

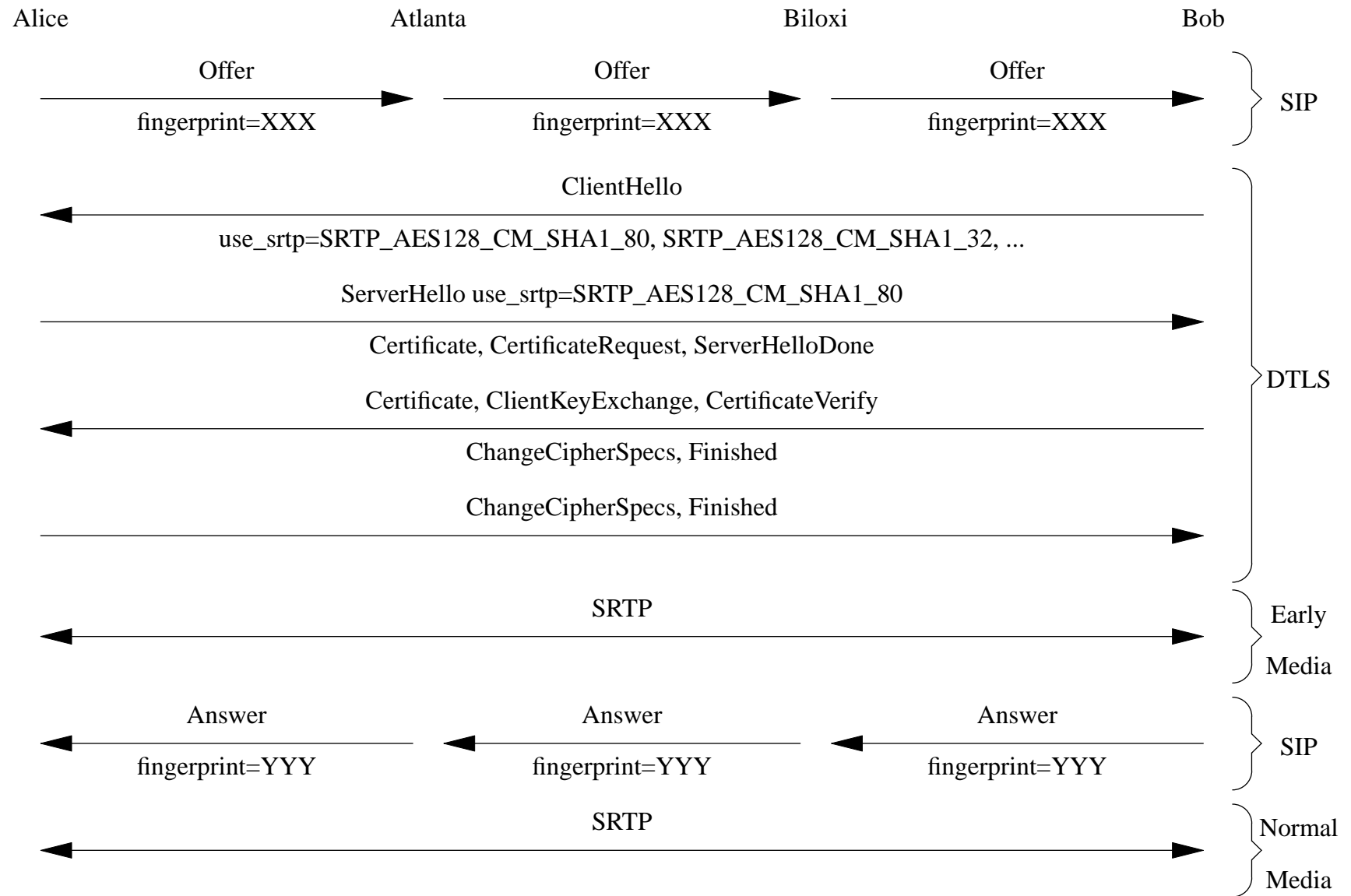
Datagram Transport Layer Security (DTLS) Extension to Establish Keys for Secure Real-time Transport Protocol (SRTP) (Phew!)

Eric Rescorla
David McGrew

Overview

- SDP signals “I’m willing to do DTLS” (and here’s my fingerprint)
- Do DTLS key exchange in media channel
 - Allows reuse of existing DTLS authentication/key establishment mechanisms
 - Use extensions to negotiate SRTP protection profiles
- Use DTLS master secret to generate SRTP traffic keys

Message Flow



TLS Handshake Extension

```
uint8 SRTPProtectionProfile[2];

struct {
    SRTPProtectionProfiles SRTPProtectionProfiles;
    uint8 srtp_mki<255>;
} UseSRTPData;

SRTPProtectionProfile SRTPProtectionProfiles<2^16-1>;

SRTPProtectionProfile SRTP_AES128_CM_SHA1_80 = {0x00, 0x01};
SRTPProtectionProfile SRTP_AES128_CM_SHA1_32 = {0x00, 0x02};
SRTPProtectionProfile SRTP_AES256_CM_SHA1_80 = {0x00, 0x03};
SRTPProtectionProfile SRTP_AES256_CM_SHA1_32 = {0x00, 0x04};
SRTPProtectionProfile SRTP_NULL_SHA1_80      = {0x00, 0x05};
SRTPProtectionProfile SRTP_NULL_SHA1_32      = {0x00, 0x06};
```

Transporting DTLS Handshake Traffic

- Current draft:
 - Carried over same channel as media
 - Directly over UDP
 - Demuxable from RTP/STUN by first byte (S 3.6.2)
 - One DTLS connection per media stream
- Other alternatives
 - In RTCP channel
 - Header extension (a la ZRTP)
- These issues are the same for any media-stream key management protocol

Requirements Evaluation

| | |
|---|---|
| R1: Forking and retargeting MUST work with all end-points being SRTP. | Yes |
| R2: Forking and retargeting MUST allow establishing SRTP or RTP with a mixture of SRTP- and RTP-capable targets. | Yes |
| R3: With forking, only the entity to which the call is finally established, MUST get hold of the media encryption keys. | Yes (separate key exchange to each peer) |
| R5: A solution SHOULD avoid clipping media before SDP answer without additional signalling. | Yes |
| R6: A solution MUST provide protection against passive attacks. | Yes (including malicious proxies) |
| R7: A solution MUST be able to support Perfect Forward Secrecy. | Yes (DHE modes) |
| R8: A solution MUST support algorithm negotiation without incurring per-algorithm computational expense. | Yes (cipher suites negotiated first) |
| R9: A solution MUST support multiple cipher suites without additional computational expense | Yes |
| R10: Endpoint identification when forking. The Offerer must be able to associate answer with the appropriate flow endpoint. In case of forking one might not want to perform a DH with every party but instead to associate the SDP response with the right end point. This is a performance related requirement. | Yes (but latency tradeoff) |
| R11: A solution MUST NOT require 3rd-party certs. If two parties share an auth infrastructure they should be able to use it. | Yes (fingerprints but 3rd-party certs are usable) |

Current status

- Bunch of drafts
 - draft-mcgrew-tls-srtp-00, draft-fischl-sipping-media-dtls-00, draft-fischl-mmusic-sdp-dtls-00
 - Looking for feedback
- Prototype implementations in OpenSSL and EyeBeam (thanks Derek MacDonald, Dragos Liciu, Jason Fischl, Nagendra Modadugu)
- Open issues
 - Best way to carry handshake traffic
 - How/what to express in SDP
 - * Not an AVT issue