Run2me

Overhead of Full Handshake

• Lengthy message exchange
• High computation cost

WiSMote
• 16 MHz MCU
• 16 bit arch.
• NIST P-256

Runtime Overhead of Full Handshake

![Graph showing the runtime overhead of a full handshake.](image_url)
Never Close Session

• Potential issues
  1. Heartbeats [RFC 6520] (e.g., stateful FW [RFC 6092])
  2. Short-lived server sessions (e.g., large-scale services)
  3. Constrained server (e.g., CoAP server)
Abbreviated Handshake

- DTLS 1.2 base specification [RFC 6347]
  - Re-use session state from previous session
    - Indicated by non-zero SessionID in ClientHello
  - No up-front agreement to use abbreviated handshake
  - Constrained devices may unnecessarily store state
Session Resumption without Server-Side State

- **TLS extension** [RFC 5077]
  - Enable one peer to remain stateless while session inactive
  - Limited to (D)TLS server->client state offloading
  - Constrained (D)TLS client scenario not considered
Session Resumption Type Negotiation

- Enable **explicit** type agreement
- Session state only stored when agreed upon
Session Resumption without Client-Side State

- Signaling strongly based on RFC 5077
  - SessionTicket Extension
  - NewSessionTicket Message
Ticket Construction Recommendations

• Most recommendations in RFC 5077 apply
• Changes needed for...
  – Cipher suite for ticket protection (→ AES CCM)
  – ClientAuthenticationType

```c
struct {
    ClientAuthenticationType client_authentication_type;
    select (ClientAuthenticationType) {
        case anonymous: struct {};
        case certificate_based: ASN.1Cert certificate_list<0..2^24-1>;
        case psk: opaque psk_identity<0..2^16-1>;
    } ClientIdentity;
```
Status and Next Steps

• Status quo
  – Signaling (mostly) done
  – Proof of concept implementation done

• ToDos
  – Extend on ticket construction
    • Allow to check certificate status

• Questions
  – Draft considered useful?
  – How and where to proceed?