

# Vorstellung der Bachelorarbeit: Anpassungen von DTLS zur sicheren Kommunikation in eingeschränkten Umgebungen

Lars Schmertmann

[lars@tzi.org](mailto:lars@tzi.org)

TZI, Universität Bremen, Deutschland

Kolloquium der AG Rechnernetze  
30.09.2013

# Gliederung

- ▶ Motivation
- ▶ Hardware & Umgebung
- ▶ DTLS
  - ▶ Handshake
  - ▶ Ausgewählte DTLS-Header
- ▶ Mögliche Lösungen
  - ▶ Handshake über CoAP
  - ▶ Stateless Header Compression
- ▶ Geeignete Ciphersuit
- ▶ Praktische Umsetzung

# Motivation

- ▶ Projekt GOBI
  - ▶ Sicherheit ist auch in eingeschränkten Umgebungen notwendig
  - ▶ Mikro-DTLS (eher Mikro-Sicherheit)
- ▶ TLS und DTLS sind bewährte Standards
- ▶ DTLS komplexer als TLS
- ▶ Problematisch in eingeschränkten Umgebungen

## Hardware & Umgebung

- ▶ Econotag: mc13224v Development-Board
  - ▶ Freescale MC13224v ARM7TDMI-S Microcontroller
  - ▶ IEEE 802.15.4 Funkstandard
  - ▶ AES Hardware-Engine
  - ▶ 128 KiB Flash-Speicher
  - ▶ 96 KiB RAM
- ▶ IEEE 802.15.4 MTU: 127 Byte
  - ▶ - 48 Byte 6LoWPAN-Header
  - ▶ - 8 Byte UDP-Header
  - ▶ = 71 Byte Nutzdaten
- ▶ SmartAppContiki = Contiki + Erbium + CoAP 13
  - ▶ 81 KiB (bei angepasster Konfiguration)

# DTLS

## Handshake

Client		Server
-----		-----
ClientHello	-0->	
		<-0- HelloVerifyRequest (D cookie)
ClientHello (+ cookie)	-1->	
		<-1- ServerHello
		<-2- *Certificate
		<-3- *ServerKeyExchange
		<-4- *CertificateRequest
		<-5- ServerHelloDone
Certificate*	-2->	
ClientKeyExchange	-3->	
CertificateVerify*	-4->	
[ChangeCipherSpec]	--->	
Finished	-5->	
		<--- [ChangeCipherSpec]
		<-6- Finished
Application Data	<----->	Application Data

# DTLS

## Ausgewählte DTLS-Header

```
struct {  
    ContentType type;  
    ProtocolVersion version;  
    uint16 epoch;  
    uint48 sequence_number;  
    uint16 length;  
    uint8  payload[length];  
} DTLS_Record;
```

= 13 Byte

```
struct {  
    HandshakeType msg_type;  
    uint24 length;  
    uint16 message_seq;  
    uint24 fragment_offset;  
    uint24 fragment_length;  
    uint8  payload[f._length];  
} Handshake;
```

= 12 Byte

## Mögliche Lösungen

Vorschläge im Entwurf von K. Hartke und O. Bergmann:  
<http://tools.ietf.org/html/draft-hartke-core-codtls-02>

- ▶ Handshake über CoAP
- ▶ Stateless Header Compression

# Mögliche Lösungen

## Handshake über CoAP - Teil 1

```
      POST /dtls --->
ClientHello

      <--- 1.02 Verify
              HelloVerifyRequest

      POST /dtls --->
ClientHello
(mit cookie)

      <--- 2.01 Created
              ServerHello (S=X)
              *Certificate
              *ServerKeyExchange
              *CertificateRequest
              ServerHelloDone
```



# Mögliche Lösungen

## Handshake über CoAP - Teil 2

```
      POST /dtls?s=X --->  
      Certificate*  
ClientKeyExchange  
CertificateVerify*  
  ChangeCipherSpec  
    Finished
```

```
<--- 2.04 Changed  
      ChangeCipherSpec  
      Finished
```

```
Application Data <--> Application Data
```

# Mögliche Lösungen

## Stateless Header Compression

```
struct {  
    uint8  :1;  
    RecordType type:2;  
    Version version:2;  
    Epoch epoch:3;  
    uint8  :3;  
    SequenceNumber snr:3;  
    RecordLength length:2;  
    uint8  payload[0];  
} DTLSRecord_t;
```

= 2 - 15 Byte

```
struct {  
    ContentType type:6;  
    ContentLength len:2;  
    uint8  payload[0];  
} Content_t;
```

= 1 - 4 Byte

## Geeignete Ciphersuit

- ▶ TLS\_PSK\_ECDH\_WITH\_AES\_128\_CCM\_8

Eigenschaften:

- ▶ Durch CCM wird keine Hash-Funktion für den MAC benötigt
- ▶ Durch den Einsatz von CBC-MAC mit PSK kann auf HMAC mit SHA-256 verzichtet werden

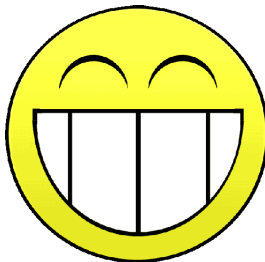
Auswirkungen auf die Programmgröße:

- ▶ Einsparung durch AES in Hardware: ~1.5KiB
- ▶ Einsparung durch CBC-MAC mit PSK: ~1.5KiB

## Praktische Umsetzung

- ▶ Ein Handshake zur Zeit
- ▶ Stack mit Push und Clear im Flash-Speicher für „Finished“
- ▶ Wechsel des Pre-shared Key nach Handshake
- ▶ Ablage der Session-Daten im Flash-Speicher

Vielen Dank für  
eure Aufmerksamkeit



# Fragen

