A Public Key Infrastructure for Wireless Sensor Networks

Dr. Wen Hu
Research Scientist
CSIRO ICT Centre, Australia
Motivations

• Is security an issue in WSNs?
  • Over-the-air-programming (bootloader)
  • 6LowPan, TinyWebService, Unix-like OS...

• Security applications, health monitoring applications, metering applications.
WSN security challenges

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

• Security algorithms are computational and memory intensives
Related work

• **Symmetric key algorithms are computational efficient**
  - AES, XTEA, TinySec
  - Provide message confidentiality
  - Poor support for message authenticity and integrity
  - Key distribution algorithm?

• **Asymmetric key algorithms are resource intensive**
  - TinyPK (RSA) and TinyECC
Message authenticity and integrity

• PKCs such as RSA are widely used in the Internet to ensure message authenticity and integrity
  • Secure hashing algorithms such as SHA-1 to ensure message integrity
  • Digital signatures to ensure message authenticity

• RSA parameters
  • Public component (e)
  • Key size (s)
Trusted Platform Module (TPM)

- Cryptography operation engine
  - RNG, SHA-1, HMAC, RSA

- Platform Configuration Registers (PCRs)
  - Stores the signatures (by SHA-1 or HMAC)
  - Remote/local attestation

- Secure I/O

- Seal Storage
TrustedFleck Primitives

```c
/* D */
uint8_t fos_tpm_getPubKey(uint8_t *pubKey);

/* T */
uint8_t fos_tpm_encryption(uint8_t *msg, uint16_t len,
                           uint8_t *pubKey, uint8_t *cipher);
uint8_t fos_tpm_decryption(uint8_t *cipher, uint8_t *msg,
                            uint16_t *len);

/* S */
uint8_t fos_tpm_sign(uint8_t *digest, uint8_t *signature);
uint8_t fos_tpm_verifySign(uint8_t *signature, uint8_t *pubKey,
                           uint16_t *digest);

/* K */
uint8_t fos_xtea_encipher(uint8_t *msg, uint8_t *key,
                          uint8_t *cipher, uint8_t nRounds);
uint8_t fos_xtea_decipher(uint8_t *cipher, uint8_t *key,
                          uint8_t *msg, uint8_t nRounds);

/* Symmetric session key retrieve and store*/
fos_xtea_getkey(uint8_t *key, uint8_t *location);
fos_xtea_storekey(uint8_t *key, uint8_t *location);
```
### 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>Encryption</th>
<th>Decryption</th>
<th>Sign</th>
<th>Verification</th>
</tr>
</thead>
<tbody>
<tr>
<td>55ms</td>
<td>750ms</td>
<td>787ms</td>
<td>59ms</td>
</tr>
</tbody>
</table>
### 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 --- Symmetric Key Request

Node A

Generates a random number $N_a$ (by fos_tpm_rand)

Decrypt with $Sk_A$, (fos_tpm_decryption)

Base

Decrypted with $Sk_{base}$, Generate a new session key ($K_{BA}$),
(fos_tpm_decryption fos_tpm_rand)

$E(Pk_{base}, N_a, Req)$
(fos_tpm_encryption)

$E(Pk_A, N_a, K_{BA})$
(fos_tpm_encryption)
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 \( N_a \) (by fos_tpm_rand)

Issue PCR challenge (index = i, \( N_a \))

Ask for A’s public key

Challenge response \( S(P_i, N_a, S_{ka}) \)

Verify the value \( P_i \) and the signature (fos_tpm_verifyPcrQuote)

A’s public key (\( P_{ka} \))
Discussions

• Secure software update

• Backward and forward secrecy

• Secure Remote Procedure Call
Conclusions

• Strong (2048-bit) asymmetric key security for message authenticity and integrity

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

• Remote attestations (trusted sensor networks)

• Easy to use
Thank you!
Trusted Primitives

```c
/*TPM PCR functions*/
uint8_t fos_tpm_pcrExtend(uint8_t pcrIndex, uint8_t *inputSha1, 
                         uint8_t *extendSha1);

uint8_t fos_tpm_pcrRead(uint8_t pcrIndex, uint8_t *outputSha1);

uint8_t fos_tpm_pcrQuote(uint8_t key_index, uint8_t *signature, 
                         uint8_t *digest);

uint8_t fos_tpm_verifyPcrQuote(uint8_t* signature, uint8_t *pubKey, 
                               uint8_t *digest);
```
Evaluations (II)

Table V. trustedFleck PCR operation computational time, current and energy consumption

<table>
<thead>
<tr>
<th>Module</th>
<th>Computational Time (ms)</th>
<th>Current (mA)</th>
<th>Energy (mJ)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fleck3 + TPM PCR read</td>
<td>5.8</td>
<td>52.8</td>
<td>0.918</td>
</tr>
<tr>
<td>Fleck3 + TPM PCR quote</td>
<td>1,400</td>
<td>64</td>
<td>268.8</td>
</tr>
<tr>
<td>Fleck3 + TPM PCR verify quote</td>
<td>900</td>
<td>52</td>
<td>140.4</td>
</tr>
<tr>
<td>Fleck3 + TPM PCR extend</td>
<td>see Fig. 8</td>
<td>51.2</td>
<td>see Fig. 8</td>
</tr>
</tbody>
</table>

![Graph showing computation time and energy consumption](image)