AEAD Limits for OSCORE

CORE, IETF 109, November 2020

AEAD Limits

- CFRG works on equations to calculate AEAD limits when the key needs to be changed.
  - Limits based on best known bounds for adversary probability of forgery of a single packet or distinguishability from a random string.
  - https://tools.ietf.org/html/draft-irtf-cfrg-aead-limits

- The important equations for AEAD_AES_128_CCM_8 (AES-CCM-16-64-128, AES-CCM-64-64-128) are:

- Confidentiality Limit: $\mathbf{q} \leq \text{sqrt}((\mathbf{p} \cdot 2^{126}) / \mathbf{l}^2)$
- Integrity Limit: $\mathbf{v} \cdot 2^{64} + (2\mathbf{l} \cdot (\mathbf{v} + \mathbf{q}))^2 \leq \mathbf{p} \cdot 2^{128}$

AEAD limit based on
- **q** Number of packets to encrypt
- **v** Number of unsuccessful invocations of AEAD

- where
  - **q** = Number of protected messages (AEAD encryption invocations)
  - **v** = Number of attacker forgery attempts (failed AEAD decryption invocations)
  - **p** = Upper bound for adversary attack success probability
  - **l** = Input length (plaintext + additional data) in blocks

Note that it is always true that $\mathbf{v} \leq \mathbf{p} \cdot 2^{64}$

AEAD Limits

- TLS 1.3, DTLS 1.3, and QUIC have adopted strict limits based on the same equations. Having strict limits is not a problem if re-keying is easy.

- For example, DTLS 1.3 uses the parameters:
  - $\mathbf{p}_q = 2^{-60}$ $\mathbf{p}_v = 2^{-57}$, (upper bound for security level, different values can be chosen)
  - $\mathbf{l} = 2^{10}$ (2^{14} bytes = 16 kB) (maximum length of TLS record layer)
- Resulting in the following limits:
  - **CCM**: Limit for CCM set to $\mathbf{q} = 2^{23}$ and $\mathbf{v} = 2^{23.5}$
  - **CCM_8**: MUST NOT be used **without additional safeguards against forgery.**
- TLS 1.3 has $\mathbf{v} = 1$ so CCM_8 can be used.

- **What assumptions are relevant for the CoAP IoT setting:**
  - The CoAP IoT setting in general (where CCM_8 is commonly used)?

AEAD Limits Considerations

- Forgery attacks are a greater threat that distinguishing attacks. For CCM_8, the limits for number of failed decryptions **v** is the limiting factor. Limits smaller than 2^{40} likely needed also for other AEAD.
- Some IoT systems might accept a higher forgery probability than (D)TLS in general.
- If the attacker uses a high-speed broadband connection, a lot of forgeries can be sent quickly. On constrained IoT systems, bandwidth is restricted, and it would take long time to send forged messages.
- Any DoS attack resulting from low limits on **v** should be compared to other DoS attacks on the system. If there are even easier ways to reduce the availability of a system, a low limit on **v** does not matter.
- Replay attacks will never invoke AEAD decryption, thus not contribute to **v**.
- For most constrained IoT systems, input lengths are much smaller than 16 kB packets used for (D)TLS. Need to specify a use a maximum input length.
- Likely possible to use limits based on the maximum seen length, which would allow better limits in a non-attack scenario.
- A forgery itself is not a huge problem if the plaintext is not processed. Might not need to "close connection". Note that the adversary gains some information with each successful forgery.
- Forgeries might look like random strings and may be discarded by CoAP. But it should probably be assumed that the adversary can do chosen prefix or postfix.

Mail thread in LAKE: https://mailarchive.ietf.org/arch/browse/lake/

OSCORE considerations

- “The maximum Sender Sequence Number is algorithm dependent (see Section 12) and SHALL be less than 2^40.” (Section 7.2.1 of RFC 8613)
  - CCM_8 limit will apply before 2^{40} (≈ 10^{12} ≈ 3587 messages per second for 10 years)
    - Guidelines for setting max SSN are needed

- Current schemes for rekeying OSCORE
  - Appendix B.2
  - EDHOC asymmetric key authenticated DH
  - OSCORE profile of ACE
  - [provisioning of master secret . . .]

Potential actions

- General problem
  - There is a need to investigate CoAP IoT relevant limits
  - Parameters p and l needs to be determined
  - Trade-off between q and v
  - Discussion started in LAKE
  - For what protocols is this discussion relevant?

- Specific for OSCORE: potential new draft updating RFC 8613
  - Introduce count of q and v
    - Client already counts q
  - How to configure maximum q and v
  - Specify actions to take when max q or max v is reached (rekey)
  - Implementation guidance

- Corresponding adaptations needed for group OSCORE
- Maybe good idea to start investigating a lightweight symmetric-key rekeying mechanism?