the key each time?

Information Flow

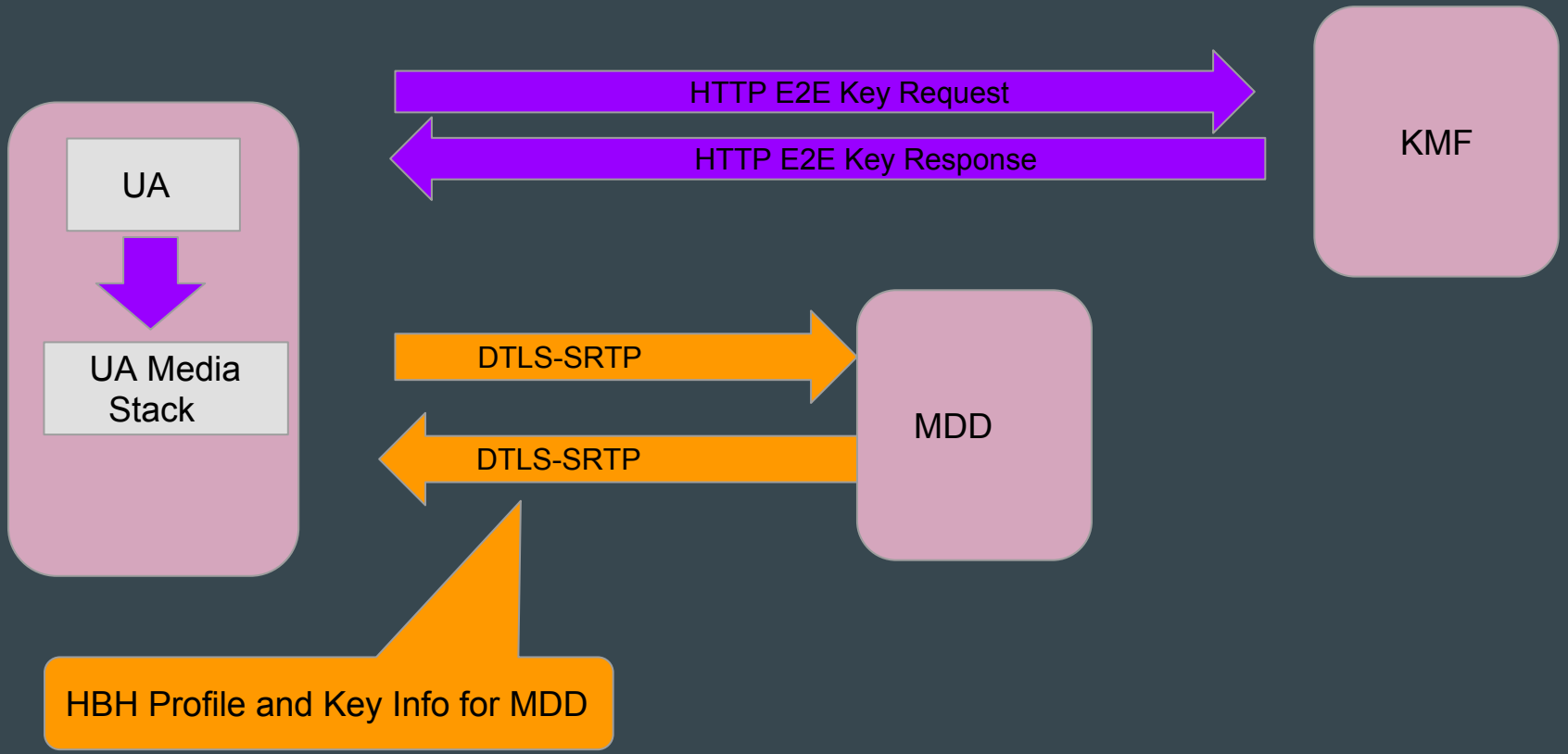
- Who generates keys?
  - For HbH, endpoint and MDD
  - For E2E, endpoint and KMF
- How are keys distributed?
  - This is the fundamental difference between the two proposals on the table
- How to have keys before they need to be used?
- How to know what keys to use when?

Overview of Ericsson Approach

- Endpoint will make a separate HTTPS POST to get the E2E keys from the KMF. The keys retrieved are not extractable from the browser into the JavaScript.
- Token to authorize the UA to fetch the keys is provided by (and therefore accessible to) the Javascript application
- The HbH key is negotiated in band with DTLS-SRTP

(More details in section 4.4 and 6.3 and of draft-westerlund-perc-webrtc-use-case-01)

HTTP and DTLS Approach

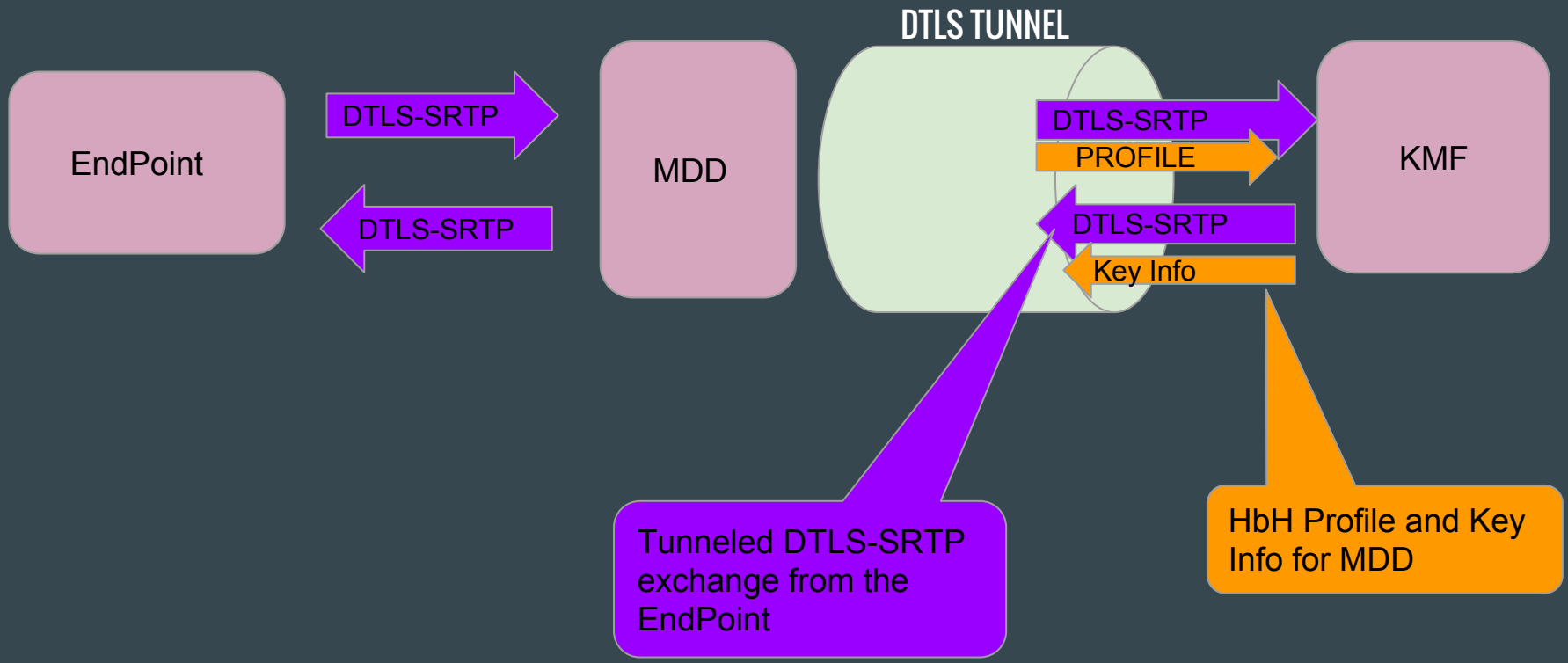


Overview of Tunnel Approach

- Endpoint does normal DTLS-SRTP keying with KMF routed through the MDD. The MDD acts like a reverse proxy for the KMF.
- The KMF and Endpoint negotiate both the HbB and E2E keys (using DTLS-SRTP)
- The KMF provides the HbH key to the MDD using the same reverse proxy tunnel

(More details in draft-jones-perc-dtls-tunnel-01.txt)

DTLS TUNNEL APPROACH



Comparison of the two approaches

Ericsson

- Effectively an out-band keying system for E2E and in-band for HbH
- Endpoints support a simple protocol for E2E key retrieval
- MDD has no interaction with KMF

Tunnel

- Effectively an in-band keying system
- Minimal changes to existing endpoints
- Guarantees MDD has keys before they are used
- Minimal changes to existing keying protocols

Thank You