A Toolkit for Construction of Authorization Service Infrastructure for the Internet of Things (IoT)

Hokeun Kim\textsuperscript{1}, Eunsuk Kang\textsuperscript{1}, Edward A. Lee\textsuperscript{1}, David Broman\textsuperscript{2}

\textsuperscript{1}University of California, Berkeley
\textsuperscript{2}KTH Royal Institute of Technology

IoTDI 2017, Pittsburgh, PA
April 19, 2017
Overview – IoT & Authorization

- Internet of Things
- Authorization (access control)
  - Critical for computer security

Benefits, but also challenges

- Existing security solutions?
  - May work well for some parts of the IoT, but not for the entire IoT!

- Proposed approach – SST
  - SST: Secure Swarm Toolkit
  - An open-source toolkit for building authorization infrastructure for the IoT
  - To address IoT security challenges
Motivation

- Challenges in IoT security[1]

---

Heterogeneity
- Security requirements & resource availability
- Connectivity (wired connections vs. mobile devices)
- System management

---

Motivation (cont'd)

• Challenges in IoT security\(^1\)

- Operation in an open (or hostile) Environment
  - Physical access & wireless access to IoT devices
  - Higher risk of being compromised
  - Must be able to revoke access of compromised IoT devices

Motivation (cont'd)

- Challenges in IoT security\(^1\)

### Scalability

- 28 billion connected devices in 2021
- 15.3 ZB data traffic in 2020
  - 1 ZB (Zetta byte) = 10^9 TB (Terra bytes)

Background: Authorization & IoT

• Authorization
  – Access control
    • "Can I enter the EECS building?"
  – Allowing/denying access to resources
  – Revoking access (e.g., lost ID card)

• Authentication
  – Identifying someone/something
    • "Member of EECS?"
  – Essential for authorization
• Many IoT platforms use TLS (or DTLS\textsuperscript{[2]}) for authentication/authorization
  – E.g., Amazon AWS IoT, OpenIoT\textsuperscript{[3]}, OSCAR\textsuperscript{[4]}, etc.

• TLS (Transport Layer Security, also called SSL/TLS)
  – Underlying security protocol for HTTPS
  – Widely used, very successful for web

Background (cont'd)

- **TLS based on a digital certificate**

  - Certificate issued by CA

- **Challenges with using TLS for the entire IoT**
  - Energy overhead of public-key crypto & certificates
  - Scalability (managing certificates for ~28 billion devices)
  - Revocation of certificates can be problematic\(^1,2\)
  - Limited support for one-to-many communication

• Challenges with applying other security solutions

— Kerberos \[1\]
  • Advantages for access revocation
  • Requires stable connection
  • Centralized architecture

— Security solutions for "Things"
  • E.g., WSN, MANET or swarm devices
  • Assume homogeneous environments
  • Not designed for Internet scale \[2\]

\[1\] C. Neuman et al., 2005. "The Kerberos Network Authentication Service (V5)". RFC 4120
\[2\] Alcaraz et al., 2010. "Wireless sensor networks and the internet of things: Do we need a complete integration?"
Proposed Approach

• SST – Secure Swarm Toolkit
  – An open-source toolkit for authentication/authorization of the IoT (available on [https://github.com/iotauth](https://github.com/iotauth))
Proposed Approach (Cont'd)

• Specific goals of SST

- Integration of existing security solutions (not inventing new ones)
- Locally centralized and globally distributed architecture
- Ease of deployment by local domain experts at a large scale
SST’s Design and Implementation

- **Auth**\(^1\)
  - Locally centralized, globally distributed authentication/authorization entity (software)
  - Java program to be deployed on edge devices\(^2\) (e.g., Intel IoT gateways)

[1] A prototype of Auth has been proposed in Kim et al., 2016. "A Secure Network Architecture for the Internet of Things Based on Local Authorization Entities"
Secure communication accessors

- Software building blocks for securely accessing Auth and the IoT services
- Encapsulate crypto keys & operations
- Help IoT developers who are not security experts

Currently available accessors (in JavaScript)

- We're still at a starting point and working on more accessors!
- For more information, see https://accessors.org
• Example: How SST (Auth and accessors) works

Encrypted with Distribution Key between Auth and Client

I want to use IoT Service!

Session Key

SecureCommClient

Request To Send

Access Response From Service

Client

SecureCommServer

SST

Server

Process Client Message

IoT Service

Respond To Client
• Example: How SST (Auth and accessors) works

• Example: How SST (Auth and accessors) works

OK, Client can access this IoT Service.

Encrypted with *Distribution Key* between Auth and IoT Server

**Session Key**

Request To Send

SecureCommClient

Initiate challenge-response

SecureCommServer

Process Client Message

Respond To Client

Client

Access Response From Service

IoT Service

Auth

TerraSwarm Research Center
• **Example: How SST (Auth and accessors) works**

Protected communication channel using session key and standard cryptography[^2]

[^2]: Followed TLS 1.2’s standard, including sequence number, encrypt-then-MAC
SST for Heterogeneity

• SST’s configuration alternatives

Underlying protocol
- TCP
- UDP

Cached session keys
- K-1: Unlimited
- K-2: Multiple
- K-3: One

Number of session key sharers
- Unlimited
- More than two (broadcasting)
- Two (server-client)

Distribution key
- D-1: No direct key distribution
- D-2: Permanent
- D-3: Updated using public key

Crypto strength & key lifetimes
- C-1: Lightweight & long
- C-2: Strong & short
- C-3: More security guarantees

Session key usage
- S-1: Authentication only
- S-2: Encryption
- S-3: Ephemeral Diffie-Hellman

Effect of knobs will be shown through experiments!

TerraSwarm Research Center
• Limiting damage from compromised entities
  – SST's design to timely *revoke* keys (session & distribution keys)
    • Must always be authorized by Auth
    • Revocation takes effect immediately

  – Even when Client with a valid session key is compromised, Auth can prevent its access to IoT Server!
SST for Scalability

- Shared key support for one-to-many communication (for data scalability)
• Shared key support for one-to-many communication (for data scalability)
• Shared key support for one-to-many communication (for data scalability)
SST for Scalability

- Shared key support for one-to-many communication (for data scalability)

UDP broadcast over a local network

Object (data) security (e.g., for information-centric networks)
SST for Scalability (cont'd)

- Globally distributed Auths (to scale with # IoT devices)
  - Trust relationships without a centralized authority

TerraSwarm Research Center
**Evaluation: Security Analysis**

**Desired Security Properties**
- Confidentiality (of data)
- Message authenticity
- Data integrity

**Threat Model**
- Network attackers
  - Eavesdrop or inject packets
- Compromised IoT Entities
  - Try to break security of others
- No compromised Auths

**Formal Security Model of SST**
- Modeled in Alloy\(^2\) (Model checking tool & language)
- Includes models for Auths, entities and communication messages

**Result:** Formally proven to satisfy the security properties!

---

[1] https://github.com/iotauth/security_analysis
Evaluation: Scalability Analysis

- Auth’s authorization tasks include
  - Communication with IoT entities and Auths
  - Cryptographic operations
  - Accessing Auth’s database (keys, access policy, etc.)

Scalability analysis result:
- Each Auth’s workload is a linear function of “number of entities per Auth”, not “total number of entities in the system”, assuming access activity per entity is fixed.
- In theory, we can always scale with increasing entities by adding Auths accordingly.
Experiments & Results

• **Effect of various configuration alternatives**
  – Estimated energy consumption for **setting up secure connections** between IoT clients & IoT servers
    • Logged crypto operations and captured packets
    • Used energy numbers from UAB[1] and SICS[2]

[1] UAB (Universitat Autònoma de Barcelona), Rifà-Pous and Herrera-Joancomartí. 2011
Estimated energy for an IoT server connected by 16, 32, and 64 clients

- **Public-key crypto**
- **Sym. crypto & MAC**
- **Network comm.**

**Tradeoffs for heterogeneity!**

- More results in our paper!
- (for IoT clients)

**Number of clients**
- 16 Clients
- 32 Clients
- 64 Clients

**Number of allowed cached session keys**
- 1
- ∞

**Underlying Protocol**
- TCP
- UDP

**Distribution key management**
- Updated
- Permanent

**Energy (mJ)**
- 0
- 200
- 400
- 600
- 800
- 1000
- 1200
- 1400
- 1600
- 1800
- 2000

**Note:** TLS was used as a reference and we do not claim that SST is better than TLS.

**Cached session keys**
- K-1
- K-2
- K-3
- Unlimited
- Multiple
- One

**Underlying protocol**
- TCP
- UDP

**Distribution key**
- D-1
- D-2
- D-3
- Updated using public key
- Permanent

**Crypto strength & key lifetimes**
- Lightweight & long
- Strong & short

**Experiments & Results (cont'd)**

**More security guarantees**

**Less energy overhead**

TerraSwarm Research Center
A sender and multiple receivers with different settings

(1) Connections with SSL/TLS
Sender \[\rightarrow\] SSL/TLS \[\rightarrow\] Receiver

(2) Shared Key + secure connections by SST
Sender \[\rightarrow\] SST secure connection \[\rightarrow\] Receiver

(3) Shared key + MQTT message broker
Sender \[\rightarrow\] TCP \[\rightarrow\] Broker \[\rightarrow\] TCP \[\rightarrow\] Receiver

(4) Shared key + UDP broadcast
Sender \[\rightarrow\] UDP broadcast \[\rightarrow\] Receiver
### Experiments & Results (cont'd)

**Estimated energy for a sender to send a 1KB-message to receivers**

<table>
<thead>
<tr>
<th>Network Setting</th>
<th># receivers</th>
<th>Estimated energy (mJ)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TLS</td>
<td>16 Receivers</td>
<td>54.0</td>
</tr>
<tr>
<td>ISC</td>
<td>32 Receivers</td>
<td>48.6</td>
</tr>
<tr>
<td>MB</td>
<td>64 Receivers</td>
<td>3.4</td>
</tr>
<tr>
<td>TLS</td>
<td>16 Receivers</td>
<td>108.1</td>
</tr>
<tr>
<td>ISC</td>
<td>32 Receivers</td>
<td>96.9</td>
</tr>
<tr>
<td>MB</td>
<td>64 Receivers</td>
<td>3.4</td>
</tr>
<tr>
<td>TLS</td>
<td>16 Receivers</td>
<td>216.2</td>
</tr>
<tr>
<td>ISC</td>
<td>32 Receivers</td>
<td>193.5</td>
</tr>
<tr>
<td>MB</td>
<td>64 Receivers</td>
<td>3.4</td>
</tr>
</tbody>
</table>

**TLS:** (1) SSL/TLS  
**ISC:** (2) Individual SST Connections + shared key  
**MB:** (3) MQTT Message broker  
**UB:** (4) UDP broadcast

- More results in our paper! (for sender initialization)

#### Tradeoff example

A sensor node (500mAh/1.5V battery) sending 1KB per minute to 64 receivers

- Expected battery life
  - <10 days with ISC (secure connections by SST)
  - 625 days with UB (UDP broadcast)
Conclusions

• Benefits of SST: Secure Swarm Toolkit
  – Authorization for a wide range of IoT from sensor nodes to safety-critical systems
  – Enable Internet-scale deployment with increasing connected devices and traffic
  – Help deployment of IoT security solutions by system designers with moderate knowledge in security
  – Possible integration with other IoT-related efforts (e.g., securing CoAP[1])

Conclusions (cont'd)

• Future work
  – Mitigation against availability attacks (e.g., Denial-of-Service attacks)
  – Detection of malicious behavior of compromised IoT entities or Auth
  – Further studies on usability of SST
  – Efficient initial setup of SST (e.g., registering IoT devices with Auth)

• For further information
  – https://github.com/iotauth