AEAD key usage limits in OSCORE

draft-hoeglund-core-oscore-key-limits-00

Rikard Höglund, RISE
Marco Tiloca, RISE

IETF 110, CoRE WG, March 8th, 2021

Problem Overview

- OSCORE uses AEAD algorithms to provide security properties
  - Confidentiality
  - Integrity

- Forgery attack against AEAD algorithms
  - Adversary may break the security properties of the AEAD algorithm
  - See **draft-irtf-cfrg-aead-limits-01**

- Need to describe relevant limits for OSCORE
  - How the forgery attack and the limits affect OSCORE
  - Necessary steps to take during message processing
  - What to do if the limits are exceeded

Limits on key usage

- What you need to count
  - ‘q’: the number of messages protected with a specific key, i.e. the number of times the algorithm has been invoked to encrypt data with that key
  - ‘v’: the number of forgery attempts that have been made against a specific key, i.e. the amount of failed decryptions that has been done with the algorithm for that key

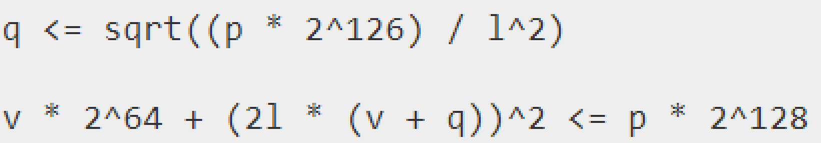
- When a peer uses OSCORE
  - The key used to protect outgoing messages is its Sender Key
  - The key used to decrypt and verify incoming messages is its Recipient Key

- Relevant counters for OSCORE
  - Counting number of times Sender Key has been used for encryption (q value)
  - Counting number of times Recipient Key has been used for failed decryption (v value)
  - Counters and limits can be added to the OSCORE Security Context

Limits for ‘q’ and ‘v’

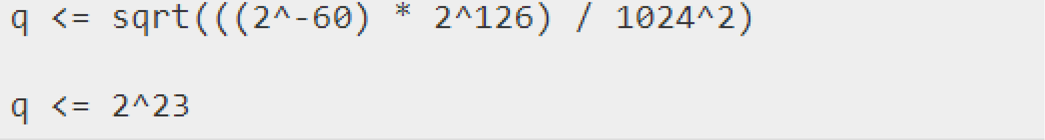
- Formula for limits for AES-CCM-16-64-128

See **draft-irtf-cfrg-aead-limits-01**

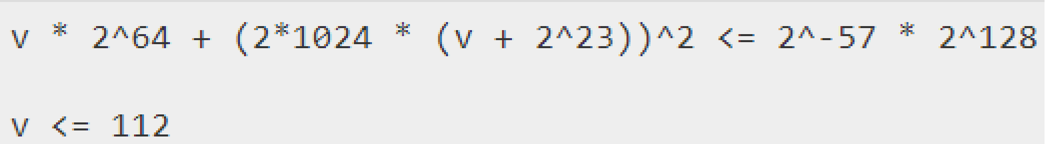


```
q <= sqrt((p * 2^126) / l^2)

v * 2^64 + (2l * (v + q))^2 <= p * 2^128
```

- Depends on assumptions for the p probability values
  - Considering the values $p_q = 2^{-60}$ and $p_v = 2^{-57}$
  - Same values used in [I-D.ietf-tls-dtls13]
- Exact limits calculated



```
q <= sqrt(((2^-60) * 2^126) / 1024^2)

q <= 2^23
```



```
v * 2^64 + (2*1024 * (v + 2^23))^2 <= 2^-57 * 2^128

v <= 112
```

‘q’: encryptions with a key

‘v’: failed decryptions with a key

New information in OSCORE Context

- Sender Context
  - ‘count_q’: Initialized to 0; incremented after encrypting with the Sender Key
  - ‘limit_q’: Limit for 'count_q'

- Recipient Context
  - ‘count_v’: Initialized to 0; incremented upon a failed decryption with the Recipient Key
  - ‘limit_v’: Limit for 'count_v'

- If 'limit_v' or 'limit_q' are reached
  - The nodes must stop using that Security Context and must rekey

- This updates RFC 8613

Methods for rekeying OSCORE

- Reasoned overview of available methods

- Appendix B.2 of OSCORE (RFC 8613)
  - https://datatracker.ietf.org/doc/html/rfc8613#appendix-B.2

- OSCORE Profile of ACE
  - https://datatracker.ietf.org/doc/draft-ietf-ace-oscore-profile/

- EDHOC protocol
  - https://datatracker.ietf.org/doc/draft-ietf-lake-edhoc/

- Manual re-configuration (not practical)

Open Points

- Best location to provide the limits for more algorithms?

- Consider the assumed probabilities for p_q and p_v
  - Are the same values relevant for OSCORE as [I-D.ietf-tls-dtls13] defines?

- Adding an expiration timer to the OSCORE Security Context
  - An “expires in” element can be added to the OSCORE Security Context, similar to the ACE ‘exi’ parameter. This would hold the lifetime of the Context in seconds
  - Is there a value in doing this from a security perspective?

Summary and next steps

- AEAD limits and their impact on OSCORE
  - Introduce counting of ‘q’ and ‘v’ values for OSCORE
  - Stop and rekey if the limits are reached
  - Overview of current rekeying methods

- Next steps
  - Cover more AEAD algorithms
  - Synchronize with other ongoing work
    - EDHOC in the LAKE WG
    - Broader relevance (e.g. (D)TLS, QUIC …) - Next presentation from John
  - Optimizations for constrained devices and implementation guidelines

Thank you!

Comments/questions?

https://gitlab.com/rikard-sics/draft-hoeglund-oscore-rekeying-limits/