symmetry

MDPI

Article

Towards a Formal IoT Security Model

Tania Martin *, Dimitrios Geneiatakis †, Ioannis Kounelis †, Stéphanie Kerckhof and Igor Nai Fovino

European Commission, Joint Research Centre (JRC), 21027 Ispra, Italy;
dimitrios.geneiatakis@ec.europa.eu (D.G.); ioannis.kounelis@ec.europa.eu (I.K.);
stephanie.kerckhof@ec.europa.eu (S.K.); igor.nai-fovino@ec.europa.eu (I.N.F.)
* Correspondence: tania.martin@ec.europa.eu
† These authors contributed equally to this work.

Received: 10 July 2020; Accepted: 2 August 2020; Published: 5 August 2020

Abstract: The heterogeneity of Internet of Things (IoT) systems has so far prevented the definition of adequate standards, hence making it difficult to compare meaningfully the security degree of diverse architectural choices. This task can be nonetheless achieved with formal methodologies. However, the dedicated IoT literature shows no evidence of a universal model allowing the security evaluation of any arbitrary system. Based on these considerations, we propose a new model that aims at being global and all-encompassing. Our model can be used to fairly analyse the security level of different IoT systems and compare them in a significant way. It is designed to be adaptive with realistic definitions of the adversary's (1) actions of interacting with IoT systems; (2) capabilities of accessing the data generated by and exchanged in IoT systems with established rules; and (3) objectives of attacking IoT systems according to the four recognised security properties of confidentiality, integrity, availability and soundness. Such a design enables the straightforward characterization of new adversaries. It further helps in providing a fine-grained security evaluation of IoT systems by either accurately describing attacks against the analysed systems or formally proving their guaranteed level of security.

Keywords: IoT; security; cryptographic model

1. Introduction

In the domain of computer science and symmetry, the development of new types of sensors and actuators supported by network connectivity is shaping the concept of the Internet of Things (IoT). This technology is completely changing the users' approach and use of the cyber-space. It breaks down the barriers between our physical daily world and the digital reality, enabling new services and paving the way towards a full digital inclusion. IoT can support a variety of applications and services in diverse domains such as healthcare, home automation, security, and vehicles. These services are built on dynamic heterogeneous architectures, based on the symbiosis of various elements (i.e., sensors, connections, and applications) with different criticalness levels.

The novelty, heterogeneity, and complexity of such systems involve however a broad and unexplored attack surface that might lead to serious consequences for the users as well as for the companies. For instance, the study of [1] demonstrated that a malicious entity is able to switch on or off the smart bulbs of an IP-based light system, or even to brick them. This can bring unexpected electrical consumption peaks in highly populated areas. A more critical incident is the IoT-based Mirai botnet that launched a DDoS attack against Dyn, a major DNS provider, and caused a significant breakdown of many famous web services, such as Twitter, GitHub, PayPal, Amazon, Netflix, and Spotify. Additionally, IoT systems often generate opaque and huge flows of information. This, combined with

the shift of the control paradigm of the devices from local to virtual and/or remote, implies new security concerns for the users such as leakage, theft, or undesired use of sensitive data.

In order to mitigate those risks, defining consolidated standards and performing deep analysis on the security level of IoT architectures becomes a necessity. A step towards this goal consists in building a well-established formal security model that could be used as a tool to make fair analyses, comparisons and security proofs of different types of IoT systems. Lots of papers related to the security in IoT systems have been published recently. Yet they present models that are usually threat-based oriented (i.e., validating the resistance of a system to a finite set of known attacks), strongly linked to very specific application scenarios, or focused on the devices as individual entities rather than as a complete system. In particular, the threat-based model of [2] is the closest reference related to formal security frameworks in the IoT field. However, it does not allow to formally prove security properties of an IoT system. Looking from the Internet protocols perspective, some automated tools have been proposed to validate their security and applications [3,4]. Existing solutions based on attack graphs [5] and similar approaches [6] require the prior knowledge of attacks' vectors. Yet, to the best our knowledge, literature in this domain with focus on IoT is very limited [2,7]. For instance in [7], authors introduce a dedicated model for analysing smart meters systems. Regarding RFID technology, many cryptographic security models related to privacy already exist, as depicted in the survey [8]. Especially, the model presented in [9] is able to compare RFID systems appropriately, considering various system architectures. To cover this gap, we propose a formal model for evaluating whether or not fundamental security primitives are provided by IoT solutions. Additionally, our model aims to be all-encompassing, meaning that it is able to compare the security performances amongst different IoT systems. Thus, access control models for IoT such as [10–14] are out of the scope of this work as they are considered as complementary security services. However, we are planning to extend our model for checking access control models for IoT in a future work.

Our main contribution is the proposal of a new model that expands the RFID privacy model of [9] to the security properties of the IoT world. First, we provide a clear description of an IoT system and its composition (devices, a backend system and users). We adapt the formal definitions of the RFID building blocks regarding the initialisation procedure and protocol(s) given in [9] to IoT systems. We generalise the definition issued in [9] of the potential adversaries so that their attacks can be carried out on IoT systems, while maintaining the same notion of adversary class representing the adversary's capabilities during an attack. Finally, we define the attacks objectives by the means of unique experiments that formalise the four well-known security properties of confidentiality, integrity, availability, and soundness. As our model aims at being lightweight and easily usable, we additionally demonstrate its practicality with the security analysis of two different IoT systems.

The rest of the paper is organised as follows. Section 2 overviews the general architecture of an IoT system and Section 3 introduces a formalization for it. Section 4 specifies the actions an adversary can perform, and the data she can obtain from these interactions. The security properties of the model are formalised in Section 5. The security analyses of two real-life IoT systems are provided in Sections 6 and 7. Finally, Section 8 concludes the paper and presents the future research.

2. Overview of IoT Systems

An IoT system is the interconnection among physical objects with cyber services. Without loss of generality, it is composed of three main entities: IoT devices, a backend system with which the IoT devices interact with, and users that handle the devices and/or interact with the backend system. We provide a high level description of these three elements below.

2.1. IoT Devices

Currently there is not a common definition for an IoT device, thus we assume that an IoT device can be either a sensor or a hub. A sensor is usually a device with a small processing capacity that performs various measurements or detects patterns (e.g., thermometer, motion sensor, or camera).

It is supported by a firmware for its operations, although some sensors in the market run lightweight versions of well-known operating systems. Due to the sensor's limited capabilities, a hub is usually needed to provide additional services to the sensor, such as data storage or Internet connectivity. Sometimes, a sensor and a hub can coexist in one single device.

2.2. Backend System

Most of IoT systems nowadays have a backend system to support and enhance the provided services. Its purpose is twofold. First, it enhances the devices' capabilities by supporting extra features and services, such as additional storage or advanced user interface with more configurable options. Secondly, it provides to the user continuous communication with the IoT system. In fact, the backend system can be seen as a cloud service. Such a service is usually accessible through a website, where the user has to first authenticate in order to get access. Such access can also be obtained through a mobile application that, in general, automatically detects if the access to the backend is needed or not, depending on the user's location.

2.3. Users

They can interact with IoT devices and manage the system through different platforms such as PCs, smartphones, or tablets. Depending on the system used, local and/or remote interactions with the IoT system are provided. For instance, in case of remote management, all the information is forwarded to the hub through the cloud service, while, if the user is acting from the same network where the hub is installed, the traffic is routed directly to it and thus no Internet access is needed.

All these entities communicate together using different protocols depending on the choices of the devices' manufacturers. The most popular ones are Zigbee, Z-wave, Wi-Fi. Some IoT systems can also involve devices that support low range communications such as RFID, NFC, or Bluetooth.

3. Formalisation of IoT Systems

In order to analyse IoT systems from a security perspective, we introduce a model that will formalise our analysis. In particular, we describe the building blocks of an IoT system, namely the initialisation procedures to set up a system and the protocol(s) executed by the entities of the system, and the IoT data sets that can be extracted from these building blocks (namely transcripts and snapshots). All the definitions of these notions are built on those of [9].

3.1. Initialization Procedures

An IoT system is setup by a procedure InitSystem that (a) generates the public and private values of the system depending on a security parameter λ; and (b) initialises the backend.

To allow a dynamic generation of IoT devices, a procedure CreateDevice is defined and called aside of InitSystem. IoT devices can potentially be setup with unique data and must consequently be registered in the system to be recognized afterwards. This action is not necessarily performed when the IoT device is created, and thus requires the definition of an independent procedure RegisterDevice. For instance, the physical action of a user may be involved, such as pressing a button on the hub.

The three procedures determine each entity's initial internal state, which is the data stored in each entity's memory. The backend internal state may furthermore contain the system database.

An example of such initialisation procedures is the Simple Service Discovery Protocol (SSDP) [15], which is supported by many IoT manufacturers. It enables the transparent configuration of IoT devices in a plug-and-play mode that requires minimum user interactions.

3.2. Protocols

The behaviours and interactions of the system entities are described through protocols. Each of them determines the actions (e.g., computation, random generation), interleaved by transitions, that the entities have to perform to reach a given objective.

For instance, a protocol can be a communication protocol where a sensor sends to the hub the real-time measured temperature of a room. Or a protocol can simply be the process of a camera which starts recording when it detects movements in its surroundings.

3.3. Transcripts

The subset of actions performed by an entity $\mathcal{X}$ during a protocol execution is called algorithm. During an algorithm execution, $\mathcal{X}$ may exchange several data related to its IoT nature with the external world, typically with other involved entities. This external view of an algorithm execution forms a data set called transcript. It can be enriched by auxiliary information extracted from the activation of transitions, e.g., the reception/emission time of a message or its recipient/issuer.

Transcripts are indexed with a counter incremented each time $\mathcal{X}$ engages in a new algorithm execution. $\pi^{i}_{\mathcal{X},\text{PROT}}$ denotes the transcript of the i^{th} execution of $\mathcal{X}$'s algorithm related to a protocol PROT.

3.4. Snapshots

Each algorithm execution may further modify the entity internal state. It is possible to capture the information contained in an entity internal state at a given time (e.g., retrieving the IoT data stored on a sensor by tampering it). Each capture defines a data set called snapshot.

Snapshots are also incrementally indexed, and $\varepsilon^{i}_{\mathcal{X}}$ denotes the i^{th} snapshot of $\mathcal{X}$'s internal state.

4. IoT Adversary

To analyse the security level of an IoT system, we introduce the notion of adversary, i.e., a malicious entity whose final objective is to attack the system. We adapt the adversary model introduced in [9], originally defined for RFID systems, to the IoT context.

In order to define an appropriate adversary, we need to use the notion of oracles and selectors. An oracle is a tool used by an adversary to simulate the interactions of an IoT system and collect data. A selector is a tool used by an adversary to access the collected data. We also need to specify the restrictions that can be applied to the use of the oracles and selectors by the adversary when she plays/interacts with the system in order to attack it. Finally, the combination of oracles, selectors and restrictions permits the definition of all the potential adversary classes that might attack an IoT system.

4.1. Oracles

The following oracles can be carried out on an already initialised IoT system. They are divided in three families, according to their functions.

(i) Oracles that dynamically add IoT devices to the system.

- $\mathcal{O}^{\text{CreateDevice}}() \rightarrow \mathcal{X}$: executes the CreateDevice procedure, and returns the label $\mathcal{X}$.
- $\mathcal{O}^{\text{RegisterDevice}}(\mathcal{X}) \rightarrow \emptyset$: executes the RegisterDevice procedure on the device $\mathcal{X}$.

  When the system is just initialised, there is no IoT device yet defined, only the backend and the public and private values of the system. Consequently, the two previous oracles are used to add devices to the analysed system.

(ii) Oracles that completely or partly execute a protocol.

- $\mathcal{O}^{\text{Execute}}(\text{PROT}, \mathcal{X}_1, \ldots, \mathcal{X}_\alpha) \rightarrow (\pi^{i_1}_{\mathcal{X}_1,\text{PROT}}, \ldots, \pi^{i_\alpha}_{\mathcal{X}_\alpha,\text{PROT}})$: executes the protocol PROT between the entities $(\mathcal{X}_1, \ldots, \mathcal{X}_\alpha)$, fills up and outputs their transcripts $(\pi^{i_1}_{\mathcal{X}_1,\text{PROT}}, \ldots, \pi^{i_\alpha}_{\mathcal{X}_\alpha,\text{PROT}})$.

This oracle reduces the adversary's capabilities to an eavesdropper, while the two following ones allow the splitting up of the previous oracle, and can be used to represent an active adversary that controls all the steps and transitions of a protocol execution.

- $\mathcal{O}^{\text{LAUNCH}}(\text{PROT}, \mathcal{X}) \rightarrow \pi^i_{\mathcal{X},\text{PROT}}$: makes $\mathcal{X}$ launch a new execution of the protocol PROT, fills up and outputs the transcript $\pi^i_{\mathcal{X},\text{PROT}}$ when all the actions related to the oracle query are performed.
- $\mathcal{O}^{\text{SEND}}(\text{PROT}, \mathcal{X}, m) \rightarrow \pi^i_{\mathcal{X},\text{PROT}}$: sends a message m to $\mathcal{X}$, fills up and outputs the transcript $\pi^i_{\mathcal{X},\text{PROT}}$ when all the actions related to the oracle query are performed.

(iii) Oracle that captures information on the system.

- $\mathcal{O}^{\text{CORRUPT}}(\mathcal{X}) \rightarrow \varepsilon^i_{\mathcal{X}}$: outputs the i^{th} snapshot $\varepsilon^i_{\mathcal{X}}$ of $\mathcal{X}$'s internal state.

This oracle formalises a powerful capability that can be given to an adversary, by making her able to corrupt a device and to capture all the information contained in the device at a given time.

4.2. Selectors

With the oracles, the adversary collects transcripts and snapshots. The selectors allow her to read the different information contained in these data sets. There is no exhaustive list of the information that is created/used/shared in an IoT system. This section simply presents the most common ones.

Given a selector s, π.s (resp. ε.s) denotes the information accessible through s contained in the transcript π (resp. the snapshot ε).

(i) Selectors related to transcripts.

- msg: ability to extract the messages sent over the communication channels.
- result: ability to extract the result of the protocol (e.g., **success** or **failure**).
- timer: ability to extract the execution time of a device.

(ii) Selectors related to snapshots.

- eeprom: ability to extract the EEPROM memory of a device at a given time.
- ram: ability to extract the RAM memory of a device at a given time.

4.3. Restrictions

Some restrictions can be applied to the use of the oracles when the adversary plays with the system. For instance, one restriction forces the use of $\mathcal{O}^{\text{REGISTERDEVICE}}$:

- `Reg`: all the devices must be registered.

Some restrictions on the use of $\mathcal{O}^{\text{CORRUPT}}$ can also be applied:

- `NonCorrSensor`: no sensor can be corrupted;
- `NonCorrHub`: no hub can be corrupted.

Restrictions related to the selectors and entities can also be applied:

- `Device`: the selectors can only be called on transcripts performed by IoT devices, i.e., sensors and hubs;
- `Internet`: the selectors can only be called on transcripts performed by entities having Internet connection.

4.4. Adversary Classes

All the different oracles, selectors and restrictions allow to explicitly define the adversary's capabilities during the game (called experiment in what follows) performed with the targeted IoT system to attack it. This is done with the concept of adversary class.

Definition 1 (Adversary class [9])**.** *An adversary class P is defined by three sets O, S, R, and is denoted by the 3-tuple* (O, S, R)*, where:*

- *O is the set of available oracles;*
- *S is the set of available selectors;*
- *R is the set of restrictions regarding the use of the available oracles, selectors, and entities during the experiment.*

Several classical adversary classes against IoT system can be defined. The goal is not to provide an exhaustive list, but to highlight the most intuitive ones. For instance, a security model for IoT systems should at least consider two types of adversary, internal and external, that can act maliciously either on a passive or an active way depending on their goal.

On the one hand, the first category consists of INSIDER adversaries that are located in the close range of an IoT system (e.g., inside a smart home); such adversaries can interact directly with every legitimate device in their surroundings. EAVESDROPPER adversaries are INSIDER with limited capabilities (as explained in [9], the adversary classes can be (partially) ordered. For instance, an INSIDER adversary is stronger than an EAVESDROPPER one): they can only listen to communications between the legitimate devices. On the other hand, EXTERNAL adversaries do not have physical access to the devices; they can only interact with the devices having an Internet connection (e.g., hub or backend system). These three categories of adversaries can be formally defined by the following classes. In order to lighten the notations, let us denote $\mathcal{O}_{\text{basic}} = \{\mathcal{O}^{\text{CreateDevice}}, \mathcal{O}^{\text{RegisterDevice}}, \mathcal{O}^{\text{Execute}}\}$ the set of basic oracles.

- $\mathsf{EAVESDROPPER} = (\mathcal{O}_{\text{basic}}, \{\mathsf{msg}, \mathsf{result}\}, \{\texttt{Reg}, \texttt{Device}\})$.
- $\mathsf{INSIDER} = (\mathcal{O}_{\text{basic}} \cup \{\mathcal{O}^{\text{Launch}}, \mathcal{O}^{\text{Send}}\}, \{\mathsf{msg}, \mathsf{result}\}, \{\texttt{Reg}, \texttt{Device}\})$.
- $\mathsf{EXTERNAL} = (\mathcal{O}_{\text{basic}} \cup \{\mathcal{O}^{\text{Launch}}, \mathcal{O}^{\text{Send}}\}, \{\mathsf{msg}, \mathsf{result}\}, \{\texttt{Reg}, \texttt{Internet}\})$.

As a simple but concrete example, an EAVESDROPPER adversary can use the oracles $\mathcal{O}^{\text{CreateDevice}}$, $\mathcal{O}^{\text{RegisterDevice}}$, $\mathcal{O}^{\text{Execute}}$, the selectors msg and result, all with the restrictions `Reg` and `Device`.

5. Security Properties

After the formalisation of IoT systems and the definition of the potential adversary classes, we define here attack objectives through four security properties, each one linked to a specific experiment. In fact, an experiment formally represents the game performed by an adversary to undermine the related security property expected by an IoT system. In order to clearly understand the experiments and their corresponding security properties, some notations must be specified as follows.

- $\mathcal{S}$ is an IoT system of global security parameter λ.
- $\epsilon(.)$ is a negligible function.
- $\mathcal{A}_P$ is the adversary $\mathcal{A}$ belonging to the class P playing an experiment *Exp*.
- $\mathcal{C}$ is the challenger, i.e., the honest entity that determines a kind of riddle that must be answered by $\mathcal{A}_P$ at the end of *Exp*.

The goal of $\mathcal{A}_P$ is to win *Exp* with non-negligible probability, in comparison to a dumb adversary who is considered to always perform random interactions during the experiment.

5.1. Confidentiality

This notion is linked to attacks aiming at retrieving confidential information. An example of such a situation is a hospital using IoT for medical follow-up. The system is composed of (1) a centralized backend containing the medical data of all the patients; (2) IoT wristbands that are worn by the patients during their stay at the hospital; and (3) IoT tablets that scan the wristbands and provide to the doctors the medical data related to the corresponding patients. In this example, an adversary may want to retrieve the medical data of the patients wearing an IoT wristband. The confidentiality experiment is described in Figure 1.

Experiment $Exp^{\text{Confident}}_{\mathcal{S},\mathcal{A}_P}(\lambda)$

1. $\mathcal{C}$ runs $\text{INITSYSTEM}(1^\lambda)$.
2. $\mathcal{C}$ defines the set of confidential information CI.
3. $\mathcal{A}_P$ interacts with the system $\mathcal{S}$, limited by her class P.
4. $\mathcal{A}_P$ outputs some information I.

$Exp^{\text{Confident}}_{\mathcal{S},\mathcal{A}_P}(\lambda)$ succeeds if I is (part of) CI.

Figure 1. Confidentiality experiment.

The confidentiality property is thus as follows.

Definition 2. *Confidentiality: An IoT system $\mathcal{S}$ is said to be P-confidential for a specific set CI of confidential information if:*

$$\forall \mathcal{A}_P \in P\colon \Pr\left(Exp^{\text{Confident}}_{\mathcal{S},\mathcal{A}_P}(\lambda) \text{ succeeds }\right) \leq \epsilon(\lambda).$$

5.2. Integrity

This notion is linked to attacks aiming at modifying (e.g., adding, changing or removing) part of the critical information contained in a system. Following the hospital example, an adversary may want to erase data of some patients, preventing them of being treated. The integrity experiment is described in Figure 2.

Experiment $Exp^{\text{Integrity}}_{\mathcal{S},\mathcal{A}_P}(\lambda)$

1. $\mathcal{C}$ runs $\text{INITSYSTEM}(1^\lambda)$.
2. $\mathcal{C}$ defines the set of critical information CRI.
3. $\mathcal{A}_P$ interacts with the system $\mathcal{S}$, limited by her class P.

$Exp^{\text{Confident}}_{\mathcal{S},\mathcal{A}_P}(\lambda)$ succeeds if (part of) CRI has been modified in $\mathcal{S}$ by $\mathcal{A}_P$.

Figure 2. Integrity experiment.

The integrity property is thus as follows.

Definition 3. *Integrity: An IoT system $\mathcal{S}$ is said to be a P-integrity system for a specific set CRI of critical information if:*

$$\forall \mathcal{A}_P \in P\colon \Pr\left(Exp^{\text{Integrity}}_{\mathcal{S},\mathcal{A}_P}(\lambda) \text{ succeeds }\right) \leq \epsilon(\lambda).$$

5.3. Availability

This notion is linked to Denial of Service (DoS) attacks. In the hospital example, an adversary may want to prohibit the IoT tablets to communicate with the rest of the system, thus blocking the doctors to do their job and jeopardizing the patients' health. The availability experiment is described in Figure 3.

Experiment $Exp^{\text{Avail}}_{\mathcal{S},\mathcal{A}_P}(\lambda)$

1. $\mathcal{C}$ runs INITSYSTEM(1^λ).
2. $\mathcal{A}_P$ interacts with the system $\mathcal{S}$, limited by her class P.
3. $\mathcal{A}_P$ designates a specific IoT device $\mathcal{D}$, and makes it interact with $\mathcal{S}$.

$Exp^{\text{Avail}}_{\mathcal{S},\mathcal{A}_P}(\lambda)$ succeeds if $\mathcal{D}$ is no longer working within $\mathcal{S}$.

Figure 3. Availability experiment.

The availability property is thus as follows.

Definition 4. *Availability: An IoT system $\mathcal{S}$ is said to be P-available if:*

$$\forall \mathcal{A}_P \in P\colon \Pr\left(Exp^{\text{Avail}}_{\mathcal{S},\mathcal{A}_P}(\lambda) \textit{ succeeds }\right) \leq \epsilon(\lambda).$$

5.4. Soundness

This notion is linked to impersonation attacks. In the hospital example, an adversary may want to insert a fake wristband that would be accepted as a legitimate IoT device of the system, in order to be treated free of charge by the hospital. The soundness experiment is described in Figure 4.

Experiment $Exp^{\text{Sound}}_{\mathcal{S},\mathcal{A}_P}(\lambda)$

1. $\mathcal{C}$ runs INITSYSTEM(1^λ).
2. $\mathcal{A}_P$ interacts with the system $\mathcal{S}$, limited by her class P.

$Exp^{\text{Sound}}_{\mathcal{S},\mathcal{A}_P}(\lambda)$ succeeds if $\mathcal{A}_P$ is considered as a legitimate IoT device of $\mathcal{S}$.

Figure 4. Soundness experiment.

The soundness property is thus as follows.

Definition 5. *Soundness: An IoT system $\mathcal{S}$ is said to be P-sound if:*

$$\forall \mathcal{A}_P \in P\colon \Pr\left(Exp^{\text{Sound}}_{\mathcal{S},\mathcal{A}_P}(\lambda) \textit{ succeeds }\right) \leq \epsilon(\lambda).$$

6. Formalising Security Attacks

The model allows a precise evaluation of the security level of IoT systems. In order to highlight this fact, we use it to analyse the four security properties of an IoT smart home system based on smart light bulbs. This technology consists of a number of light bulbs (i.e., IoT sensors), a dedicated hub to which the sensors connect to, a backend service that can communicate with the hub over the Internet, and a mobile application that enables the user to control the lights both from a local network or via the Internet.

Let us consider the case in which a user would like to control the smart home's light bulbs using his smartphone, while he is either outside or inside the home. To do so, he must firstly successfully complete the PAIRING protocol (as illustrated in Figure 5) between his smartphone and the hub in order to have remote access to the related sensors.

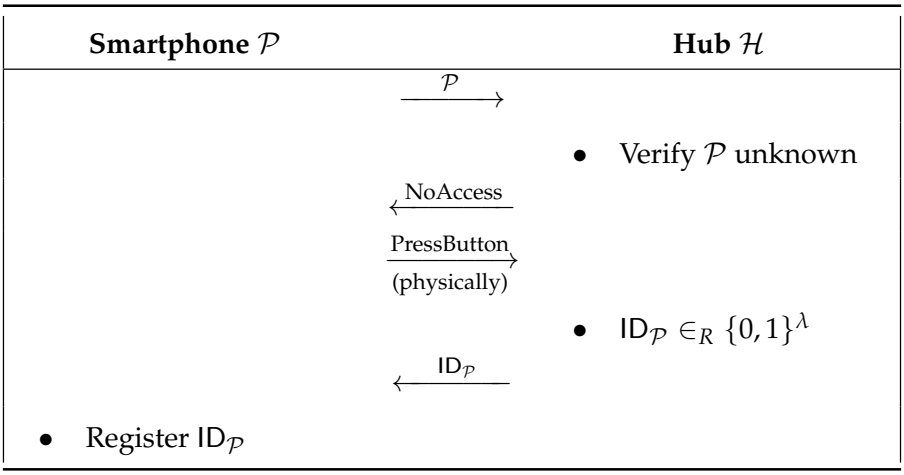

Figure 5. PAIRING protocol (smart light bulbs system).

Once his smartphone is paired, the user can launch the corresponding mobile application and send commands to the light bulbs either from the inside through the hub (as depicted in Figure 6), or from the outside via the cloud service that acts as a relay between the smartphone and the hub. This scenario is exploited what follows to analyse the security level of this smart home system $\mathcal{S}$.

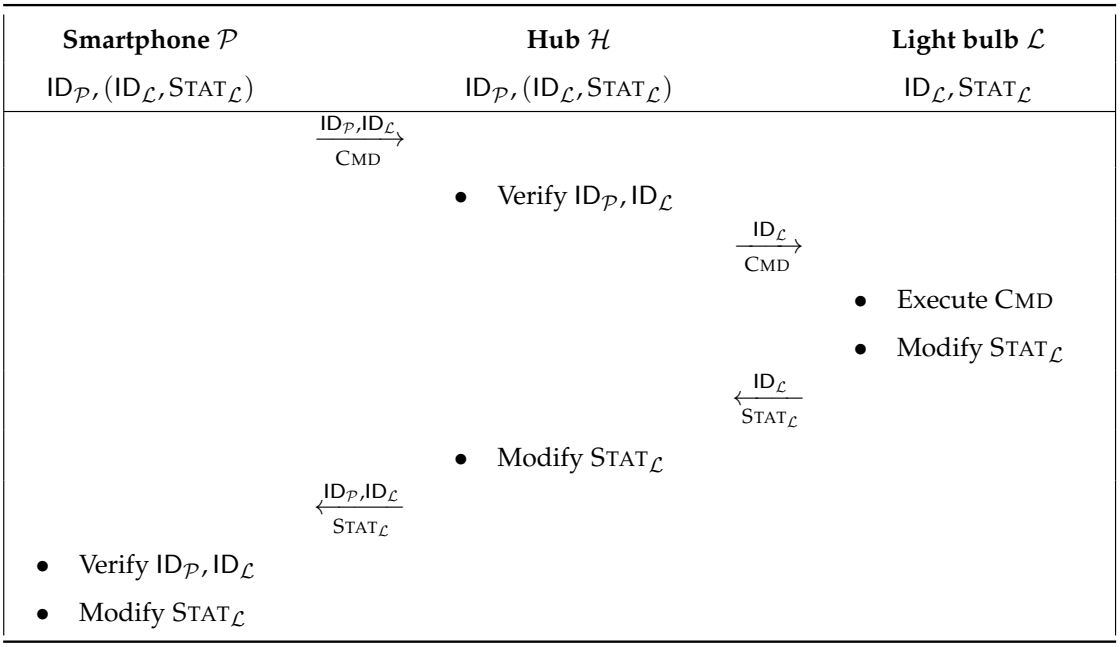

Figure 6. SEND_CMD protocol from inside (smart light bulbs system).

6.1. Confidentiality

An adversary can try to retrieve some confidential information about $\mathcal{S}$, e.g., the unique identifier $\mathsf{ID}_{\mathcal{P}}$ that is attributed by the hub to the smartphone during the PAIRING protocol. To do so, she only needs to eavesdrop the PAIRING protocol (Figure 5) to retrieve this confidential information, since this identifier is sent in cleartext from the hub to the smartphone. To formalise this attack, let us consider the EAVESDROPPER class. The attack performed by $\mathcal{A}_{\mathsf{EAVESDROPPER}}$ on $\mathcal{S}$ during $Exp^{\text{Confident}}_{\mathcal{S},\mathcal{A}_{\mathsf{EAVESDROPPER}}}(\lambda)$ is as follows.

1. The challenger $\mathcal{C}$ runs INITSYSTEM(1^{λ}).
2. $\mathcal{C}$ defines the set of confidential information CI={ID of all the devices of $\mathcal{S}$}.
3. $\mathcal{A}_{\mathsf{EAVESDROPPER}}$ interacts with $\mathcal{S}$ with the following steps.

- She calls:
  - $2 \times \mathcal{O}^{\text{CreateDevice}}()$

  and obtains the labels $\mathcal{H}$ (for the hub) and $\mathcal{P}$ (for the user's smartphone).
- To register the two devices within $\mathcal{S}$, she calls:
  - $\mathcal{O}^{\text{RegisterDevice}}(\mathcal{H})$;
  - $\mathcal{O}^{\text{RegisterDevice}}(\mathcal{P})$.
- To eavesdrop a PAIRING protocol execution between $\mathcal{H}$ and $\mathcal{P}$, she calls:
  - $\mathcal{O}^{\text{Execute}}(\text{PAIRING}, \mathcal{H}, \mathcal{P})$.
- She thus collects the transcripts:
  - $\pi^1_{\mathcal{H},\text{PAIRING}}$;
  - $\pi^1_{\mathcal{P},\text{PAIRING}}$.
- She retrieves the value of $\mathsf{ID}_\mathcal{P}$ with the call:
  - $\pi^1_{\mathcal{P},\text{PAIRING}}.\mathsf{msg} = (\mathcal{P}, \text{NoAccess}, \mathsf{ID}_\mathcal{P})$.

4. $\mathcal{A}_{\mathsf{EAVESDROPPER}}$ outputs the information I=($\mathsf{ID}_\mathcal{P}$).

Consequently, this $Exp^{\text{Confident}}_{\mathcal{S},\mathcal{A}_{\mathsf{EAVESDROPPER}}}(\lambda)$ succeeds as I is part of CI. $\mathcal{S}$ does not provide EAVESDROPPER-confidentiality when CI={ID of all the devices of $\mathcal{S}$}.

6.2. Integrity

An adversary can try to modify the status of a light bulb without the knowledge of the user, and thus desynchronise this information in $\mathcal{S}$. To do so, she must be active and perform a man-in-the-middle attack to prevent the command message to be received by the light bulb. To formalise this attack, let us consider the INSIDER class. The attack performed by $\mathcal{A}_{\mathsf{INSIDER}}$ on $\mathcal{S}$ during $Exp^{\text{Integrity}}_{\mathcal{S},\mathcal{A}_{\mathsf{INSIDER}}}(\lambda)$ is as follows.

1. The challenger $\mathcal{C}$ runs $\text{INITSYSTEM}(1^\lambda)$.
2. $\mathcal{C}$ defines the set of critical information CRI={status STAT of all the light bulbs of $\mathcal{S}$}.
3. $\mathcal{A}_{\mathsf{INSIDER}}$ interacts with $\mathcal{S}$ with the following steps.

- As the previous attack, she calls:
  - $2 \times \mathcal{O}^{\text{CreateDevice}}()$;
  - $\mathcal{O}^{\text{RegisterDevice}}(\mathcal{H})$;
  - $\mathcal{O}^{\text{RegisterDevice}}(\mathcal{P})$;
  - $\mathcal{O}^{\text{Execute}}(\text{PAIRING}, \mathcal{H}, \mathcal{P})$.
- To start a new SEND_CMD protocol execution with $\mathcal{P}$ for turning ON a light bulb $\mathcal{L}$, she calls:
  - $\mathcal{O}^{\text{Launch}}(\text{SEND\_CMD}, \mathcal{P})$.
- She thus collects the transcript:
  - $\pi^1_{\mathcal{P},\text{SEND\_CMD}}$.
- She retrieves the values sent by $\mathcal{P}$ with the call:
  - $\pi^1_{\mathcal{P},\text{SEND\_CMD}}.\mathsf{msg} = (\mathsf{ID}_\mathcal{P}, \mathsf{ID}_\mathcal{L}, \text{CMD})$, where $\text{CMD} = \{\text{light ON}\}$.
- She forwards this message to $\mathcal{H}$ with the call:
  - $\mathcal{O}^{\text{Send}}(\text{SEND\_CMD}, \mathcal{H}, (\mathsf{ID}_\mathcal{P}, \mathsf{ID}_\mathcal{L}, \{\text{light ON}\}))$.

  $\mathcal{H}$ forwards the command to the light bulb $\mathcal{L}$.
- At that moment, $\mathcal{A}_{\mathsf{INSIDER}}$ "intercepts" the message: in the model, this action is represented by the fact that $\mathcal{A}_{\mathsf{INSIDER}}$ does not send any message to the light bulb $\mathcal{L}$, which thus remains OFF.
- But, to make $\mathcal{H}$ believe that $\mathcal{L}$ received its message and answered positively to it, she calls:
  - $\mathcal{O}^{\text{Send}}(\text{SEND\_CMD}, \mathcal{H}, (\mathsf{ID}_\mathcal{L}, \text{STAT}_\mathcal{L}))$ with $\text{STAT}_\mathcal{L} = \{\text{ON}\}$.

  $\mathcal{H}$ thus records that $\text{STAT}_\mathcal{L} = \{\text{ON}\}$, even though it is not true on $\mathcal{L}$'s side.
- Finally, to make also $\mathcal{P}$ believe that everything went well and that $\mathcal{L}$ is turned ON, she calls:

– $\mathcal{O}^{\text{SEND}}(\text{SEND\_CMD}, \mathcal{P}, (\text{ID}_{\mathcal{P}}, \text{ID}_{\mathcal{L}}, \text{STAT}_{\mathcal{L}}))$.

$\mathcal{P}$ also records that $\text{STAT}_{\mathcal{L}} = \{\text{ON}\}$.

Consequently, this $Exp^{\text{Integrity}}_{\mathcal{S},\mathcal{A}_{\text{INSIDER}}}(\lambda)$ succeeds since the status $\text{STAT}_{\mathcal{L}}$ of the light bulb $\mathcal{L}$, i.e., part of the CRI={STAT of all the light bulbs of $\mathcal{S}$}, has been modified by $\mathcal{A}_{\text{INSIDER}}$. $\mathcal{S}$ does not provide INSIDER-integrity.

6.3. Availability

An adversary can try to perform a Denial of Service (DoS) attack in order to make the system unavailable anymore. She can thus use different techniques that might cause a DoS on the system by targeting any of its components (i.e., the hub, the sensors, or the cloud service). Here, we focus on the hub's robustness to deal with low rate DoS. To do so, the adversary simply sends numerous requests to the hub. To formalise this attack, let us again consider the INSIDER class. The attack performed by $\mathcal{A}_{\text{INSIDER}}$ on $\mathcal{S}$ during $Exp^{\text{Avail}}_{\mathcal{S},\mathcal{A}_{\text{INSIDER}}}(\lambda)$ is as follows.

1. The challenger $\mathcal{C}$ runs $\text{INITSYSTEM}(1^{\lambda})$.
2. $\mathcal{A}_{\text{INSIDER}}$ interacts with $\mathcal{S}$ with the following steps.
   - As the first attack, she calls:
     – $2 \times \mathcal{O}^{\text{CREATEDEVICE}}()$;
     – $\mathcal{O}^{\text{REGISTERDEVICE}}(\mathcal{H})$;
     – $\mathcal{O}^{\text{REGISTERDEVICE}}(\mathcal{P})$;
     – $\mathcal{O}^{\text{EXECUTE}}(\text{PAIRING}, \mathcal{H}, \mathcal{P})$.
   - To start a new SEND_CMD protocol execution with the hub $\mathcal{H}$ and directly send the first message to it, she calls a certain amount X of times:
     – $\mathcal{O}^{\text{SEND}}(\text{SEND\_CMD}, \mathcal{H}, (rnd, \text{ID}_{\mathcal{L}}, \text{CMD}))$, where *rnd* is a random value replacing $\text{ID}_{\mathcal{P}}$.

     Of course, these messages are not accepted by the hub $\mathcal{H}$.
3. $\mathcal{A}_{\text{INSIDER}}$ designates the hub $\mathcal{H}$ and calls:
   – $\mathcal{O}^{\text{EXECUTE}}(\text{SEND\_CMD}, \mathcal{P}, \mathcal{H}, \mathcal{L})$.

   The hub $\mathcal{H}$ is unable to perform correctly the protocol execution.

Consequently, this $Exp^{\text{Avail}}_{\mathcal{S},\mathcal{A}_{\text{INSIDER}}}(\lambda)$ succeeds as $\mathcal{A}_{\text{INSIDER}}$ has been able to prevent the hub to work properly again within the system. Note that the same attack could have been formalised with an EXTERNAL adversary. Thus, $\mathcal{S}$ does not provide neither INSIDER-availability nor EXTERNAL-availability.

6.4. Soundness

An adversary can try to impersonate a legitimate user in order to send illegitimate/undesired commands to the light bulbs. To do so, she simply performs a replay attack: she reuses a previous command eavesdropped between the legitimate user's smartphone and the hub. To formalise this attack, let us again consider the INSIDER class. The attack performed by $\mathcal{A}_{\text{INSIDER}}$ on $\mathcal{S}$ during $Exp^{\text{Sound}}_{\mathcal{S},\mathcal{A}_{\text{INSIDER}}}(\lambda)$ is as follows.

1. The challenger $\mathcal{C}$ runs $\text{INITSYSTEM}(1^{\lambda})$.
2. $\mathcal{A}_{\text{INSIDER}}$ interacts with $\mathcal{S}$ with the following steps.
   - As the first attack, she calls:
     – $2 \times \mathcal{O}^{\text{CREATEDEVICE}}()$;
     – $\mathcal{O}^{\text{REGISTERDEVICE}}(\mathcal{H})$;
     – $\mathcal{O}^{\text{REGISTERDEVICE}}(\mathcal{P})$;
     – $\mathcal{O}^{\text{EXECUTE}}(\text{PAIRING}, \mathcal{H}, \mathcal{P})$.

- To eavesdrop a SEND_CMD protocol execution between $\mathcal{P}$, $\mathcal{H}$ and $\mathcal{L}$, she calls:
  - $\mathcal{O}^{\text{EXECUTE}}(\text{SEND\_CMD}, \mathcal{P}, \mathcal{H}, \mathcal{L})$.
- She thus collects the transcripts:
  - $\pi^1_{\mathcal{P},\text{SEND\_CMD}}$;
  - $\pi^1_{\mathcal{H},\text{SEND\_CMD}}$;
  - $\pi^1_{\mathcal{L},\text{SEND\_CMD}}$.
- She retrieves all the values send by the legitimate smartphone $\mathcal{P}$ with the call:
  - $\pi^1_{\mathcal{P},\text{SEND\_CMD}}.\mathsf{msg} = (\mathsf{ID}_{\mathcal{P}}, \mathsf{ID}_{\mathcal{L}}, \text{CMD}, \text{STAT}_{\mathcal{L}})$.
- To start a new SEND_CMD protocol execution with $\mathcal{H}$ and directly send the first message to it, she calls:
  - $\mathcal{O}^{\text{SEND}}(\text{SEND\_CMD}, \mathcal{H}, (\mathsf{ID}_{\mathcal{P}}, \mathsf{ID}_{\mathcal{L}}, \text{CMD}))$.

  $\mathcal{H}$ will accept this message as a legitimate one, and thus will forward to the bulb $\mathcal{L}$ the command sent by $\mathcal{A}_{\mathsf{INSIDER}}$.

Consequently, this $Exp^{\text{Sound}}_{\mathcal{S},\mathcal{A}_{\mathsf{INSIDER}}}(\lambda)$ succeeds as $\mathcal{A}_{\mathsf{INSIDER}}$ has been considered as a legitimate IoT device (namely a smartphone) by the hub $\mathcal{H}$. $\mathcal{S}$ does not provide INSIDER-soundness.

Though this type of attacks assumes that the adversary acts as an INSIDER of $\mathcal{S}$'s network, it might also be performed by eavesdropping similar requests between the smartphone and the cloud service. This might consequently introduce a vulnerability in $\mathcal{S}$'s architecture if the adversary can reach $\mathcal{S}$'s router from the outside world.

7. Formalising Security Proof

In this section, we analyse the confidentiality level of a smart thermostat system very similar to the one with smart light bulbs. It also has a PAIRING and a SEND_CMD protocols, as depicted in Figures 7 and 8. The main difference is that all the devices of the system are able to compute a PRF (Pseudo-Random Function) in order to exchange encrypted data.

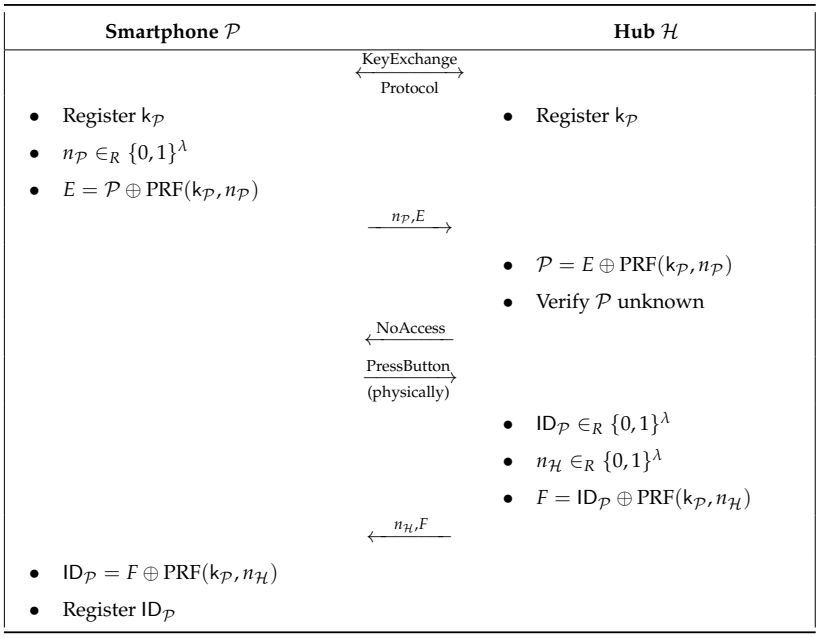

Figure 7. PAIRING protocol (smart thermostat system).

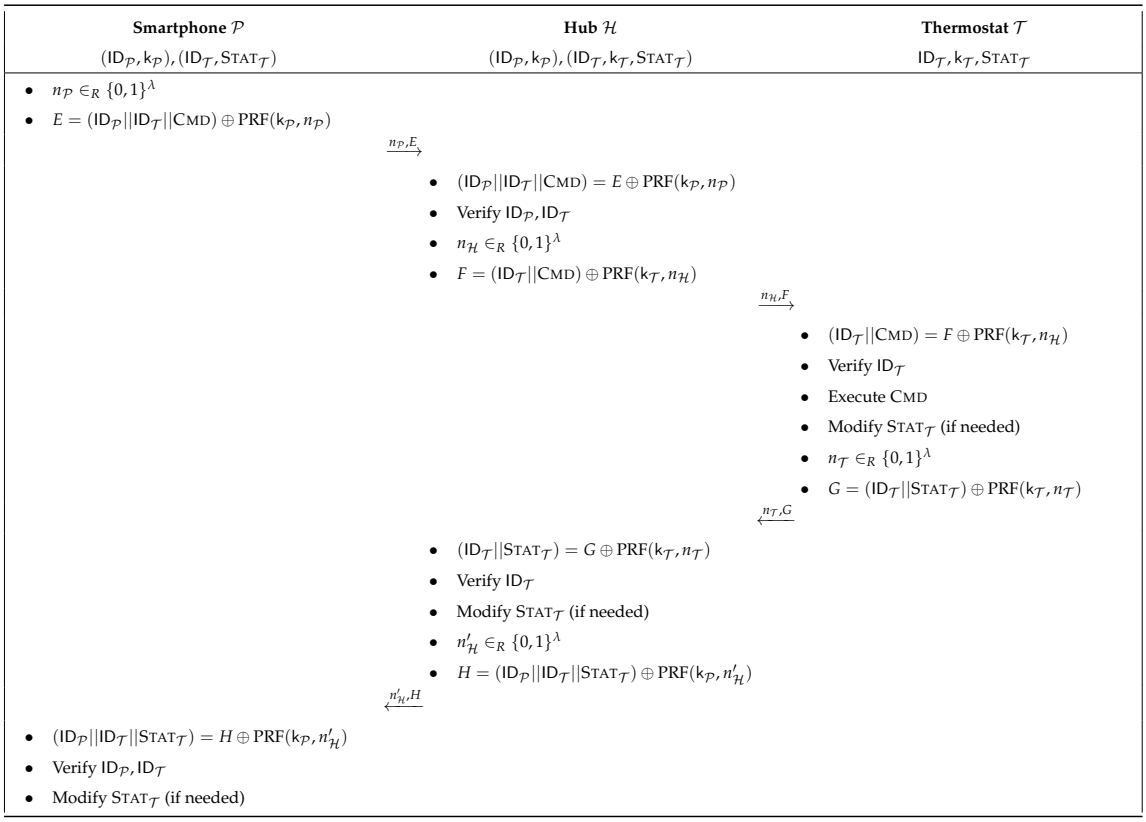

Figure 8. SEND_CMD protocol from inside (smart thermostat system).

Theorem 1. *The smart thermostat system $\mathcal{S}$ is INSIDER-confidential, for CI = {ID of all the devices of $\mathcal{S}$}.*

Proof. We use the game technique as described by Shoup in [16] (only modifications of the original experiment are specified in the games: unspecified queries stay unchanged) to prove the confidentiality of the smart thermostat system $\mathcal{S}$ against INSIDER adversaries. First, we assume that the key exchange protocol used at the beginning of the PAIRING protocol is secure against man-in-the-middle attacks (for instance, this is not the case for the original well-known Diffie-Hellman protocol).

Game 0: This game corresponds to the confidentiality experiment $Exp^{\text{Confident}}_{\mathcal{S},\mathcal{A}_{\mathsf{INSIDER}}}(\lambda)$ performed by the adversary $\mathcal{A}_{\mathsf{INSIDER}}$ on $\mathcal{S}$ of security parameter λ. Let CI={ID of all the devices of $\mathcal{S}$}. Let S_0 denote the event "I is (part of) CI" in Game 0. Thus, we have that $\Pr(S_0) = \Pr(Exp^{\text{Confident}}_{\mathcal{S},\mathcal{A}_{\mathsf{INSIDER}}}(\lambda)$ succeeds).

Let us assume that $\mathcal{A}_{\mathsf{INSIDER}}$ performs q $\mathcal{O}^{\text{SEND}}$ queries to the hub during the PAIRING protocol. These $\mathcal{O}^{\text{SEND}}$ queries are called "encryption queries" in the sequel of the proof. From Game 0, we introduce q transitions games as follows.

Game i: This is the same game as Game $(i-1)$ except that one additional encryption query has been replaced as defined below:

- the i^{th} first encryption queries are such that $F = \mathsf{ID}_\mathcal{P} \oplus \mathrm{PRF}(n, n_\mathcal{H})$, where $n \in_R \{0,1\}^{\lambda_\mathsf{k}}$ and λ_k is the output size of the key $\mathsf{k}_\mathcal{P}$, and thus F is the result of the fixed value $\mathsf{ID}_\mathcal{P}$ XORed with a pure PRF (i.e., used only with random values);
- the $(q-i)$ next encryption queries are such that $F = \mathsf{ID}_\mathcal{P} \oplus \mathrm{PRF}(\mathsf{k}_\mathcal{P}, n_\mathcal{H})$, where F is the result of the fixed value $\mathsf{ID}_\mathcal{P}$ XORed with a keyed PRF (i.e., used with the secret key $\mathsf{k}_\mathcal{P}$).

Let S_i denote the event "I is (part of) CI" in Game i, and $\Pr(S_i)$ denote its success probability.

Game $(i-1)$ to Game i (for $1 \leq i \leq q$) only differ from one encryption query. As stated by Shoup in [16], the probability to distinguish a keyed PRF from a pure PRF is called the PRF advantage $\mathrm{Adv}_{\mathcal{S}}^{\mathrm{PRF}}$ and is negligible. This implies:

$$\forall i \text{ s.t. } 1 \leq i \leq q : |\Pr(S_{i-1}) - \Pr(S_i)| = \mathrm{Adv}_{\mathcal{S}}^{\mathrm{PRF}} \leq \epsilon(\lambda).$$

At the end of these q steps, the following game is obtained.

Game q: This is the game where all the q encryption queries during the PAIRING protocol have been replaced by $F = \mathsf{ID}_{\mathcal{P}} \oplus \mathrm{PRF}(n, n_{\mathcal{H}})$,. Let S_q denote the event "I is (part of) CI" in Game q, and $\Pr(S_q)$ denote its success probability.

From Game q, the same reasoning can be applied for the SEND_CMD protocol. Let thus assume that $\mathcal{A}_{\mathsf{INSIDER}}$ performs q' encryption queries to the devices of $\mathcal{S}$. This also implies:

$$\forall j \text{ s.t. } 1 \leq j \leq q' : |\Pr(S_{q+j-1}) - \Pr(S_{q+j})| = \mathrm{Adv}_{\mathcal{S}}^{\mathrm{PRF}} \leq \epsilon(\lambda).$$

Game $(q+q')$: This is the final game of this proof, where all the q' encryption queries during the SEND_CMD protocol have been replaced by a pure PRF. Let S_{final} denote the event "I is (part of) CI" in Game $(q+q')$, and $\Pr(S_{\text{final}})$ denote its success probability. Since $\mathcal{A}_{\mathsf{INSIDER}}$ is unable to recover any ID from a pure PRF, she can only try to output a random value I: therefore $\Pr(S_{\text{final}}) \leq \epsilon(\lambda)$.

From all the transitions, the conclusion is that:

$$\begin{aligned}
\Pr(Exp^{\text{Confident}}_{\mathcal{S},\mathcal{A}_{\mathsf{INSIDER}}}(\lambda) \text{ succeeds}) &= \Pr(S_0) - \Pr(S_1) + \Pr(S_1) - \ldots - \Pr(S_q) \\
&\quad + \Pr(S_q) - \ldots - \Pr(S_{\text{final}}) + \Pr(S_{\text{final}}) \\
&\leq \Pr(S_{\text{final}}) + \sum_{i=0}^{q-1} \Big| \Pr(S_i) - \Pr(S_{i+1}) \Big| \\
&\quad + \sum_{j=0}^{q'-1} \Big| \Pr(S_{q+j}) - \Pr(S_{q+j+1}) \Big| \\
&\leq \Pr(S_{\text{final}}) + (q+q').\mathrm{Adv}_{\mathcal{S}}^{\mathrm{PRF}} \leq \epsilon(\lambda).
\end{aligned}$$

□

Consequently, this $Exp^{\text{Confident}}_{\mathcal{S},\mathcal{A}_{\mathsf{INSIDER}}}(\lambda)$ succeeds with negligible probability. This proves Theorem 1: the smart thermostat system $\mathcal{S}$ described in this section provides INSIDER-confidentiality when $\mathrm{CI} = \{\mathsf{ID} \text{ of all the devices of } \mathcal{S}\}$.

8. Conclusions and Future Research

In this paper, we proposed a new model able to analyse the security properties of IoT systems. The model is composed of (1) the formalisation of IoT systems; (2) the definition of the potential adversary classes based on oracles, selectors, and restrictions that are inspired from real-life examples of IoT attacks; and (3) the definition of four well-established security properties (confidentiality, integrity, availability, soundness).

The purpose of our model is twofold: it can be used either to accurately describe attacks against IoT systems, or to formally prove the security level guaranteed by IoT systems. Formal proofs are a tremendous support to identify the critical elements of an IoT system. In particular, they help in highlighting the vulnerabilities and weakest links, and thus assist the IoT designers in building better and more secure systems. They can be used during the design phase in order to set the security targets of the implementation. Then, when the security tests are performed, the results can be verified against

the initial claims and, depending on the outcome, adjustments may be performed in the initial design of the system. Since formal proofs guarantee a certain security level, they form a universal security evaluation able to compare fairly different IoT systems. Our model could consequently also assist in building an IoT certification framework.

Inheriting the positive specificities of the RFID privacy model [9], our model also provides extensibility and granularity in the analyses. The extensibility is brought with the possibility to add new features to the model (i.e., new oracles, selectors, restrictions), and the granularity is given with the practicability to define as many potential adversaries as possible.

Nonetheless, such features of extensibility and granularity might bring some limitations and intricacy in the security analysis. First, our model is designed to allow the definition of any adversary class. This can add some difficulties in ordering the strength of the adversaries, thus rendering troublesome to compare the security level of different IoT systems. Secondly, in order to entitle an irrefutable security level to an IoT system, our model should be used to provide one formal proof for a given adversary class P and one attack for the consecutive stronger adversary class. This should prove that the analysed IoT system ensures the given P security level (e.g., P-confidentiality). In practice, this can be complex to put in place since the extensibility and granularity of our model allows the finding and introduction of new adversary classes that might be inserted between supposedly consecutive classes.

In the near future, we are planning to extend our model to support access control mechanisms formalisation and proofs in IoT systems, covering secure data access requirements as already identified in other related works [14].

Author Contributions: Conceptualization, T.M., D.G., I.K., S.K.; methodology, T.M., D.G., I.K., S.K.; formal analysis, T.M. and S.K.; writing–review and editing, T.M., D.G., I.K., S.K.; supervision, I.N.F.; project administration, I.N.F. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

References

1. Ronen, E.; Shamir, A.; Weingarten, A.O.; O'Flynn, C. IoT Goes Nuclear: Creating a ZigBee Chain Reaction. In Proceedings of the IEEE Symposium on Security and Privacy—SP 2017, San José, CA, USA, 22–26 May 2017; pp. 195–212.
2. Mohsin, M.; Anwar, Z.; Husari, G.; Al-Shaer, E.; Rahman, M.A. IoTSAT: A Formal Framework for Security Analysis of the Internet of Things (IoT). In Proceedings of the Conference on Communications and Network Security—CNS 2016, Philadelphia, PA, USA, 17–19 October 2016; pp. 180–188.
3. Armando, A.; Basin, D.A.; Boichut, Y.; Chevalier, Y.; Compagna, L.; Cuéllar, J.; Drielsma, P.H.; Héam, P.C.; Kouchnarenko, O.; Mantovani, J.; et al. The AVISPA Tool for the Automated Validation of Internet Security Protocols and Applications. In Proceedings of the International Conference on Computer Aided Verification—CAV 2005, Edinburgh, Scotland, UK, 6–10 July 2005; pp. 281–285.
4. Cremers, C.J.F. The Scyther Tool: Verification, Falsification, and Analysis of Security Protocols. In Proceedings of the International Conference on Computer Aided Verification—CAV 2008, Princeton, NJ, USA, 7–14 July 2008; pp. 414–418.
5. Jha, S.; Sheyner, O.; Wing, J. Two Formal Analyses of Attack Graphs. In Proceedings of the IEEE Computer Security Foundations Workshop—CSFW-15, Cape Breton, NS, Canada, 24–26 June 2002; pp. 49–63.
6. Mauw, S.; Oostdijk, M. Foundations of Attack Trees. In Proceedings of the 8th International Conference on Information Security and Cryptology—ICISC 2005, Seoul, Korea, 1–2 December 2005; Volume 3935, pp. 186–198.

7. Tabrizi, F.M.; Pattabiraman, K. Formal Security Analysis of Smart Embedded Systems. In Proceedings of the 32nd Annual Conference on Computer Security Applications—ACSAC 2016, Los Angeles, CA, USA, 5–9 December 2016; pp. 1–15.
8. Coisel, I.; Martin, T. Untangling RFID Privacy Models. *J. Comput. Networks Commun.* **2013**, *2013*, 710275. [CrossRef]
9. Avoine, G.; Coisel, I.; Martin, T. Untraceability Model for RFID. *IEEE Trans. Mob. Comput.* **2014**, *13*, 2397–2405. [CrossRef]
10. Kayes, A.S.M.; Han, J.; Colman, A.; Islam, M.S. RelBOSS: A Relationship-Aware Access Control Framework for Software Services. In *On The Move to Meaningful Internet Systems—OTM 2014*; Springer: Berlin/Heidelberg, Germany, 2014; pp. 258–276.
11. Kayes, A.S.M.; Rahayu, W.; Dillon, T.S.; Chang, E. Accessing Data from Multiple Sources Through Context-Aware Access Control. In Proceedings of the 17th IEEE International Conference On Trust, Security And Privacy in Computing and Communications/12th IEEE International Conference On Big Data Science And Engineering—TrustCom/BigDataSE, New York, NY, USA, 1–3 August 2018; pp. 551–559.
12. Kayes, A.S.M.; Rahayu, W.; Dillon, T.S. Critical Situation Management Utilizing IoT-Based Data Resources through Dynamic Contextual Role Modeling and Activation. *Computing* **2019**, *101*, 743–772. [CrossRef]
13. Tu, D.Q.; Kayes, A.S.M.; Rahayu, W.; Nguyen, K. ISDI: A New Window-Based Framework for Integrating IoT Streaming Data from Multiple Sources. In Proceedings of the Advanced Information Networking and Applications—AINA 2019, Matsue, Japan, 27–29 March 2019; Volume 926, pp. 498–511.
14. Kayes, A.S.M.; Kalaria, R.; Sarker, I.H.; Islam, M.S.; Watters, P.A.; Ng, A.; Hammoudeh, M.; Badsha, S.; Kumara, I. A Survey of Context-Aware Access Control Mechanisms for Cloud and Fog Networks: Taxonomy and Open Research Issues. *Sensors* **2020**, *20*, 9. [CrossRef] [PubMed]
15. UPnP Forum. *UPnP^{TM} Device Architecture 1.1*; Technical Report; UPnP: Beaverton, OR, USA, 2008.
16. Shoup, V. *Sequences of Games: A Tool for Taming Complexity in Security Proofs*; Cryptology ePrint Archive, Report 2004/332; IACR: Las Vegas, NV, USA, 2004.