RFC 4895bis: SCTP Authentication

draft-ietf-tsvwg-rfc4895-bis-02

Michael Tüxen (tuexen@fh-muenster.de)
Randall Stewart (randall@lakerest.net)
Peter Lei (peterlei@netflix.com)
Hannes Tschofenig (hannes.tschofenig@gmx.net)

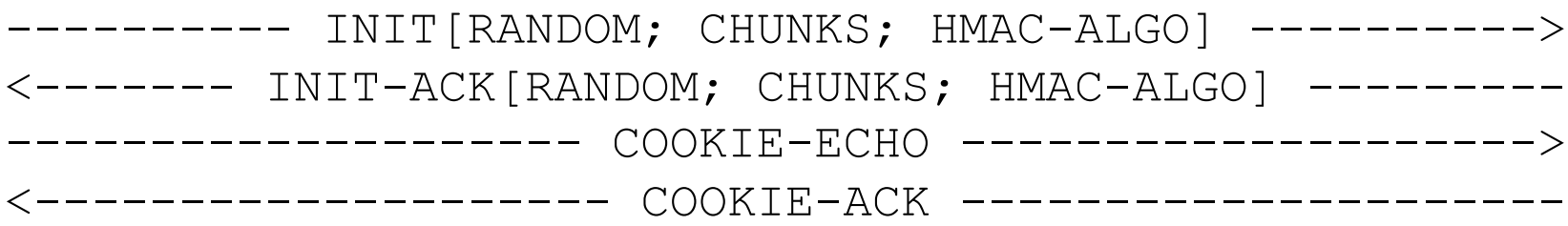
Scope

- Incorporate relevant changes from draft-nagesh-sctp-auth-4895bis-00
- Address two security issues reported by Ericsson:
  - Use direction dependent keys to mitigate reflection attacks.
  - Don’t use different HMAC algorithms with the same keys.
- Generalize HMAC to MAC.
- Add more algorithms, potentially retire HMAC-SHA-1.
- Add socket API considerations for improved control of the SCTP AUTH usage.

Status

- draft-tuexen-tsvwg-rfc4895-bis-00
  Submit RFC 4895 as an ID.
- draft-tuexen-tsvwg-rfc4895-bis-01
  Update to xmlv3.
- draft-tuexen-tsvwg-rfc4895-bis-02
  Wordsmithing and updating references.
- draft-tuexen-tsvwg-rfc4895-bis-03
  Minor editorial change.
- draft-tuexen-tsvwg-rfc4895-bis-04
  Add socket API related updates required for DTLS/SCTP.
- draft-tuexen-tsvwg-rfc4895-bis-05
  Remove ekr from list of authors, improve socket API.
- draft-tuexen-tsvwg-rfc4895-bis-06
  Update Acknowledgements.
- draft-tuexen-tsvwg-rfc4895-bis-00
  Same as above.
- draft-tuexen-tsvwg-rfc4895-bis-01
  Incorporate draft-nagesh-sctp-auth-4895bis-00, editorial changes, update IANA section.
- draft-tuexen-tsvwg-rfc4895-bis-02
  Introduce directional keys.

SCTP AUTH Handshake



Key Management (1)

- Ensure that the RANDOM parameter values used by both endpoints are different.
  - In client/server scenarios this is easy to accomplish.
  - In case of INIT collision the sending of ABORTs are required, if both sides chose the same 32 byte random number. However, this is very rare.
- Local and remote key vector
  - Local key vector based on parameters sent.
  - Remote key vector based on parameters received.
  - key vector = RANDOM || CHUNKS || HMAC ALGO
  - Local key vector and remote key vector are different.

Key Management (2)

- send context = local key vector || remote key vector
- receive context = remote key vector|| local key vector
- send key = KDF(shared_key, send_context, key len)
- receive key = KDF(shared_key, receive_context, key len)
- Using KDF as described in RFC 5926, Section 3.1.
- shared_key is controlled by the API and might be empty!
- Provided in the document based on HMAC-SHA-256.

Backwards Compatibility

- Must be enabled by the upper layer.
- Non directional keys are only used if no directional keys are supported by the peer.
- send key and receive key are the concatenation of
  - min(local key vector, remote key vector)
  - shared key
  - max(local key vector, remote key vector)

Computing the AUTH chunk

- When sending, compute MAC(send_key, chunks) and put it into the value of the AUTH chunk.
- When receiving, compute MAC(receive_key, chunks) and verify that this is the chunk value of the AUTH chunk.

Next Steps

- Double checking: out of scope is improving the replay protection for chunks with sequence numbers:
  - DATA and SACK/NR-SACK
  - I-DATA and SACK/NR-SACK
  - ASCONF and ASCONF ACK
  - RE CONFIG
- Use a formula-based description instead of a text based one.
- Generalize HMAC to MAC.
- Add more algorithms. Which ones?
- Address comments already received and all upcoming comments.