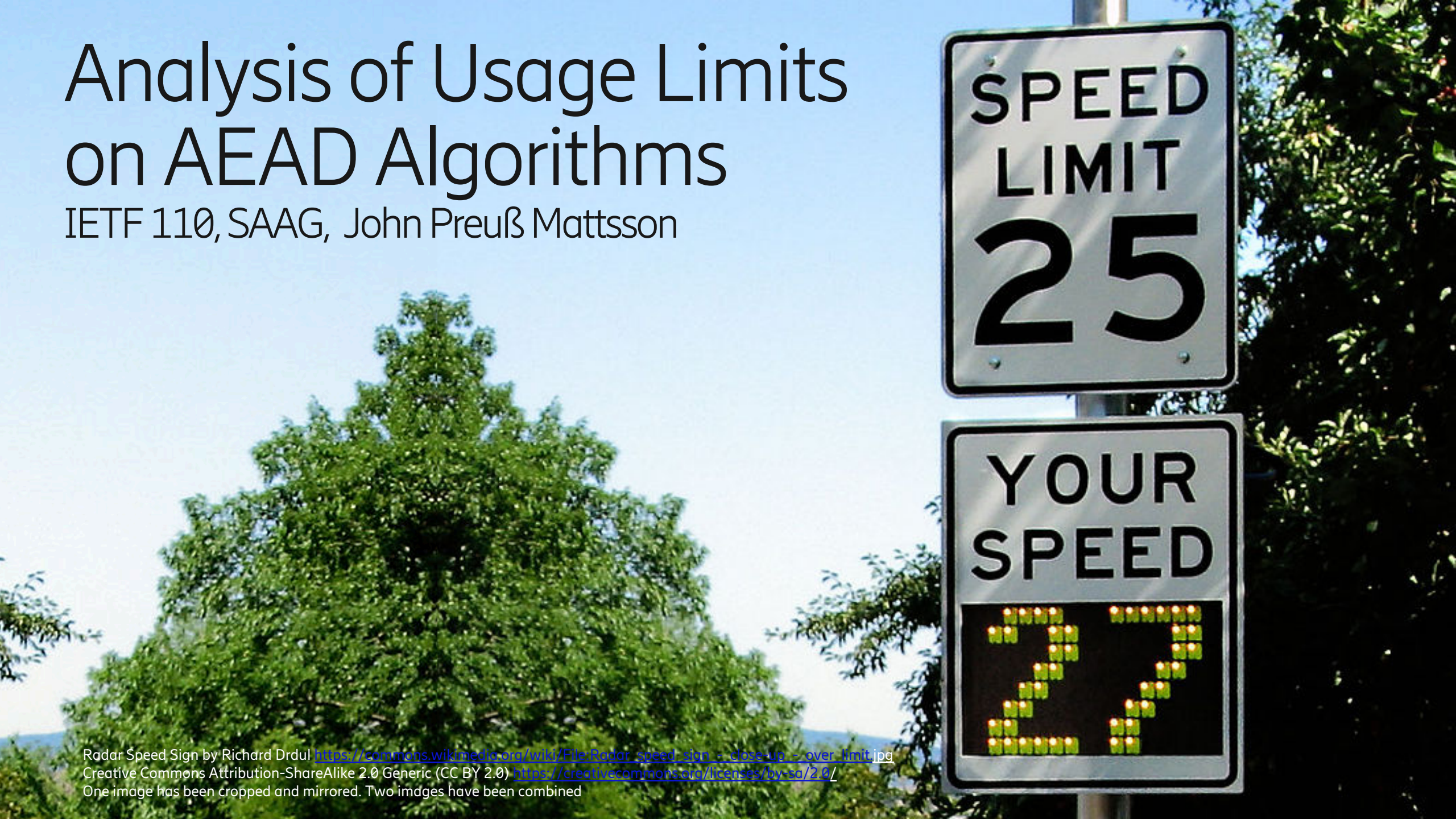
Analysis of Usage Limits on AEAD Algorithms

IETF 110, SAAG, John Preuß Mattsson



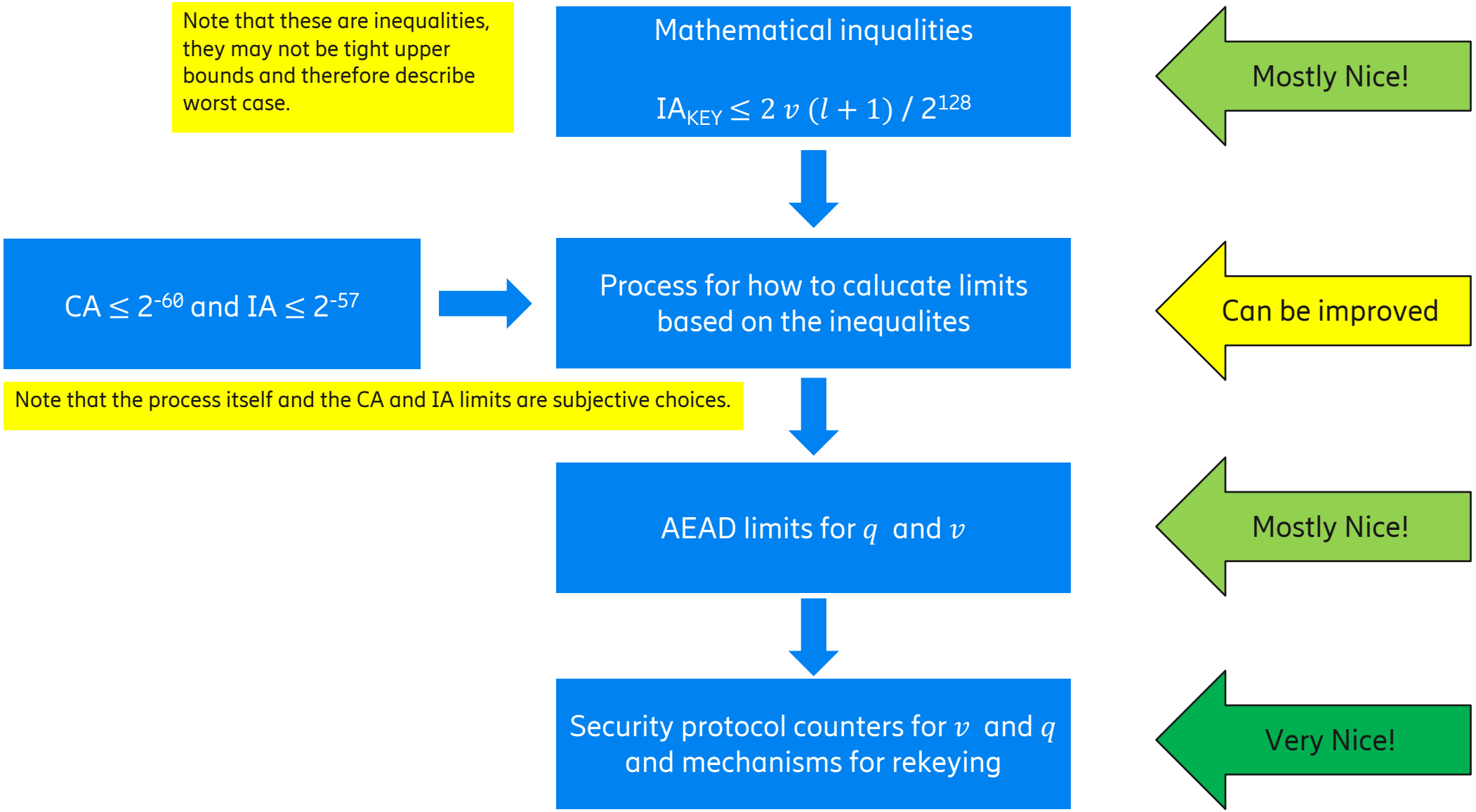
Usage Limits on AEAD Algorithms Overview

- AEAD limits have recently been discussed for TLS, DTLS, QUIC, and OSCORE:
  - https://datatracker.ietf.org/doc/draft-irtf-cfrg-aead-limits/
  - https://datatracker.ietf.org/doc/rfc8446/
  - https://datatracker.ietf.org/doc/draft-ietf-tls-rfc8446bis/
  - https://datatracker.ietf.org/doc/draft-ietf-tls-dtls13/
  - https://datatracker.ietf.org/doc/draft-ietf-quic-tls/
  - https://datatracker.ietf.org/doc/draft-hoeglund-core-oscore-key-limits/

- Use mathematical single-key and multi-key inequalities for CA (Confidentiality Advantage) and IA (Integrity Advantage) to calculate limits for:
  - q the number of protected messages (AEAD encryption invocations)
  - v the number of attacker forgery attempts (failed AEAD decryption invocations)
  - l the maximum length of each message (in blocks)
- Rekeying must be done before the q and v limits are met.

- If done correctly, rekeying gives also forward secrecy, which limits the impact of key compromise.
  - Rekeying with (EC)DHE gives additional protection by forcing attackers to keep being active.

The AEAD limits work consists of 4 steps



Analysis of the inequalities (single-key)

- The bounds for AES [AEBounds] [CCM-ANALYSIS] assumes that AES is a random permutation. This is reflected in the the CA ≈ $q^2\ l$ / 2^{128} inequalities.

- The bounds for ChaCha20 [ChaCha20Poly1305Bounds] [AEBounds] assumes that ChaCha20 is a random function. This in not reflected in the the CA ≤ $v\ l$ / 2^{103} inequality used in DTLS, TLS, QUIC, and CFRG. The inequality is taken purely from the integrity degradation and does the ChaCha20 stream cipher injustice by suggesting it provides much worse confidentiality than AES-CTR for small q, which is not true.

- We recommend treating ChaCha20 as a random function similar to the way AES is treated as a random permutation, which would imply CA = 0 for ChaCha20.

- It should be noted that CA and IA are practically very different:
  - CA is typically used for an offline attack, while IA is typically used for online attacks.
  - IA is directly related to a practical attack (forgery) while CA is more theoretical (distinguishing) and might not be directly related with any practical attack.

Analysis of the suggested "calculating limits" step

- Current suggested process is to set limit for CA and IA per key and based on the inequalities calculate limits for q and v. TLS, DTLS and QUIC use approximately CA ≤ 2^{-60} and IA ≤ 2^{-57}.

- This mostly leads to practically usable limits that improves security. The process do however also give strange and misleading results.

- The suggested process lead to the recommendation that the ideal MAC needs to be rekeyed. This does not make sense and does of course not improve security. The suggested process suggests that the ideal 64-bit MAC and CCM_8 needs to be rekeyed extremely often. DTLS 1.3 more of less forbids CCM_8 due to the rekeying requirement. For low v and q, CCM_8 behaves very close to the ideal 64-bit MAC.

- The suggested process misleadingly gives the idea that frequent rekeying can keep security high.
  - With frequent rekeying, CA and IA per key can be kept almost arbitrary low. E.g. GCM IA ≤ 2^{-117}
  - While CA for the whole connection (with a max number of packets) is bounded, IA for the whole connection is unbounded.
  - For some of the advantages (AES-GCM IA, ChaCha20-Poly1305 CA and IA) rekeying it not shown to increase security or lower advantages for the whole connection.

Linear and superlinear inequalities

It is easy to see from the inequalities if rekeying improves security for the connection. Superlinear equations e.g. $(v + q)^2$ needs to be rekeyed before the security level gets to low.

— ChaCha20-Poly1305 IA	$\mathrm{IA} \le v \cdot l / 2^{103}$	Rekeying does not improve connection advantage
— AES-GCM IA	$\mathrm{IA} \le v \cdot l / 2^{127}$	Rekeying does not improve connection advantage
— AES-CCM IA	$\mathrm{IA} \le v / 2^{128} + l^2 (v + q)^2 / 2^{126}$	Rekeying does improve connection advantage*
— AES-CCM_8 IA	$\mathrm{IA} \le v / 2^{64} + l^2 (v + q)^2 / 2^{126}$	Rekeying does improve connection advantage*
— ChaCha20-Poly1305 CA	$\mathrm{CA} = 0$	Rekeying does not improve connection advantage
— AES-GCM CA	$\mathrm{CA} \le (l + q)^2 / 2^{129}$	Rekeying does improve connection advantage
— AES-CCM CA	$\mathrm{CA} \le l^2 \cdot q^2 / 2^{126}$	Rekeying does improve connection advantage

Analysis of the suggested "calculating limits" step

- When attacker cost measured in encryption or decryption queries, it seem like rekeying might a bit surprisingly increase IA for the whole connection.

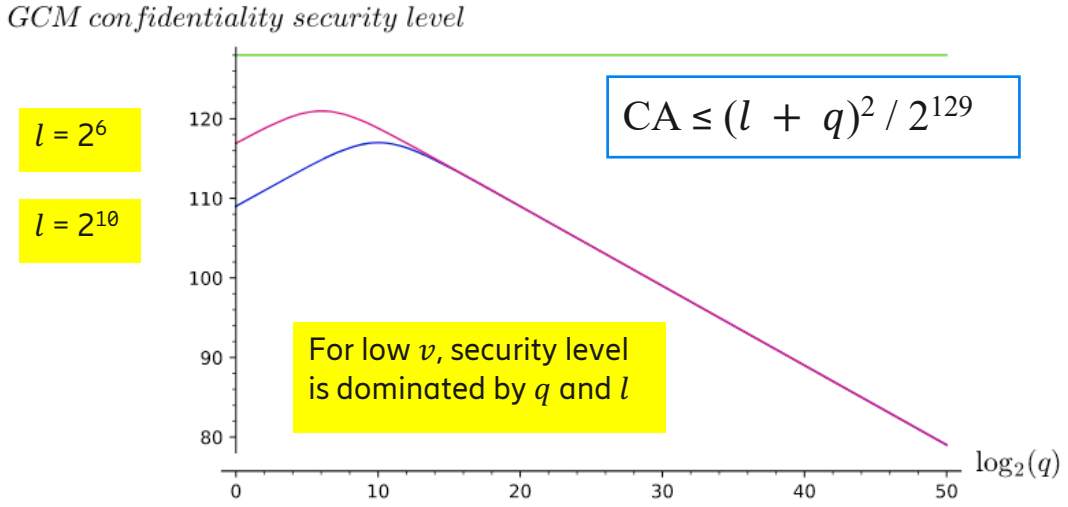
- The limits $q = 2^{23}$ and $l = 2^{10}$ makes CCM_8 deviate from an ideal MAC also for small values of v. An application/protocol using CCM_8 should probably chose smaller values, e.g. $q = 2^{20}$ and $l = 2^{8}$. With these values CCM_8 performs like an ideal 64-bit MAC to until $v = 2^{35}$.

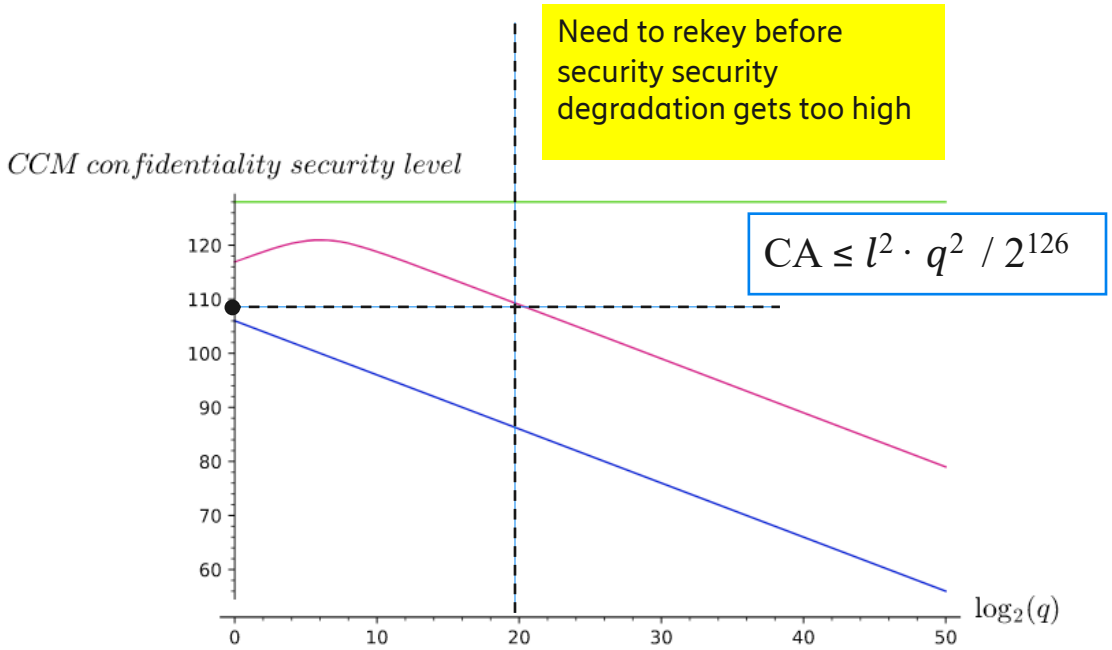
- The process of limiting CA and IA per key does not seem like the right thing to do for a security protocol where each connection has many keys, communication between two parties can use many connections, and adversaries can often trick the parties to tear down the old connection and set up a new connection.

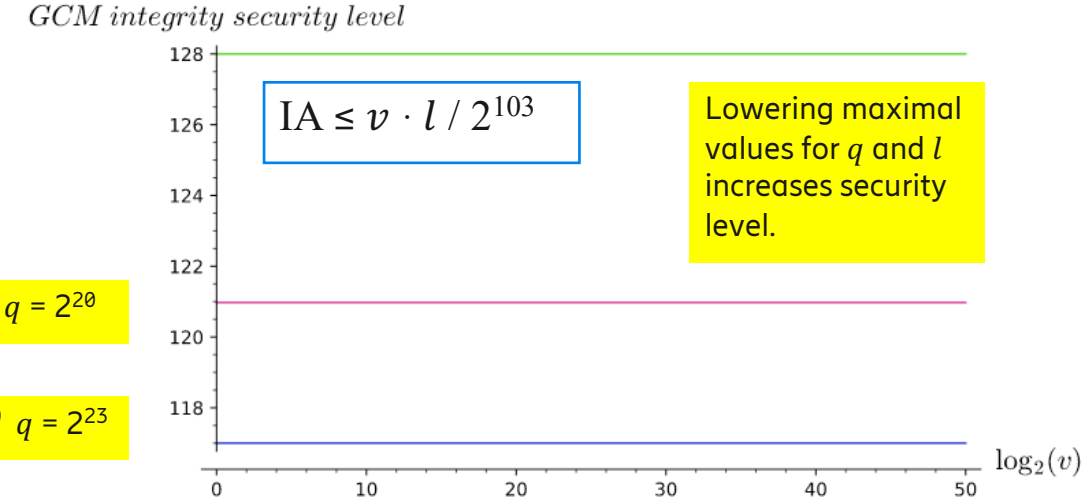
- An easier process may be to calculate the security level = min(attacker cost / advantage) and put limits on the security level for distinguishing and forgery. The security level is minimized over all possible adversaries. This seems to avoid the misleading results of the currently suggested process (rekeying ideal MAC, rekeying gives arbitrary small CA and IA per key, and rekeying increases IA for the connection).

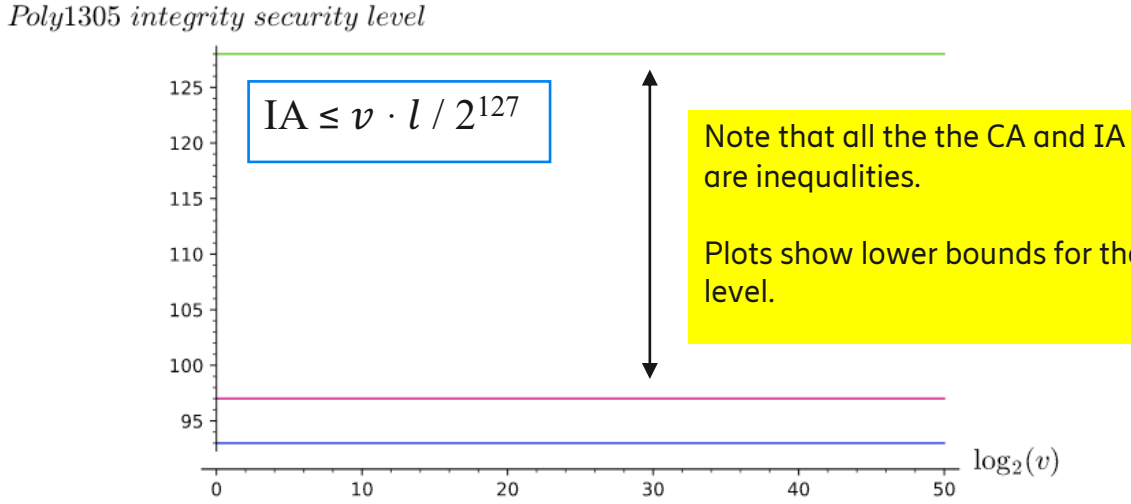
- The suggested process has lead to quite arbitrary but practically useful limits for (D)TLS and QUIC
  - People are maybe taking the specific process and exact limits a bit too serious.
  - We don't think the process as currently specified should be an IETF/IRTF recommendation.

Lower bounds for security levels (in bits)



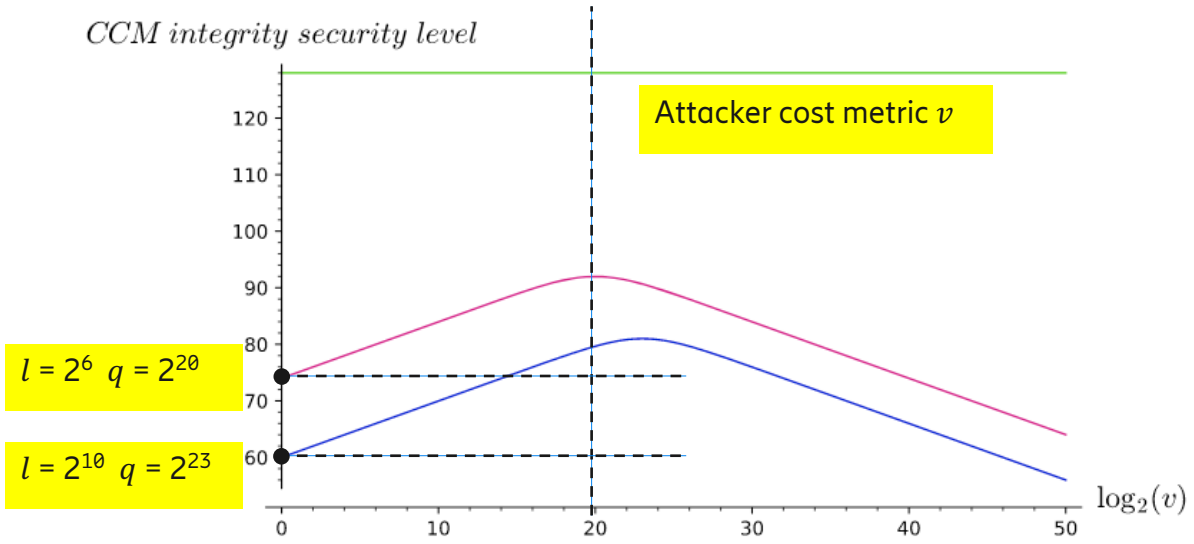


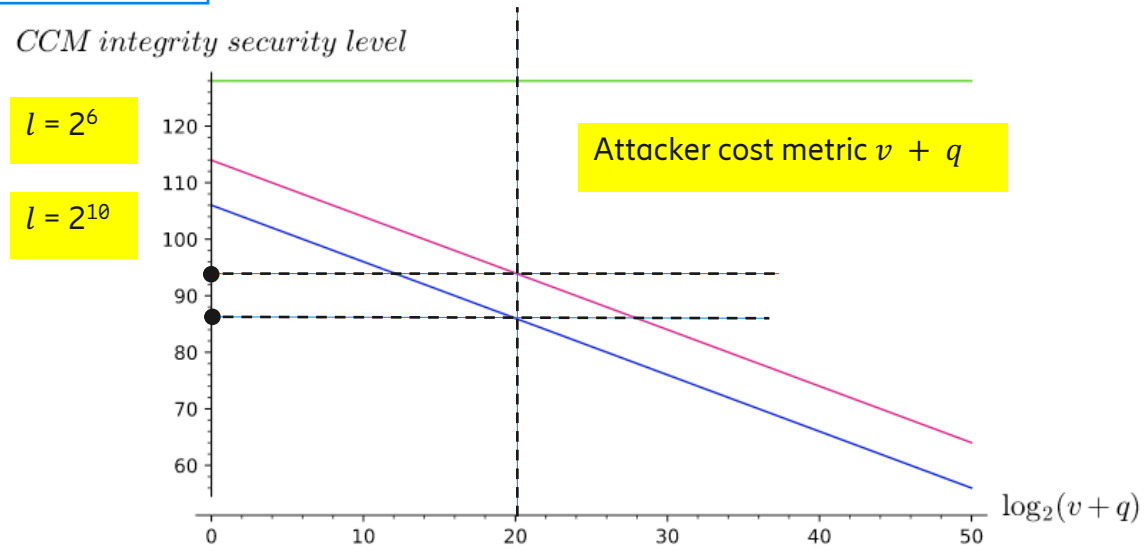


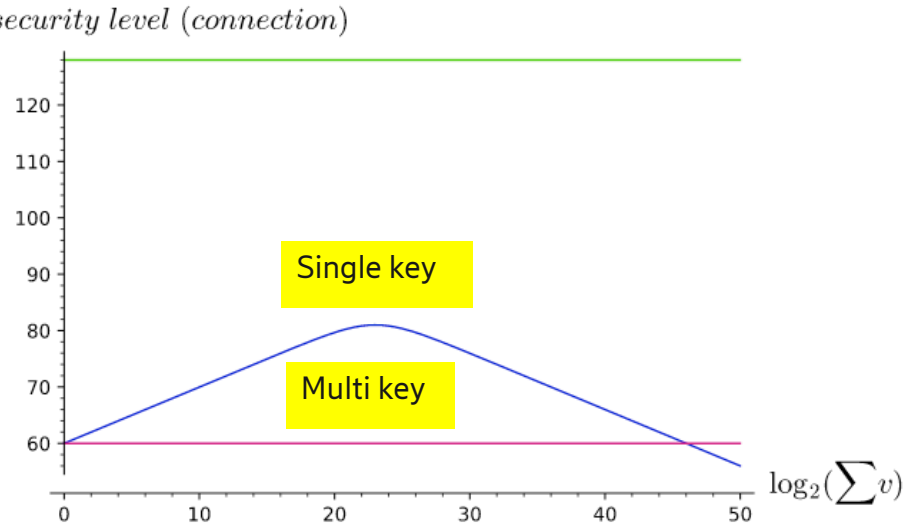


Lower bounds CCM integrity security level (in bits)

$$\mathrm{IA} \le v / 2^{128} + l^2 (v + q)^2 / 2^{126}$$







Security level is the minimized over all allowed v

$$\mathrm{IA}_{\mathrm{con}} \le (v + 2^{23})^2 / 2^{106}$$

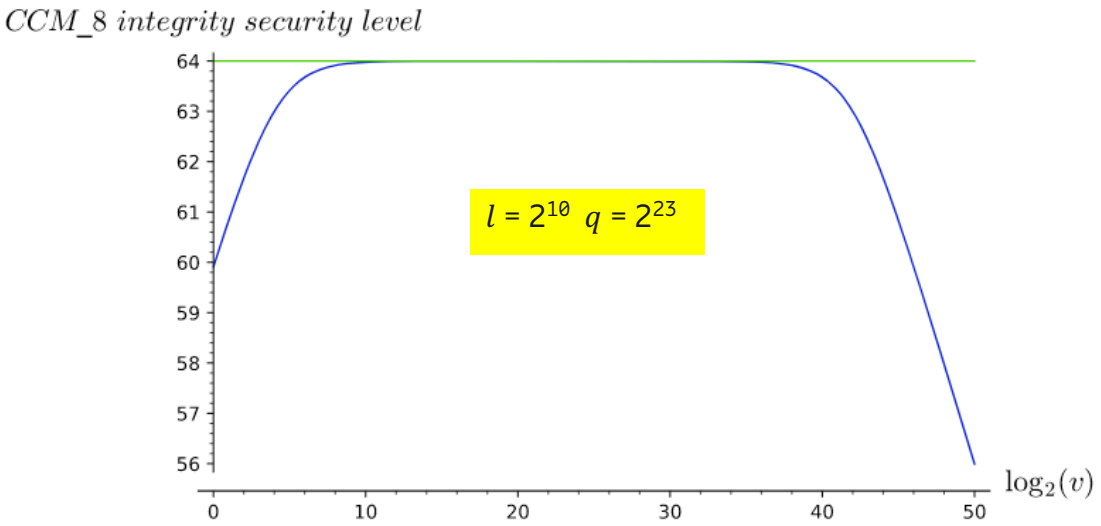
$$\mathrm{IA}_{\mathrm{con}} \le v \cdot (1 + 2^{23})^2 / 2^{106}$$

If attacker cost is measured in only decryption queries or only encryption queries, rekeying can give higher advantage per connection (but not a lower security level).

An optimal attacker do one forgery per key.

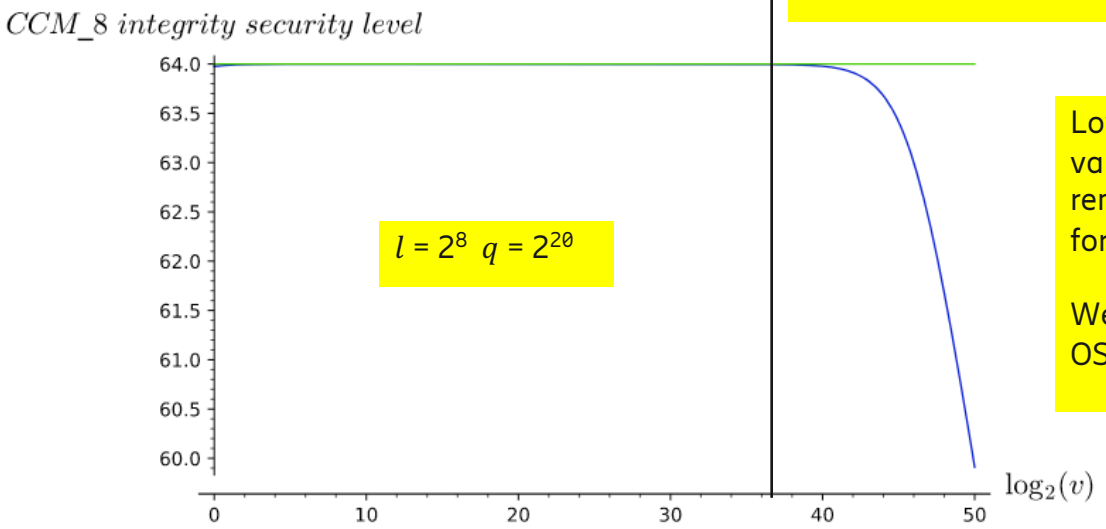
Lower bounds CCM_8 integrity security level (in bits)

$$\text{IA} \leq v / 2^{64} + l^2 (v + q)^2 / 2^{126}$$



$l = 2^{10}$ $q = 2^{23}$

For low v, security level is dominated by q and l



$l = 2^{8}$ $q = 2^{20}$

Need to rekey before security degradation gets too high

Lowering maximal values for q and l removes deviation for low v.

We recommend OSCORE to do so

CCM_8 behaves very much like the ideal 64-bit MAC.

Suggestions for (OS)CORE and IoT

- Security protocol counters for v and q and mechanisms for rekeying are necessary.
- Frequent rekeying with forward secrecy limits the impact of key compromise, this might be even more important that the AEAD advantages.
- Use CA = `0` for ChaCha20-Poly1305. Rekeying for linear inequalities (ChaCha20-Poly1305 CA/IA and AES-GCM IA) does not improve security level or the advantage for the connection.
- CCM_8 is a very close to a perfect 64-bit MAC for low values of q and v. No problem at all to continue using CCM_8 as long as 64-bit forgery probability is acceptable.
- 64-bit forgery probability is definitely acceptable in constrained IoT. To break 64-bit security against online brute force an attacker would on average have to send 4.3 billion messages per second for 68 years, which is infeasible in constrained IoT radio technologies.
- Consider using smaller limits than $q = 2^{23}$ and $l = 2^{10}$, this improves security for AES-GCM CA and AES-CCM(_8) CA/IA.
- The current process and limits should be taken with a pinch of salt. Suggestions for (OS)CORE:
  - Use a process that puts limits on the security level for distinguishing and forgery.
  - $q, v = 2^{20}$ for AES-GCM CA and AES-CCM CA/IA, $v = 2^{30}$ for CCM_8 IA, not limit for other?
  - $q, v = 2^{20}$ for all algorithms?
  - $l = 2^{8}$ (4 kB) ?