DTLS improvements

For constrained networks

Connection delays, payload size

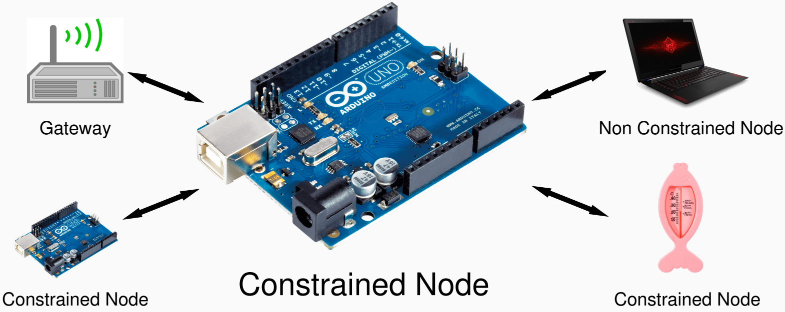
Philippe PITTOLI, Pierre DAVID, Thomas NOËL

July 5, 2016

ICube Laboratory – Strasbourg university

Introduction and context

Introduction: challenge

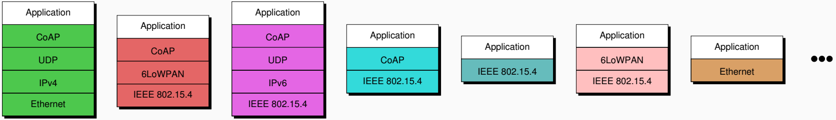


16 MHz, 16 KB RAM, 128 KB Flash

Communication with very constrained objects

Example: class 1 constrained node (RFC 7228)

Introduction: various stacks



Security layer adaptable to various kind of networks

Objectives

Objectives

- Fast secured communication with constrained nodes
- Communication stack independent

Metrics

1. Connection and communication delays
2. Solution cost
   - memory, communication

Assessment: asymmetric cryptography too costly [1]

- more than 2 s for a signature check (8 MHz)

[1] An Liu and Peng Ning, ***TinyECC: A Configurable Library for Elliptic Curve Cryptography in Wireless Sensor Networks***, In Proceedings of the 7th International Conference on Information Processing in Sensor Networks, IPSN '08, Washington, DC, USA, 2008. IEEE Computer Society.

Context

Hypothesis

- Only symmetric cryptography
- Pre-shared encryption keys
  - IETF use case (ACE WG)
- Known identities
  - deduced (ex: via MAC or IP addresses, via the application...)
- Unique encryption key between two nodes at a time
  - no need for several security contexts

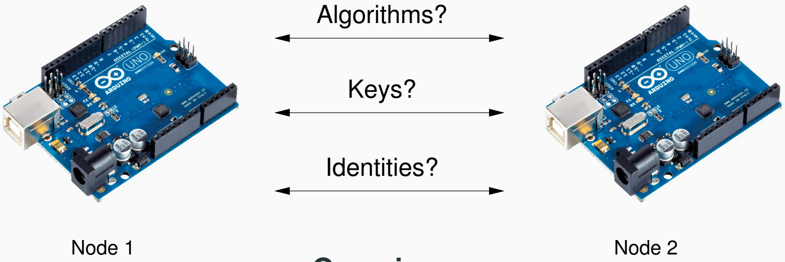
Used protocol: Datagram Transport Layer Security

- **Protocol stack independent**
- **Proposed in constrained networks**

Datagram Transport Layer Security

Datagram Transport Layer Security (DTLS)

Provides communication security between two nodes (RFC 6347, v1.2, January 2012)



Overview

DTLS: unsuitable for constrained environments

Connection cost

- 10 messages exchanged
  - without certificates nor optional messages

Message cost after connection establishment

- 29 bytes overhead per message

DTLS optimizations

DTLS: negotiation without optional messages

Client — Server

ClientHello : Algorithm list →

HelloVerifyRequest : Cookie ←

ClientHello : Algorithm list, Cookie →

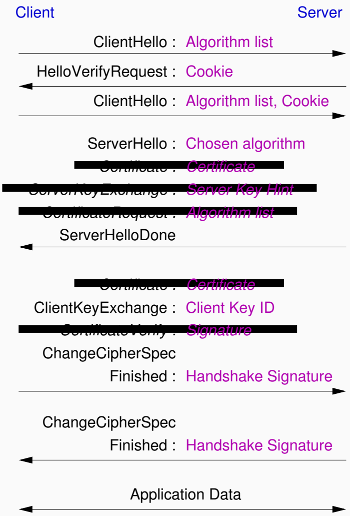
ServerHello : Chosen algorithm
[redacted]
ServerHelloDone ←

[redacted]
ClientKeyExchange : Client Key ID
[redacted]
ChangeCipherSpec
Finished : Handshake Signature →

ChangeCipherSpec
Finished : Handshake Signature ←

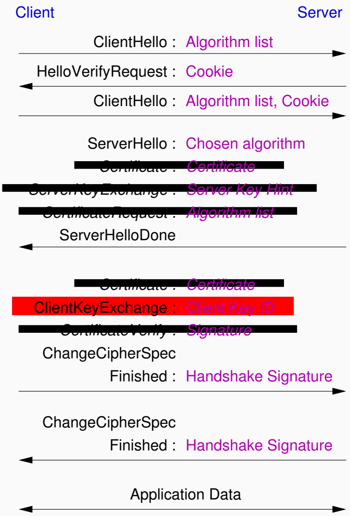
Application Data ↔

DTLS: negotiation without optional messages



10 mandatory messages in DTLS 1.2

DTLS: negotiation without optional messages



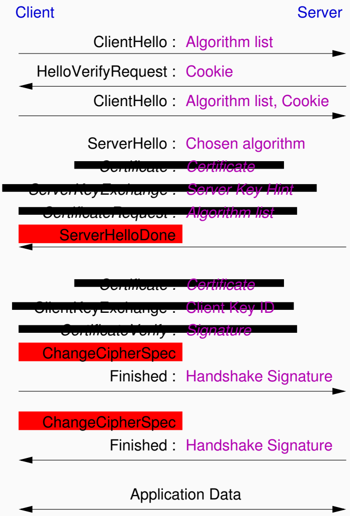
First optimization

Removal of messages related to identity exchange

Justification

- Known identity
- Unique encryption key between two nodes

DTLS: remaining messages



Second optimization

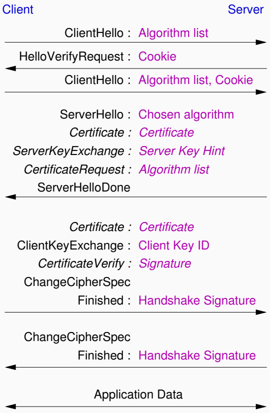
Removal of unnecessary messages

Justification

- Fixed message order

Comparison

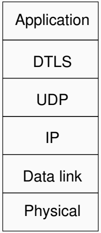
Original DTLS



Optimized DTLS

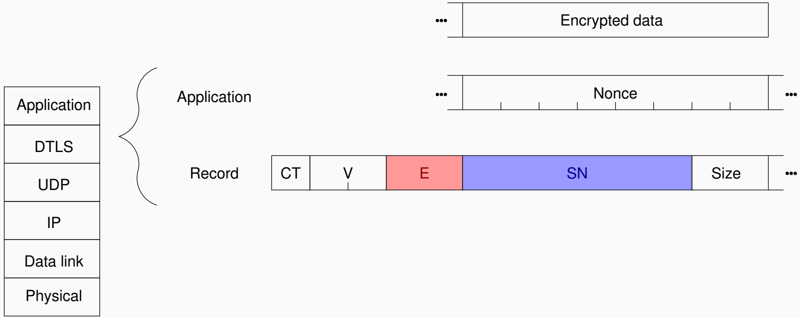
6 remaining messages without functionality loss

DTLS: layer optimization



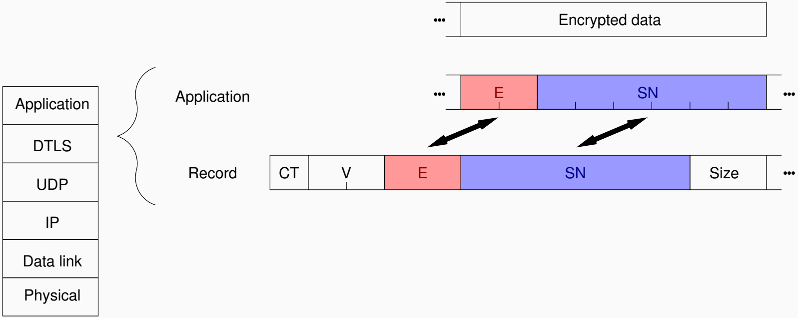
DTLS stack

DTLS: layer optimization



DTLS headers: Record and Application

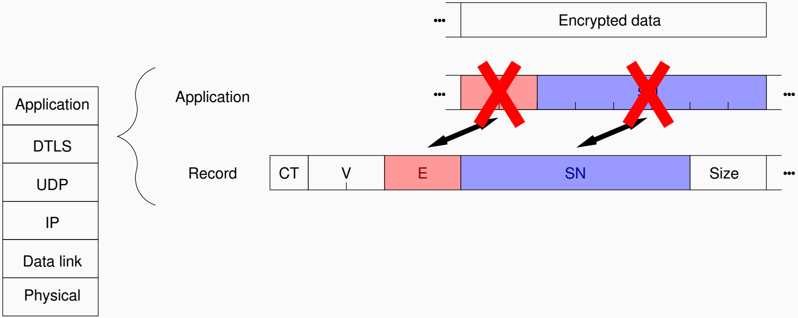
DTLS: layer optimization



Nonce needs to be unique by session, so by convention

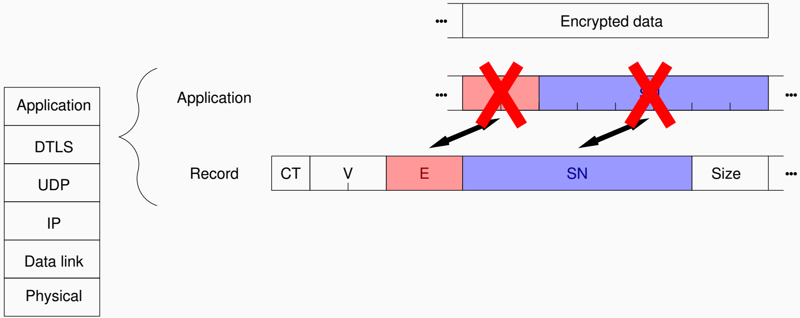
Application header copies 2 fields from Record header

DTLS: layer optimization



Field copy removal: 8 byte gain

DTLS: layer optimization



Field removal without consequences over security

Represents 6% of the total packet size in IEEE 802.15.4

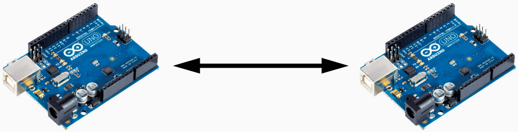
9% payload gain over original DTLS

Experimentations and results

Test environment

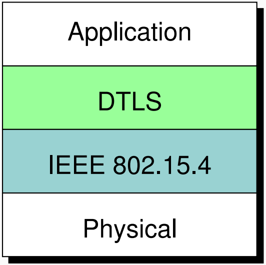
Hardware

ATMega128RFA1: 16 MHz, 16 KB SRAM, 128 KB Flash



DTLS Client

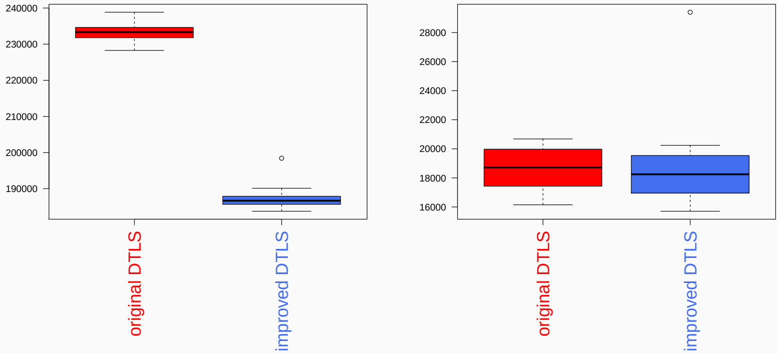
DTLS Server



Minimal hardware and software architectures

to quantify the impact of DTLS

Results: connection and communication delays



Connection delay

(100 connection est.)

Communication delay

(3000 exchanged messages)

20% faster connection

Results: memory footprint

Target

ATMega128RFA1: 16 MHz, 16 KB SRAM, 128 KB Flash

	RAM	Flash
Original DTLS	11.2 kB	49 kB
Optimized DTLS	**10.3 kB**	**46.6 kB**
Without security	0.897 kB	8.0 kB

Memory footprint

Memory footprint gain: 5.1% (RAM), 1.8% (Flash)

Conclusion

Conclusion

Same security level as original DTLS

- Same security context exchanged

Low memory footprint cost

Connection and communication delays

- 20% faster connection
- Reduced fragmentation
  - 21 bytes overhead per message instead of 29

9% payload gain over original DTLS