OSCAR: Object Security Architecture for the Internet of Things*

Mališa Vučinić ★, Bernard Tourancheau ✦, Franck Rousseau ✦, Andrzej Duda ✦, Laurent Damon ★, and Roberto Guizzetti ★.

★ STMicroelectronics
✦ LIG - Drakkar

Crolles, April 7th 2014

This presentation will make you believe public key cryptography is less costly than symmetric key (and radio communication).
• Motivation – why not just (D)TLS
• OSCAR – concepts behind
• OSCAR – dive deep
• Implementation & Performance Evaluation
• Conclusions & Future Work
• Constrained Application Protocol (CoAP) in the final stage of the standardization process targeting specifically IoT applications

• CoAP main features that fulfill application requirements are [1]:
  • Group communication i.e. multicast support
  • Asynchronous message exchanges
  • Proxy and caching capabilities
  • Low overhead
  • Header mapping to HTTP
  • End-to-End Security

Alice (DTLS client) and Bob (DTLS server) establish an authenticated, secure channel (bidirectional information flow) once the handshake completes.
Motivation (2/6)

• CoAP + DTLS features (CoAPs):
  • Group communication i.e. multicast support ✗
  • Asynchronous message exchanges ±
  • Proxy and caching capabilities ✗
  • Low overhead
  • Header mapping to HTTP
  • End-to-End Security
DTLS Recap (2/3)

- Preamble
- MAC
- Adaptation
- IPv6
- UDP
- DTLS Record
- CoAP
- Application Data
- Encrypted and Integrity Protected
- FCS
Motivation (4/6)

- **CoAP + DTLS features (CoAPs):**
  - Group communication i.e. multicast support  
    - x
  - Asynchronous message exchanges  
    - ±
  - Proxy and caching capabilities  
    - x
  - Low overhead  
    - x
  - Header mapping to HTTP  
    - x
  - End-to-End Security  
    - x
Motivation (5/6)

- CoAP + DTLS features (CoAPs):
  - Group communication i.e. multicast support: ✗
  - Asynchronous message exchanges: ±
  - Proxy and caching capabilities: ✗
  - Low overhead: ✗
  - Header mapping to HTTP: ✗
  - End-to-End Security: ✓
• Security that can’t support **basic** application requirements is of no use

• Fundamental design choices of CoAP and DTLS are incompatible
  • (D)TLS targets connection oriented **point-to-point** application flows (Voice over IP, some online games)
  • Only basic request-response mechanism of CoAP could be regarded as connection-oriented. What about:
    • Asynchronous notification and observation of resources with dynamic group membership
    • Caching and integration with the Cloud

• TCP and its three-way handshake (syn, syn-ack, ack) were ruled out from LLNs due to “terrible performance”
  • 3 RTT and 10-15 packet DTLS handshake, with completion time from several seconds to **one minute** (depending on the duty-cycle)?
  • Statements like “it [handshake] is performed only once during the initialization phase and/or later (rarely) for re-handshake” [3] should make us think as if we may as well hard-code logical topologies and interactions

---

OSCAR – concepts behind (1/3)

• **Requirement 1:** Make security features compatible with application requirements, not vice-versa

• **Requirement 2:** Allow E2E security in presence of statefull gateways that do not allow direct communication from the outside and the WSN

• **Requirement 3:** Backwards compatibility with plain DTLS approach, as standardized in CORE WG, to support existing deployments

• **Requirement 4:** High practical value targeting IETF efforts on End-to-End security and Authorization in constrained environments (CORE, DICE and ACE WGs)

• **Requirement 5:** Minimal energy consumption to allow most energy-stringent devices, like GreenNet nodes
**Idea 1:** A stateless security architecture

- Allows caching, eases group communication and asynchronous exchanges
- Solution: Application-level security i.e. Object security (CMS, JOSE)
- Protect from communication-related replay attacks by binding object-security encryption keys with underlying CoAP duplicate detection mechanism
• **Idea 2:** Move the burden of security handshake away from constrained servers
  • Introduce a semi-trusted, non-constrained third party that will do the hard work
  • Constrained servers respond with secured objects (resource representations) regardless of the identity of the client

• **Idea 3:** Jointly approach problems of End-to-End security and Authorization
  • Split confidentiality and authenticity trust domains
  • Confidentiality used to provide access-control for group members
  • Authenticity strongly tied to the originator of the information (individual sensor)
• We use the Producer-Consumer model to provide security
  • Producers: sensors, smart-meters, motion detectors, switches, …
  • Consumers: actuators, mobile devices, collection centers, human users, …

• Producers’ main task is to generate information and to secure it **independently** of possibly many consumers
  • We **decouple** the public-key cryptographic overhead from network communication on the producer side
    • Results in functional simplicity of producers (constrained nodes)
  • Producers update **secured** resources as they are observed in the environment
    • This allows lots of application-specific optimizations to reduce the cryptographic overhead
  • Producers respond to all requests with access-protected resource representations (symmetric encryption)
    • Main processing burden is shifted away from producers (constrained servers)

• Consumers fetch the information either from intermediate proxies, the Cloud, by direct CoAP request/response interaction or they are asynchronously notified of changes (CoAP observe option)
Authorization

Server

Client

GET /Temp

Temp

Resource representation pre-signed with P's private key

On-the-fly symmetric encryption with key derived from access-secret

Temp

Humidity

CO

Location

.well-known/core

Access-secret A

Access-secret B

Access-secret N

Constr. Server P
In summary:

- We heavily use digital signatures to provide authenticity of information tied to individual sensors (source authentication)
  - Surprisingly good performance results in comparison with DTLS-only approach
  - Work on-going to support use-cases where this is not practical
- We use confidentiality to provide capability-based access-control by symmetric encryption
  - Protection against communication related replay-attacks by binding the actual encryption key to the duplicate detection mechanism of CoAP
- Authorization Server(s) in charge of authentication and distribution of appropriate access-secrets
- Implicit compatibility with multicast and caching
- DTLS used for communication with Authorization Servers and to enable backwards compatibility
CoAP + OSCAR features:

- Group communication i.e. multicast support ✓
- Asynchronous message exchanges ✓
- Proxy and caching capabilities ✓
- Low overhead ±
- Header mapping to HTTP ✓
- End-to-End Security ✓
- Authorization and Access Control ✓
Implementation & Performance Evaluation

• Developed object-security library for Contiki
  • Supports encrypted, signed and encrypted/signed object types.
  • Coupled with CoAP to provide cipher negotiation capabilities and replay protection

• Evaluation
  • Two hardware platforms at 21.3 MHz:
    • WiSMote - 16-bit MSP430, 16K RAM, CC2520 radio transceiver
    • GreenNet tags - STM32L, 32K RAM, RF200W radio transceiver
  • We study scalability as a function of number of clients per server
    • Number of simultaneous DTLS sessions is limited due to memory constraints of nodes
    • With a simple application for evaluation purposes, we could fit up to 3 simultaneous sessions for WiSMote. Same number used with GreenNet to have comparable results.
  • Pre-shared key based cipher suite for DTLS using only **symmetric** key operations
  • TinyECC library for OSCAR using secp160r1 elliptic curve for signing
  • Typical 6LoWPAN stack (CoAP, UDP, IPv6, 6LoWPAN, 802.15.4)

• Methodology
  • Each point averaged over five 3-hour runs and plotted with 95% conf. intervals
Server-side Total Energy Consumption

WiSMote

ST GreenNet
Client-side Request/Response Delays

Request/Response RTT (s)

Clients/Session Slots

DTLS-Lithe, STM GreenNet
OSCAR, STM GreenNet
DTLS-Lithe, WiSMote
OSCAR, WiSMote
Client-side Request/Response Energy

![Graph showing Request/Response Energy (mJ) vs. Clients/Session Slots]

- DTLS-Lithe, STM GreenNet
- OSCAR, STM GreenNet
- DTLS-Lithe, WiSMote
- OSCAR, WiSMote
Conclusions & Future Work

• Established E2E security and authorization framework that actually supports application requirements

• Can provide E2E security even in presence of statefull gateways

• Particularly useful for use-cases where high number of clients per-constrained-server is expected
  • Smart city a very good example

• Future work required
  • Use-cases that require streaming where constant digital signing is unfeasible
  • Key management and authorization policies
Hvala lijepa!*
Questions?

*Merci bien!
Backup
Implementation & Performance Evaluation

- Time [s]
  - Wismote
  - STM GreenNet

- Energy [mJ]
  - Wismote
  - STM GreenNet

<table>
<thead>
<tr>
<th></th>
<th>Sign</th>
<th>Verify</th>
<th>Sign</th>
<th>Verify</th>
</tr>
</thead>
<tbody>
<tr>
<td>secp160r1</td>
<td>1</td>
<td>0.5</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>secp192r1</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>2</td>
</tr>
</tbody>
</table>

- Wismote
- STM GreenNet