TLS/DTLS 1.3 Profiles for the Internet of Things

draft-ietf-uta-tls13-iot-profile-03

draft-ietf-uta-tls13-iot-profile.authors@ietf.org

Open Issues

https://github.com/thomas-fossati/draft-tls13-iot/issues

- CCM_8 troubles
- Relaxing the initial timer values
- Long connections without renegotiation
- Examples of client EE cert IDs

CCM_8 troubles

https://github.com/thomas-fossati/draft-tls13-iot/issues/7

- The integrity limits of CCM_8 (i.e., its ability to survive repeated forgeries) are very low due to the reduced tag size
- This creates a cheaply exploitable DoS surface when used in DTLS, making CCM_8 basically unusable without *very* careful risk evaluation
- Unfortunately, CCM_8 is the only MTI ciphersuite in CoAP (and RFC7925) — no plan B ☹

CCM_8 troubles (cont.)

https://github.com/thomas-fossati/draft-tls13-iot/issues/7

Discussing / evaluating the deprecation strategy for CCM_8

Alternatives:

- Precisely describe the parameters combination (l, v, q) and the associated risk and let the user decide? (This is the approach taken by OSCORE.)
  - PRO: actionable
  - CON: leaving (even moderately) complicated risk assessment to the user tends to end up badly
- Just say “NOT RECOMMENDED.”, and avoid two solid pages of caveats?

CCM_8 troubles (cont.)

https://github.com/thomas-fossati/draft-tls13-iot/issues/7

What ciphersuite should we promote instead?

- CCM (w/ full 16 bytes tag)
  - PRO: easy upgrade path
  - CON: if we want to increase compatibility with IoT cloud services, GCM is a better fit as none of the major providers currently supports CCM ciphersuites
- TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
  - PRO: robust, well supported
  - CON: The unavoidable cost both in peek RAM usage (+660 bytes) and per-packet overhead (+8bytes) (*) but the latter is what gives us acceptable security

In the (near?) future NIST lightweight crypto outcome might give us a more constrained-friendly option.

Relaxing the initial timer value

https://github.com/thomas-fossati/draft-tls13-iot/issues/13

RFC 7925 RECOMMENDs a very high (9s) initial timer value for the DTLS 1.2 handshake.

Reason were:

- Avoiding spurious retransmits when the public key operation is taking long on a constrained node
- Avoiding congestion on LLNs when the loss is genuine and the flight to retransmit is bulky (e.g., certificate), and maybe fragmented
- To cater for very high latency variance in certain access technology (e.g., GSM-SMS)

Relaxing the initial RTO (cont.)

https://github.com/thomas-fossati/draft-tls13-iot/issues/13

DTLS 1.3 now offers per-record retransmission and therefore less congestion risk on loss, which makes point 2 less relevant

- Should we soften the initial RTO recommendations for TLS 1.3?
- Options:
  - Keep RFC 7925, i.e.: 9s
  - Relax to RFC 2988, i.e.: 3s
  - Further relax to default RFC 6347 / RFC-to-be 9147 (both inherit from RFC 6298), i.e.: 1s

Long connections without renegotiation

https://github.com/thomas-fossati/draft-tls13-iot/issues/8

This is about what recommendation we can make (now that renegotiation is gone) to deal with semi-permanent, mutually authenticated connections that need to rekey and check the associated certificate credentials? A common use case in Industrial IoT.

- There has been a long discussion on the list [1]

[1] https://mailarchive.ietf.org/arch/msg/tls/vTxwj2iShME6c7AHg_Ub-_eS_fM/

Examples of client EE cert IDs

https://github.com/thomas-fossati/draft-tls13-iot/issues/15

Now that we have relaxed the requirement to stick to EUI-64 for client EE IDs, we should add a few examples from IoT-specific profiles, e.g.:

- GSMA eUICC [1]
- LwM2M [2]

[1] https://www.gsma.com/esim/wp-content/uploads/2021/02/SGP.14-v2.1.pdf

[2] http://www.openmobilealliance.org/release/LightweightM2M/V1_2-20201110-A/HTML-Version/OMA-TS-LightweightM2M_Core-V1_2-20201110-A.html