Grant Agreement Number: **248113/O70**

Project acronym: **IoTSec**

Project full title:

Security in IoT for Smart Grids

D 2.1.2
Privacy-preservation framework

Due delivery date: M12

Actual delivery date: M12

Organization name of lead participant for this deliverable:

Simula Research Laboratory

Dissemination level		
PU	Public	
RE	Restricted to a group specified by the consortium	**X**
CO	Confidential, only for members of the consortium	

Deliverable number:	D 2.1.2
Deliverable responsible:	Ming-Chang Lee
Work package:	WP2
Editor(s):	Ming-Chang Lee

Author(s)	
Name	**Organisation**
Ming-Chang Lee	Simula
Yan Zhang	UIO and Simula

Document Revision History			
Version	**Date**	**Modifications Introduced**	
		Modification Reason	**Modified by**
V01	30.09.2016		

1 ABSTRACT

In this report, we introduce an executable model for modeling smart homes based on Real-Time ABS, which is a formal executable language for modeling distributed systems. The purpose is to model the communication protocols of smart-home devices and how these devices communicate with each other. By modeling all above-mentioned properties, we are able to see whether homeowners' privacy will be exposed or not when some traffic analysis tool, such as eavesdropping, are used.

2 EXECUTIVE SUMMARY

This document summarizes deliverable D2.1.2 of project 248113/O70 (IoTSec), a research project supported by the Research Council of Norway (RCN). Full information on this project is available online at http://cwi.unik.no/IoTSec:Home

List of Authors

Ming-Chang Lee (Simula)

Yan Zhang (UIO and Simula)

I. TABLE OF CONTENTS

II. TABLE OF FIGURES AND TABLES

3 INTRODUCTION

Currently, smart environments such as smart homes for the activities of daily living (ADL) are widespread since the sensor devices get smarter, smaller and cheaper recently [1]. These sensor devices can sense, monitor, or control our homes, our cars, or our bodies. Sensor data created by these sensor devices can be collected and sent to data centers and to be analyzed. Typically, a smart home, which is one of Internet of Things (IoT) applications, have various sensors to monitor the home environment such that homeowners' living experience can be improved.

As the popularity of smart home increases, security and privacy are two main issues. It is necessary to ensure that devices of smart homes are secure and do not reveal homeowners' privacy. However, commercial off-the-shelf (COTS) IoT devices do not have proper authentication or data encryption. Therefore, these devices might suffer Denial of Service (DoS) attacks, eavesdropping, or other malicious attacks [2]. Furthermore, even though people use encryption technologies to protect privacy, these technologies cannot protect the privacy of the sender or receiver, especially with regard to the communication patterns such as how long, how often, what time, who, data formats, and the length of message, present in the network traffic of each day [3]. These communication patterns offer useful information for potential attackers to discover homeowners' behavior or privacy. Using some traffic analysis tools, potential attackers might be able to observe whether people live in their homes or not, what the layouts of their homes are, the location of a specific room, and so forth.

In this report, we present an executable model to show that homeowners' privacy might be exposed by using traffic analysis. Our model targets on modeling the communication protocols of smart-home devices and how these devices communicate with each other. The model is written in the Real-Time Abstract Behavioral Specification language (ABS for short), which is a formal executable modeling language focusing on modeling of distributed systems and virtualized software. The Real-Time ABS language combines functional and imperative programming styles with a Java-like syntax and a formal semantics.

The rest of this report is organized as follows. Section 2 shows the background and related work of this report. Section 3 presents the details of our model. Section 4 concludes the report and outlines our future work.

4 BACKGROUND AND RELATED WORK

Yoshigoe et al. [2] studied the privacy of smart-home devices in home residence settings and presented how homeowner's privacy could be compromised via network traffic analysis. The authors used Samsung *SmartThings* products [4][5] as a case study. Figure 1 illustrates the architecture of a smart home with *SmartThings* devices. The authors collected the following four traffic patterns between the *SmartThings Hub* and the Cloud Server:

1. **No Active Events**: After the SmartThings Hub was turned on, as long as no active events take place, the authors have observed a regular traffic pattern, which is called the default traffic pattern. The default traffic pattern consists of a TCP *keepalive* (KA) packet (60 bytes), the associated ACK packet (54 bytes), a hub KA packet (123 bytes), and the associated ACK packet for the hub (123 bytes). The TCP KA packets and Hub KA packets are sent periodically every 10 seconds and 60 seconds, respectively.
2. **Door Open/Close**: When the Hub receives a message from the door sensor that is activated, implying the corresponding door is opened/closed, a particular traffic pattern can be obtained. More specifically, the Hub sends a 123-byte packet to the Cloud Server and the Cloud Server responds the Hub with a packet of the same size. This handshaking will repeat twice. This event might reset the timers for sending both TCP KA packets and Hub KA packets.
3. **Motion Event**: When the Hub receives a message from a motion sensor that is triggered, a particular traffic pattern can be obtained as follows: When a motion sensor senses a movement, the Hub sends a 123-byte packet to the Cloud Server and the Cloud Server responds the Hub with a packet of the same size. This handshaking will repeat for three times, and it might reset the timers for sending both TCP KA packets and Hub KA packets as well.
4. **LED Control**: When a homeowner turns on/off his/her LED bulb through the SmartThings application on his/her smartphone, a particular traffic pattern can be obtained as below: When a homeowner remotely turns on/off a LED bulb by using his/her smartphone application. The Cloud Server will first send a 139-byte packet to the Hub and the Hub replies the Cloud Server with a 123-byte packet. The handshaking will repeat for five times, except that the fourth packet sent from the Hub is 139 bytes, instead of 123 bytes. Similar to the other three events, this event might reset timers for sending both TCP KA packets and Hub KA packets.

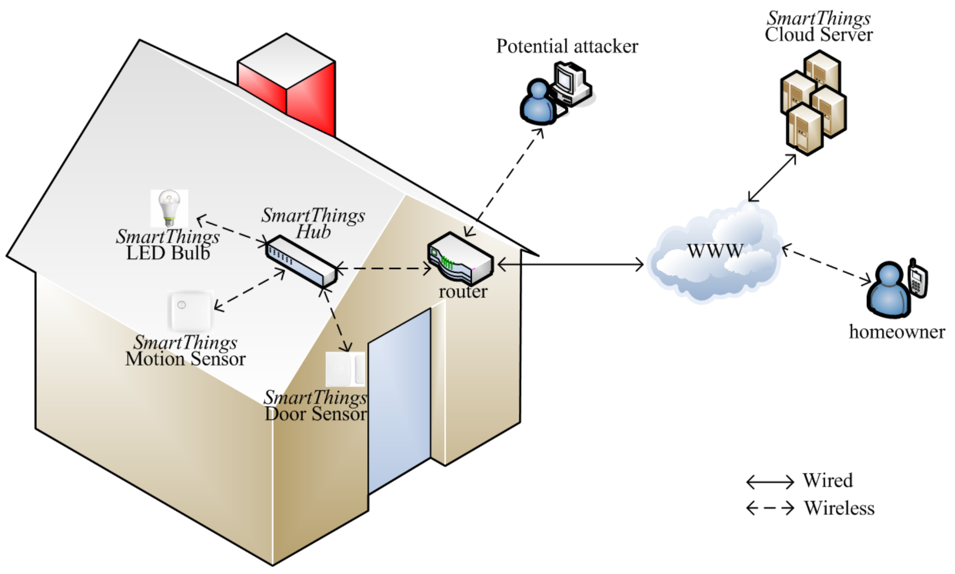

Figure 1 - The architecture of a smart home with *SmartThings* devices.

5 OUR MODEL

In the section, we model the communications of each component in Figure 1. We limit our code presentation to the main interfaces of our model to simplify the description. Our model consists of the following six interfaces:

```
interface Hub;
interface CloudServer;
interface Door;
interface Motion;
interface Phone;
interface Bulb;
```

The SmartThings Hub implements interface Hub, which has the following six methods:

```
interface Hub{
  Unit sendTCPKA(String hub, CloudServer server, String tcpKA, Int size);
  Unit sendHubKA(String hub, CloudServer server, String hubKA, Int size);
  Pair<String, Int> receiveDeviceKA(String noDevice, CloudServer server, String deviceKA, Int size);
  Pair<String, Int> sendRemoteControlKA(String noDevice, Bulb bulb, String deviceKA, Int size);
  Unit sendApplicationFollowup(String noDevice, Server CloudServer, String application, Int size);
  Unit receiveApplicationFollowup(String noDevice, String ACK, Int size);
}
```

Method sendTCPKA is used to model sending a message to the Cloud Server every 10 seconds from the SmartThings Hub. The message includes the ID of the Hub, the information of the Cloud Server, a TCP KA packet (denoted by tcpKA), and the associate packet size (60 bytes).

Method sendHubKA is used to model sending a message to the Cloud Server every 60 seconds from the Hub. The message includes the ID of the Hub, the information of the Cloud Server, a Hub KA packet, and the corresponding packet size (123 bytes).

Method receiveDeviceKA allows the Hub to receive a KA packet from a smart device. The packet content includes the ID of the device, the information of the Cloud Server, the KA packet sent from the device, and the corresponding size.

Method sendRemoteControlKA is used to send a message to the LED Bulb from the Hub. The message includes the ID of the device, the information of the Bulb, the KA packet sent from the device, and the corresponding size.

Method sendApplicationFollowup is used to send a message to the Cloud Server from the Hub. This message includes the ID of the device, the information of the Cloud Server, an application follow-up packet sent from the Hub, and the associate size (60 bytes).

Finally, method receiveApplicationFollowup is used to receive an ACK message from a smart device. This ACK message includes the ID of the device, the KA ACK packet sent from the device, and the corresponding size.

The Cloud Server implements interface CloudServer, which has the following six methods:

```
interface CloudServer {
  Pair<String,Int> receiveTCPKAACK(String hub, String ack, Int size);
  Pair<String,Int> receiveHubKAACK(String hub, String ack, Int size);
  Pair<String,Int> receiveDeviceKAACK(String hub, String ack, Int size);
  Pair<String, Int> sendRemoteControlKA(String noDevice,Bulb bulb,Hub hub,String deviceKA, Int size);
  Unit receiveApplicationFollowup(String noDevice, String ack, Int size);
  Unit sendApplicationFollowup(String noBulb, Hub hub, String application, Int followup_size);
}
```

Method receiveTCPKAACK is used to model replying a message to the Hub every 10 seconds from the Cloud Server. The message includes the ID of the Hub, a TCP KA ACK packet (denoted by ack), and the associate packet size (54 bytes).

Method receiveHubKAACK is used to model replying a message to the Hub every 60 seconds from the Cloud Server. The message includes the ID of the Hub, a Hub KA ACK packet, and the corresponding packet size (123 bytes).

Method receiveDeviceKAACK allows the Cloud Server to receive a KA ACK packet from the Hub. The packet content includes the ID of the Hub, the KA ACK packet sent from the device, and the corresponding size.

Method sendRemoteControlKA sends the message to control the LED Bulb. The message includes the ID of the device, the information of Bulb, the information of the Hub, the KA packet sent from the device, and the associate size.

Method receiveApplicationFollowup is used to receive an ACK message from the device, this ACK message includes the ID of the device, a KA ACK packet sent from the device, and the corresponding size.

Finally, method sendApplicationFollowup is used to send an application follow-up message to the Hub. The application follow-up message includes the ID of the Bulb, the information of the Hub, an application follow-up packet, and the associate size (60 bytes).

The Door Sensor implements interface Door with a sendDoorKA method to send the ID of the Door, the information of the Hub, the information of the Cloud Server, a Door KA packet (denoted by doorKA), and the corresponding size (123 bytes).

```
interface Door{
  Unit sendDoorKA(String noDoor, Hub hub, CloudServer server, String doorKA, Int size);
}
```

The Motion Sensor implements interface Motion with a sendMotionKA method to send the ID of the Motion, the information of the Hub, the information of the Cloud Server, a Motion KA packet, and the associate size (123 bytes).

```
interface Motion{
  Unit sendMotionKA(String noMotion, Hub hub, CloudServer server, String motionKA, Int size);
}
```

The Smart Phone implements interface Phone with a sendBulbKA method to send the ID of the Bulb, the information of the Bulb, the information of the Hub, the information of Cloud Server, a Bulb KA packet, and the corresponding size (139 bytes).

```
interface Phone{
  Unit sendBulbKA(String noBulb, Bulb bulb, Hub hub, CloudServer server, String bulbKA, Int size);
}
```

The LED Bulb Sensor implements interface Bulb with a receivedRemoteControlKA method to send the ID of the Bulb, a Bulb KA packet, and the associated size.

```
interface Bulb{
  Pair<String,Int> receivedRemoteControlKA(String noBulb, String bulbKA, Int size);
}
```

Based on the above modeling, we are able to capture different traffic patterns between the SmartThings Hub and the Cloud Server under different events. In this way, as long as a packet traffic matches one of these modeled traffic pattern, the corresponding event and the corresponding smart devices can be identified, implying that the corresponding homeowner's privacy are exposed.

6 CONCLUSION AND FUTURE WORK

In this report, we have built an executable model to simulate different traffic patterns between smart devices, the SmartThings Hub, and the Cloud Server. We found that smart devices in a smart home will expose homeowners’ privacy by using traffic pattern analysis to distinguish different traffic patterns. In our future work, we would like to extend this model to see if the layout of a smart home can be inferred by analyzing traffic data sent from smart devices and smart sensors. In addition, we would like to accordingly propose a privacy-preservation framework to avoid attackers from guessing/obtaining/inferring homeowners’ privacy or behaviors.

7 REFERENCES

[1] Park, H., Basaran, C., Park, T., & Son, S. H. (2014), "Energy-Efficient Privacy Protection for Smart Home Environments Using Behavioral Semantics," *Sensors*, *14*(9), 16235-16257.

[2] Yoshigoe, K., Dai, W., Abramson, M., & Jacobs, A. (2015, December), "Overcoming invasion of privacy in smart home environment with synthetic packet injection," In *TRON Symposium (TRONSHOW), 2015* (pp. 1-7). IEEE.

[3] J. Raymond, "Traffic analysis: Protocols, Attacks, Design Issues, and Open Problems," *Designing Privacy Enhancing Technologies*, pp. 10-29, Springer Berlin Heidelberg, 2001.

[4] Security Announcement from SmartThings, https://community.smartthings.com/t/security-announcement-from-smartthings/10950, 2015.

[5] SmartThings, https://www.smartthings.com/ (2016.09.29)