SUPPORT FOR SDES IN WEBRTC

RTCWEB INTERIM MEETING 31/1-1/2

DECISIONS TO BE MADE

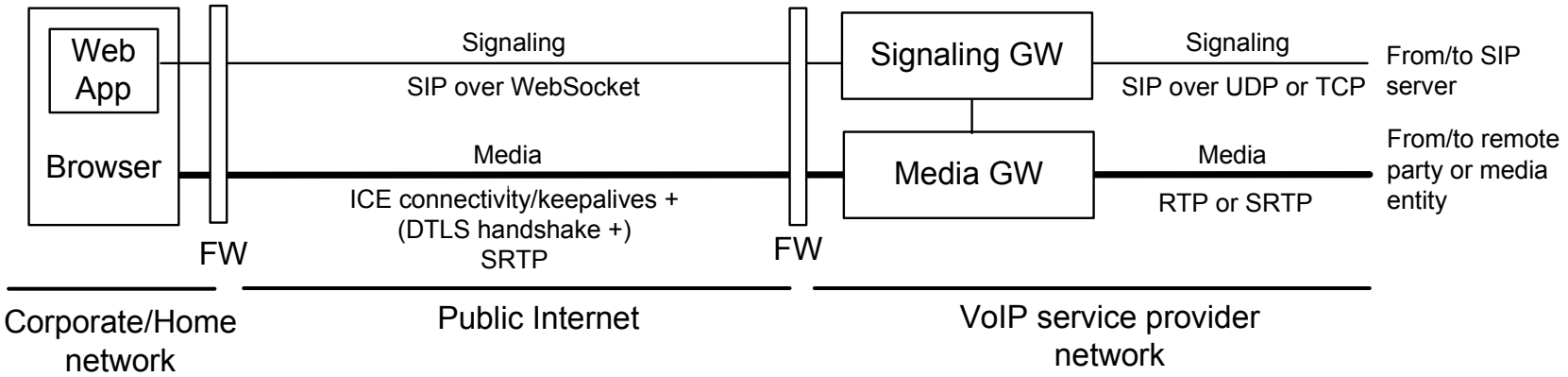
- Which SRTP key management schemes should be supported in WebRTC?
  - Only DTLS-SRTP
  - Both DTLS-SRTP and SDES
- If SDES is supported, should it be optional or mandatory to implement in browsers?
- The following question can be saved for later:
  - How should an application enable SDES?
    - explicitly via a separate JavaScript function/parameter
    - negotiated via SDP offer/answer
  - What kind of consent mechanisms are required?

WHY EVEN CONSIDER SDES?

- The main use for SDES is interworking with other VoIP systems
- Interworking is an important use case
  - Millions of existing SIP/RTP devices
    - Desk-phones
    - Soft-phones
    - Conference phones
    - Analog Telephone Adapters
  - Approximately 5 Billion mobile phones and 1.5 Billion landlines are reachable through PSTN gateways
  - Several services available such as conferencing and voicemail
  - 4G mobile phones will use SIP/RTP for voice/video communication (with plain RTP or SRTP + SDES)

WHY IS INTERWORKING SIMPLIFIED?

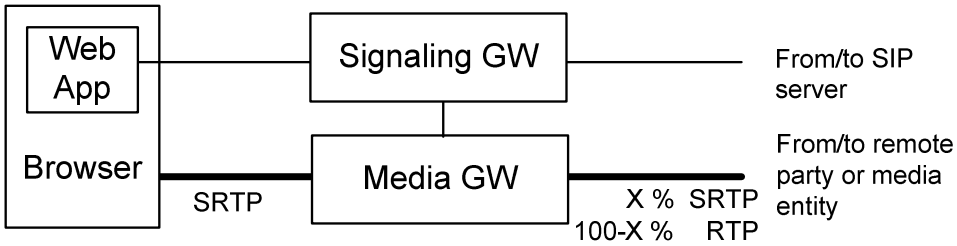
Reason 1: Reduced Complexity of WebRTC-SIP Media Gateway



- A media-plane gateway might always be needed, but we should at least strive to make it as simple as possible
- Media gateways are typically very expensive since they need to handle a large number of users and involve special purpose hardware
- Already today there are SBCs that perform SRTP termination on behalf of endpoints with SDES based keying
  (DTLS-SRTP is uncommon)

WHY IS INTERWORKING SIMPLIFIED?

Reason 2: Reduced Processing (Less SRTP Terminations)



- A large part of the existing SIP/RTP devices support SRTP and most of them that do, use SDES based keying
- If SDES is supported by browsers, a significant part (X %) of all calls would not need to be encrypted/decrypted by the gateway
- The percentage X of devices supporting SRTP + SDES appears to be growing. Also note that future 4G handsets use SIP/RTP for voice communication and will support SDES.
- Nearly all calls have to be encrypted/decrypted by the gateway if only DTLS-SRTP is supported by browsers

SECURITY IMPLICATIONS OF SDES

Case 1: The application cares about its reputation and will try to protect its users

- The web applications that use SDES do this at their own risk and must take the necessary precautions
  - The application download and the signaling must be TLS protected
  - The browser and the remote endpoint should delete the keys from memory once the call is over
  - The keys should not be stored/logged by signaling intermediaries
  - Care must be taken when writing the web application to avoid any untrusted/malicious code from being executed as part of the application
- Conclusion: Its possible to use SDES securely but you don’t have the same room for mistake as with DTLS-SRTP

SECURITY IMPLICATIONS OF SDES

Case 2: The application is the attacker and it will try to intercept the user's call if given the chance

- Even if the media transport was perfectly secured the application could still do any of the following
  - Create a new PeerConnection object and forward the stream to a third party
  - Call MediaStream.record()
  - Repeatedly call Canvas.drawImage(Video), Canvas.getImageData()
  - Mozilla Audio Data API, W3C Web Audio API, and W3C MediaStream Processing API all have functions for accessing the data buffers of an <audio> element
  - Or do any of the above but on the remote user's side
- In the current W3C specification, the user is only prompted once when navigator.getUserMedia() is called

SECURITY IMPLICATIONS OF SDES

Case 2: The application is the attacker and it will try to intercept the user's call if given the chance (contd.)

- Lets assume the problem on the previous slide can be handled through additional consent dialogs
- The malicious application would then attempt to decrypt the SRTP traffic instead
- If the application can turn on SDES this should be fairly easy
  - The hardest part is to re-route the media
  - Could the use of SDES be controlled through user consent? (e.g. similar to MediaStream.record())
  - How should the implications of SDES be explained to the user?
- Bear in mind that a dedicated attacker is able to intercept even a DTLS-SRTP protected call provided that
  - The users are not alarmed by the "new fingerprint" warning (will this warning be shown? And if so, how intrusive will it be?)
  - The proposed identity mechanism can be turned off via JavaScript without causing alarm, or the users are willing to accept one of the application's own identity providers

ALTERNATIVE SOLUTION IF SDES IS UNACCEPTABLE

- There are two problems with DTLS-SRTP
  - Interworking: the keys are always negotiated which makes it impossible to interwork with SDES
  - Implementation: Everything that affects the media plane is harder to implement
- An alternative solution is to let each party encrypt its SDES key (and possibly other information as well)
  - This solution is compatible with SDES and does not affect the media plane
  - The encryption could be done using either the remote peer's public key or a negotiated Diffie-Hellman key
  - Could be combined with the proposed identity mechanism
  - Downside: requires one extra half-roundtrip and details need to be worked out


ERICSSON