Signaling and Media Security

SIP and Security

SIP entities are potential target of a number of attacks, e.g.

- Masquerading / impersonation / identity spoofing
- Man-in-the-middle (MITM) attacks
- Eavesdropping
  - Media streams
  - Call signaling
- Traffic analysis
- Theft of service
- Connection / session disruption or termination
- Replay
- Denial of service (DoS)

Typical threats for distributed applications using the public Internet

Some countermeasures:

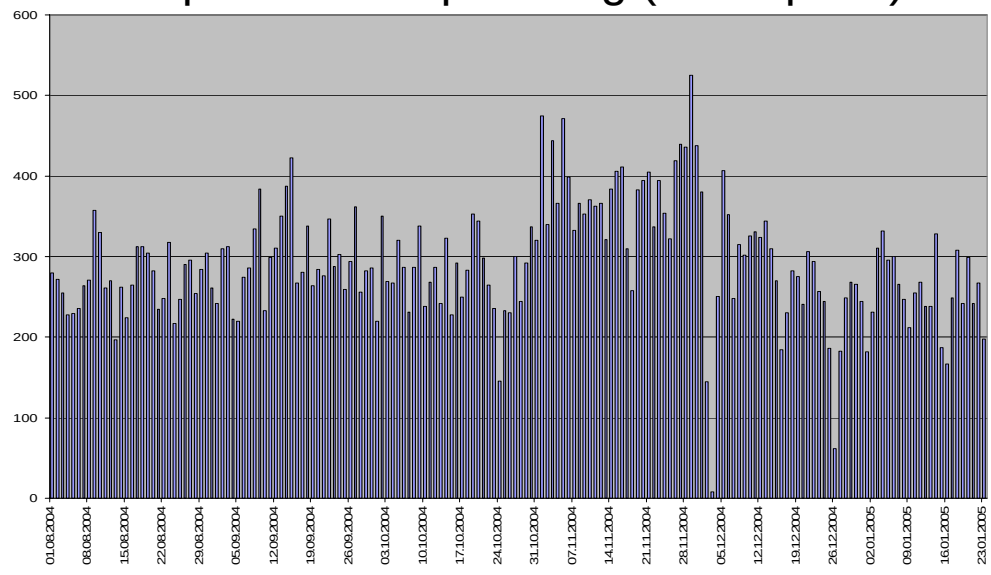
- Client and server authentication
- Request authorization
- Encryption
- Message integrity checks + replay protection

Reuse existing security mechanisms.

Why SIP Security?

- Ensure privacy
  (media encryption, anonymous calls, personalized services, ...)
- Billing and accounting
  (probably pay for services, assured bandwidth, etc.)
- Regulatory requirements
  - Call id blocking
  - Prosecute abuse (call tracing facility)
  - Emergency call service
  - Multi-Level Priorization and Preemption

Sample E-Mail Spam Log (incomplete)



SIP & SPAM

- SPAM for IP telephony: SPIT [**`draft-ietf-sipping-spam-05.txt`**]
  - Call spam (some 3 orders of magnitude cheaper than PSTN)
  - Instant messaging spam
  - Presence (subscription) spam
- Some countermeasures
  - Content filtering (like with email)
    - Does not work for voice and video; applicable to IM
  - Blacklists
    - Large (virtually infinite) supply of source addresses makes successful filtering difficult
    - Authentication (if used) may help
  - Whitelists
    - Work well for IM (“buddy lists”)
    - Problem of introducing someone (first contact)
  - Consent-based communications
    - Requests seeking consent may be intrusive

HELSINKI UNIVERSITY OF TECHNOLOGY
NETWORKING LABORATORY

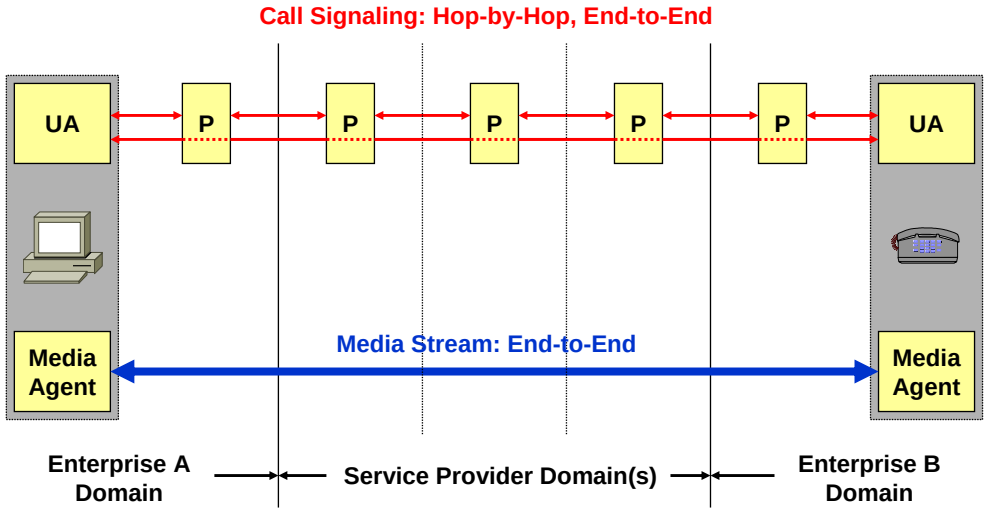
SIP & SPAM

- Some countermeasures
  - Reputation systems
    - Positive: make people’s acceptance depend on ‘praise’ by others
    - Negative: make people’s non-acceptance depend on ‘curse’ by others
    - Positive ones seem better negative ones
    - Users in whitelists (i.e., buddy lists) usually implicitly “praised”
  - Address obfuscation
    - Similarly effective or ineffective as for email addresses
  - Limited use addresses
    - May defeat the purpose of reachability (how to manage addresses?)
  - Turing tests
    - Depend on human recognition of images, words, etc.
    - May be “outsourced”
  - Computational puzzles
    - Must be workable with very heterogeneous devices (2 orders of magnitude in proc. power)
    - Spammer may have many ‘zombies’ at their disposal

SIP & SPAM

- Some countermeasures
  - Payments at risk
    - Sender pays always; receiver refunds if ok
    - Transactional cost?
  - Legal action
    - Yes: spam is a social problem in the first place
    - Need to track down people in the first place
  - Circles of trust
    - Does this scale to large domains or institutions?
  - Centralized SIP providers
    - Transitive trust between local SIP providers and inter-domain SIP providers
- Anti-spam measures
  - To be take proactively!
  - Provide strong user authentication
  - Use white lists within (service provider) “networks”
  - Solve the introduction problem

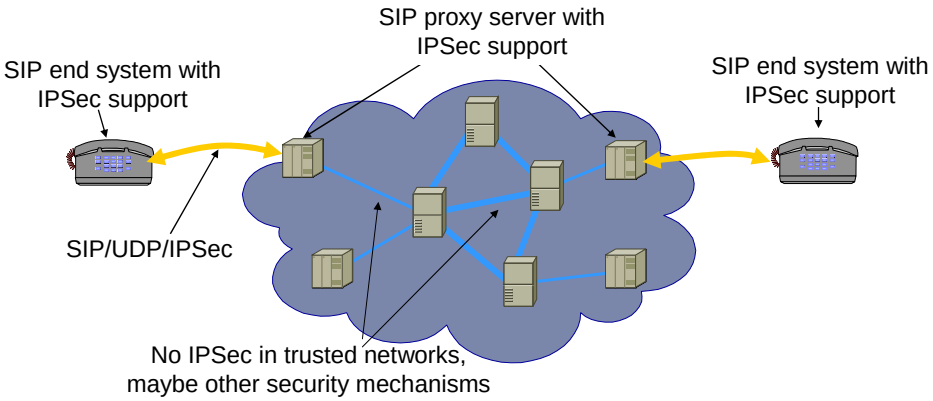
SIP Security Overview



Hop-by-hop Encryption of SIP Messages

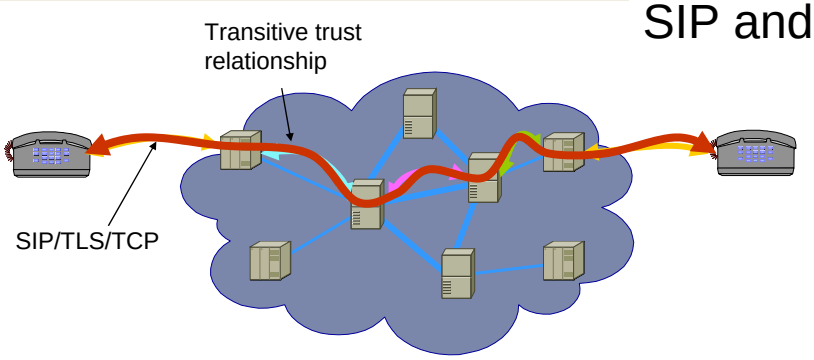
- Lower layer mechanisms
  - Applicability depends on link layer technology
- VPN-like tunnel using IPSec
  - Suitable e.g. for coupling sites of a company
  - Need OS-support (required for IPv6 anyway)
- SIP over TLS (Transport Layer Security)
  - Access to outbound proxy
  - Call routing to ITSP
  - Call routing between neighboring ITSPs (agreements!)
  - In most cases, only servers have certificates
- Chain of trust: suitable also for authentication
  - e.g. in trusted networks

Hop-by-hop encryption with IPSec



- Possible deployment scenario
  - IPSec between hosts inside an administrative domain
  - Established trust relationships, pre-shared keys
- Security functions independent of SIP layer

SIP and TLS



- Hop-By-Hop security
  - TLS secures connections between SIP entities
  - TCP only!
- Useful for hosts with no pre-existing trust association
- Usage of TLS is coupled with SIP
  - Needs to be signalled, e.g, in Via headers
  - MUST be implemented by proxy servers, redirect server and registrars
- Basic model
  - A UA establishes a trust association with his outbound proxy (TLS connection)
  - The proxy build trust associations with other proxies/UAs
    - May require certificate exchange and validation
- Problems
  - Call setup delays
  - Resources for open TLS connections

End-to-End Encryption of SIP Messages

- Impossible for entire message: proxies do call-routing
- Use SIP message bodies to carry confidential information
  - Can be used to achieve end-to-end encryption
    - E.g., when the network cannot be trusted or other mechanisms are not available
  - Does only protect the message body, not the header fields
- S/MIME
  - Allows to encapsulate arbitrary content for security purpose
  - Encryption and digital signature
- Issue: Key distribution problem *prior to call*
  - No global PKI
  - Certificates for SIP proxies, typically not for users
  - Typically no shared secrets
  - Validation of end user certificates will have to rely on public key infrastructures (or pre-configured certificates)

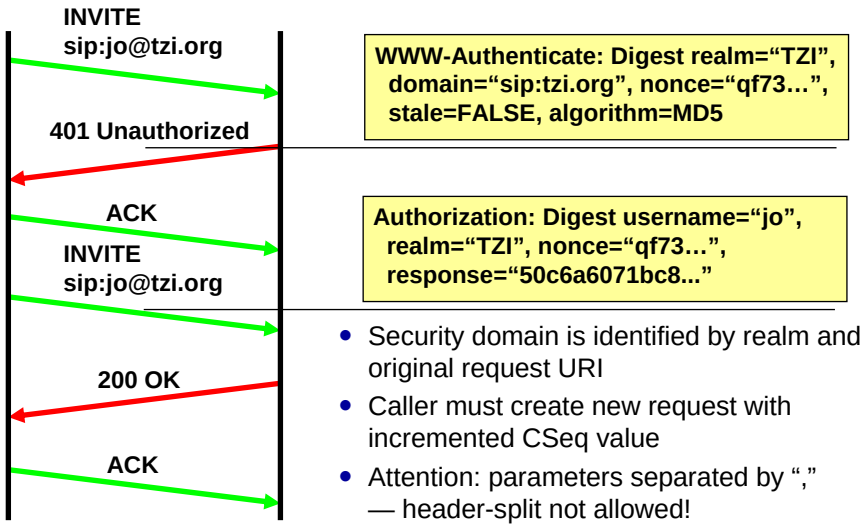
SIP Authentication

- Validate a peer's identity
- Hop-by-Hop
  - Use underlying TLS + certificates where available (usually servers)
- End-to-middle (infrastructure)
  - Authenticate UA against a SIP server (e.g., proxy, registrar)
  - Requires centralized assertion of identity
  - Requires commonly trusted infrastructure components
    - Typically reflects contractual relationships quite well
- End-to-end
  - Use S/MIME and X.509 certificates (usually against a common CA)
  - Option: self-signed certificates
    - Allows to assure that you have talked to the same peer before
    - Similar strength to SSH; easier deployment

Simple SIP Authentication

- HTTP *digest* authentication
  - basic authentication deprecated
- Challenge with WWW-Authenticate: header field
- Authorization: header field
  - End-to-end authentication of UAC
  - Digitally sign message body and header fields following the *Authorization:* header
  - Use canonical form
  - Fields to be changed must precede *Authorization*

Authentication: Example Call Flow

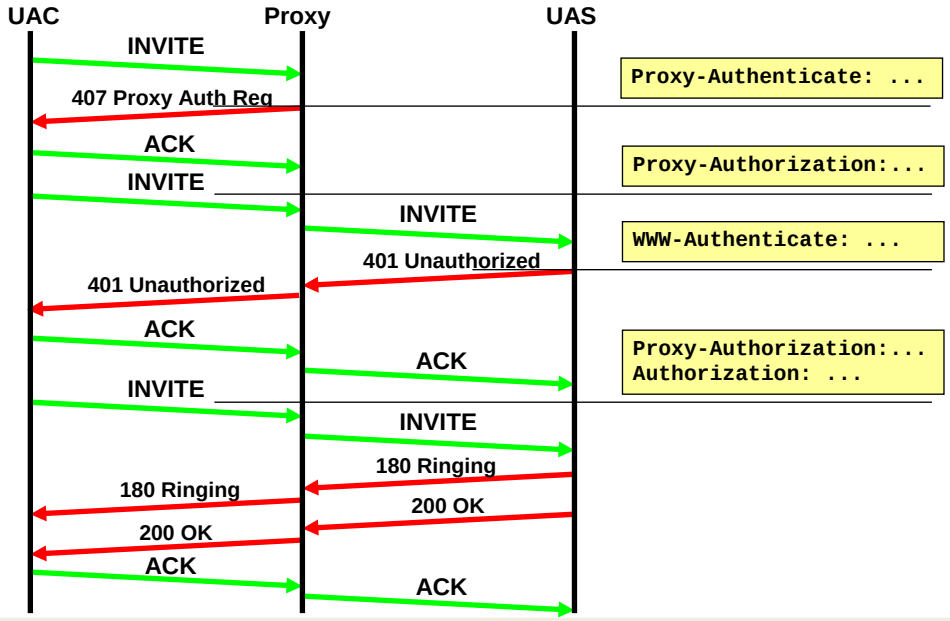


- Security domain is identified by realm and original request URI
- Caller must create new request with incremented CSeq value
- Attention: parameters separated by "," — header-split not allowed!

Authentication for Proxies

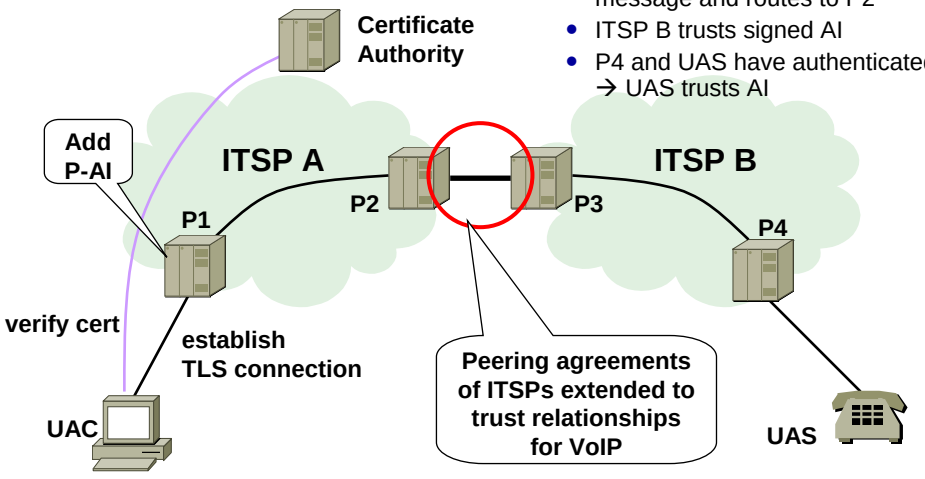
- Similar to endpoints (HTTP Digest)
- Proxy rejects client request with "407 Proxy Auth required"
  - Proxy-Authenticate: header
  - Multiple proxies along the path may challenge
- Client resubmits request with credentials for Proxy
  - in Proxy-Authorization: header
  - Multiple headers with credentials may need to be included

Multiple Authentication Steps



Asserted Identity and Transitive Trust Relationships

- UAC establishes TLS connection
  - Proxy provides certificate
  - UAC verifies cert
- Password-based authentication of calling user
- P1 adds P-Asserted-Identity:, signs message and routes to P2
- ITSP B trusts signed AI
- P4 and UAS have authenticated → UAS trusts AI

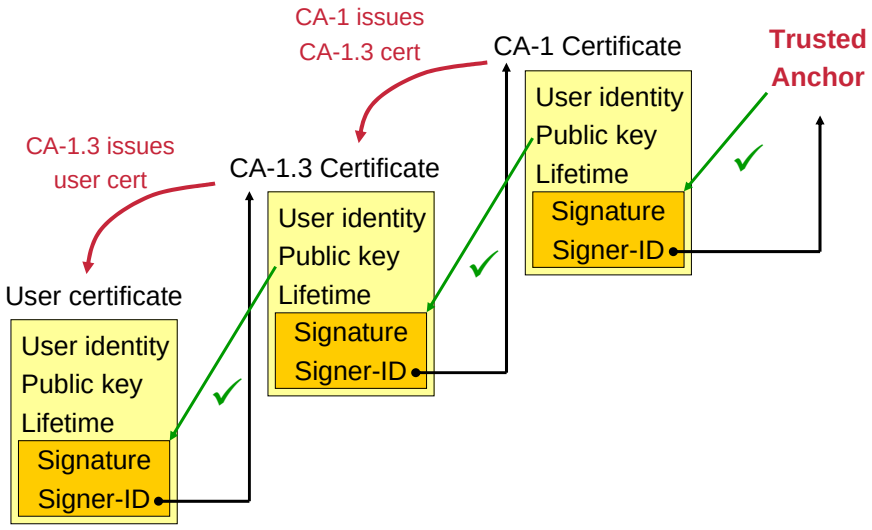


Privacy of Calling Party Information

- Option-tag privacy to enforce treatment at proxies
- Extension header Privacy:
  - Initiator can specify a required level of privacy
  - Typically, intermediary has to obscure certain header fields
    → B2BUA
  - Different types of privacy information (header, session, user)
- Use asserted identity mechanism if authentication required
  → need trust relationships between ITSPs

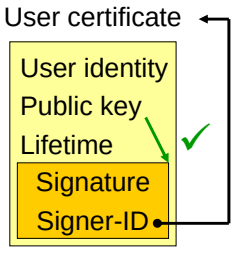
End-to-End Signaling Security: Certificates and S/MIME

CA Hierarchy in Public Key Infrastructure



Self-Signed Certificates

- Certificates usually handed out by CA
  - Workable for servers
  - Impractical + expensive + hard to deal with for end users
- Alternative: self-signed certificate
  - Self-generated key pair
  - Self-created certificate
    - Signed with the own key
  - Sufficient to satisfy (syntactical) protocol requirements (e.g., TLS)
  - Semantics may suffice for certain applications, too
    - Certificates convey during initial contact
    - Stored at both peers
    - Allows to validate that the same peer was contacted before
  - Limitations
    - First contact MUST be trusted
    - Certificate revocation difficult

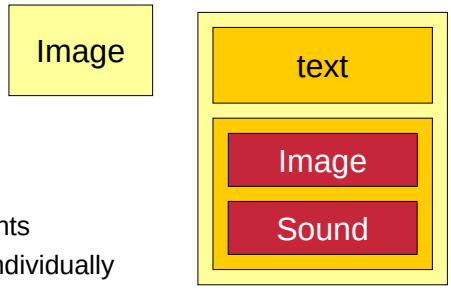


Reminder

- Please send us your group information
  - "Last name:First name:Student ID:email address"
- Assignment 2 will be presented on Thursday
  - Tomorrow, we will talk about NAT and firewall traversal
- Remember: there is no fixed deadline for assignment 1, but don't let it slip (too much).
- If you have still questions about assignment 1, bring them on Thursday. There will be enough time.

MIME

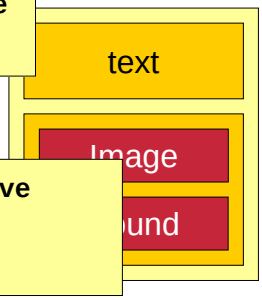
- Multipurpose Internet Mail Extensions
  - Not just mail: used with HTTP, SIP, and many other application protocols
- Define the purpose of a piece of content (in a message body)
  - Type, encodings
  - Intended interpretation
  - Specify additional parameters
  - Allow for references
- Allow for multipart contents
  - Arbitrarily nested pieces of contents
  - Specify the above for each part individually



MIME

- application protocols
- message body)
  - Allow for references
- Allow
  - Arb
  - Spe

```
Content-Type: image/jpeg
Content-Length: 5489
Content-Transfer-Encoding: base64
Content-ID:  42
Content-Description: an image of a tree
Content-Disposition: attachment; ...
```




```
Content-Type: multipart/alternative
Content-Type: multipart/mixed
Context-Type: multipart/related
```


S/MIME

- Example for an *application layer* security mechanism.
- Security services for sending and receiving MIME data
  - Authentication, message integrity, non-Repudiation of origin, confidentiality
- Origin: e-mail. But not restricted to e-mail
  - May be used with HTTP, SIP, and other application protocols
- Relies on the *Cryptographic Message Syntax* (CMS)
  - Specification for authenticating, digitally signing and encrypting contents
  - Security functions are applied on *envelopes* containing message content (or other envelopes)
  - May embed certificates and other key management information
- Uses public key cryptography and certificates
  - Most useful with a PKI (e.g., CA hierarchy)

S/MIME Processing Steps

- S/MIME messages are combination of MIME bodies and CMS content types
  - Several MIME types as well as several CMS content types are used
- Data to be secured is always a canonical MIME entity
  - Secure MIME entity: sub-part, sub-parts, or an entire MIME entity
  - Canonicalization depends on the respective content type
- Optionally, apply transfer encoding
- MIME entity and security information as input to CMS processing
  - Will result in a CMS object
- CMS object is wrapped in MIME

Resulting S/MIME Messages

- Content-type: application/pkcs7-mime
  - PKCS#7: Cryptographic Message Syntax (CMS) Version 1.5 (RFC2315)
    - Current spec of CMS: RFC 3852; ASN.1-encoded cryptographic objects
  - ;smimetype= parameter to indicate what the object is used for
    - enveloped-data, signed-data, compressed-data
    - Concatenation possible: signed-encrypted-data
  - ;name=filename.suffix and ;filename=filename.suffix
    - application/pkcs7-mime: SignedData + EnvelopedData -> .p7m
    - application/pkcs7-mime: CompressedData -> .p7z
    - application/pkcs7-signature: SignedData -> .p7s
    - Filename is arbitrary; no longer than 8 characters
- Signing only: multipart/signed
  - Does not require S/MIME software on the receiver for viewing the contents

Sample Encoding: Encryption

```
Content-Type: application/pkcs7-mime; smime-type=enveloped-data;
            name=smime.p7m
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename=smime.p7m

rfvbnj756tbBghyHhHUujhJhjH77n8HHGT9HG4VQpfyF467GhIGfHfYT6
7n8HHGghyHhHUujhJh4VQpfyF467GhIGfHfYGTrfvbnjT6jH7756tbB9H
f8HHGTrfvhJhjH776tbB9HG4VQbnj7567GhIGfHfYT6ghyHhHUujpfyF4
0GhIGfHfQbnj756YT64V
```

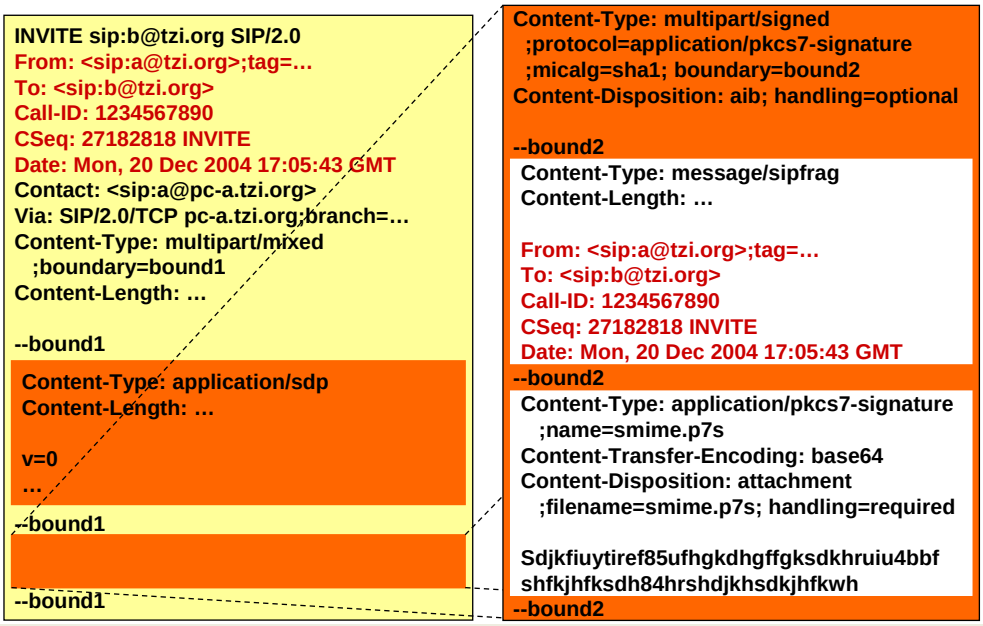

SIP and S/MIME for End-to-End Encryption

- Carry (partial) SIP messages in S/MIME body
  - Content-Type: message/sipfrag (or: message/sip)
  - Apply MIME security to message body
  - Use multiparty/mixed if necessary
    - e.g. to also carry SDP bodies
- Include confidential headers in message body
  - E.g. From:, Contact:, Subject:
- Keep headers required for routing “outside” in the clear
  - Some outer header fields remain visible
    - To, From, Call-ID, Cseq, …
  - Some mandatory headers may be populated with dummy values
  - Some outer header fields may also be changed in transit
    - Request-URI, Via, Record-Route, …
- May also be used for end-to-end authentication

SIP and S/MIME for End-to-End Authentication

- Header fields are included in a signed S/MIME message body
  - Idea similar to encryption: message/sip, message/sipfrag
  - Include enough headers to prevent replay
    - From:, Date: Contact: (To:, Call-ID:, CSeq:)
- UA may sign the message itself: straightforward
  - Workable—but receiver has to obtain UA's certificate
  - Work in progress to locate Certificate servers per domain
    - UA may also use self-signed certificate
  - Option: SIP-based certificate distribution
    - Publication of certificates (PUBLISH)
    - Retrieve certificates, updates, revocations (SUBSCRIBE/NOTIFY)
- Authenticated Identity Body (AIB)
  - Indicated by: Content-Disposition: message/aib

Authenticated Identity Body (AIB)



Third-party Asserted Identity

- Absence of a PKI: remote UA needs to revert to server cert
- Approach: Have an authentication service sign on its behalf
  - Requires (local) trust relationship between UA and auth server
    - Located via Service-Route: or configured preloaded Route:
    - Ideally, direct connection via TLS
    - Server identity established via certificate, UA identity via digest auth
  - Assumes that authentication server's certificate to be known
    - Mostly the case anyway
  - Special case: UA acts as authentication server for itself
- Procedure
  - Cannot use AIB: servers cannot add message bodies
  - Revert to SIP headers
    - Identity: contains signed hash over parts of the message
    - Identity-Info: provides pointer to server certificate
    - Error code to request identity proof: 428 Use Identity
  - Authentication service validates UA's identity locally and adds identity assertion headers

Identity Example

```
INVITE sip:bob@biloxi.exmple.org SIP/2.0
Via: SIP/2.0/TLS pc33.atlanta.example.com;branch=z9hG4bKnashds8
To: Bob <sip:bob@biloxi.example.org>
From: Alice <sip:alice@atlanta.example.com>;tag=1928301774
Call-ID: a84b4c76e66710
CSeq: 314159 INVITE
Max-Forwards: 70
Date: Thu, 21 Feb 2002 13:02:03 GMT
Contact: <sip:alice@pc33.atlanta.example.com>
Content-Type: application/sdp
Content-Length: 147

v=0
o=UserA 2890844526 2890844526 IN IP4 pc33.atlanta.example.com
s=Session SDP
c=IN IP4 pc33.atlanta.example.com
t=0 0
m=audio 49172 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

Identity Example

INVITE sip:bob@biloxi.exmple.org SIP/2.0
Via: S
To: Bo
From:
Call-ID
CSeq:
Max-F
Date:
Contact: <sip:alice@pc33.atlanta.ex
Content-Type: application/sdp
Content-Length: 147

v=0
o=Use
s=Ses
c=IN I
t=0 0
m=audio 49172 RTP/AVP 0
a=rtpmap:0 PCMU/8000

**sip:alice@atlanta.example.com:sip:bob@biloxi.example.org:a84b4c76e66710:314159 INVITE: **
Thu, 21 Feb 2002 13:02:03 GMT:alice@pc33.atlanta.example.com:v=0
o=UserA 2890844526 2890844526 IN IP4 pc33.atlanta.example.com
s=Session SDP
c=IN IP4 pc33.atlanta.example.com
t=0 0
m=audio 49172 RTP/AVP 0
a=rtpmap:0 PCMU/8000

SHA1 w/ RSA Encryption

CyI4+nAkHrH3ntmaxgr01TMxTmtjP7MASwIiNRdupRI1vpkXRvZXx1ja9k0nB2sN
3W+v1PDsy32MaqZi0M5WfEkXxbgTnPYW0jIoK8HMyY1VT7egt0kk4XrKFCHYWGCl
sM9CG4hq+YJZTMaSROoMUBhikVIjnQ8ykeD6UXNOyfI=

Identity Example

INVITE sip:bob@biloxi.exmple.org SIP/2.0
Via: SIP/2.0/TLS pc33.atlanta.example.com;branch=z9hG4bKnashds8
To: Bob <sip:bob@biloxi.example.org>
From: Alice <sip:alice@atlanta.example.com>;tag=1928301774
Call-ID: a84b4c76e66710
CSeq: 314159 INVITE
Max-Forwards: 70
Date: Thu, 21 Feb 2002 13:02:03 GMT
Contact: <sip:alice@pc33.atlanta.example.com>
Identity: CyI4+nAkHrH3ntmaxgr01TMxTmtjP7MASwIiNRdupRI1vpkXRvZXx1ja9k0nB2sN
3W+v1PDsy32MaqZi0M5WfEkXxbgTnPYW0jIoK8HMyY1VT7egt0kk4XrKFCHYWGCl
sM9CG4hq+YJZTMaSROoMUBhikVIjnQ8ykeD6UXNOyfI=
Identity-Info: https://atlanta.example.com/cert
Content-Type: application/sdp
Content-Length: 147

v=0
o=UserA 2890844526 2890844526 IN IP4 pc33.atlanta.example.com
s=Session SDP
c=IN IP4 pc33.atlanta.example.com
t=0 0
m=audio 49172 RTP/AVP 0
a=rtpmap:0 PCMU/8000

Summary

- SIP signaling security readily available (in theory)
  - Lower layer mechanisms (hop-by-hop): IPsec, TLS
    - Encryption
    - Requires trust in SIP server nodes along the path
  - SIP layer authentication (hop-by-hop and end-to-end)
    - Based upon shared secret or PKI
  - SIP layer encryption (S/MIME)
- Issues with SIP signaling security (in practice)
  - PKI available but unsuitable for end users
    - Self-signed certificates and authentication servers as a reasonable remedy
  - But: S/MIME virtually not implemented
    - Rely on identity headers (authentication only; no message content protection)
  - Hence: signaling path is not trusted
    - Make key exchange work across an insecure media path?
    - Move keying into the media path?

Media Security

Media Streams and Security

- Simple protection for plain RTP streams
  - Addresses only encryption
  - Key distribution out of scope
- RTP Packets
  - Encrypted as a whole
  - Sequence number + timestamp initialized to a random value
    - Serves as weak initialization vector
- RTCP Packets
  - Prefixed with a 32 bit random value for each packet sent
- Use Secure RTP (SRTP) for real security

Secure RTP (SRTP)

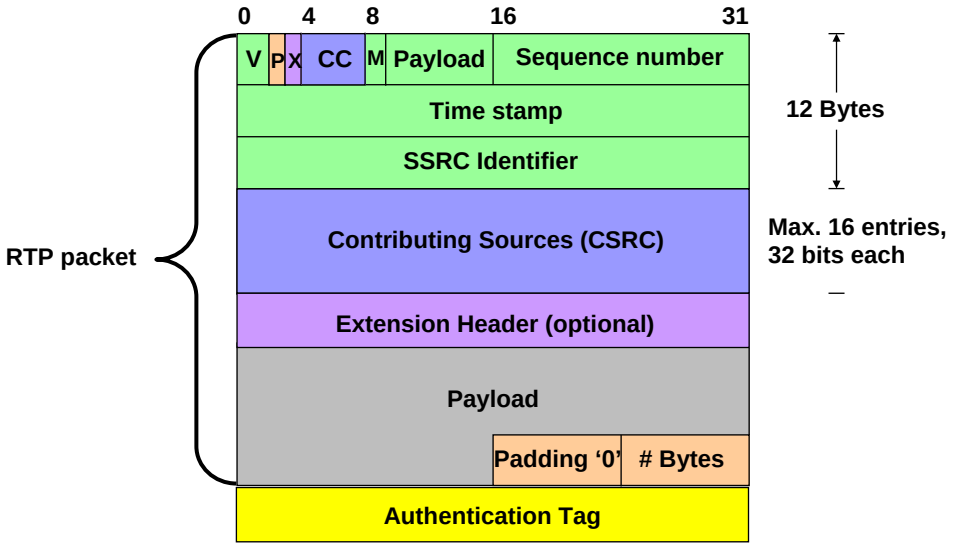
- Motivation
  - RTP itself does not *really* address security
  - TLS not applicable (works only for RTP over TCP/SCTP)
  - IPSEC not always practical
    - Not universally available (kernel deployment issues)
    - Encrypting RTP headers not always a good idea: sabotages header compression
- SRTP features
  - Confidentiality
    - Encryption of the RTP payload
  - Integrity and authenticated communication
    - Authentication tags for the RTP header and the RTP payload
  - Protection against replay attacks
    - Indexing authenticated packets
  - Secures RTP and RTCP

SRTP Design Goals

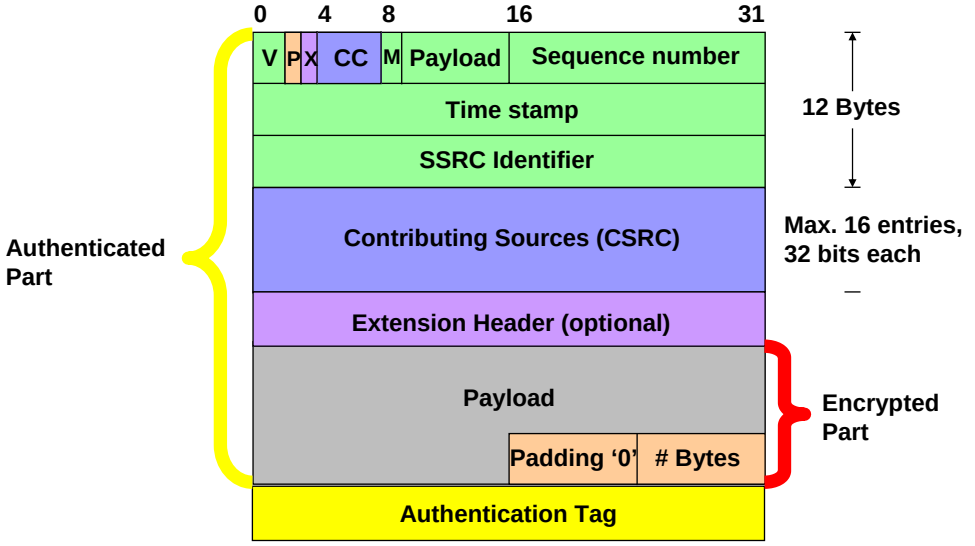
- Low computational cost, small footprint
- Minimal increase in packet size
- No error propagation
  - Bit error in one packet does not affect subsequent packets
- Preservation of RTP/UDP/IP header compression efficiency
  - Leave RTP header unmodified and unencrypted
  - Implemented as a RTP/RTCP profile

Designed with wireless & VoIP requirements in mind

SRTP Packet



SRTP Packet



SRTP Cryptography (1)

- Framework for encryption and authentication
  - Encryption:
    - AES (Used as an additive stream cipher)
  - Authentication:
    - HMAC-SHA1
    - TMMH/16 (Truncated Multi-Modular Hash Function)
      A simple, quick and DSP-friendly hash function
  - Extensible with other algorithms
- Additive stream ciphers
  - Encryption algorithm is used to generate a key stream
    - Input: packet index, session key, salting key
      (adding randomness to packet)
  - Key stream for given packet is added (XOR) to the payload

SRTP Cryptography (2)

- Usage of keys
  - Different keys for encryption and authentication
  - Different keys for RTP and RTCP
- Key derivation
  - The four keys can be derived from a single *master key*
  - Simplifies key exchange
  - Allows for re-keying during a session: New keys can be derived from the master key without explicit out-of-band signalling
  - Maintaining backwards and forward secrecy
  - Generating new keys requires sender-receiver synchronization
    - *Key Derivation Rate* is part of initial set of crypto parameters (as is master key)

SRTP Open Issues

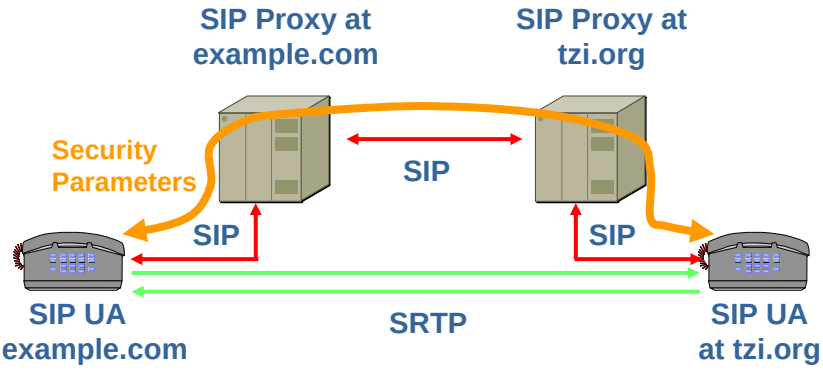
- Data origin authentication
  - SRTP: One authentication key per session (at a time)
  - Authentication of sender works *only* for unicast sessions
  - For multicast, MAC cannot authenticate individual senders
    - Authentication key is shared
    - Can only be used to prove that sender is a valid group member

Multicast Security

- Key management
  - SRTP needs some configuration information beforehand:
    - Algorithms, modes of operation
    - Master and salting keys
    - Session key derivation rates
  - Not conveyed by SRTP

Key Management Extensions for SIP and RTSP

Key Management for MM Conferencing

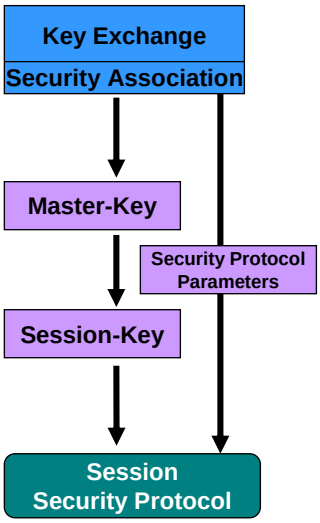


- Main task
  - Securely exchange master keys
  - MUST support end-to-end protection of keying material
- Convey additional information
  - How to derive session keys from master keys
  - Re-keying mechanisms and policies
- SIP considerations
  - Minimize number of round trips and call setup latency
  - Caller must be able to generate session key on its own
  - Time constraints and message size have to be considered

MIKEY: Multimedia Internet KEYing

- Key management scheme for real-time applications
  - Key exchange
  - Specification of cryptographic algorithms and policies
- Three alternative key exchange schemes
  - Pre-shared key
  - Public-key encryption
  - Diffie-Hellman key exchange
- Usable with conference control protocols
  - Mappings to SIP/SDP and RTSP/SDP defined

Key Management Procedure



- Key exchange mechanisms
  - Create security association
  - Exchange master key
- Derivation of session keys
  - Using master key
  - Potentially multiple session keys
- Parameterization of Session Security Protocol
  - Session key
  - Security protocol parameters

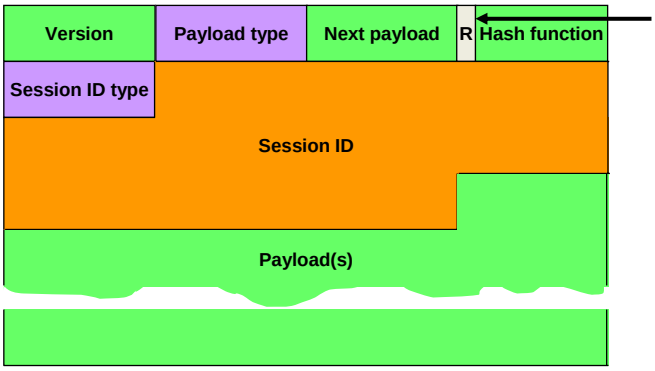
MIKEY Key Exchange Schemes (1)

- Pre-Shared Key
  - Master key is randomly chosen by initiator
  - Encrypted (using a symmetrical algorithm) with the pre-shared key and sent to invitee
  - Key exchange message is authenticated
  - Most efficient variant
    - Symmetric cryptography
    - Small amount of data needs to be sent
  - Not scalable (pre-shared keys have to be known in advance)

MIKEY Key Exchange Schemes (2)

- Public-Key
  - Master key is randomly chosen by initiator
  - Encrypted using invitee's public key
  - Key exchange message is signed (with the private key of initiator) and sent to invitee
  - Scalable but resource demanding
    - Public key cryptography
    - Certificate handling may involve additional interactions
  - Relies on availability of public key infrastructures
- Diffie-Hellman key exchange
  - Two parties compute a shared key
  - No pre-shared keys, no transport of keys
  - Computationally expensive

MIKEY Payload Format



Payload types

- Pre-shared key
- Pre-shared key verification message
- Public key
- Public key verification message
- Diffie-Hellman init
- Diffie-Hellman response
- Error

Mapping MIKEY to SDP

- Extension attribute `key-mgmt`
  - specifies used key management protocol (e.g. MIKEY)
  - contains key management PDU as Base64-encoded data object
- Can be specified at session- or media-level
- Example:

Crypto Session 1
Crypto Session 2

```
a=key-mgmt:mikey uiSDF9sdhs727gheWsnDSJD...
m=audio 49000 RTP/SAVP 98
a=rtpmap:98 AMR/8000
m=video 2232 RTP/SAVP 31
```


Lightweight “Key Exchange”

- MIKEY is quite complex to implement
  - Optimized solution for protected signaling paths only developed
- SDP Security Description (“sdescription”) attribute
  - Carries keying information in SDP
    - For one-way configuration or key exchange
  - Replace the underspecified k= attribute by something useful
  - Session or media level attribute: a=crypto

```
a=crypto:1 AES_CM_128_HMAC_SHA1_80 inline:PS1uQCVee…txaVBR|2^20|1:32
```

Tag (for O/A) Ciphersuite (auth + crypt) Keying parameters: location (+value)

- Key info text represents specific keying material, e.g. for SRTP
  - Key and salt concatenated, key lifetime (in packets), master key identifier : length in SRTP
- Followed by optional session parameters (inherited from SRTP)
  - Key derivation rate ($1 – 2^{24}$), no encryption for SRTP/SRTCP, no auth for SRTP
  - FEC parameters, window size hints (to protect against replay attacks)

SDP Example

```
v=0
o=jdoe 2890844526 2890842807 IN IP4 10.47.16.5
s=-
c=IN IP4 10.47.16.5/127
t=0 0
m=video 51372 RTP/SAVP 31
a=crypto:1 AES_CM_128_HMAC_SHA1_80 \
    inline:d0RmdmcmVCspeEc3QGZiNWpVLFJhQX1cfHAwJSoj|2^20|1:32
m=audio 49170 RTP/SAVP 0
a=crypto:1 AES_CM_128_HMAC_SHA1_32   \
     inline:NzB4d1BINUAvLEw6UzF3WSJ+PSdFcGdUJShpX1Zj|2^20|1:32
a=crypto:2 AES_CM_128_HMAC_SHA1_80   \
     inline:udhf9843yiuhrfkuyf834yhkwhkjew3uhkjhdew8kjhwhkw|2^20|1:32
m=audio 49172 RTP/SAVP 0
```


Offer / Answer Behavior

- Offerers proposes one or more crypto attributes per media stream to be protected
  - Keying material included is meant for transmission from offerer to answerer
- Answerer chooses exactly one cipher suite
  - Identified by means of the initial tag value
  - Includes keying material used for transmission to offerer in response
- If no cipher suite is acceptable to the answerer
  - Reject request
  - May include simcaps to indicate preferred cipher suites
- Both sides to initialize their respective SRTP contexts
- Similar for non-offer/answer exchanges (e.g. RTSP)

Media Security Today

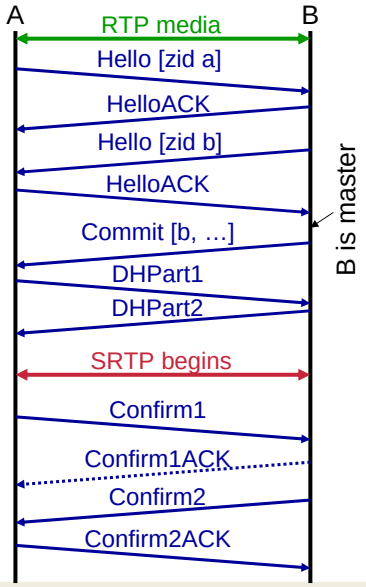
- Limited support in products and services
- Very limited actual deployment
- Dependency on signaling protection
  - SDescriptions depend on it
    - S/MIME not implemented in endpoints
  - MIKEY self-contained protection not widely implemented either
- May not work with service providers
  - If end-to-end encryption is needed because they may want to peek
  - If end-to-end integrity protection is needed because they may want to alter (or are not capable of processing multipart MIME and S/MIME bodies)
- Yet people want media security
- Solutions exploring alternative paths coming up

Example: In-band Key Exchange with ZRTP

- No dependency on signaling or keying infrastructure
  - Yet these may improve security
- Media protection
  - Three shared keys
    - Ephemeral per call key (negotiated via Diffie-Hellman, destroyed after call)
    - Cached key between calls (some key parts retained, linking one call to the next)
    - Externally fed key from signaling
  - Works with the ephemeral key alone
    - Non-available keys will be ignored during initial negotiation phase
- Mapping to the RTP media stream
  - New RTP extension header
  - Uses RTP “No-Op” payload format
- Media stream signaling may be augmented/backed up in SDP
  - Indicate ZRTP capability
  - Exchange some keying material (additional protection)

Keying with ZRTP

- Initial determination of ZRTP capabilities
  - Exchange of endpoint id (zid) for indexing cached keys
  - Independent Hello messages + acks
  - Commit indicates common parameter set
- Commit and Diffie-Hellman
  - Generation of shared keys
  - Multi-stream optimization (D-H only once)
- Detecting MITM attacks
  - Text exchanged by the protocol is diplayed to and read by one user; checked by the other
- Move back to cleartext RTP operation
  - GoClear, ClearACK
- Reliability
  - Binary exponential backoff for messages
  - Different timers for Hello and the rest
- Media streams: regular SRTP



Example: In-band Key Exchange with ZRTP

- Hyped to some extent
  - Appears to do the job, though
- Fundamental architectural issue: RTP for signaling
  - Could use RTCP instead (timeliness?)
- Practical issues
  - Does not work with shared keys for multiparty conferencing
  - Cumbersome(?) initial authentication procedure
  - Backwards compatibility may rather be theoretical
  - RTP no-op payload has never been finished (expired I-D)
  - Patents…

Example: RTP over DTLS

- Minimal dependency on signaling or keying infrastructure
  - Exchanging public-key fingerprints in SDP
- Media protection: DTLS
  - Initial DTLS exchange similar to TLS
    - Need to explicitly deal with packet losses
  - Media packet encryption using TLS records
    - Avoid loss of crypto context in case packets get lost
    - Borrow explicit state signaling from IPsec ESP
- Mapping to the RTP media stream
  - RTP over DTLS
  - Alternative: SRTP
- Architectural integrity preserved