WebRTC Endpoints with PERC

draft-westerlund-perc-webrtc-use-case-00

Magnus Westerlund
John Mattsson

Outline

- Introduction
- Goal
- Use Case
  - Contextual Communication
  - Outsourcing
- Challenges
- Requirements
- Strawman Solutions
- Next Step

Introduction

- The WG have in its charter:
  - “The solution should be implementable by both SIP (RFC3261) and WebRTC endpoints (draft-ietf-rtcweb-overview)”
  - “This working group will perform the following work:
    - …
      3. Document models considered for integrating the solution with WebRTC, SIP and CLUE establishment of conferencing sessions.”
- draft-westerlund-perc-webrtc-use-case-00
  - Use cases
  - Challenges
  - Requirements

Goals

- Raise awareness of challenges with WebRTC endpoints
- Ensure that the architecture and trust model we define considers these challenges
- Consider how the WebRTC integrations will performed
  - Authentication solutions
  - Group key management solutions

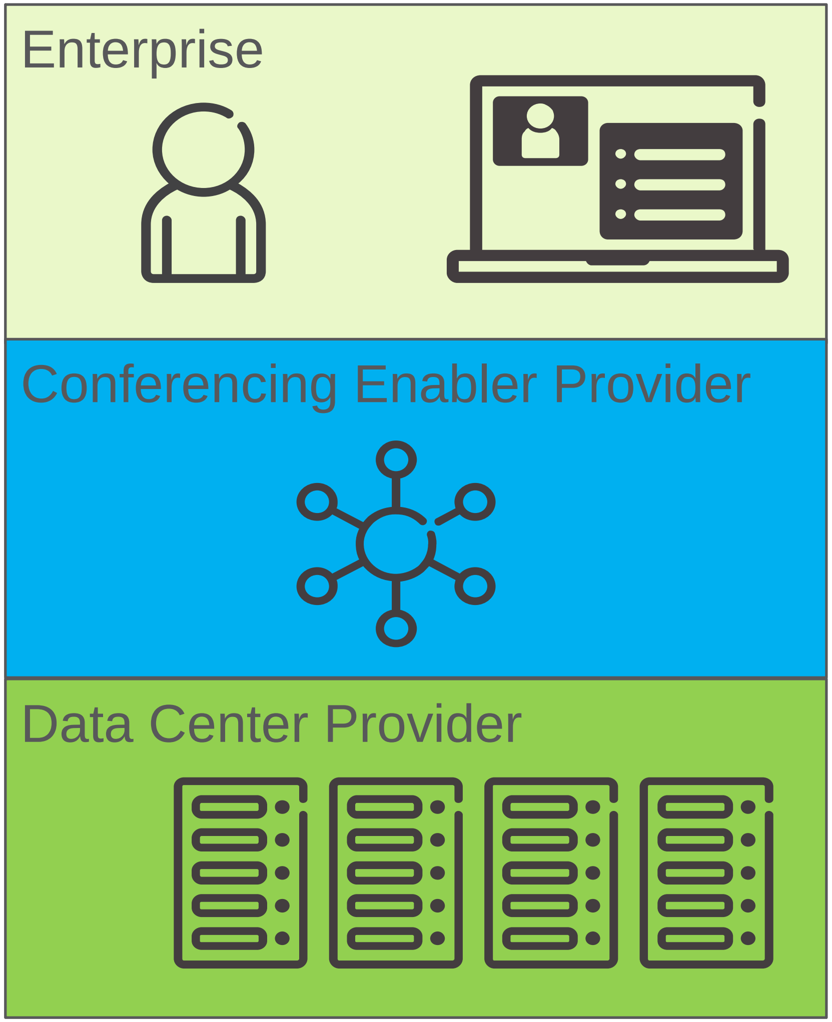
Use Case

- Contextual Communication
  - Communication related to data, information or operation
  - Presenting the context together with necessary communication options
  - Examples
    - Help desk for application where user and assistance collaborative operates app
    - Medical Expert consultation showing patient's sensor and test data, and medical history

- Contextual Communication based on WebRTC
  - Integrated around existing web-tools / front-ends
  - Servers in data centers

Outsourcing

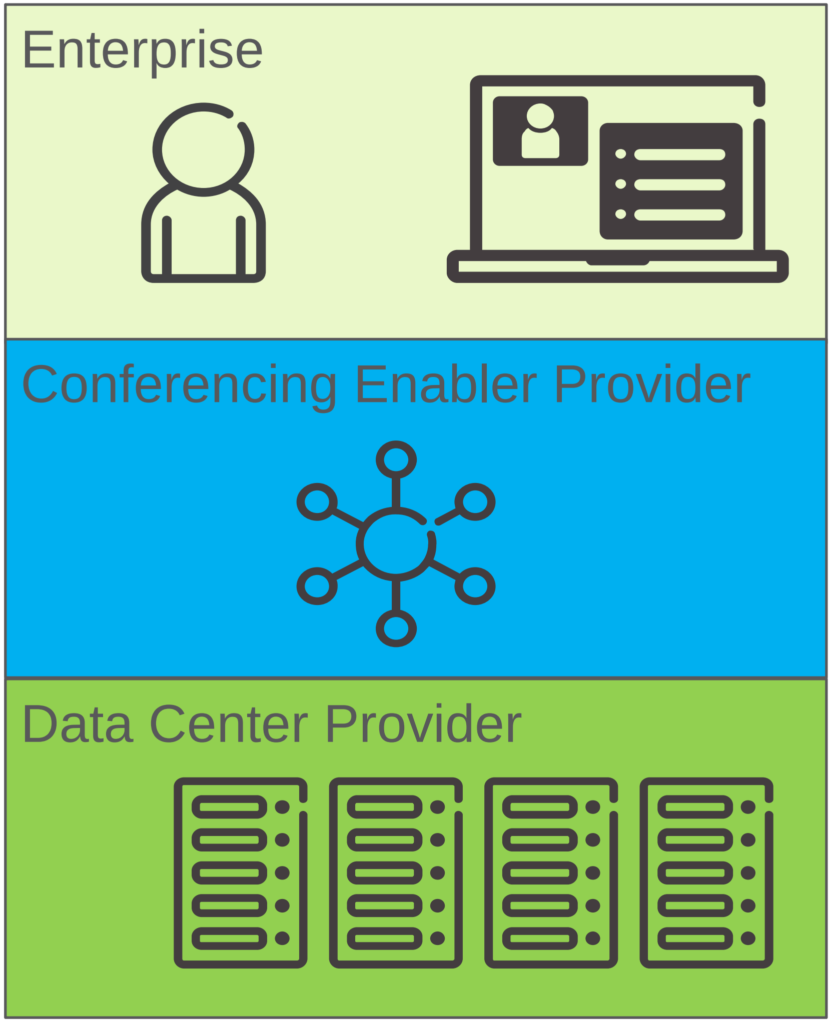
- Outsourcing functions have been very common
- Considering an Enterprise’s contextual communication
- Source Conferencing from a provider:
  - Session Signaling
  - Media Distribution Device
  - STUN/TURN
- All run on third party DCs
- Multiple players:
  - Need Trust boundaries



Trust Model

- Trusted
  - Those that are trusted with the media content of the RTP conference as well as any keying material

- Semi-Trusted
  - "honest-but-curious”
  - No media content access
  - Session establishment
  - Forwarding Decisions
- Semi-Trusted
  - Infrastructure used by Conference Provider
  - Lacking direct control



Enterprise

Conferencing Enabler Provider

Data Center Provider

WebRTC Challenges

- WebRTC is very practical in Contextual communication
- Need to ensure that WebRTC endpoints can become trusted PERC endpoints
- The PERC system must be reasonable to integrate with existing Web infrastructures in Enterprise and Service Providers

- Challenges we found:
  - JavaScript (Dis-)Trust
  - Forcing the use of Security
  - Protecting E2E Secrets
  - Securing the Authorization
  - Authentication Mechanisms
  - Participant Dynamics

Challenges - JavaScript (Dis-)Trust

- The JavaScript must considered no more than semi-trusted
  - Runs the PeerConnection establishment and associated with the session establishment
  - May not be Origins own scripts
  - Vulnerable to Cross site attacks etc.
- JavaScript MUST NOT be able to:
  - Get access to any keys
  - Access plain text media in browser (Isolation mode)
    - Applies to both locally captured and received over PeerConnection
  - Send or Forward media unless protected by the same e2e keys used when media arrived
    - Alternative is to forbid forwarding

Challenges –
Forcing the use of Security

- The web servers that are origin for a Web Service needs to be able to force usage of End-to-End security.
  - Loading application code for context or sub-context from other providers (Outsourcing session establishment)
  - Prevent that compromised JavaScript in UA revert to regular PeerConnections to be able to capture media
- The origin needs method for setting security policies on how media is treated in UA’s context
  - Forcing use of End-to-End Security
  - Forcing all media processing into Isolation Mode
  - Force which KMF instance that MUST be used for key management

Challenges – Protecting the E2E Secrets

- To maintain the confidentiality and integrity of the media exchange the End-to-End keys must be protected:
  - Only provided them to trusted UA and Identified Participants
  - JavaScript must only handle keys by reference
  - The key must not be possible to apply to other usages
    - This could enable that one could extract the key or plaintext, or at least simplify a cracking of the key
  - Keys are only need in UA during participation in conference
    - Only retrieve it when needed or just before usage
    - Destroy key when conference ends or web context is closed

Challenges – Securing the Authorization

- The participant will need to prove its right to participate in a conference.
  - Most likely by authenticating itself to a trusted node
    - Access to the KMF
    - Allowed to establish the media path with the MDD
- The resulting authorization must not be possible misuse on a behaving UA:
  - Prevent it from being used from another endpoint
    - JavaScript MUST NOT be able to move the authorization to a non-trusted endpoint and use that to retrieve the key(s)

Challenges – Authentication Mechanisms

- Enterprises as well as other service provider already have authentication mechanisms they support
  - Re-use mechanisms when sufficiently safe
  - Need to be flexible in which mechanisms are used by participants to assert their identity
- The design should avoid requiring implementation in UA for a specific authentication mechanism

Challenges – Participant Dynamics

- The charter contains a SHOULD goal on a higher security level:
  - A late joiner MUST be unable to access media content prior to joining.
  - A participant leaving MUST be unable to access media after having left
- This will be challenging!
  - Can not trust the session establishment signalling (semi-trusted)
    - But, it can be used as a hint
    - Trusted function to maintain current rooster?
    - Need for active keep-alives
  - What timeliness is needed, 100 ms, 1 sec or 30 sec?
  - A lot of rekeying events
    - Avoid media interruptions for the active participants
  - A Participant must be able to check who the other participants are
- We need to consider how to solve this from start!

Requirements (1/3)

A. The Web application running in the User Agent MUST NOT be able to compromise the content confidentiality.
   - Including getting access to media content (raw or unencrypted) in the user agent through API or shared resources.

B. The conference provider's application (server as well as in the user agent) MUST NOT be able to downgrade the intended security properties and policies established by the service provider and the core application.

Requirements (2/3)

C. The key material for the end-to-end protection MUST NOT be possible to extract from any web application.
   - The user agent MUST protect the key-material against extraction by user or other software running on the same device.
   - The key material MUST be bound to the usage its intended to prevent leakage.
   - Upon the termination of the conference or the browsing context containing the application the key material SHALL be deleted

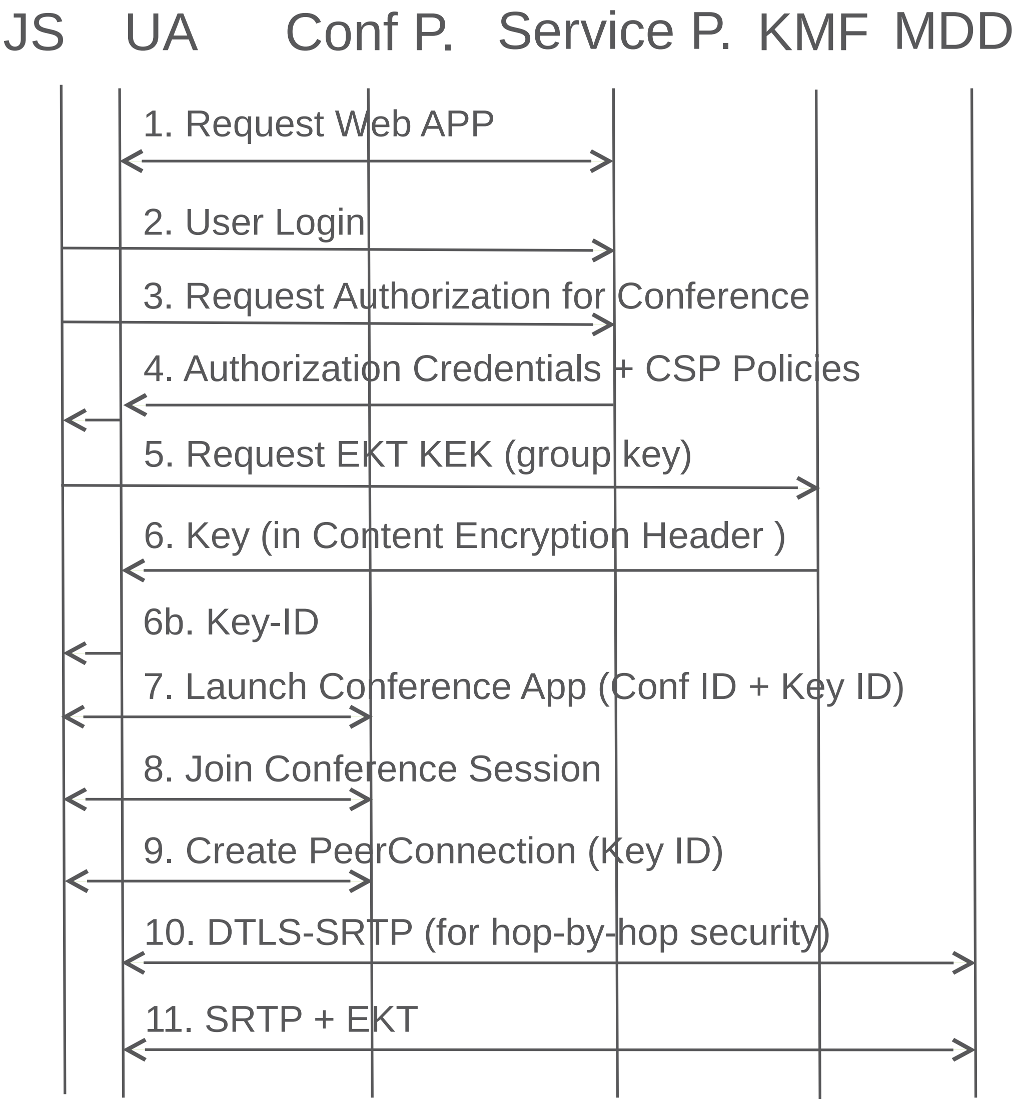
D. Different Authorization methods MUST be supported.
   - It's preferable that authorization methods can be supported without user agent modifications.
   - The authorization credentials MUST be bound to endpoint where the participant provided its credentials.

Requirements (3/3)

E. The design SHOULD support confidentiality where only the current set of participants has access to the media content.

Straw Man Solution

- Service sets that E2E security is to be used
- Service Authenticates user
- With Policies in place conference provider application can be loaded into (sub-)context



JS UA Conf P. Service P. KMF MDD

1. Request Web APP
2. User Login
3. Request Authorization for Conference
4. Authorization Credentials + CSP Policies
5. Request EKT KEK (group key)
6. Key (in Content Encryption Header)
6b. Key-ID
7. Launch Conference App (Conf ID + Key ID)
8. Join Conference Session
9. Create PeerConnection (Key ID)
10. DTLS-SRTP (for hop-by-hop security)
11. SRTP + EKT

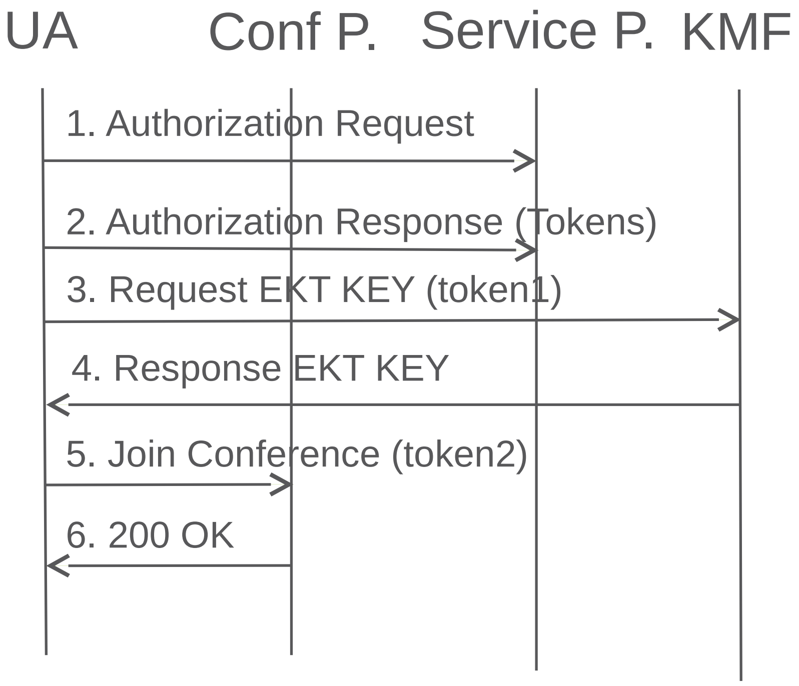
Policy

- To ensure that Origin’s policies on secure media handling are followed:
  - JavaScripts are not trusted
  - Policies must be applied to Web Context in UA without JS interfering
- Proposal:
  - Use Content Security Policies (CSP) directives to UA
  - Goes in HTTP header or “root” document
  - Consumed by UA

- New Directives:
  - Isolation Mode: Force web context into media isolation
  - KMF Instance: A directive identifying the KMF instance that is allowed to provide keys
    - Identified using Certificate Fingerprints
  - End-To-End Security MUST be used:
    - May be implicit from above

Authorization

- Flexible Authorization
- Easily integrated
- Re-usable for multiple purposes
  - Login into Conference
  - Retrieve EKT KEK
- Use of OAUTH2 would enable this



UA

Conf P.

Service P.

KMF

1. Authorization Request

2. Authorization Response (Tokens)

3. Request EKT KEY (token1)

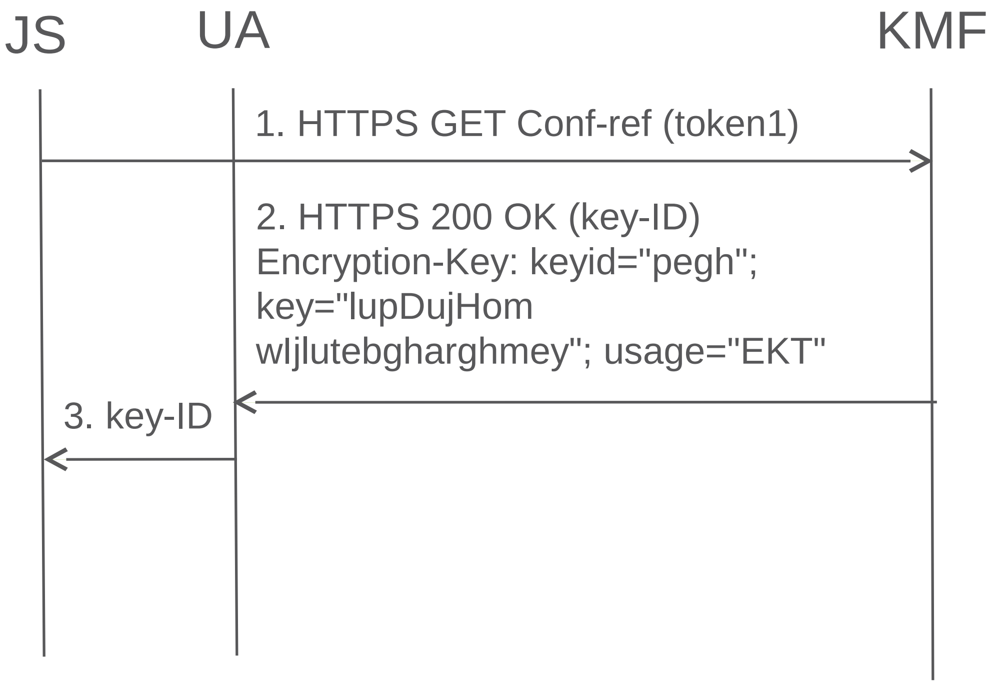
4. Response EKT KEY

5. Join Conference (token2)

6. 200 OK

EKT KEK Distribution

- The EKT Key Encryption Key (KEK)
- Distribute it with the HTTP header for Encrypted Content-Encoding:
  - draft-thomson-http-encryption
- Storage of Key on UA
  - Should be bound to Context
- JS only gets Key-ID



JS

UA

KMF

1. HTTPS GET Conf-ref (token1)

2. HTTPS 200 OK (key-ID)
Encryption-Key: keyid="pegh";
key="lupDujHom
wljlutebgharghmey"; usage="EKT"

3. key-ID

Handling Group Dynamics

- For new participant:
  - Derive new EKT master key based on old
  - Each endpoint generates new e2e SRTP Master Key
  - Avoids having to distribute EKT master key
    - Only authenticated signal to start using new
- For leaving participant:
  - KMF must push new group key
  - Need for active keep-alives to maintain soft state for who is current participants
- Implications on EKT
  - Needs e2e SRTP Master Key Identifier (MKI)
  - Needs Key generation counter to determine when derivation is needed
    - Use expanded SPI?
- Each Key ID + Generation bound to particular rooster instance