Securing the Internet of Things with DTLS

Thomas Kothmayr[1], Wen Hu[2], Corinna Schmitt[1], Michael Brünig[2], Georg Carle[1]

[1]Department of Computer Science, Chair for Network Architectures and Services, Technische Universität München, Germany

[2]CSIRO ICT Centre, Australia

kothmayr@in.tum.de, {schmitt, carle}@net.in.tum.de, {wen.hu, michael.bruenig}@csiro.au

Abstract

Usecases for wireless sensor networks, such as building automation or patient care, often collect and transmit sensitive information. Yet, many deployments currently do not protect this data through suitable security schemes. We propose an end-to-end security scheme build upon existing internet standards, specifically the Datagram Transport Layer Security protocol (DTLS). By relying on an established standard existing implementations, engineering techniques and security infrastructure can be reused which enables easy security uptake. We present a system architecture for this scheme and show its feasibility through the evaluation of our implementation.

Categories and Subject Descriptors

J.7 [**Computer Applications**]: Computers in Other Systems

General Terms

Design, Standardization

Keywords

Wireless Sensor Network (WSN), DTLS, Security

1 Introduction

After the standardization of the physical and MAC layer (IEEE 802.15.4) as well as the routing (6LoWPAN RPL) and application layer (CoAP) for the Internet of Things, an effort to standardize security follows naturally. In the internet, security is often achieved through the Transport Layer Security protocol (TLS). However, it requires reliable messaging and is therefore not suited to lossy low power networks. An alternative that is built upon datagram semantics exists in the Datagram Transport Layer Security (DTLS) protocol [3] which we propose as a basis for the standardization of an end-to-end security scheme in the Internet of Things.

2 Hardware

We rely on the RSA algorithm to secure the key exchange. Previous work has shown RSA to be infeasible for deployment on sensor nodes when implemented in software [1]. Hu et al. presented a sensor node platform called secfleck which uses a Trusted Platform Module (TPM) as a cryptographic accelerator for RSA operations [1]. We use the successor of secfleck, which features an Atmel Cortex SAM3U4E micro controller at 48 MHz and an integrated Atmel AT97SC3203S TPM for our purposes [2]. A TPM offers the unique benefit of physical tamper resistance, meaning an attacker cannot gain knowledge about the keys stored in the TPM even if physical access is possible. Furthermore the TPM acts as a hardware accelerator for RSA operations and as a unique source of identity based on the RSA keypair generated within the TPM.

3 Datagram Transport Layer Security

DTLS is an adaption of TLS for datagram based communication which provides equivalent security guarantees. Figure 1 shows a client authenticated handshake as defined by DTLS and supported in our implementation.

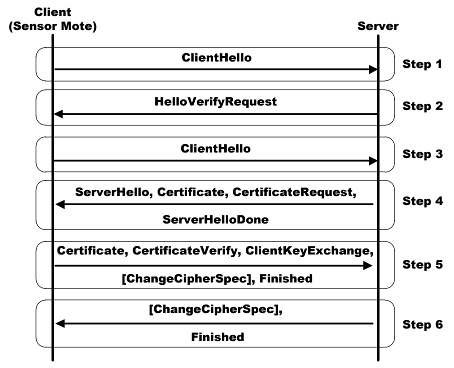


Figure 1. Client authenticated DTLS Handshake

The modifications made to TLS mostly consist of adding explicit sequence numbers to the messages and the introduction of retransmission timers during the handshake. This enables implementations to handle message loss and reordering during the handshake phase of the protocol. DTLS only allows block ciphers for encryption because stream ciphers, which are also allowed by TLS, are not randomly accessible and thus unsuitable for datagram transport.

4 System Architecture

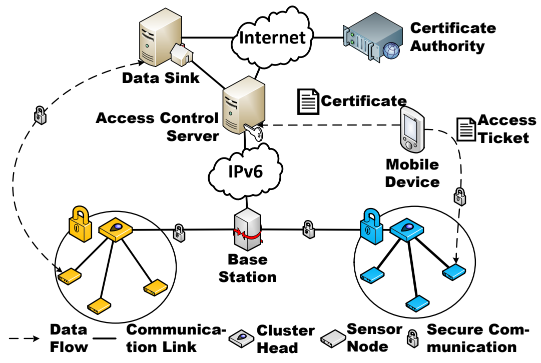


Figure 2. System Architecture

Figure 2 illustrates our architecture which is built around the usage of DTLS as security protocol. Every mote establishes a secured connection with the Data Sink after booting. Both the sensor node and the Data Sink are authenticated: If the mote supports hardware acceleration of RSA operations this is achieved by using TLS' RSA key exchange algorithm. During the handshake certificates signed by a trusted Certificate Authority which contain the mote's and server's RSA public key are exchanged. If a mote is incapable of performing RSA operations it is authenticated via a modification of the TLS Pre Shared Key algorithm. This means every mote supports one form of the DTLS handshake.

5 Evaluation

We have implemented a prototype that supports a client and server authenticated DTLS handshake as a proof of concept. On the server side we use OpenSSL 1.0.0d with minor modifications: The padding for RSA signature verification has been changed from PKCS#1 version 1.5 to version 2 and the Client only has to sign a SHA1 hash instead of the concatenation of a MD5 and SHA1 hash. These changes were necessary to maintain compatibility with the TPM hardware. We also introduced a 500ms delay between sending two handshake packets from the DTLS server to avoid flooding the sensor mote with data.

	1024 bit			2048 bit		
Drop Rate	Min.	Avg.	Max.	Min.	Avg.	Max.
0%	5,789	5,851	5,938	6,861	6,949	7,065
5%	5,835	16,435	27,592	16,600	26,945	39,680
10%	11,045	51,973	171,925	21,706	37,386	52,443

Table 1. Connection latency over a lossy link (ms)

Table 1 shows the average time over ten measurements that was needed to establish a DTLS connection when using 1024 and 2048 bit RSA keys for the server and Certificate Authority X.509 certificates. The column "drop rate" specifies the chance for a packet to be lost in the link layer. Lost packets result in all information from the current step being retransmitted so that the receiver has a chance to reconstruct it. We use BLIP as the routing protocol which leaves 100 byte of payload in each 802.15.4 packet. This results in 14 and 18 packets being sent for step 4 (size 1,311 byte) and 5 (1,794 byte) with 1024 bit RSA key length and 19 and 24 packets for step 4 (size 1,835 bytes) and 5 (2,337 bytes) with 2048 bit RSA keys respectively. Measurements have been conducted with a retransmission time of 5 seconds and a Maximum Transmission Unit (MTU) of 500 byte for both the server and sensor node. The measurements show that smaller packets due to smaller key sizes generally lead to a decrease in latency. The outlier at 10% packet loss with 1024 bit keys was caused by the finished message from the server being lost twice. With our current toolchain this inevitably leads to a timeout and restart for the handshake, causing a very large latency of over 170 seconds in that particular case. Overall, the figures show that even a relatively small percentage of lost packets leads to a significant increase in retransmissions and latency.

	Current (mA)	Time (ms)	Energy (mJ)
Computation	11.4	49	2.1
Radio TX	18	350	23.8
TPM Start	57.6	880	191
TPM TWI Bus	49.4	730	136
TPM Verify	59.8	120	27
TPM Encrypt	57	43	9.2
TPM Sign	57.6	875	190
Total			**579 mJ**

Table 2. Energy usage breakdown

The energy consumption of a successful handshake with 2048-bit RSA keys and without packet loss is shown in Table 2. The computation energy is the amount of energy spent for parsing the received certificates, hashing each handshake message and computing the HMAC for the last message as well as encrypting it. Radio transmission includes the standard TinyOS CSMA channel access. The overall energy usage proves that this is a feasible key exchange method: Our motes are powered by 3.7 V battery pack rated at 6600 mAh. In total it stores 87,900 J, meaning a mote could perform over 150,000 key exchange operations on one battery charge.

6 Conclusion and Future Work

Our prototype implementation has shown a DTLS Handshake with strong security parameters is feasible for key establishment in the Internet of Things. Future work will focus on an efficient scheme for key renegotiation and reducing the size of certificate messages and/or the sensitivity to link layer packet loss.

7 References

[1] W. Hu, P. Corke, W. Shih, and L. Overs. secfleck: A public key technology platform for wireless sensor networks. *Wireless Sensor Networks*, pages 296–311, 2009.

[2] W. Hu, B. Kusy, C. Richter, M. Brünig, and C. Huynh. Demo abstract: Radio-diversity collection tree protocol. In *IPSN*, 2011.

[3] E. Rescorla and N. Modadugu. Rfc4347: Datagram transport layer security. *IETF, Request For Comments*, 2006.

Securing the Internet of Things with DTLS

ACM SenSys
Nov 2011

Thomas Kothmayr, Corinna Schmitt, Georg Carle
Technische Universität München, Munich, Germany
kothmayr@in.tum.de, [schmitt, carle]@net.in.tum.de

Wen Hu, Michael Brünig
CSIRO ICT Centre, Brsibane, Australia
[Wen.Hu, Michael.Bruenig]@csiro.au

Technische Universität München

CSIRO

Motivation

Many Usecases for Wireless Sensor Networks involve collection and transmission of sensitive data

- E.g building automation, medical applications, metering, etc..

Standardization for other layers in network stack progressing

- Physical & MAC: IEEE 802.15.4
- Routing & Transport: 6LoWPAN, RPL
- Application: CoAP (Constrained Application Protocol)

Standardized security based on existing internet protocols offers many benefits

- Established security engineering practices
- Reuse of existing infrastructure (e.g. Certificate Authorities)
- Reuse of existing implementations (e.g. OpenSSL, GnuTLS, etc..)

→ **Increased security uptake**

Hardware

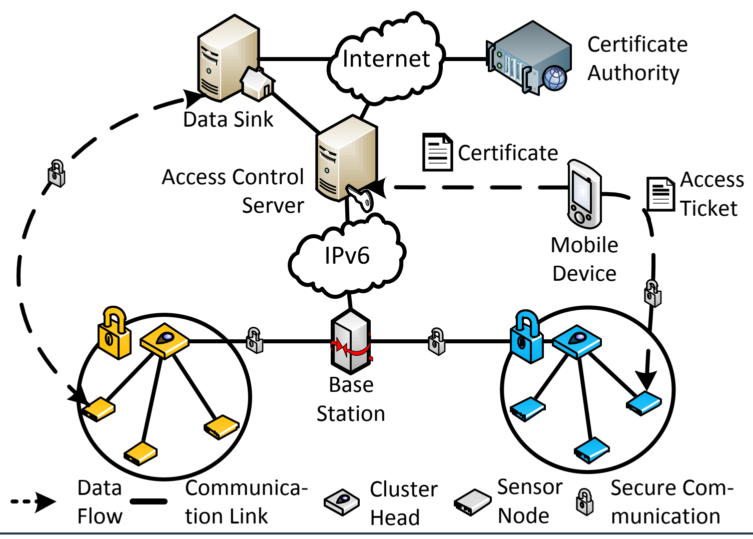
Sensornode with embedded Trusted Platform Module (TPM)

- Atmel SAM3U Microcontroller, 48 MHz
- 50 kB SRAM
- AT97SC3203S TPM

TPM has tangible security benefits

- Tamper proof storage of RSA-Keys
- Hardwaresupport for RSA Operations
- Unique Identity (asymmetric key-pair)

System Architecture



Security based on Datagram Transport Layer Security protocol (DTLS)

- Data protected from data source to sink
- Confidentiality, integrity and authenticity guaranteed through protocol

Multiple levels of security

- RSA capable devices authenticated via X.509 certificates during key exchange
- Constrained devices perform variant of TLS Pre Shared Key algorithm
- Data Sink authenticated via certificate either directly with the mote or with Access Control Server
- Motes organized in Clusters with common key to enable in-network aggregation

Peer-To-Peer Communication supported

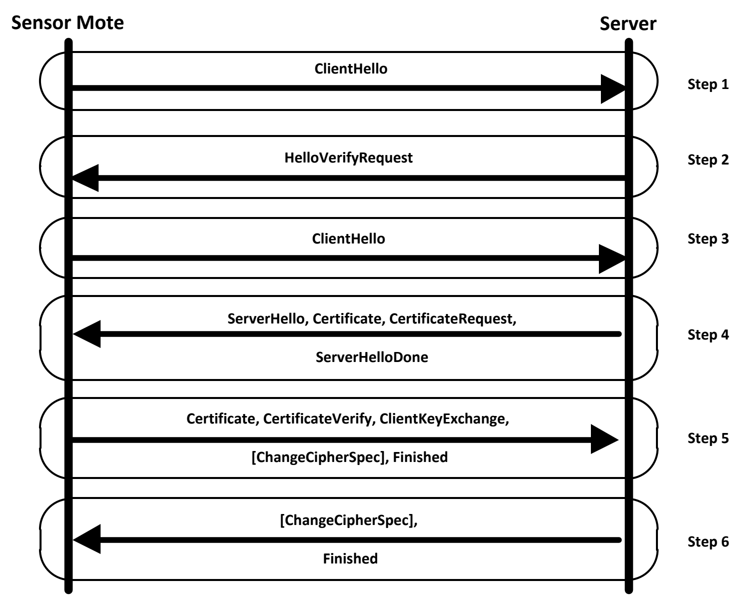
- Access Control Server grants Tickets to authenticated devices with sufficient rights
- Device requests conection from communication partner, key establishment based on DTLS

→ Device level access control

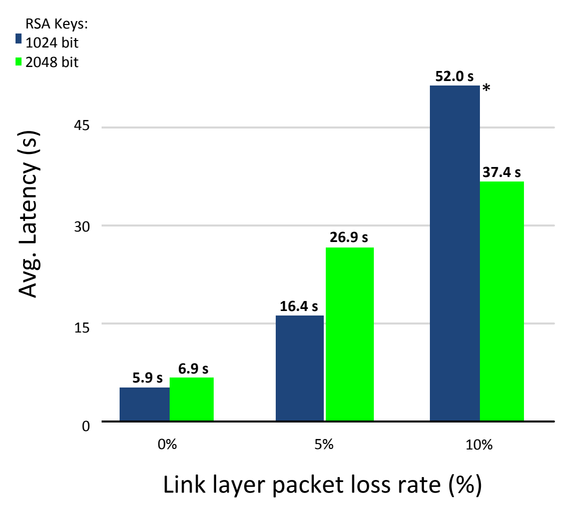
Application layer security protocol

- Drawbacks: routing information unprotected, potential for DOS Attacks
- Benefits: Only need keys for Data consumers instead of all neighbours, packet routing without additional decryption overhead, untrusted devices can still forward data

DTLS Handshake



Latency



*Caused by lost „finished" messages from server → Handshake times out and is restarted

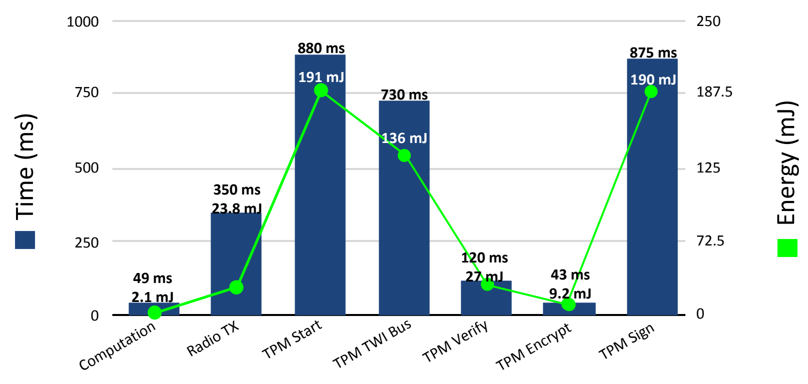
Chart shows the average latency over ten DTLS Handshakes

- Retransmission timer 5s
- Timeout 60s
- MTU 500 byte

Steps 4 and 5 of the handshake require large packets

- 1,311 & 1,794 bytes for 1,024-bit RSA Keys
- 1,835 & 2,337 bytes for 2,048-bit RSA Keys

Energy Cost



Handshake protected by 2,048-bit RSA:

- 579 mJ spent in total
- More than 15,000 key exchanges on one AA NiMH battery (2100mAh)

Conclusion

A Client and Server authenticated DTLS Handshake for WSNs is feasible

- TPM consumes most of the energy, but necessary for RSA
- Connection latency sensitive to lost packets, but only leads to major increase in energy usage if handshake times out and TPM operations have to be performed again

Future work

- Efficient key renegotiation
- Reducing message size or improving retransmission scheme → Handshake timeouts become less likely
- Implementing pre shared key algorithm for contrained devices