Indraj: Digital Certificate Enrollment for Battery-powered Wireless Devices

Zhitao He        Martin Furuhed        Shahid Raza
Scope

- Small, battery-powered devices
  - Sensors and actuators
  - 3-10 years battery lifetime
  - 5$ MCU with onchip Flash and RAM, and
  - Lossy and low power radio: IEEE 802.15.4, BLE

- From RFC 7228 "Terminology for Constrained-Node Networks":
  - Class 2 devices: ~ 50 KiB RAM, ~ 250 KiB Flash
    - Sufficient to run size-optimized IPv6 stack
  - Energy class 1: periodic battery recharge/replacement, or
  - Energy class 2: non-replaceable primary battery
  - Power class 1: low power, connected, with perhaps high latency

- Wireless data
  - Encrypted with good, symmetric cipher e.g. AES
  - But how to install the keys?
The Internet of things: protected, precariously managed

- Heritage in wireless PANs
  - ZigBee, BLE, etc
  - Private ownership: single master, a few “pet” gadgets
  - Proprietary key provisioning of symmetric keys
    - $N$ devices $\Rightarrow N$ keys
    - Non-standard protocol for key renewal/revocation
  - Stretched to larger, residential or industrial WLANs
    - 30 – 100 nodes

- Problem: poorly scalable with complex applications
  - Arbitrarily many users
  - Device-to-device communication
  - $N$ users/devices $\Rightarrow N^2$ keys
Scaling up the Internet of things securely

• Our vision
  • Leverage global device connectivity enabled by IPv6 and Co.
  • Assign unique digital certificate per device
  • Device certs are anchored to a trusted root certificate
    • i.e. use public key infrastructure
    • “Things” become first-class citizens on Internet

• First challenge: initial enrollment
  • Security
  • Cost efficiency
    • Cannot expect same amount of effort as installing a website certificate
Solution: automation of certificate enrollment

Our automated enrollment procedure
Enrollment Protocol Design

- Messages encrypted by DTLS with PSK-based handshake
- Two request-response round trips between enrolling device and CA
  1. GET /cacerts
  2. POST /simpleenroll
- Message semantics and encoding reused from Enrollment over Secure Transport (EST) protocol (RFC7030)
  - Standard X.509v3 certificate format
  - Standard message encapsulation formats
  - Protocol supported by mainstream CA software vendors
  - Cipher strength evolves in sync with DTLS
Adapting the EST protocol stack for higher performance

- Replace EST underlying HTTPS transport with CoAPs (CoAP over DTLS)
  - CoAP (RFC7252) has smaller header size than HTTP
  - DTLS (RFC6347) and UDP provides lower latency than TCP/TLS

- Result: a lightweight enrollment protocol
  - Compatible functionality and equivalent level of security as EST
  - Optimized performance
  - Automated: no human interaction
Example enrollment timeline

- Long messages are divided into blocks at CoAP layer
- Further fragmentations occur at lower layers: UDP/IP/6LoWPAN
Modular implementation of Indraj Client on Contiki OS

- ARM Cortex-M3 CPU @ 16 MHz
  - 128 kB Flash
  - 32 kB RAM
  - HW ECC & AES cryptos
  - IEEE 802.15.4 radio
- Compiled with Contiki IPv6 stack

Memory usage
Evaluation set-up: network and workload

- **Client**
  - Sends a 200-byte CSR to server
  - Radio runs 6LoWPAN stack with CSMA MAC
    - A range of duty cycles are tested

- **Server**
  - Toy example CA application running atop open source Java CoAP library *Californium*
  - Returns two CA certs to client request (600 B)
  - Issues client cert (300 B)

- **Border router**
  - Routes IP traffic between radio 6LoWPAN network and server localhost Linux stack
Enrollment latency

- Full radio duty cycle
  - 4 s
  - Dominated by heavy-duty CPU processing
  - Potential reduction with higher CPU freq.
- Low radio duty cycles
  - 5 – 10 s
  - Increasing DTLS connection overhead
- Other unaccounted factors
  - Extra hops over intermediate nodes
  - Border router-to-CA server latency
Energy and power

- Energy consumption
  - One-off cost, between re-enrollments or battery replacement
  - <0.01% of battery capacity

- Potential problem for weakly powered devices
  - E.g. energy harvesting circuits
  - Short-term voltage drop resets device
Limitations

• Device lacks a true random number generator
  • For generating its key-pair locally
  • Solution: use server-side key generation of EST protocol; but
    • The server generated private key leaves a trace on server storage and on wireless channel
  • New chips have built-in TRNG

• Device lacks trustworthy wall clock to validate its certificate
  • No good solution, only partial solutions
  • Fallback: manufacturer-installed certificate that is valid for very long time
Standardization in progress

• IETF draft: **EST over secure CoAP** (EST-coaps)
  • Started on 2018-02-28
  • Latest revision 2019-03-08
    • DTLS authentication of client uses a preinstalled certificate by manufacturer
    • PSK-based DTLS handshake no longer supported
  • Adopted by OMA LwM2M standard as alternative certificate mode
Conclusions

• We have designed an automated enrollment protocol for battery-powered wireless devices
  • Lightweight
  • Secure

• Future work
  • Certificate revocation

• Open source plan
  • Pull request to Contiki-NG project on Github
  • https://github.com/contiki-ng/contiki-ng
Thank You!

zhitao.he@assaabloy.com

martin.furuhed@nexusgroup.com

shahid.raza@ri.se