Embedded Security: From Sensor Networks to Internet of Things (IoT)

Dr. Wen Hu, Michael Bruenig, Thomas Kothmayr (TUM), Corinna Schmitt (U Zurich)
Principal Research Scientist/Research Project Leader
CSIRO Digital Productivity Flagship, Australia
Wireless Sensor Networks
- Homogenous devices
- Resource, energy and form factor limited
Cryptography challenges in sensor networks

- Very limited resources
  - 8-bit/16-bit microcontrollers
  - Less than 10KB RAM
  - AA batteries

- Security algorithms are computational and memory intensives
SKC vs. PKC

- **Symmetric Key Cryptography (SKC)**
  - Low computation cost
  - Smaller key sizes

- **Public Key Cryptography**
  - It provides more security than SKC…but it requires a nontrivial amount of processing power and memory

Past 2004 2007 Future

- Impossible to use PKC
- Doubt in using PKC
- Possible to use PKC
Cryptography engines

• Symmetric cryptography engine
  • AES 128-bit, new transceivers such as Atmel AT86RF212 and AT86RF230

• Asymmetric cryptography engine
  • SHA-1, 1024/2048-bit RSA
secFleck
### Table I. Comparison of RSA encryption times.

<table>
<thead>
<tr>
<th>Public Exponent (e)</th>
<th>Software 1024 bit</th>
<th>Software 2048 bit</th>
<th>Hardware 2048 bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>0.45s</td>
<td>65s</td>
<td>N/A</td>
</tr>
<tr>
<td>65,537</td>
<td>4.185s</td>
<td>450s</td>
<td>0.055s</td>
</tr>
</tbody>
</table>

### Table II. RSA computation time in trustedFleck for \( e = 65,537 \) and 2048 bit key.

<table>
<thead>
<tr>
<th></th>
<th>Encryption</th>
<th>Decryption</th>
<th>Sign</th>
<th>Verification</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>55ms</td>
<td>750ms</td>
<td>787ms</td>
<td>59ms</td>
</tr>
</tbody>
</table>
### Evaluation (II)

#### Table III. trustedFleck current consumption

<table>
<thead>
<tr>
<th>Module</th>
<th>Current (mA)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fleck3 (without radio, node idle)</td>
<td>8.0</td>
</tr>
<tr>
<td>Fleck3 + Receive</td>
<td>18.4</td>
</tr>
<tr>
<td>Fleck3 + Transmit</td>
<td>36.8</td>
</tr>
<tr>
<td>Fleck3 + TPM encryption</td>
<td>50.4</td>
</tr>
<tr>
<td>Fleck3 + TPM decryption</td>
<td>60.8</td>
</tr>
<tr>
<td>Fleck3 + TPM signature</td>
<td>60.8</td>
</tr>
<tr>
<td>Fleck3 + TPM signature verification</td>
<td>50.4</td>
</tr>
</tbody>
</table>

#### Table IV. trustedFleck (RSA and XTEA) encryption energy consumption for one bit of data.

<table>
<thead>
<tr>
<th>Platform</th>
<th>Current (mA)</th>
<th>Time (μs)</th>
<th>Energy (μJ)</th>
</tr>
</thead>
<tbody>
<tr>
<td>RSA (software, $e = 65,537, 2048$ bit key)</td>
<td>8.0</td>
<td>219,730</td>
<td>7,030.0</td>
</tr>
<tr>
<td>RSA (hardware, $e = 65,537, 2048$ bit key)</td>
<td>50.4</td>
<td>27</td>
<td>5.4</td>
</tr>
<tr>
<td>XTEA (software, 128 bit key)</td>
<td>8.0</td>
<td>18</td>
<td>0.6</td>
</tr>
</tbody>
</table>
Examples --- secure communications

Node A

Generates a random number $N_a$ (by $\text{fos TPM rand}$)

Decrypt with $\text{Sk}_A$, ($\text{fos TPM decryption}$)

Base

Decrypt with $\text{Sk}_\text{base}$,

Generate a new session key ($K_{BA}$),

($\text{fos TPM decryption}$ $\text{fos TPM rand}$)

$E(Pk_{\text{base}}, N_a, Req)$

$\text{fos TPM encryption}$

$E(Pk_A, N_a, K_{BA})$

($\text{fos TPM encryption}$)

Secure communication using SKC with $K_{BA}$
Examples --- remote attestation

**Attestator A**

*During boot time, update PCR I (Pi)*
(fos_tpm_pcrExtend)

*Obtain Pi and generate a signature* 
(fos_tpm_pcrQuote)

**Challenger C**

*Generates a random number Na (by fos_tpm_rand)*

*Issue PCR challenge (index = i, Na)*

*Ask for A’s public key*

*A’s public key (Pk_a)*

*Challenge response S(Pi, Na, Sk_a)*

*Verify the value Pi and the signature (fos_tpm_verifyPcrQuote)*
Summary

• Strong (2048-bit) asymmetric key cryptography for message authenticity and integrity, strong symmetric key cryptography for message confidentiality

• Affordable (financially, form factor, and energy consumption)

• Remote (platform and data) attestation for content trustworthiness
• Internet of Things
  – Heterogeneous devices (cortex M*)
  – Standard approach (802.15.4, RPL/6LoWPAN, COAP…)

[Images of electronic devices]
Motivation

- **Current situation:**
  - Many different use cases exist: Building system, medical apps, acquisition of resources.
  - Main task of sensor networks is the collection and transmission of different data.

- **Problem:**
  - Data can include sensitive information.
  - Trend: Integration of wireless sensor networks into the Internet (Internet of Things).
  - Trustworthiness of participants can differ.

- **Requests for the security solution:**
  - Confidentiality
  - Data Integrity
  - Data authenticity
Usage of standards

- Wireless Sensor Networks are comparable to Peer-to-Peer networks:
  - Self-organizing network of sensor nodes
  - Basic tasks of a node: Collect data, simple data processing, and forward data
  - Constrained memory, battery and computational power
  - IPv6 Connectivity → Nodes connected to Internet

- Different standardized security solutions exist:
  - Technologies and implementations (e.g. OpenSSL) exist and are well proven
  - Existing infrastructures (e.g. certificate authorities) can be used again.

- Different standards for network stack in WSNs already exist:
  - Physical & MAC Layer: IEEE 802.15.4
  - Routing & Transport Layer: 6LoWPAN, RPL
  - Application Layer: CoAP
Benefits of a standards based approach

Reuse of:
- Implementations (OpenSSL, etc..)
- Engineering techniques
- Infrastructure (Certificate Authorities, etc..)
- Expertise and Experience

\[ \rightarrow \text{Easier security uptake} \]

<table>
<thead>
<tr>
<th>Hardware used:</th>
</tr>
</thead>
<tbody>
<tr>
<td>TelosB / IRIS</td>
</tr>
<tr>
<td>OPAL-Mote</td>
</tr>
<tr>
<td>50kbyte SRAM</td>
</tr>
<tr>
<td>48 MHz Microcontroller</td>
</tr>
<tr>
<td>Trusted Platform Module</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Application</th>
<th>DTLS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Security</td>
<td>UDP → BLIP, RPL</td>
</tr>
<tr>
<td>Transport</td>
<td>IPv6 →</td>
</tr>
<tr>
<td>Network</td>
<td></td>
</tr>
<tr>
<td>Medium Access / Physical</td>
<td>IEEE 802.15.4</td>
</tr>
</tbody>
</table>
Opal node (front)

Microcontroller (32-bit)

LED

Radio
Opal node (back)

- Micro SD card slot
- Radio
- TPM
DTLS – Ultra short introduction

DTLS:

• Adaption of TLS for datagram transport
• Server and Client negotiate Hash algorithm and Cipher in Handshake
• Different authentication methods
  – RSA, DAS, DH, ECC, PSK,…
  – For us: RSA and later PSK

A DTLS Based End-To-End Security Architecture for the Internet of Things with Two-Way Authentication
Connecting to data sink

A DTLS Based End-To-End Security Architecture for the Internet of Things with Two-Way Authentication
P2P Connection

A DTLS Based End-To-End Security Architecture for the Internet of Things with Two-Way Authentication
Evaluation - DTLS Handshake

• Least understood component in IoT context

• Previous work evaluated other components
  • „Sizzle: A standards-based end-to-end security architecture for the embedded internet”
    → Server authenticated handshake with RSA and ECC
  • “Securing Communication in 6LoWPAN with Compressed IPSec”
    → Compression techniques for IPSec header during application data transfer

Challenge: IoT embedded nodes are limited to their resources!
• System’s performance
  – Packet handling
  – DTLS handshake performance
• Energy consumption
• Memory consumption
Evaluation - System’s performance (packet handling)

- Linear increase of round trip time
- Jumps approximately every 100 bytes
  - 128 bytes maximum MTU in layer 2 by IEEE 802.15.4
- Including header and tailer
- Jumps occur earlier when sending DTLS protected packets
  - Additional DTLS header, HMAC size, Initialization Vector

- Increasing packet size and processing overhead lead to an increased end-to-end transmission latency for DTLS packets compared to plaintext packets.
- The decreased performance for transmission latency is mostly due to the large packet overhead of up to 64 bytes.
- Calculation times DOES NOT contribute significantly:
  - SHA-1 hash of 255 bytes plain text message: 9 ms
  - Encryption with AES-128: 12 ms
Evaluation - System’s performance (DTLS handshake)

- Measurement duration:
  - Beginning of the handshake establishment
  - Client received a “FINISHED” message
- 15 measurements for each type of handshake
- Timeout: 5 sec

- Large standard deviation is caused by implementation behavior when messages lost.
  - DTLS states that an implementation should wait for an answer for a set amount of time after sending a flight.
  - Retransmission if no answer is received during this period.
- Time to execute a handshake is shorter for smaller RSA-keys and reduced by almost 2 sec when client authentication is omitted in the handshake.
- Packet loss mainly in multi-hop environment and larger DTLS messages are sent.
- Total energy consumption of client does not increase significantly
  - All TPM operations are only executed after successful receipt of all relevant server messages.
Energy draw for a fully authenticated DTLS handshake on OPAL node

![Graph showing energy draw over time](image)

Energy cost = \( \frac{U_{\text{probe}}}{R} \times t \times U_{\text{battery}} \)

<table>
<thead>
<tr>
<th></th>
<th>Current</th>
<th>fully authenticated handshake</th>
<th>server authenticated handshake</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computation</td>
<td>11.4 mA</td>
<td>35 ms, 1.59 mJ</td>
<td>33 ms, 1.50 mJ</td>
</tr>
<tr>
<td>Radio TX</td>
<td>18 mA</td>
<td>242 ms, 17.4 mJ</td>
<td>70 ms, 5.03 mJ</td>
</tr>
<tr>
<td>TPM Start</td>
<td>52.2 mA</td>
<td>836 ms, 174.46 mJ</td>
<td>836 ms, 174.5 mJ</td>
</tr>
<tr>
<td>TPM TWI</td>
<td>43.6 mA</td>
<td>688 ms, 120.0 mJ</td>
<td>476 ms, 83.0 mJ</td>
</tr>
<tr>
<td>TPM Verify</td>
<td>51.8 mA</td>
<td>59 ms, 12.2 mJ</td>
<td>56 ms, 11.6 mJ</td>
</tr>
<tr>
<td>TPM Encrypt</td>
<td>51.8 mA</td>
<td>39 ms, 8.07 mJ</td>
<td>40 ms, 8.28 mJ</td>
</tr>
<tr>
<td>TPM Sign</td>
<td>52.2 mA</td>
<td>726 ms, 151.5 mJ</td>
<td>-</td>
</tr>
<tr>
<td>Total</td>
<td>485.2 mJ</td>
<td>283.9 mJ</td>
<td>283.9 mJ</td>
</tr>
<tr>
<td>CPU idle</td>
<td>11.4 mA</td>
<td>3965 ms, 180.7 mJ</td>
<td>2265 ms, 103.2 mJ</td>
</tr>
<tr>
<td>Radio idle</td>
<td>18 mA</td>
<td>3758 ms, 270.4 mJ</td>
<td>2228 ms, 160.3 mJ</td>
</tr>
<tr>
<td></td>
<td></td>
<td>936.4 mJ</td>
<td>547.4 mJ</td>
</tr>
</tbody>
</table>

A DTLS Based End-To-End Security Architecture for the Internet of Things with Two-Way Authentication
Evaluation - Memory consumption

- Fully authenticates handshake with 2048-bit RSA keys
- OPAL resources: 48 kB RAM / 256 kB ROM

RAM consumption (byte)
→ Total: 17,839 byte RAM

ROM consumption (byte)
→ Total: 63,383 byte ROM
Established WSN at Department

Nodes with data collection purpose:

1. IRIS with mts300 or mts400
2. TelosB with activated sensors

- Gateway (TelosB)
- TelosB with aggregation purpose
- Opal

A DTLS Based End-To-End Security Architecture for the Internet of Things with Two-Way Authentication

Recording of received TinyPFIX messages in Listenr provided by TinyOS
Summary

• Today’s challenge:
  • Connection of different infrastructures on base of IP-communication
  • Internet of Things

• Adoption of powerful and well known protocols is suitable!
  • A standard based security architecture with two-way authentication for the Internet of Things was developed.
  • The authentication is performed during a fully authenticated DTLS handshake.
    – Exchange of X.509 certificates containing RSA keys
  • Secure provisioning:
    – Message integrity
    – Confidentiality
    – Authenticity
  • Solution has affordable energy, end-to-end latency, and

• Interoperability can be ensured with different vendors
• Application scenarios exchangeable

A DTLS Based End-To-End Security Architecture for the Internet of Things with Two-Way Authentication
On-going work

• Opal on a chip (TI CC2538)
  • Cortex M3 (32KB RAM and 512KB ROM)
  • IEEE 802.15.4 radio
  • RSA, ECC in hardware
  • ~$6

• OpenMote

• Has the dominated factor moved back to wireless transmissions?

• More advanced crypto approaches?

• Bluetooth LE security?
References


